Parkinson’s Disease Prognostic

Use protein and peptide data measurements from Parkinson's Disease patients to predict the progression of the disease

Corentin Lobet

Data

train_peptides.csv Mass spectrometry data at the peptide level. Peptides are the component subunits of proteins.

- `visit_id` - ID code for the visit.
- `visit_month` - The month of the visit, relative to the first visit by the patient.
- `patient_id` - An ID code for the patient.
- `UniProt` - The UniProt ID code for the associated protein. There are often several peptides per protein.
- `Peptide` - The sequence of amino acids included in the peptide. See this table for the relevant codes. Some rare annotations may not be included in the table.
- `PeptideAbundance` - The frequency of the amino acid in the sample.

train_proteins.csv Protein expression frequencies aggregated from the peptide level data.

- `visit_id`
- `visit_month`
- `patient_id`
- `UniProt`
- `NPX` - Normalized protein expression. The frequency of the protein's occurrence in the sample. May not have a 1:1 relationship with the component peptides as some proteins contain repeated copies of a given peptide.

train_clinical_data.csv

- `visit_id`
- `visit_month`
- `patient_id`
- `updrs_[1-4]` - The patient's UPDRS socres. Higher numbers indicate more severe symptoms. Each sub-section covers a distinct category of symptoms, such as mood and behavior for Part 1 and motor functions for Part 3.
- `upd23b_clinical_state_on_medication` - Whether or not the patient was taking medication such as Levodopa during the UPDRS assessment. Expected to mainly affect the scores for Part 3 (motor function).

Data

Clinical

	visit_id	patient_id	visit_month	updrs_1	updrs_2	updrs_3	updrs_4	upd23b_clinical_state_on_medication
0	55_0	55	0	10.0	6.0	15.0	NaN	NaN
1	55_3	55	3	10.0	7.0	25.0	NaN	NaN
2	55_6	55	6	8.0	10.0	34.0	NaN	NaN
3	55_9	55	9	8.0	9.0	30.0	0.0	On
4	55_12	55	12	10.0	10.0	41.0	0.0	On

: Found 248 unique patient_id values
: Found 17 unique visit_month values

Proteins

	visit_id	visit_month	patient_id	UniProt	NPX
0	55_0	0	55	O00391	11254.3
1	55_0	0	55	O00533	732430.0
2	55_0	0	55	O00584	39585.8
3	55_0	0	55	O14498	41526.9
4	55_0	0	55	O14773	31238.0

: Found 248 unique patient_id values
: Found 15 unique visit_month values
: Found 227 unique UniProt values

Peptides

	visit_id	visit_month	patient_id	UniProt	Peptide	PeptideAbundance
0	55_0	0	55	O00391	NEQEQPLGQWHLS	11254.3
1	55_0	0	55	O00533	GNPEPTFSWTK	102060.0
2	55_0	0	55	O00533	IEIPSSVQQVPTIIK	174185.0
3	55_0	0	55	O00533	KPQSAVYSTGSNGILLC(UniMod_4)EAEGEPQPTIK	27278.9
4	55_0	0	55	O00533	SMEQNGPGLEYR	30838.7

: Found 248 unique patient_id values
: Found 227 unique UniProt values
: Found 968 unique Peptide values

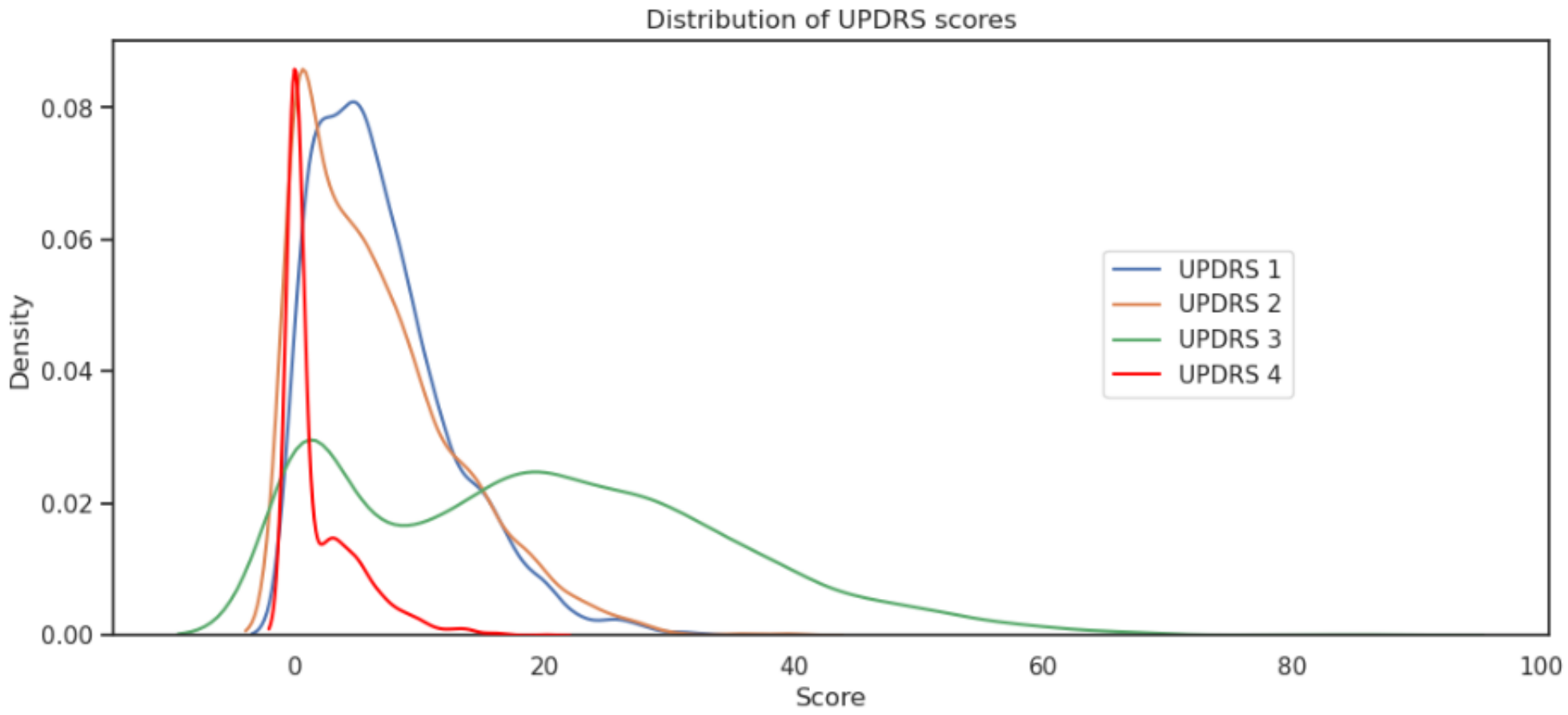
Exploratory Data Analysis

UPDRS Scores

- **UPDRS 1**: Non-Motor Experiences of Daily Living (0 - 52)
- **UPDRS 2**: Motor Experiences of Daily Living (0 - 52)
- **UPDRS 3**: Motor Examination (0 - 132)
- **UPDRS 4**: Motor Complications (0 - 24)

	count	mean	std	min	25%	50%	75%	max
updrs_1	2614.000000	7.110559	5.525955	0.000000	3.000000	6.000000	10.000000	33.000000
updrs_2	2613.000000	6.743590	6.323230	0.000000	1.000000	5.000000	10.000000	40.000000
updrs_3	2590.000000	19.421236	15.000289	0.000000	6.000000	19.000000	29.000000	86.000000
updrs_4	1577.000000	1.861763	3.022112	0.000000	0.000000	0.000000	3.000000	20.000000

UPDRS Scores



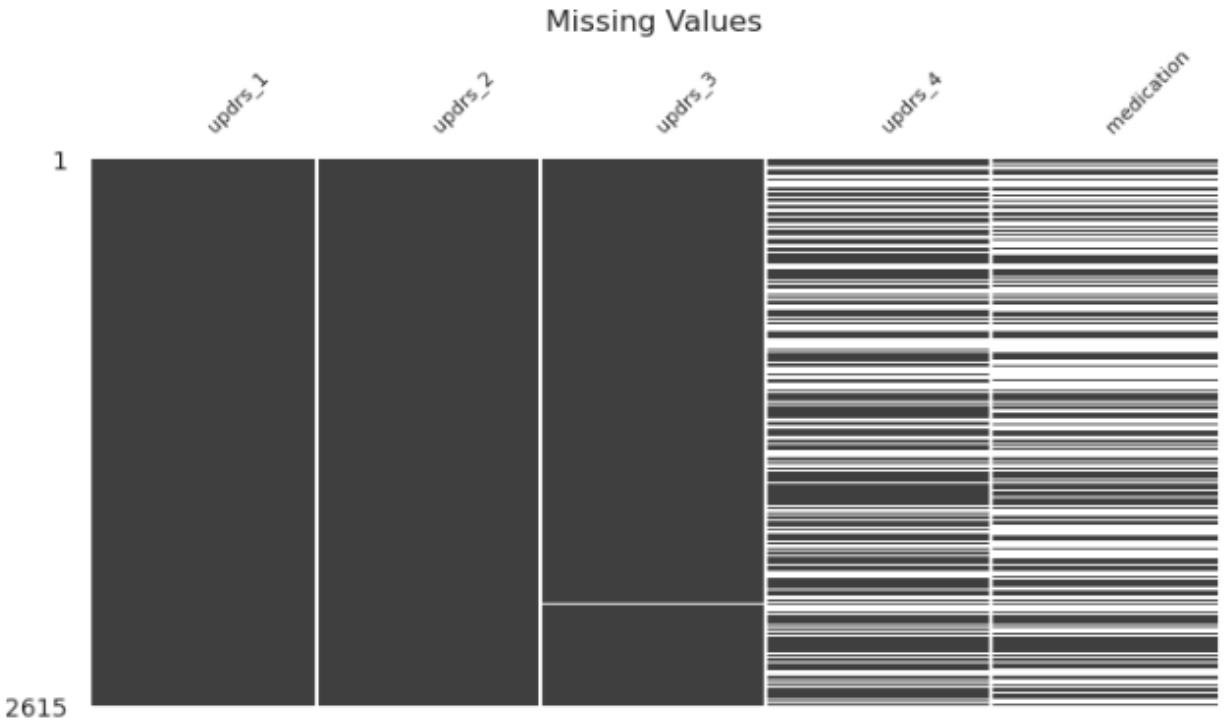
Missing Clinical Data

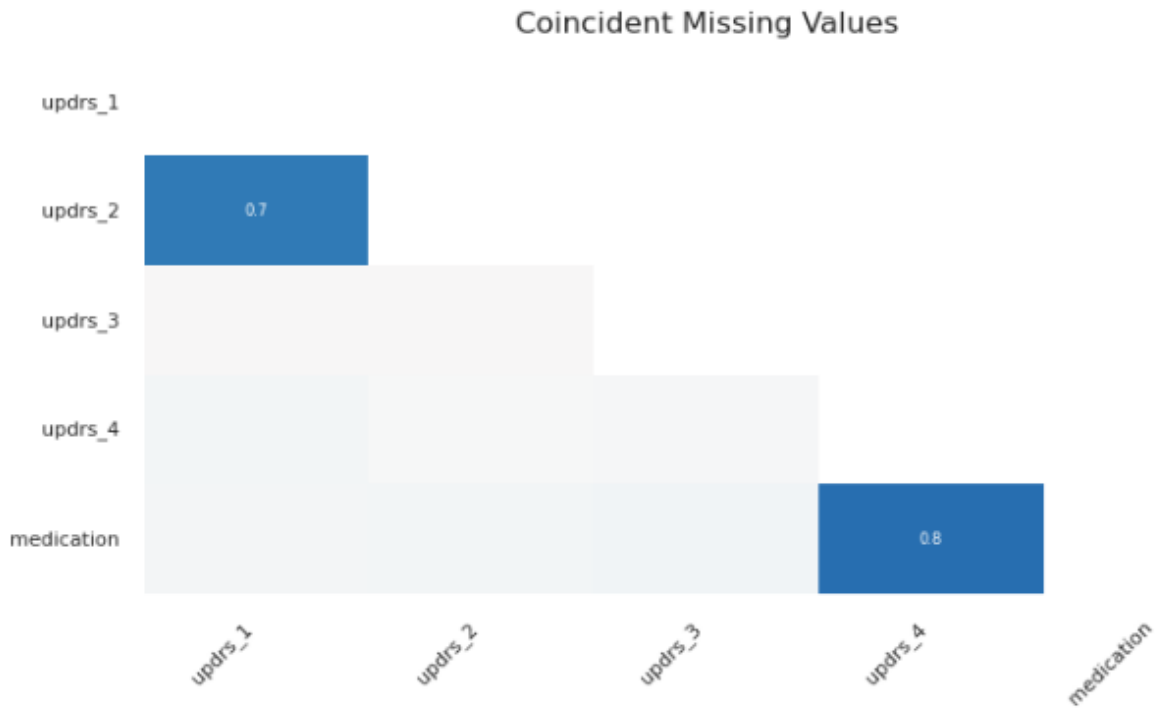
Proportions of Missing Values

	updrs_1	updrs_2	updrs_3	updrs_4	medication
Proportion NA	0.038241	0.076482	0.956023	39.694073	50.745698

Number of missing UPDRS scores per visit

mean	std	min	25%	50%	75%	max
0.407648	0.50227	0.0	0.0	0.0	1.0	3.0





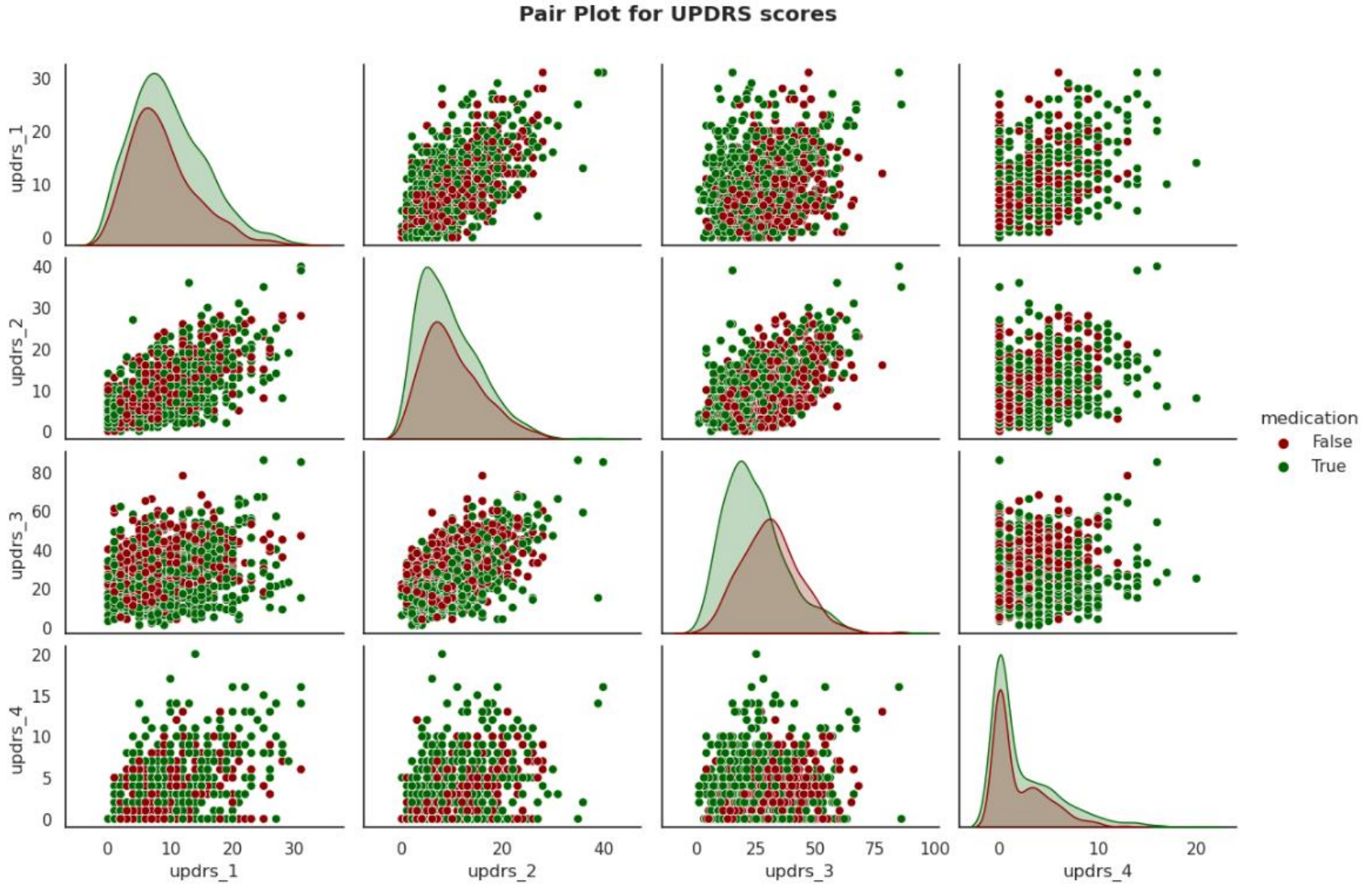
Role of Medication

Mean

medication	updrs_1	updrs_2	updrs_3	updrs_4
False	8.758285	10.037037	30.871345	1.986166
True	9.734194	9.563871	24.368970	2.450000

Median

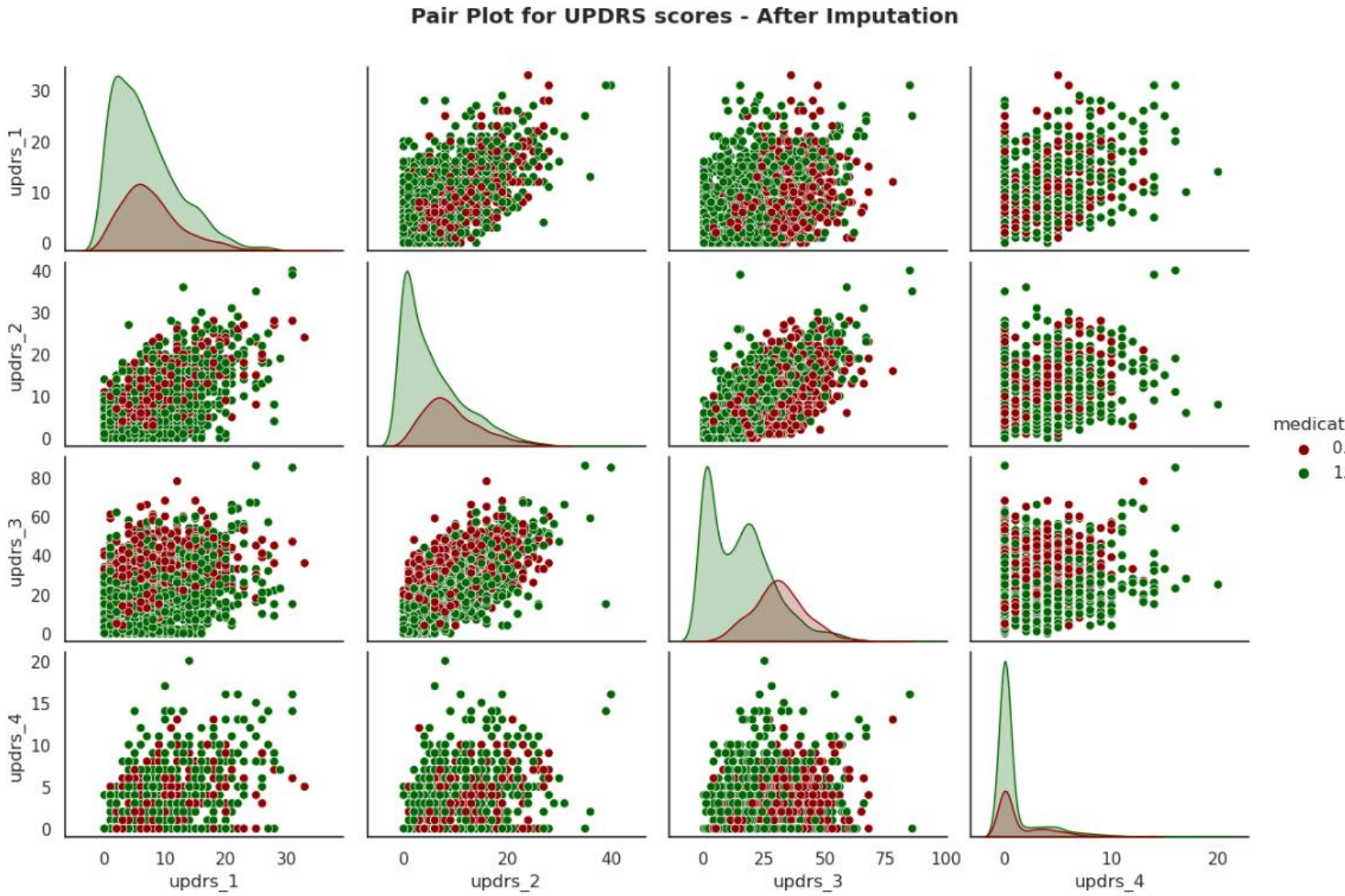
medication	updrs_1	updrs_2	updrs_3	updrs_4
False	8.000000	9.000000	30.000000	0.000000
True	9.000000	8.000000	22.000000	0.000000



Role of Medication

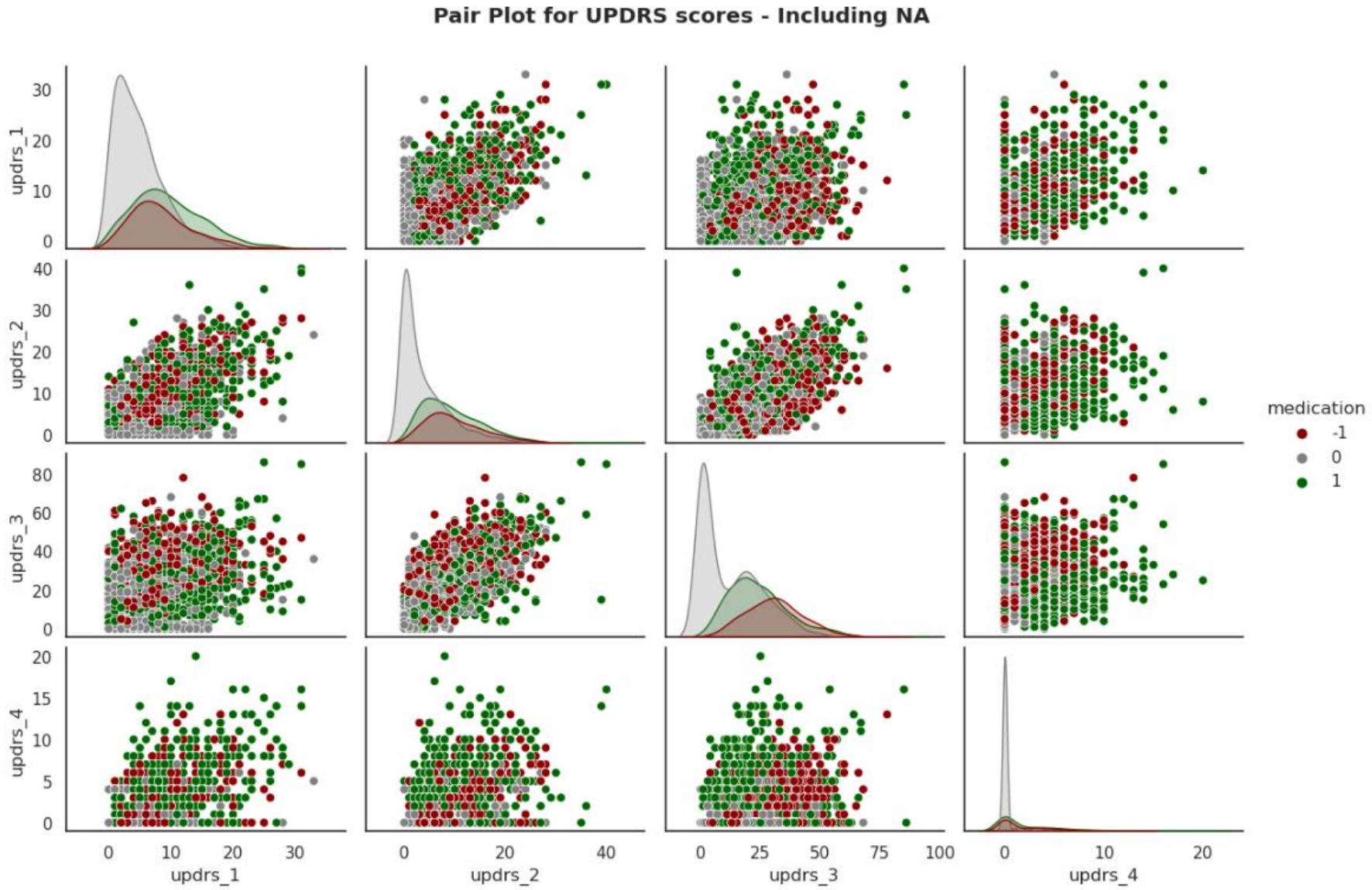
What if we impute the 50% missing values?

