

Unsupervised Learning

Anomaly Detection

Peerapon Siripongwutikorn

Machine Learning (67/2)

Topics

Basic concepts in anomaly detection

Statistical-based techniques

- ◆ Mahalanobis distance
- ◆ Histogram based outlier scores

ML-based techniques

- ◆ Isolation forest
- ◆ One-class SVM

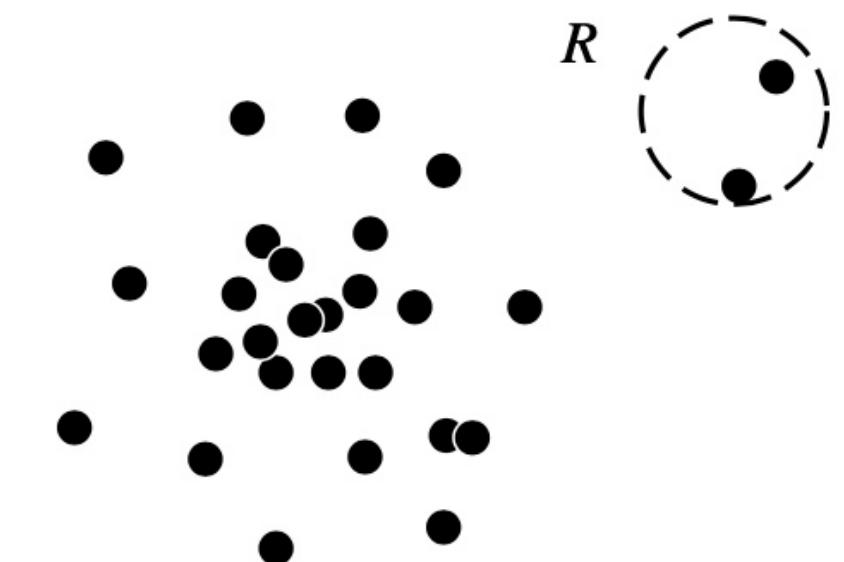
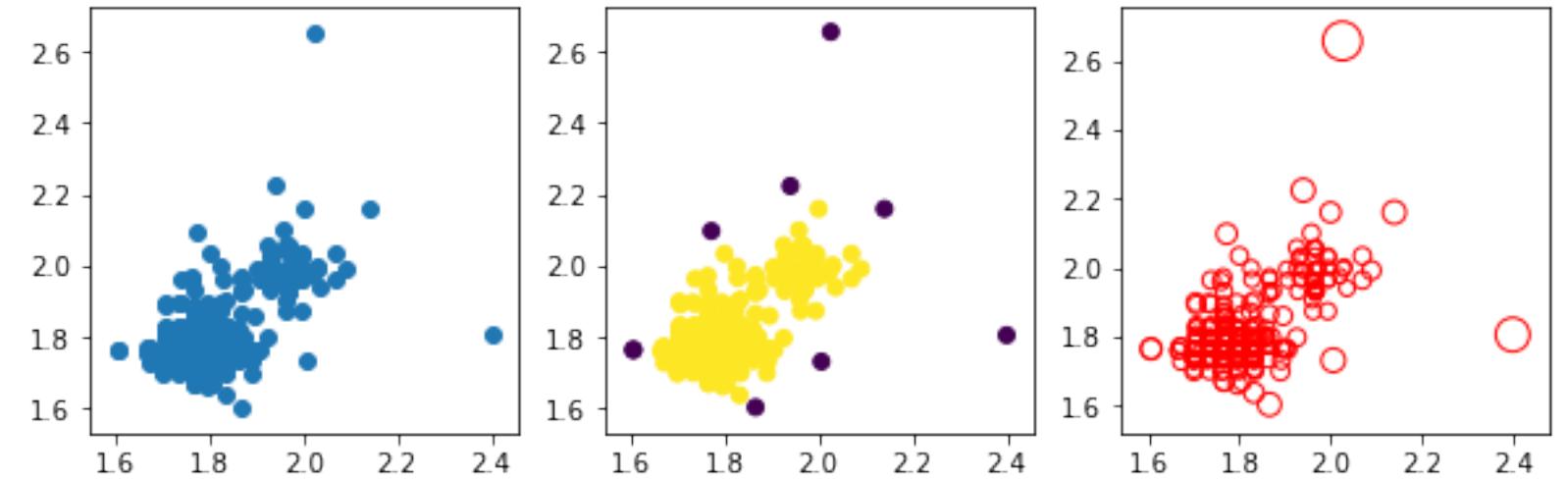
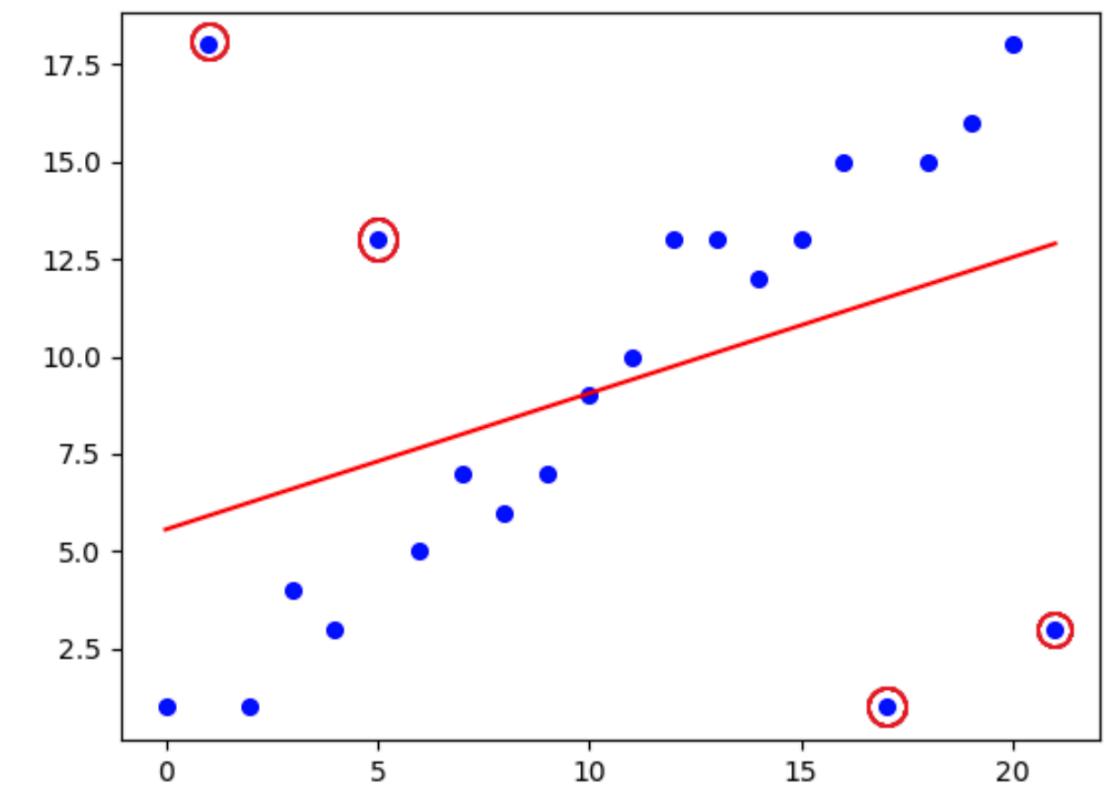
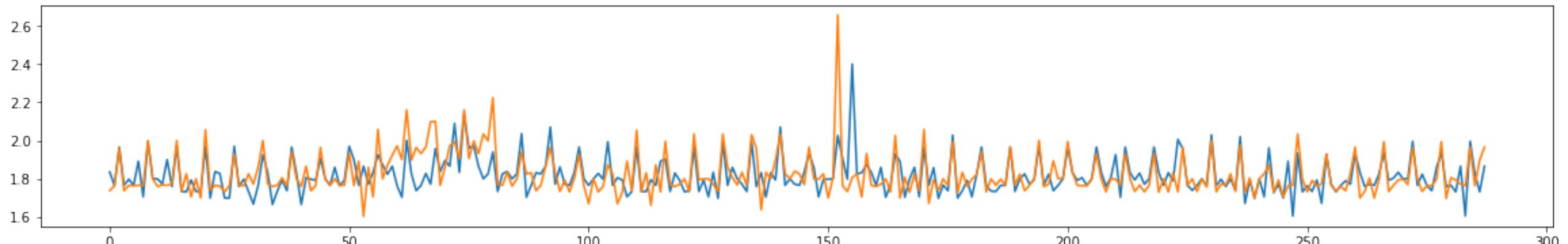
Latent-variable approach

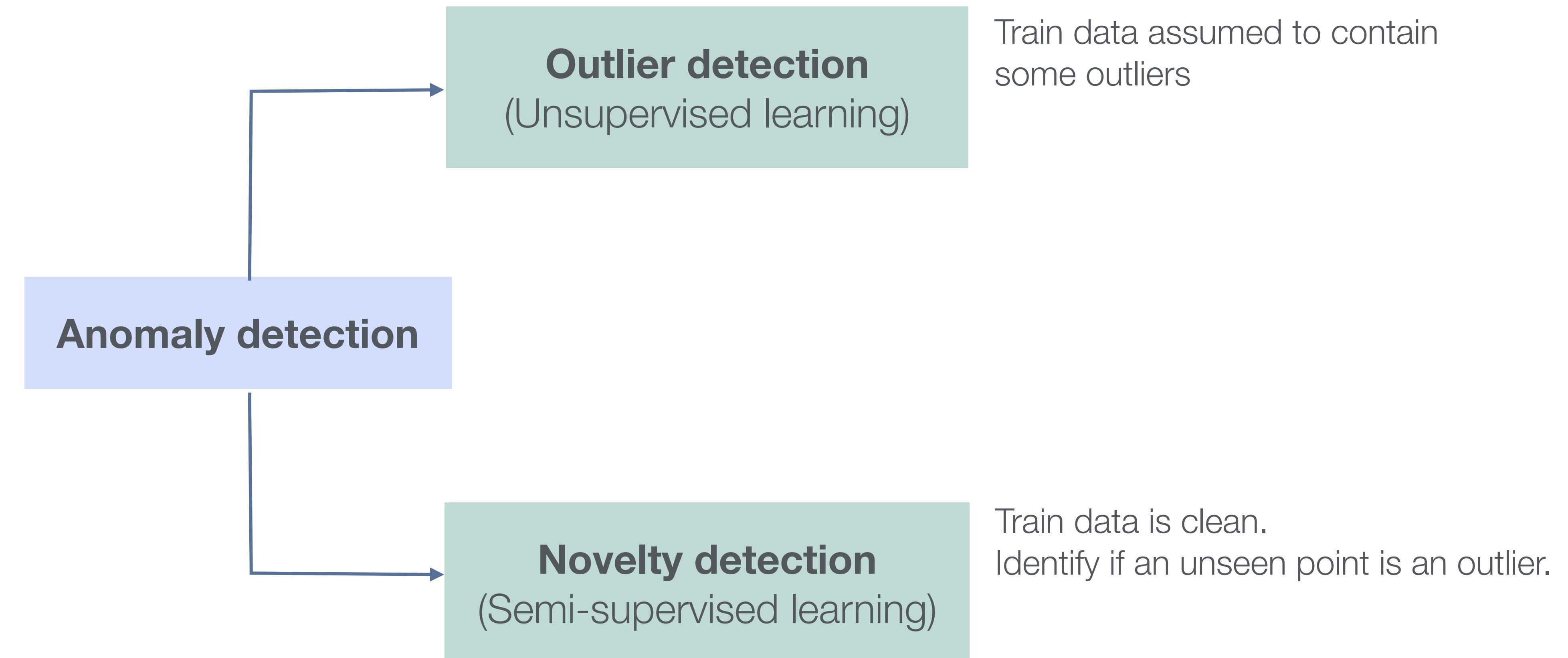
Anomalies

Something that deviates significantly from the majority (*inliers*) or different from expected behavior

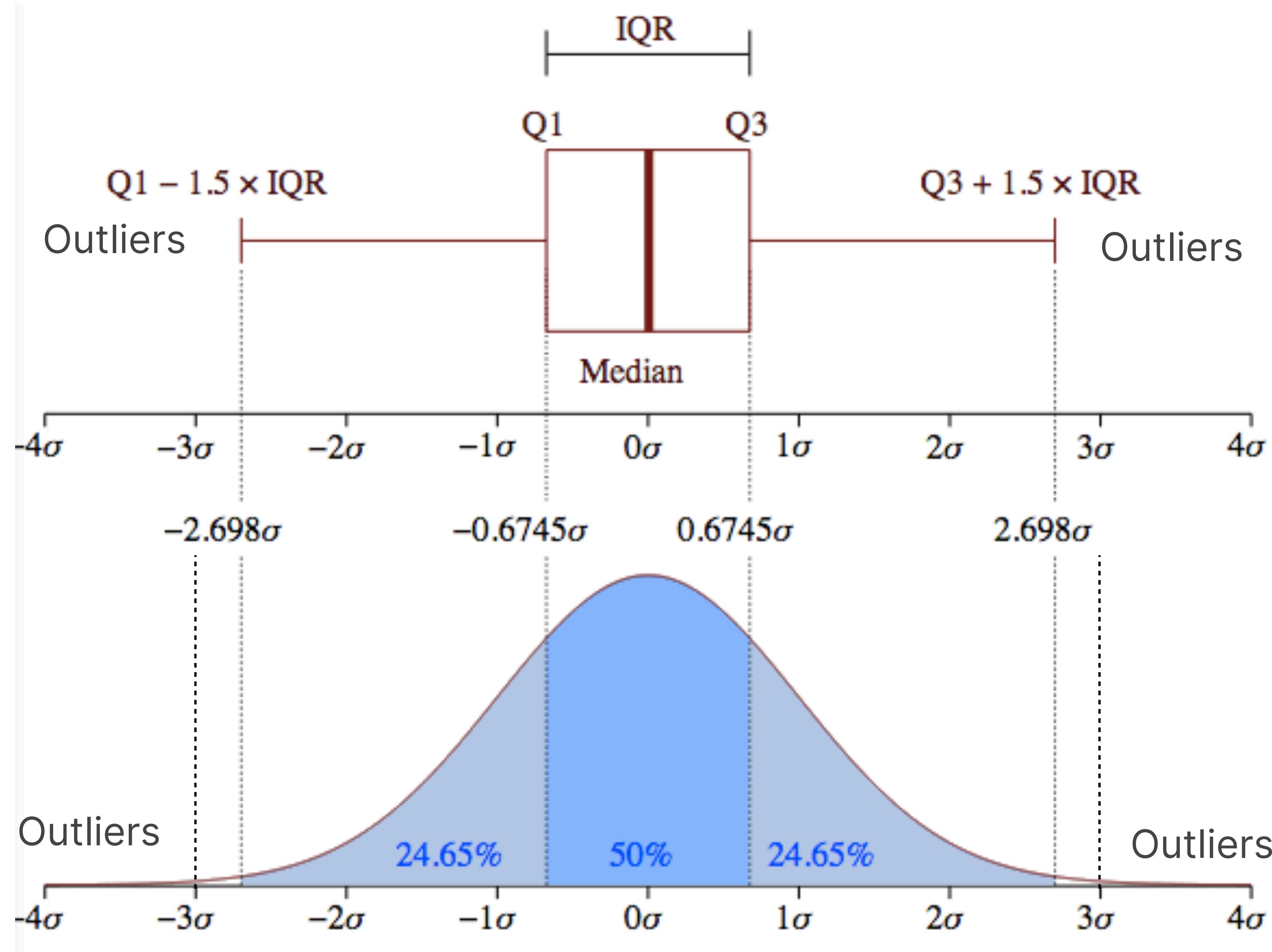
- Ex: Frauds, network intrusions, or system failures.
- Often used interchangeably with *outliers* but some define them differently.
- Not the same as *noises*

Anomalies (or outliers) should be detected and removed if developing supervised models.





