Machine Learning in MX

Melanie Vollmar

Diamond Light Source

UK

Overview

- ML definition
- Supervised vs unsupervised
- Classification vs regression
- Missing data and standardization
- Data splitting
- Exploratory data analysis (EDA)
- Finding parameters
- Classifiers/Estimators
- Classifier assessment
- Prediction assessment
- Tutorial data

ML definition

- Part of the field of Artificial Intelligence
- Pattern recognition
- Features learned from training data and identified in new data

Supervised vs unsupervised

- Supervised

  training samples have a known solution, e.g. which class they belong to

- Unsupervised

  training samples have no known solution and the algorithm identifies how to group samples, e.g. clustering

- Semi-supervised learning, reinforcement learning

Classification vs regression

- Classification

  predicting/recognising two or more classes

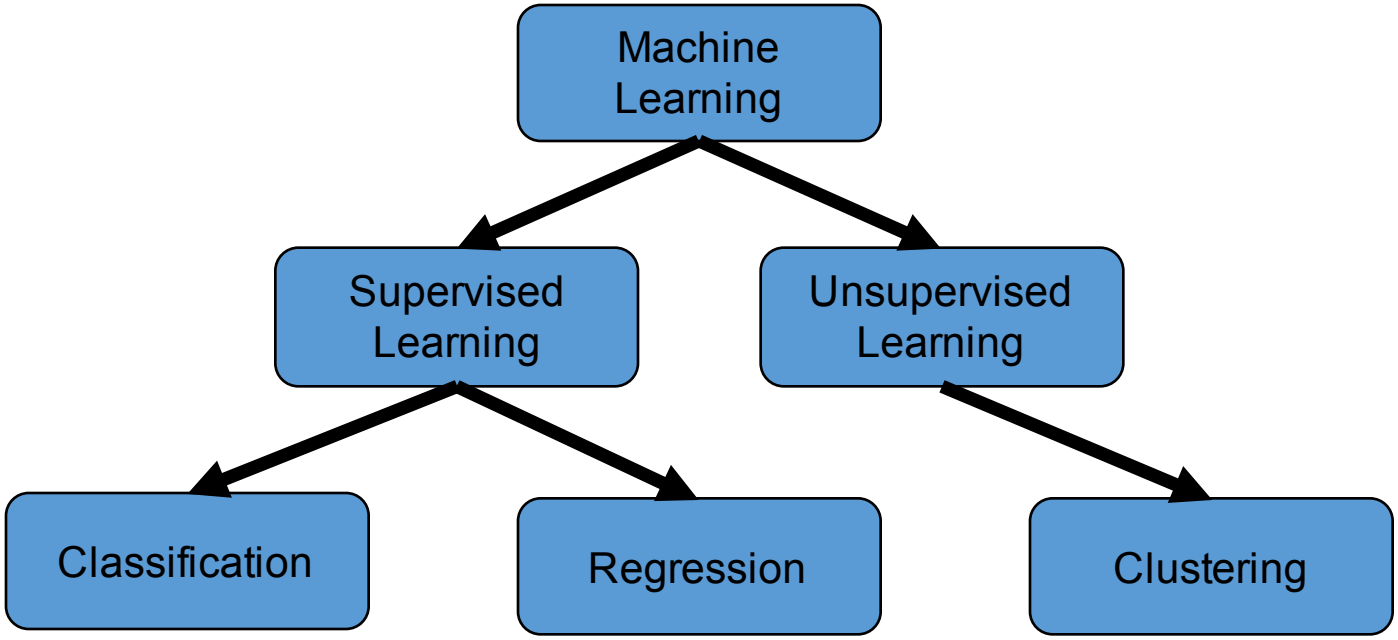
  a cat, a dog, a house

- Regression

  predicting a value

  house price, market value at stock exchange

Classification vs regression
and
Supervised vs unsupervised



Missing data and standardization

- Replace missing values by mean or 0
- Some algorithms, e.g. SVMs, require this step;

Data splitting

- Training and parameter search on “training set” which represents the larger part of the data overall
- Testing and assessing the trained model with “testing set” usually 20-30% of the data
- Optionally a “calibration set” of 5% of the data
- Cross-validation → often used with training set
- Stratification → to ensure class distribution in imbalanced classes is maintained (ideally classes are equally present)
- Class weights