- Interpolate UPDRS by patient
- KNN Impute Medication
- 67% predictive accuracy from new to existing => Va bene

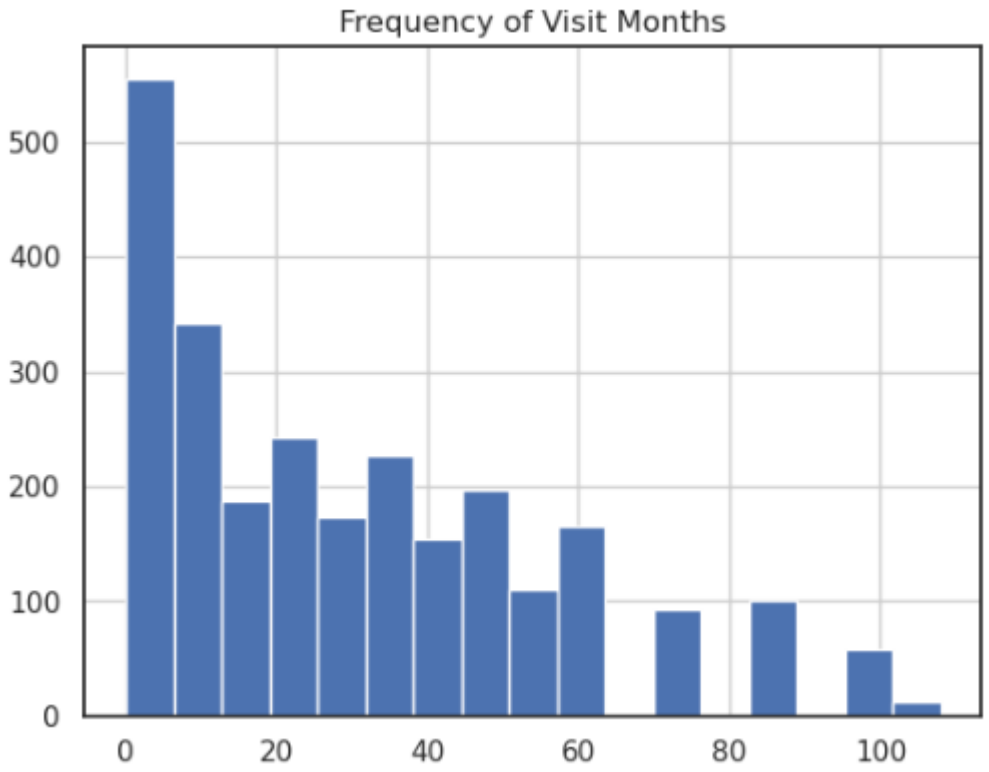


Role of Medication

Information in NAs?

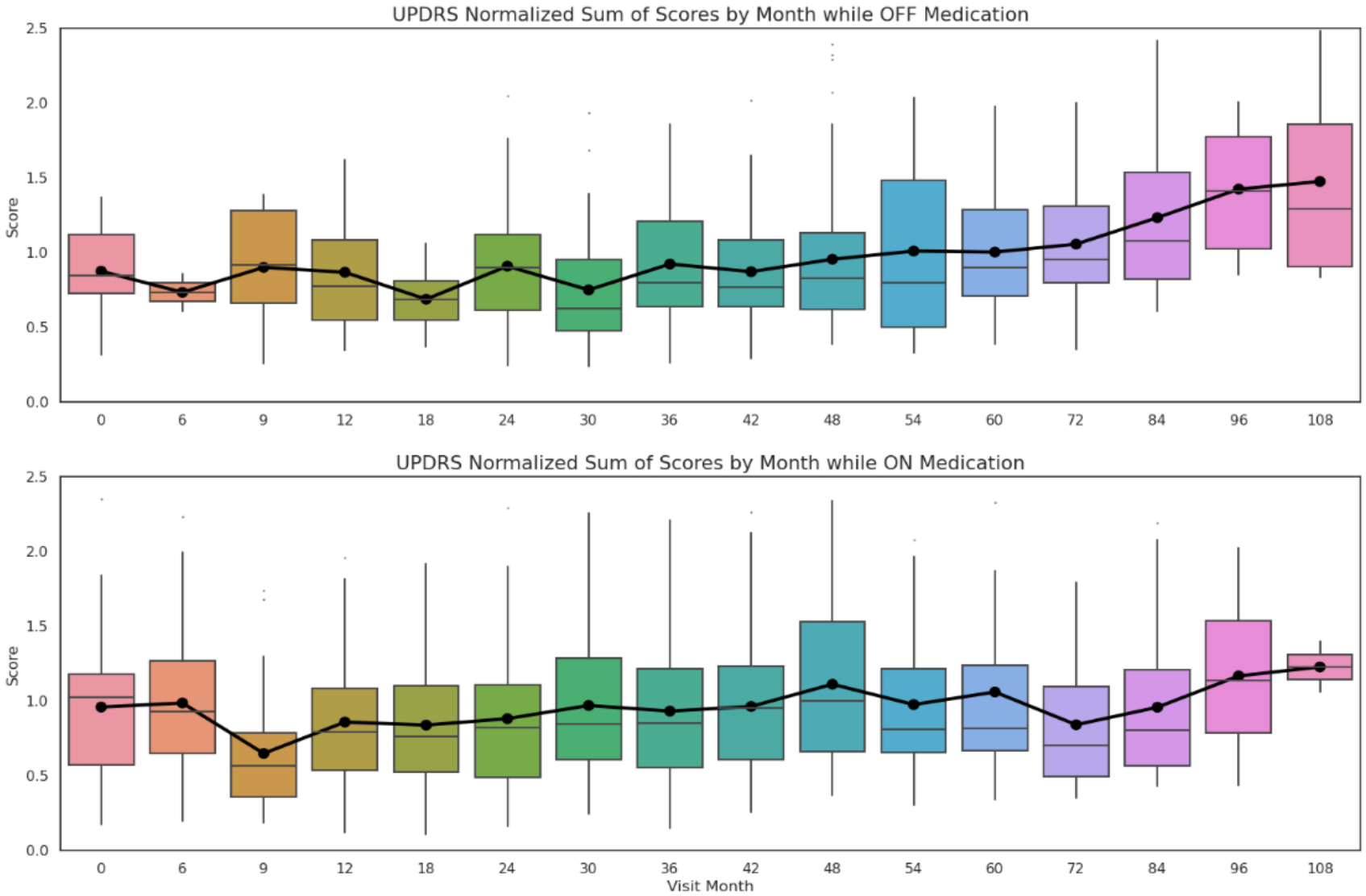


Evolution of Scores

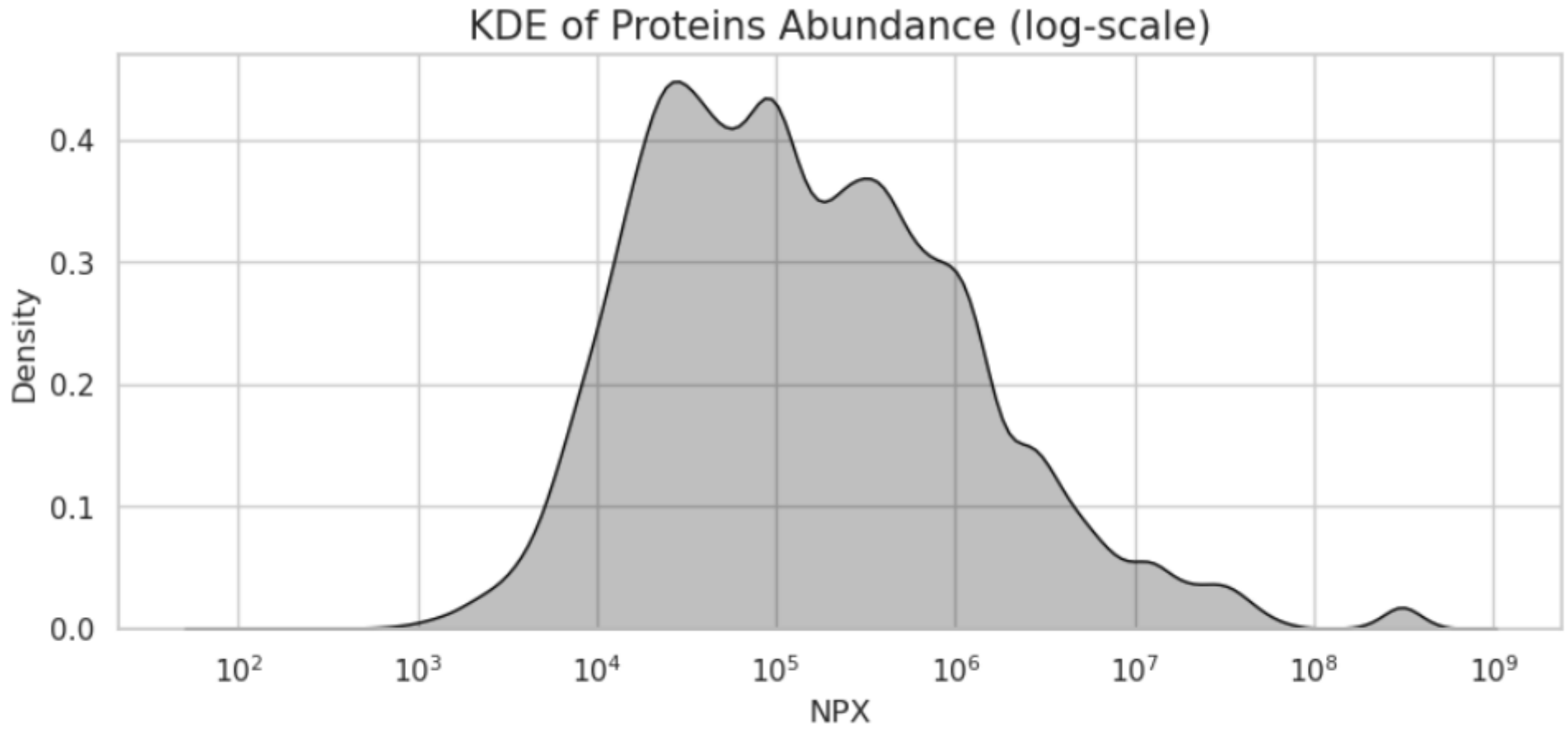


Proportion of patients starting at month 0
=> 100%

Evolution of Scores

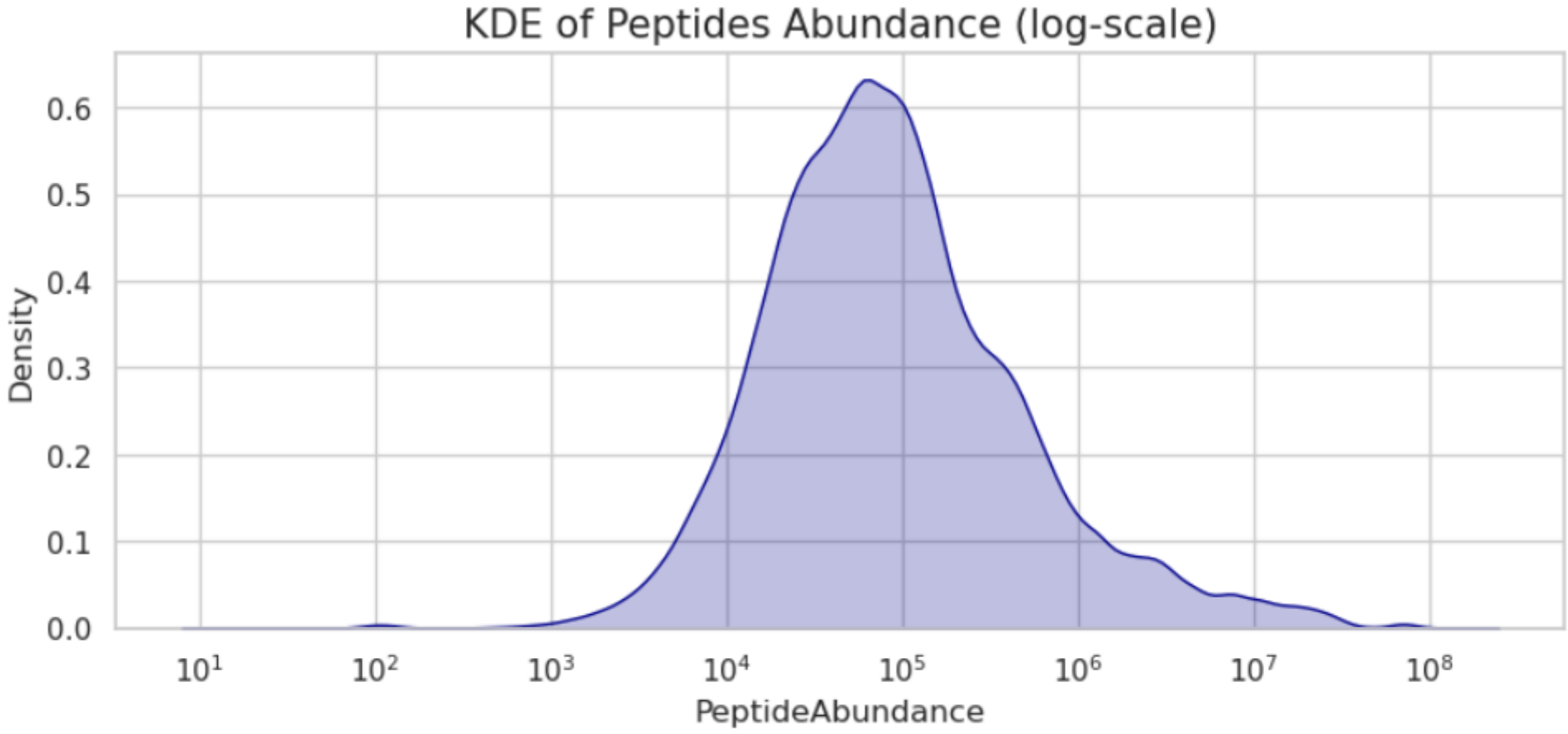


Proteins Abundance



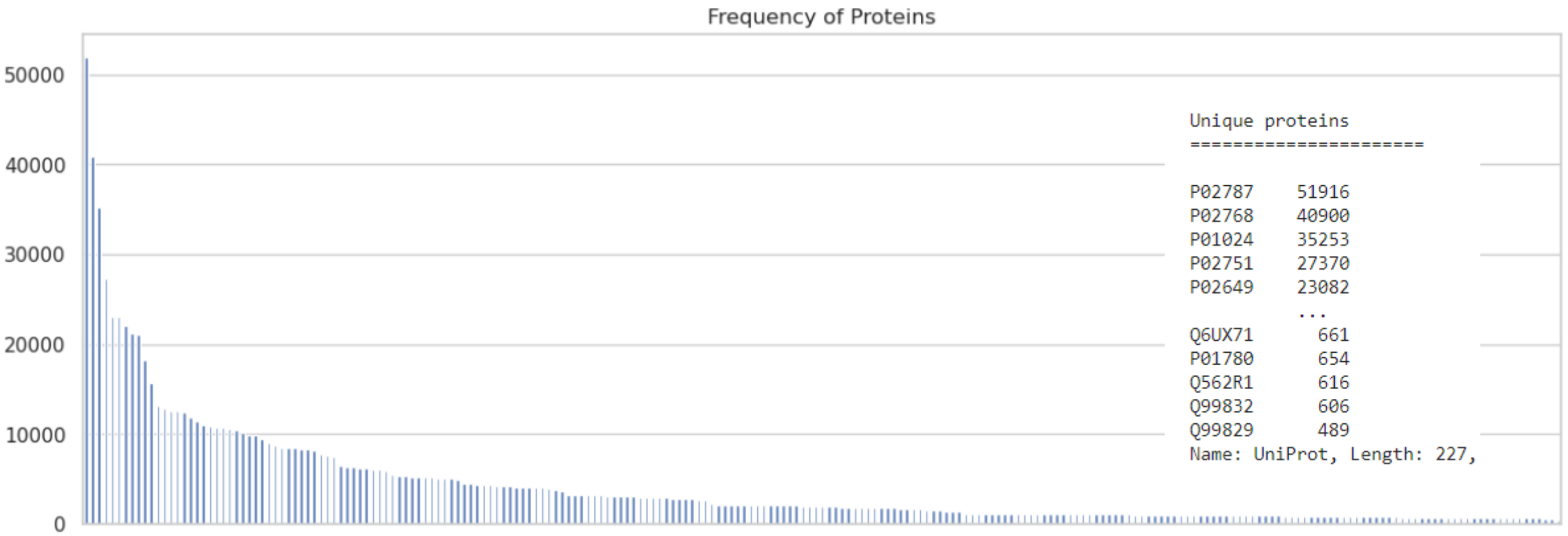
	count	mean	std	min	25%	50%	75%	max
NPX	232741.0	2.712077e+06	2.224155e+07	84.6082	29464.4	113556.0	563894.0	613851000.0

Peptides Abundance

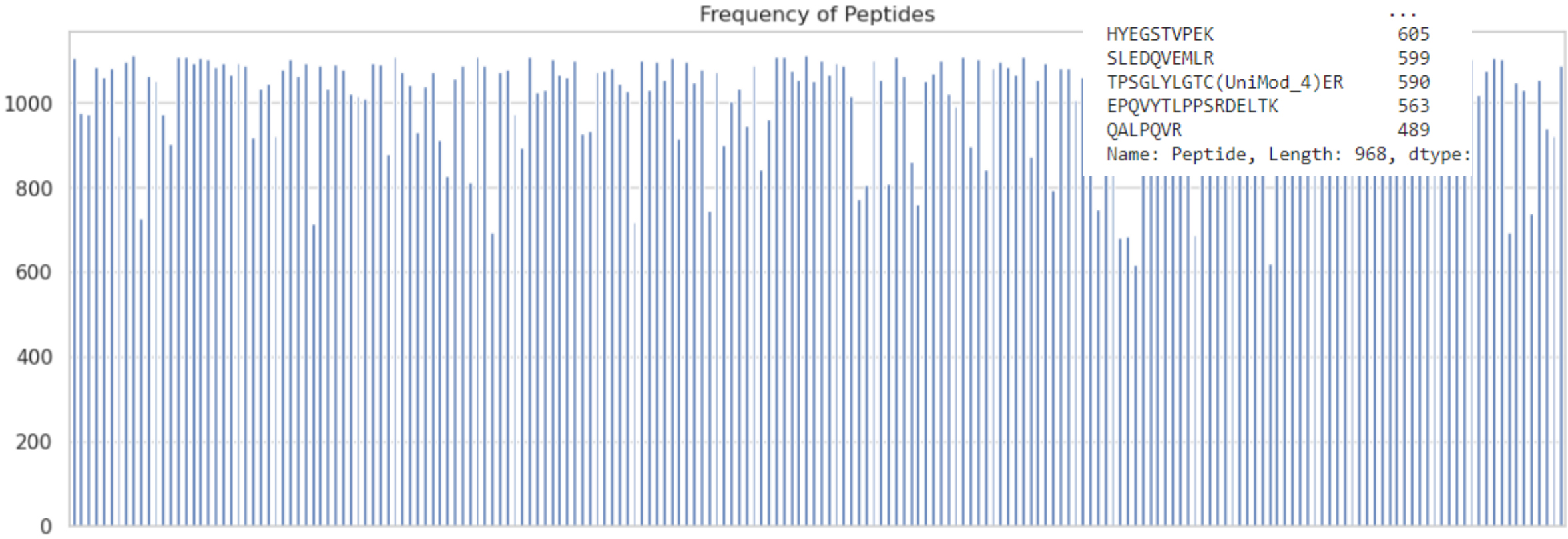


	count	mean	std	min	25%	50%	75%	max
PeptideAbundance	981834.0	642890.245933	3.377989e+06	10.9985	28174.25	74308.3	221338.75	178752000.0