Recap: Box Plot, Z-Score, and Outliers

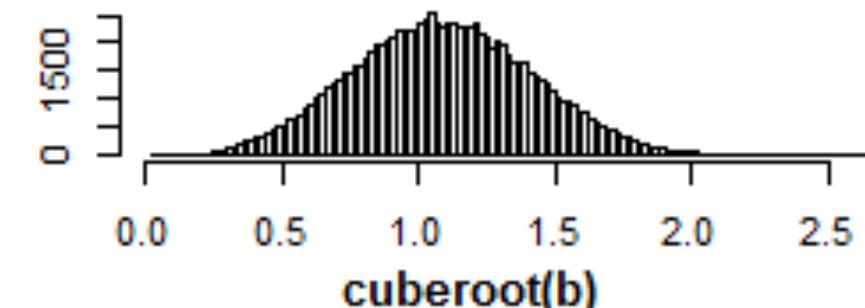
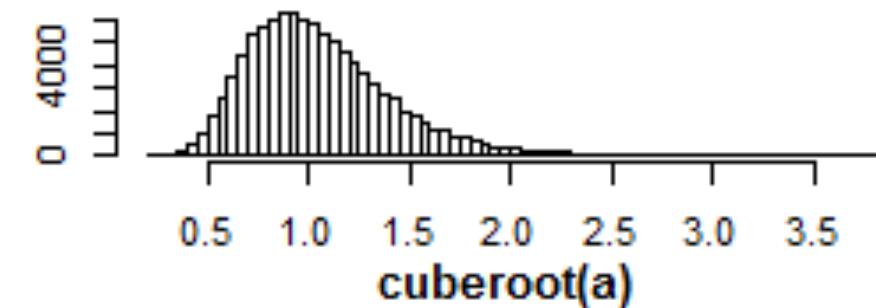
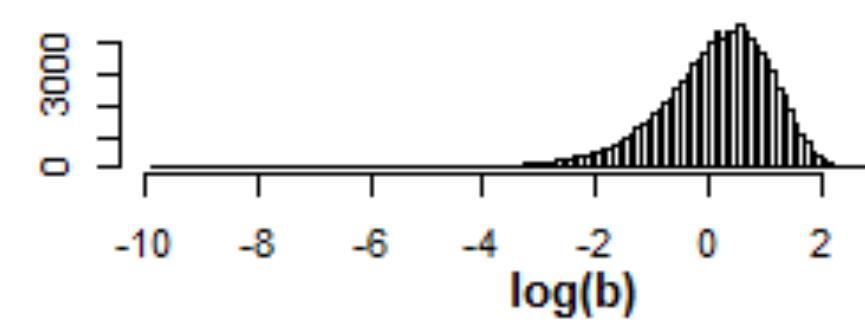
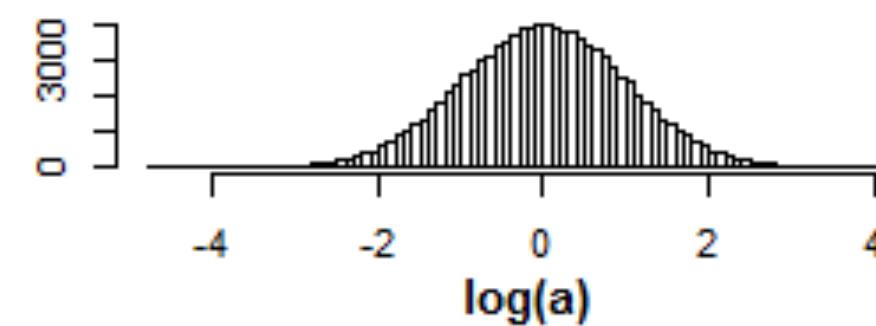
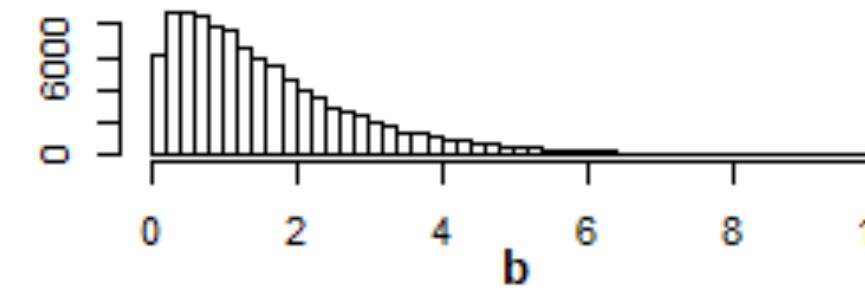
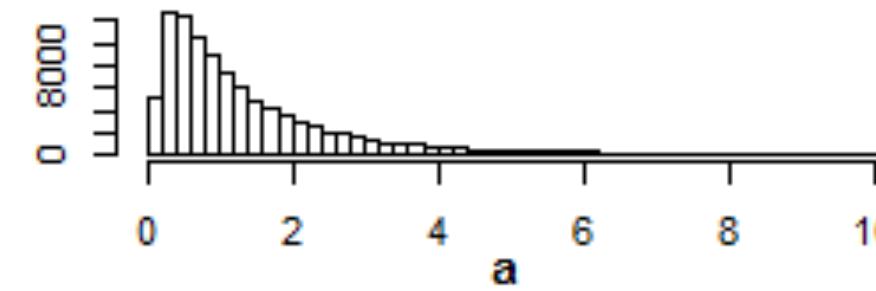


$$z_i = \frac{x_i - \bar{x}}{s}$$

Unusual observations	Outliers
$3 > z > 2$	$ z > 3$

Fixing Skewed distribution

Data should not be too skewed when using box plot or z-scores to identify outliers.



Log transformation, e.g., \log_e , \log_2 , \log_{10}

Power transformation, e.g., \sqrt{x} , $x^{1/3}$, x^2

$$y(\lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(y) & \text{if } \lambda = 0 \end{cases} \quad (-5 \leq \lambda \leq 5)$$

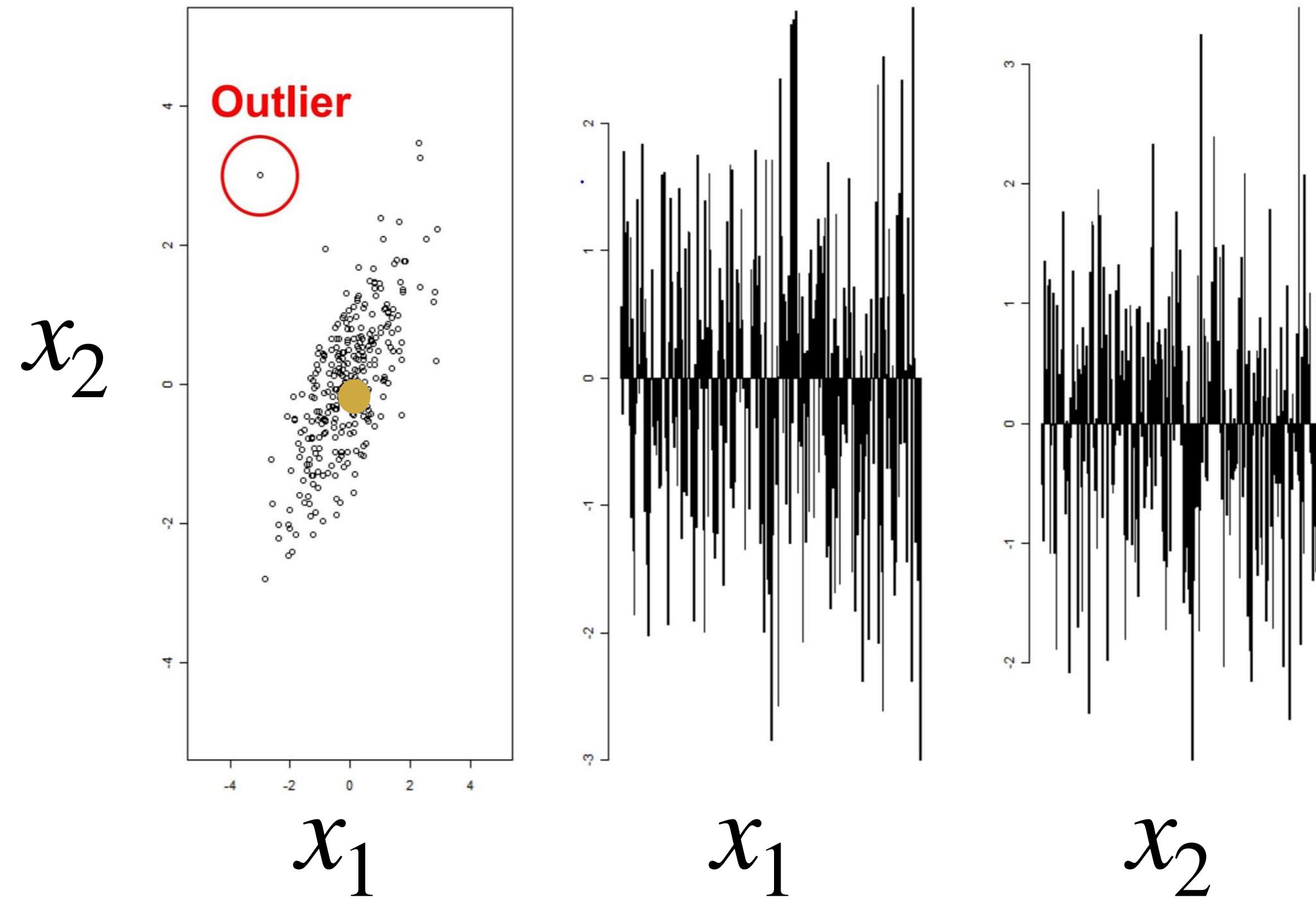
Box-cox transformation

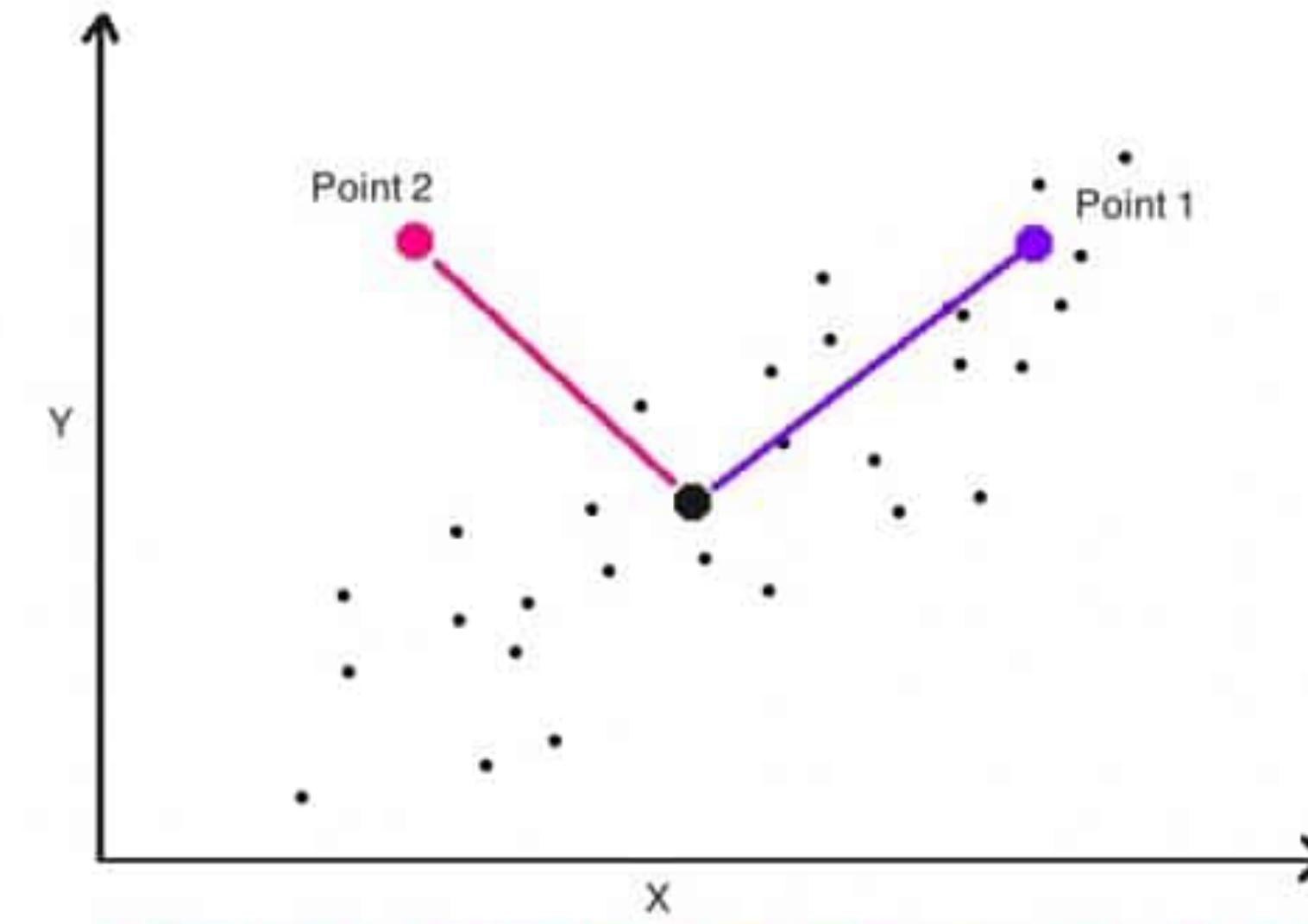
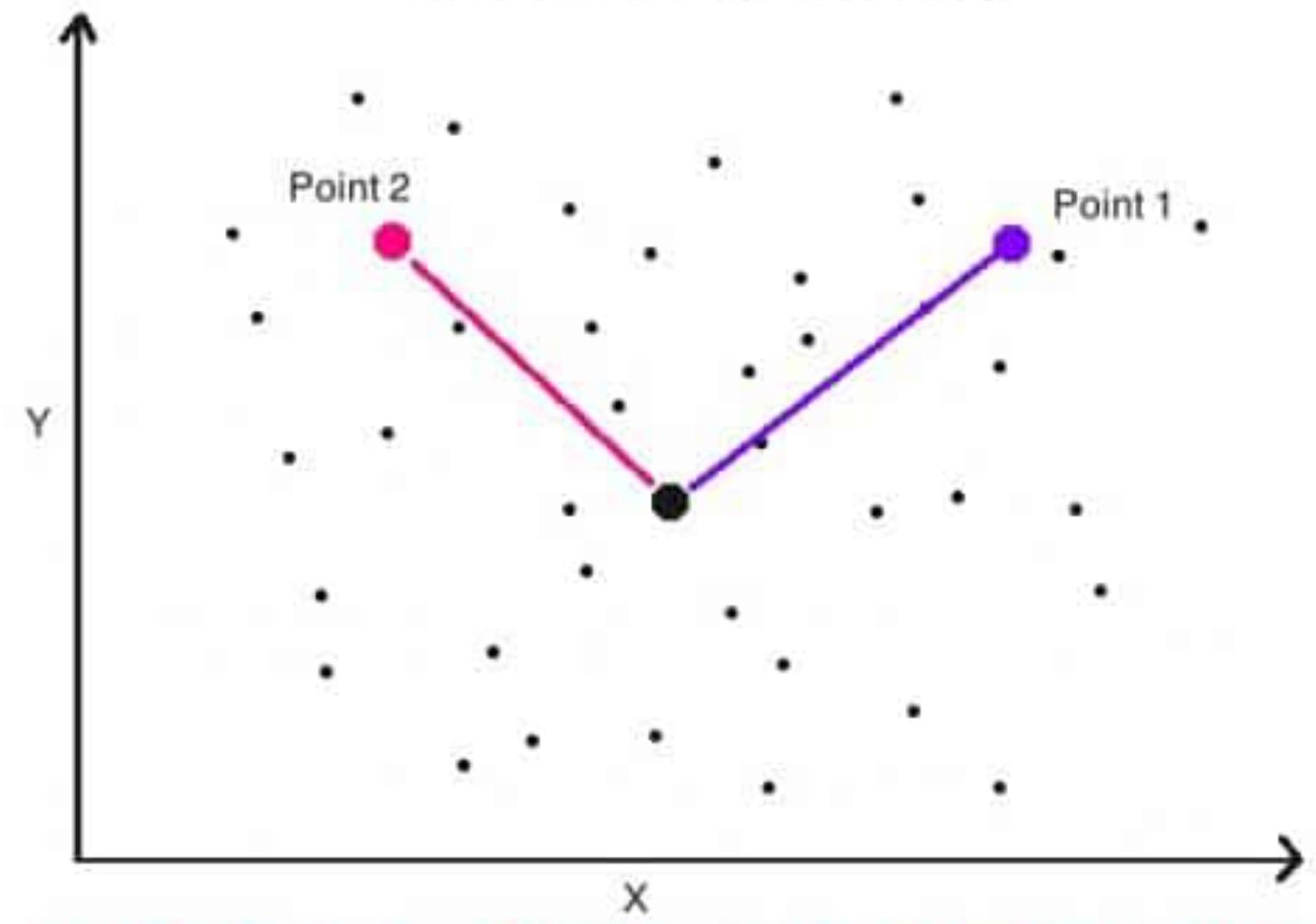
```
from scipy.stats import boxcox  
df['boxcox_x'], _ = boxcox(df.x)
```

Multivariate Outlier

Applying univariate techniques to individual features in multivariate data does not work.

Euclidean distance from center could be used to identify outliers.

