Vector derivatives

Topics : Anamoly / Novelty detection

"Known"

1. Using existing techniques ✓
2. Elliptical- envolope → MV · Gaussian
3. Isolation Forest → DT & RF
4. One- class SVMs → SVMs
5. Local outlier factor
   ↳ KNN & density

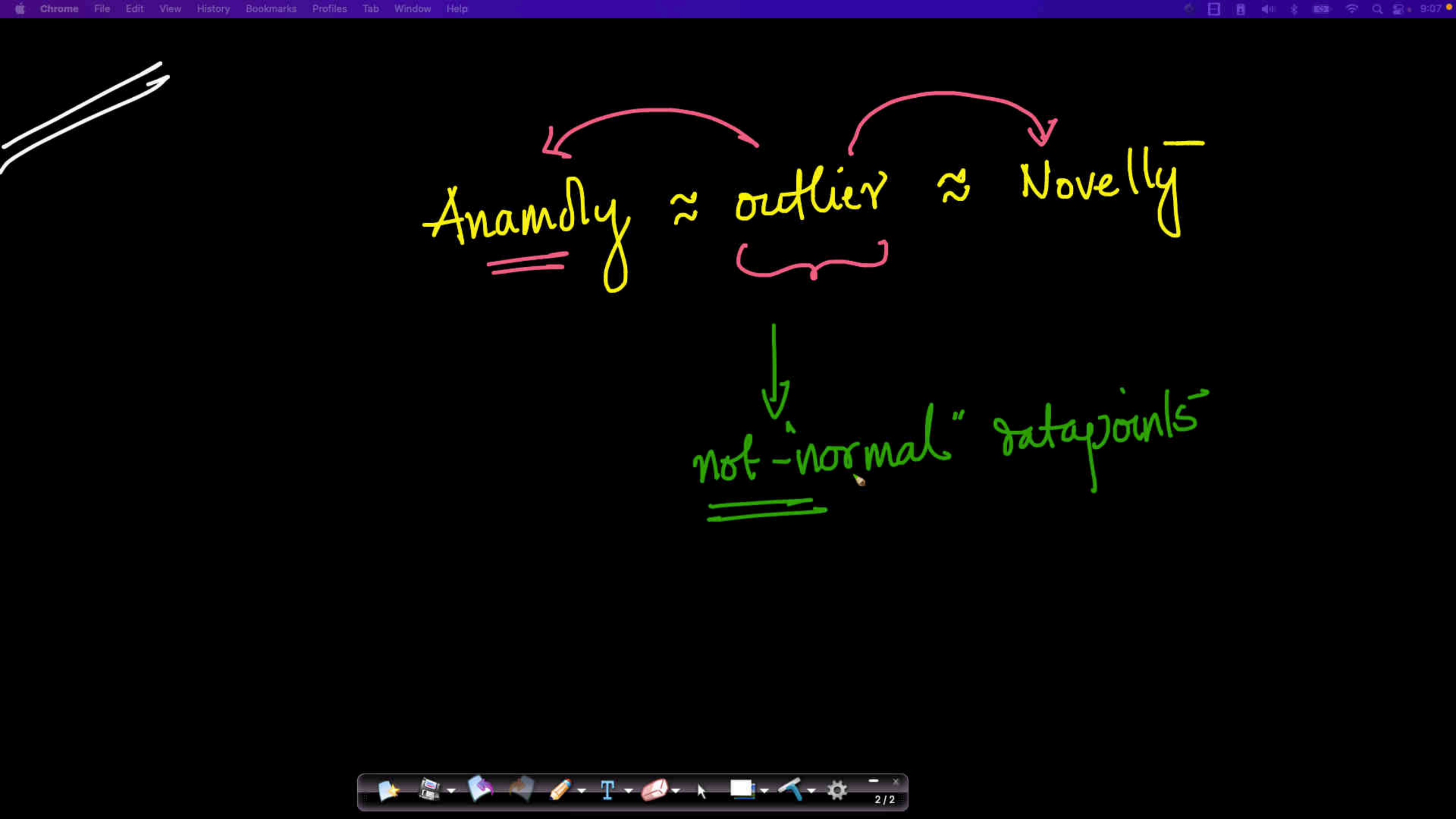
Intuition ✓

Math ✓

Code ✓

✓ Toy - datasets

Real - datasets


Anamoly ≈ outlier ≈ Novelly
not-normal" datapoints
2 / 2

Techniques
[outlier / anomalies / Novel]

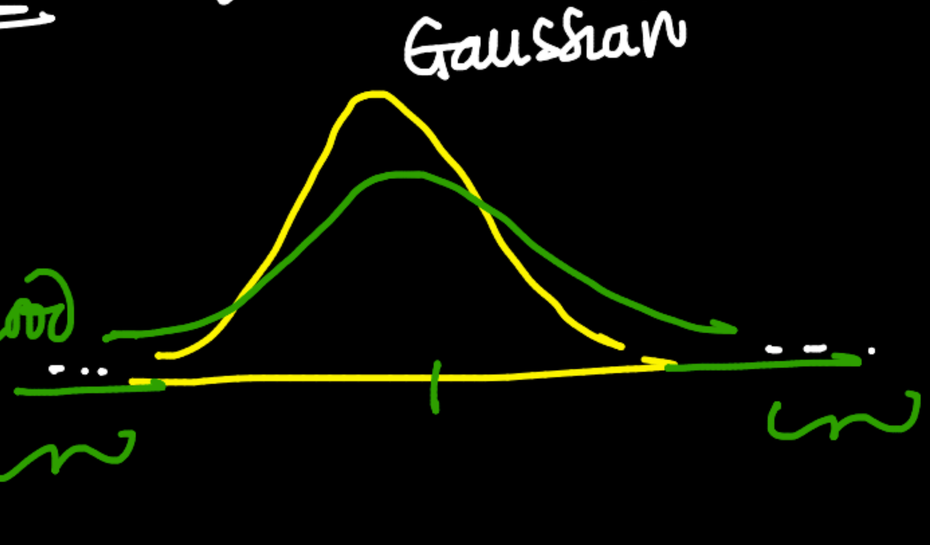
1. DBSCAN → density ; outliers
2. RANSAC → Robust → model-specific outlier
3. Boxplot — IQR → univariate → MAD , Z-score
   $\frac{x_i - \mu}{\sigma}$, Kurtosis
4. KNN → [avg-dist k-nearest neigh] ↑
5. Viz of e_i's : errors
   → model specific (bias-variance)

Measure of peakedness ✗

Kurtosis → measure of tailedness ✓

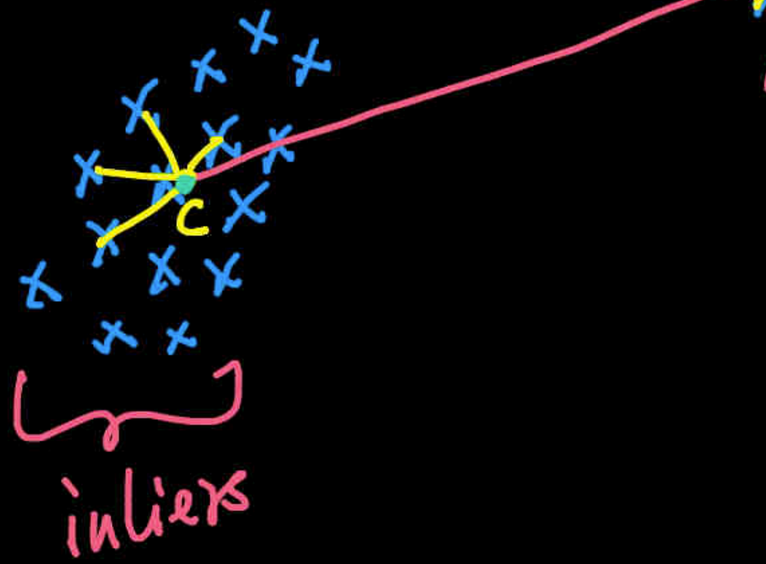
Kurt(Gaussian) = 3

higher kurt → more likelihood of outliers



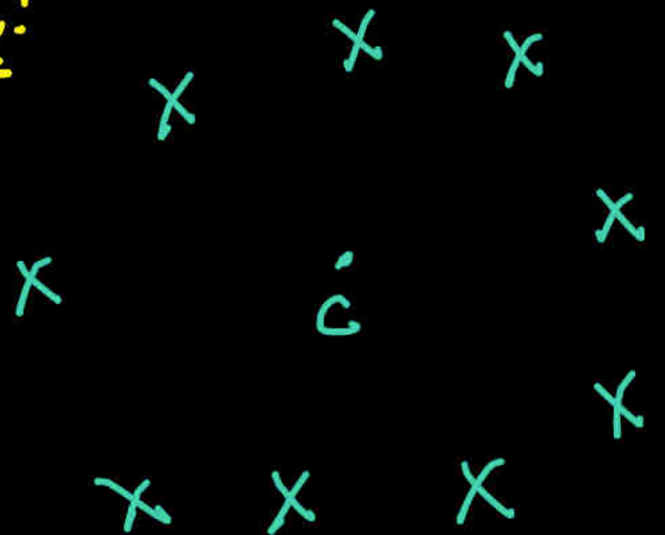
Q

Case 1:

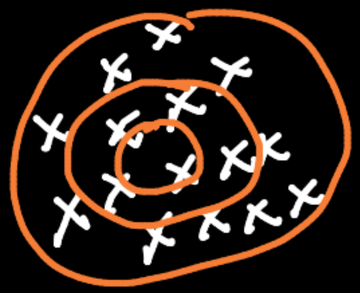


[Suggested]

Case 2:

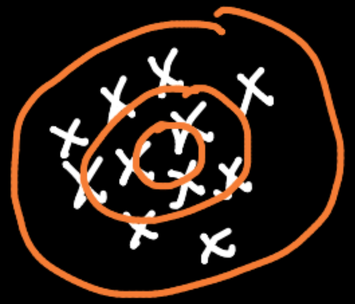


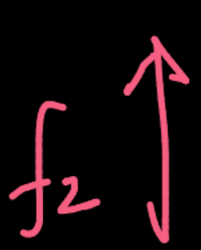
Case 3:



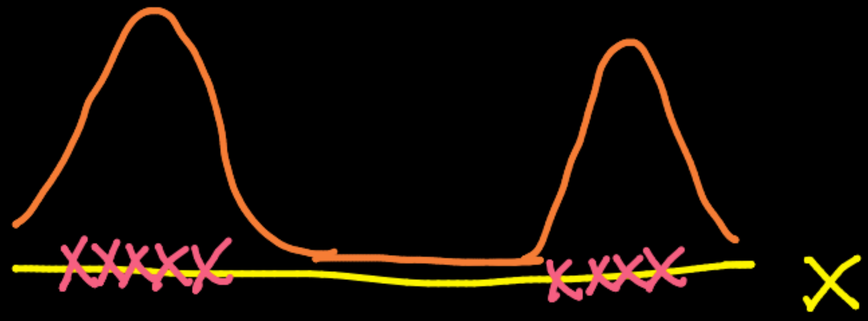
x ✓

bi-modal data





x ✓



1D

Known: $X \sim \text{distb}(\theta)$ (θ: params)

Distribution based

f. $X : x_1\ x_2\ \ldots\ x_n$

Let $X \sim N(\mu, \sigma^2)$

$\mu \approx$ sample-mean

problem: param of the distb are often corrupted by outliers

Soln:

soln 1: drop plz 3σ - away

soln 2: 5 - 95 वाँ percentiles

soln 3: CLT & bootstrapping → outlier-free

soln 4: IQR

soln 5: Median ≈ Mean; Std-dev ≈ MAD

soln 6: Generative models

HINT: $X \sim \text{Gaussian}(\underbrace{\mu, \sigma^2}_{\theta})$: Model

ROBUST

↳ Robustly estimate θ of my model

θ → $[\mu, \sigma^2]$

RANSAC

Z-score → outliers

$\hat{\theta}_1$ ← $n < N$

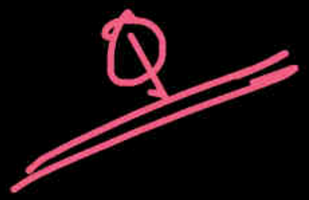
Q

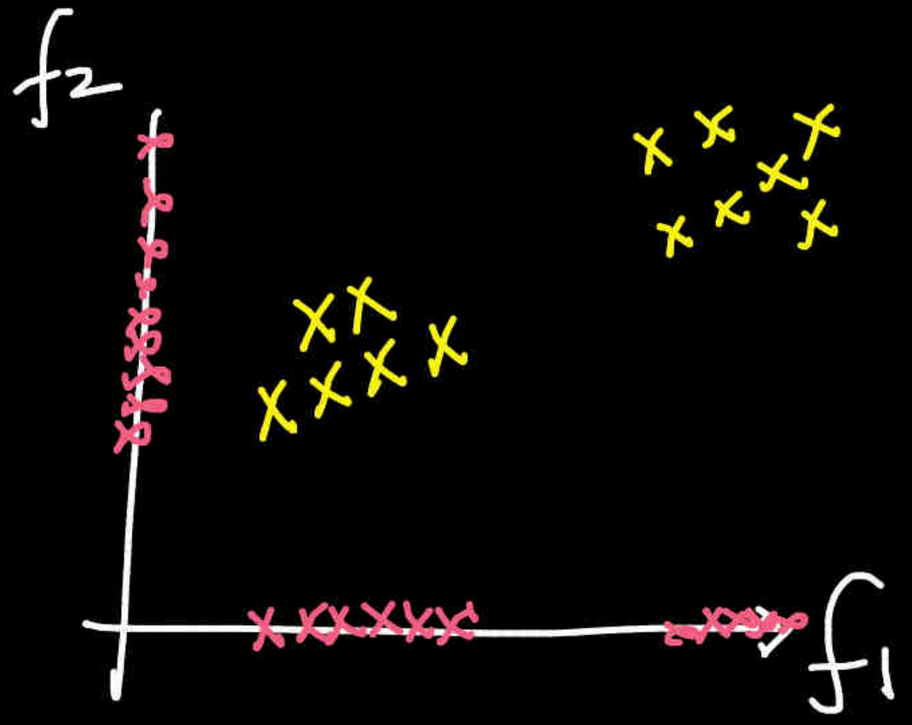
Can we use sample std-dev (s) as a stopping criterion

s_1

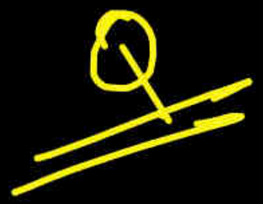
s_2

$\{s_t, s_{t+1}\}$





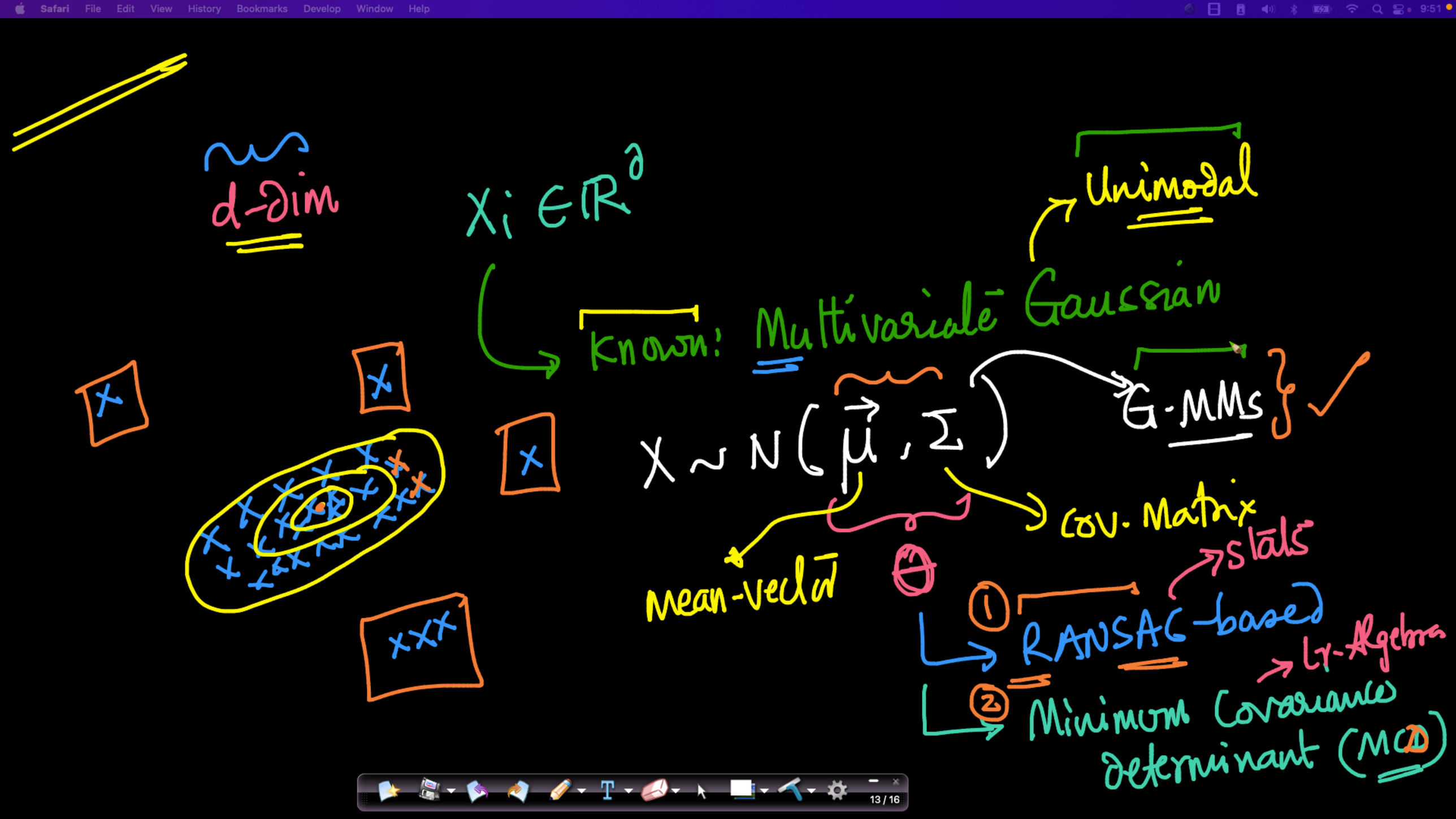
Univariate percentiles through projections



RANSAC: model based outlier detn

↓

Gaussian (μ, σ^2)


d-Dim
$X_i \in \mathbb{R}^d$
Known: Multivariate Gaussian
Unimodal
$X \sim N(\vec{\mu}, \Sigma)$
G-MMs
Mean-vector
Cov. Matrix
Θ
1
RANSAC-based
Scales
2
Minimum Covariance determinant (MCD)
Ly-Algebra
13 / 16

Elliptical Envelope

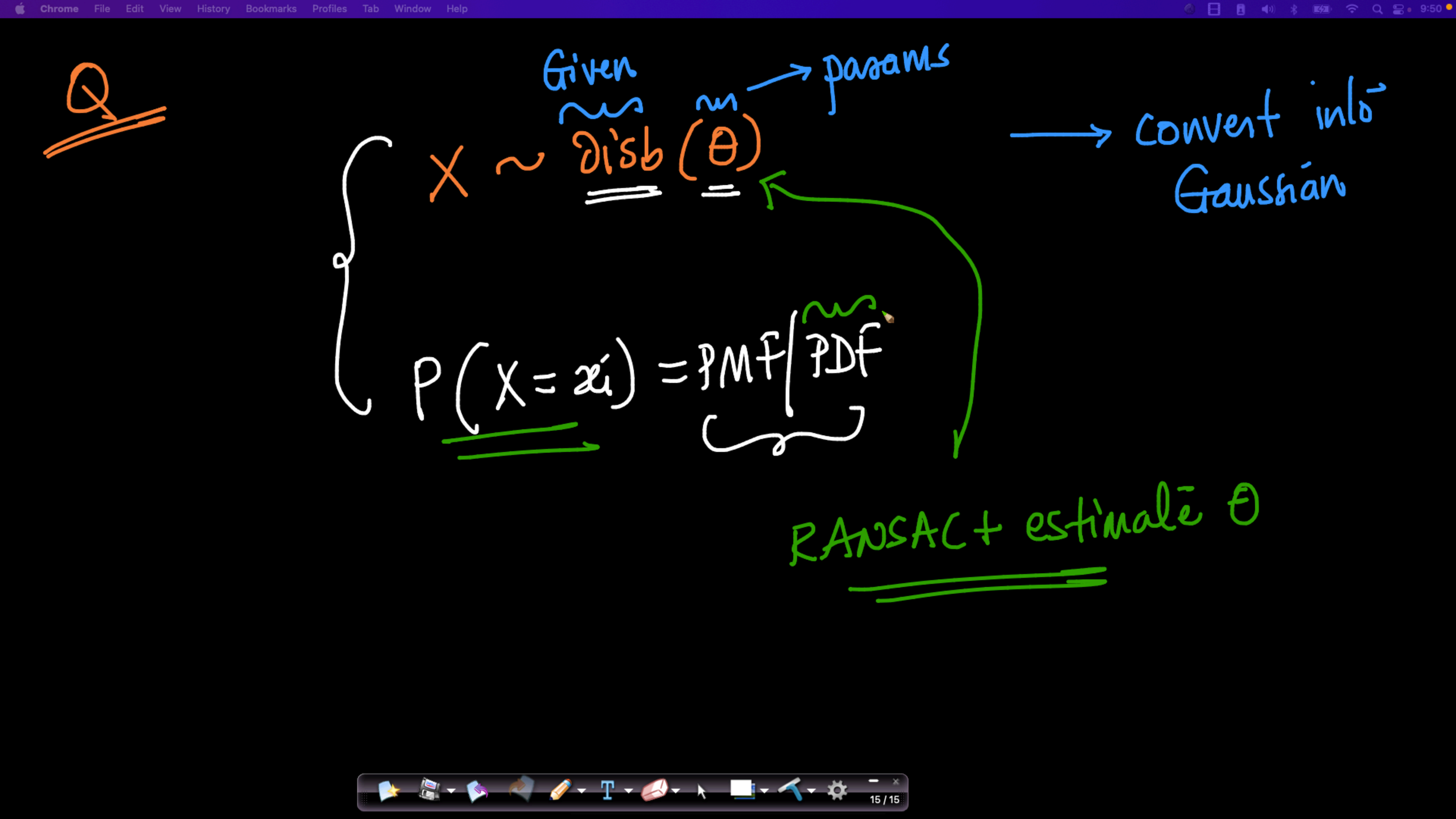
Given: $x_i \in \mathbb{R}^d$

$X \sim N(\vec{\mu}, \Sigma)$

$X \sim$ Unimodal

→ "robustly" estimate $\vec{\mu}, \Sigma$

→ remove points which are outliers


Q
Given
params
$X \sim \text{Distb}(\theta)$
$P(X = x_i) = \text{PMF} / \text{PDF}$
Convert into Gaussian
RANSAC + estimate θ

Multivariate normal distribution

From Wikipedia, the free encyclopedia

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$$

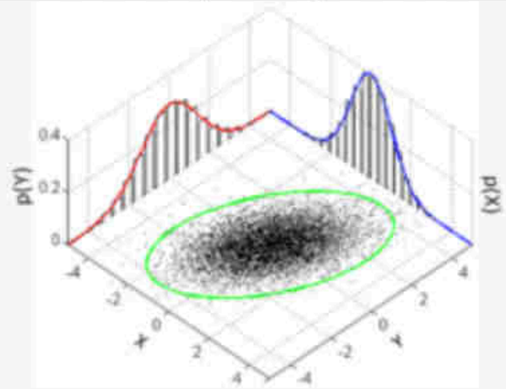
"MVN" redirects here. For the airport with that IATA code, see Mount Vernon Airport. For the mvn build-automation software, see Apache Maven.

In probability theory and statistics, the **multivariate normal distribution**, **multivariate Gaussian distribution**, or **joint normal distribution** is a generalization of the one-dimensional (univariate) normal distribution to higher dimensions. One definition is that a random vector is said to be k-variate normally distributed if every linear combination of its k components has a univariate normal distribution. Its importance derives mainly from the multivariate central limit theorem. The multivariate normal distribution is often used to describe, at least approximately, any set of (possibly) correlated real-valued random variables each of which clusters around a mean value.

Contents [hide]

Multivariate normal

Probability density function



Many sample points from a multivariate normal distribution with $\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 3/5 \\ 3/5 & 2 \end{bmatrix}$, shown along with the 3-sigma ellipse, the two marginal distributions, and the two 1-d histograms.