Proteins Frequency



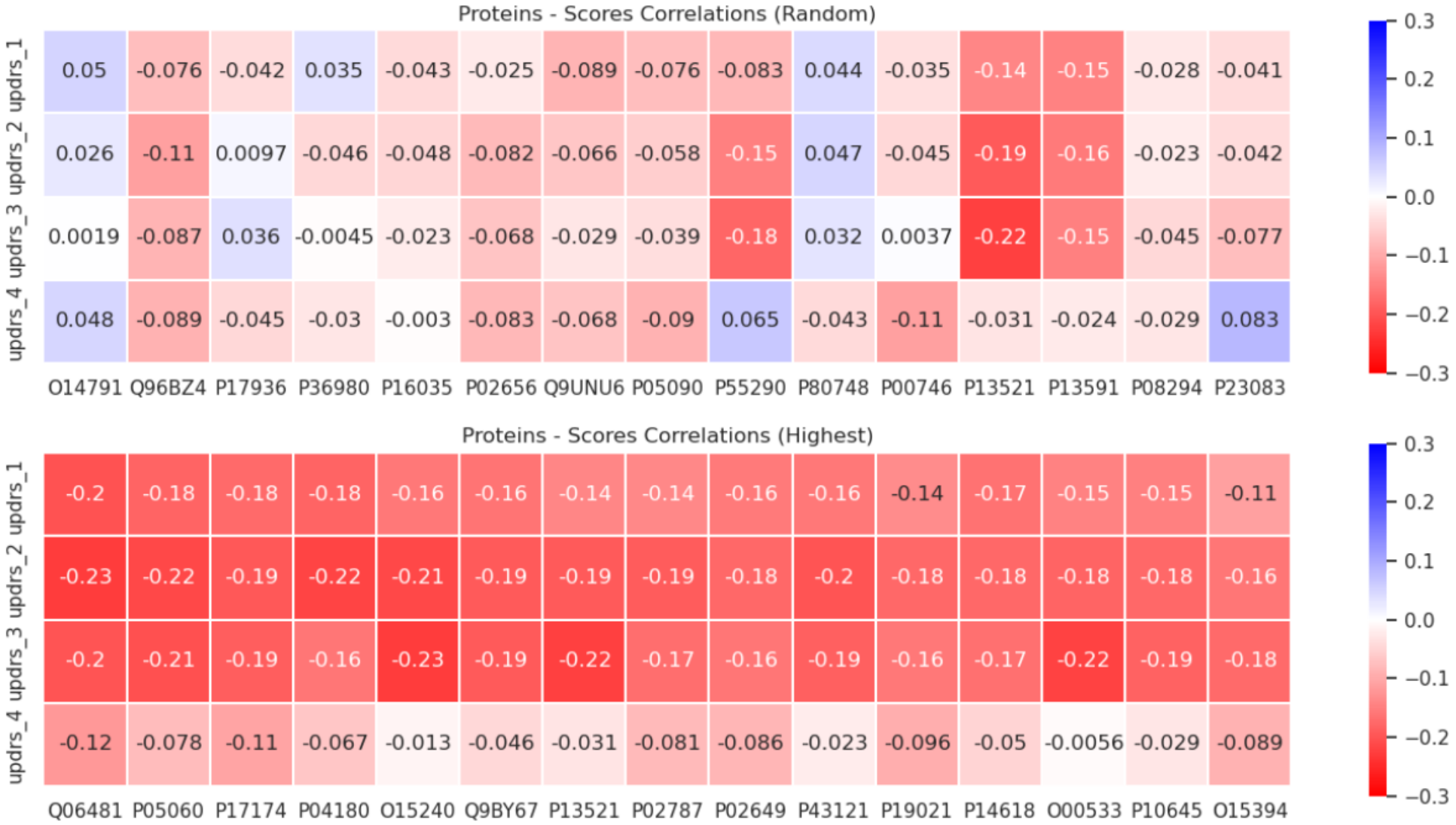
Peptides Frequency



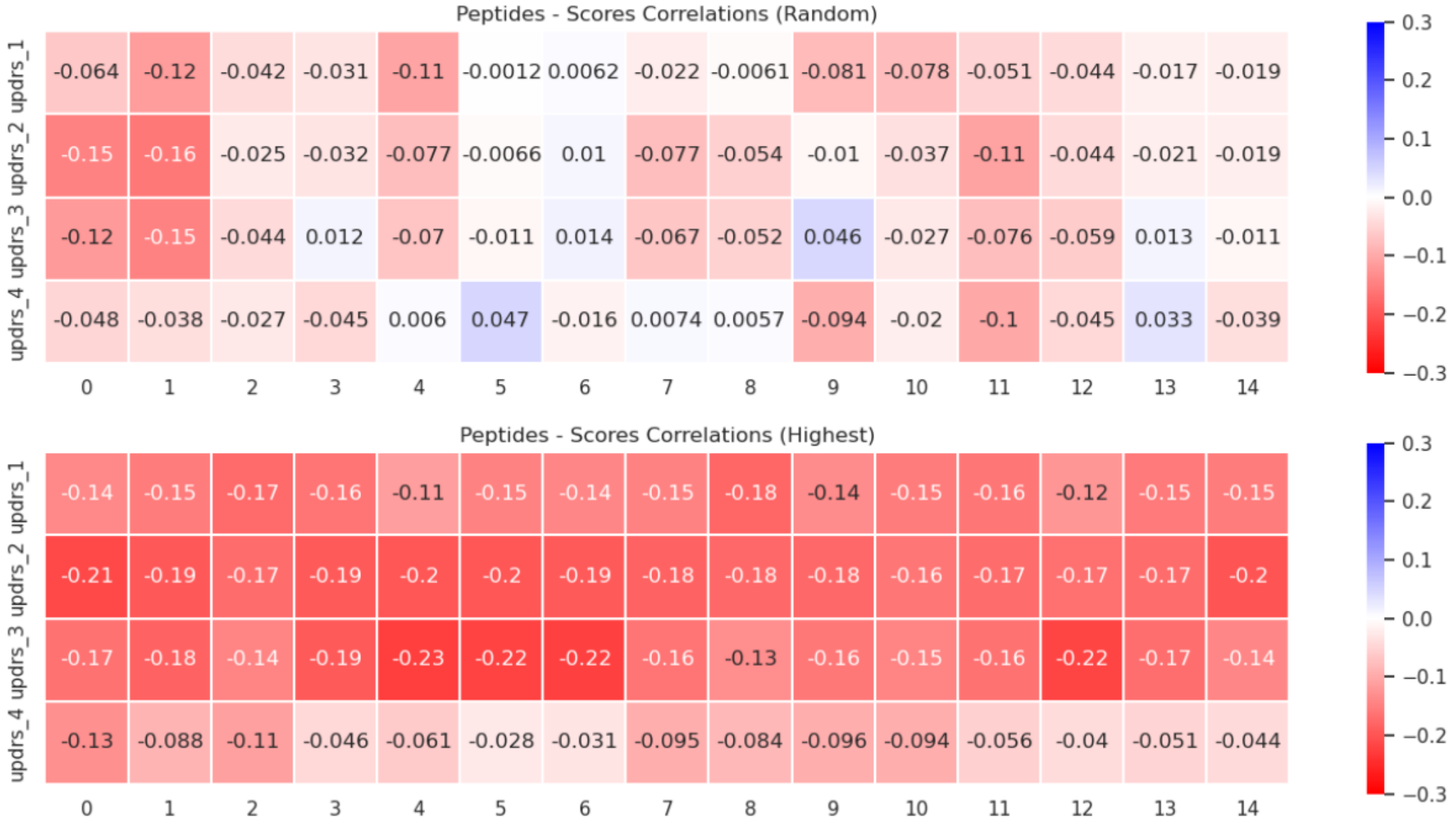

```
Unique peptides
======================

TLLSNLEEAK                1113
IPTTFENGR                 1113
NILTSNNIDVK               1113
KYLYEIAR                  1113
AIGYLNTGYQR               1113
                          ...
HYEGSTVPEK                 605
SLEDQVEMLR                 599
TPSGLYLGTC(UniMod_4)ER     590
EPQVYTLPPSRDELTK           563
QALPQVR                    489
Name: Peptide, Length: 968, dtype:
```

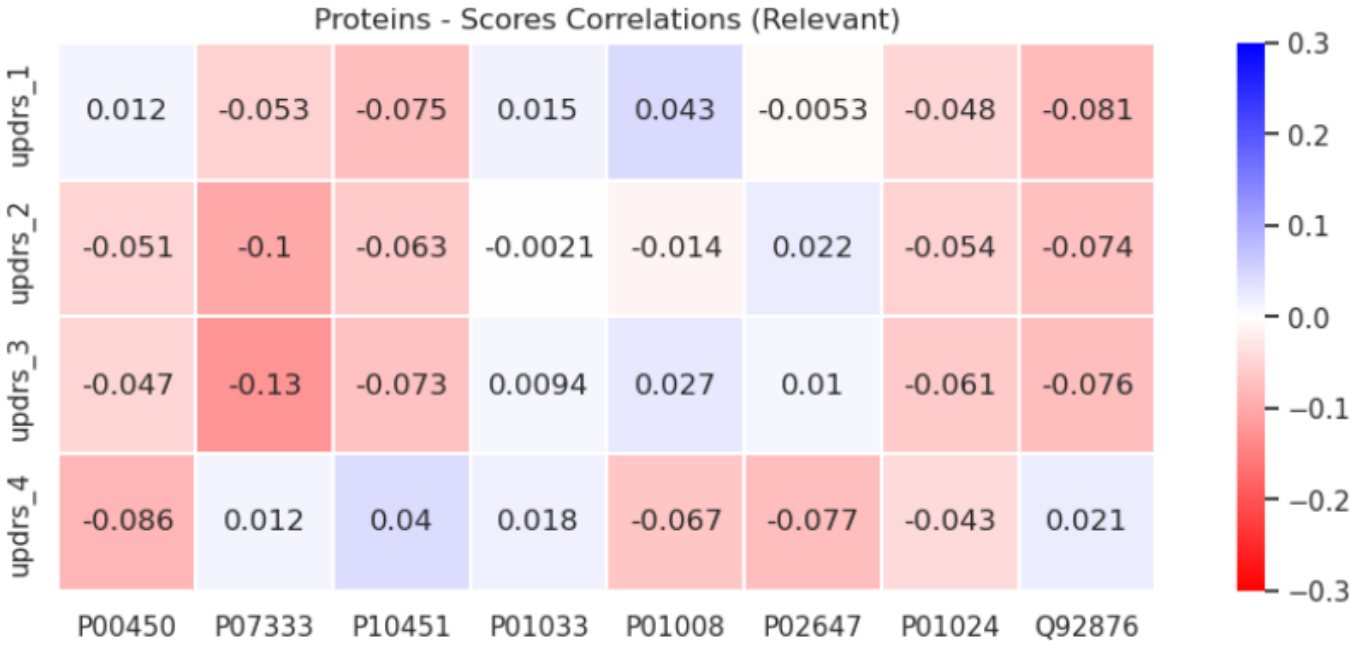
Correlations Scores - Proteins

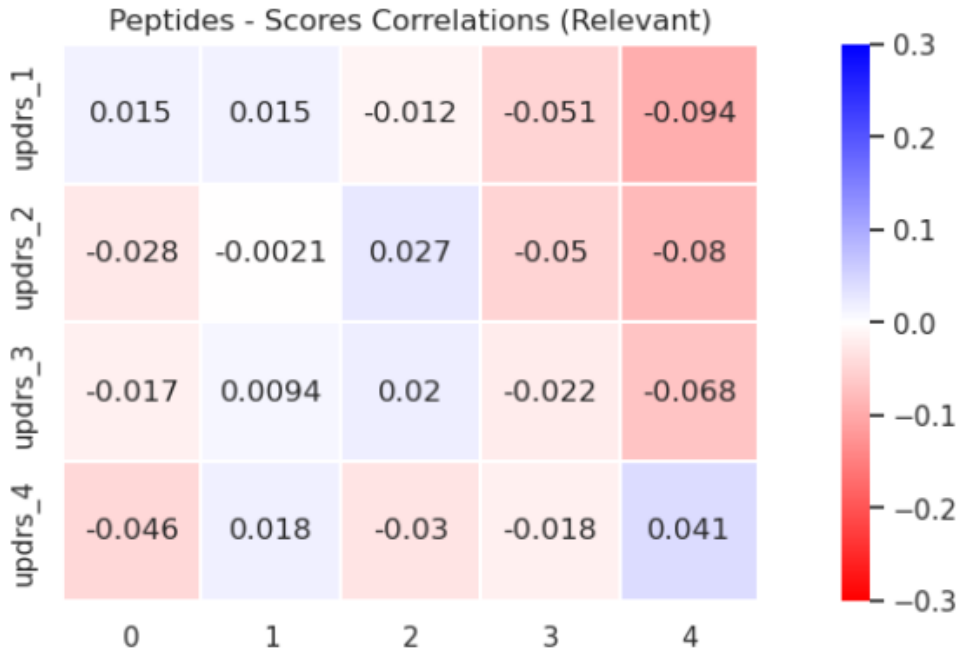


Correlations Scores - Peptides



Cherry-picked Correlations





Essential Preprocessing

Remove Irrelevant Data

Clinical: Ids 2615 Length 2615
Proteins: Ids 1113 Length 232741
Peptides: Ids 1113 Length 981834

Remains 40.8% of the original data,
1068 visits, 248 patients

Medication

- set NAs to 0
- create a dummy that takes value 1 if medication is NA

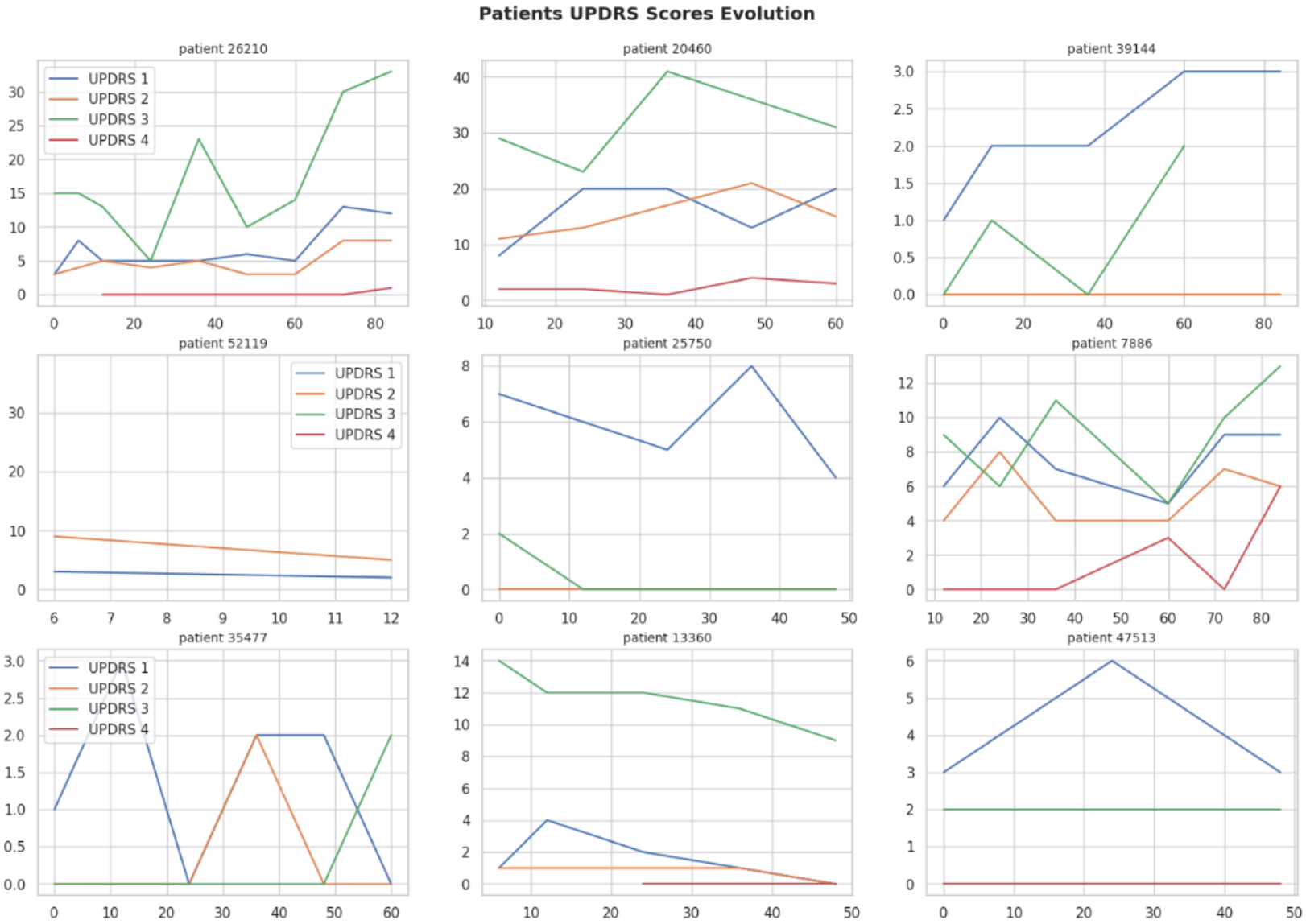
Prediction Task

Predict UPDRS scores 12 months ahead

from current UPDRS scores, medication,

abundances, stage (month)

Remove Irrelevant Data



Remove Irrelevant Data

No Imputation to get more cases

- No clear patterns for interpolation
- We cannot use patient information in for test data (new patient)

Keep cases that have next year data

Data available for 20.8% of the original data, 543 visits

Clinical: Ids 543 Length 543
Proteins: Ids 543 Length 113026
Peptides: Ids 543 Length 475488

Relative Abundance

$$\text{Relative Peptide Abundance} = \frac{\text{Peptide Abundance}}{\text{Corresponding Protein Abundance}}$$

Temporal Features

- Visit month => indicates stage of the disease
- Visit n months before? $n = 3,6,9,12,15,18$ => more frequent visits may be related to the severity

Features

We obtain 2,175 columns

- the visit month
- the 2 dummies for medication
- the 4 UPDRS scores
- 6 indicators on recent visits from 3 months earlier to 18 months earlier
- 227 protein abundance measurements
- 968 peptide abundance measurements
- 968 peptide relative abuncance measurements

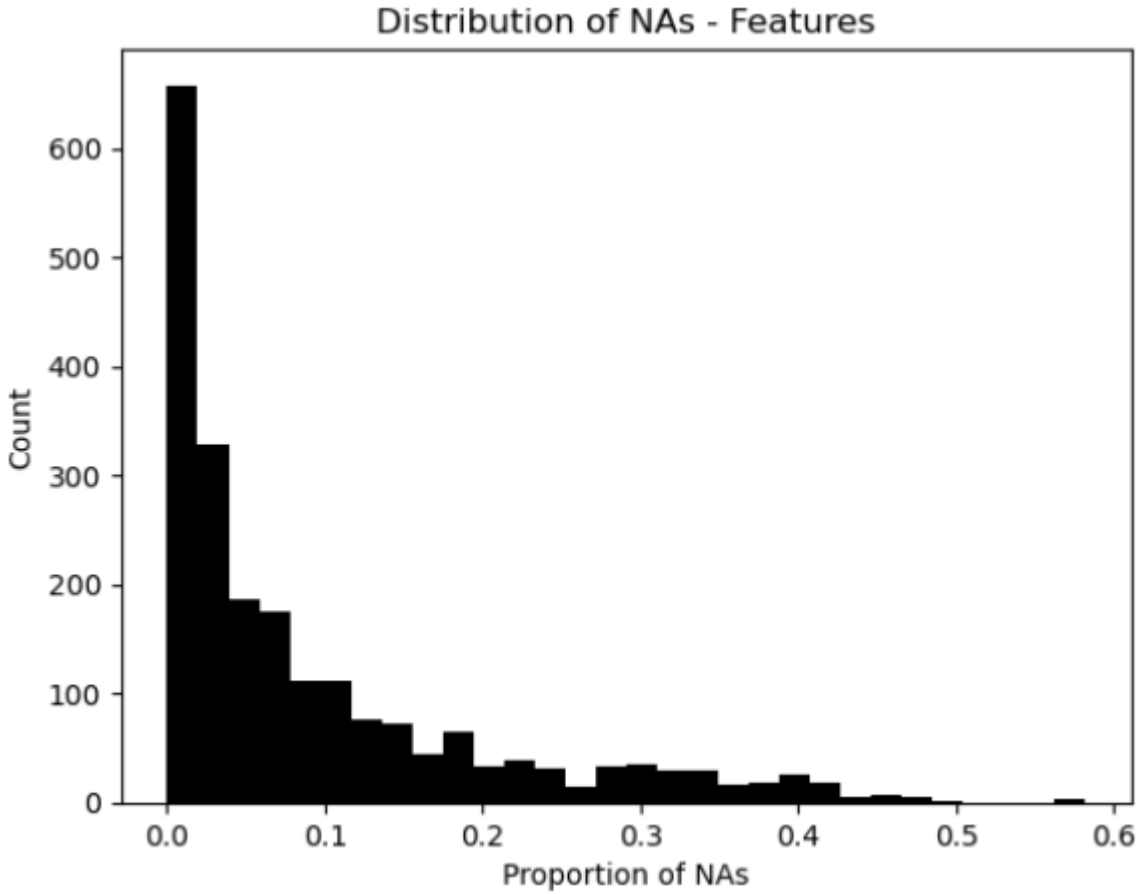
Experimental Preprocessing

Preliminary Notes

Some preprocessing tasks require being applied
separately to each CV fold training set