EDA

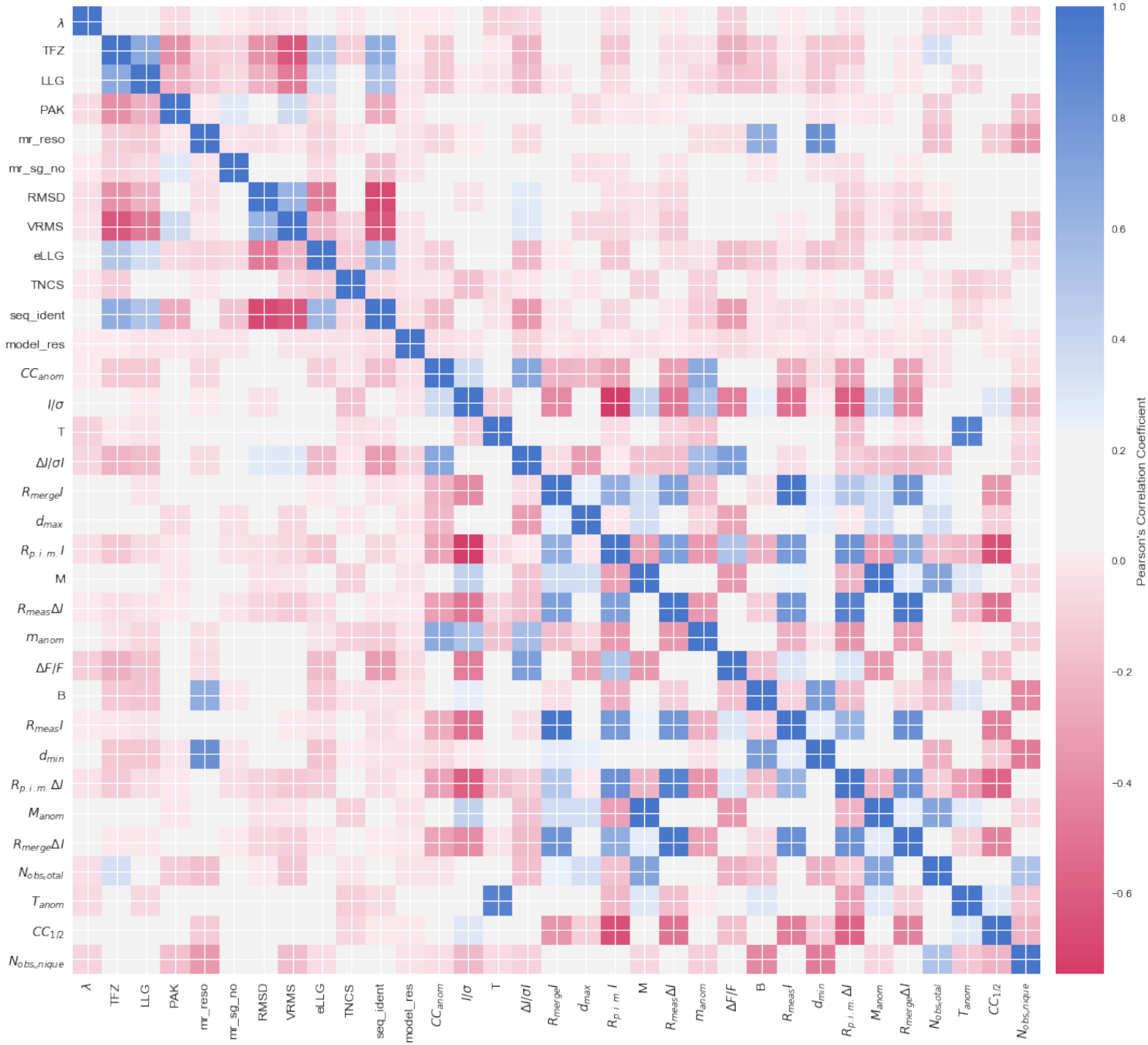
Data plotting

- Pearson's correlation coefficient
- Scatter plot matrix
- Histogram for each feature
- PMF(Probability Mass Function), PDF(Probability Density Function) and CDF(Cumulative Distribution Function) for data continuity
- Empirical cumulative distribution function