Notation	$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
Parameters	$\boldsymbol{\mu} \in \mathbf{R}^k$ — location $\boldsymbol{\Sigma} \in \mathbf{R}^{k \times k}$ — covariance (positive semi-definite matrix)

16 / 16

Elliptical-envelope : Gaussian + Unimodal

disadv

→ non Unimodal data

→ MV-Gaussian

Runtime application self-protection

V • T • E

Popular techniques [edit]

Many anomaly detection techniques have been proposed in literature.[1][10] Some of the popular techniques are:

- Statistical (Z-score, Tukey's range test and Grubbs's test)
- Density-based techniques (k-nearest neighbor,[11][12][13] local outlier factor,[14] isolation forests,[15][16] and many more variations of this concept[17])
- Subspace-,[18] correlation-based[19] and tensor-based [20] outlier detection for high-dimensional data[21]
- One-class support vector machines[22]
- Replicator neural networks,[23] autoencoders, variational autoencoders,[24] long short-term memory neural networks[25]
- Bayesian networks[23]
- Hidden Markov models (HMMs)[23]
- Minimum Covariance Determinant[26][27]
- Clustering: Cluster analysis-based outlier detection[28][29]
- Deviations from association rules and frequent itemsets
- Fuzzy logic-based outlier detection
- Ensemble techniques, using feature bagging,[30][31] score normalization[32][33] and different sources of diversity[34][35]

The performance of methods depends on the data set and parameters, and methods have little systematic advantages over another when compared across many data sets and parameters.[36][37]

Software [edit]

- ELKI is an open-source Java data mining toolkit that contains several anomaly detection algorithms, as well as index acceleration for them.

sklearn.covariance.EllipticEnvelope

class sklearn.covariance.**EllipticEnvelope**(*, *store_precision=True*, *assume_centered=False*, *support_fraction=None*, *contamination=0.1*, *random_state=None*) [source]

An object for detecting outliers in a Gaussian distributed dataset.

Read more in the User Guide.

Parameters:

store_precision : *bool, default=True*

Specify if the estimated precision is stored.

assume_centered : *bool, default=False*

If True, the support of robust location and covariance estimates is computed, and a covariance estimate is recomputed from it, without centering the data. Useful to work with data whose mean is significantly equal to zero but is not exactly zero. If False, the robust location and covariance are directly computed with the FastMCD algorithm without additional treatment.

support_fraction : *float, default=None*

d in the support of the raw MCD estimate. If None, the

sklearn.covariance.EllipticEnvelope

class `sklearn.covariance.`**`EllipticEnvelope`**(**, store_precision=True, assume_centered=False, support_fraction=None, contamination=0.1, random_state=None*) [source]

An object for detecting outliers in a Gaussian distributed dataset.

Read more in the User Guide.

Parameters:

store_precision : *bool, default=True*
Specify if the estimated precision is stored.

assume_centered : *bool, default=False*
If True, the support of robust location and covariance estimates is computed, and a covariance estimate is recomputed from it, without centering the data. Useful to work with data whose mean is significantly equal to zero but is not exactly zero. If False, the robust location and covariance are directly computed with the FastMCD algorithm without additional treatment.

support_fraction : *float, default=None*
...d in the support of the raw MCD estimate. If None, the

sklearn.covariance.EllipticEnvelope

class `sklearn.covariance.`**`EllipticEnvelope`**(**, store_precision=True, assume_centered=False, support_fraction=None, contamination=0.1, random_state=None*) [source]

An object for detecting outliers in a Gaussian distributed dataset.

Read more in the User Guide.

Parameters:

store_precision : *bool, default=True*
Specify if the estimated precision is stored.

assume_centered : *bool, default=False*
If True, the support of robust location and covariance estimates is computed, and a covariance estimate is recomputed from it, without centering the data. Useful to work with data whose mean is significantly equal to zero but is not exactly zero. If False, the robust location and covariance are directly computed with the FastMCD algorithm without additional treatment.

support_fraction : *float, default=None*
d in the support of the raw MCD estimate. If None, the

Read more in the User Guide.

Parameters:

store_precision : *bool, default=True*

Specify if the estimated precision is stored.

assume_centered : *bool, default=False*

If True, the support of robust location and covariance estimates is computed, and a covariance estimate is recomputed from it, without centering the data. Useful to work with data whose mean is significantly equal to zero but is not exactly zero. If False, the robust location and covariance are directly computed with the FastMCD algorithm without additional treatment.

support_fraction : *float, default=None*

The proportion of points to be included in the support of the raw MCD estimate. If None, the minimum value of support_fraction will be used within the algorithm: `[n_sample + n_features + 1] / 2`. Range is (0, 1).

contamination : *float, default=0.1*

The amount of contamination of the data set, i.e. the proportion of outliers in the data set. Range is (0, 0.5].

random_state : *int, RandomState instance or None, default=None*

Determines the pseudo random number generator for shuffling the data. Pass an int for

unction calls. See Glossary.

scikit learn
Prev Up Next
scikit-learn 1.1.1
Other versions
Please cite us if you use the software.
sklearn.covariance.EllipticEnvelope
Examples using sklearn.covariance.EllipticEnvelope

Specify if the estimated precision is stored.

assume_centered : *bool, default=False*

If True, the support of robust location and covariance estimates is computed, and a covariance estimate is recomputed from it, without centering the data. Useful to work with data whose mean is significantly equal to zero but is not exactly zero. If False, the robust location and covariance are directly computed with the FastMCD algorithm without additional treatment.

support_fraction : *float, default=None*

The proportion of points to be included in the support of the raw MCD estimate. If None, the minimum value of support_fraction will be used within the algorithm: `[n_sample + n_features + 1] / 2`. Range is (0, 1).

contamination : *float, default=0.1*

The amount of contamination of the data set, i.e. the proportion of outliers in the data set. Range is (0, 0.5].

random_state : *int, RandomState instance or None, default=None*

Determines the pseudo random number generator for shuffling the data. Pass an int for reproducible results across multiple function calls. See Glossary.

Attributes:

location_ : *ndarray of shape (n_features,)*

Estimated robust location.

covariance_ : *ndarray of shape (n_features, n_features)*

References

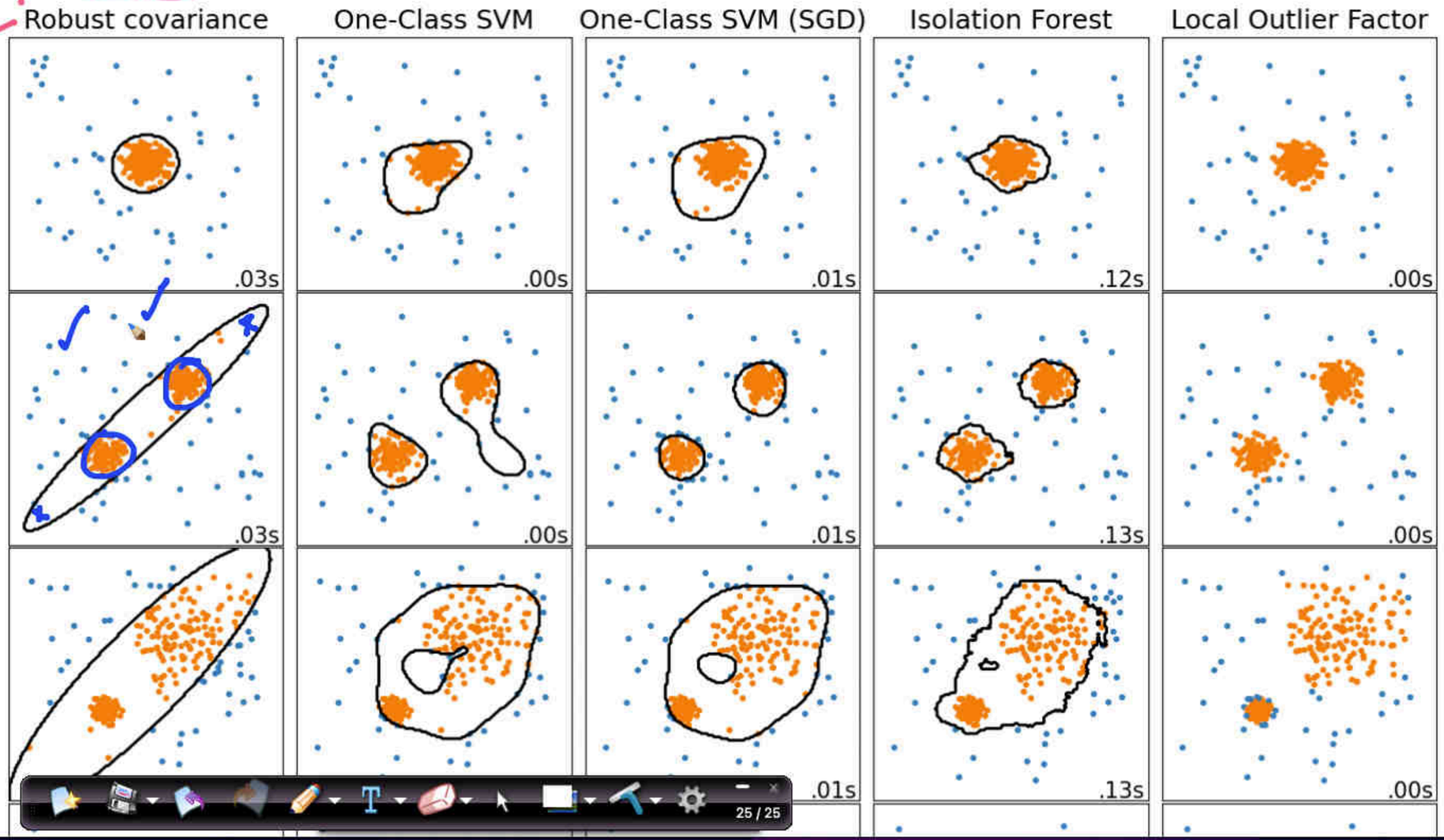
1 Rousseeuw, P.J., Van Driessen, K. "A fast algorithm for the minimum covariance determinant estimator" Technometrics 41(3), 212 (1999)

Examples

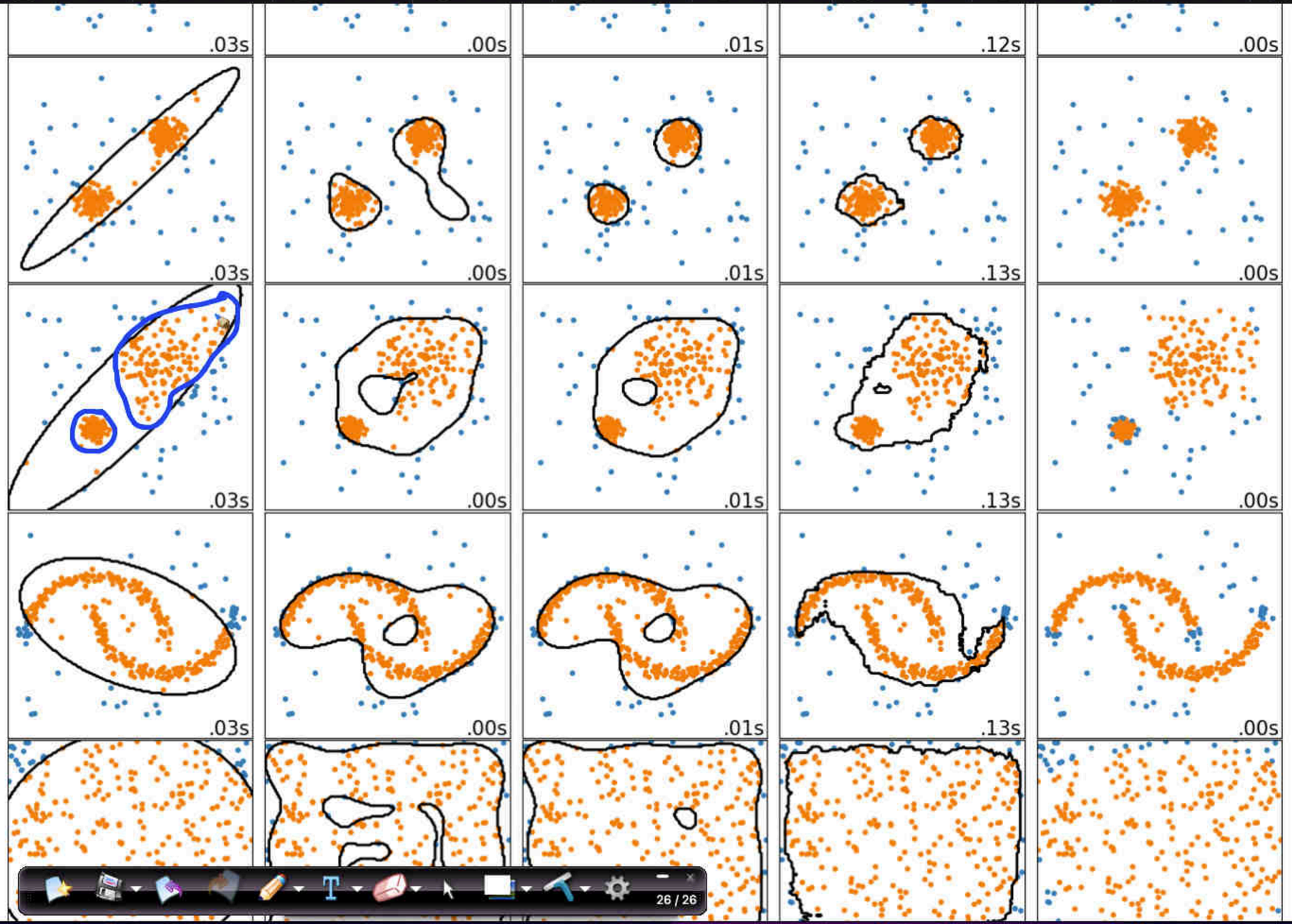
```
>>> import numpy as np
>>> from sklearn.covariance import EllipticEnvelope
>>> true_cov = np.array([[.8, .3],
...                      [.3, .4]])
>>> X = np.random.RandomState(0).multivariate_normal(mean=[0, 0],
...                                                  cov=true_cov,
...                                                  size=500)
>>> cov = EllipticEnvelope(random_state=0).fit(X)
>>> # predict returns 1 for an inlier and -1 for an outlier
>>> cov.predict([[0, 0],
...              [3, 3]])
array([ 1, -1])
>>> cov.covariance_
array([[0.7411..., 0.2535...],
       [0.2535..., 0.3053...]])
>>> cov.location_
array([0.0813... , 0.0427...])
```

Methods

Finally, note that parameters of the models have been here handpicked but that in practice they need to be adjusted. In the absence of labelled data, the problem is completely unsupervised so model selection can be a challenge.

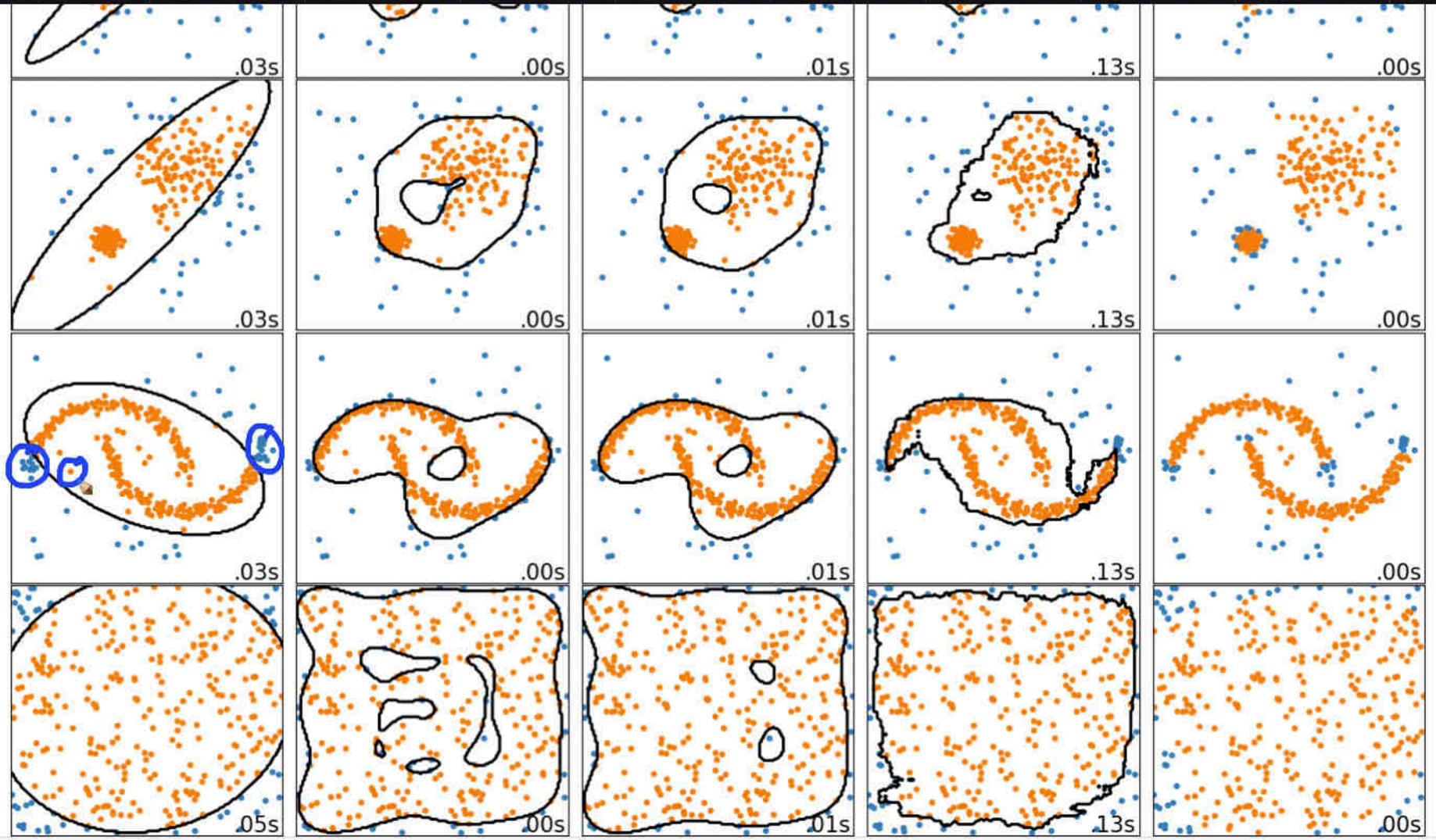


scikit learn
Prev Up Next
scikit-learn 1.1.1
Other versions
Please cite us if you use the software.
Comparing anomaly detection algorithms for outlier detection on toy datasets



Toggle Menu

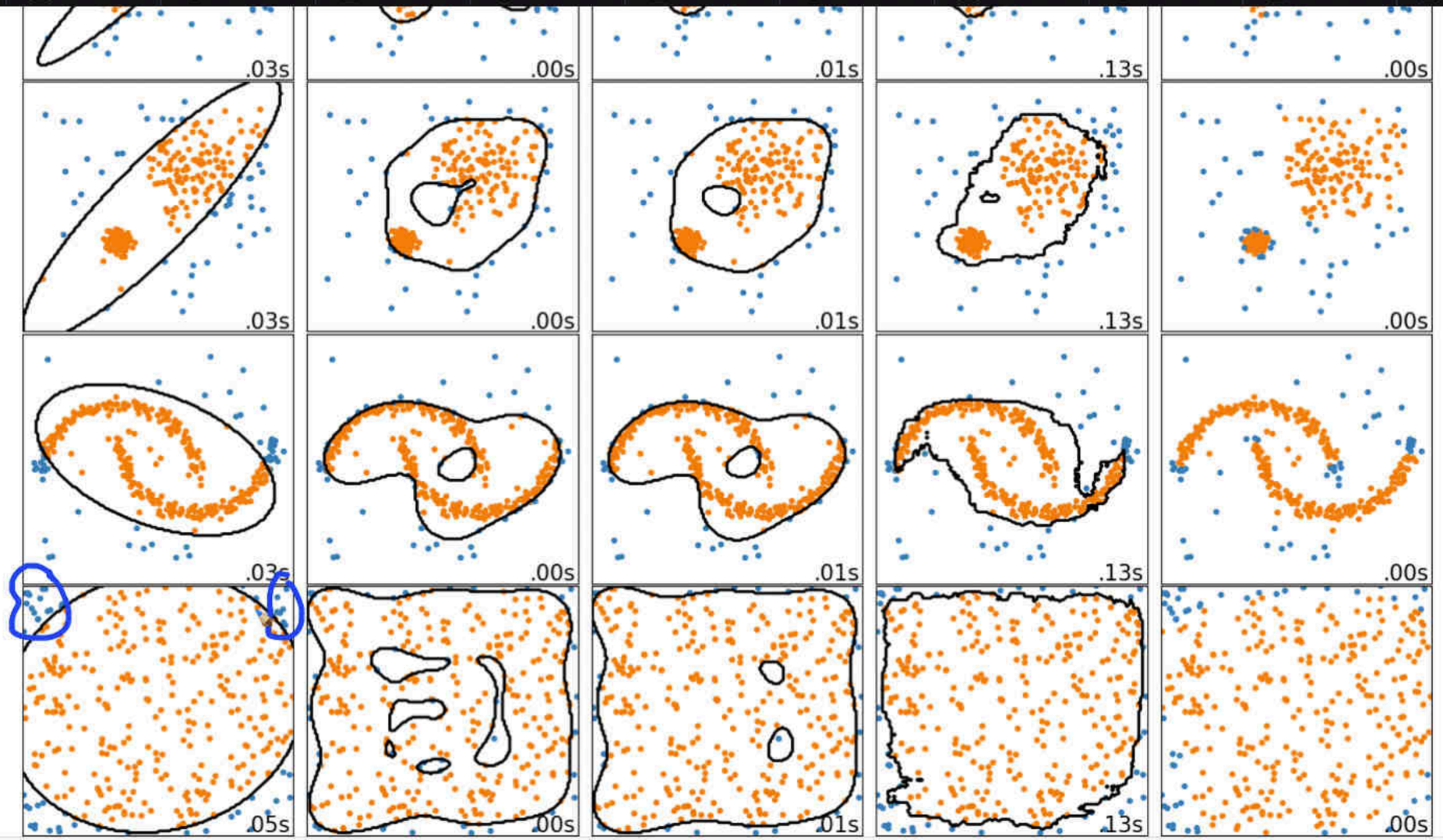
scikit learn
Prev Up Next
scikit-learn 1.1.1
Other versions
Please cite us if you use the software.
Comparing anomaly detection algorithms for outlier detection on toy datasets



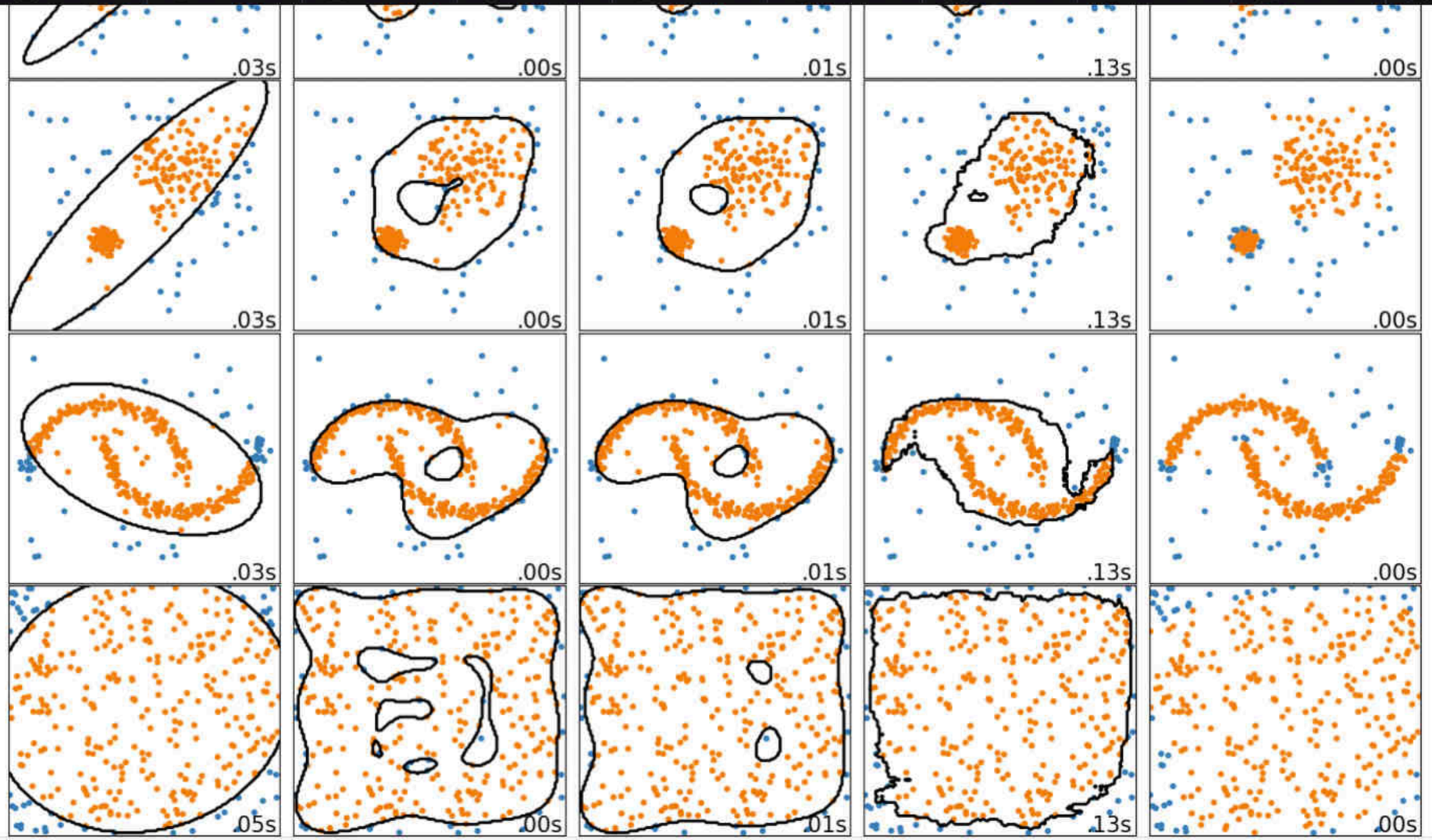

```
# Author: Alexandre Gramfort <alexandre.gramfort@inria.fr>
#         Albert Thomas <albert.thomas@telecom-paristech.fr>
# License: BSD 3 clause

im
```