Data Processing choices are also constitute
hyperparameters

Features Imputation



Imputation Strategies

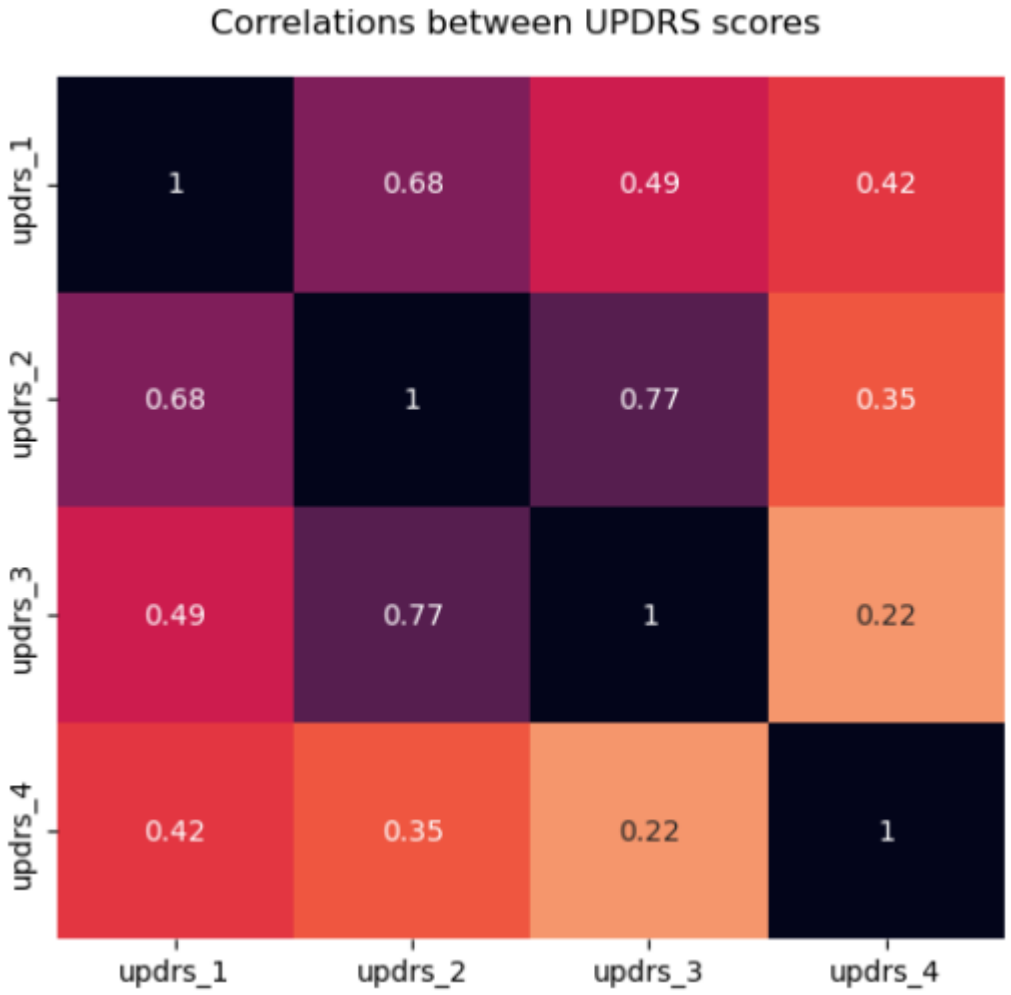
- 0s
- Mean
- Median

Targets Imputation

	updrs_1	updrs_2	updrs_3	updrs_4
Prop NA	0.0	0.0	0.92081	33.333333

Keep track of Imputed values for evaluation

Targets Imputation



Imputation Strategies

- Mean
- Median
- KNN

Features Augmentation

- $X' = X$
- $X' = \{X, \sqrt{X}\}$
- $X' = \{X, \sin(X)\}$
- $X' = \{X, \sqrt{X}, \sin(X)\}$

Normalizing Features

$$X' = \frac{(X - \mu_X)}{\sigma_X}$$

Features Screening - SIS

The Sure Independence Screening method simply consists in keeping $k < J$ features by selecting the ones that are the most correlated with the target. Remind that:

$$\rho_{x,y} = \frac{\text{cov}(x, y)}{\sigma_x \sigma_y} = \frac{\mathbb{E}[(x - \mu_x)(y - \mu_y)]}{\sigma_x \sigma_y}$$

Considering that our set of features X is normalized (centered around the mean and scaled by the standard deviation) we have, for any feature x:

$$\rho_{x,y} \equiv \mathbb{E}[xy]$$

We can thus compute the vector ω which is proportional to the marginal correlations:

$$\omega = X^T y \equiv N \rho_{X,y}$$

We will keep a proportion p of features which absolute score $|\omega_j|$ is among the $\lfloor Jp \rfloor$ highest values $|\omega|$. ($\lfloor \cdot \rfloor$ is the floor operator).

Note since we have 4 target variables we will perform the screening on all of them and keep all the features selected, meaning that the parameter p of this algorithm does not determine the final proportion $\tilde{p}$ of selected features. The overall proportion could be at max

$$\tilde{p}_{\text{max}} = \begin{cases} 4p & \text{if } p \leq 0.25, \\ 1 & \text{otherwise.} \end{cases}$$

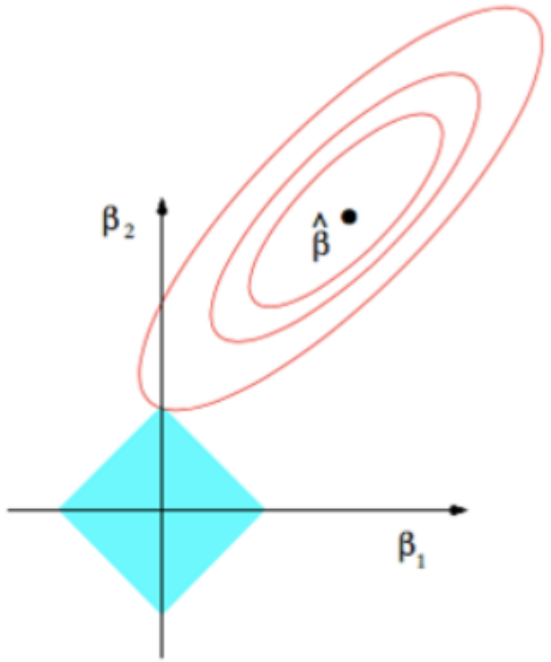
while the minimum is $\tilde{p}_{\text{min}} = p$ in the case the same features are selected for the four targets.

Features Selection - Lasso

We use a Lasso regression to select features, either without the SIS or after as a complement. The Lasso is the 1st order ($q = 1$ below) of the linear regression regularization techniques. Instead of minimizing the least squares of residuals we add to the latter loss function a *soft constraint* term as follows ($\boldsymbol{M}^{\cdot n}$ denotes the element-wise power of a vector or matrix):

$$\hat{\beta} = \text{argmin}_{\beta} \, (y - \hat{y})^{\cdot 2} + \alpha |\beta|^{\cdot q}$$

The estimated coefficients are constrained to be on the contour of the blue region below (source: Hastie *et al.* (2001/2009)):

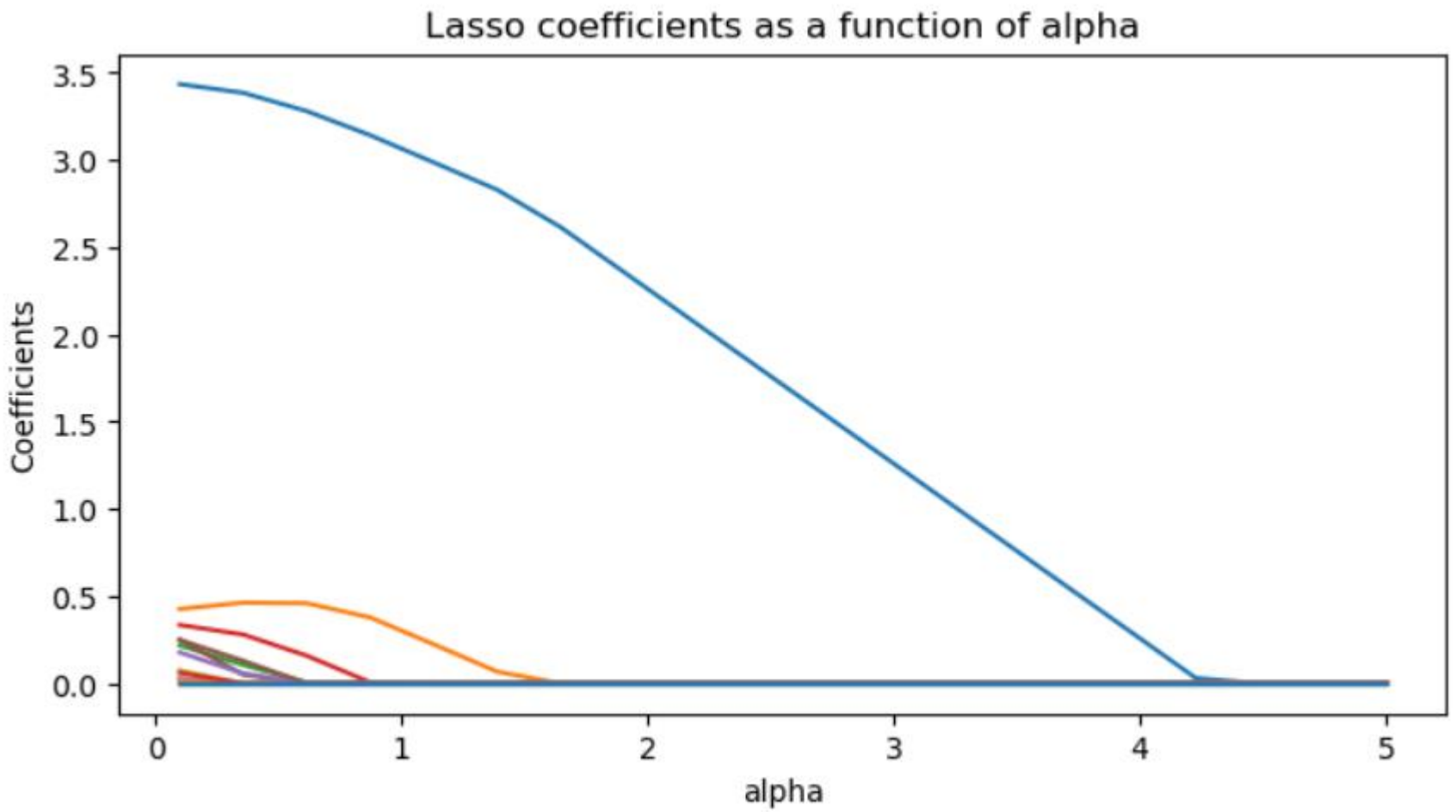


+ Rescaling

Like for SIS we will repeat it for the 4 targets and keep the union of the 4 selections.

Features Selection - Lasso

- SIS 1% + Lasso



Dimensionality Reduction - PCA

Principal Component Analysis (PCA)

Principal Components (PCs) are mutually uncorrelated affine projections of the data $X \in \mathbb{R}^{N \times J}$. PCA consists in a projection onto a subspace of dimension $q \leq J$ (or $q \leq N$ if $N \leq J$). The basic idea to obtain the *best* linear components can be depicted by an iterative procedure. The first PC is the linear combination of the J variables that has the maximum variance among all linear combinations. The second PC is the variance-maximizing linear combination that is orthogonal to the first PC, etc. This maximum variance property can be obtained by minimizing the the sum of squared distances between points X_i and their projection onto the PC.

Following Hastie *et al.* (2001/2009)) we propose the following formalization. Our objective is to find the dimension-q representation $f(\lambda)$ of our data X, which can be represented by a rank-q hyperplance (we consider that X is centered and scaled):

$$f(\lambda) = \mu + V_q \lambda(X) \equiv V_q \lambda$$

where V_q is a $J \times q$ matrix with orthogonal columns and lambda is a q-dimensional vector. We can find λ by minimizing the least squares of the reduced representation, $X_i - V_q \lambda_i \in \mathbb{R}^J$:

$$\min_{\lambda_i, V_q} \sum_i \|X_i - V_q \lambda_i\|^2 \Longrightarrow \hat{\lambda}_i = V_q^T X_i$$

$V_q^T V_q$ is the *projection matrix*, that is $V_q^T V_q X_i$ is the projection of X_i onto the subspace described by the first q PCs of X. In order to find V_q we minimize the least squares between the projection and the real values:

$$\min_{V_q} \sum_i \|X_i - V_q^T V_q X_i\|^2$$

In practice all the PCs (J or N in number) can be found by solving the singular value decomposition of X:

$$X = U D V^T$$

where U and V are respectively $N \times J$ and $J \times J$ orthogonal matrices, and D is a $J \times J$ diagonal matrix with $d_1 \geq \cdots \geq d_J \geq 0$. Any V_q is just given by the first q columns of V and the PCs of X are given by $U D$.

Dimensionality Reduction - PCA

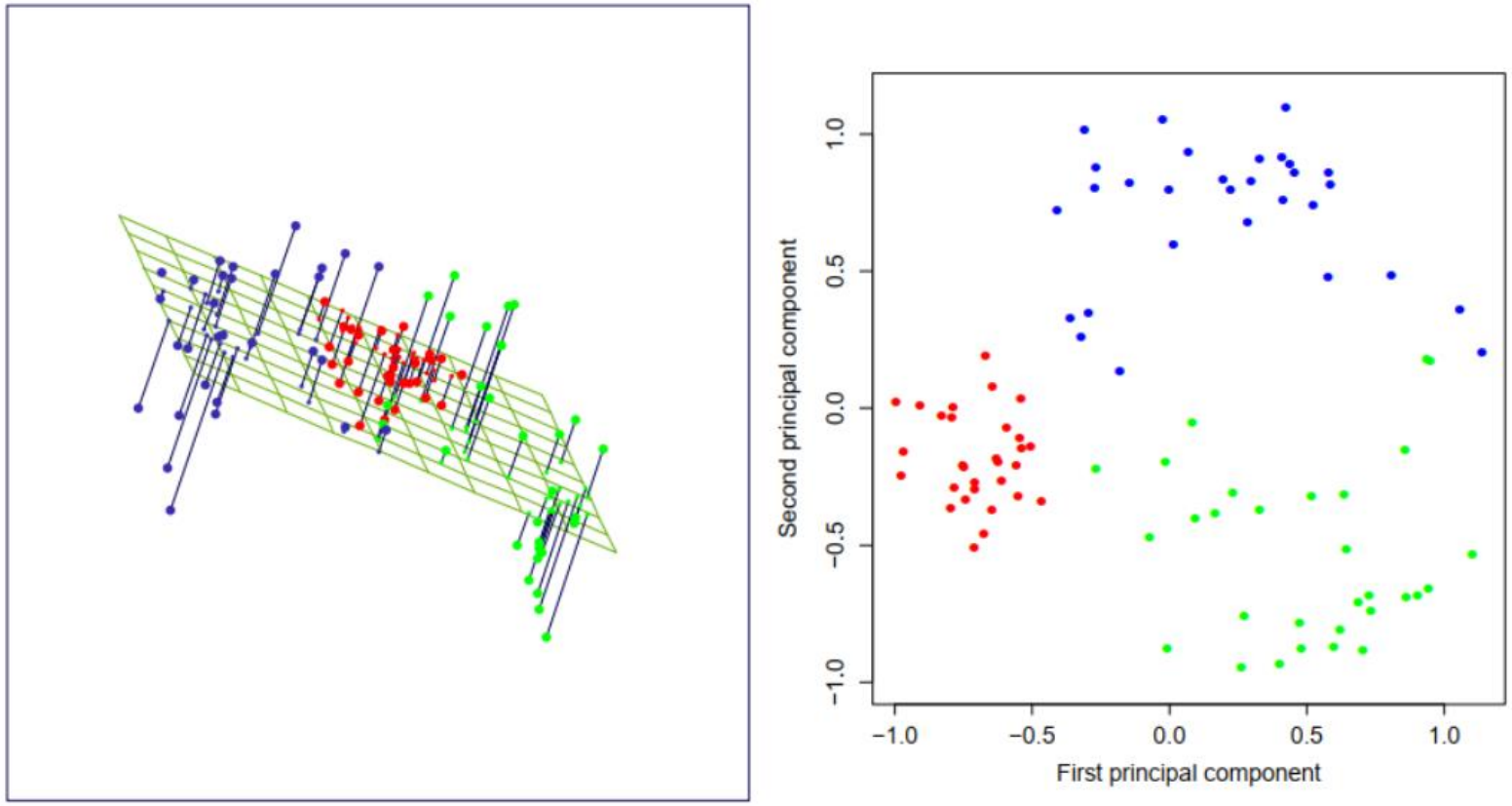


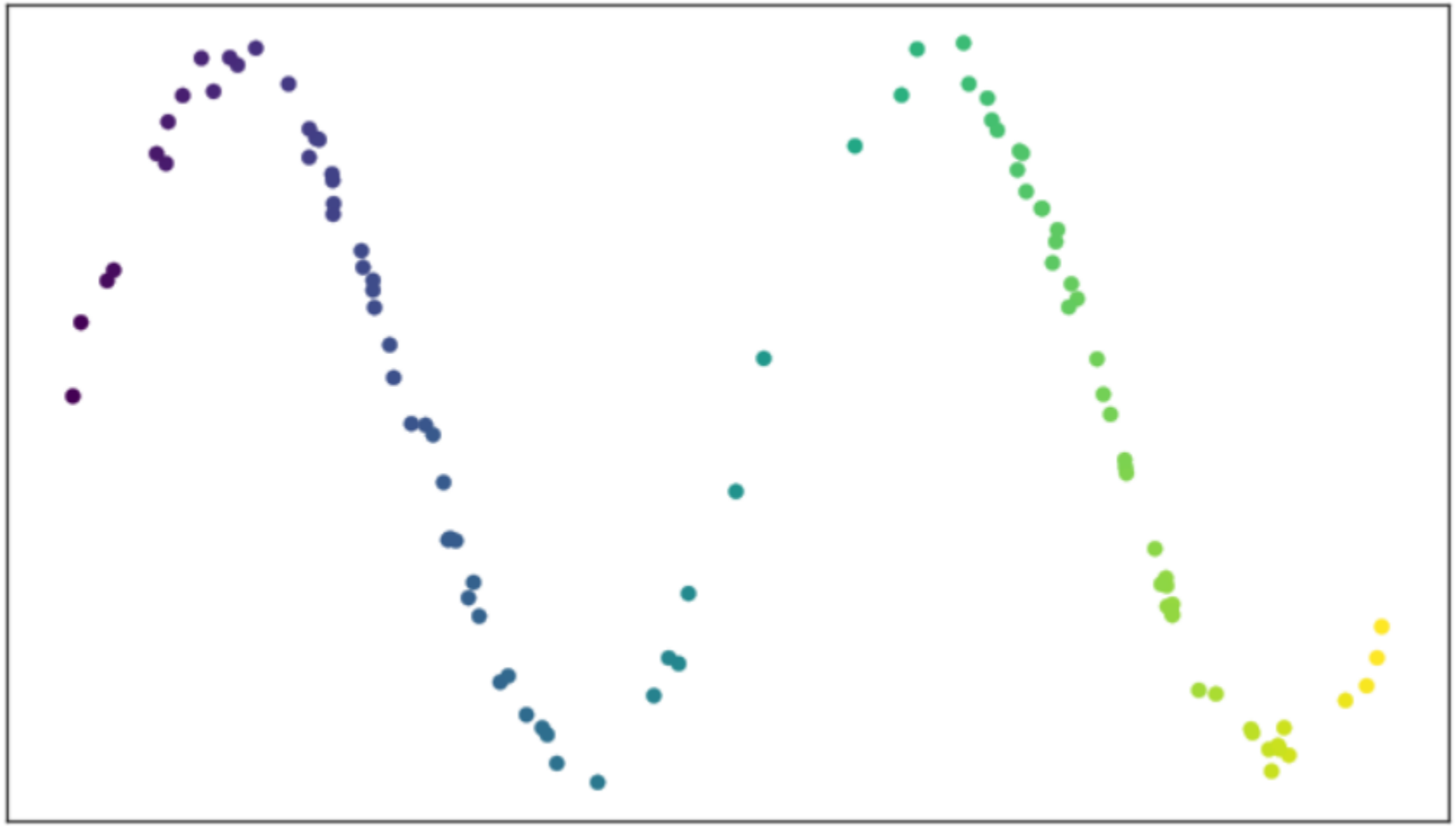
FIGURE 14.21. *The best rank-two linear approximation to the half-sphere data. The right panel shows the projected points with coordinates given by* $\mathbf{U}_2\mathbf{D}_2$*, the first two principal components of the data.*

Dimensionality Reduction - UMAP

The UMAP algorithm of McInnes *et al.* (2018 has been designed to perform the same job as the t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm[4] while overcoming its low performance for large datasets. These algorithms reduce the dimensionality from J to q by iteratively moving points in the q-dimensional space such that the dissimilarity from the original J-dimensional space decreases. Each of the two spaces is associated with a $N \times N$ matrix which entries give the probability that two points are neighbors.

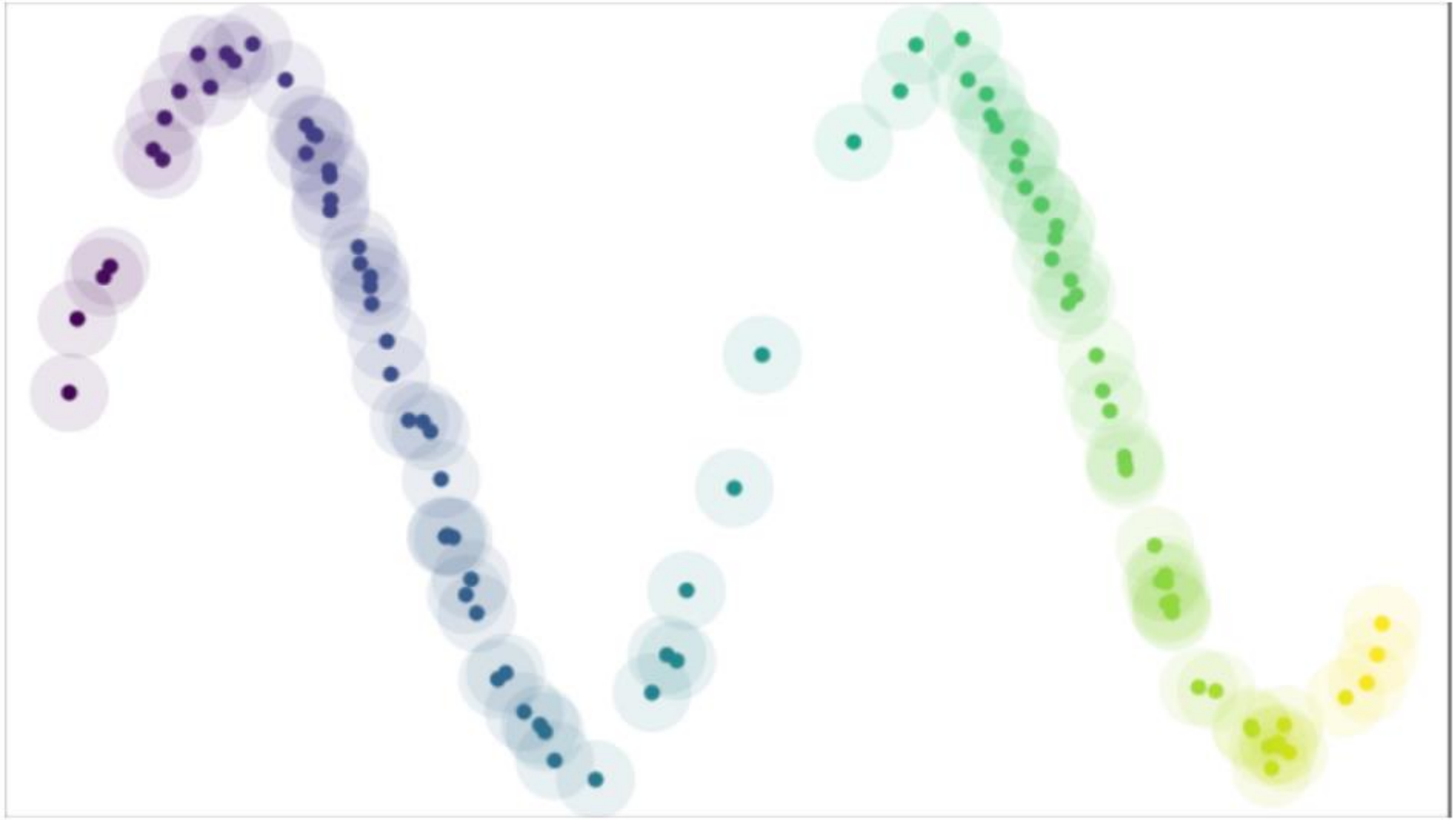
UMAP first constructs a *fuzzy simplicial complex* of the original space. While in practice the algorithm does not work with simplicial complices but distances, the authors used this topological theory to prove that this approach preserves the structure of the data. This representation is a graph which nodes are the N data points and weighted edges represent the probabilities of being in the same neighborhood (cluster). The iterative procedure will then attempt to reconstruct the same structure in the q-dimensional space.