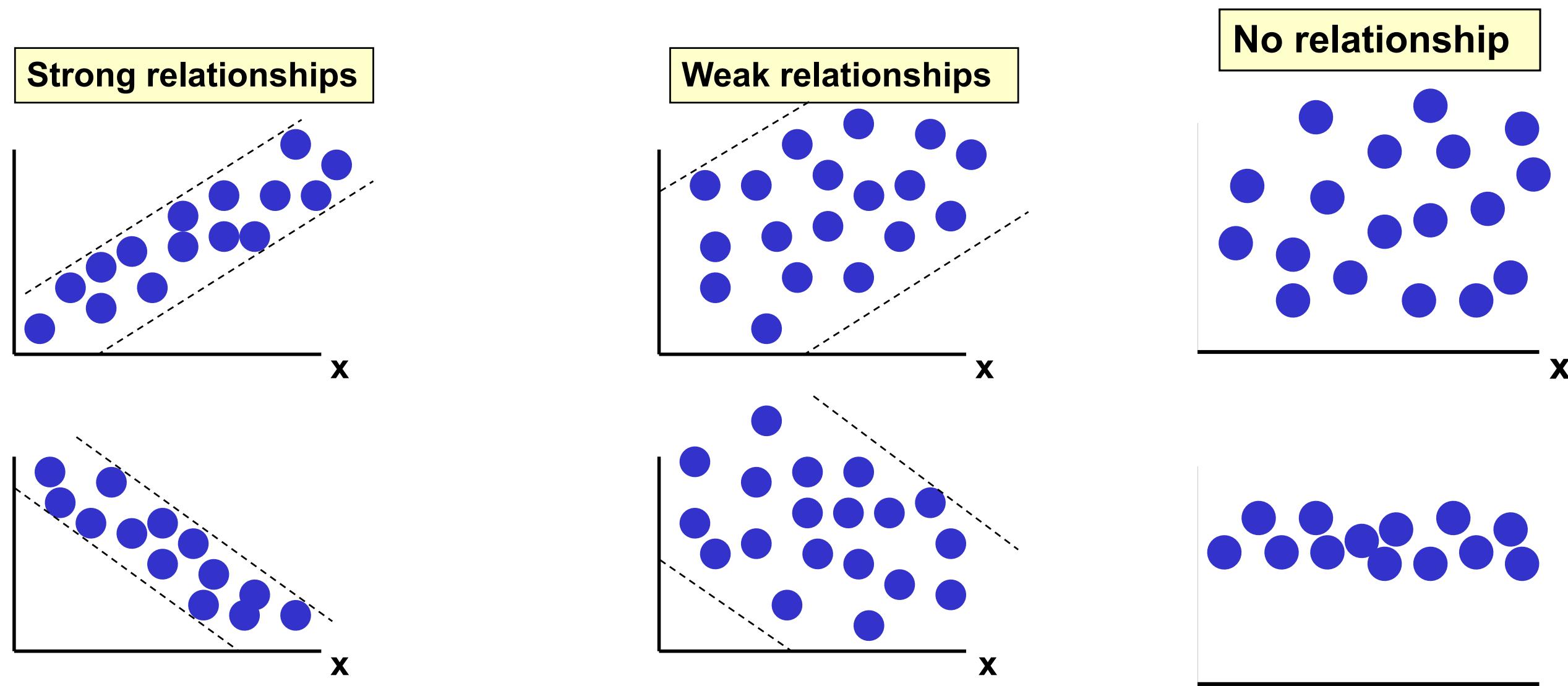
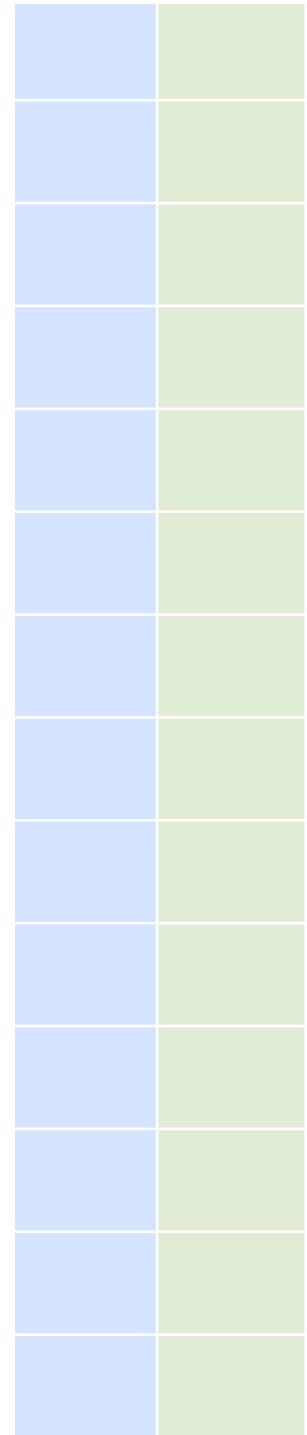




Covariance

Features
 $X_1 \ X_2$

$\mathbf{X} = [\mathbf{X}_1, \ \mathbf{X}_2]$



$$\text{Cov}(X_1, X_2) \triangleq \sigma_{12} = \frac{\sum_i (x_{1i} - \bar{x}_1)(x_{2i} - \bar{x}_2)}{n}$$

Covariance Matrix

Represent all pair-wise linear correlation relationships in dataset.

	X_1	X_2	X_3
X_1			
X_2			
X_3			

$$\Sigma = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) & \text{Cov}(X_1, X_3) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) & \text{Cov}(X_2, X_3) \\ \text{Cov}(X_3, X_1) & \text{Cov}(X_3, X_2) & \text{Cov}(X_3, X_3) \end{bmatrix}$$

For two variables,

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$$

For three variables,

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_2^2 & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_3^2 \end{bmatrix}$$

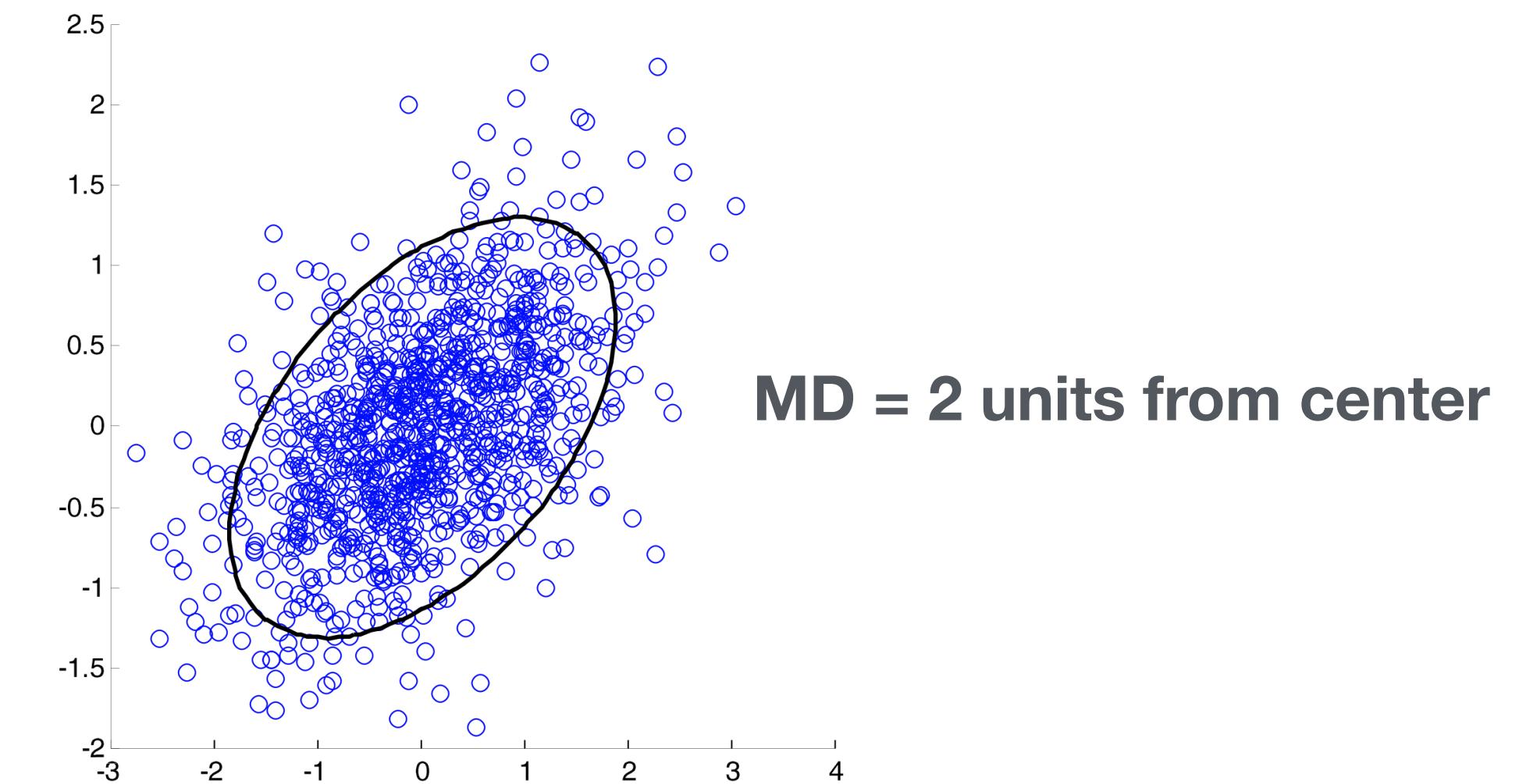
Mahalanobis Distance (MHD)

Assume multivariate normal data

Measure how many σ 's away \mathbf{x}_i is from $\boldsymbol{\mu}$ (centroid) considering the feature correlations:

$$\text{MHD}(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

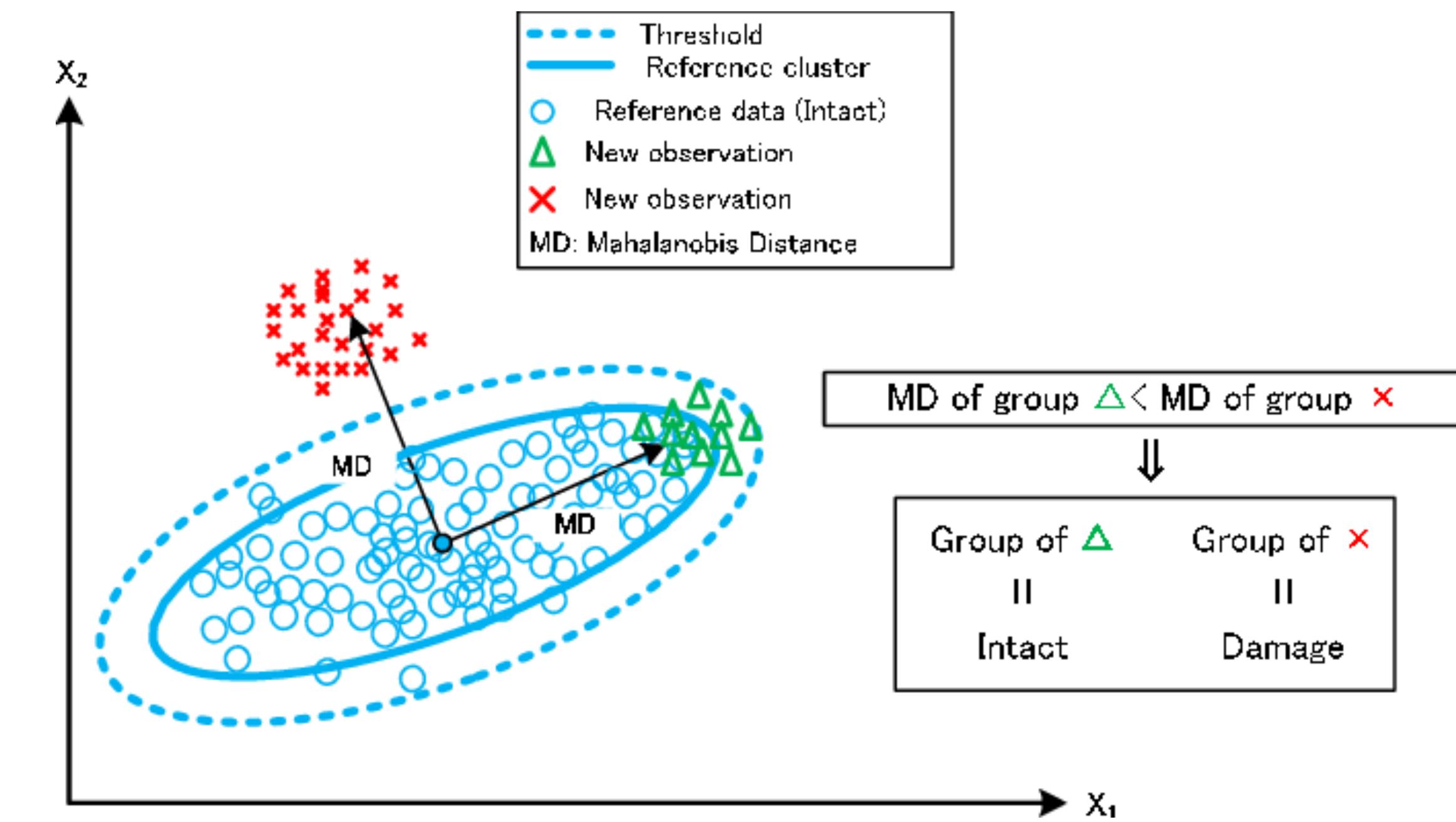
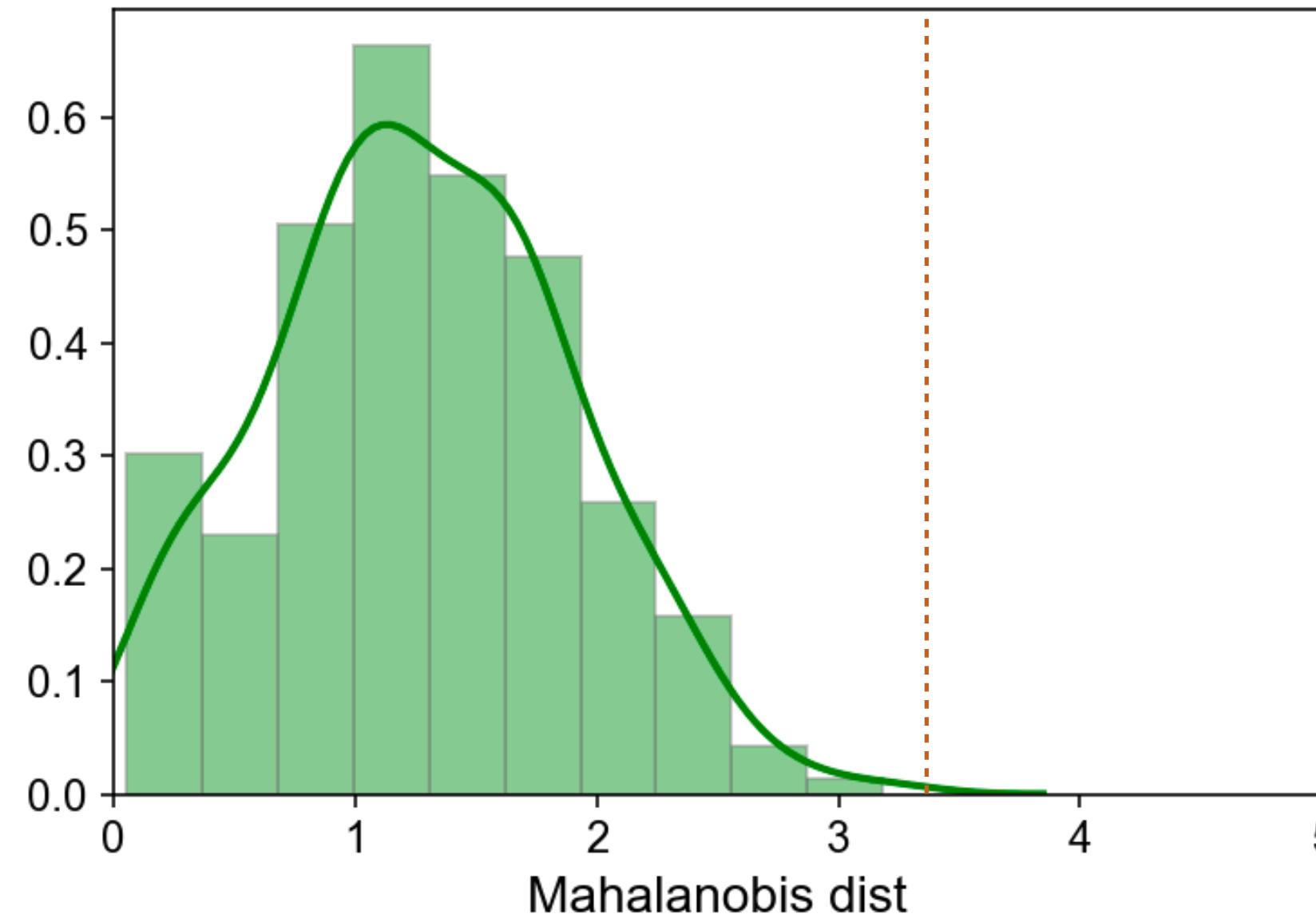
$$\boldsymbol{\Sigma} = \begin{bmatrix} \text{Cov}(X_1, X_1) & \text{Cov}(X_1, X_2) \\ \text{Cov}(X_2, X_1) & \text{Cov}(X_2, X_2) \end{bmatrix}$$



For uncorrelated data, MHD equals the euclidean distance of standarized features.

MHD Properties

For multivariate normal data of k variables, the squared MHD $\sim \chi^2(k)$.

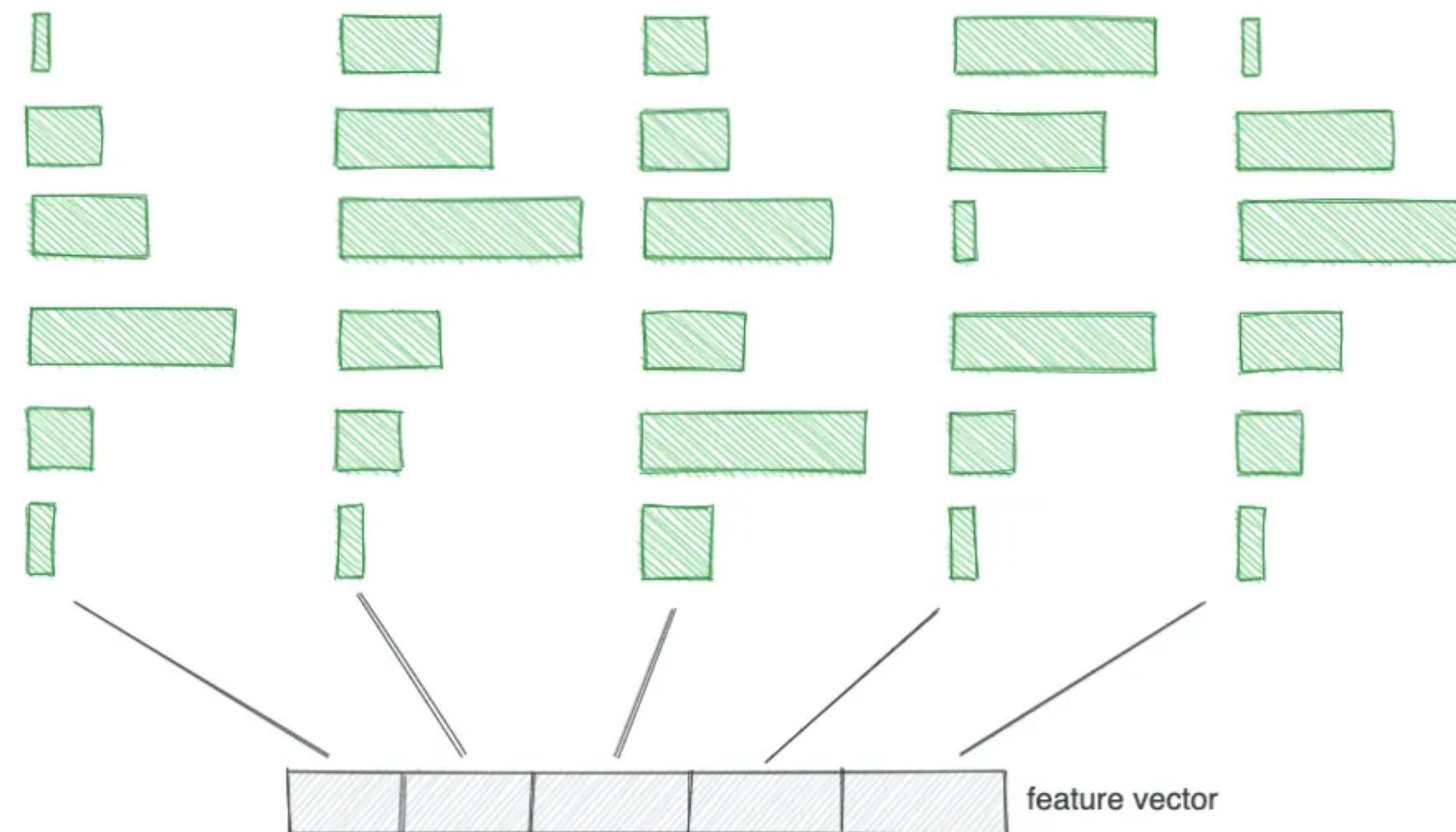


Histogram-based Outlier Scores (HBOS)

Create a normalized histogram (relative frequency counts) of each feature.