Toggle Menu

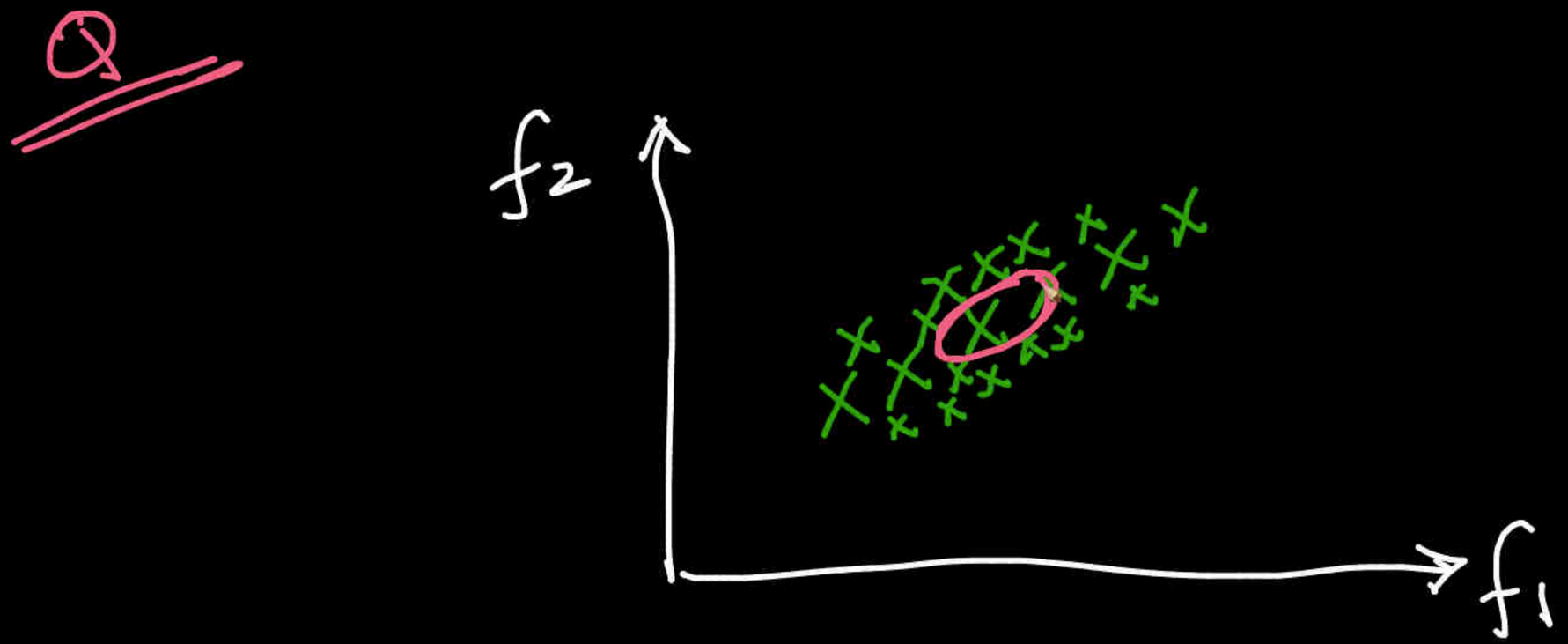
scikit learn
Prev Up Next
scikit-learn 1.1.1
Other versions
Please cite us if you use the software.
Comparing anomaly detection algorithms for outlier detection on toy datasets




```
# Author: Alexandre Gramfort <alexandre.gramfort@inria.fr>
#         Albert Thomas <albert.thomas@telecom-paristech.fr>
# License: BSD 3 clause

im
```



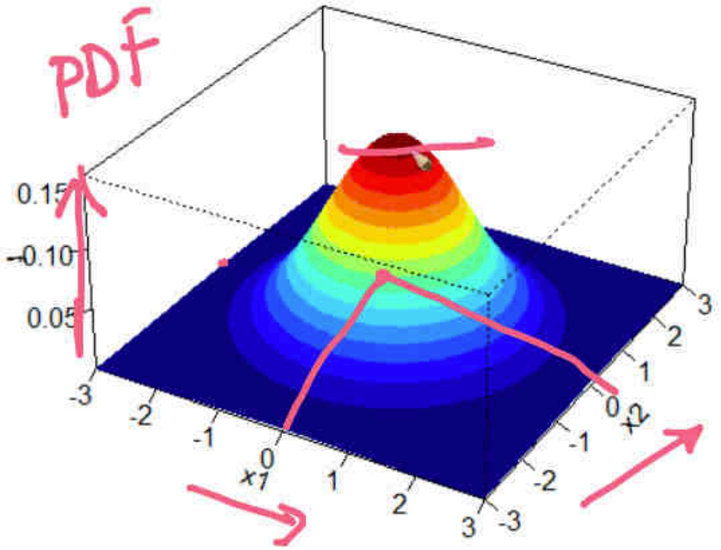

```
# Author: Alexandre Gramfort <alexandre.gramfort@inria.fr>
#         Albert Thomas <albert.thomas@telecom-paristech.fr>
# License: BSD 3 clause

im
```

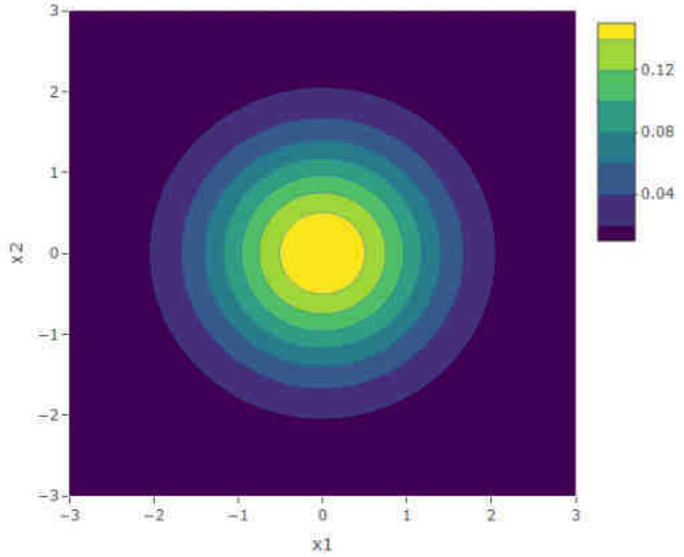

Q
f_2
f_1

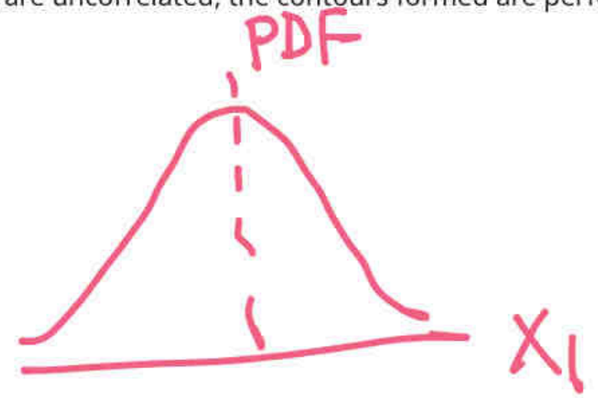
Case 1: $X_1, X_2 \sim \mathcal{N}(0, 1)$ with correlation 0.

The equation for the correlation ρ is given by $\rho = \frac{\sigma_{12}}{\sqrt{\sigma_{11}\sigma_{22}}}$. Hence $\sigma_{12} = \rho\sqrt{\sigma_{11}\sigma_{22}}$. When $\rho = 0$, $\sigma_{11} = 1$ and $\sigma_{22} = 1$, $\sigma_{12} = \rho = 0$.

We substitute $\mu_1, \mu_2 = 0$, $\sigma_{11}, \sigma_{22} = 1$ and $\sigma_{12} = 0$ in Equation (1) and obtain the following 3D plot and contour plot. The shape of the bivariate normal distribution is again similar to a that of a bell. The multivariate normal distribution reaches its peak at (μ_1, μ_2). Thus in this example, the maximum is reached at $(0, 0)$. Here, since X_1 and X_2 are uncorrelated, the contours formed are perfect circles.

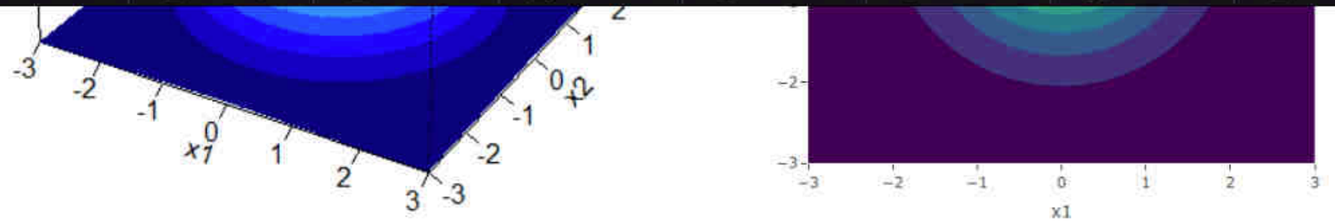






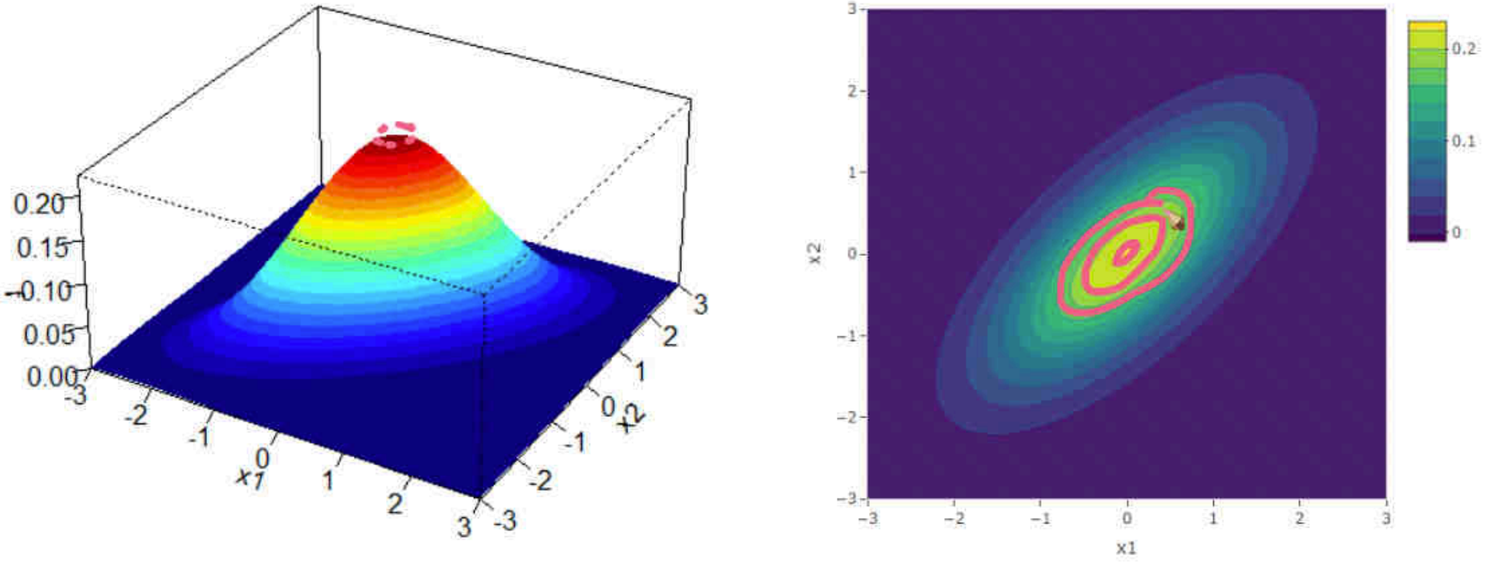
Case 2: $X_1, X_2 \sim \mathcal{N}(0, 1)$ with correlation 0.7.

Since $\rho = 0.7$ and $\sigma_{12} = \rho\sqrt{\sigma_{11}\sigma_{22}}$, then $\sigma_{12} = 0.7$. By inputting the values of the means, variances and covariances in Equation (1), we obtain the following plots. Again we obtain a bell-shaped bivariate distribution. Since the correlation between X_1 and X_2 is positive, we obtain elliptical contours. It is like taking the circular contours of the uncorrelated case and elongate them along the diagonal $x_2 = x_1$.



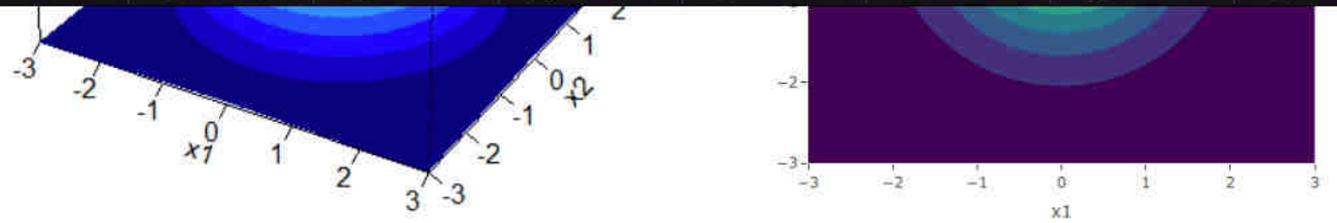
Case 2: $X_1, X_2 \sim \mathcal{N}(0, 1)$ with correlation 0.7.

Since $\rho = 0.7$ and $\sigma_{12} = \rho\sqrt{\sigma_{11}\sigma_{22}}$, then $\sigma_{12} = 0.7$. By inputting the values of the means, variances and covariances in Equation (1), we obtain the following plots. Again we obtain a bell-shaped bivariate distribution. Since the correlation between X_1 and X_2 is positive, we obtain elliptical contours. It is like taking the circular contours of the uncorrelated case and elongate them along the diagonal $x_2 = x_1$.



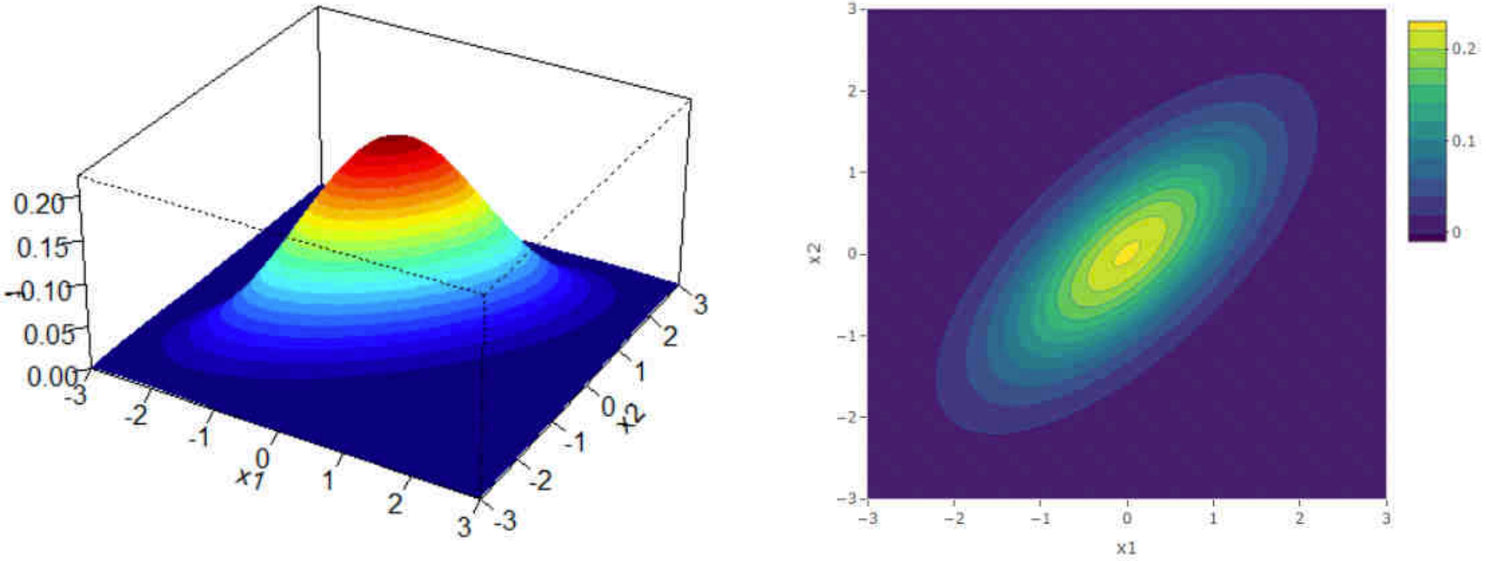
Case 3: $X_1, X_2 \sim \mathcal{N}(0, 1)$ with correlation -0.7.

The following are the plots for the case when the correlation is negative. Similar to the second case, we have elliptical contours. However in this case, the circular contours are elongated along the second diagonal, that is, the line $x_2 = -x_1$.



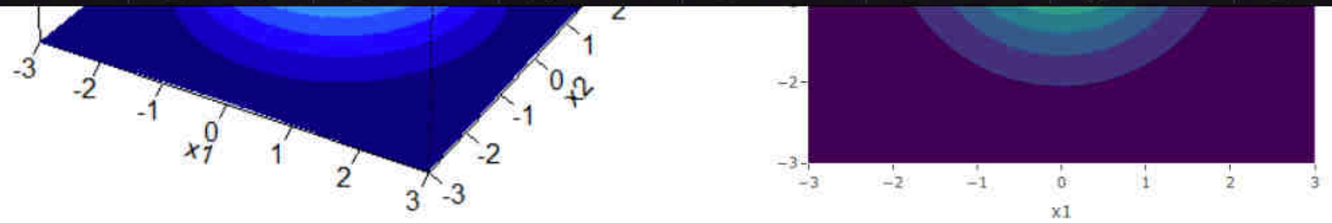
Case 2: $X_1, X_2 \sim \mathcal{N}(0, 1)$ with correlation 0.7.

Since $\rho = 0.7$ and $\sigma_{12} = \rho\sqrt{\sigma_{11}\sigma_{22}}$, then $\sigma_{12} = 0.7$. By inputting the values of the means, variances and covariances in Equation (1), we obtain the following plots. Again we obtain a bell-shaped bivariate distribution. Since the correlation between X_1 and X_2 is positive, we obtain elliptical contours. It is like taking the circular contours of the uncorrelated case and elongate them along the diagonal $x_2 = x_1$.



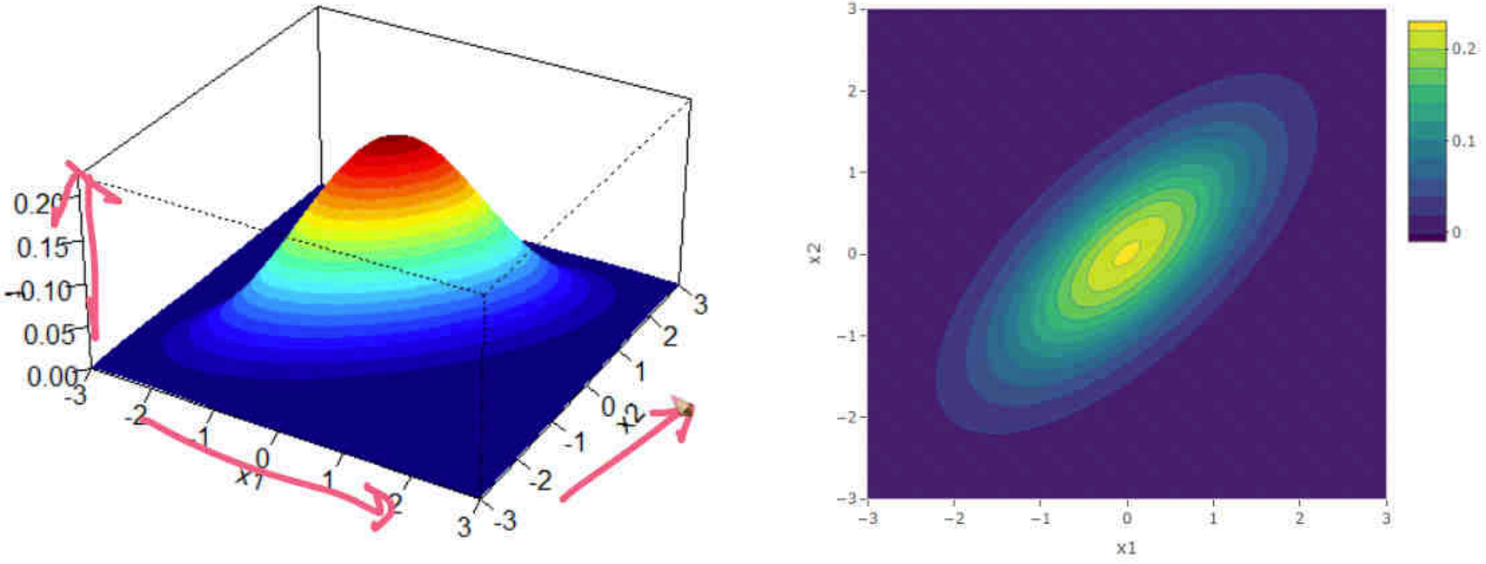
Case 3: $X_1, X_2 \sim \mathcal{N}(0, 1)$ with correlation -0.7.

The following are the plots for the case when the correlation is negative. Similar to the second case, we have elliptical contours. However in this case, the circular contours are elongated along the second diagonal, that is, the line $x_2 = -x_1$.



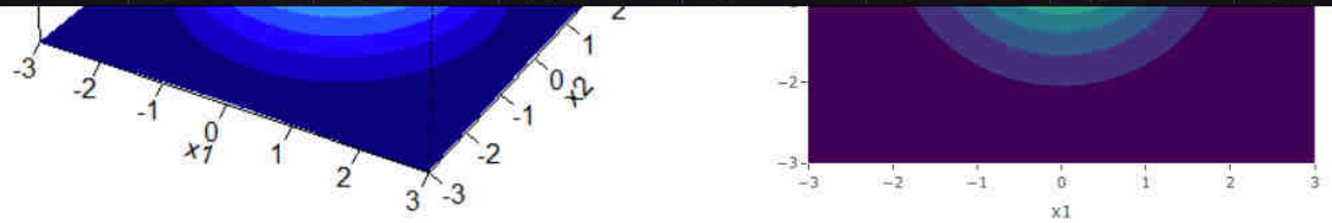
Case 2: $X_1, X_2 \sim \mathcal{N}(0, 1)$ with correlation 0.7.

Since $\rho = 0.7$ and $\sigma_{12} = \rho\sqrt{\sigma_{11}\sigma_{22}}$, then $\sigma_{12} = 0.7$. By inputting the values of the means, variances and covariances in Equation (1), we obtain the following plots. Again we obtain a bell-shaped bivariate distribution. Since the correlation between X_1 and X_2 is positive, we obtain elliptical contours. It is like taking the circular contours of the uncorrelated case and elongate them along the diagonal $x_2 = x_1$.



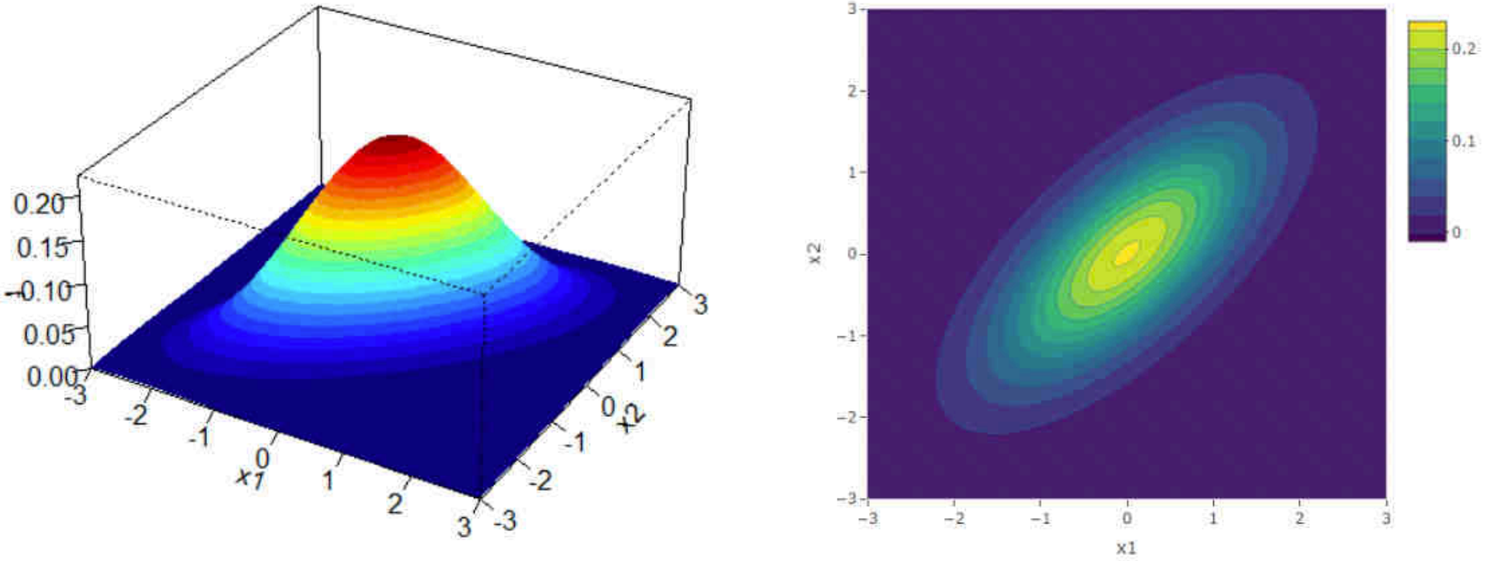
Case 3: $X_1, X_2 \sim \mathcal{N}(0, 1)$ with correlation -0.7.