EDA

Data plotting

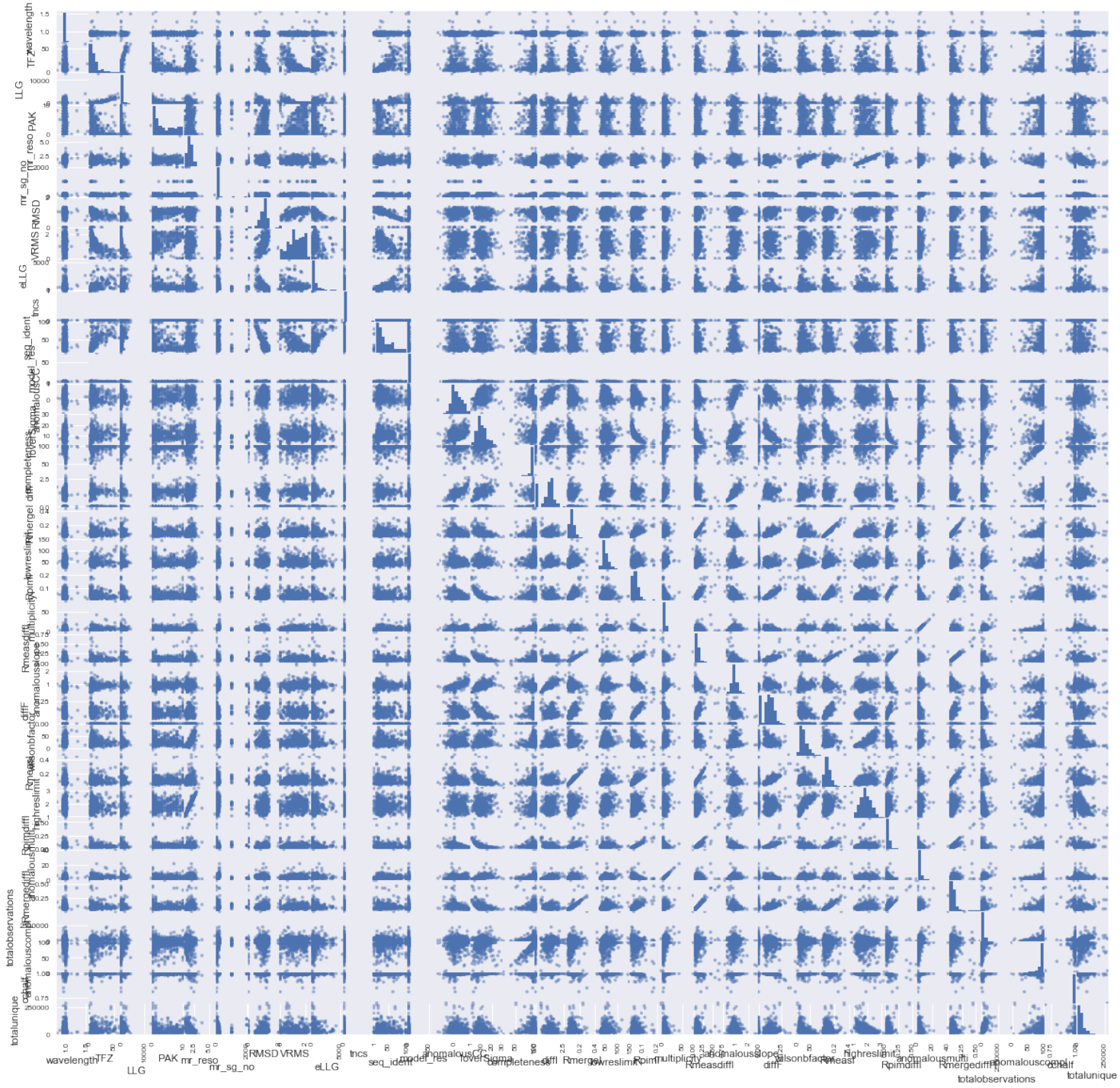
- Pearson's correlation coefficient



EDA

Data plotting

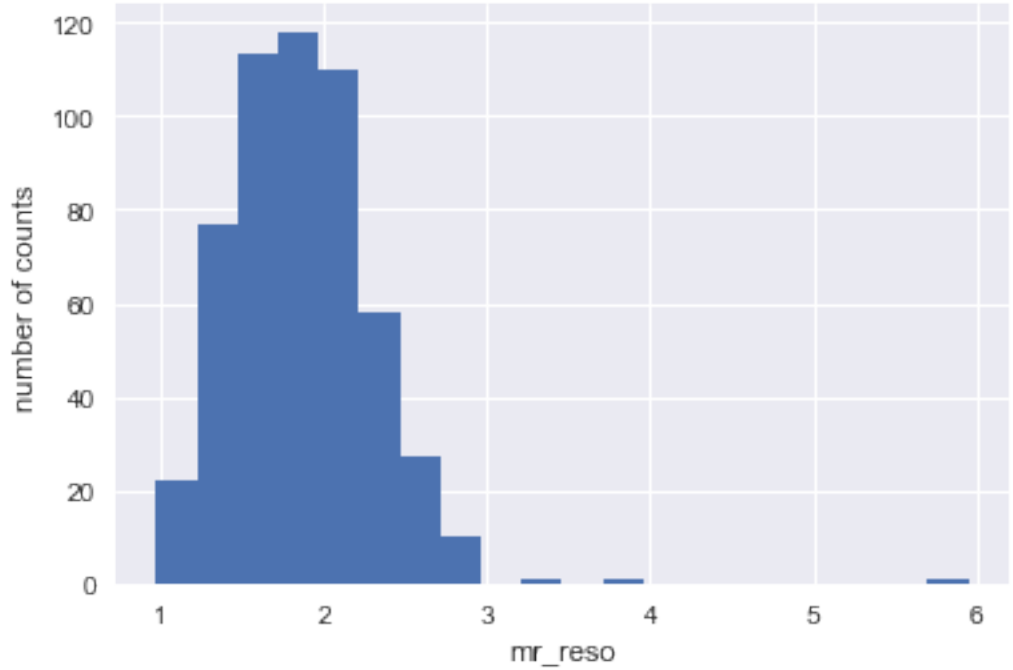
- Scatter plot matrix



EDA

Data plotting

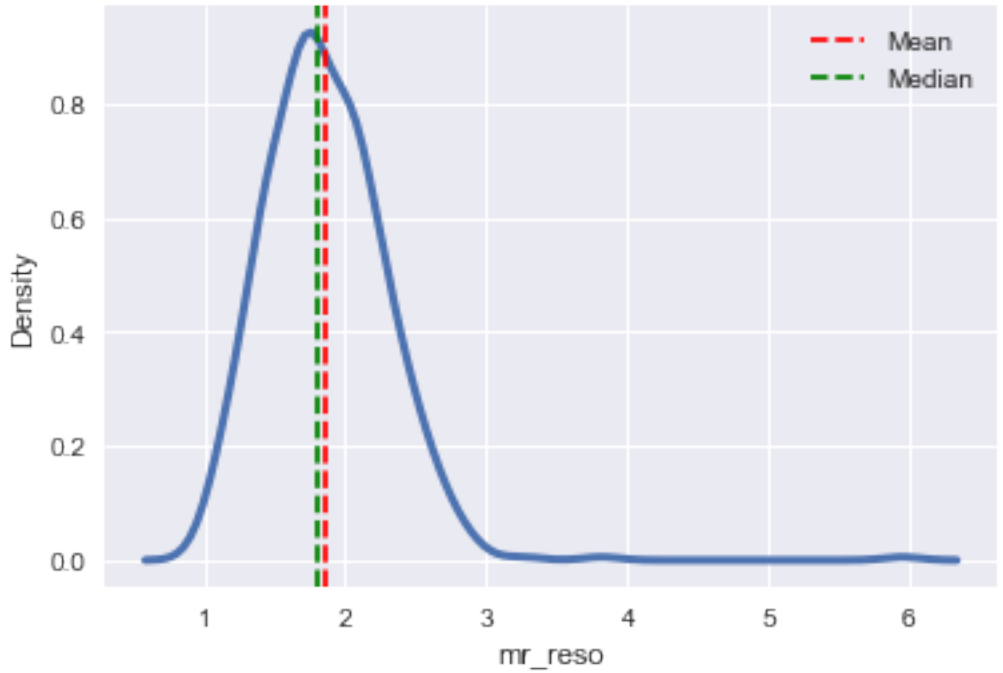
- Histogram for each feature



EDA

Data plotting

- PMF(Probability Mass Function), PDF(Probability Density Function) and CDF(Cumulative Distribution Function) for data continuity