Calculate a score based on how likely a particular data point is to fall within the histogram bins.

An anomalous feature vector will occupy unlikely bins in one or several of its dimensions.



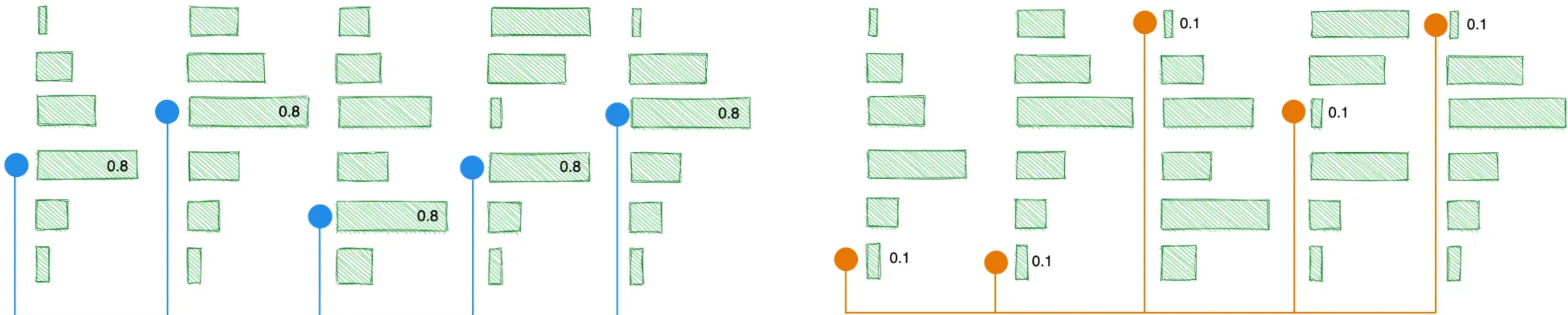
For a given data point of d features $\mathbf{x} = [x_1, x_2, x_3, \dots, x_d]$, multiply the inverse of bin heights of each feature to obtain the HBOS as

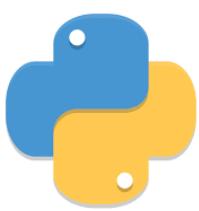
$$\text{HBOS}(\mathbf{x}) = \log \prod_{i=1}^d \frac{1}{\text{hist}(x_i)} = \sum_{i=1}^d \log \frac{1}{\text{hist}(x_i)}$$

Positive constant α can be added to $\text{hist}(x_i)$ to prevent overflow.

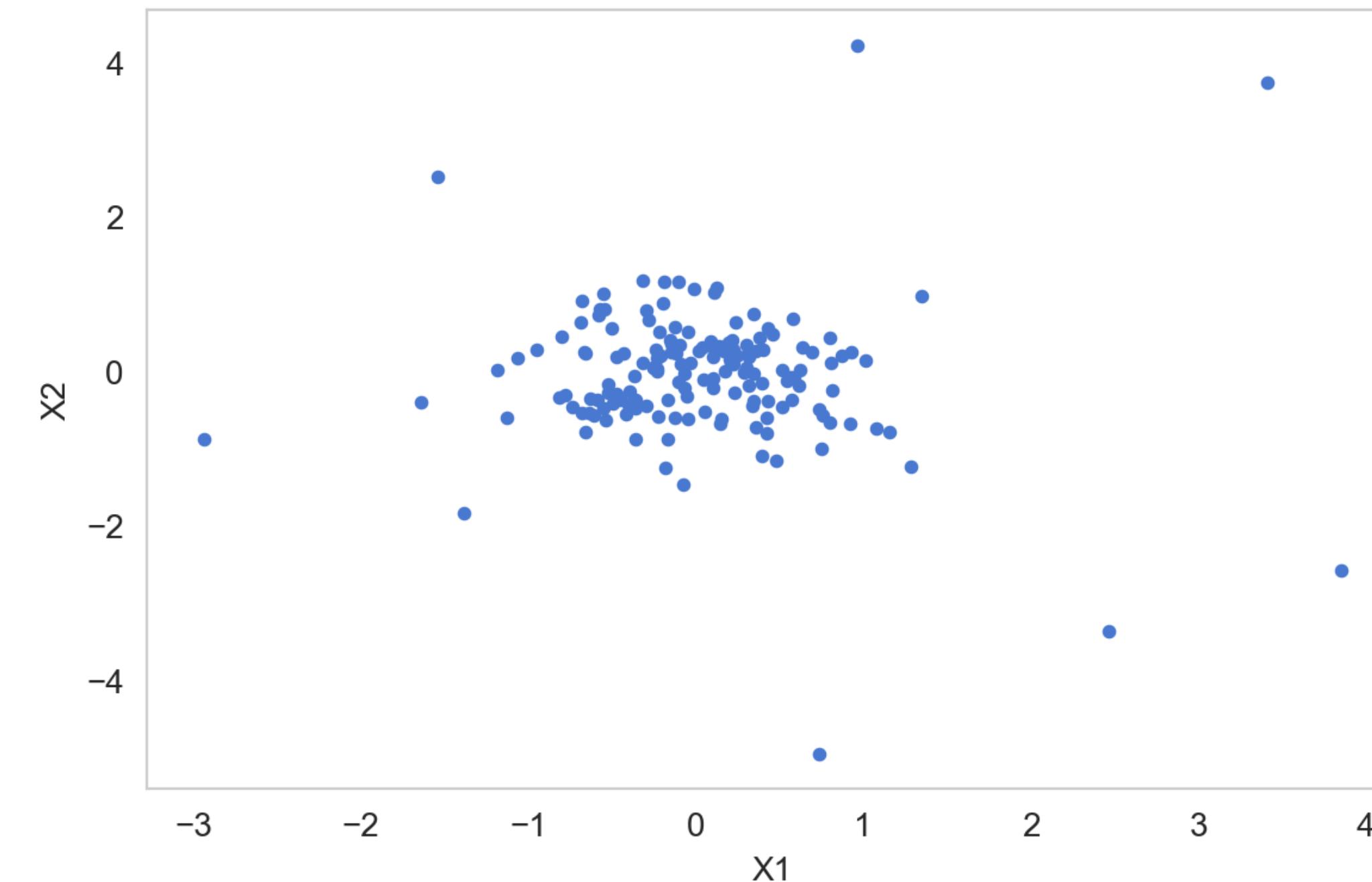
$$\text{HBOS}(\mathbf{x}_{normal}) = \log \prod_{i=1}^5 \frac{1}{0.8} = 0.484$$

$$\text{HBOS}(\mathbf{x}_{anom}) = \log \prod_{i=1}^5 \frac{1}{0.1} = 5$$





```
# Run this command if you have not done so.  
!python -m pip install -U pip  
!python -m pip install -U pyod  
  
# Generate synthetic data  
normal_data = np.random.normal(0, 0.5, (150, 2))  
anomalous_data = np.random.uniform(-5, 5, (10, 2))  
  
input_df = pd.DataFrame(np.vstack([normal_data, anomalous_data]), columns=['X1', 'X2'])  
input_df.plot.scatter(x='X1', y='X2', figsize=(6,4), s=10);  
plt.tight_layout();
```





```
# Model fitting
from pyod.models.hbos import HBOS

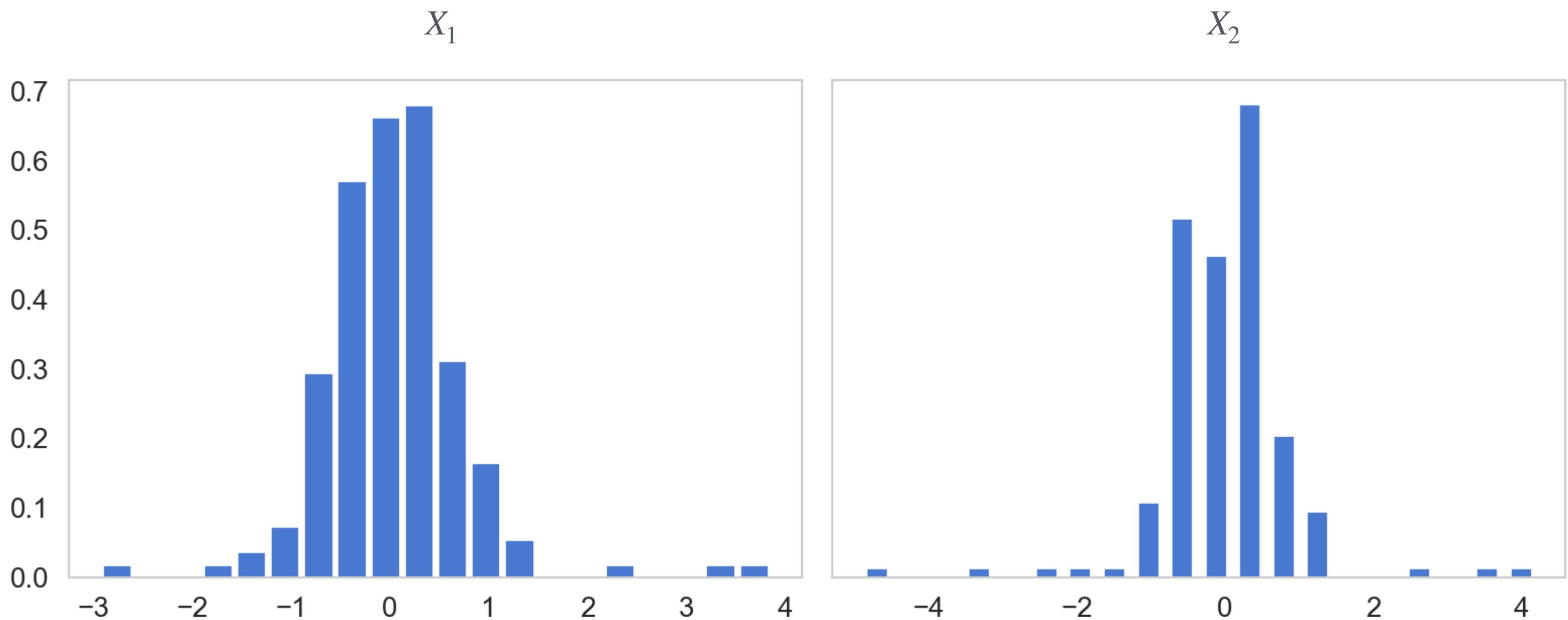
n_bins = 20
hbos = HBOS(n_bins=n_bins, contamination=0.06)
hbos.fit(input_df)

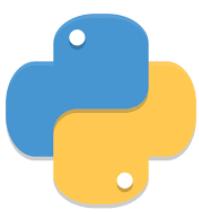
# Identify outliers
y_scores = hbos.decision_function(input_df)
y_pred = hbos.predict(input_df)
print(f'Outlier threshold: {hbos.threshold_:.3f}')
```

Outlier threshold: 4.062

```
pd.DataFrame(np.stack((y_scores, y_pred), axis=1), columns=['Score', 'Prediction'])
```

	Score	Prediction
0	1.049848	0.0
1	1.969470	0.0
2	2.833767	0.0
3	3.533437	0.0
4	6.007077	1.0
...
155	5.049545	1.0
156	6.007077	1.0
157	3.528013	0.0
158	3.771829	0.0
159	6.215586	1.0

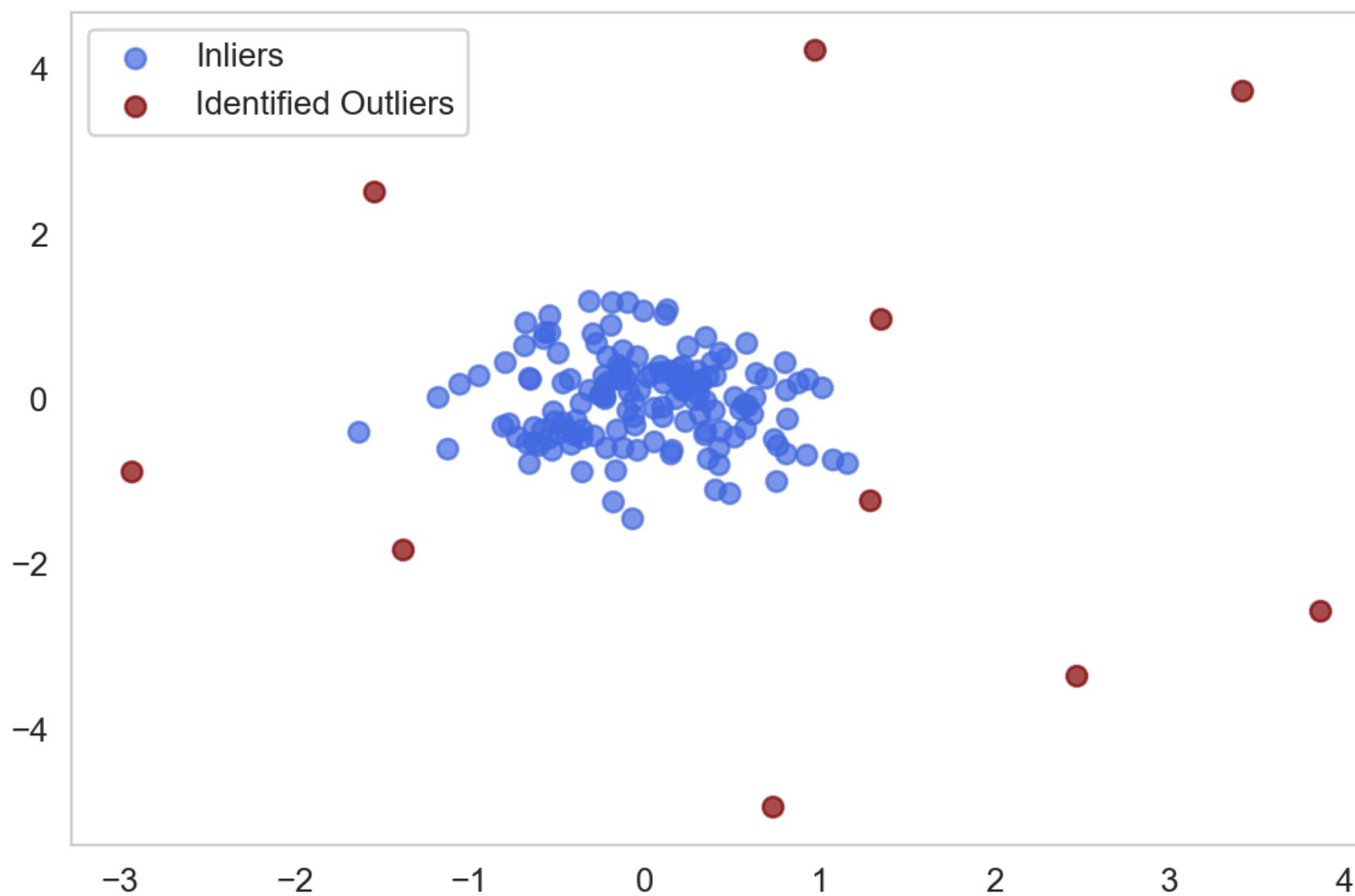




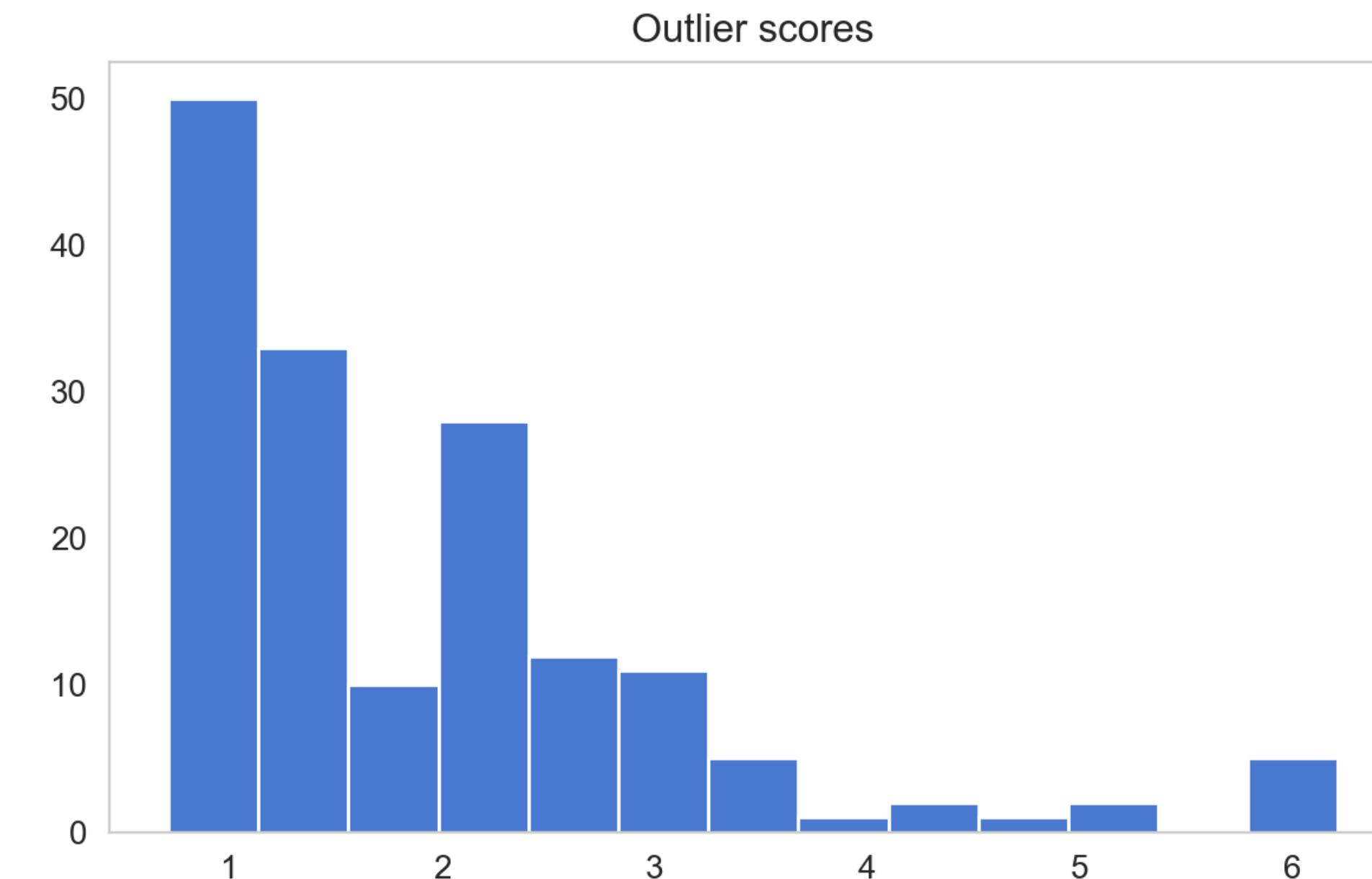
```
# Extract inliers and outliers to visualize
outliers = input_df[y_pred==1]
inliers = input_df[y_pred==0]

plt.figure(figsize=(6, 4));
plt.scatter(inliers.X1, inliers.X2, label='Inliers', color='royalblue', alpha=0.7);
plt.scatter(outliers.X1, outliers.X2, label='Identified Outliers', color='maroon', alpha=0.7);

plt.legend();
plt.tight_layout();
```



Proper decision threshold can be determined from the histogram of outlier scores.