The following are the plots for the case when the correlation is negative. Similar to the second case, we have elliptical contours. However in this case, the circular contours are elongated along the second diagonal, that is, the line $x_2 = -x_1$.



Case 2: $X_1, X_2 \sim \mathcal{N}(0, 1)$ with correlation 0.7.

Since $\rho = 0.7$ and $\sigma_{12} = \rho\sqrt{\sigma_{11}\sigma_{22}}$, then $\sigma_{12} = 0.7$. By inputting the values of the means, variances and covariances in Equation (1), we obtain the following plots. Again we obtain a bell-shaped bivariate distribution. Since the correlation between X_1 and X_2 is positive, we obtain elliptical contours. It is like taking the circular contours of the uncorrelated case and elongate them along the diagonal $x_2 = x_1$.



Case 3: $X_1, X_2 \sim \mathcal{N}(0, 1)$ with correlation -0.7.

The following are the plots for the case when the correlation is negative. Similar to the second case, we have elliptical contours. However in this case, the circular contours are elongated along the second diagonal, that is, the line $x_2 = -x_1$.

Isolation Forest

2007 - ...

$x_i \in \mathbb{R}^2$

$$\mathcal{D} = \{x_i\}_{i=1}^{n}$$

Idea:

→ build many trees (like RF)

→ each tree: (random tree)
- → randomly pick a feature
- → randomly threshold it

→ build each tree till the leafnode consists of just one datapoint

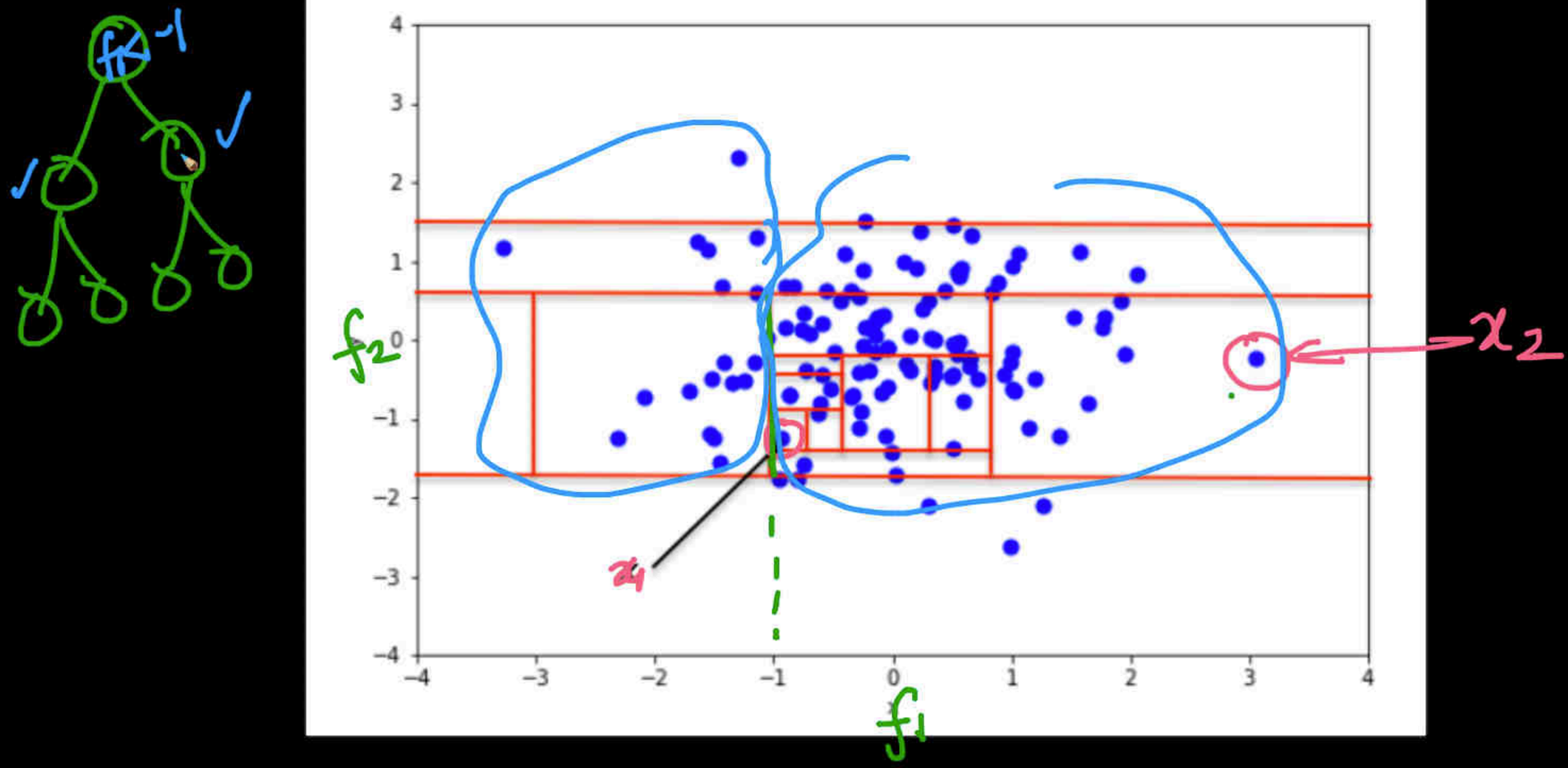


Fig. 2 - an example of isolating a non-anomalous point in a 2D Gaussian distribution.

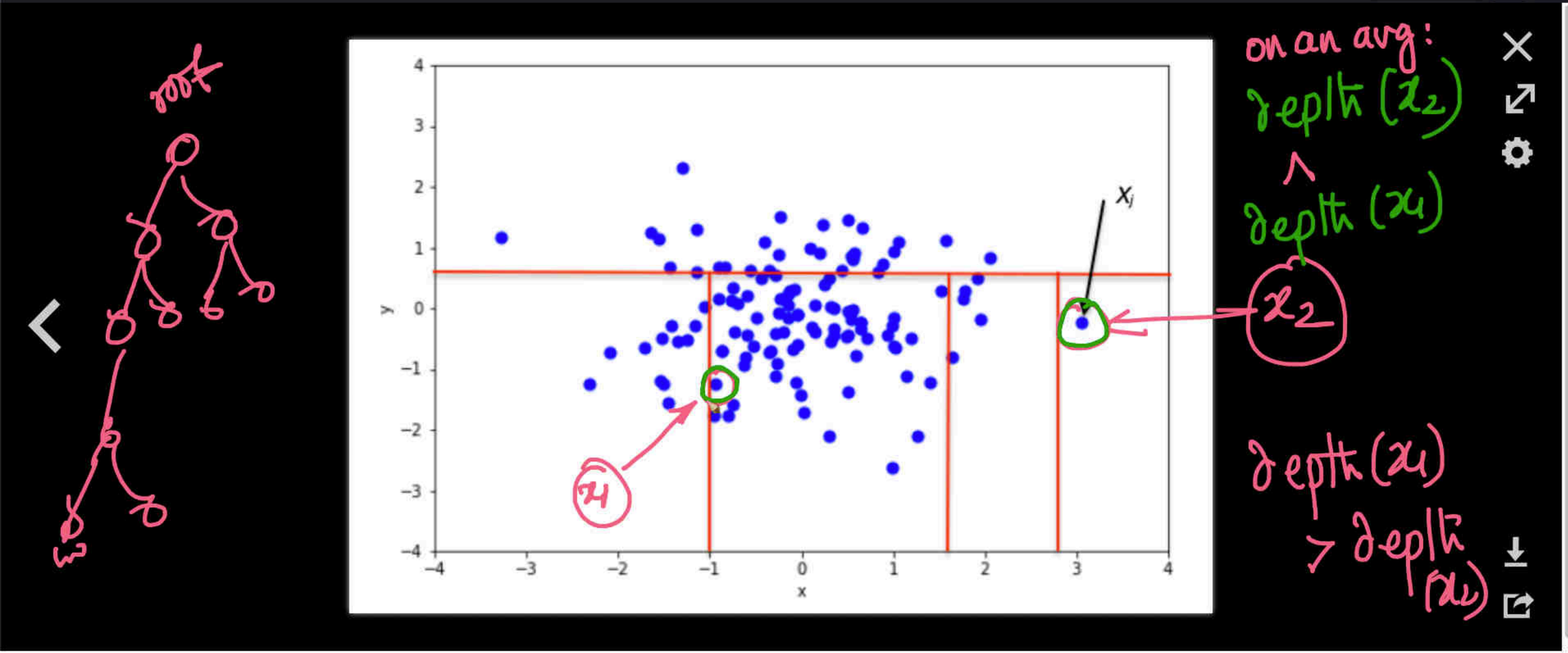


Fig. 3 - an example of isolating an anomalous point in a 2D Gaussian distribution.

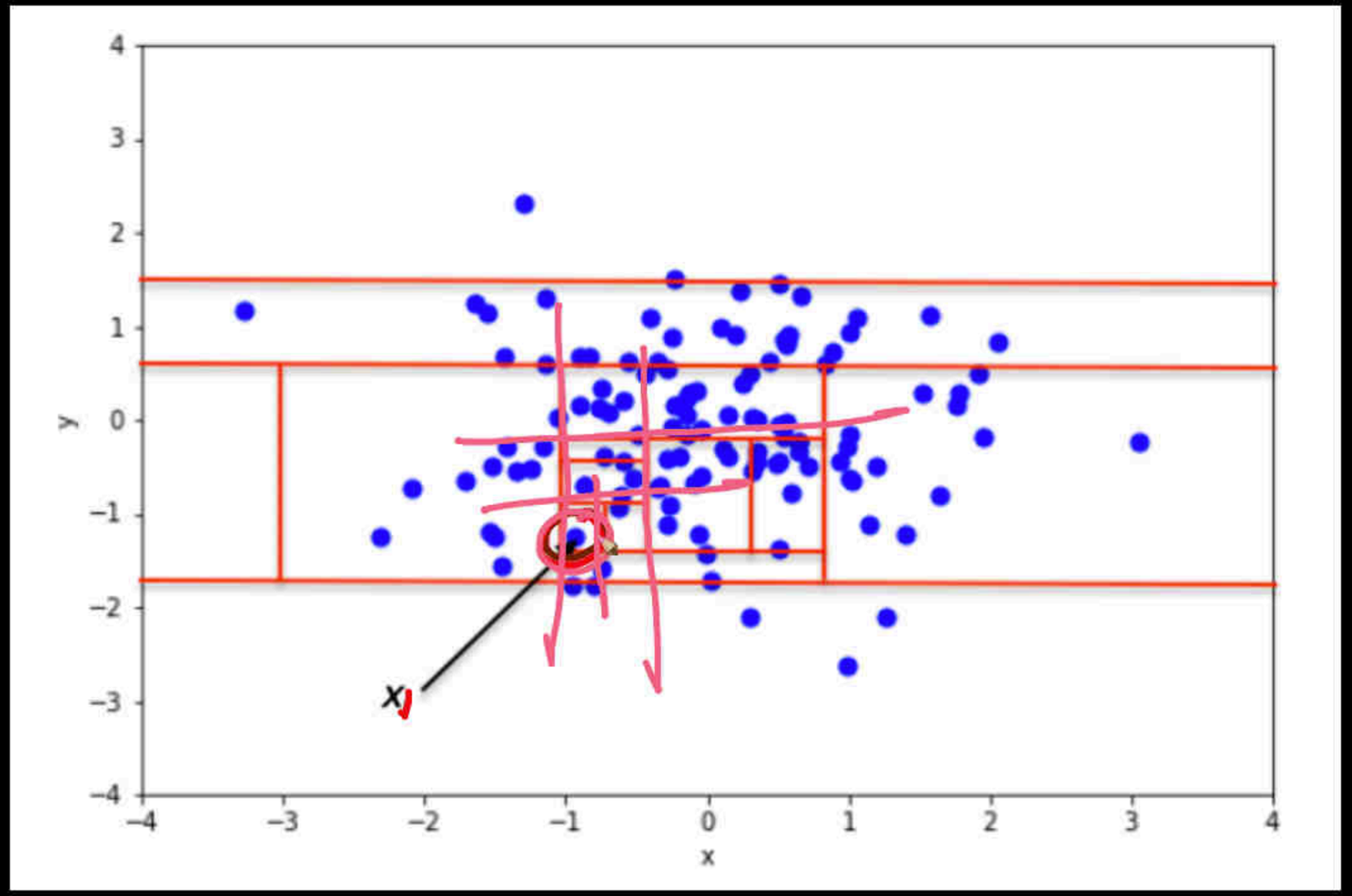


Fig. 2 - an example of isolating a non-anomalous point in a 2D Gaussian distribution.

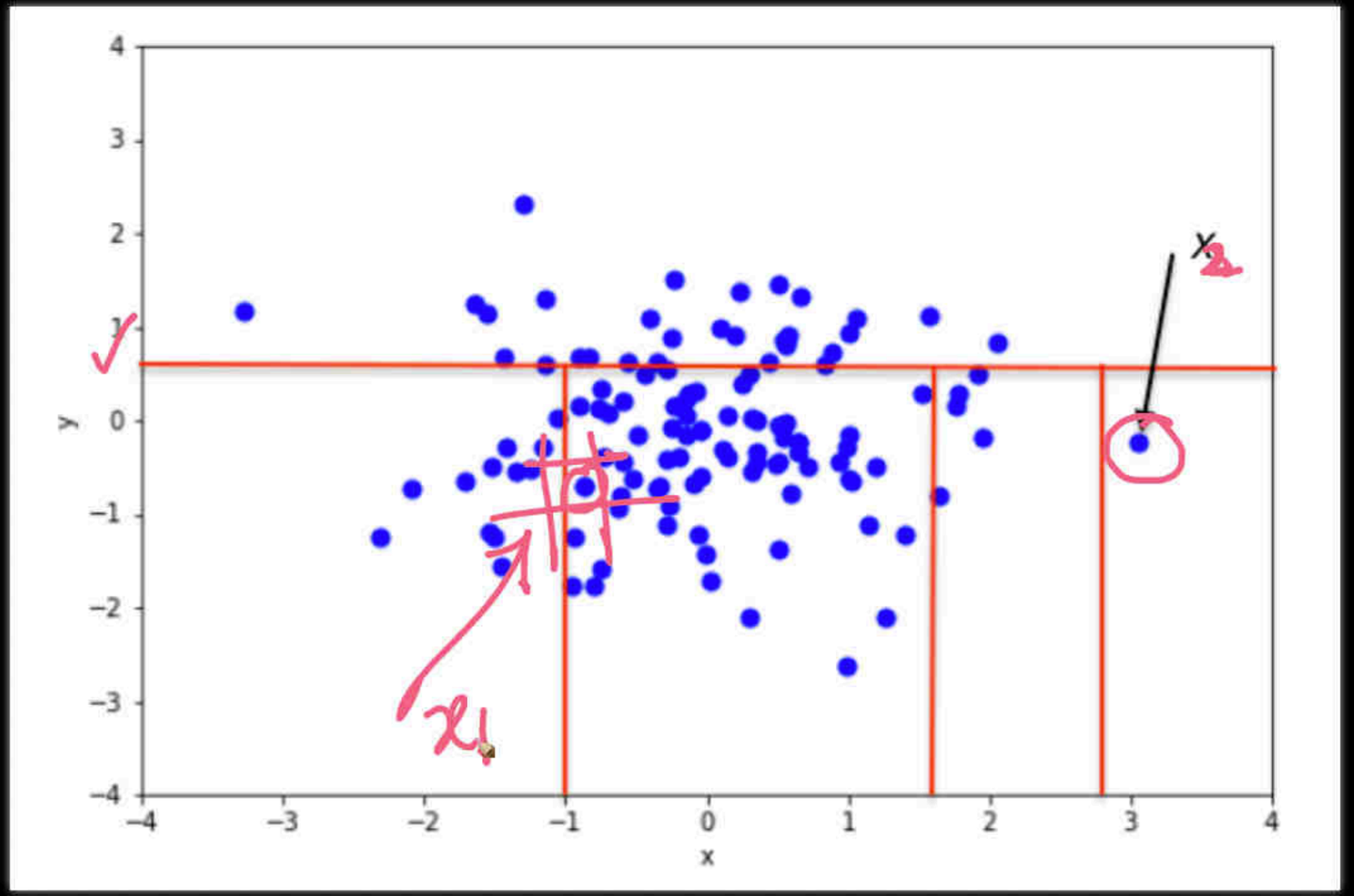


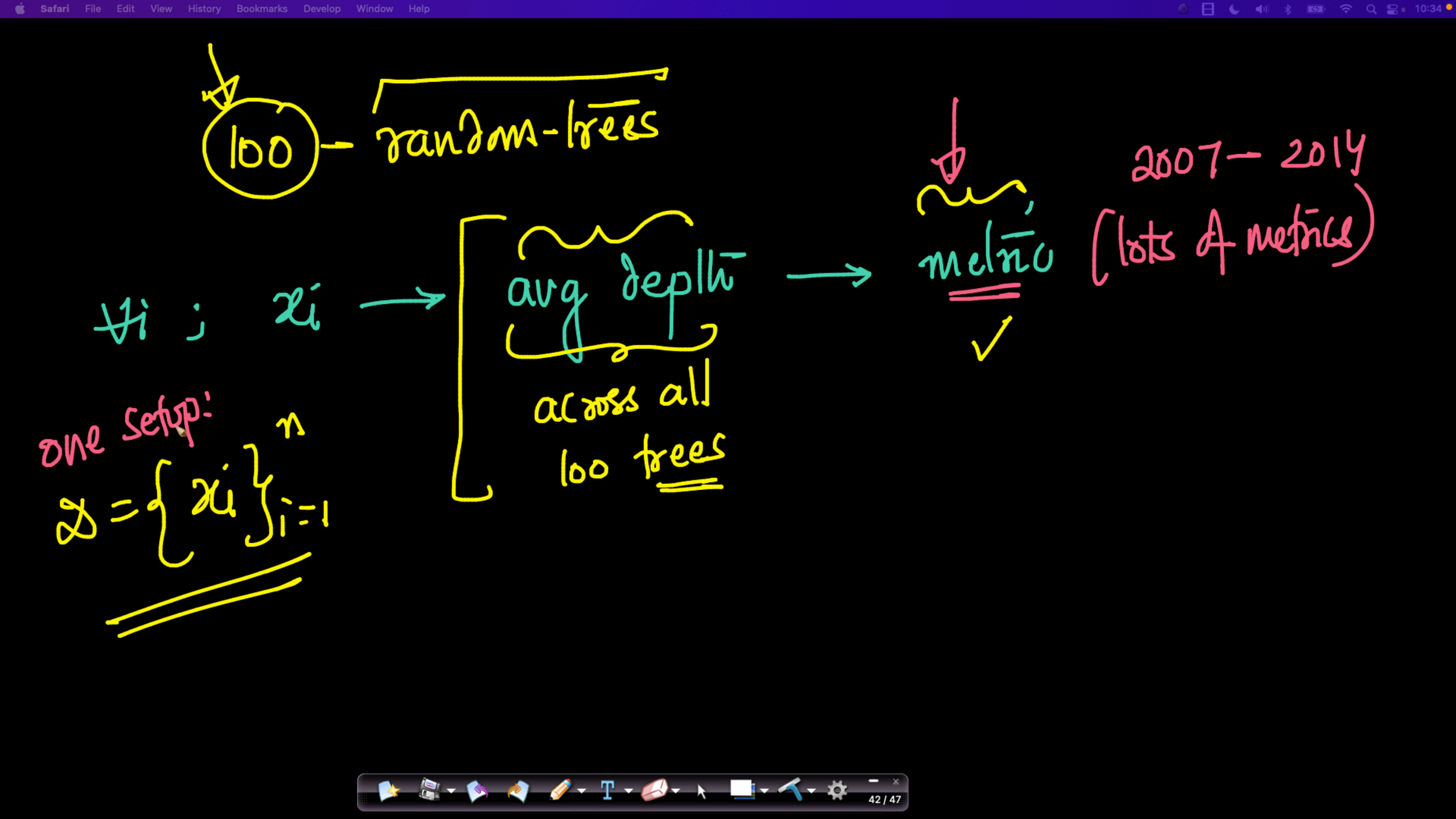
Fig. 3 - an example of isolating an anomalous point in a 2D Gaussian distribution.

Safari File Edit View History Bookmarks Develop Window Help 10:29

Intuition

Isolate each pt ← Isolation forest

- outliers (fewer cuts) → lower depth on an avg in the random trees
- inliers (more cuts) → more depth on an avg, ...


100 - random-trees
∀i ; xi
avg depth
across all
100 trees
metric
2007 - 2014
(lots of metrics)
one setup:
n
x = {xi}i=1

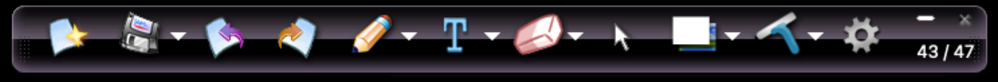
different - setup:

no y_i's

✓ $\mathcal{D}_{Tr} = \{x_i\}_{i=1}^{n}$ → iforest ✓

✓ $\{\mathcal{D}_{Te} = \{x_i\}_{i=1}^{m}$ → outliers

x_q → avg. depth of x_q across all trees in iforest



n = # data points

Sample of $n' < n$

$\mathcal{D}_{TV}$

iForest : density

depth when isolating

$\mathcal{D}_{TE}$

As a consequence, the estimation of average $h(x)$ for external node terminations is the same as that of the unsuccessful searches in BST, that is[11]

$$c(m) = \begin{cases} 2H(m-1) - \frac{2(m-1)}{n} & \text{for } m > 2 \\ 1 & \text{for } m = 2 \\ 0 & \text{otherwise} \end{cases}$$

where n is the testing data size, m is the size of the sample set and H is the harmonic number, which can be estimated by $H(i) = ln(i) + \gamma$, where $\gamma = 0.5772156649$ is the Euler-Mascheroni constant.

The value of c(m) above represents the average of $h(x)$ given m, so we can use it to normalise $h(x)$ and get an estimation of the anomaly score for a given instance x:

$$s(x, m) = 2^{\frac{-E(h(x))}{c(m)}}$$

where $E(h(x))$ is the average value of $h(x)$ from a collection of iTrees. It is interesting to note that for any given instance x:

- if s is close to 1 then x is very likely to be an anomaly
- if s is smaller than 0.5 then x is likely to be a normal value
- if for a given sample all instances are assigned an anomaly score of around 0.5, then it is safe to assume that the sample doesn't have any anomaly

Open source implementations [edit]

Original implementation:

- Isolation Forest, an algorithm that detects data-anomalies using binary trees written in R. Released by the paper's first author Liu, Fei Ton

avg. depth of x_i → 

Outliers

labelled outliers → highly imbalanced binary classfn.

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$$

y_0 : inliers

sklearn.ensemble.IsolationForest

class sklearn.ensemble.**IsolationForest**(*, *n_estimators=100*, *max_samples='auto'*, *contamination='auto'*, *max_features=1.0*, *bootstrap=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*) [source]

Isolation Forest Algorithm.

Return the anomaly score of each sample using the IsolationForest algorithm

The IsolationForest 'isolates' observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node.

This path length, averaged over a forest of such random trees, is a measure of normality and our decision function.

Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees

scikit learn

Prev Up Next

scikit-learn 1.1.1
Other versions

Please cite us if you use the software.

sklearn.ensemble.IsolationForest

Examples using sklearn.ensemble.IsolationForest

collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies.

Read more in the User Guide.

New in version 0.18.

Parameters:

n_estimators : *int, default=100*

The number of base estimators in the ensemble.

max_samples : *"auto", int or float, default="auto"*

The number of samples to draw from X to train each base estimator.

- If int, then draw `max_samples` samples.
- If float, then draw `max_samples * X.shape[0]` samples.
- If "auto", then `max_samples=min(256, n_samples)`.