PMF → how likely is each value for variable X

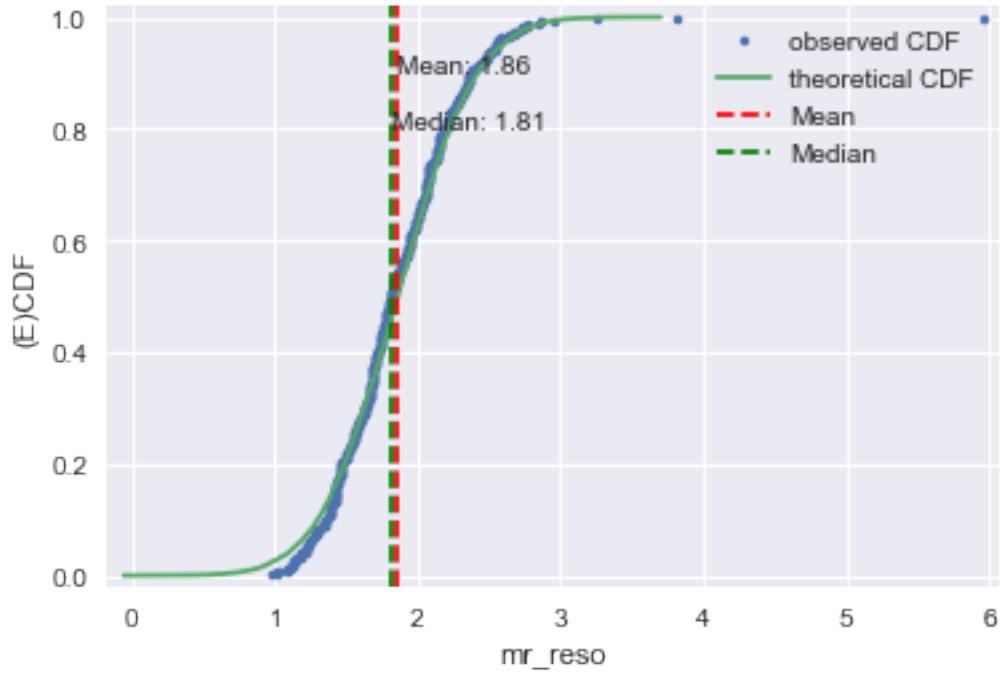
PDF → to describe continuous data, e.g. Gaussian

CDF → summed probabilities for all values

EDA

Data plotting

- Empirical cumulative distribution function



EDA

- Curse of dimensionality
  - More dimensions/features in ratio to number of samples available → data becomes sparse
  - Results become unreliable
  - Classifiers become unstable
  - Solution
    - Dimensionality reduction, e.g. principal component analysis (PCA)
    - Recursive feature elimination

Finding parameters

- Manually choosing settings
- Grid search over a parameter range
- Randomized search of parameter combinations chosen randomly

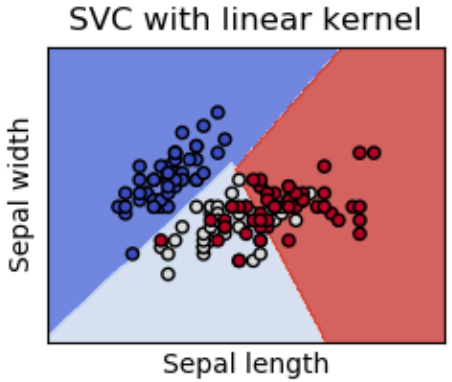
Classifiers/Estimators

- Support vector machine with linear kernel
- Support vector machine with radial-base function kernel
- Decision tree
- Decision tree with Bagging
- Decision tree with AdaBoost
- Random forest
- Extreme random forest
- K nearest neighbor

Classifiers/Estimators

Support vector machine with linear kernel

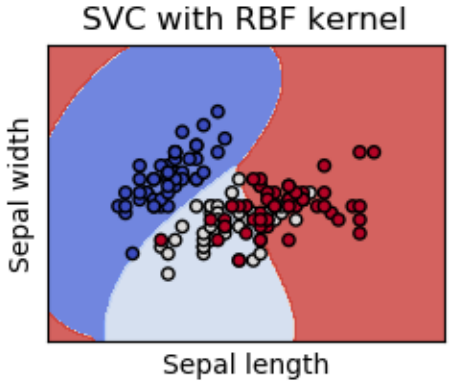
- Separate samples by linear decision boundary
- Samples closest to the boundaries define the boundaries → memory efficient
- Maximum separation between classes/groups of samples has to be achieved
- Works well with high-dimensional data, i.e. large number of features compared to number of samples
- Very large number of features → prone to overfitting
- Getting probability estimates is computationally expensive
- Data needs to be scaled/standardized (same scaling for training and test data)