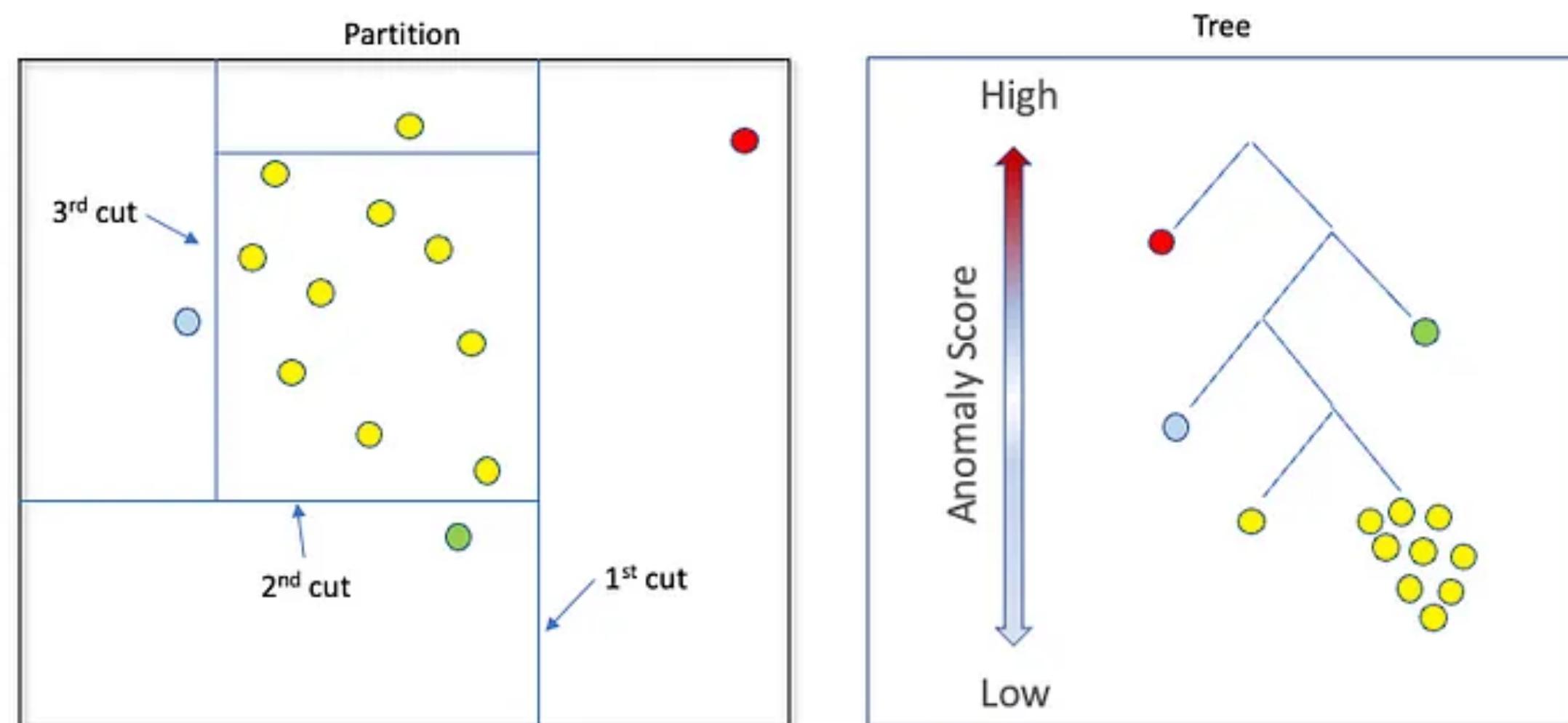


Isolation Forest

Recursively split the data points into halves such that each observation gets isolated from the others to obtain an *isolation tree*.

No target variable to guide the splittings.

Because anomalies, if any, lie away from the cluster of data points, they would be easily isolated compared to regular data points.



Generating Isolation Tree (iTTree)

For a random subset of data of size ψ , recursively select a random feature and split the data at a random threshold of the feature.

Stop when

- ◆ the node has fewer than a predefined number of points (1 or 2), or
- ◆ the max depth (if defined) is reached.
- ◆ all data points at the node have the same value for every feature (rarely happens).

0	3	1	0	2	3	3	2	1	1
1	1	0	0	7	1	2	2	3	3
1	2	2	0	0	3	3	2	2	2
9	8	9	10	7	7	8	9	2	3
3	2	2	1	3	2	1	5	6	0
17	41	14	25	33	29	46	21	25	10
4	1	1	5	0	0	7	1	2	2
4	0	1	6	2	0	0	3	8	5
1	6	3	3	9	10	7	7	2	2
1	2	2	4	3	3	2	2	2	0

Path length (of the leaf node)

- ◆ # splits before isolation
- ◆ # edges from the root to the leaf node.

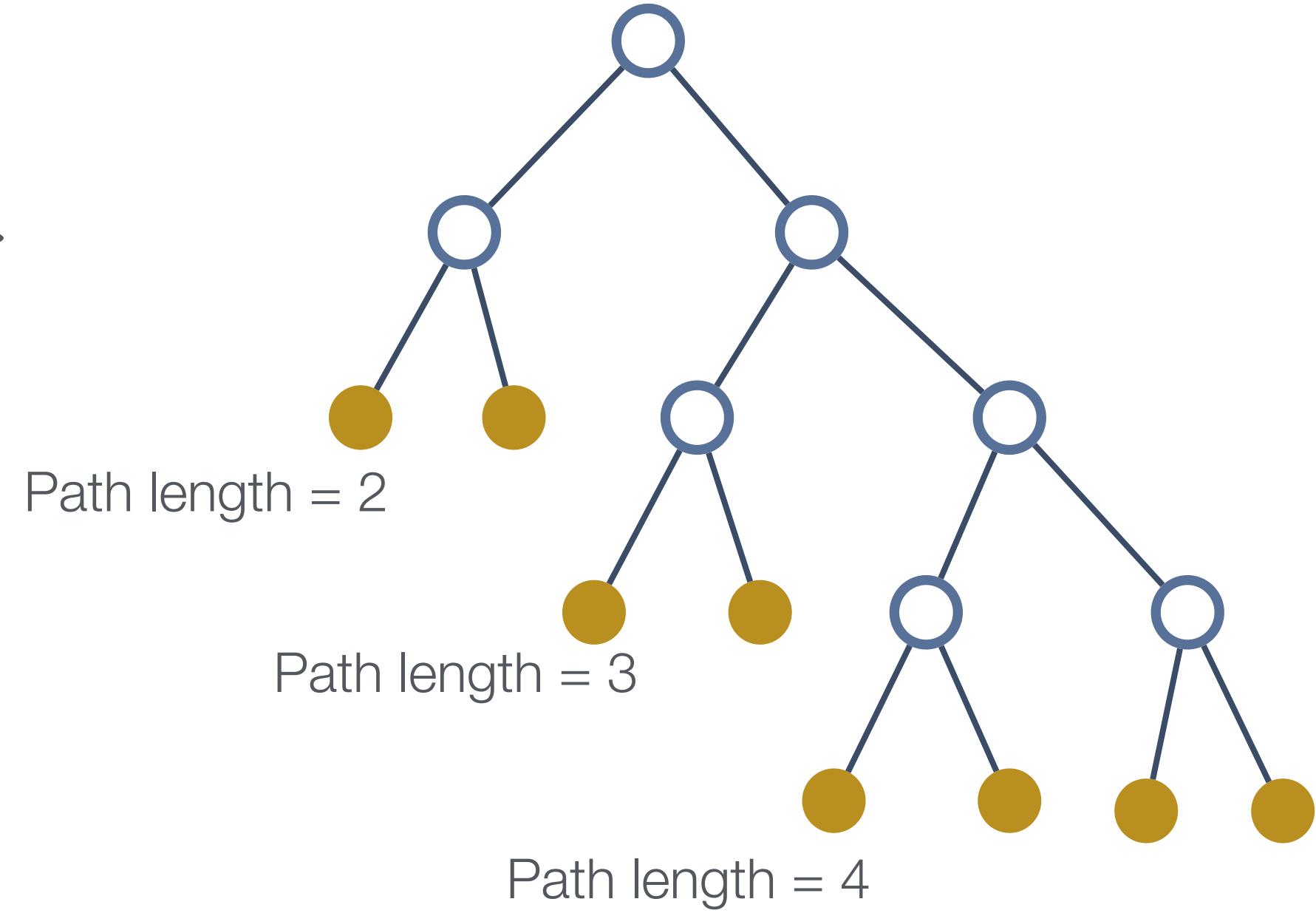
Random partitioning tend to produce noticeably shorter paths for anomalies.

For a binary tree with ψ leave nodes, the average # comparisons for unsuccessful search is

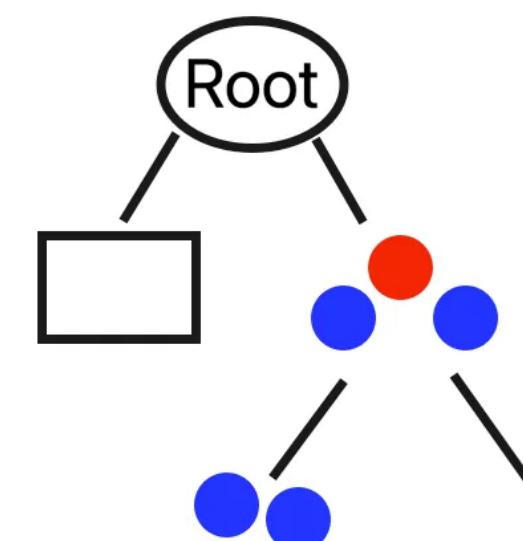
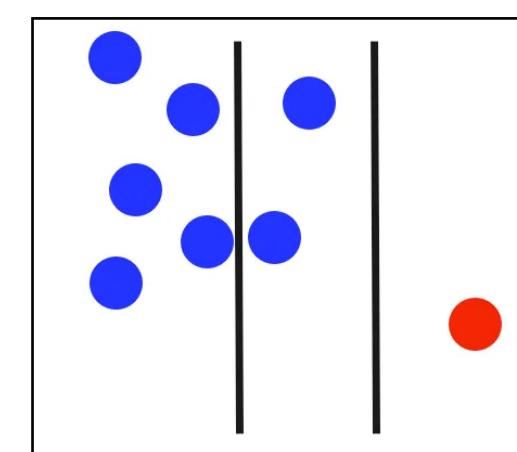
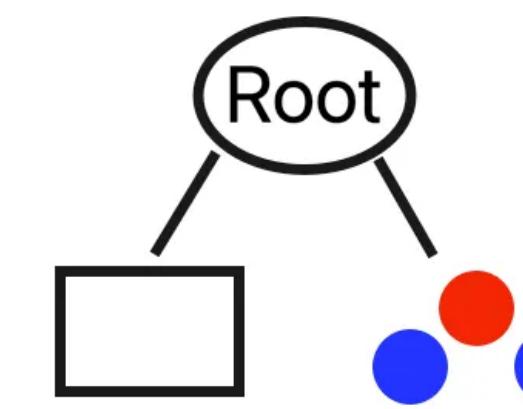
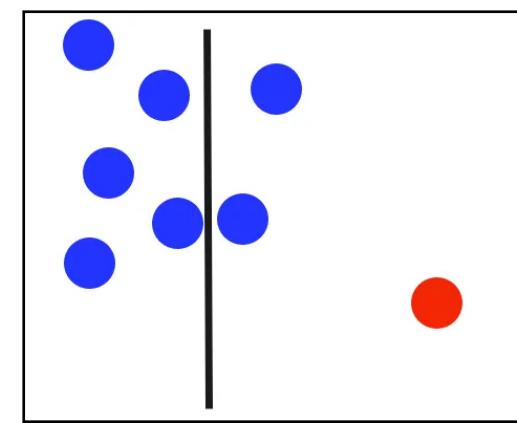
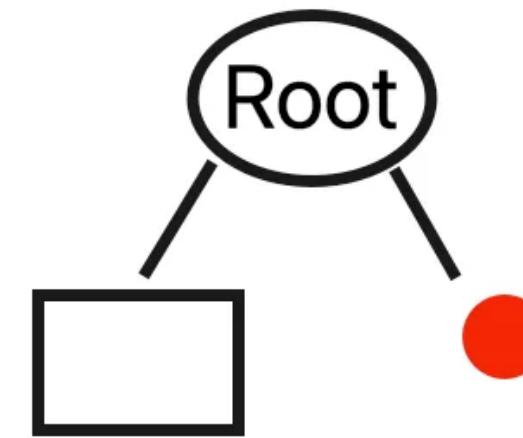
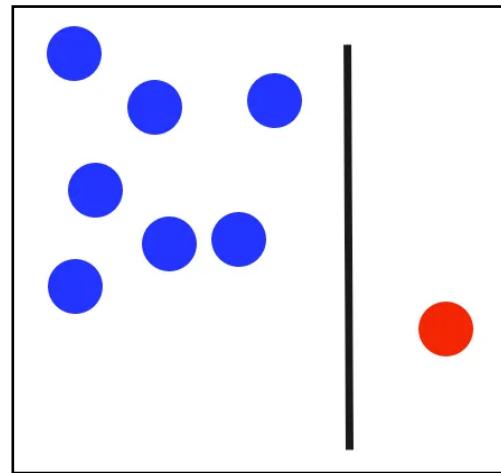
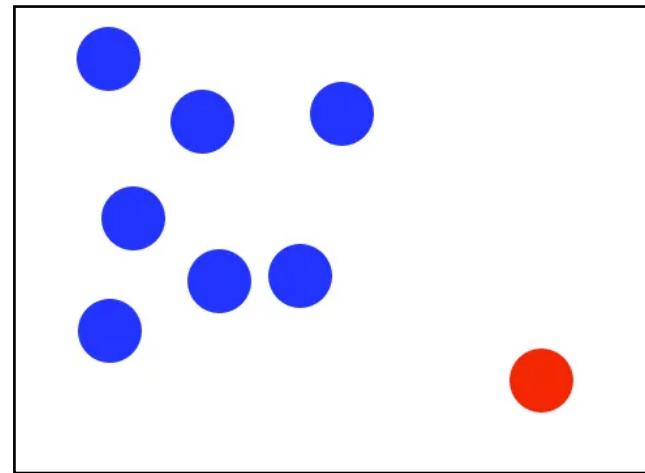
$$c(\psi) = \begin{cases} 2H(\psi - 1) - 2(\psi - 1)/n & \text{for } \psi > 2, \\ 1 & \text{for } \psi = 2, \\ 0 & \text{otherwise.} \end{cases}$$

$$H(i) = \log(i) + 0.5772156649 \quad (\text{Harmonic number})$$

Used as the expected path length if most data are inliers.