If max_samples is larger than the number of samples provided, all samples will be used for all trees (no sampling).

contamination : *'auto' or float, default='auto'*

The amount of contamination of the data set, i.e. the proportion of outliers in the data set. Used when fitting to define the threshold on the scores of the samples.

- If 'auto', the threshold is determined as in the original paper.
- If float, the contamination should be in the range (0, 0.5].

New in version 0.18.

Parameters:

n_estimators : *int, default=100*

The number of base estimators in the ensemble.

max_samples : *"auto", int or float, default="auto"*

The number of samples to draw from X to train each base estimator.

- If int, then draw `max_samples` samples.
- If float, then draw `max_samples * X.shape[0]` samples.
- If "auto", then `max_samples=min(256, n_samples)`.

If max_samples is larger than the number of samples provided, all samples will be used for all trees (no sampling).

contamination : *'auto' or float, default='auto'*

The amount of contamination of the data set, i.e. the proportion of outliers in the data set. Used when fitting to define the threshold on the scores of the samples.

- If 'auto', the threshold is determined as in the original paper.
- If float, the contamination should be in the range (0, 0.5].

Changed in version 0.22: The default value of `contamination` changed from 0.1 to `'auto'`.

...0

...number of features to draw from X to train each base estimator.

scikit learn
Prev Up Next
scikit-learn 1.1.1
Other versions
Please cite us if you use the software.
sklearn.ensemble.IsolationForest
Examples using sklearn.ensemble.IsolationForest

1 Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. "Isolation forest." Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on.

2 Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. "Isolation-based anomaly detection." ACM Transactions on Knowledge Discovery from Data (TKDD) 6.1 (2012): 3.

Examples

```
>>> from sklearn.ensemble import IsolationForest
>>> X = [[-1.1], [0.3], [0.5], [100]]
>>> clf = IsolationForest(random_state=0).fit(X)
>>> clf.predict([[0.1], [0], [90]])
array([ 1,  1, -1])
```

Methods

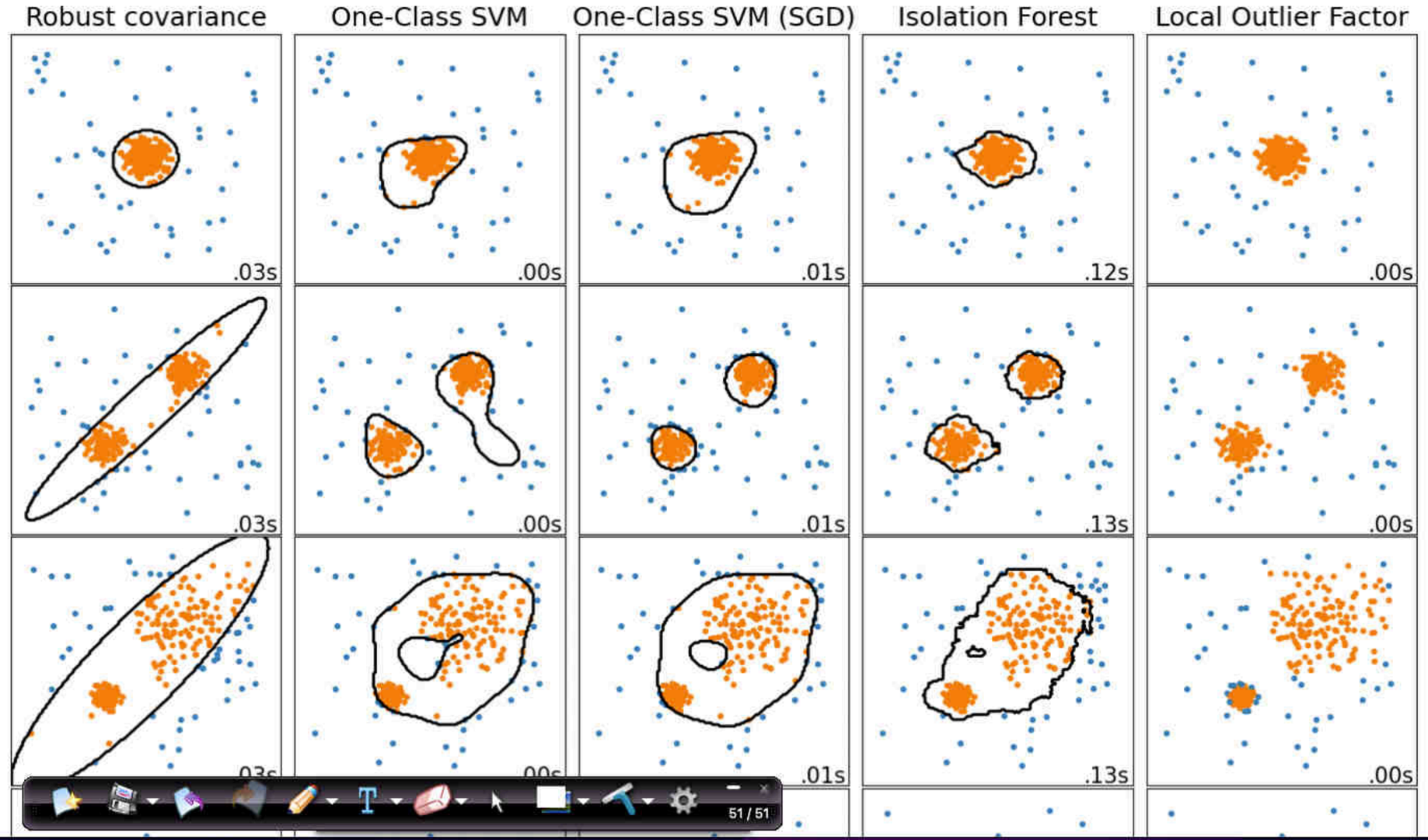
decision_function(X)	Average anomaly score of X of the base classifiers.
fit(X[, y, sample_weight])	Fit estimator.
fit_predict(X[, y])	Perform fit on X and returns labels for X.
get_params([deep])	Get parameters for this estimator.
predict(X)	Predict if a particular sample is an outlier or not.
score_samples(X)	Opposite of the anomaly score defined in the original paper.
set_params(**params)	Set the parameters of this estimator.

[source]

scikit learn
Prev Up Next
scikit-learn 1.1.1
Other versions
Please cite us if you use the software.
Comparing anomaly detection algorithms for outlier detection on toy datasets

While these examples give some intuition about the algorithms, this intuition might not apply to very high dimensional data.

Finally, note that parameters of the models have been here handpicked but that in practice they need to be adjusted. In the absence of labelled data, the problem is completely unsupervised so model selection can be a challenge.

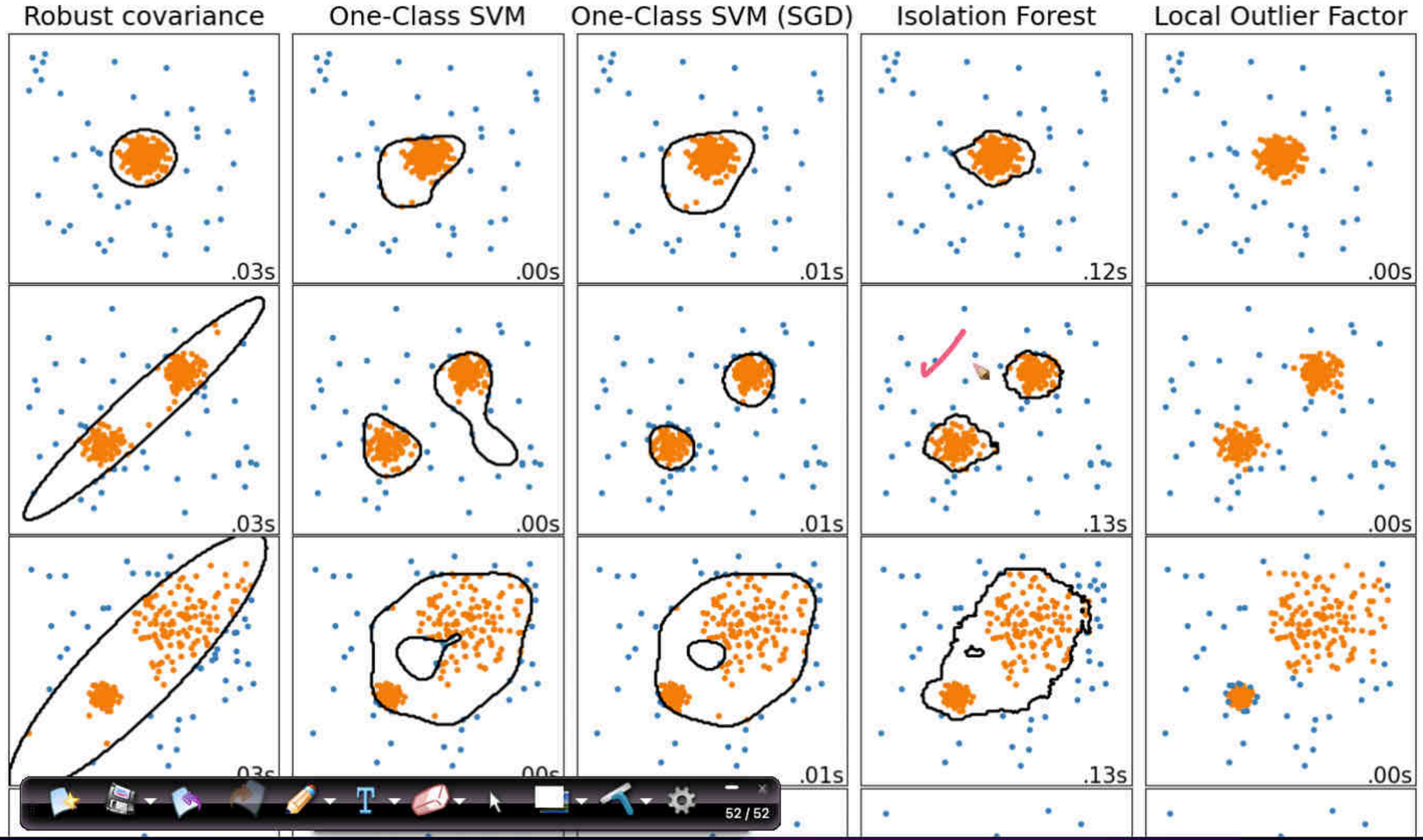


Toggle Menu

scikit learn
Prev Up Next
scikit-learn 1.1.1
Other versions
Please cite us if you use the software.
Comparing anomaly detection algorithms for outlier detection on toy datasets

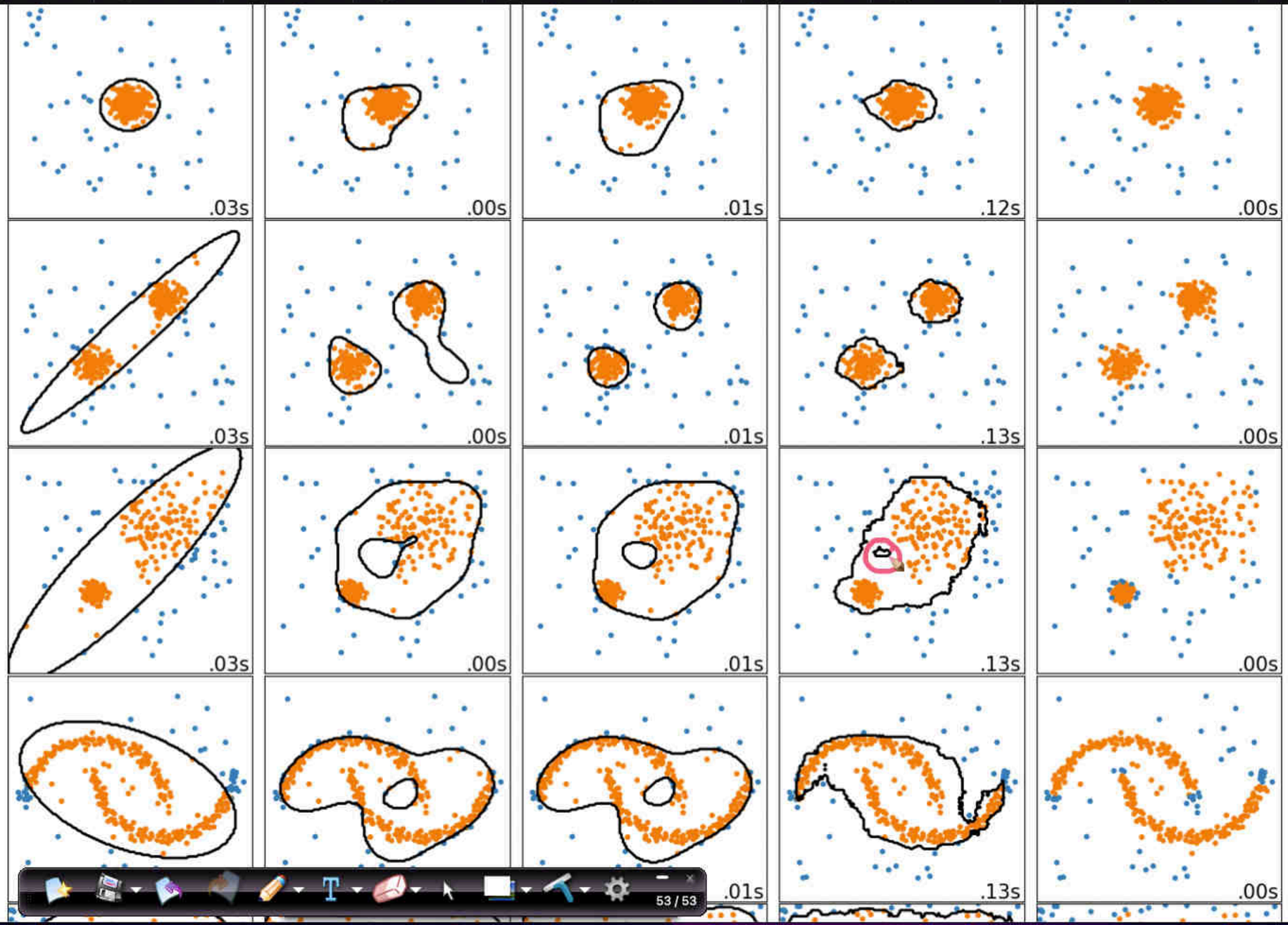
While these examples give some intuition about the algorithms, this intuition might not apply to very high dimensional data.

Finally, note that parameters of the models have been here handpicked but that in practice they need to be adjusted. In the absence of labelled data, the problem is completely unsupervised so model selection can be a challenge.

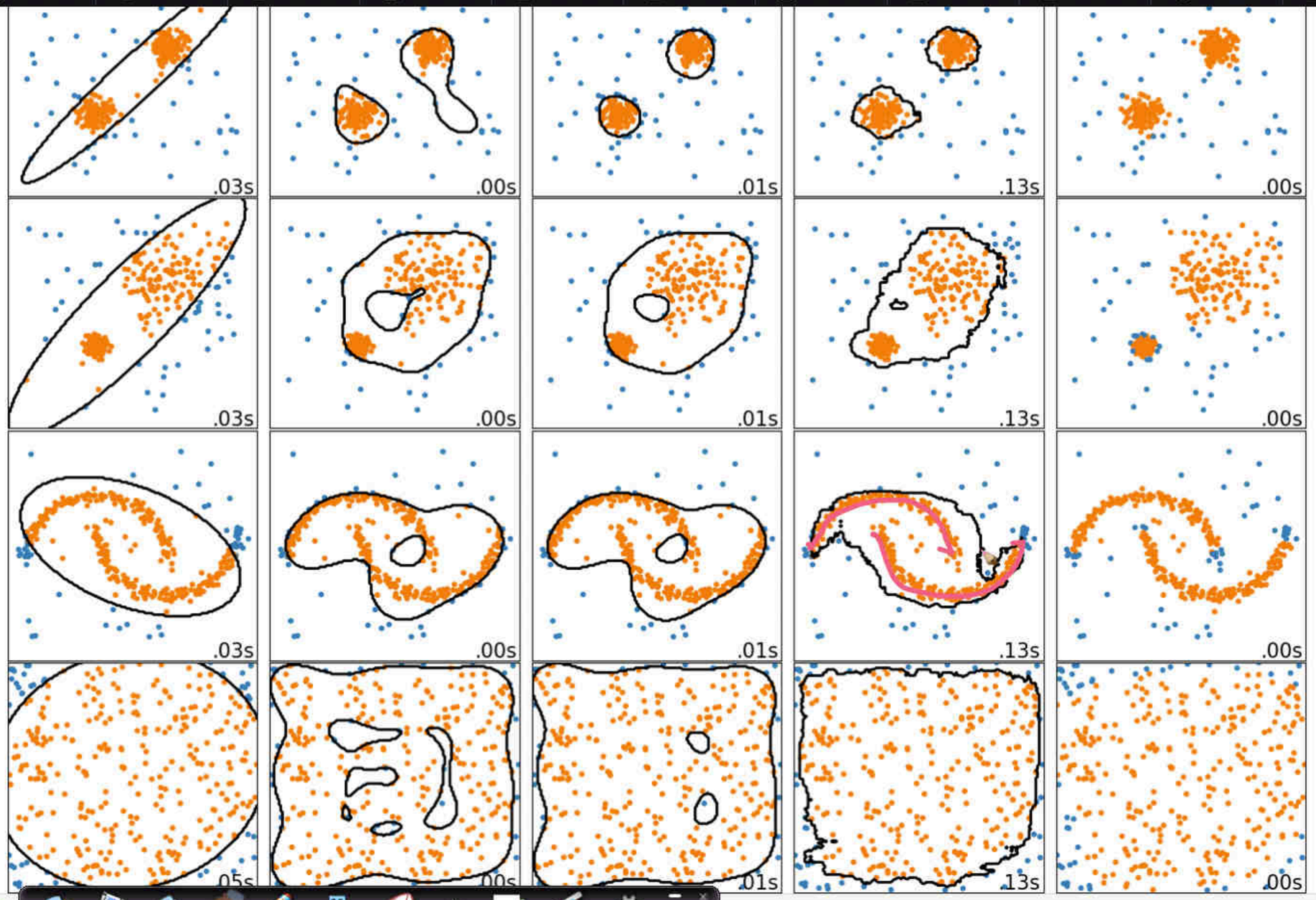


Toggle Menu

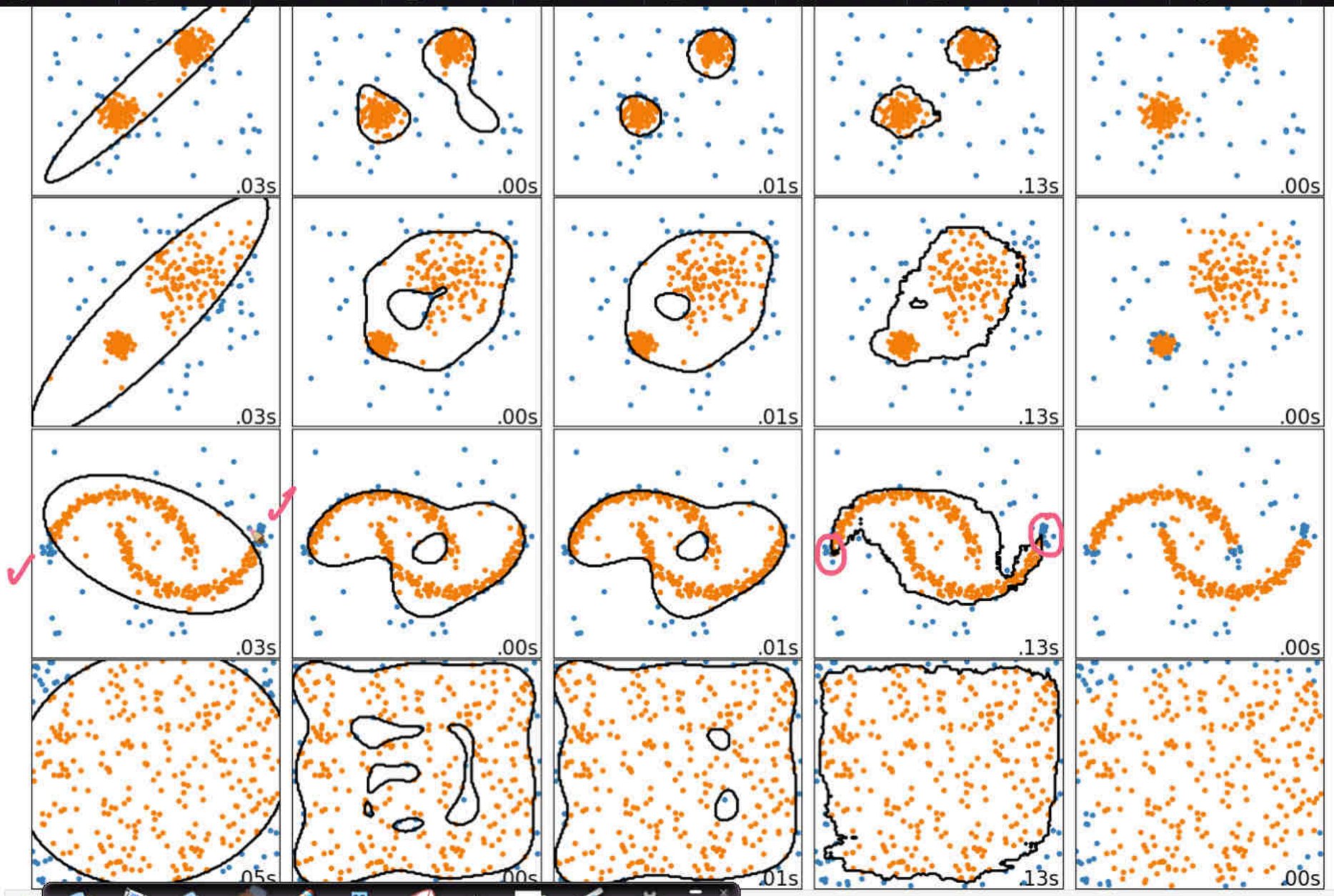
Comparing anomaly detection algorithms for outlier detection on toy datasets



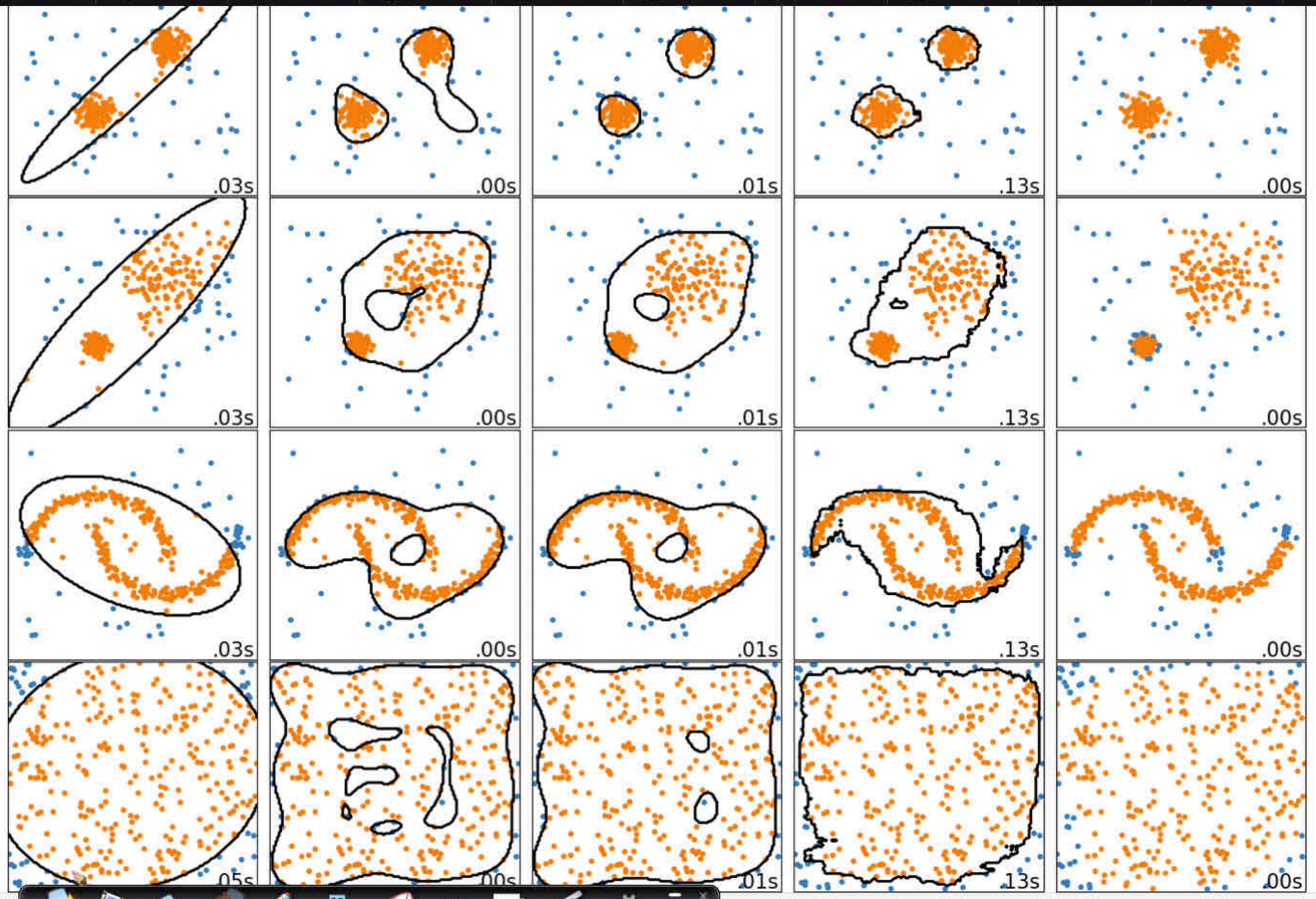
Comparing anomaly detection algorithms for outlier detection on toy datasets