Dimensionality Reduction - UMAP



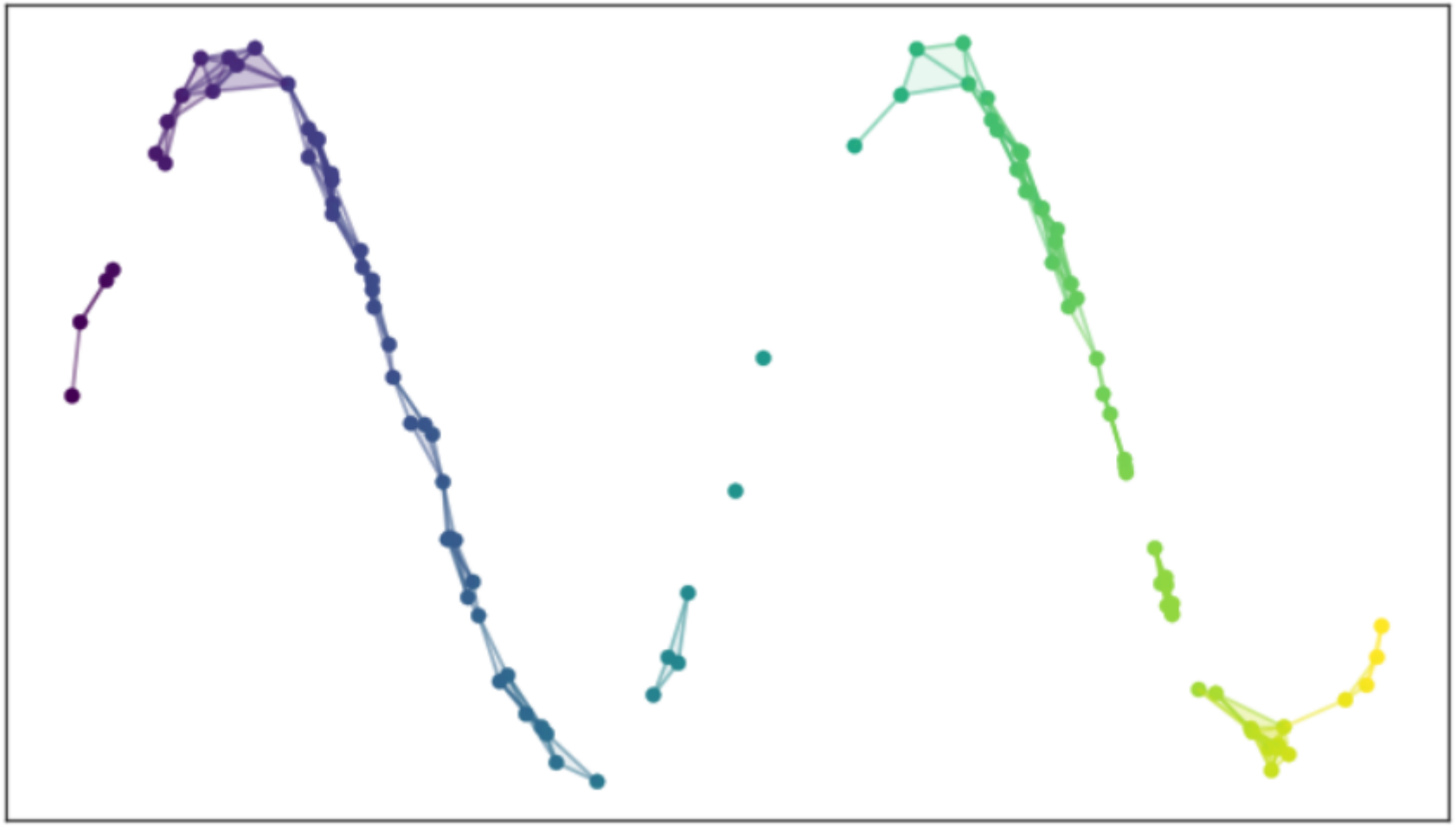
Dimensionality Reduction - UMAP

UMAP builds a ball outwards from each point.

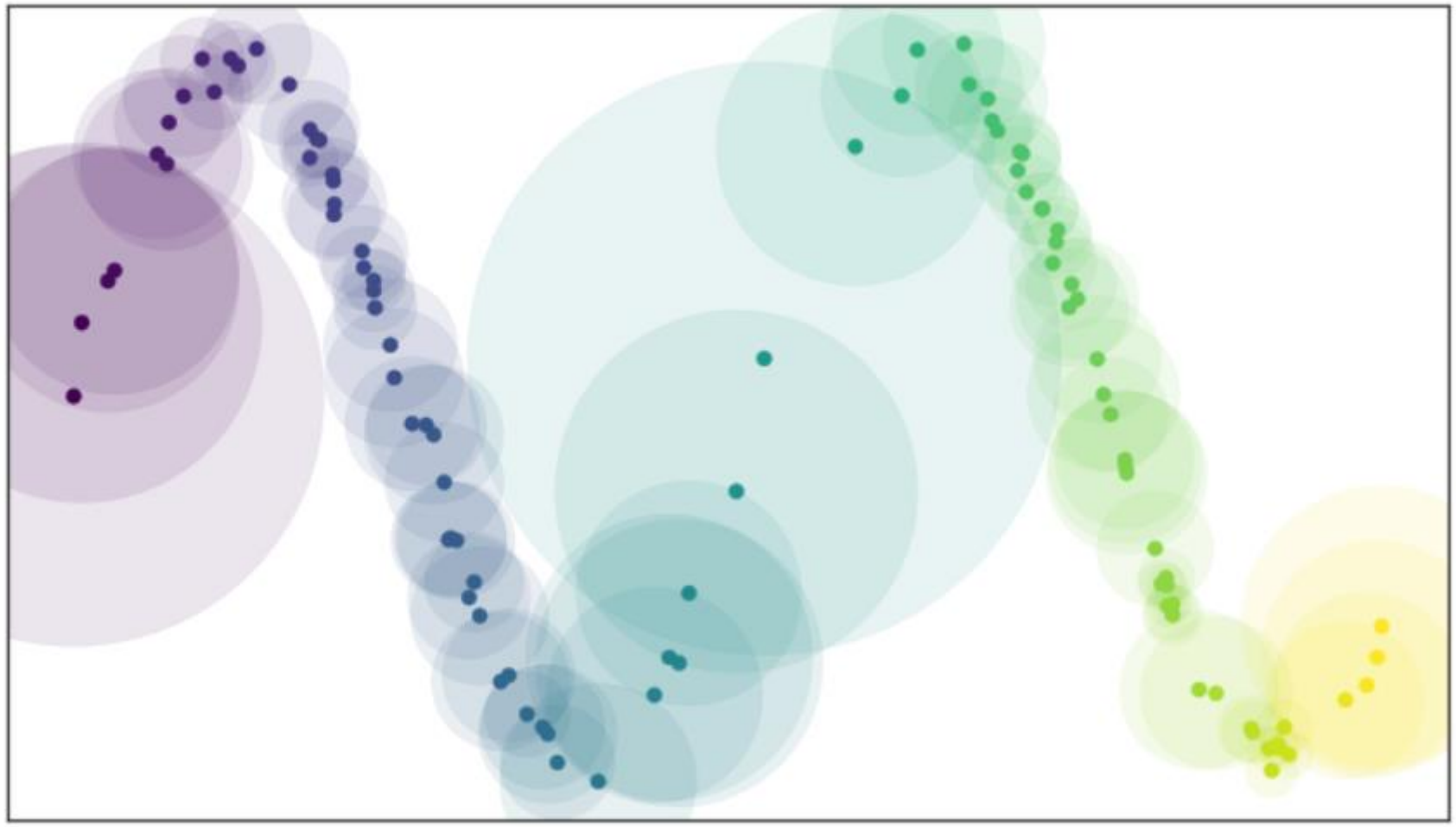


Dimensionality Reduction - UMAP

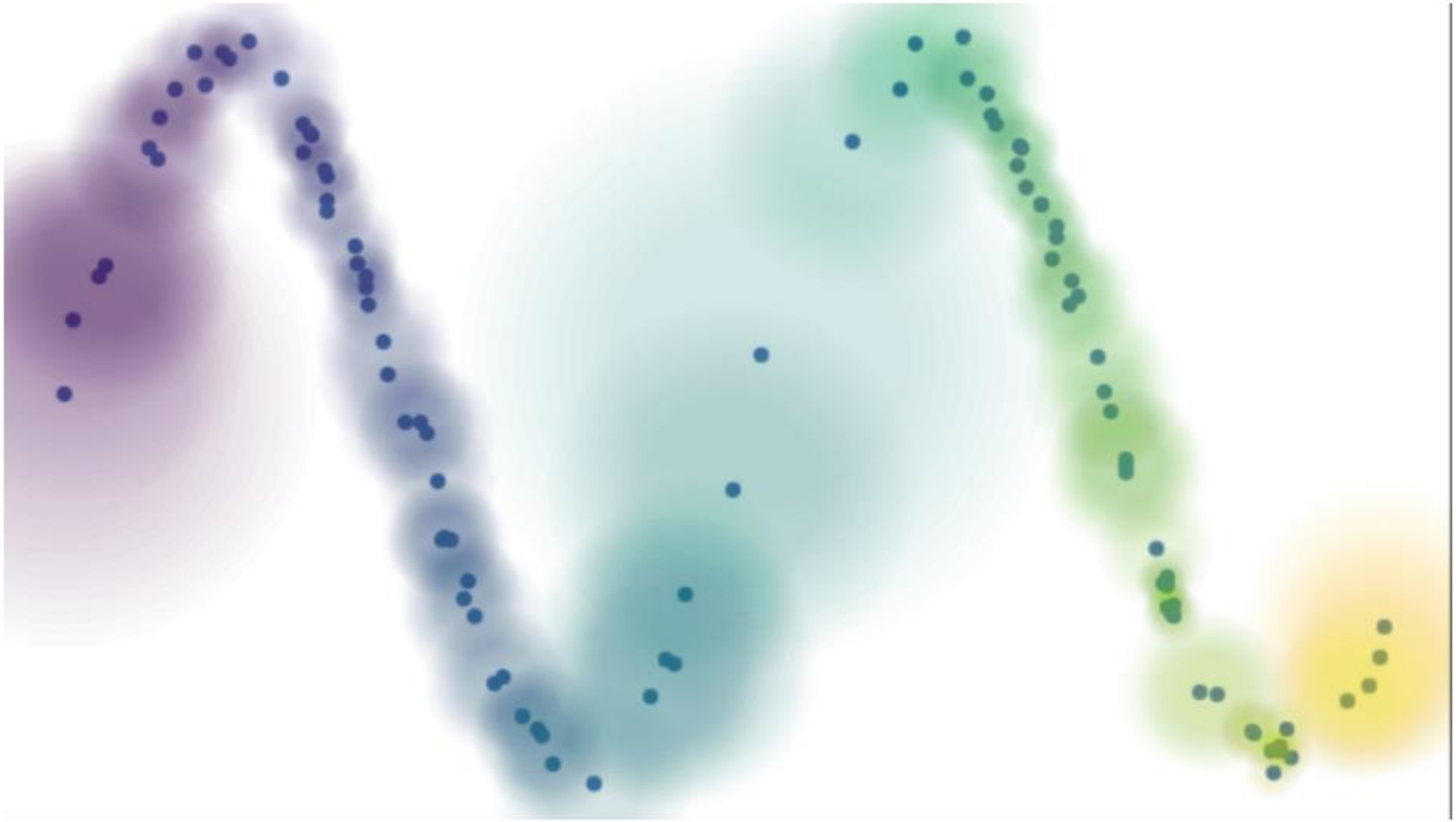
And the edges are obtained when balls intersect.