Classifiers/Estimators

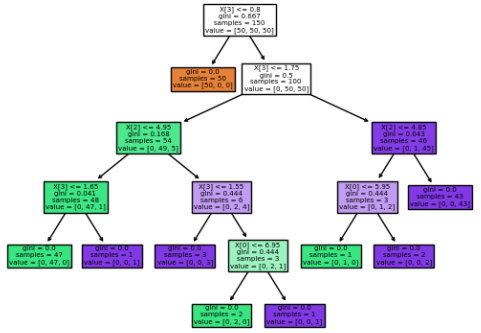
Support vector machine with radial-base function kernel

- Separate samples by non-linear decision boundary
- Samples closest to the boundaries define the boundaries → memory efficient
- Maximum separation between classes/groups of samples has to be achieved
- Works well with high-dimensional data, i.e. large number of features compared to number of samples
- Very large number of features → prone to overfitting
- Getting probability estimates is computationally expensive
- Data needs to be scaled/standardized (same scaling for training and test data)



Classifiers/Estimators

Decision tree



- Simple to understand and to interpret; human readable
- Requires little data preparation; no support for missing values
- Able to handle both numerical and categorical data
- Able to handle multi-output problems
- white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic; black box model (e.g. in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests → reliability of the model
- Overfitting by over-complex trees that do not generalise the data well
- Decision trees instability → ensemble
- Decision tree learners create biased trees for dominant classes

Classifiers/Estimators

Decision tree with Bagging

- Contains several instances of a black-box estimator on random subsets of the original training set
- Aggregation of their individual predictions to form a final prediction
- Reduce the variance of a base estimator (e.g. a decision tree), by introducing
- reduce overfitting
- work best with strong and complex models (e.g. fully developed decision trees), in contrast with boosting methods which usually work best with weak models (e.g. shallow decision trees)
- Here: we use replacement

Classifiers/Estimators

Decision tree with AdaBoost

- sequence of weak learners (i.e. models that are only slightly better than random guessing, such as small decision trees)
- Used until all samples
- applying weights after each round to incorrectly predicted samples
- Continue until all samples classified
- Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence

Classifiers/Estimators

Random forest

- each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set
- splitting each node either from all input features or a random subset of size max_features
- Two sources of randomness → decrease the variance of the forest estimator
- taking an average of those predictions → reduced variance, sometimes at the cost of a slight increase in bias
- averaging their probabilistic prediction, instead of letting each classifier vote for a single class.

Classifiers/Estimators

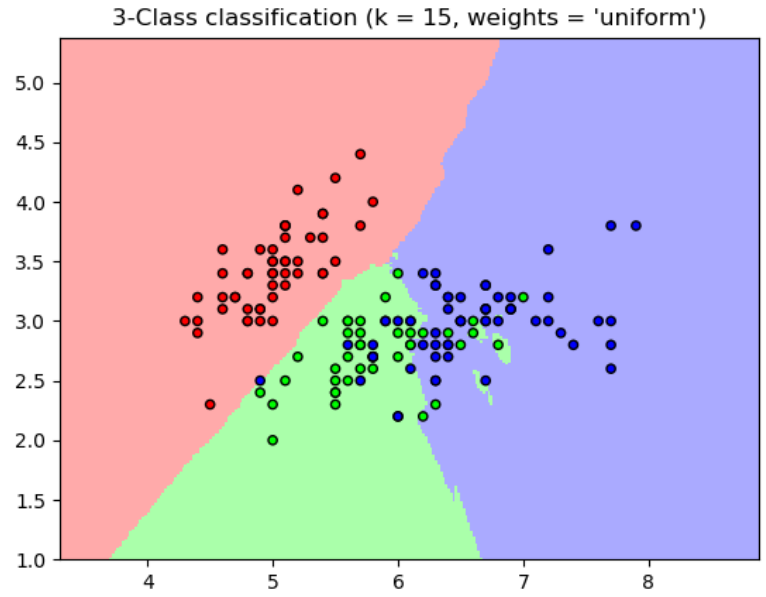
Extreme random forest

- More randomness in the way splits are computed
- Random subset of candidate features is used
- Discriminative thresholds are drawn at random for each candidate
- reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias

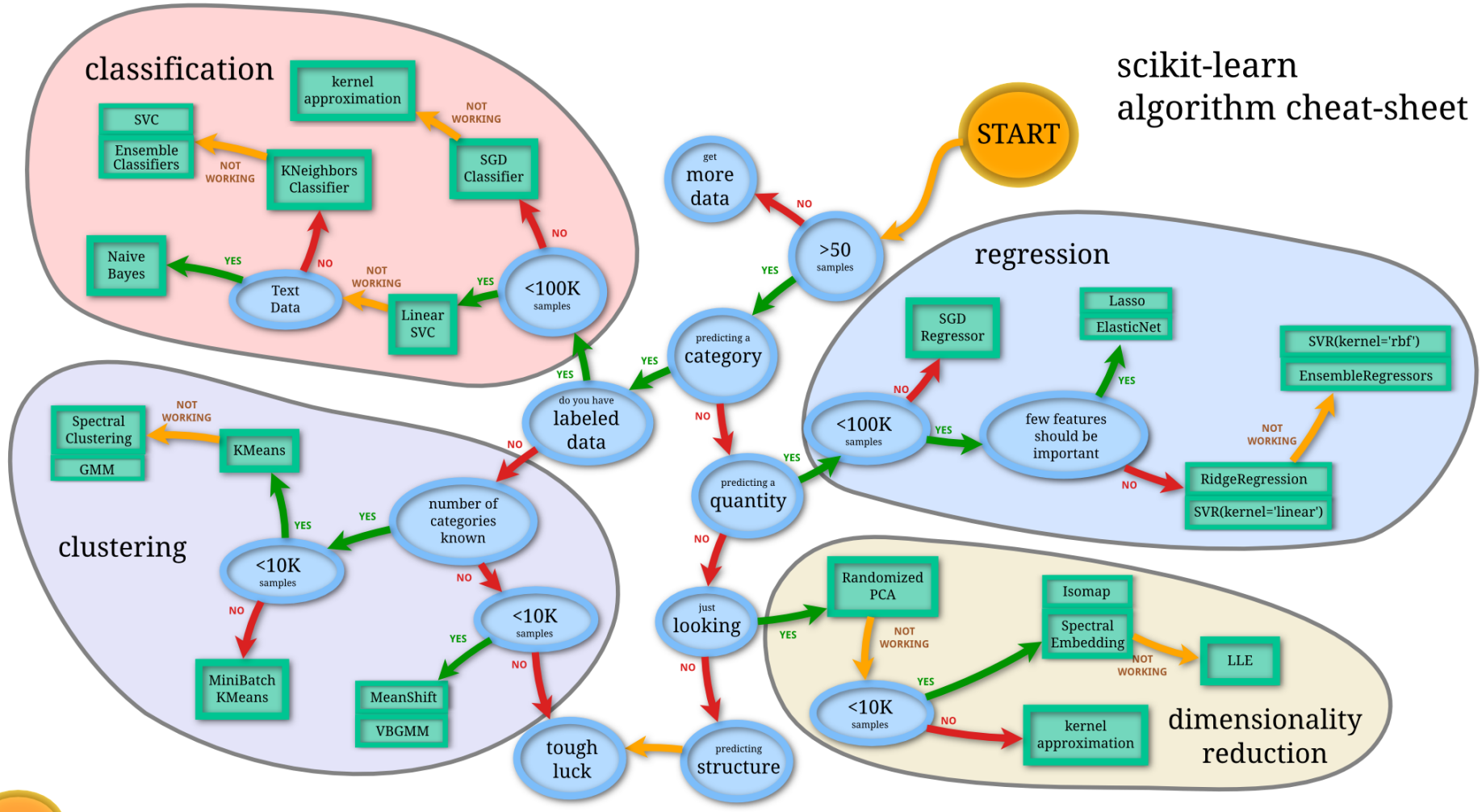
Classifiers/Estimators

K nearest neighbor

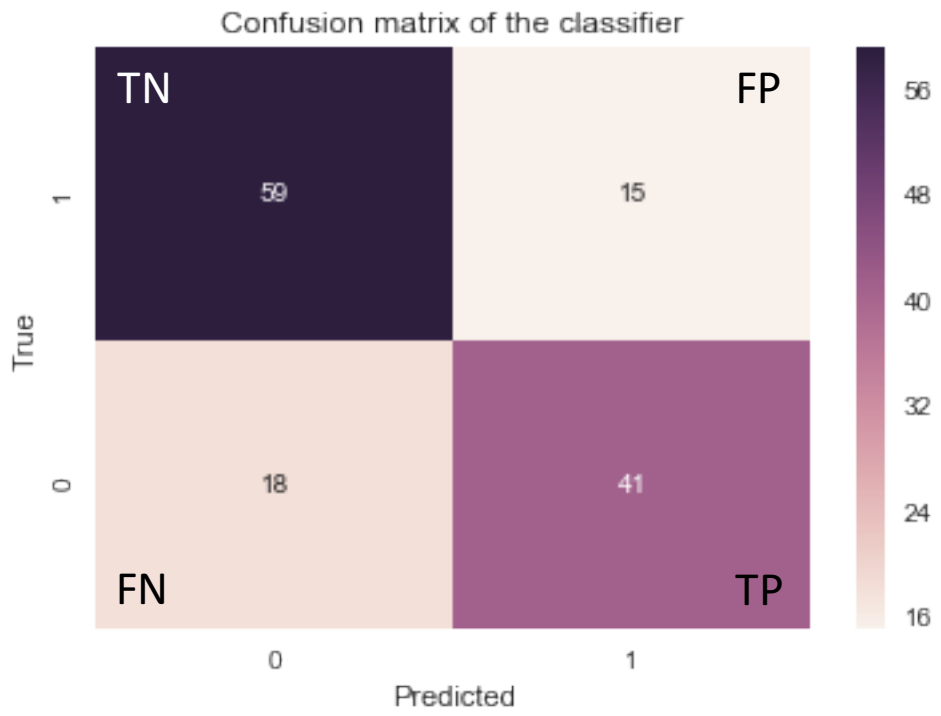
- Neighbors-based classification
- *instance-based learning* or *non-generalizing learning* → does not learn a general rule to identify the classes
- Classification is computed using simple majority → a query point is assigned the data class which has the most representatives within the nearest neighbors of the point



Which estimator to choose?