Comparing anomaly detection algorithms for outlier detection on toy datasets



Comparing anomaly detection algorithms for outlier detection on toy datasets



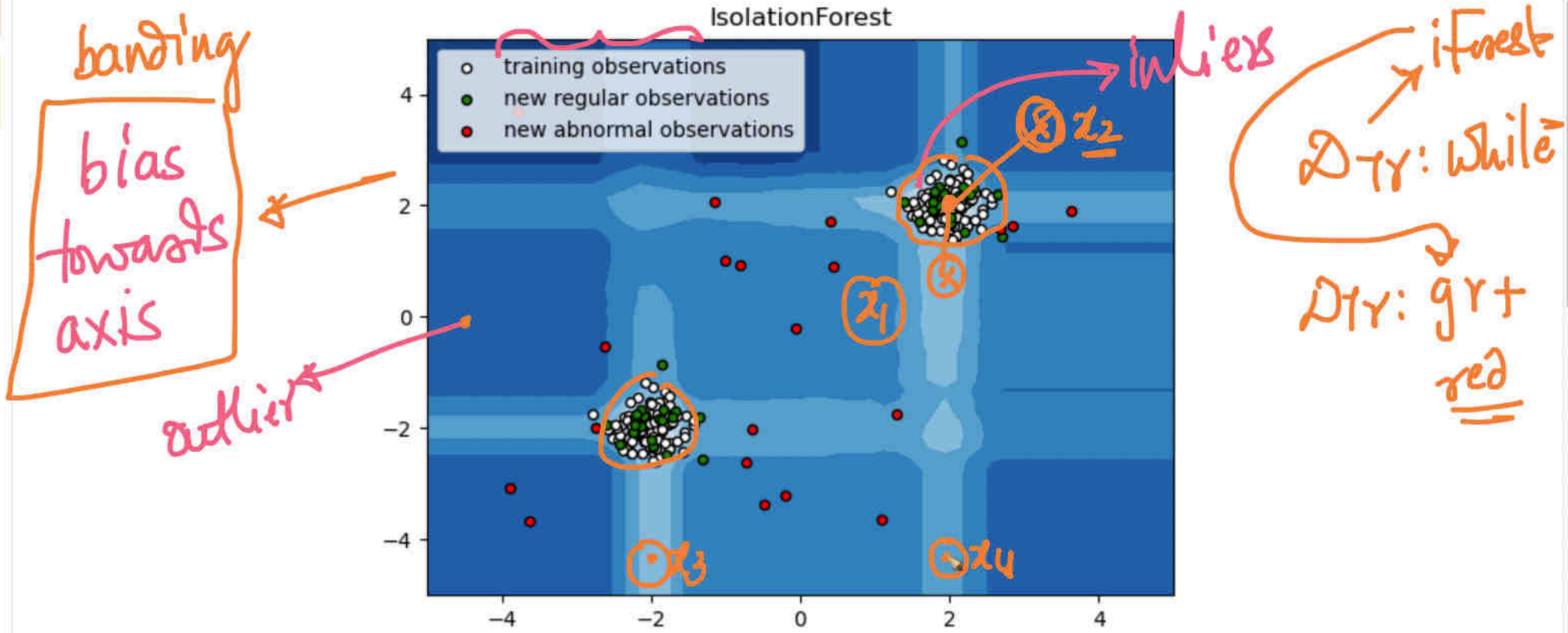
disadv

↳ contamination - param → heuristics

↳ axis parallel splits

scikit learn
Prev Up Next
scikit-learn 1.1.1
Other versions
Please cite us if you use the software.
IsolationForest example

Random partitioning produces noticeable shorter paths for anomalies. Hence, when a forest of random trees collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies.




```
import numpy as np
import matplotlib.pyplot as plt
fr
```

$$\min_{C,\gamma,\xi_i} \ \gamma^2 + \lambda \sum_{i=1}^{n} \xi_i$$

hyper-param

slack

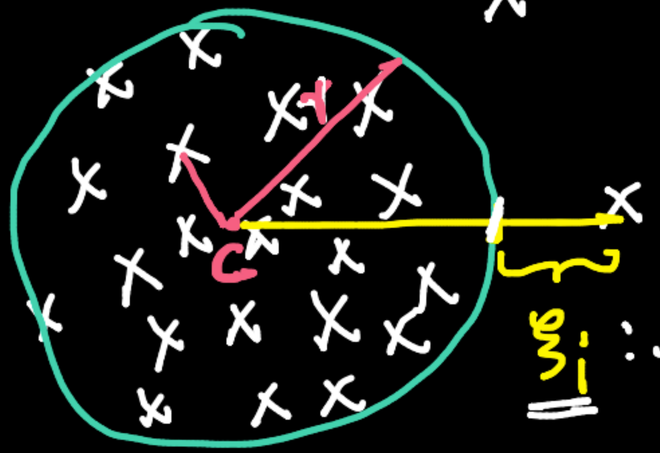
$$\text{s.t} \quad \text{dist}^2(x_i, C) \le \gamma^2 + \xi_i^2 \quad \forall i$$

$$\xi_i \ge 0$$

Soft-Margin

RBF Kernalize this

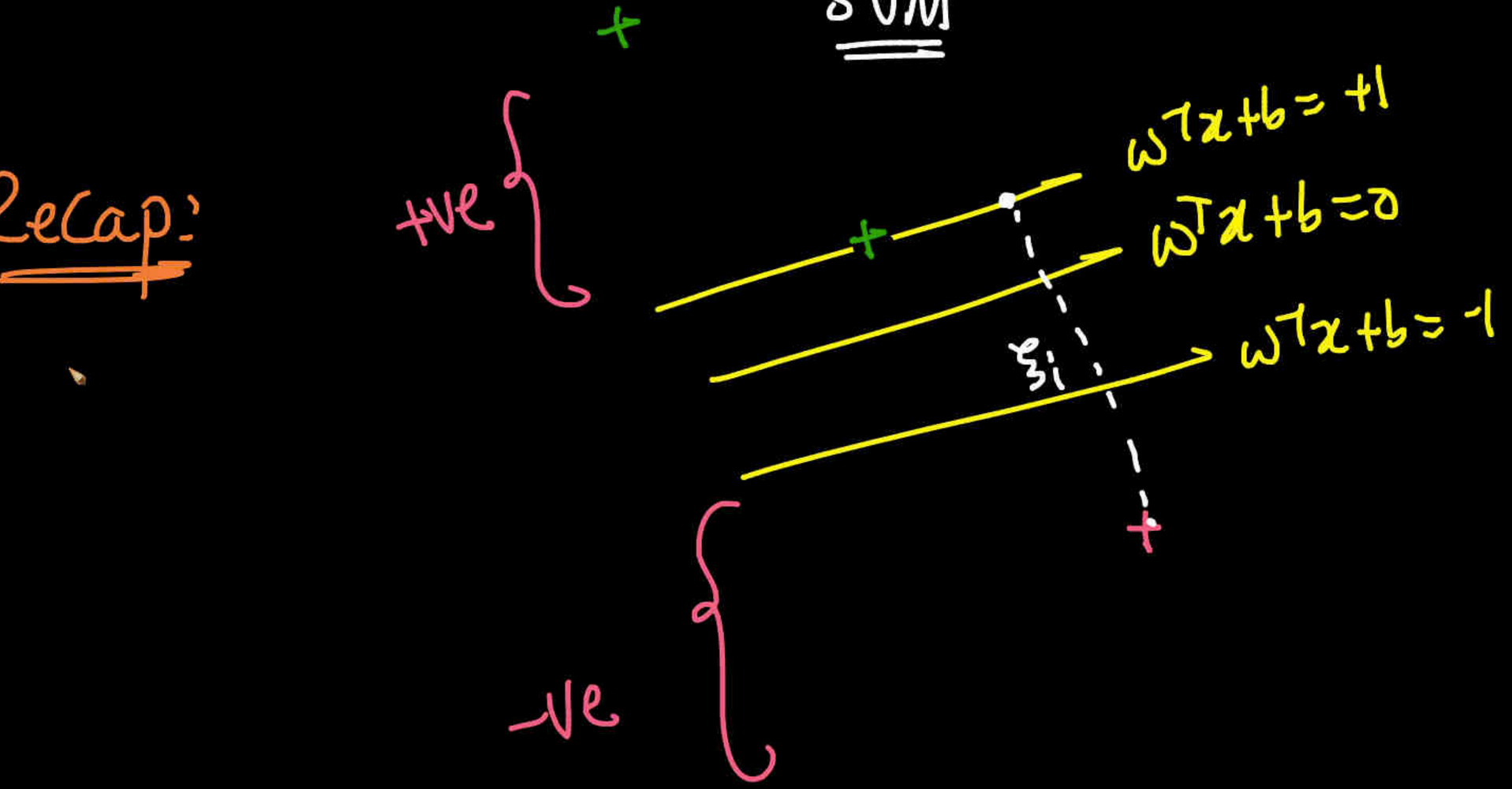
One-class SVM → Kernel trick

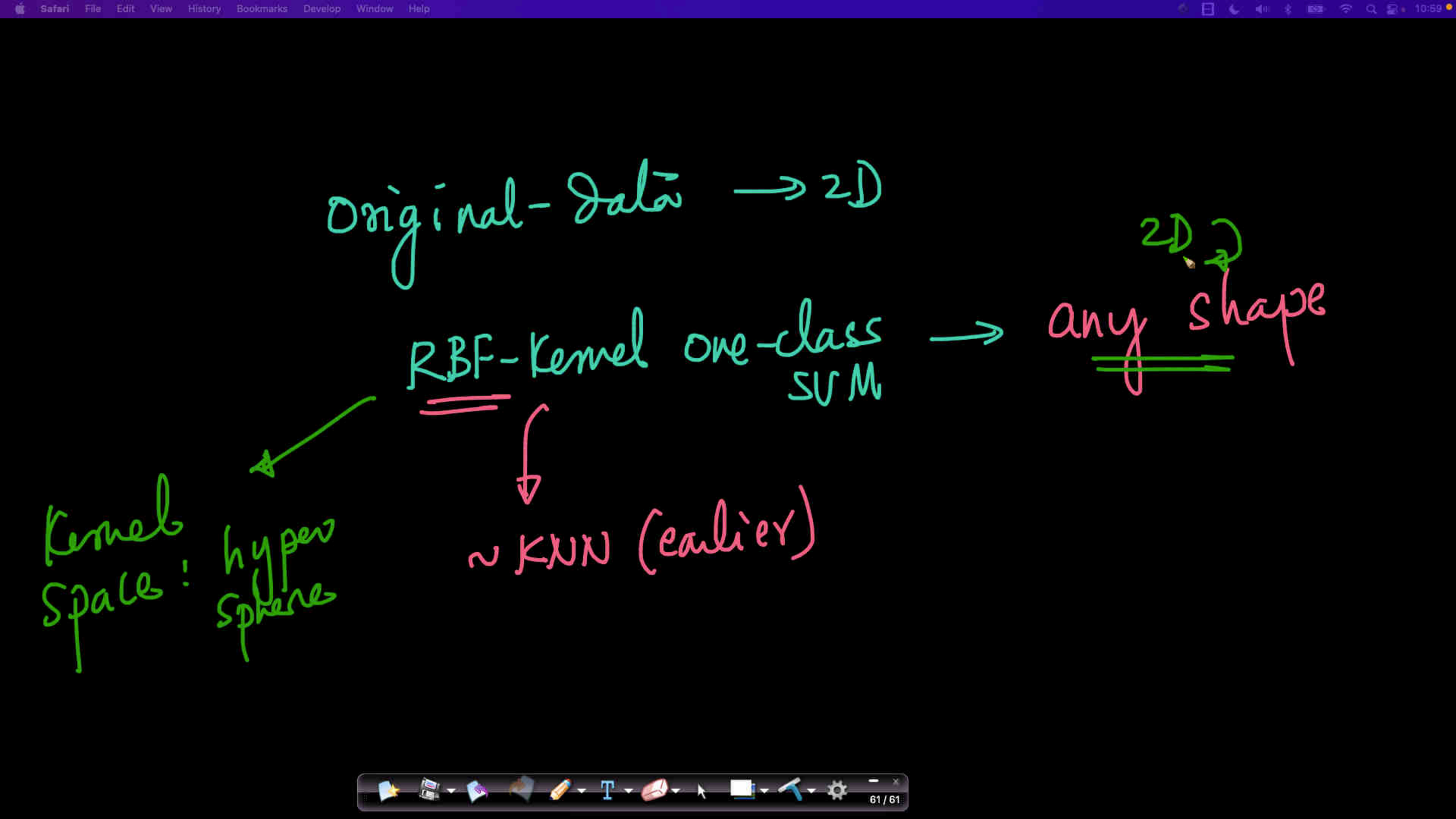


$$\xi_i = \begin{cases} 0 & \text{if } d(x_i, C) \le \gamma \\ d(x_i, C) - \gamma & \text{o/w} \end{cases}$$

SVM

Recap:




Original-Data → 2D
RBF-Kernel one-class SVM → any shape
2D
Kernel space: hyper sphere
~ KNN (earlier)
61 / 61

Safari File Edit View History Bookmarks Develop Window Help

disadv

→ Kernel Selection (RBF: default)

→ as n↑ : time complx increases → SGD

→ SVMs disadvantages

sklearn.svm.OneClassSVM

class sklearn.svm.**OneClassSVM**(*, *kernel='rbf'*, *degree=3*, *gamma='scale'*, *coef0=0.0*, *tol=0.001*, *nu=0.5*, *shrinking=True*, *cache_size=200*, *verbose=False*, *max_iter=-1*) [source]

Unsupervised Outlier Detection.

Estimate the support of a high-dimensional distribution.

The implementation is based on libsvm.

Read more in the User Guide.

Parameters:

kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'*
Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to precompute the kernel matrix.

degree : *int, default=3*
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : *{'scale', 'auto'} or float, default='scale'*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto' uses 1 / n_features

scikit learn
Prev Up Next
scikit-learn 1.1.1
Other versions
Please cite us if you use the software.
sklearn.svm.OneClassSVM
Examples using sklearn.svm.OneClassSVM

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features.

Changed in version 0.22: The default value of `gamma` changed from 'auto' to 'scale'.

coef0 : *float, default=0.0*
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

tol : *float, default=1e-3*
Tolerance for stopping criterion.

nu : *float, default=0.5*
An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. Should be in the interval (0, 1]. By default 0.5 will be taken.

shrinking : *bool, default=True*
Whether to use the shrinking heuristic. See the User Guide.

cache_size : *float, default=200*
Specify the size of the kernel cache (in MB).

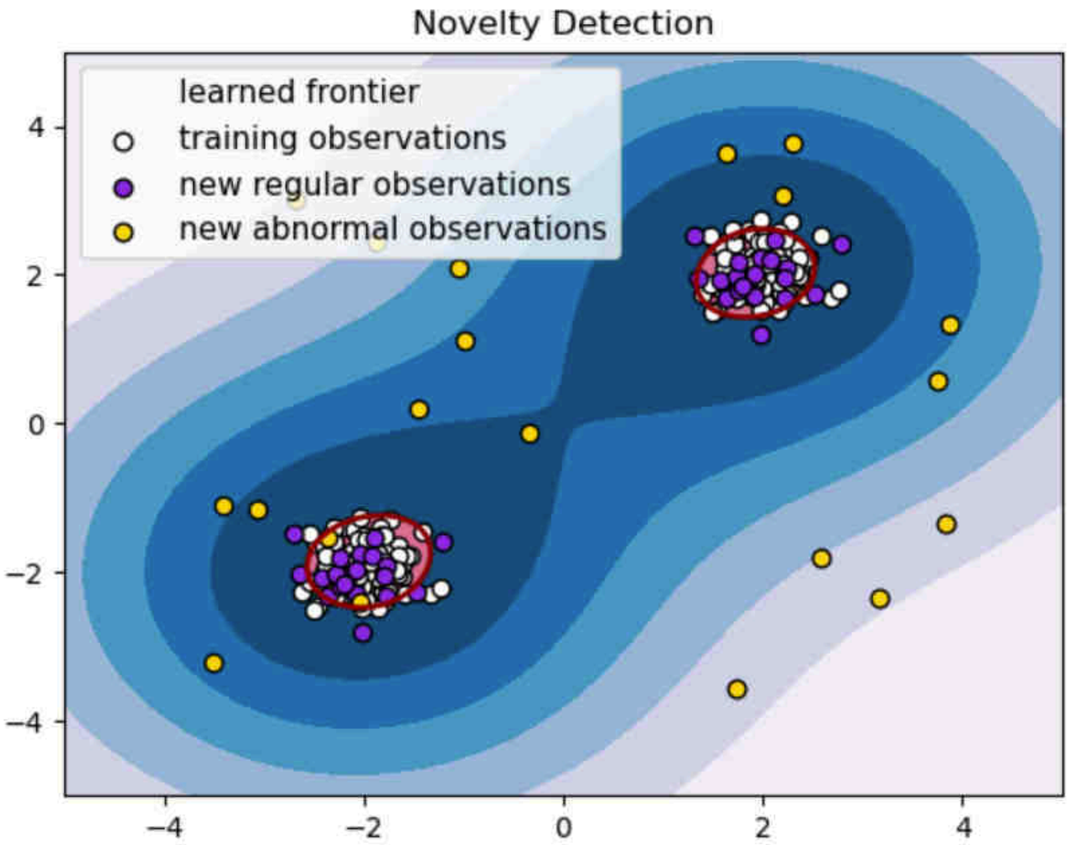
verbose : *bool, default=False*
Enable verbose output. Note that this setting takes advantage of a per-process runtime setting in libsvm that, if enabled, may not work properly in a multithreaded context.

max_iter : *int, default=-1*
Hard limit on iterations within solver, or -1 for no limit.

scikit learn
Prev Up Next
scikit-learn 1.1.1
Other versions
Please cite us if you use the software.
One-class SVM with non-linear kernel (RBF)

An example using a one-class SVM for novelty detection.

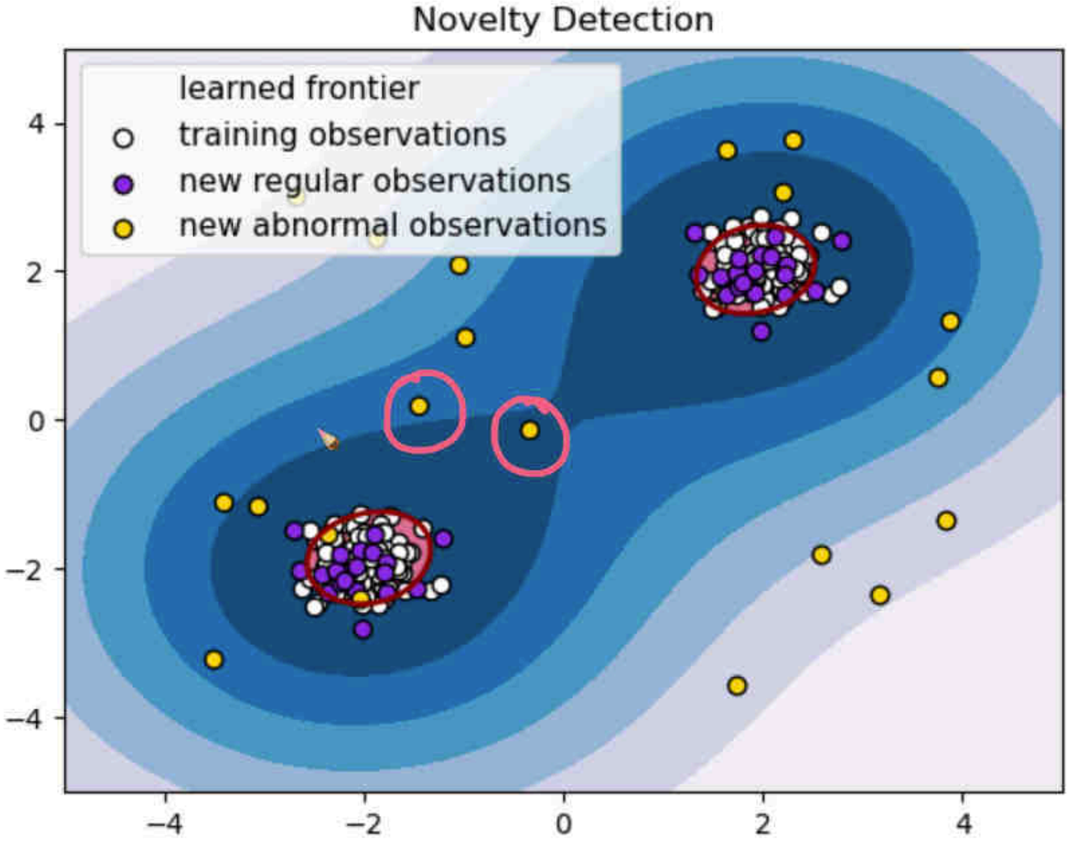
One-class SVM is an unsupervised algorithm that learns a decision function for novelty detection: classifying new data as similar or different to the training set.