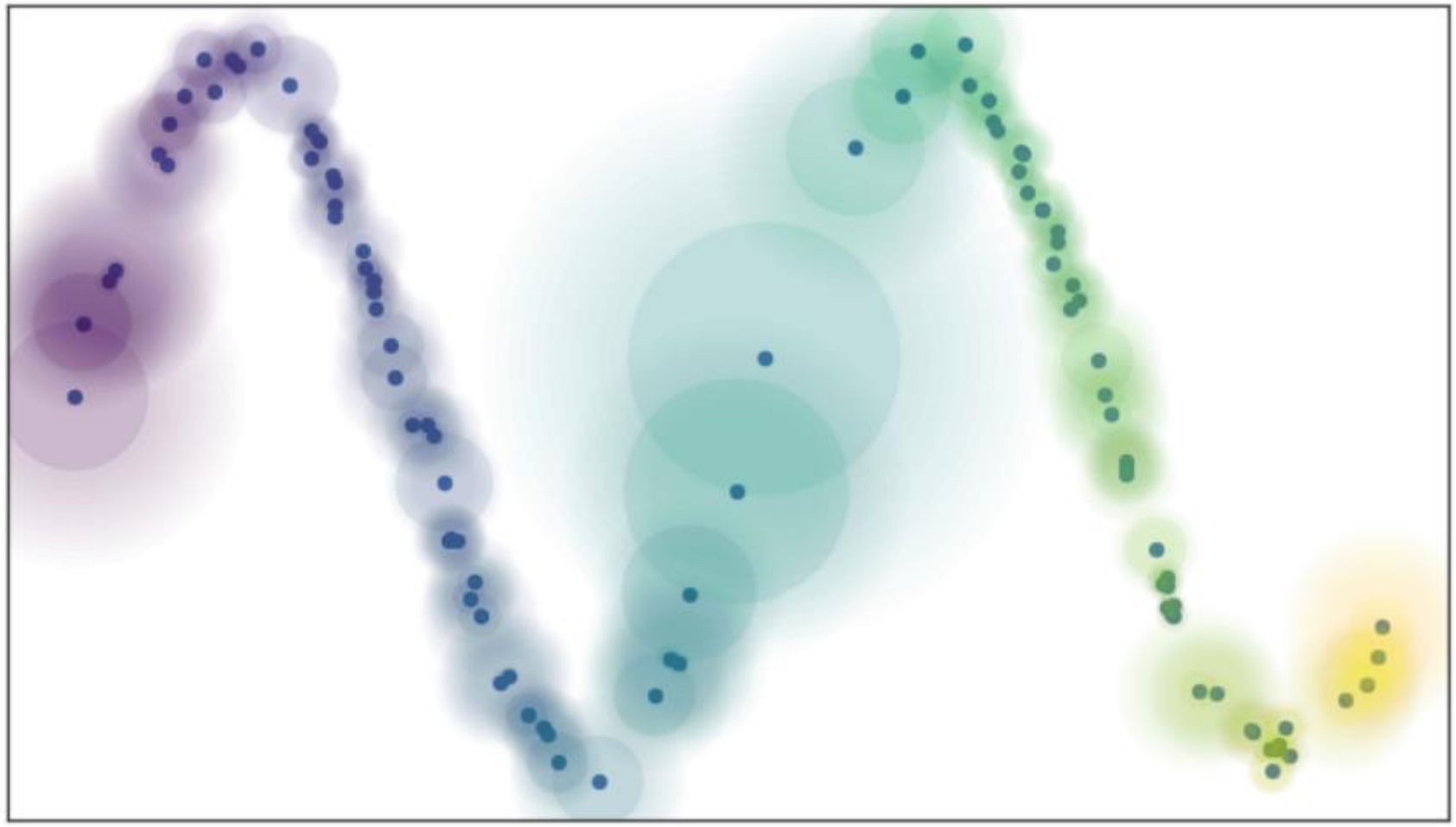
Dimensionality Reduction - UMAP



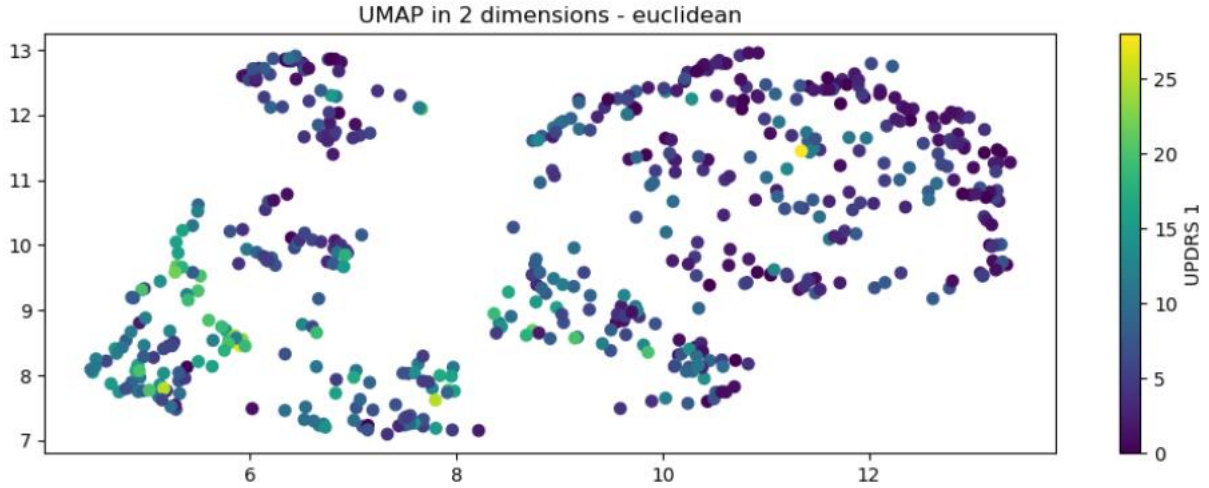
Dimensionality Reduction - UMAP

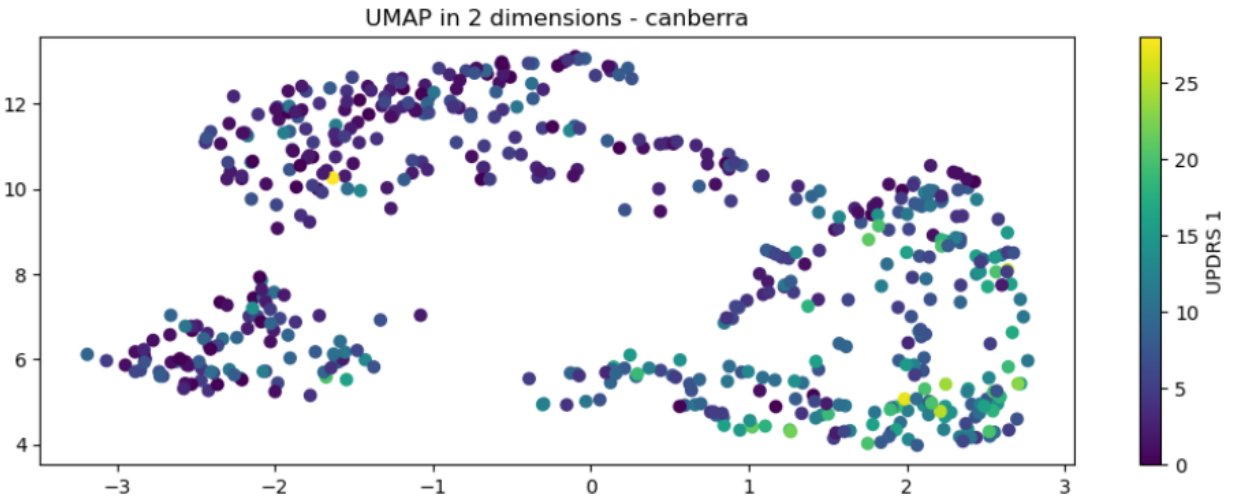


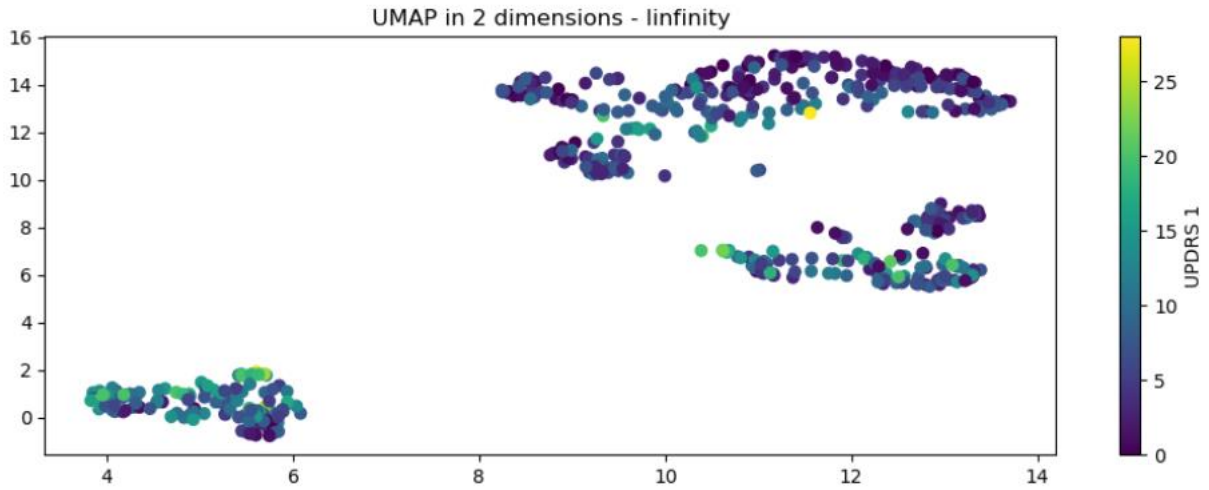
Dimensionality Reduction - UMAP



Dimensionality Reduction - UMAP







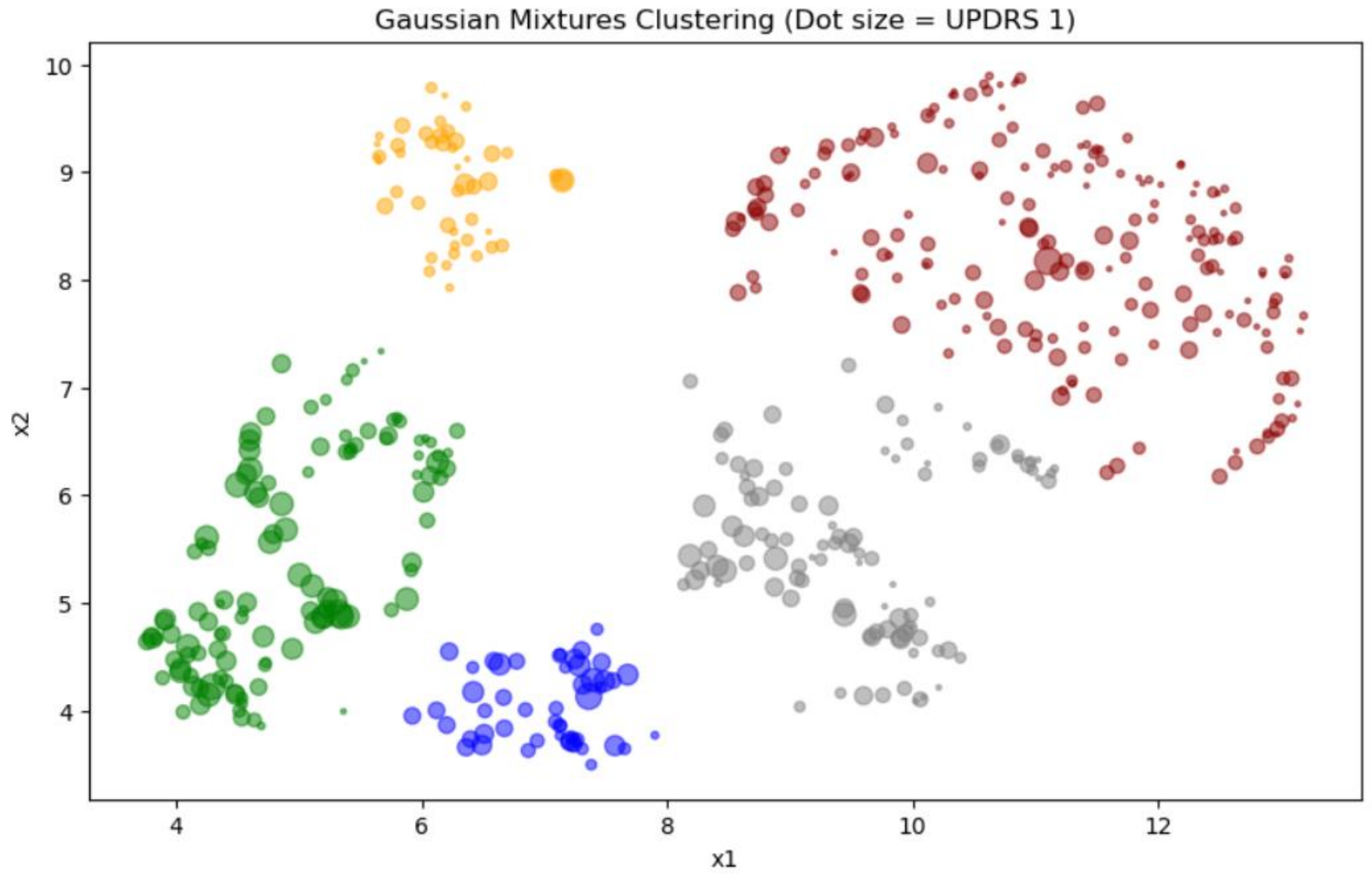
Clustering - GM

Gaussian Mixtures

A Gaussian mixture model (GMM) is a probabilistic model that assumes the data points originate from a combination of several Gaussian distributions with unknown parameters. In other words, it considers the possibility that each data point may have been generated by one of multiple Gaussian distributions. This expands upon the idea of k-means clustering by incorporating information not only about the means (centers) of these underlying Gaussians but also about their covariance structure, which describes the relationships between different dimensions of the data. By accounting for both the centers and covariances of the latent Gaussians, GMMs provide a more flexible and expressive approach to clustering than traditional k-means.

Clustering - GM

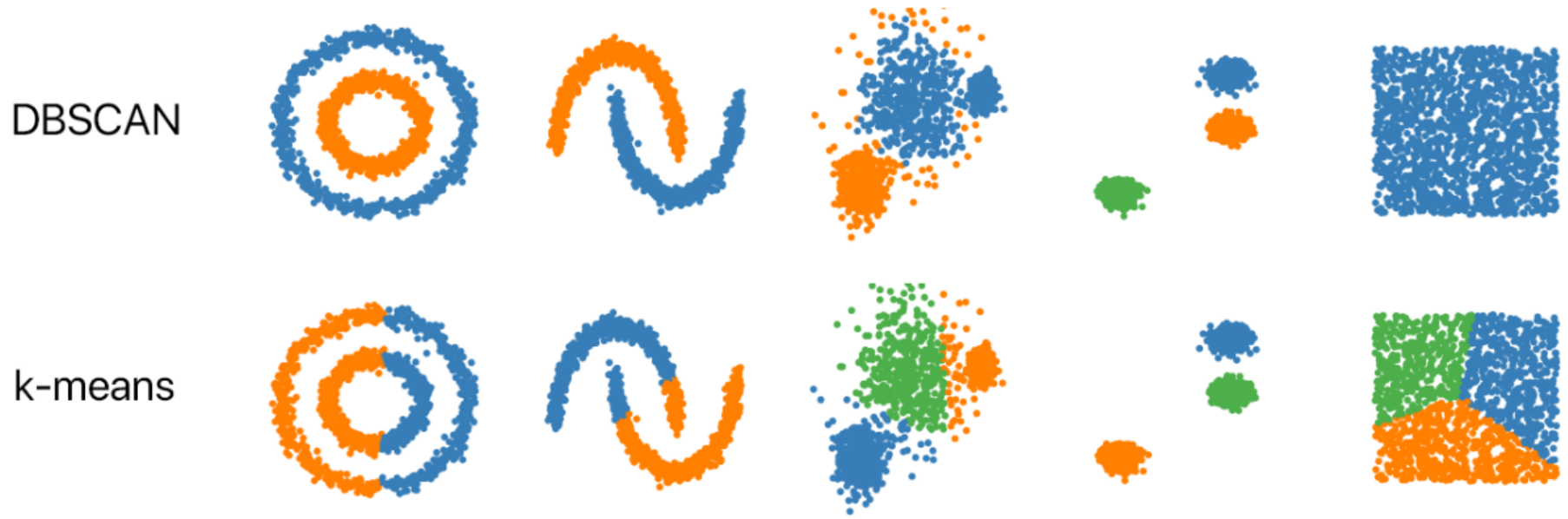
SIS 1% +
UMAP 2 components +
GM 5 clusters



Clustering - HDBSCAN

Hierarchical Density-based Spatial Clustering of Applications with Noise (HDBSCAN)

DBSCAN[5] attempts to identify clusters as humans do, that is, by looking at the density of points groups, or identifying grapes. Balls are drawn around points and *core points* are defined as those with the most neighbors. Clusters are then simply groups of points that have overlapping balls or neighbors with overlapping balls. This is found by iteratively checking the neighboring core points' neighbors until no more neighbors are found. Non-core points that are not assigned to a cluster of core points are called *outliers* and are typically assigned to the "cluster -1".



DBSCAN comes with only two important hyperparameters that significantly impact the clusters it finds: the ball size and the distance metric. Despite this and the fact that it is a non-parametric algorithm, DBSCAN is prone to the highly curse of dimensionality (many features) and the ball size parameter makes it highly unstable. Indeed, a slight modification in the ball size can lead to assign more points to a cluster, but most importantly the first clusters to be constructed are the first to attract the non-core points that can be assigned to more than one cluster. The repartition of points among clusters is thus very sensitive to this parameter.

Clustering - HDBSCAN

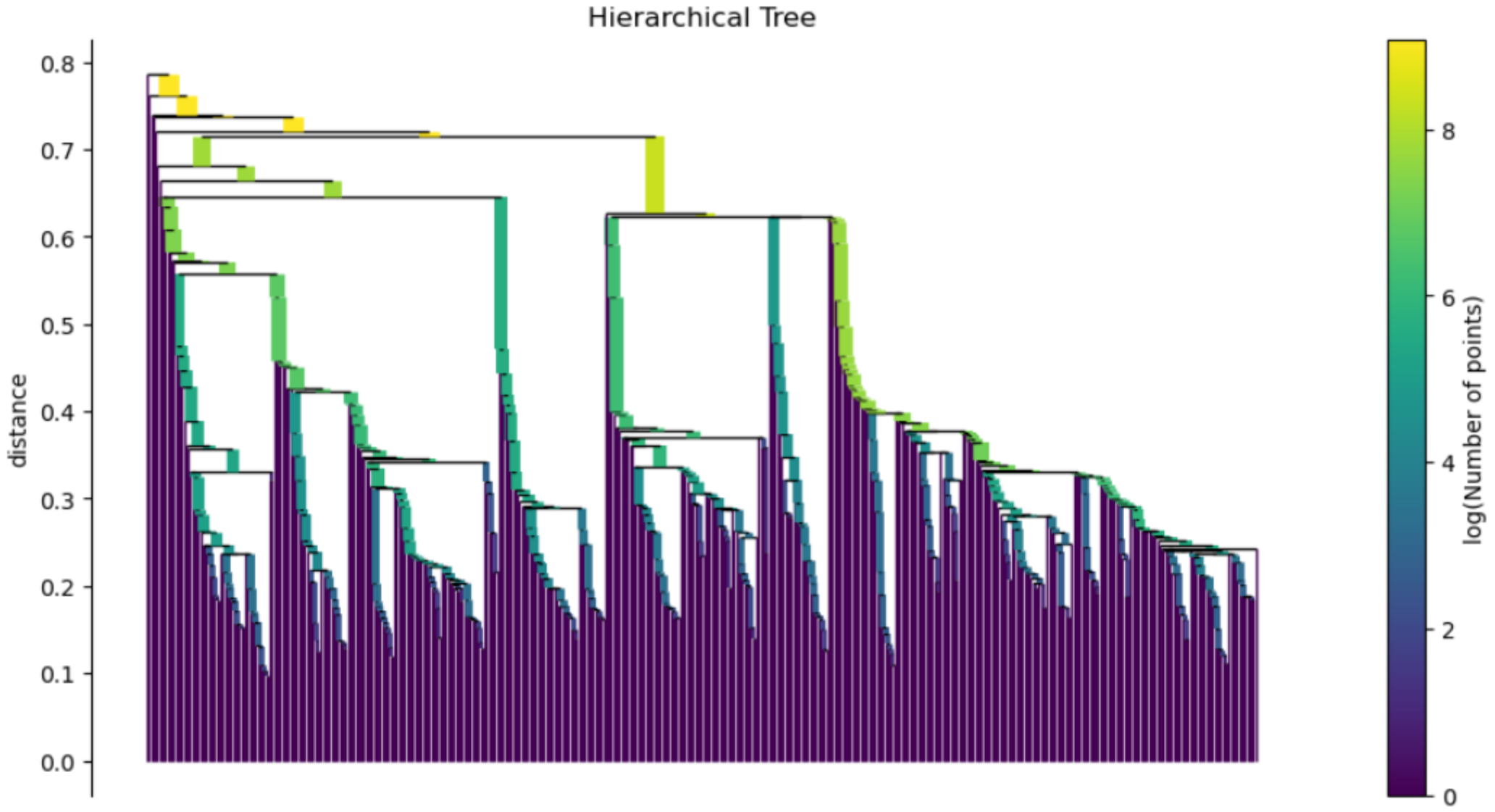
"*HDBSCAN is a clustering algorithm that extends DBSCAN by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based on the stability of clusters.*" HDBSCAN package

Instead of declaring whether a point is dense (core) or not dense (not-core), HDBSCAN defines the density on a continuous scale as a function of the ball radius required for DBSCAN to consider the point as core (core distance). That is, like for UMAP, the radius is defined as the distance to the k-th nearest neighbor. We then define the distance between any two points as the maximum between their distance and both their core distance. Again like for UMAP we can imagine that the data are represented by a graph. Here, the weighted edges equal the previously defined distance.

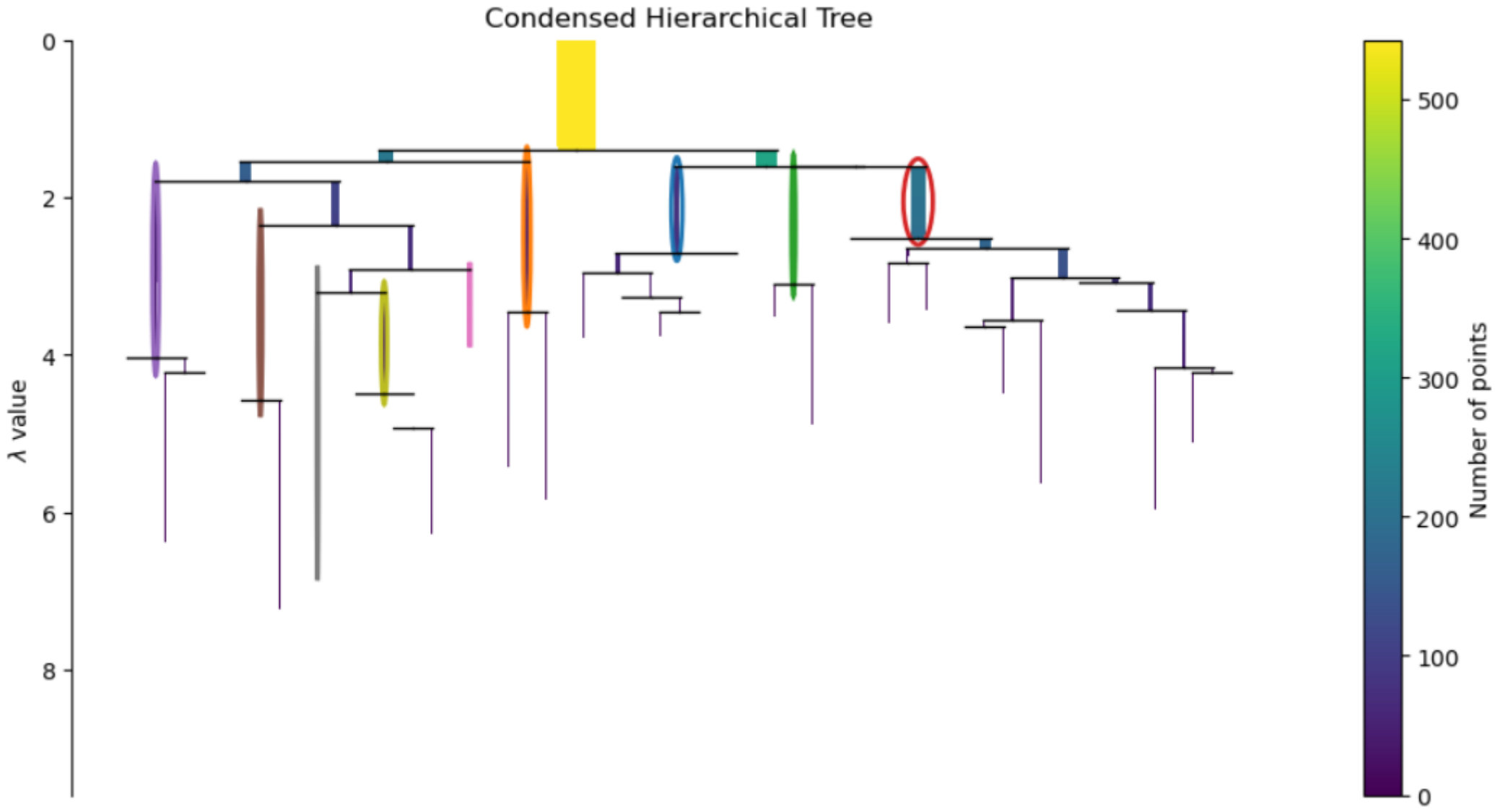
HDBSCAN defines a distance threshold, any edge above this value is dropped from the graph and the remaining edges form the clusters. To obtain a hierarchical clustering structure, the algorithm performs this step for multiple values of the threshold. As the threshold distance decreases we obtain more clusters. Instead of defining the flat clusters by cutting the tree as done usually, HDBSCAN iteratively remove *irrelevant splits*, that is, not considering them as splits leading to new clusters whenever the leaf size is too small. Doing this, the tree is both smartly cut (many branches are dropped but not always at the same vertical level) and smartly condensed (outliers are not included in clusters and merges can occur higher than the lower leaf level, that is, we have observe a merge then a split along the same parent branch). Finally, some clusters of the condensed tree are dropped if they are not big enough.

Moreover, while DBSCAN only predicts cluster labels, HDBSCAN predicts probabilities. Note however that it only predicts a probability (a strength) for the assigned clustered, unlike a Gaussian Mixture Clusterer that predicts probabilities for all clusters. Note also that whereas DBSCAN is computationally efficient and well-suited for moderate-sized datasets, it can face challenges with high-dimensional data due to the curse of dimensionality. HDBSCAN, on the other hand, incorporates optimizations to handle larger datasets and high-dimensional spaces more effectively.

Clustering - HDBSCAN

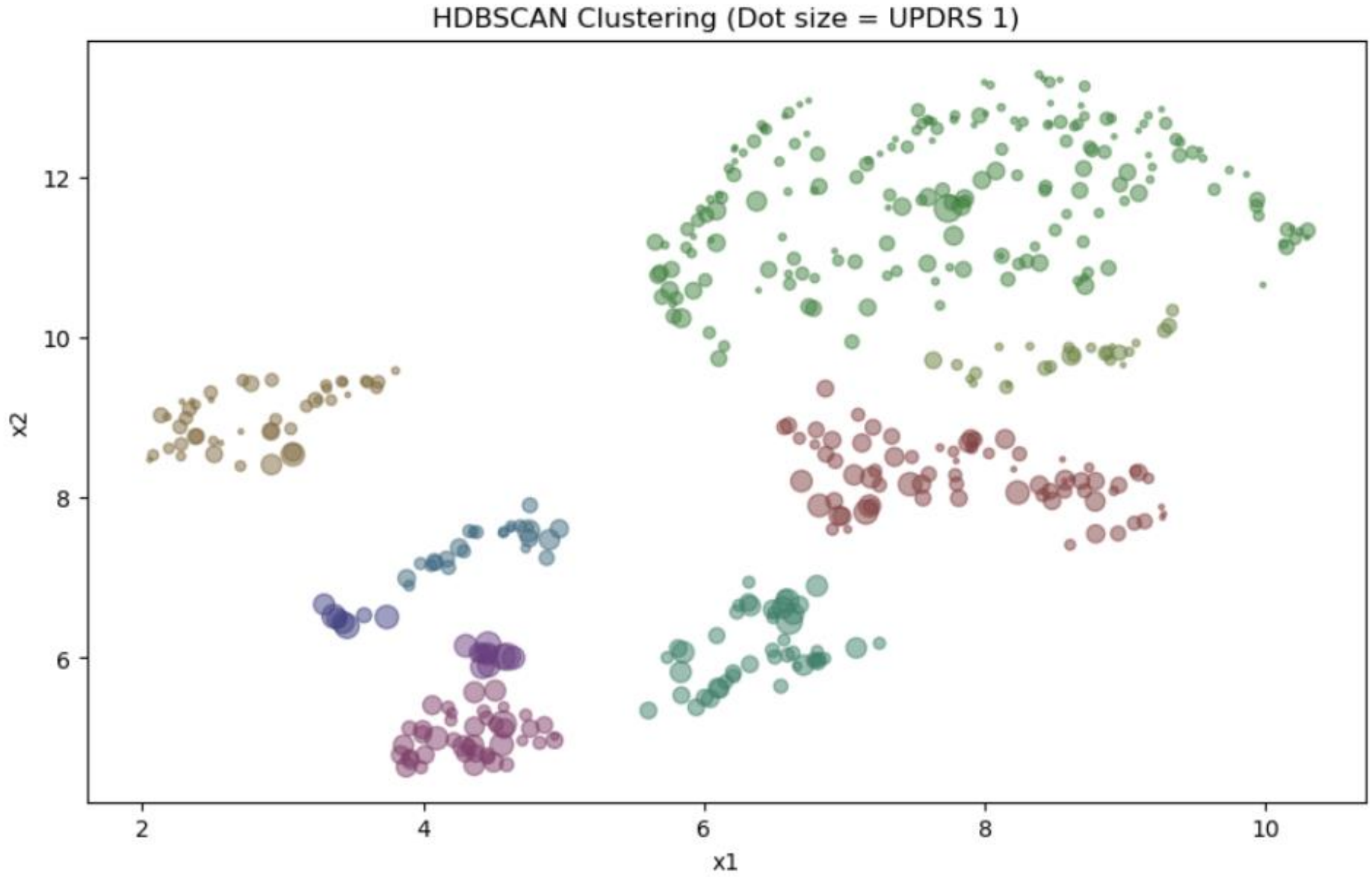


Clustering - HDBSCAN



Clustering - HDBSCAN

SIS 1% +
UMAP 2 components +
HDBSCAN 9 clusters



Data Augmentation

We compute for each data point (for the training sets only) its average frequency. Let $f_{i,u}$ be the sample relative frequency of encountering the value u_i for the UPDRS u. The average frequency is thus simply:

$$f_i = \frac{1}{4} \sum_{u=1}^{4} f_{i,u}$$

This indicator will be used as the probability to generate a *neighbor* of any data point. The probabilities are computed from a linear softmax on the inverse of average frequencies:

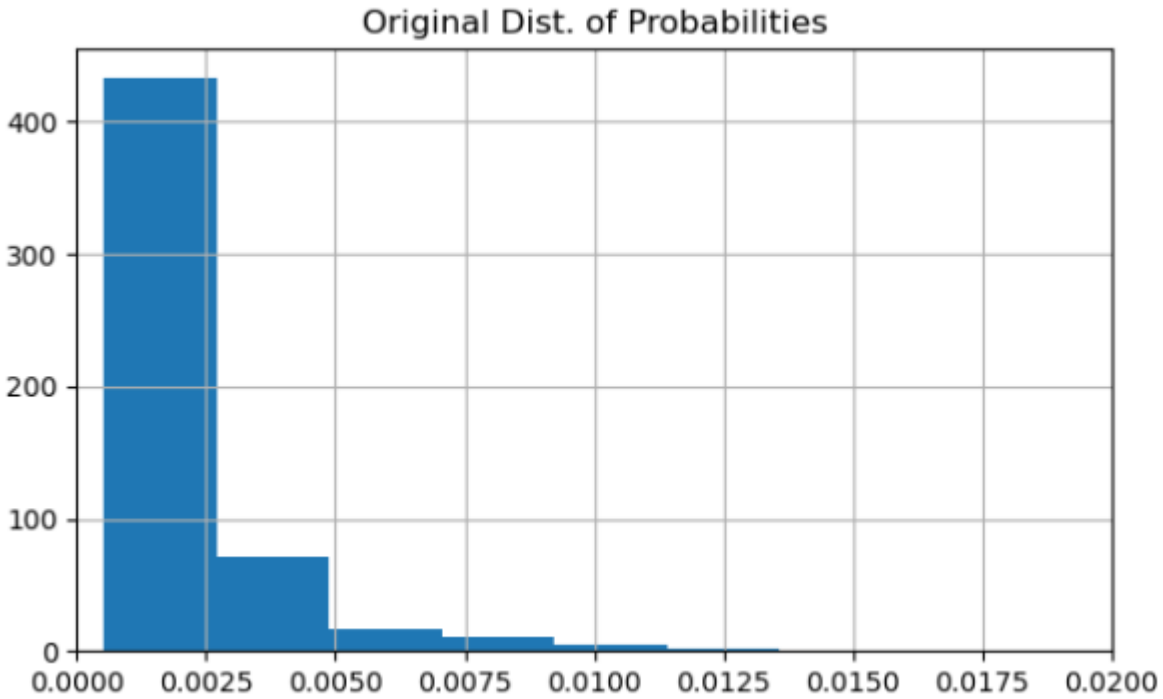
$$p_i = \frac{1/f_i}{\sum_i 1/f_i}$$

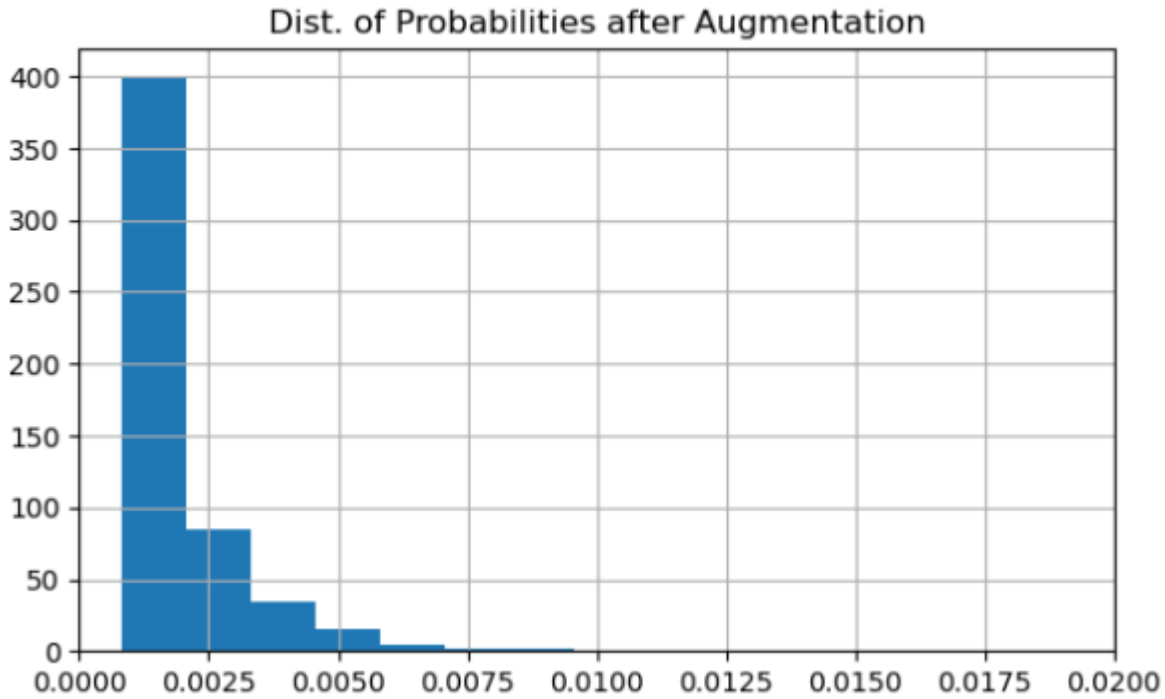
The idea behind is to rebalance the joint distribution of UPDRS scores. For this sake we will need to periodically recompute these frequencies as we generate artficial data points.

Data Augmentation

Below we apply this technique for creating 500 artificial data points. The scores are generated by adding $c \in \{-2, -1, 0, -1, 2\}$ with respective probabilities $p_c \in \{0.05, 0.3, 0.3, 0.3, 0.05\}$ to ever UPDRS score.

This leads to new probabilities more concentrated close to 0, which indicates that we obtain more homogeneous average frequencies. Note that the histograms exclude artificial data since in this case we never generate new cases from artificial data.

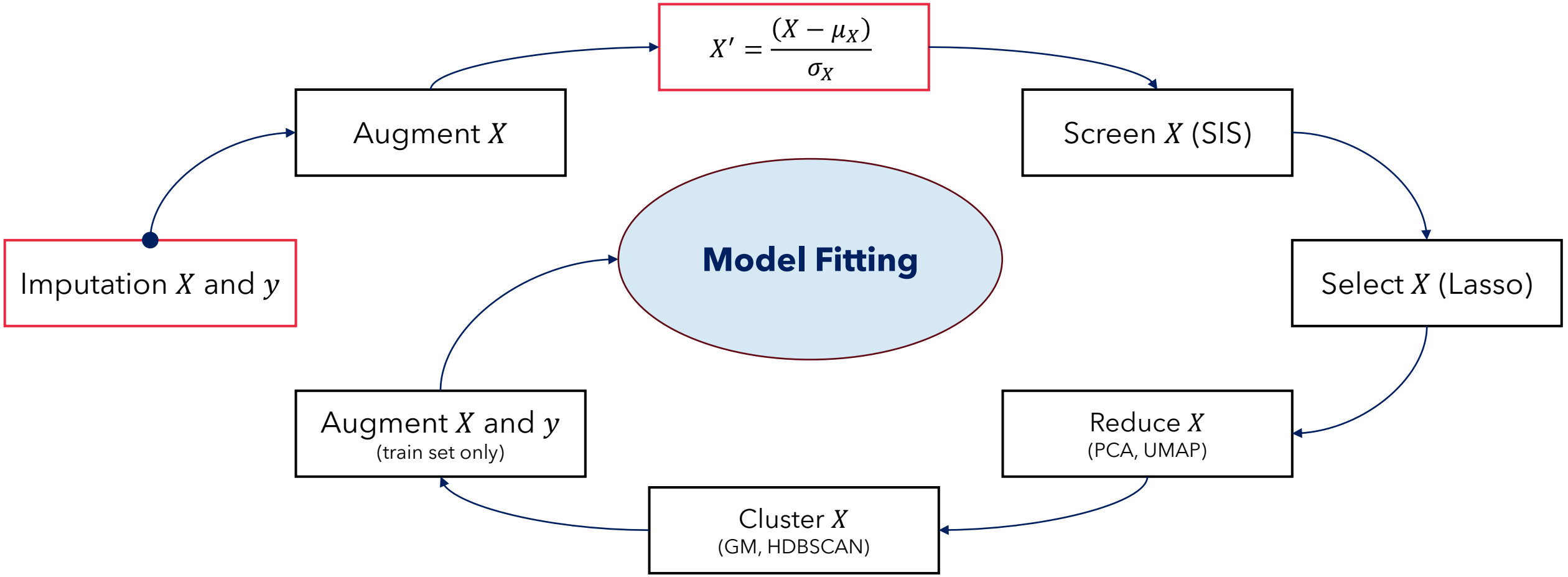




Data Augmentation

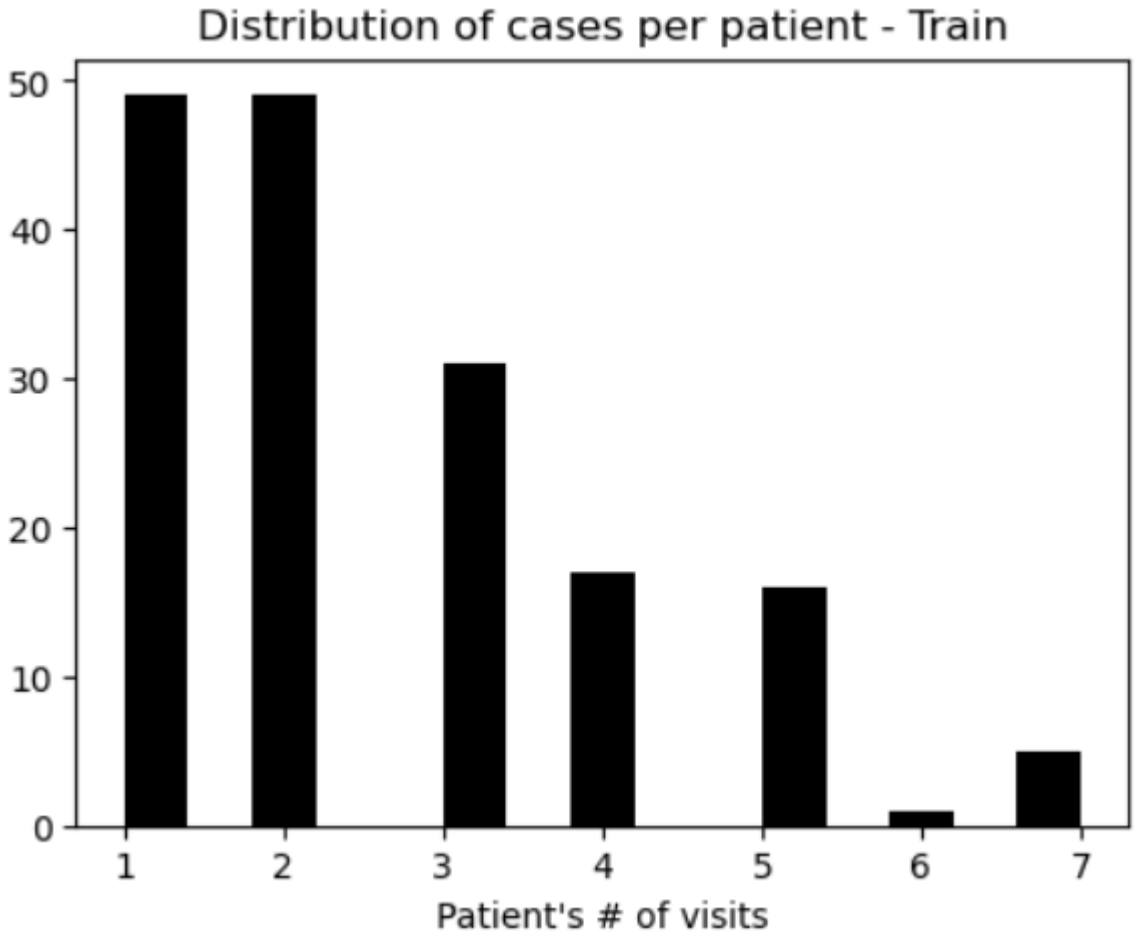
- How many points should we generate?
- Should we add noise to features as well? How much?
- Should we generate from artificial data?

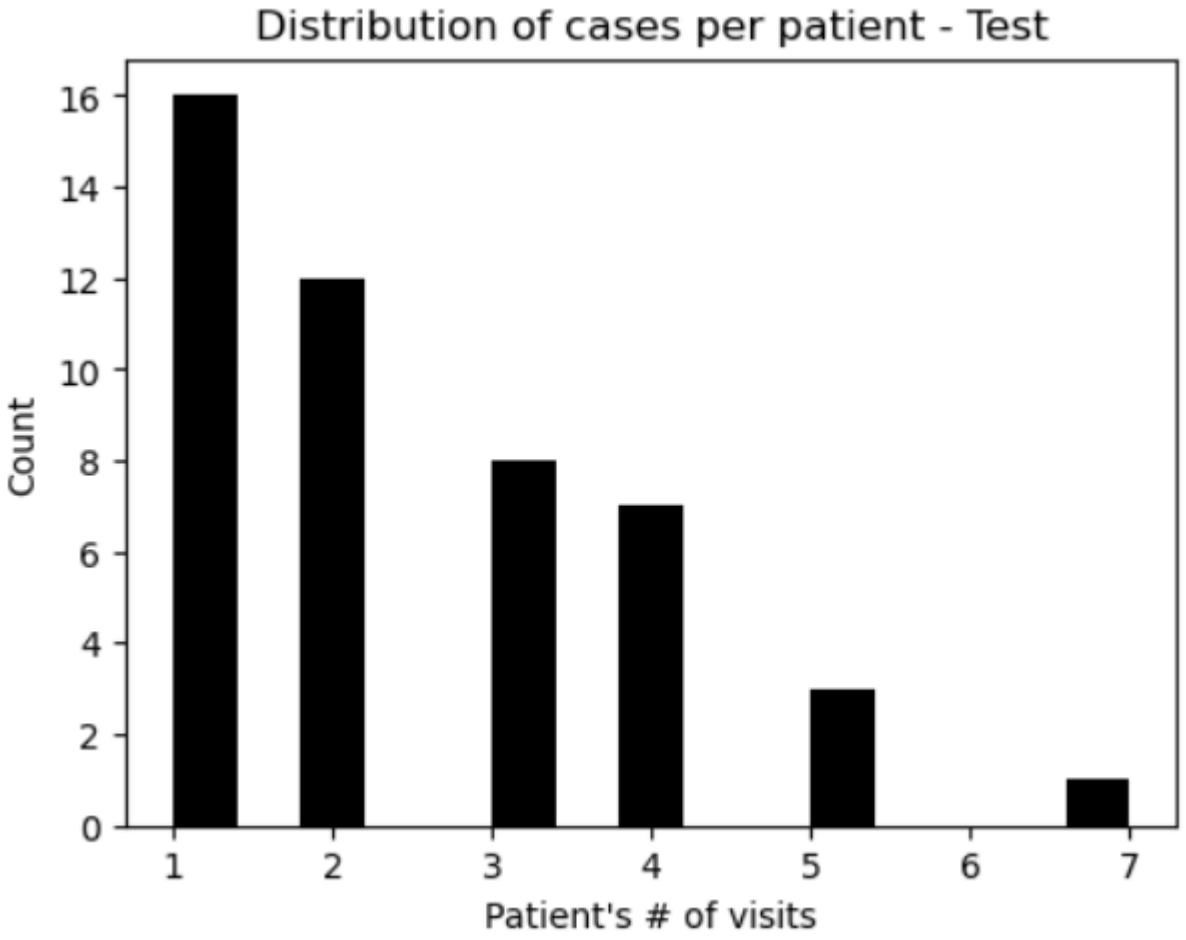
Putting it All Together


$X' = \frac{(X - \mu_X)}{\sigma_X}$
Augment X
Screen X (SIS)
Imputation X and y
Model Fitting
Select X (Lasso)
Augment X and y
(train set only)
Reduce X
(PCA, UMAP)
Cluster X
(GM, HDBSCAN)

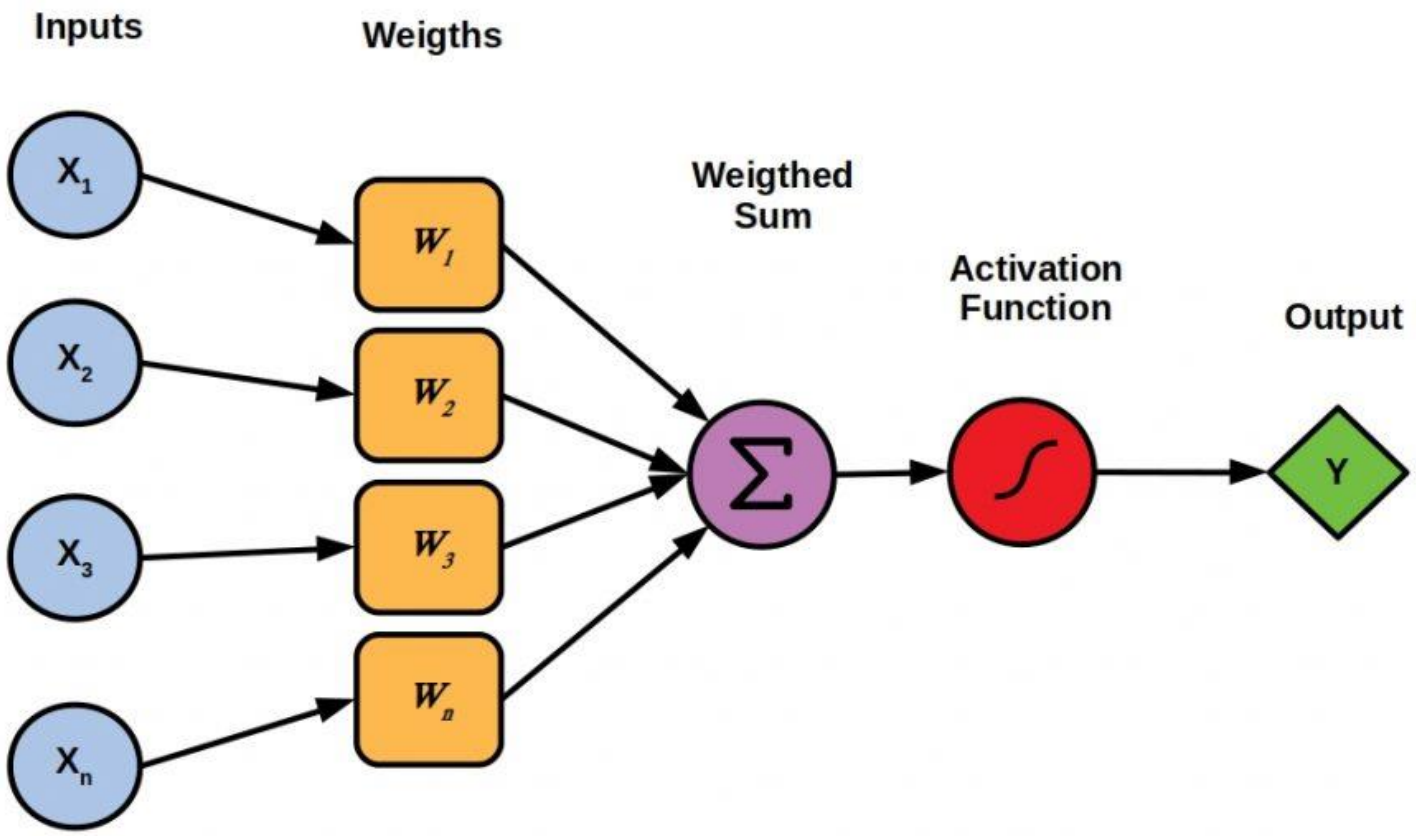
Machine Learning Setup

Train-test Split

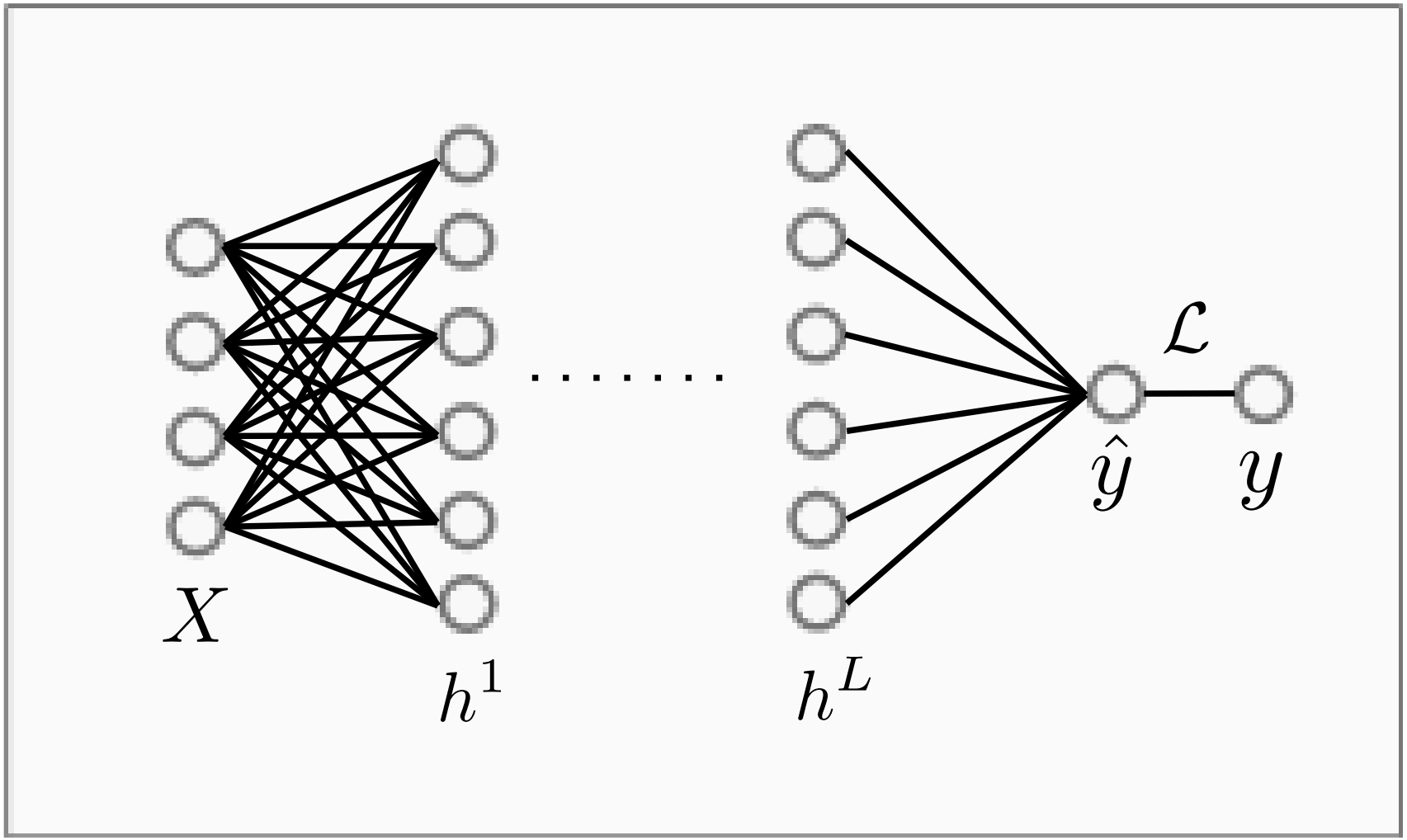




Perceptron


Inputs
Weigths
Weigthed Sum
Activation Function
Output
X1
X2
X3
Xn
W1
W2
W3
Wn
Σ
Y