Generate lots of iTrees from random subsamples of the dataset, and determine the average path length of x from those trees.



of splits before isolation = 2

of splits before isolation = 3

...

of splits before isolation = 4

Average = 3

Anomaly Score

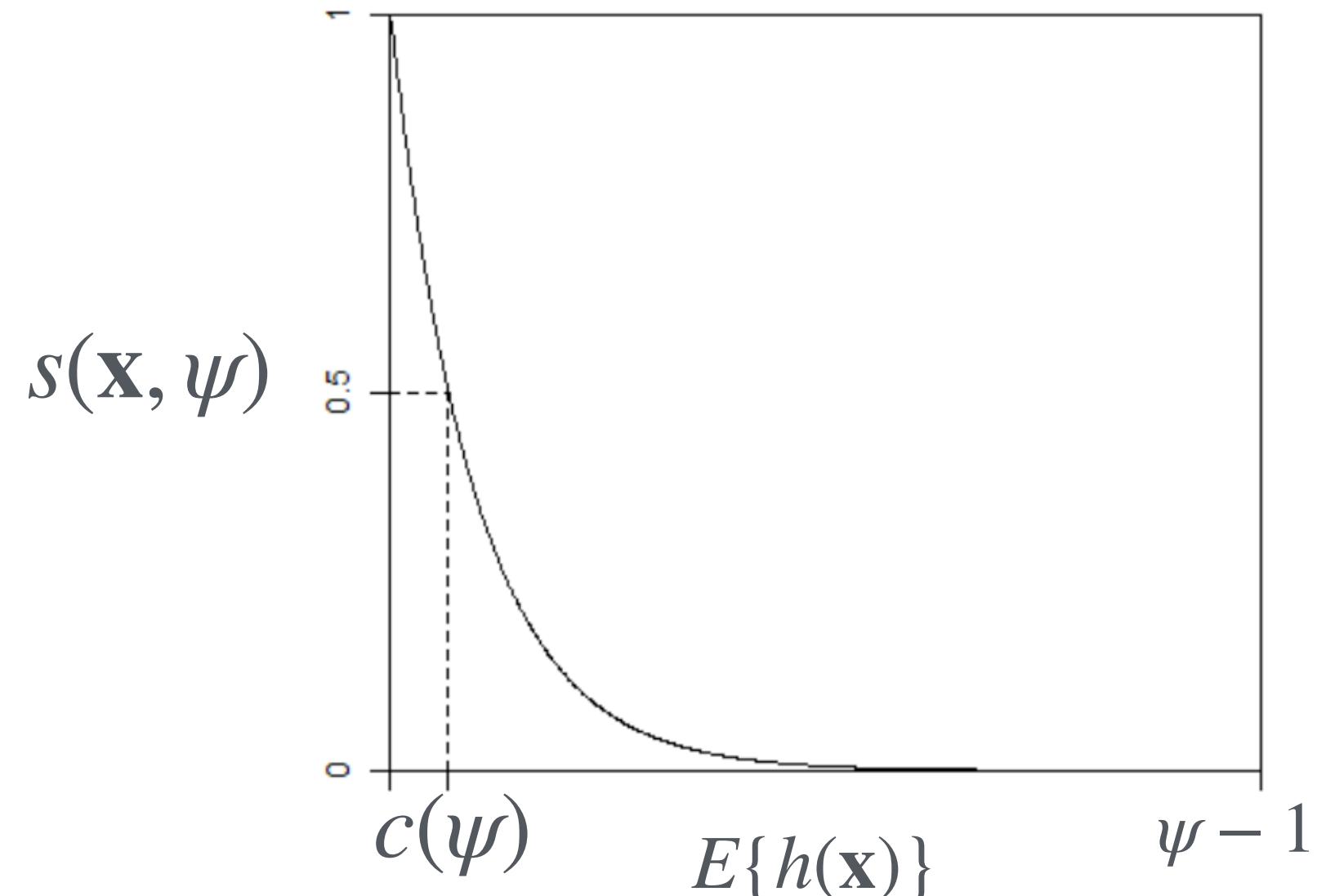
The anomaly score $s(\mathbf{x})$ of an instance \mathbf{x} is computed based on the normalized average path length from subset size ψ as

$$s(\mathbf{x}, \psi) = 2^{-\mathbb{E}\{h(\mathbf{x})\}/c(\psi)}$$

- ◆ Inliers: $E\{h(\mathbf{x})\} \rightarrow c(\psi)$, $s \rightarrow 0.5$
- ◆ Outliers: $E\{h(\mathbf{x})\} \rightarrow 0$, $s \rightarrow 1$
- ◆ $E\{h(\mathbf{x})\} \rightarrow \psi - 1$, $s \rightarrow 0$

Flagging anomalies:

- ◆ Samples with $s \ll 0.5$ can safely be regarded as inliers.
- ◆ Those with $s \rightarrow 1$ are considered anomalies. So, the threshold 0.5 is used the default value to flag anomalies.





```
from sklearn.ensemble import IsolationForest
```

```
isof_clf = IsolationForest(random_state=42, max_samples='auto')
```

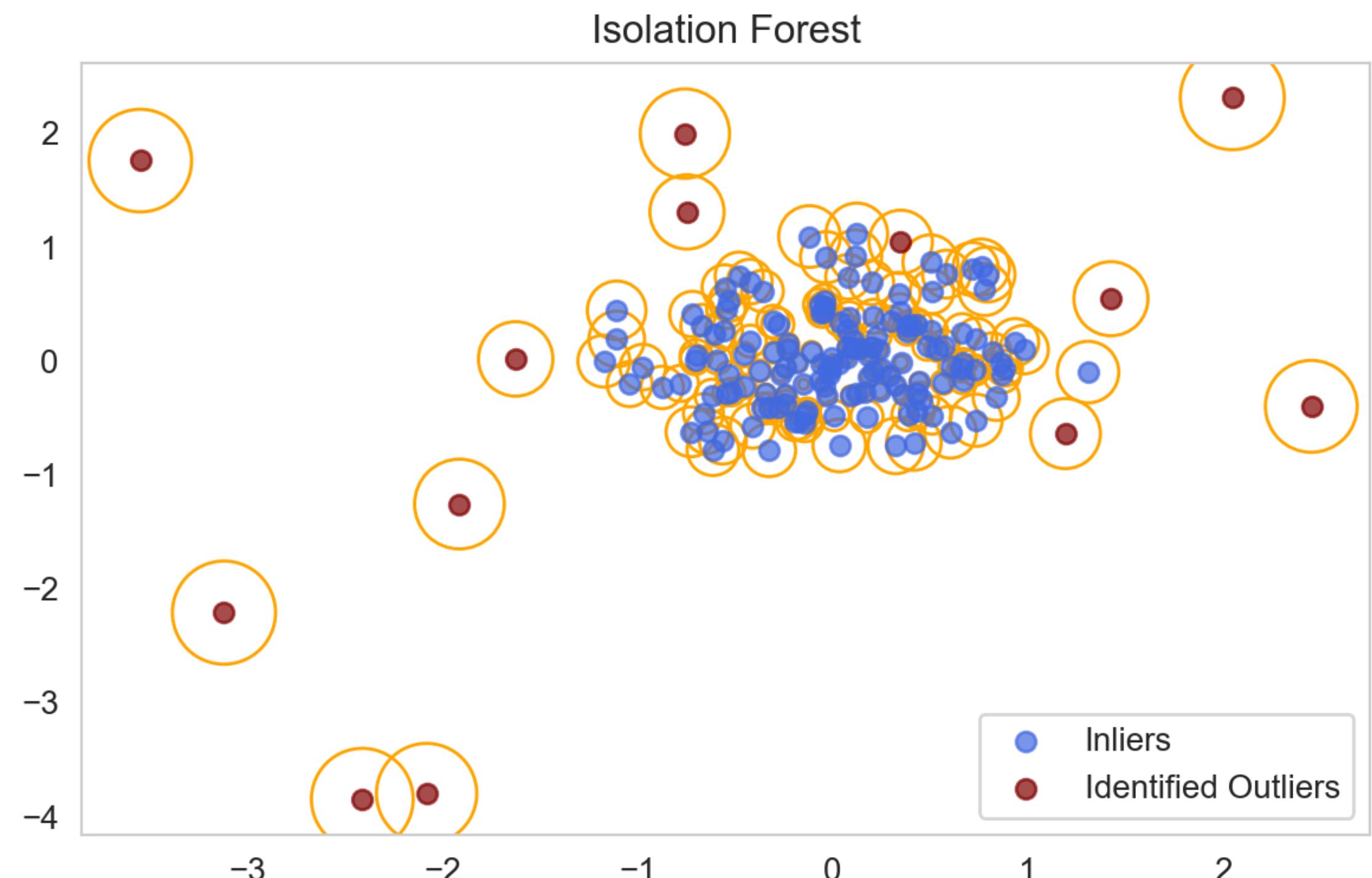
```
y_pred = isof_clf.fit_predict(input_df)
```

y_pred

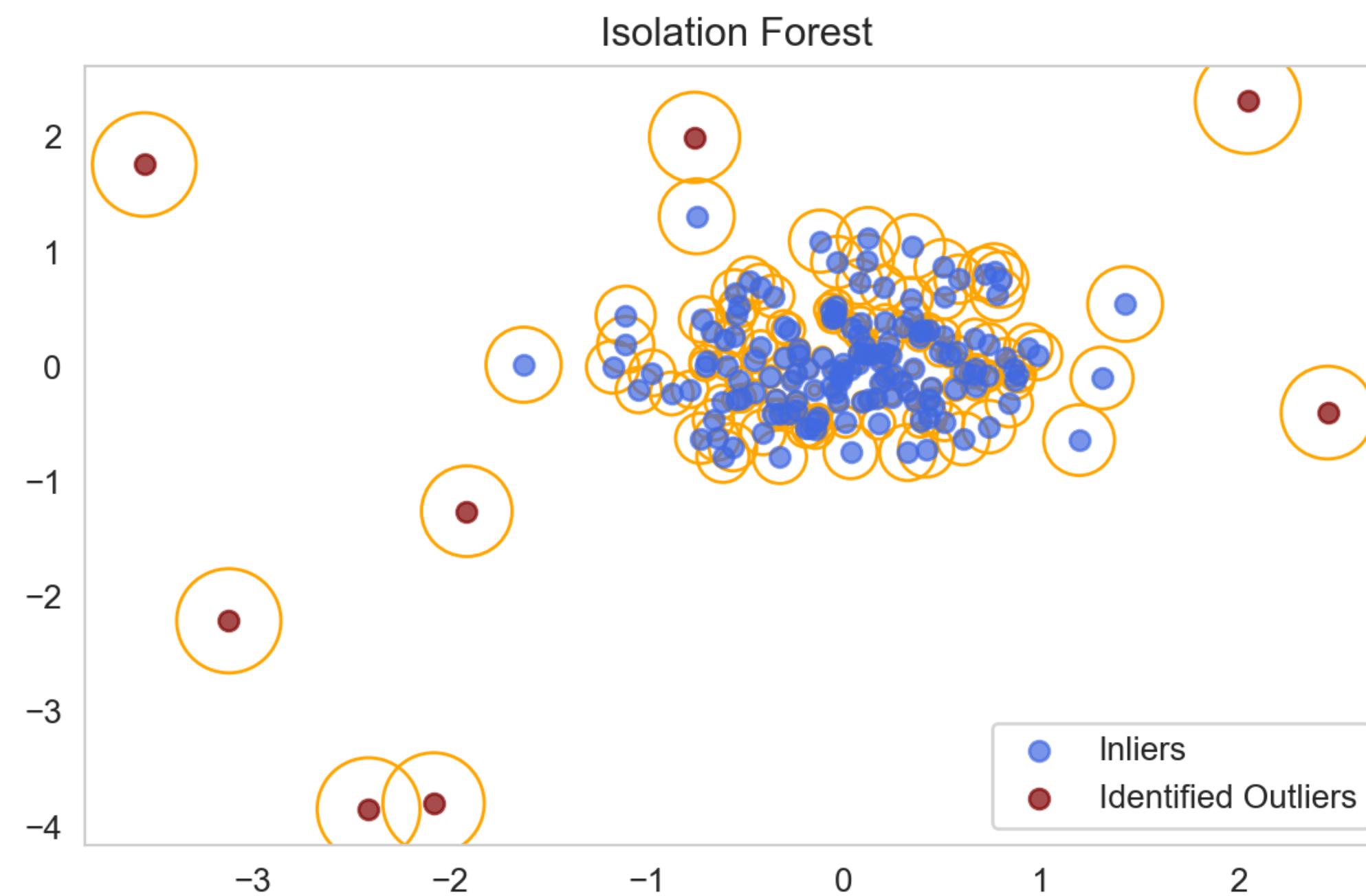
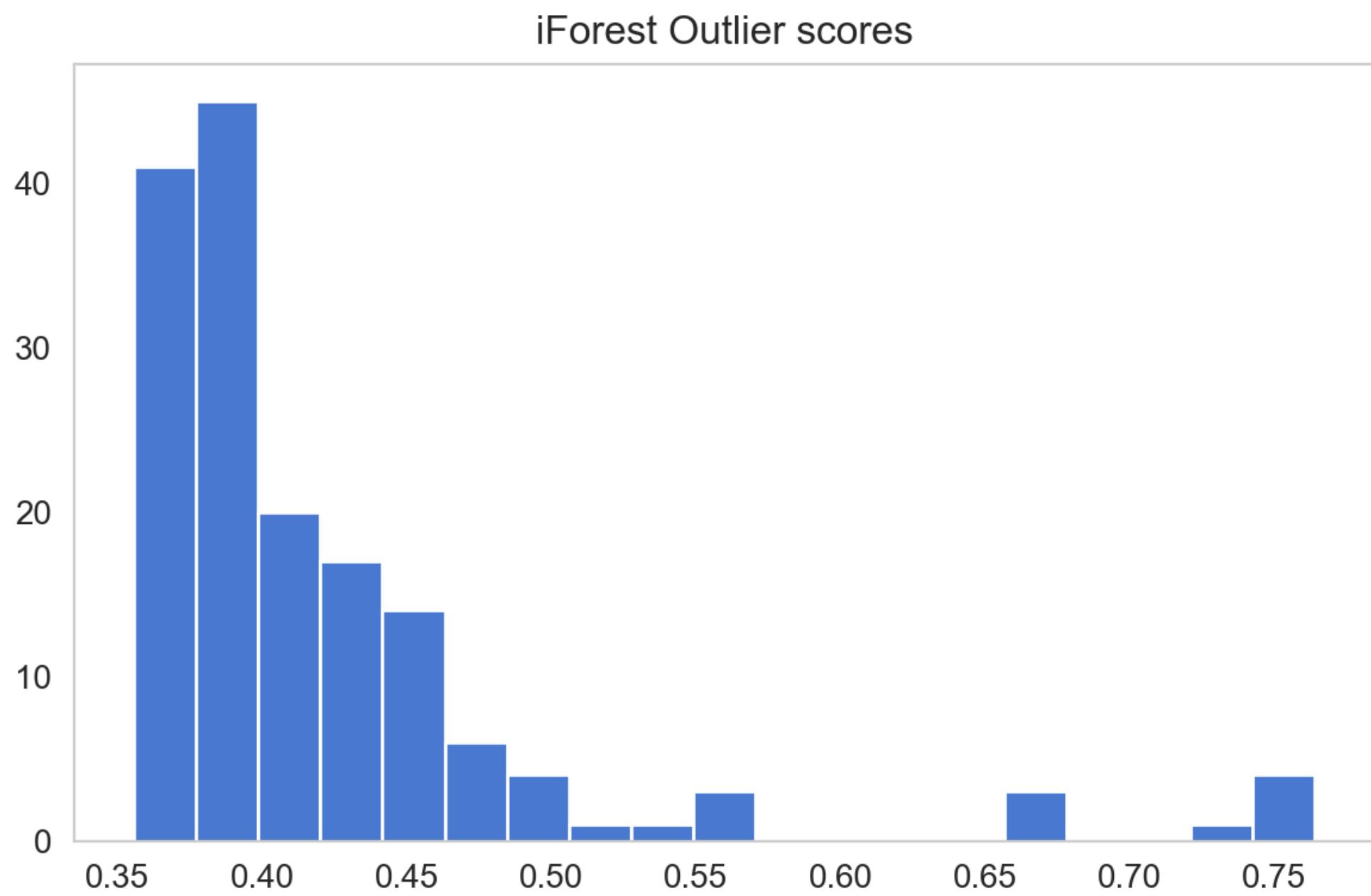
```
isof_scores = -isof_clf.score_samples(input_df)
```

isof_scores

	X1	X2
0	0.658597	0.258447
1	-0.302188	-0.031452
2	-0.423834	-0.228524
3	0.107967	0.210495
4	-0.080863	-0.188804
...
155	2.348850	-1.713857
156	1.997780	3.835402
157	3.679225	2.552838
158	-2.769838	-2.276746
159	2.472803	-3.586134



From the histogram of outlier scores, a more appropriate anomaly threshold can be determined.