Classifier assessment

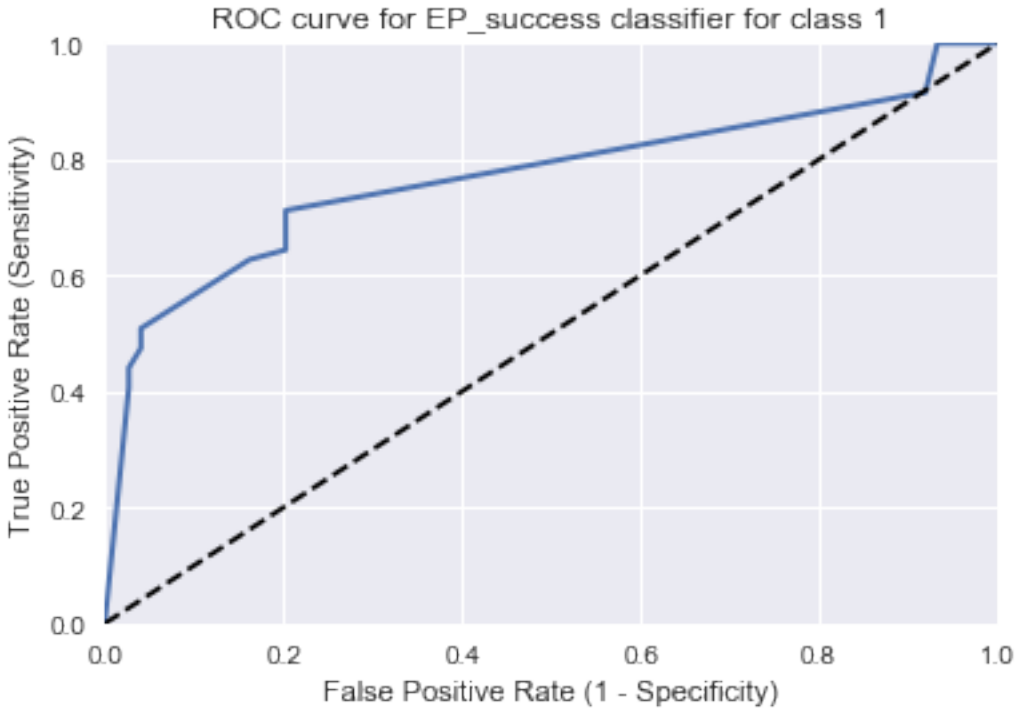


Classifier assessment

- $Classification\ accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- $Classification\ error = \frac{FP+FN}{TP+TN+FP+FN}$
- $Sensitivity\ or\ TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$
- $Specificity\ or\ TNR = \frac{TN}{N} = \frac{TN}{TN+FP} = 1 - FPR$
- $False-positive\ rate\ or\ FPR = \frac{FP}{N} = \frac{FP}{FP+TN} = 1 - TNR$
- $Precision\ or\ PPV = \frac{TP}{TP+FP}$
- $F1\ score = \frac{2}{\frac{1}{precision} + \frac{1}{sensitivity}} = \frac{TP}{TP + \frac{FN+FP}{2}}$

Classifier assessment

- Receiver-operator curve and area-under-the-curve



AUC for test set class 1: 0.769697663765

Prediction assessment

- Predict() → gives the class/score for the prediction
- Predict_proba() gives the probability for the class
- Adjusting prediction thresholds