Multi-layer Perceptron (fully-connected)



Evaluation Metric

Denote by $i \in \mathcal{I} = 1 \dots N$ the individuals (patients) and by $s \in S = 1 \dots 4$ the UPDR scores to forecast. *The following formula applies to the test set and can differ during the training process due to data sparsity.*

$$\text{SMAPE} = \frac{200}{NS} \sum_{i=1}^{N} \sum_{s=1}^{4} \text{SAPE}_{i,s} \quad \text{with} \quad \text{SAPE}_{i,s} = \begin{cases} \frac{|\hat{y}_{i,s} - y_{i,s}|}{|\hat{y}_{i,s}| + |y_{i,s}|} & \text{if } (|\hat{y}_{i,s}|, |y_{i,s}|) \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Hyperparameters

Simplistic set => 716 millions settings

Still 293 millions without redundancy

```
hyperparameters = {
    # ML model
    "lr": [0.1, 0.01, 0.001],
    "dropout": [0, 0.5],
    "RoP": [None, 100, 300],
    # imputation
    "impute_features_meas": ['0', 'mean', 'median'],
    "impute_clin": ['mean', 'median', 'knn'],
    # feature augmentation
    "augment_features_sqrt": [True, False],
    "augment_features_sin": [True, False],
    # feature screening
    "sis_prop": [0.1, 0.25, 0.5, None],
    # feature selection
    "lasso_alpha": [0.1, 1, 10, None],
    "lasso_rescale": [True, False],
    # dimensionality reduction
    "pca_exp_var": [0.9, 0.99, None],
    "umap_prop": [0.1, 0.25, 0.5, None],
    "umap_n_neighbors": [5, 15],
    "umap_distance": ["euclidean", "linfinity", "canberra"],
    # clustering
    "gm_n_clusters": [2, 4, 10, 20, None],
    "hdbscan_min_cluster_size": [5, 20, 100, None],
    # data augmentation
    "n_artificial": [100, 500, 1000, None],
    "gen_from_artificial": [True, False],
    "noise_X": [0, 0.01, 0.1]
}
```

ML Experiments

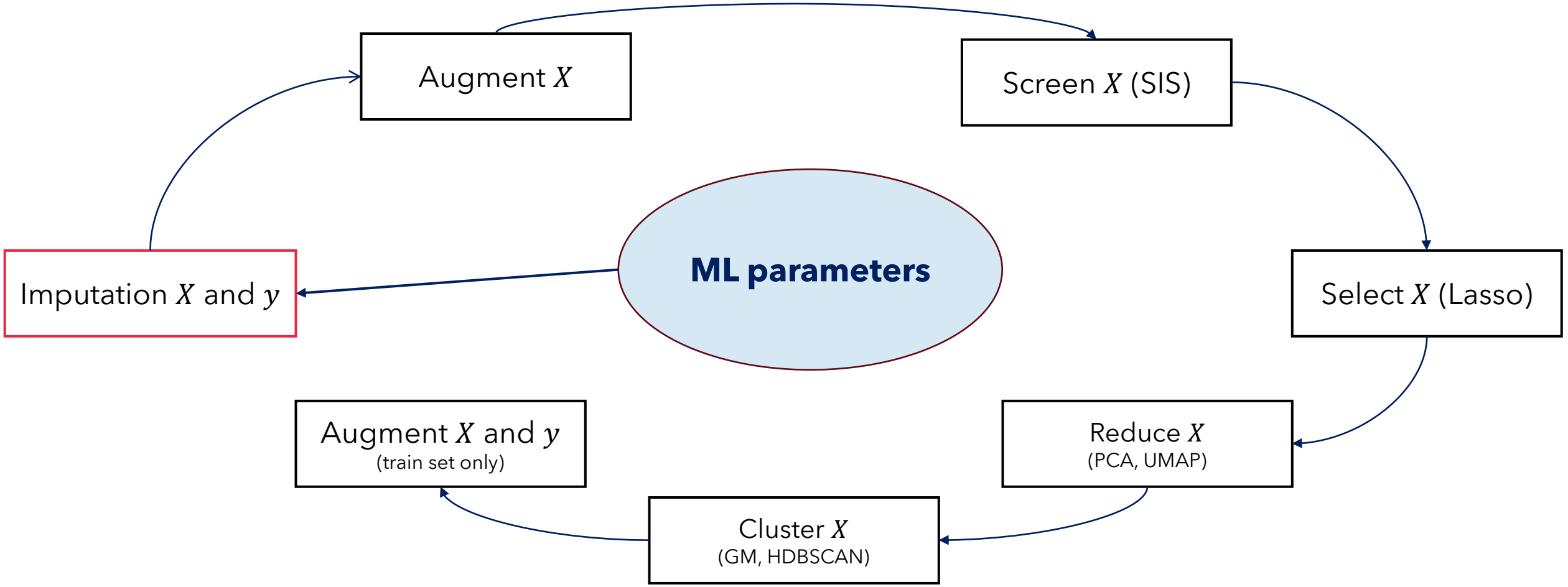
Initial Setup

```
params = {
    # ML model
    "lr": 0.1, "dropout": 0,  "RoP": None,
    # imputation
    "impute_features_meas": '0', "impute_clin": 'mean',
    # feature augmentation
    "augment_features_sqrt": False, "augment_features_sin": False,
    # feature screening
    "sis_prop": 0.1,
    # feature selection
    "lasso_alpha": None, "lasso_rescale": False,
    # dimensionality reduction
    "pca_exp_var": 0.95, "umap_prop": None, "umap_n_neighbors": 15, "umap_distance": 'euclidean',
    # clustering
    "gm_n_clusters": None, "hdbscan_min_cluster_size": None,
    # data augmentation
    "n_artificial": None, "gen_from_artificial": False, "noise_X": 0
}
```

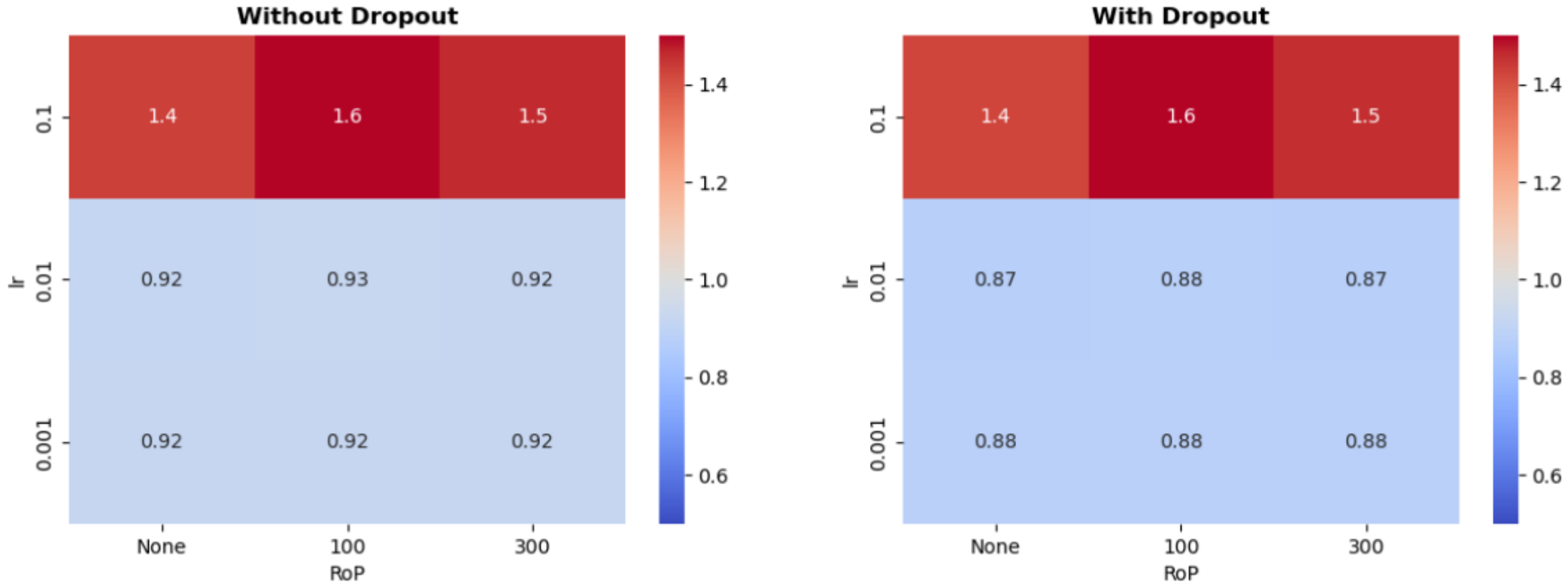
Search Strategy

Quick and Dirty…

Search Strategy

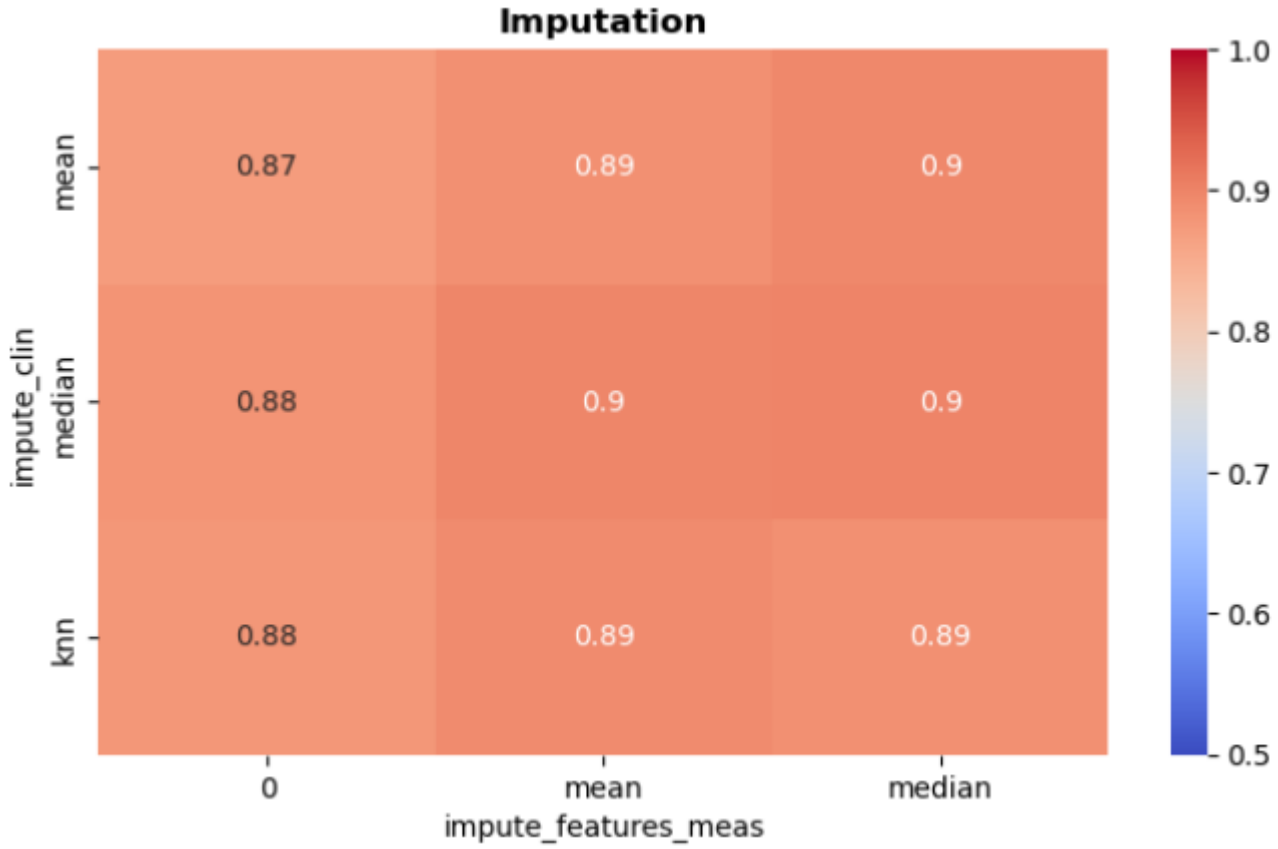


Model Hyperparameters



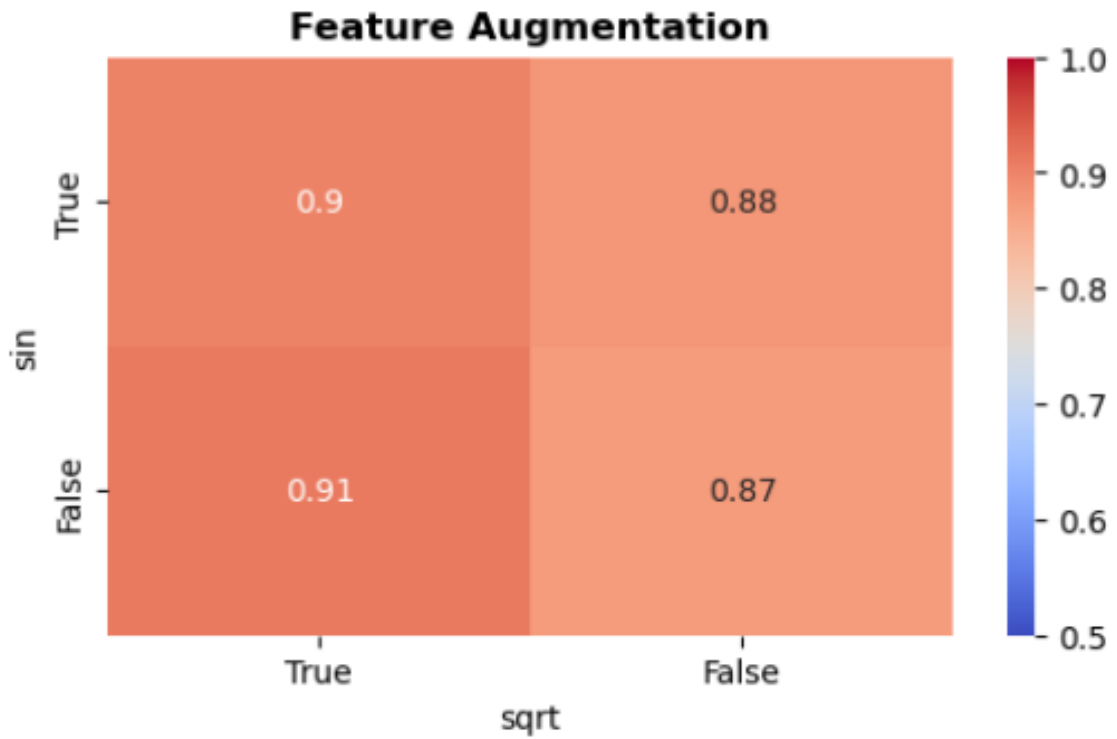

```
params['lr'] = 0.01
params['dropout'] = 0.5
params['RoP'] = None
```

Imputation Hyperparameters



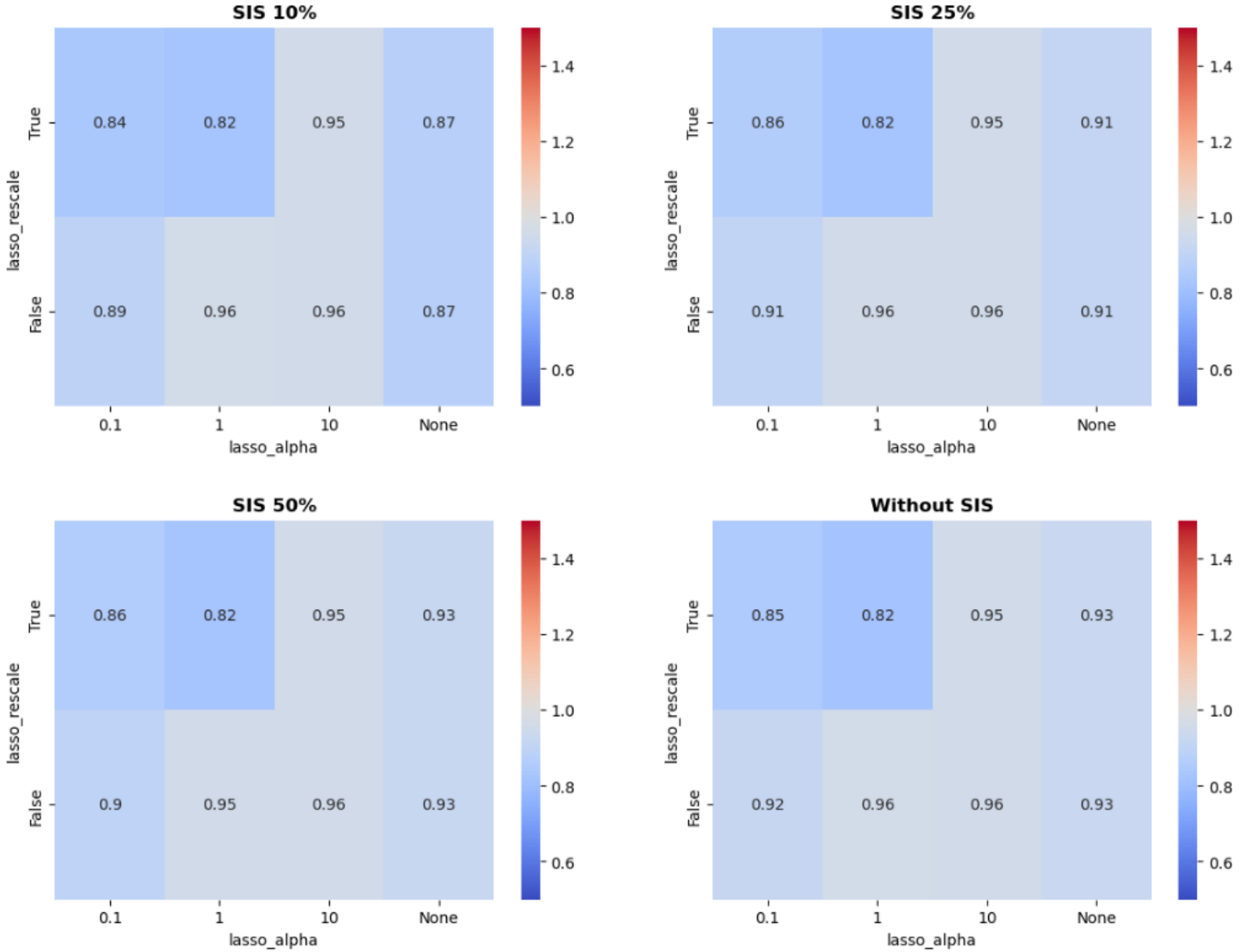

```
params['impute_features_meas'] = "0"
params['impute_clin'] = "mean"
```

X Augmentation Hyperparameters




```
params['augment_features_sqrt'] = False
params['augment_features_sin'] = False
```

SIS / Lasso Hyperparameters




```
params['sis_prop'] = None
params['lasso_alpha'] = 0.5
params['lasso_rescale'] = True
```

SMAPE = 0.822

Other Hyperparameters

I was lazy to make plots…

Other Hyperparameters

Dimension Reduction	**No change**	**0.822 => 0.822**
Clustering	**No change**	**0.822 => 0.822**

Data Augmentation	`params["n_artificial"] = 100`
	`params["noise_X"] = 0.05`
	`params["gen_from_artificial"] = True`
		0.822 => 0.815

Testing

Without preprocessing	1.021
With preprocessing	0.898

Testing

Without preprocessing	1.021
With preprocessing	0.898
Naive Model	0.871

References

Data

•AMP®-Parkinson's Disease Progression Prediction

Academic

•Holden, Finseth, Sillau and Berman (2018) "Progression of MDS-UPDRS Scores Over Five Years in De Novo Parkinson Disease from the Parkinson's Progression Markers Initiative Cohort", *Movement Disorders*
•Shi *et al.* (2015) "Cerebrospinal Fluid Peptides as Potential Parkinson Disease Biomarkers: A Staged Pipeline for Discovery and Validation", *Molecular and Cellular Proteomics*
•Fan and Lv (2008) "Sure independence screening for ultrahigh dimensional feature space", *Journal of the Royal Statistical Society*
•Pearson (1901) "On lines and planes of closest fit to systems of points in space" *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*
•Hastie, Tibshirani and Friedman (2001/2009) "The Elements of Statistical Learning Data Mining, Inference, and Prediction, Second Edition", *Springer*
•McInnes, Healy and Melville (2018) "UMAP: Uniform Manifold Approximation and Projection", *The Journal of Open Source Software*
•Hinton and Roweis (2002) "Stochastic Neighbor Embedding", *NeurIPS Proceedings*
•Van der Maaten and Hinton (2008) "Visualizing Data using t-SNE", *Journal of Machine Learning Research*
•Campello, Moulavi and Sander (2013) "Density-Based Clustering Based on Hierarchical Density Estimates", *Pacific-Asia Conference on Knowledge Discovery and Data Mining*
•Malzer and Baum (2020) "A Hybrid Approach To Hierarchical Density-based Cluster Selection", *2020 IEEE international conference on multisensor fusion and integration for intelligent systems*

Others

•MDS website
•Bottom Up Proteomics
•Progenesis Technical Note
•Amino acid table
•t-SNE
•Coenen and Pearce - Understanding UMAP
•McInnes - UMAP package
•Andrews - Dimension Reduction Handouts
•John Healy - HDBSCAN, Fast Density Based Clustering, the How and the Why