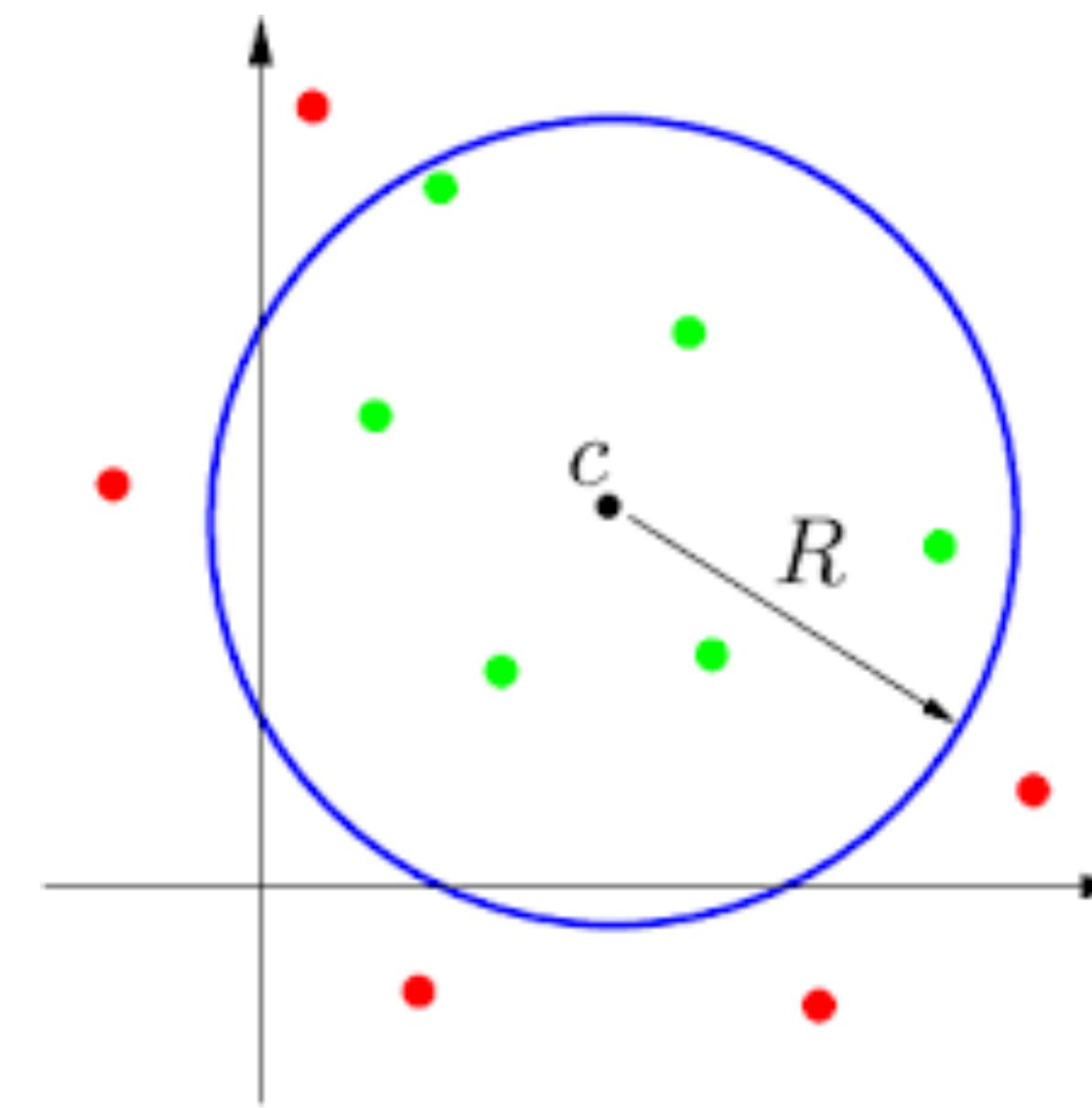
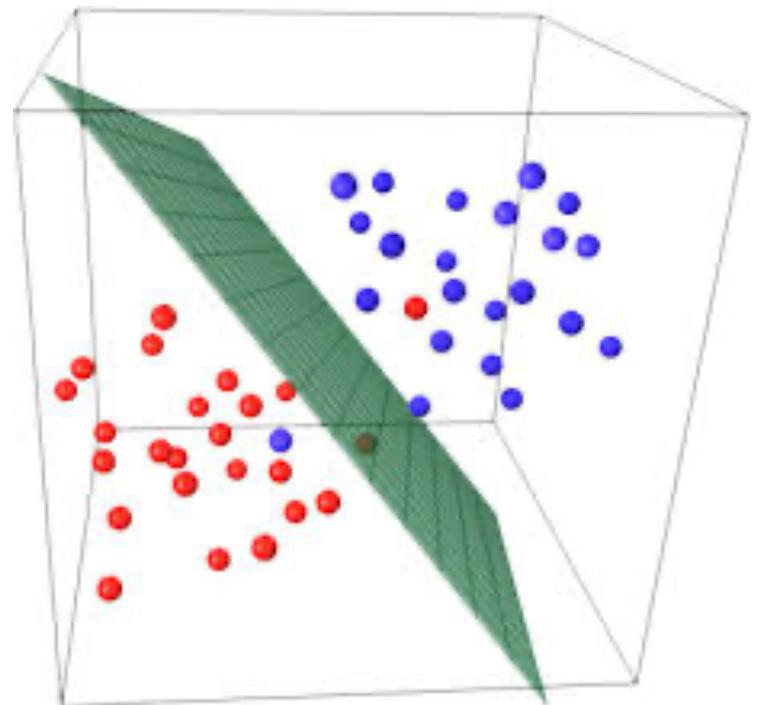
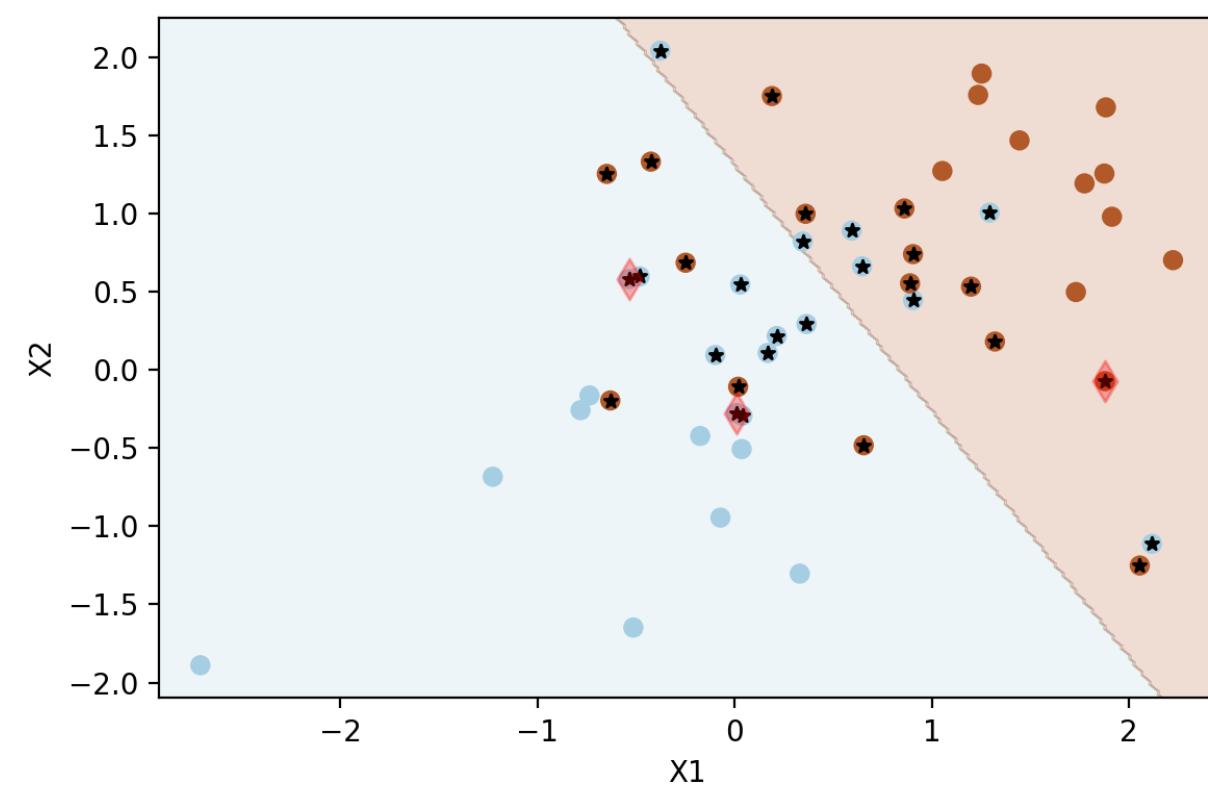


One-Class SVM

A regular SVM algorithm tries to find a hyperplane that best separates two classes of data points.

For one-class SVM,

- ◆ Training data is assumed to contain only normal data.
- ◆ Create a hypersphere with smallest radius or boundaries that attempt to "enclose" the data.



The optimal hypersphere with centre c and radius r with flexibility to tolerate outliers to an extent is obtained from

$$\min_{r,c,\zeta} r^2 + \frac{1}{\nu n} \sum_{i=1}^n \zeta_i$$

$$\text{subject to, } \|\Phi(x_i) - c\|^2 \leq r^2 + \zeta_i \quad \forall i = 1, 2, \dots, n$$

The hyperparameter "nu" $0 < \nu < 1$ trades-off the model's ability to categorize data as normal or abnormal.

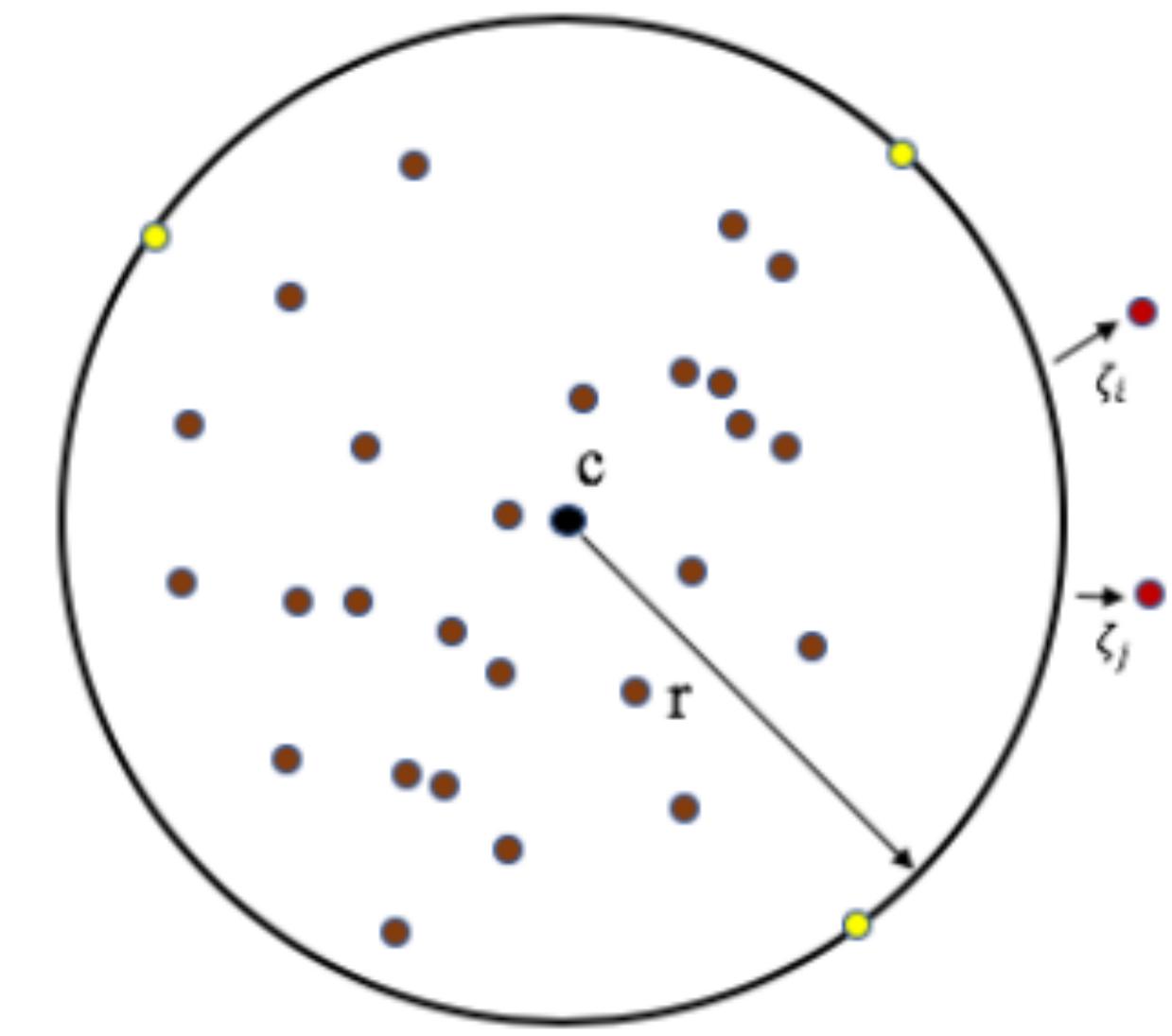
- ◆ Smaller ν = Larger sphere
- ◆ Should be set to % suspected outliers.

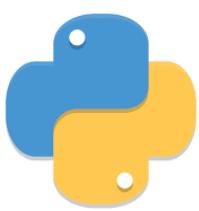
In Scikit-learn, outlier scores of OCSVM are the value of $\Phi(x_i)$, the signed distance to the separating hyperplane.

- ◆ Positive values -> inliers
- ◆ Negative values -> outliers

```
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)
```

[\[source\]](#)





```
from sklearn.svm import OneClassSVM
```

```
KERN = 'RBF'
```

```
NU, GAMMA = 0.01, 0.1
```

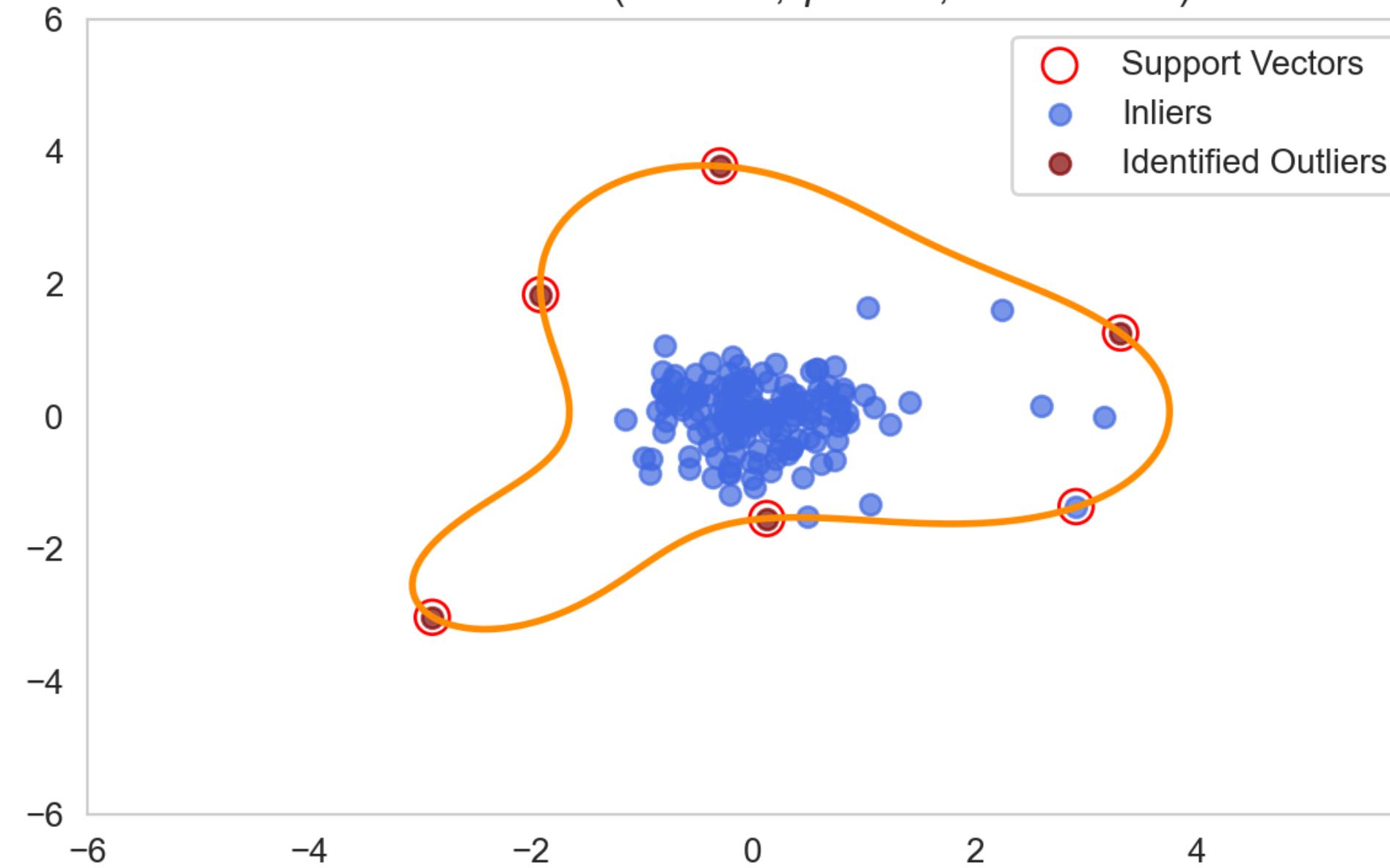
```
ocsvm_clf = OneClassSVM(nu=NU, kernel=KERN.lower(), gamma=GAMMA)  
ocsvm_clf.fit(input_df)
```

```
y_pred = ocsvm_clf.predict(input_df)
```