RBF

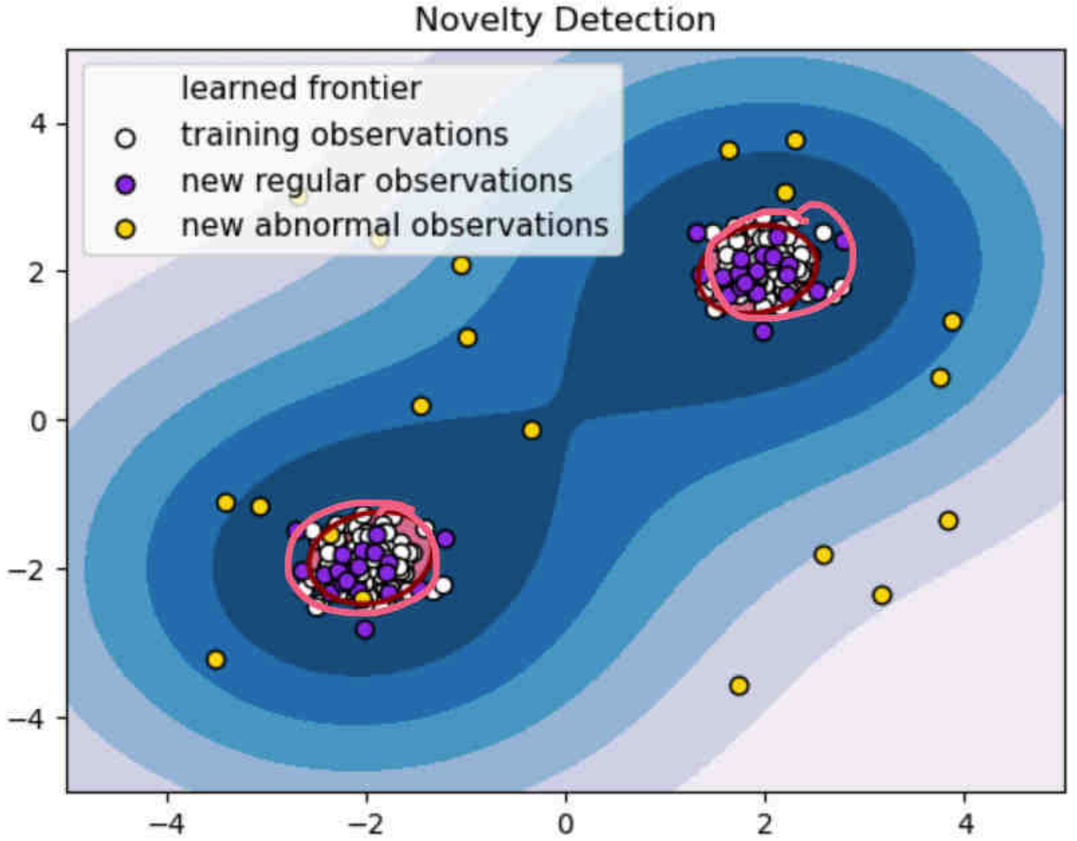
D_{tr} → 1class SVM

D_{te} : purple + yellow



error train: 20/200 ; errors novel regular: 9/40 ; errors novel abnormal: 2/40

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.font_manager
from sklearn import svm

xx, yy = np.meshgrid(np.linspace(-5, 5, 500), np.linspace(-5, 5, 500))
#
X = 0.3 * np.random.randn(100, 2)
```



error train: 20/200 ; errors novel regular: 9/40 ; errors novel abnormal: 2/40

RBF → KNN

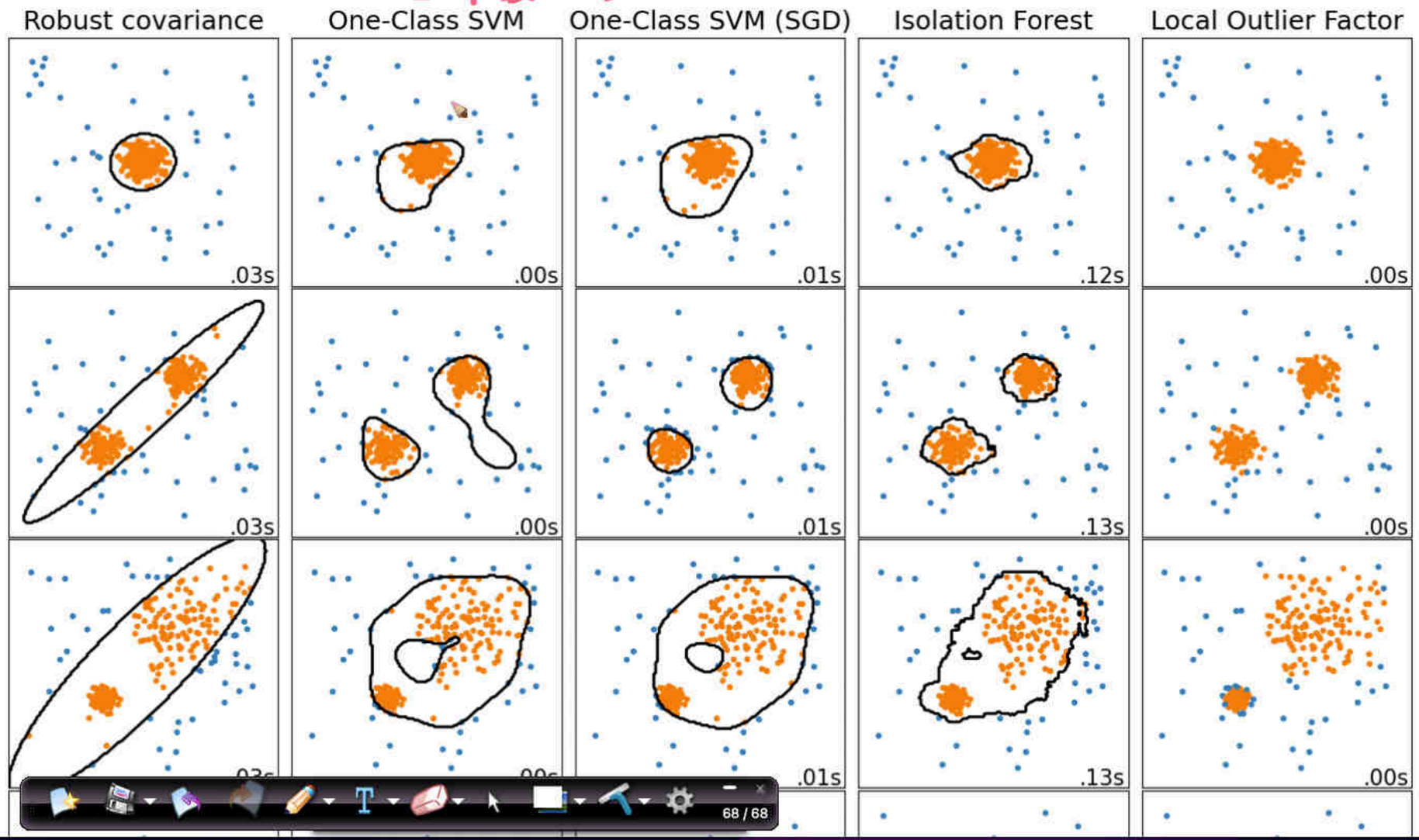
U-complex Shapes

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.font_manager
from sklearn import svm

xx, yy = np.meshgrid(np.linspace(-5, 5, 500), np.linspace(-5, 5, 500))
#
X = 0.3 * np.random.randn(100, 2)
```

scikit learn
Prev Up Next
scikit-learn 1.1.1
Other versions
Please cite us if you use the software.
Comparing anomaly detection algorithms for outlier detection on toy datasets

While these examples give some intuition about the algorithms, this intuition might not apply to very high dimensional data.

Finally, note that parameters of the models have been here handpicked but that in practice they need to be adjusted. In the absence of labelled data, the problem is completely unsupervised so model selection can be a challenge.



Toggle Menu

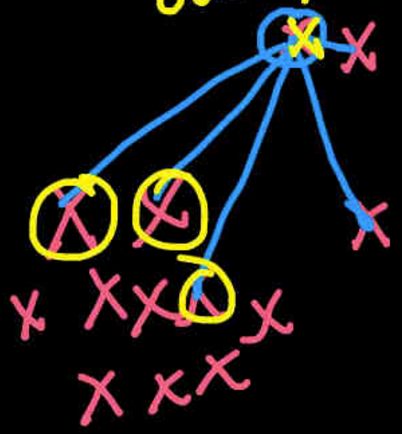
Local outlier Factor

Intuition

[Compare a pts density with its K neighbors → low / high]

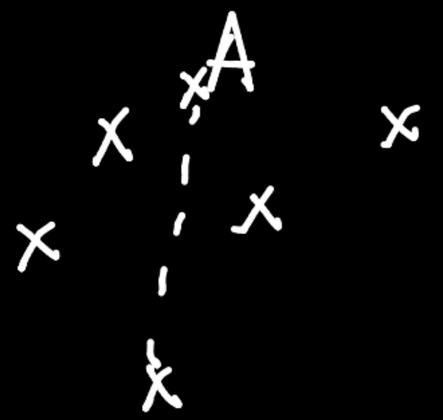
[KNN density]

outlier → density-layer



(a) ① k-dist(A) = distance from A to its k^{th}-NN

↳ high



(1b) Set: $N_k(A)$ = set of k-nearest neighbors of A

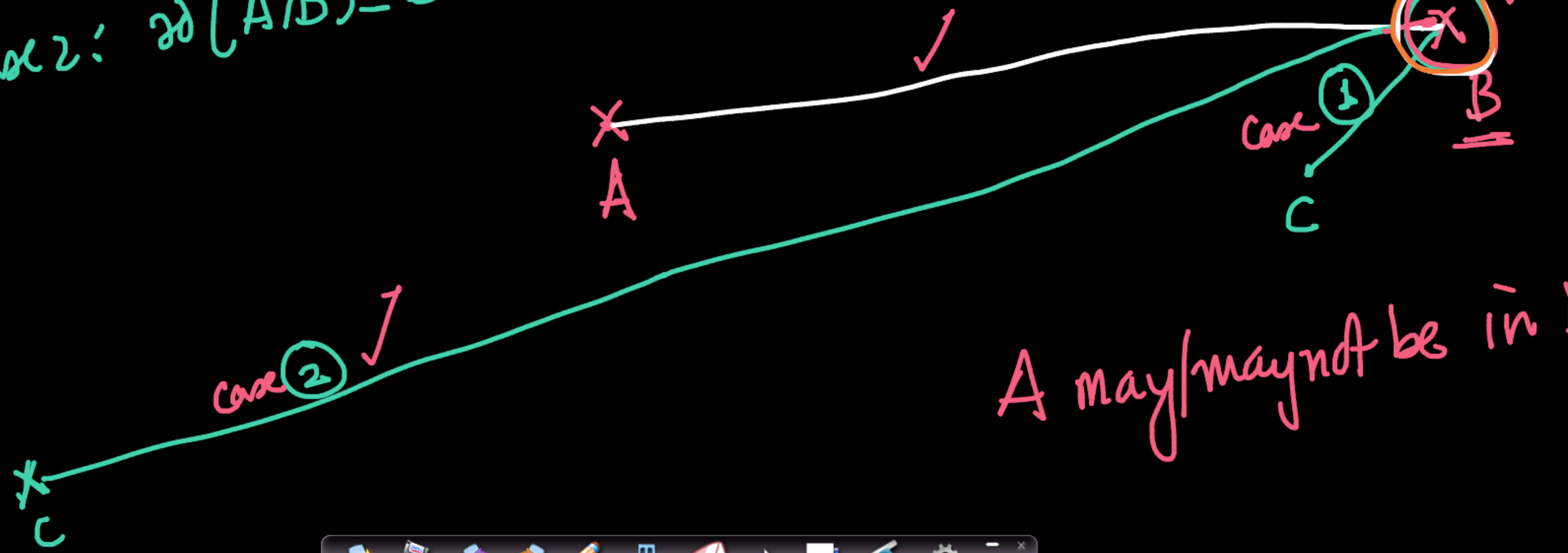
dist (2)

reachability-dist(A, B) = rd(A, B)

= Max { dist(A, B), kDist(B) }

Case 1: rd(A, B) = dist(A, B)

Case 2: rd(A, B) = BC



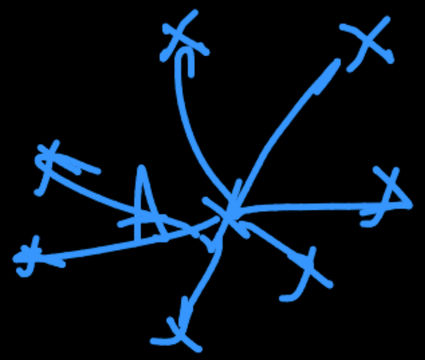
A may/maynot be in $N_k(B)$

density

local-reachability-density$_k(A)$ = lrd$_k(A)$

$$lrd_k(A) = \frac{\sum_{B \in N_k(A)} rd_k(A,B)}{|N_k(A)|}$$

: avg (rd) between A & k-neighbors



$$LOF_k(A) = \frac{\sum_{B \in N_k(A)} lrd_k(B)}{|N_k(A)|} \Big/ lrd_k(A)$$

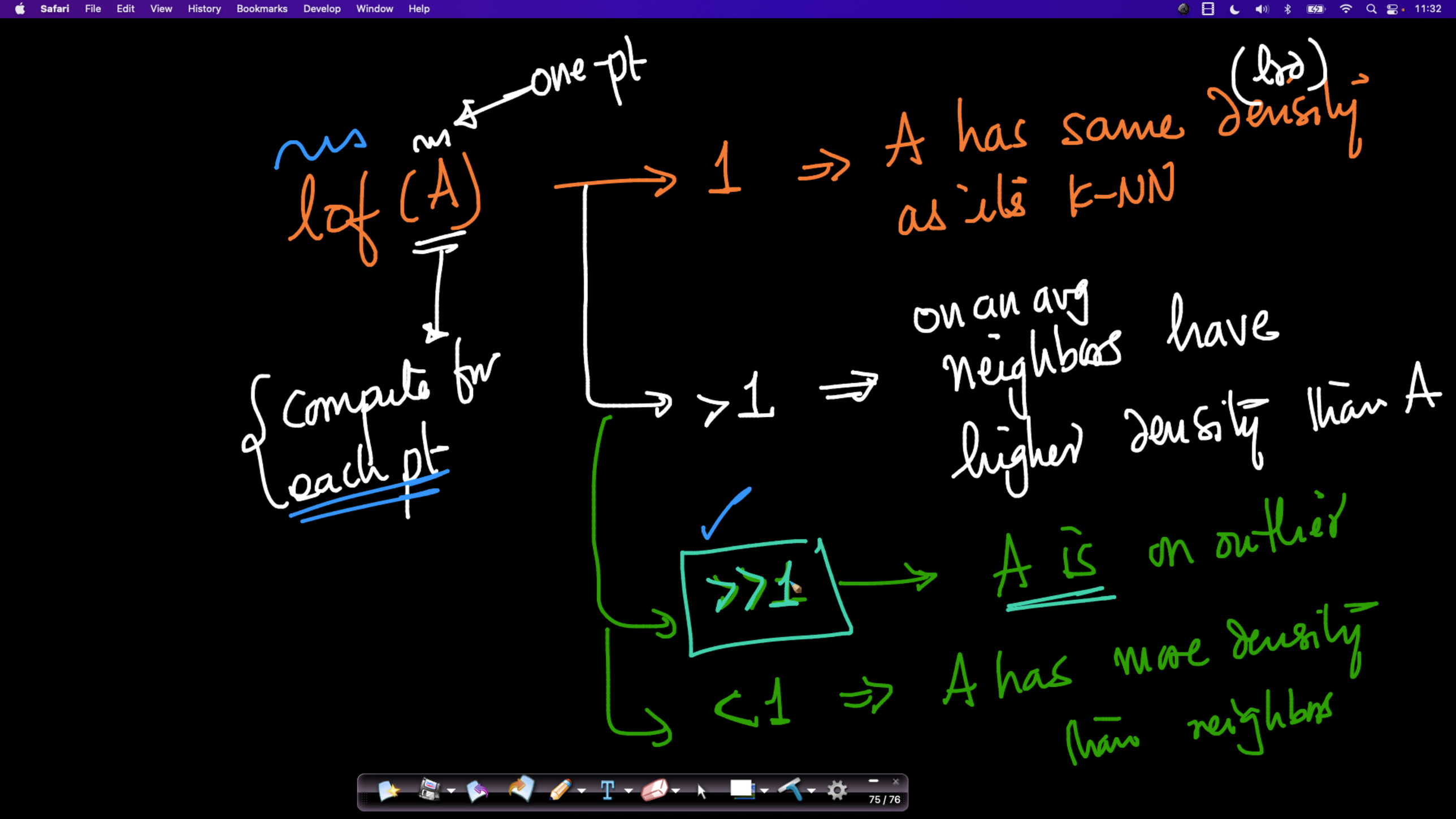
avg. neighborhood density

$$\frac{\text{avg-neighborhood-density (lrd) of } A}{\text{density (lrd) of } A}$$

(< 1)

$(> 1\ ?)$

(1)

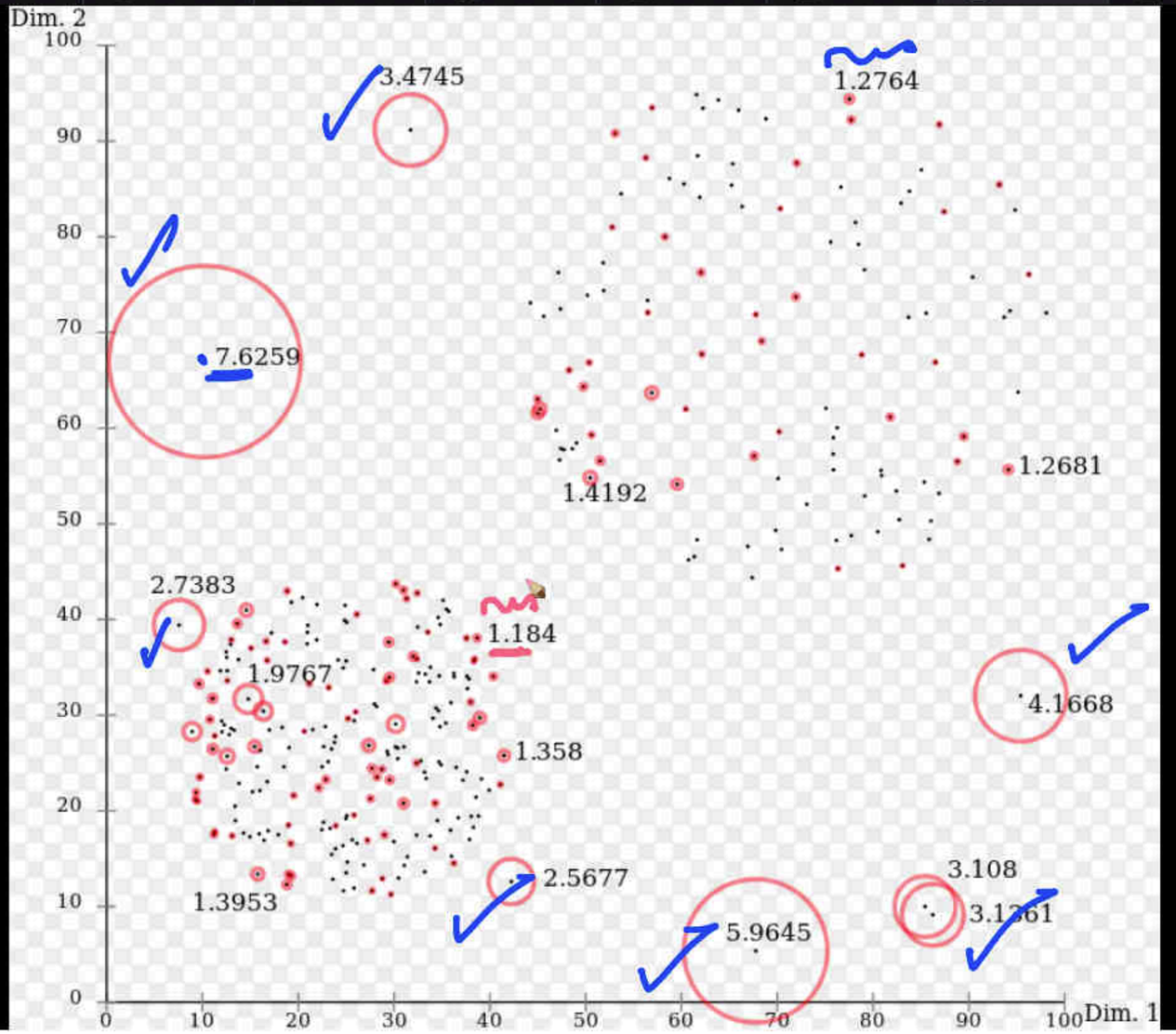

Safari File Edit View History Bookmarks Develop Window Help
11:32
one pt
lof (A)
Compute for each pt
= 1 ⇒ A has same density as its k-NN
(lrd)
> 1 ⇒ on an avg neighbors have higher density than A
>> 1
A is an outlier
< 1 ⇒ A has more density than neighbors
75 / 76

disadv

→ K ; LOF(A) >> 1 (Th) ✓

→ high dim → (non-euclidean)

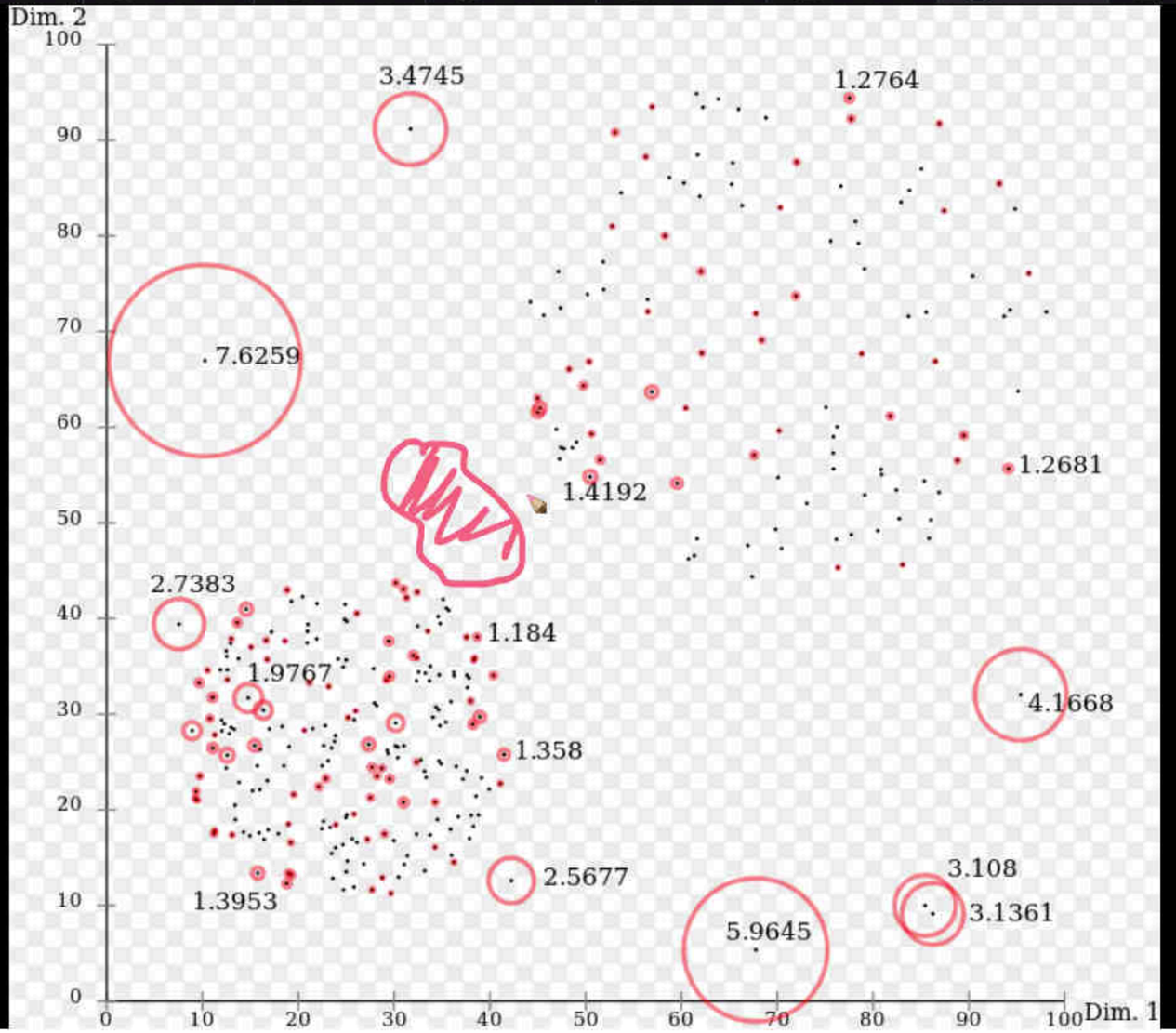
→ time-complx (KNN for each x_i)



∀i:
LOF(xi)

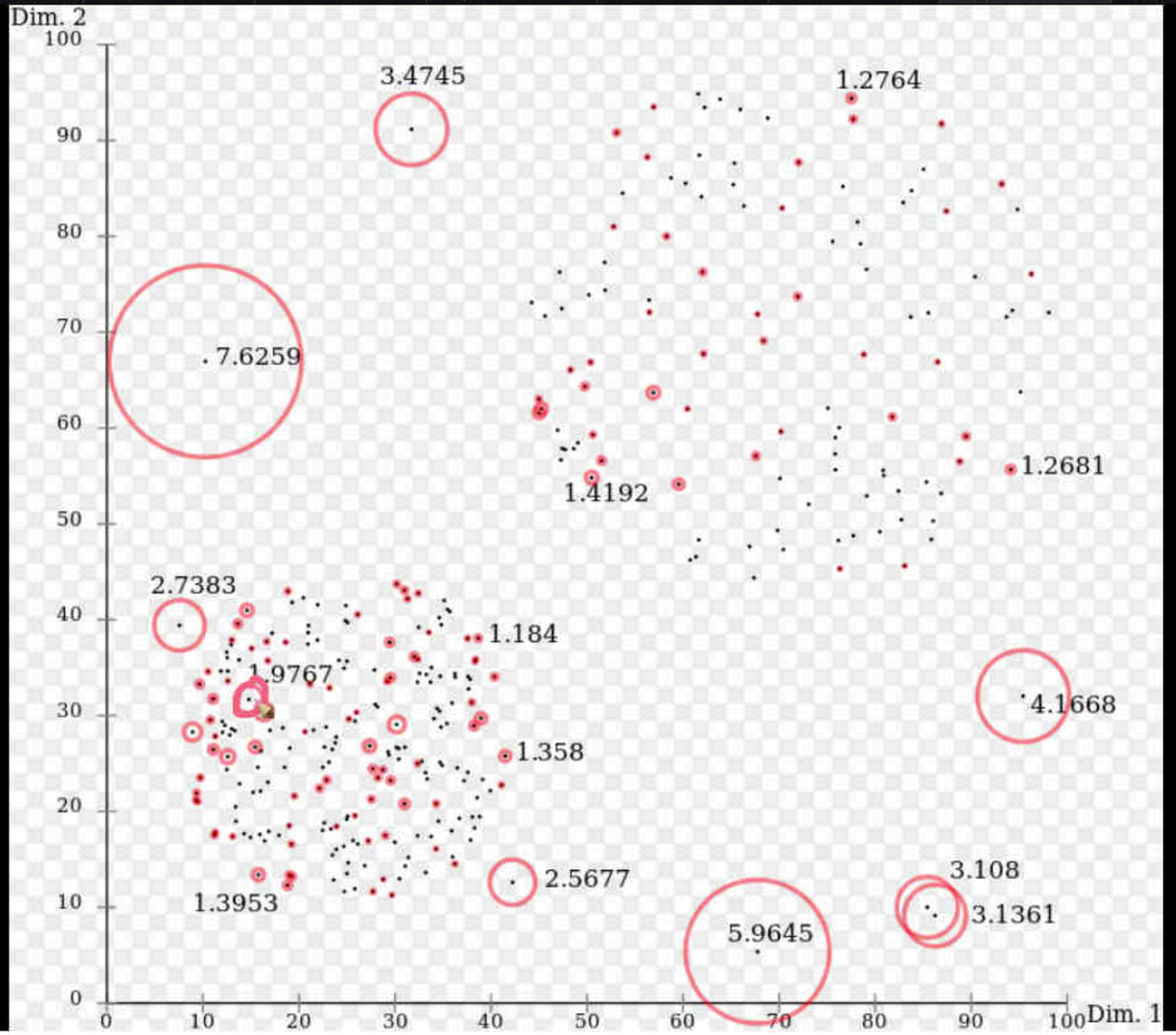
LOF scores as visualized by ELKI. While the upper right cluster has a comparable density to the outliers close to the bottom left cluster, they are detected correctly.