Tutorial data

JCSG → 507 structures
SGC → 303 structures

Phasing method:
S/MAD → 446
Native → 364

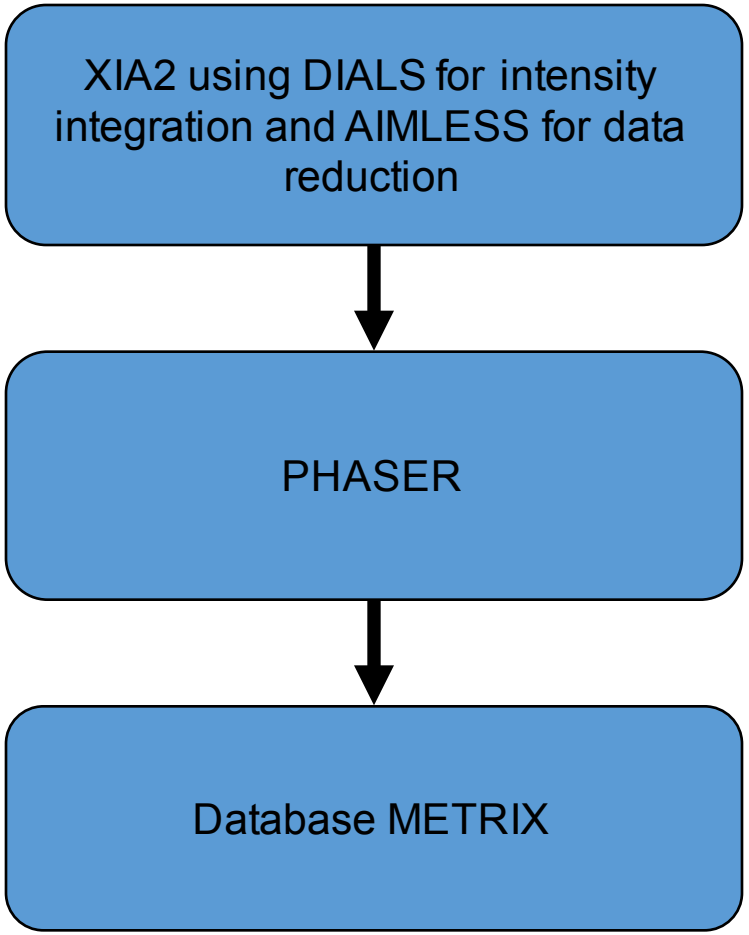
Training data:
Samples → 664

Resolution range:
1.05 – 3.8Å

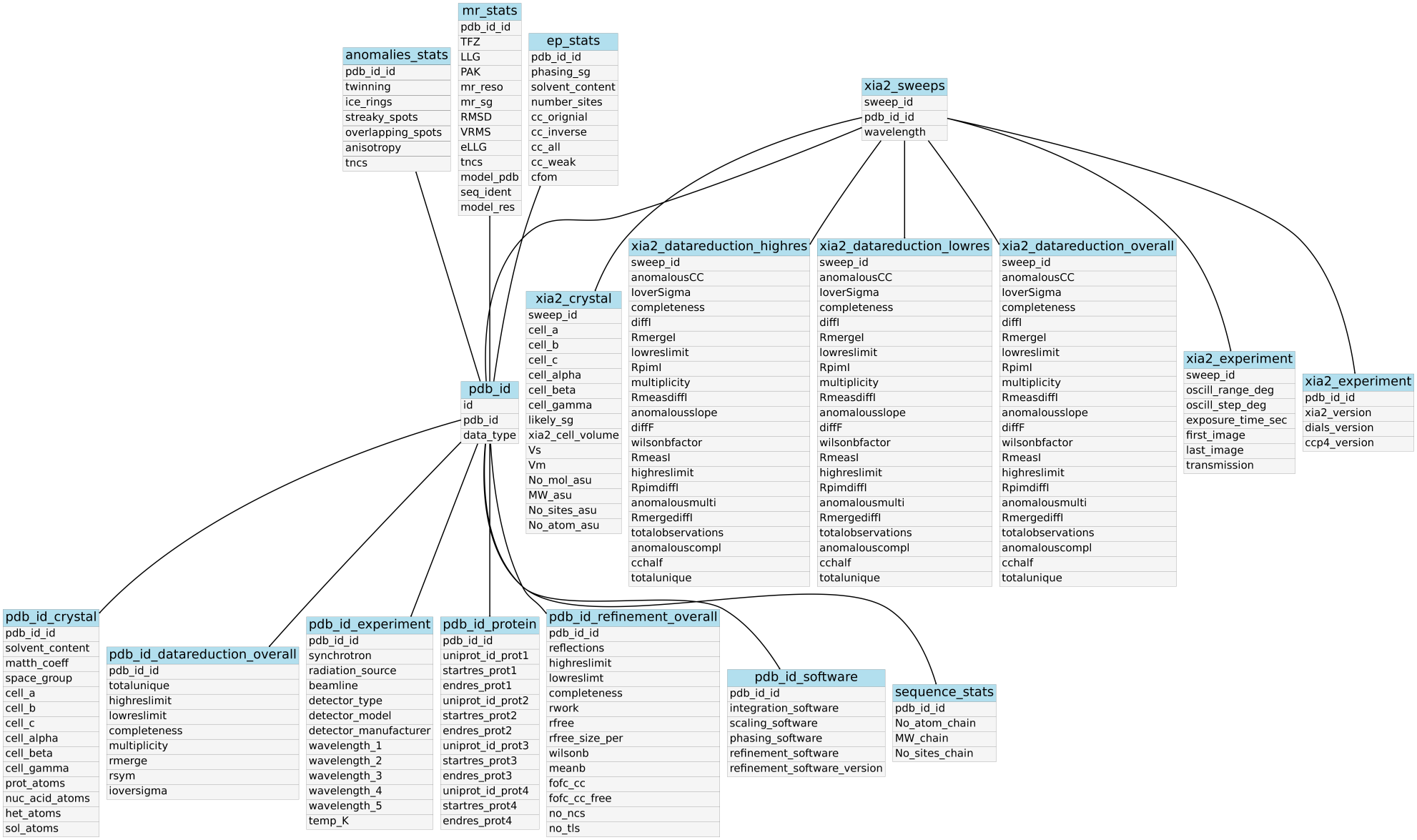
Detector type:
CCD, PAD

X-Ray source:
Synchrotron, in-house

Protein:
6 – 100kDa



Tutorial data



Tutorial data

	sweep_id	pdb_id_id	wavelength	MR_success	TFZ	LLG	PAK	mr_reso	\
0	144	1003	0.9797	0.0	5.8	33.0	5.0	1.38	
1	145	1003	0.9184	0.0	5.8	33.0	5.0	1.38	
2	146	1003	0.9792	0.0	5.8	33.0	5.0	1.38	
3	918	101	0.9793	0.0	4.1	17.0	1.0	1.66	
4	919	101	0.9537	0.0	4.1	17.0	1.0	1.66	

	mr_sg	mr_sg_no	...	wilsonbfactor	RmeasI	highreslimit	\
0	P1211	4	...	11.28	0.0723	1.38	
1	P1211	4	...	11.30	0.0846	1.41	
2	P1211	4	...	11.21	0.0879	1.39	
3	H3	146	...	20.88	0.0910	1.66	
4	H3	146	...	20.26	0.0937	1.67	

	RpimdiffI	anomalousmulti	RmergediffI	totalobservations	anomalouscompl	\
0	0.0442	1.73	0.0477	290215	91.57	
1	0.0475	1.75	0.0512	275712	92.33	
2	0.0412	1.73	0.0444	284616	91.64	
3	0.0607	1.88	0.0644	206737	95.25	
4	0.0624	1.91	0.0660	208630	97.15	

	cchalf	totalunique
0	0.9983	87299
1	0.9976	82033
2	0.9972	85517
3	0.9972	56237
4	0.9973	55392

Tutorial data

cd home

git clone https://github.com/ecacomsig/Melk-2019.git

conda activate pyxem

conda install seaborn