More details



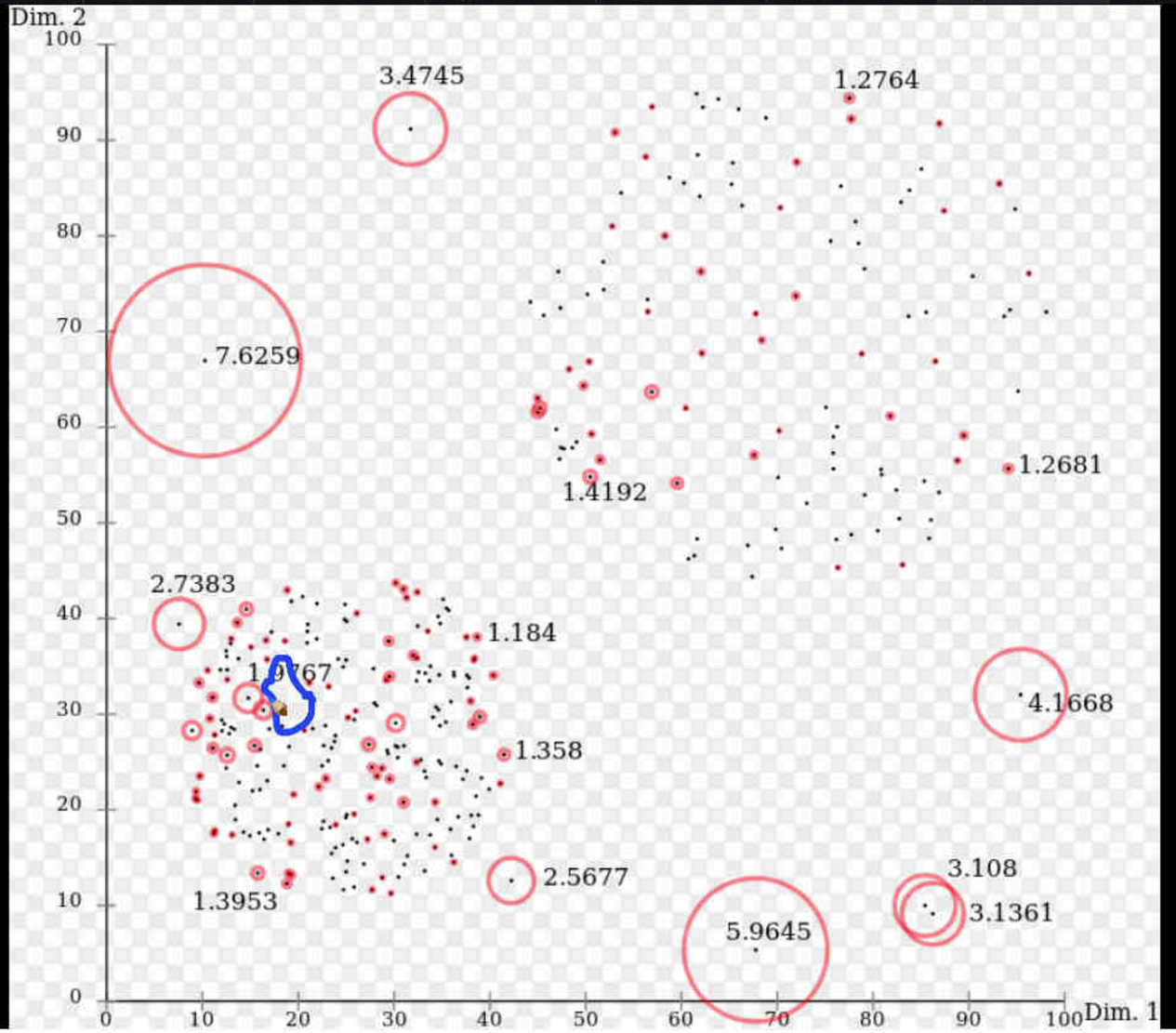
LOF scores as visualized by ELKI. While the upper right cluster has a comparable density to the outliers close to the bottom left cluster, they are detected correctly.

More details



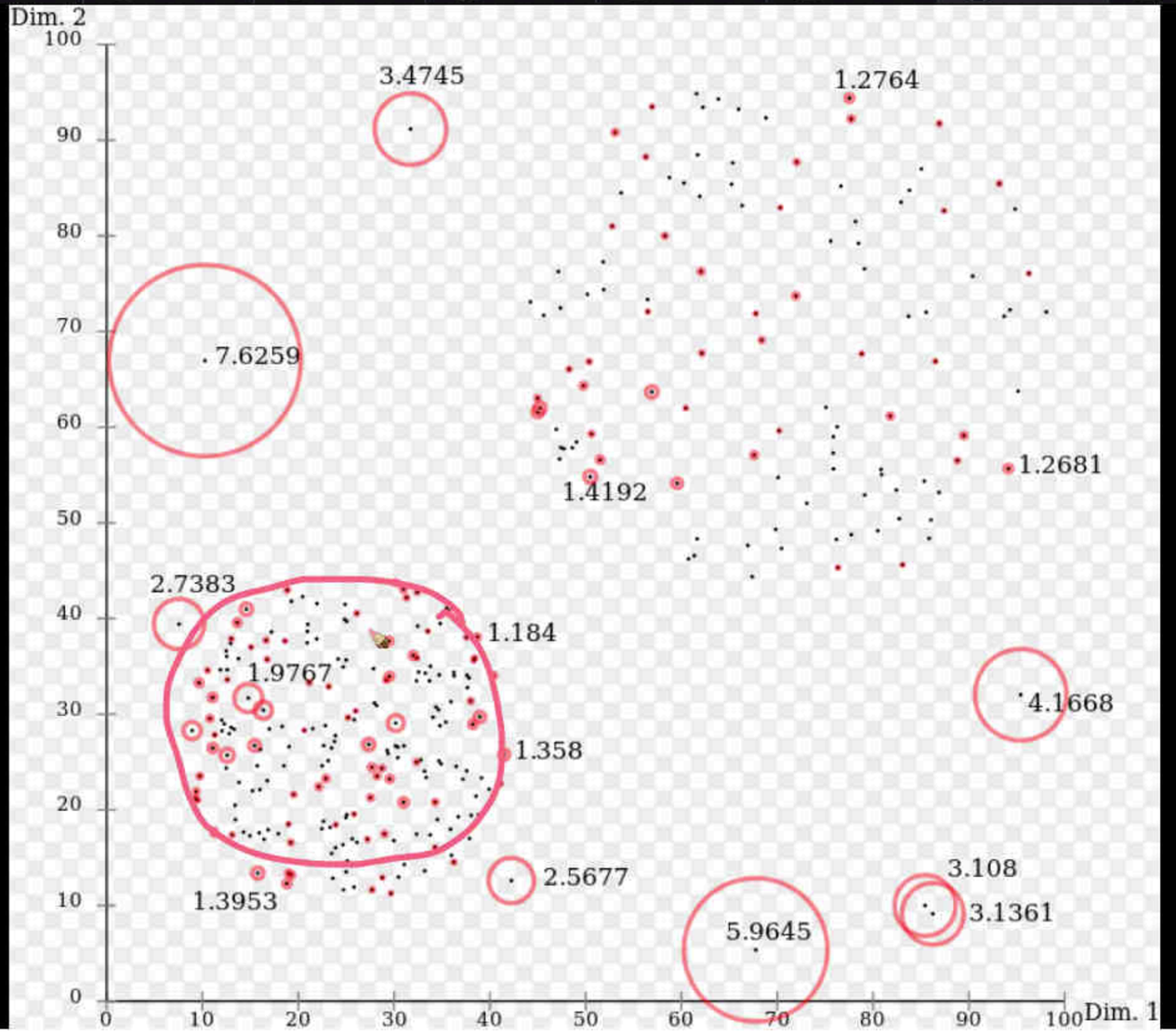
LOF scores as visualized by ELKI. While the upper right cluster has a comparable density to the outliers close to the bottom left cluster, they are detected correctly.

More details

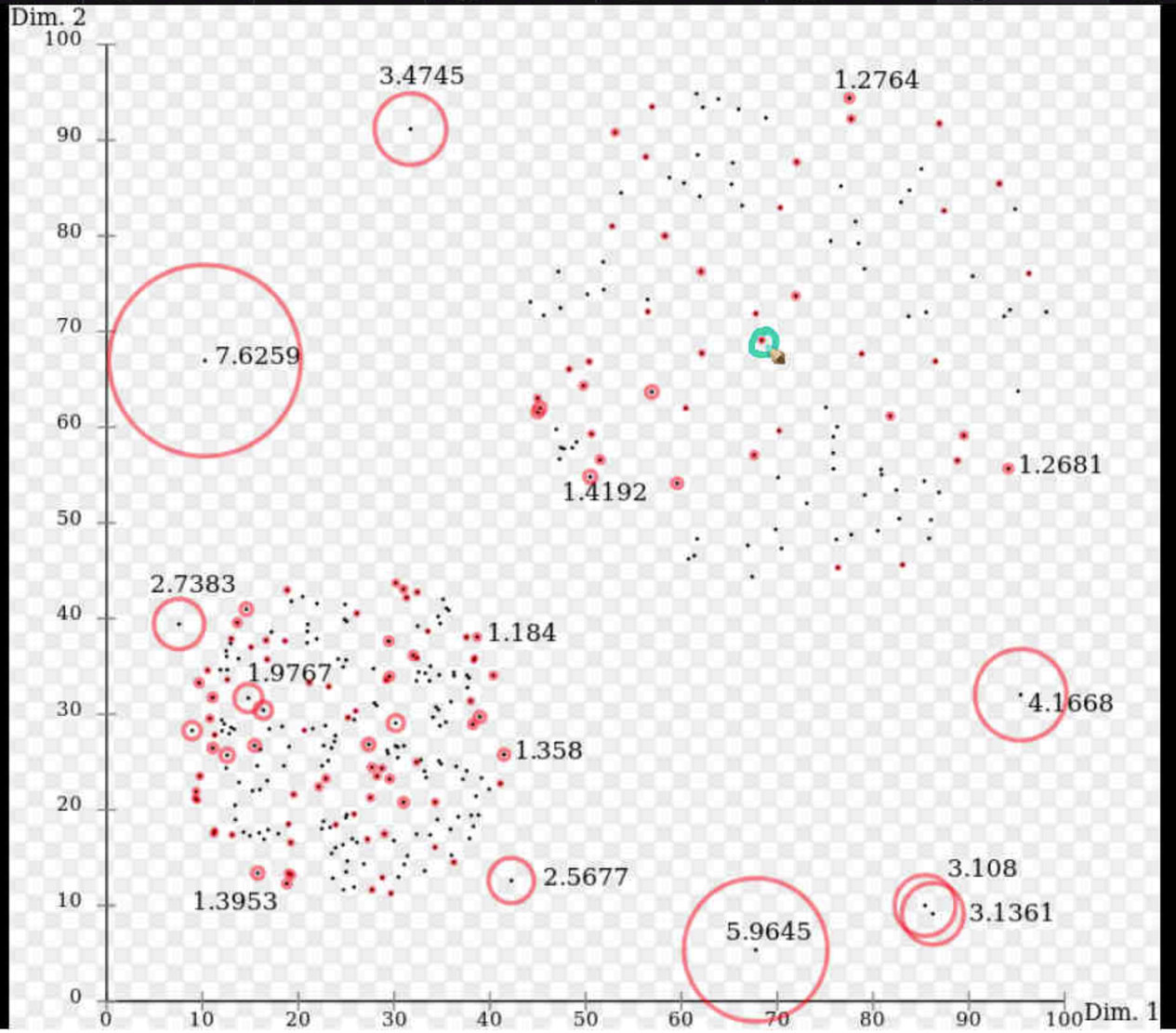


LOF scores as visualized by ELKI. While the upper right cluster has a comparable density to the outliers close to the bottom left cluster, they are detected correctly.

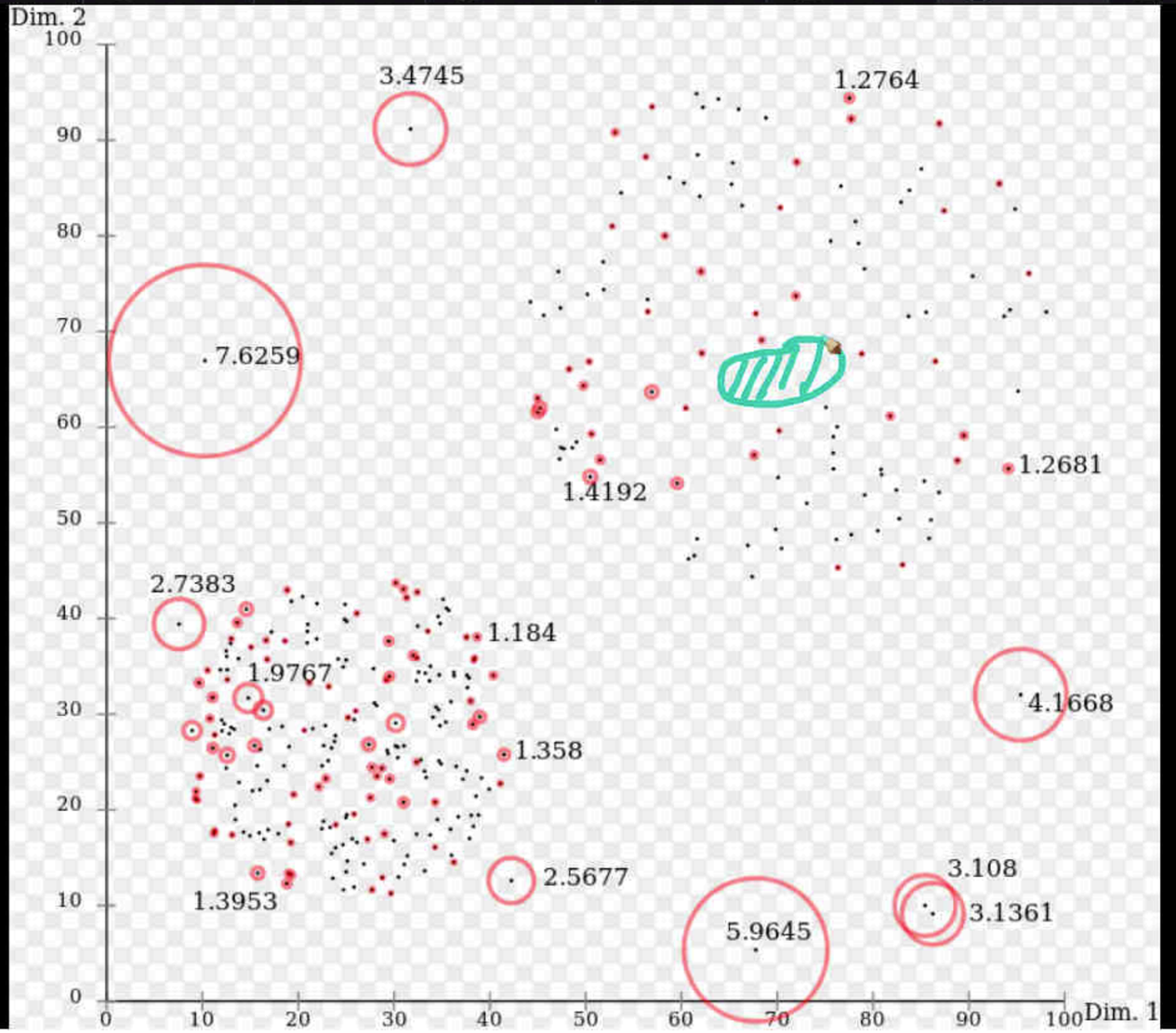
More details



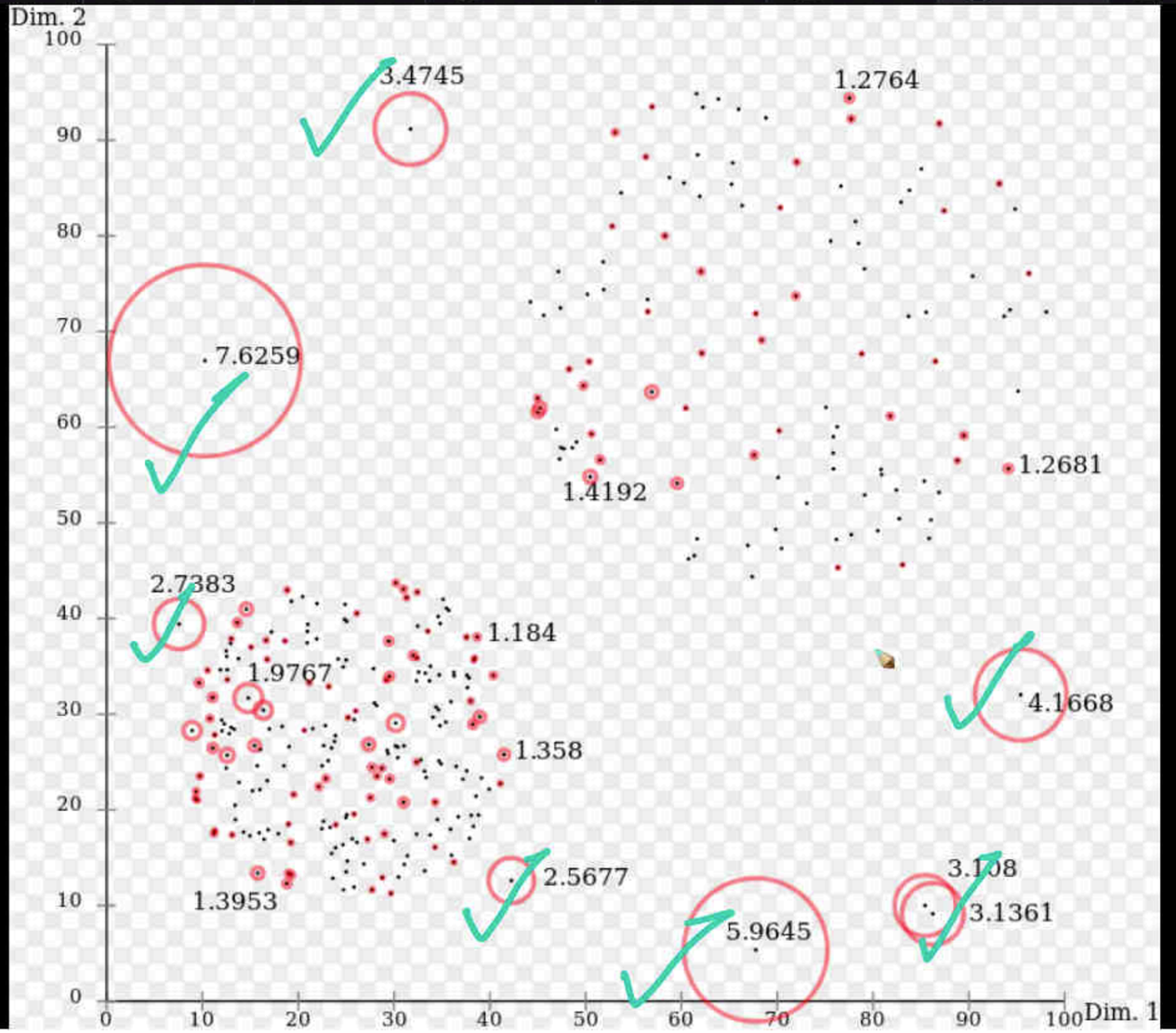
LOF scores as visualized by ELKI. While the upper right cluster has a comparable density to the outliers close to the bottom left cluster, they are detected correctly.



LOF scores as visualized by ELKI. While the upper right cluster has a comparable density to the outliers close to the bottom left cluster, they are detected correctly.

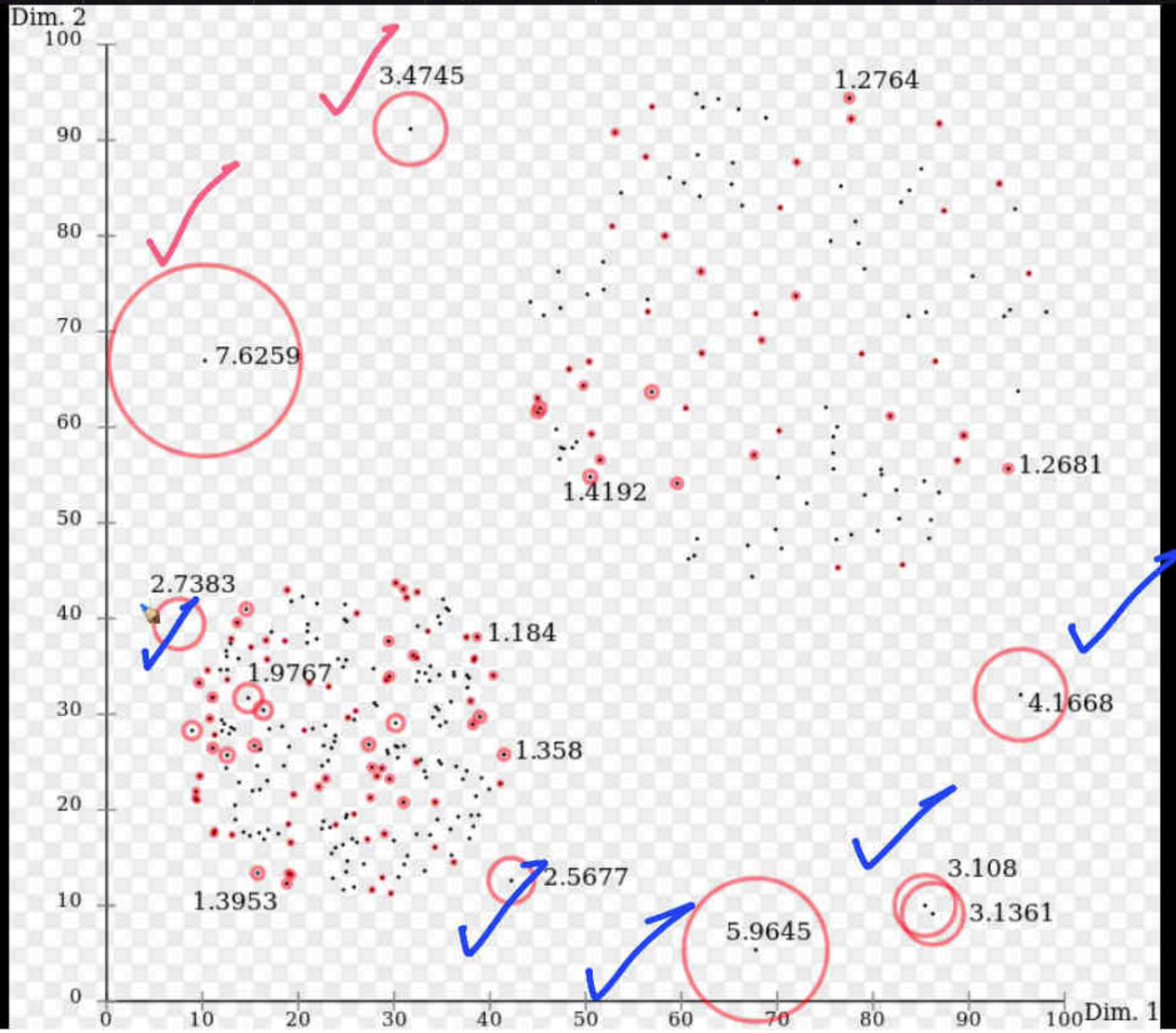


LOF scores as visualized by ELKI. While the upper right cluster has a comparable density to the outliers close to the bottom left cluster, they are detected correctly.



LOF scores as visualized by ELKI. While the upper right cluster has a comparable density to the outliers close to the bottom left cluster, they are detected correctly.

More details



LOF scores as visualized by ELKI. While the upper right cluster has a comparable density to the outliers close to the bottom left cluster, they are detected correctly.

More details

2.7. Novelty and Outlier Detection

Many applications require being able to decide whether a new observation belongs to the same distribution as existing observations (it is an *inlier*), or should be considered as different (it is an *outlier*). Often, this ability is used to clean real data sets. Two important distinctions must be made:

outlier detection:	The training data contains outliers which are defined as observations that are far from the others. Outlier detection estimators thus try to fit the regions where the training data is the most concentrated, ignoring the deviant observations.
novelty detection:	The training data is not polluted by outliers and we are interested in detecting whether a **new** observation is an outlier. In this context an outlier is also called a novelty.

Outlier detection and novelty detection are both used for anomaly detection, where one is interested in detecting abnormal or unusual observations. Outlier detection is then also known as unsupervised anomaly detection and novelty detection as semi-supervised anomaly detection. In the context of outlier detection, the outliers/anomalies cannot form a dense cluster as available estimators assume that the outliers/anomalies are located in low density regions. On the contrary, in the context of novelty detection, novelties/anomalies can form a dense cluster as long as they are in a low density region of the training data, considered as normal in this context.