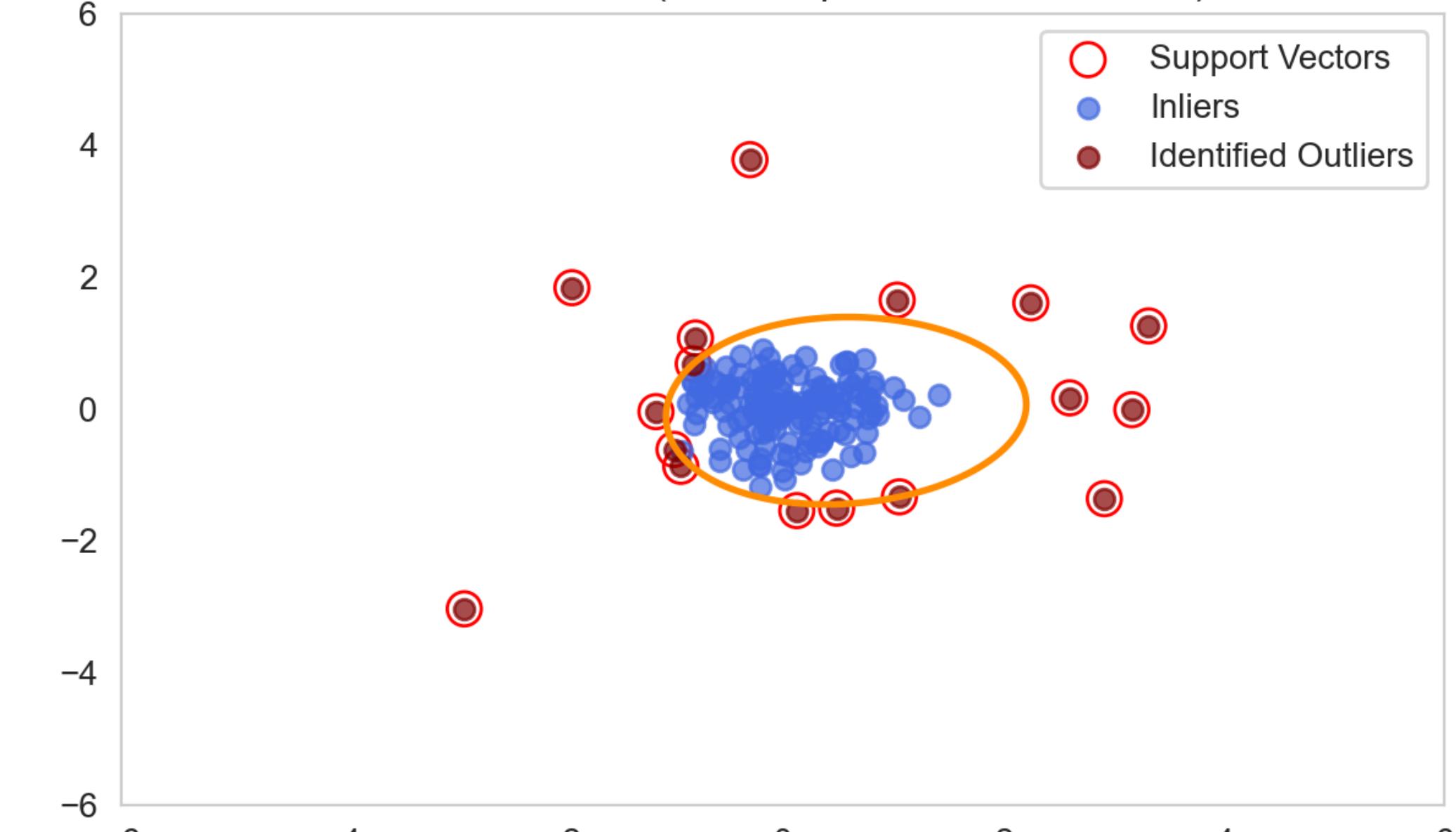
```
ocsvm_scores = ocsvm_clf.decision_function(input_df)
```

	x1	x2
0	0.658597	0.258447
1	-0.302188	-0.031452
2	-0.423834	-0.228524
3	0.107967	0.210495
4	-0.080863	-0.188804
...
155	2.348850	-1.713857
156	1.997780	3.835402
157	3.679225	2.552838
158	-2.769838	-2.276746
159	2.472803	-3.586134

One-class SVM ($\nu=0.01, \gamma = 0.1$, kernel=RBF)

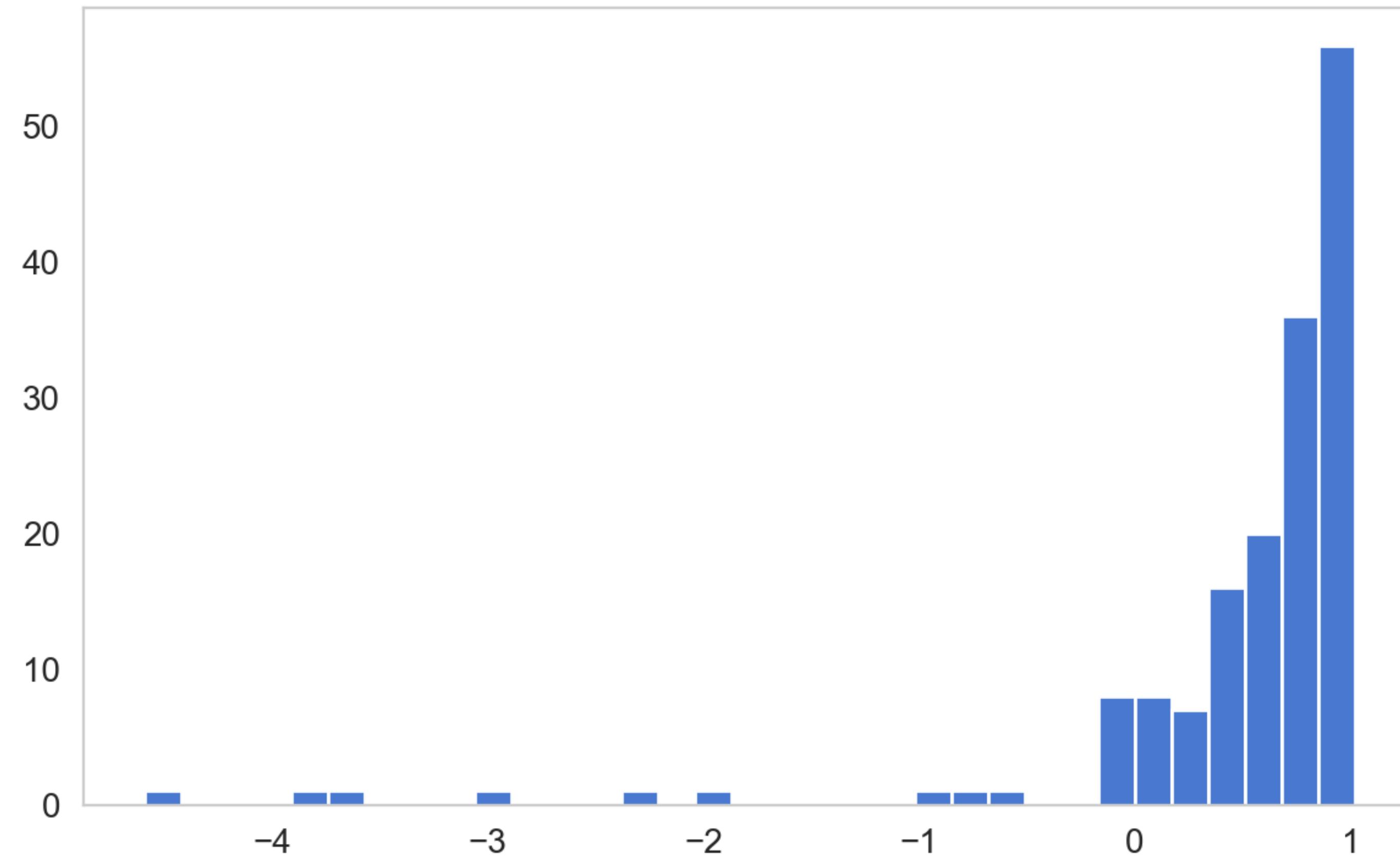


One-class SVM ($\nu=0.1, \gamma = 0.1$, kernel=RBF)





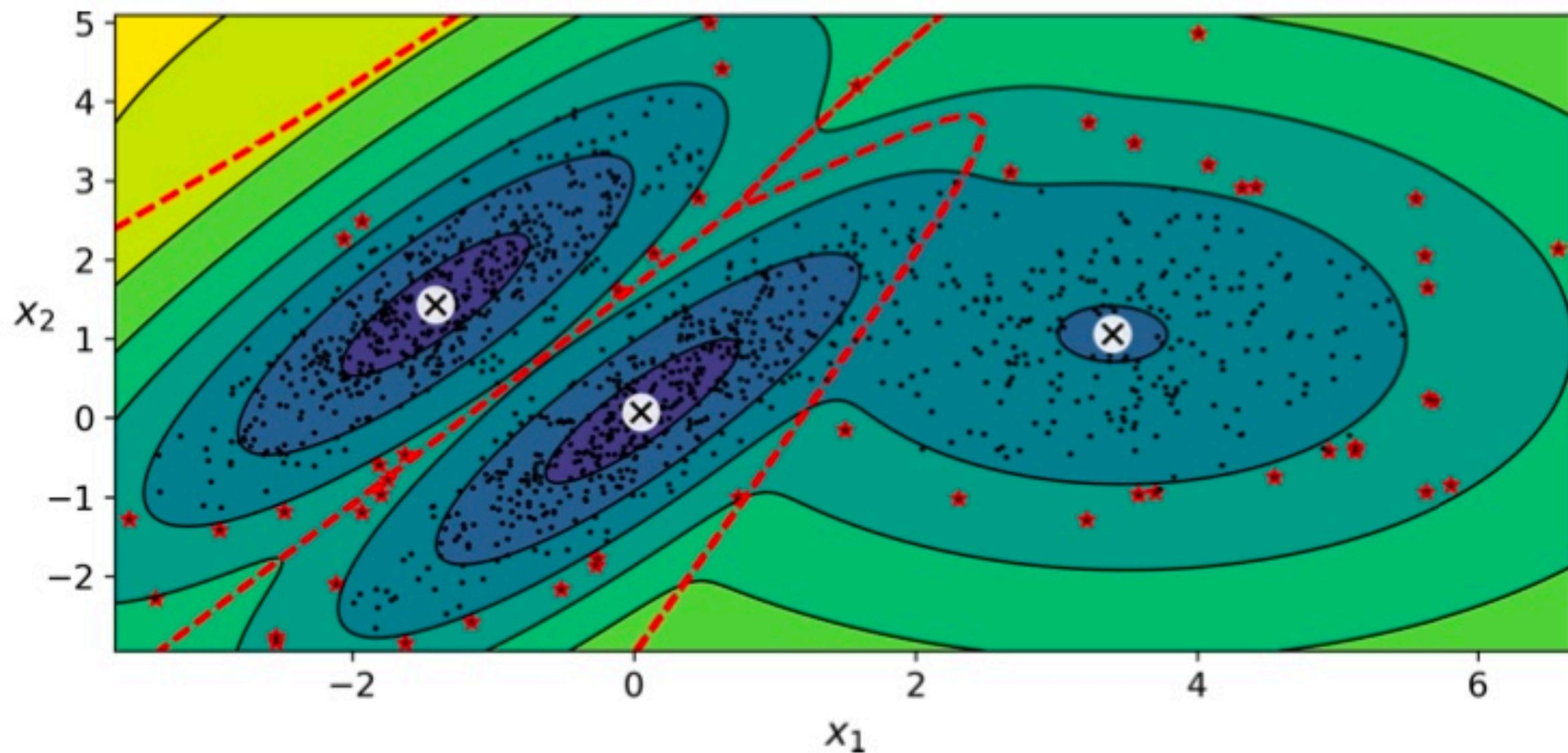
OCSVM Outlier scores



Clustering-based Anomaly Detection: GMM

Retrieve densities $f(\mathbf{x})$ of all data points

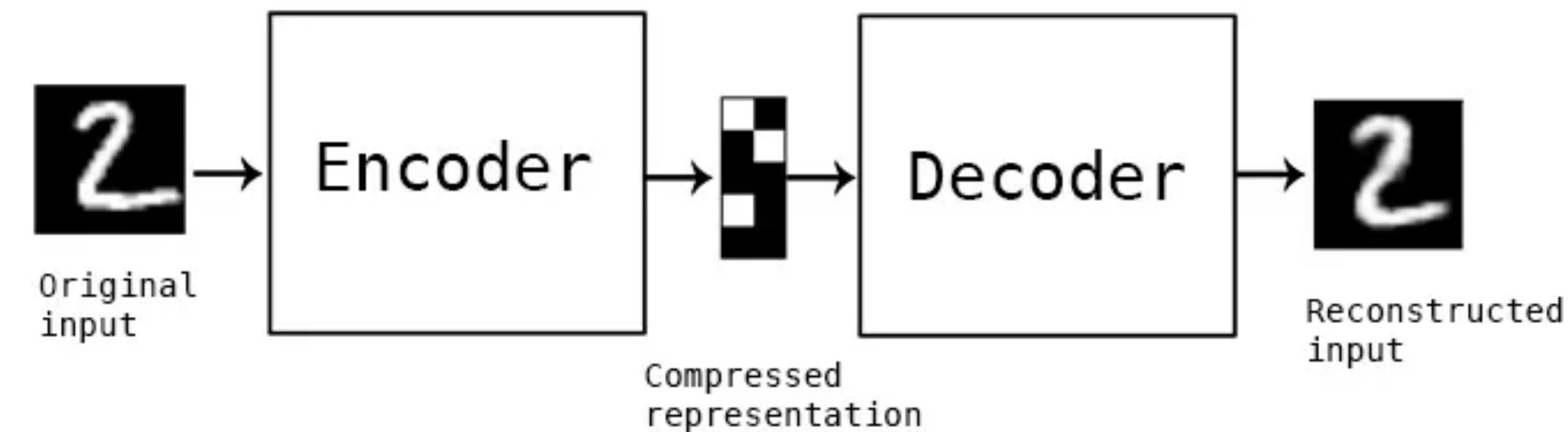
Any instance located in a low-density region, e.g., beyond a given low-percentile threshold, can be considered an anomaly.



```
densities = gm.score_samples(X)
PERC_THRES = 4
density_threshold = np.percentile(densities, PERC_THRES)
anomalies = X[densities < density_threshold]
```

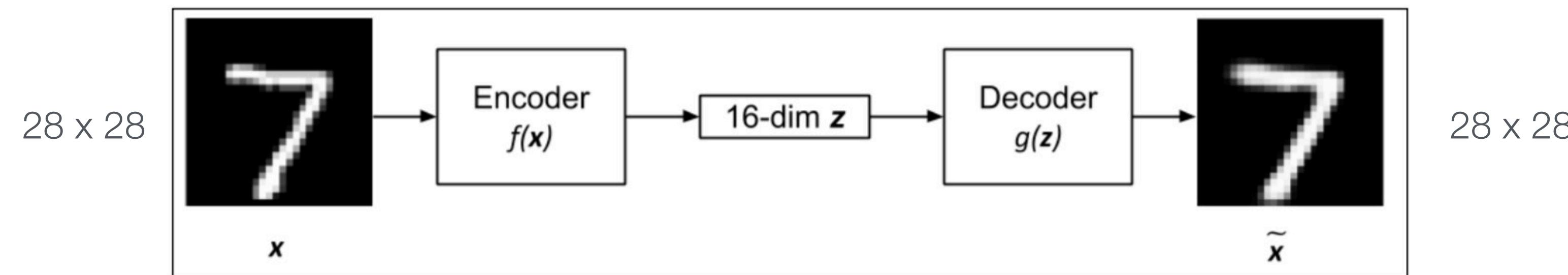
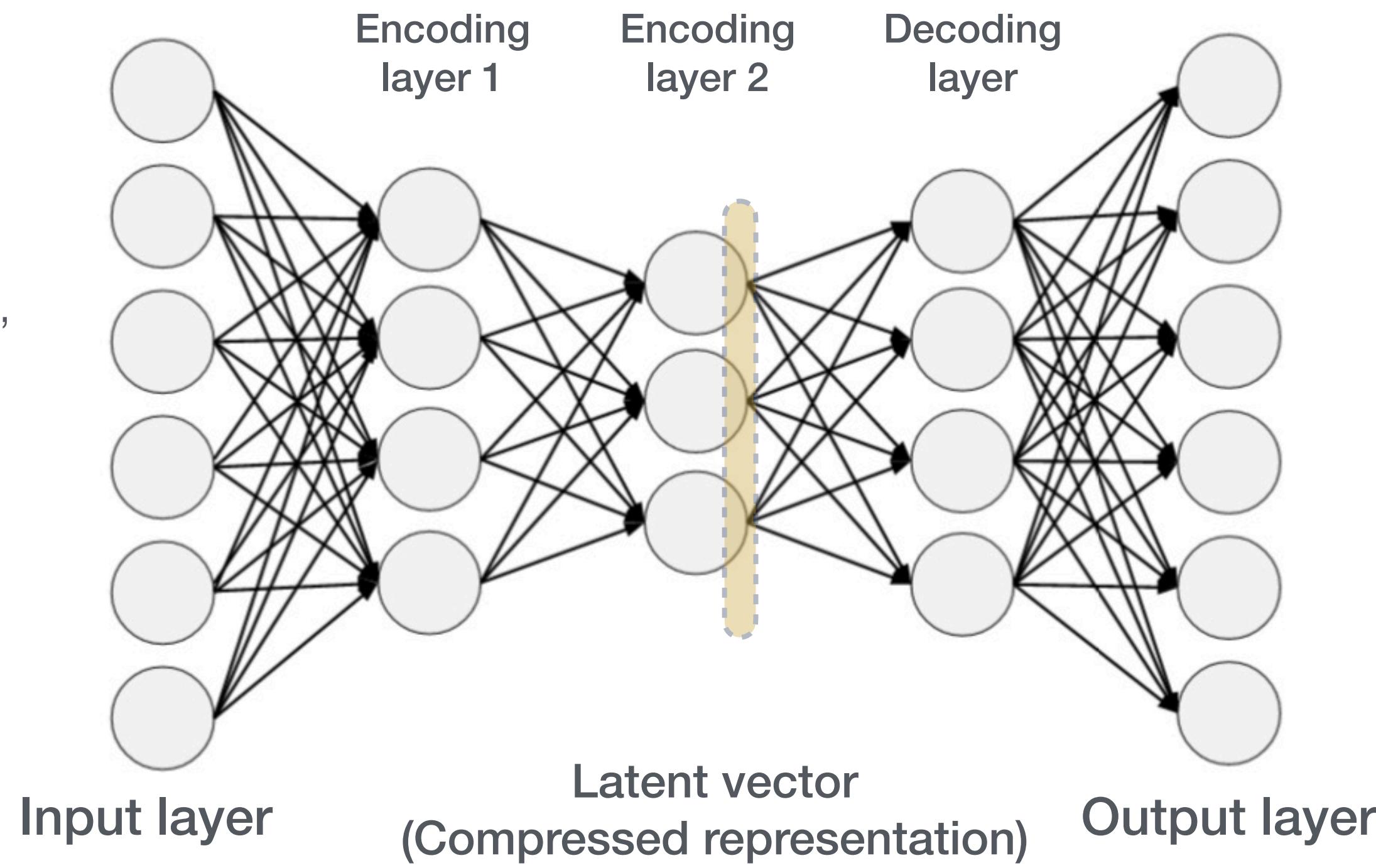
Autoencoder (AE)

Feed-forward network intends to reproduce the input or its variants by using encoder-decoder architecture.

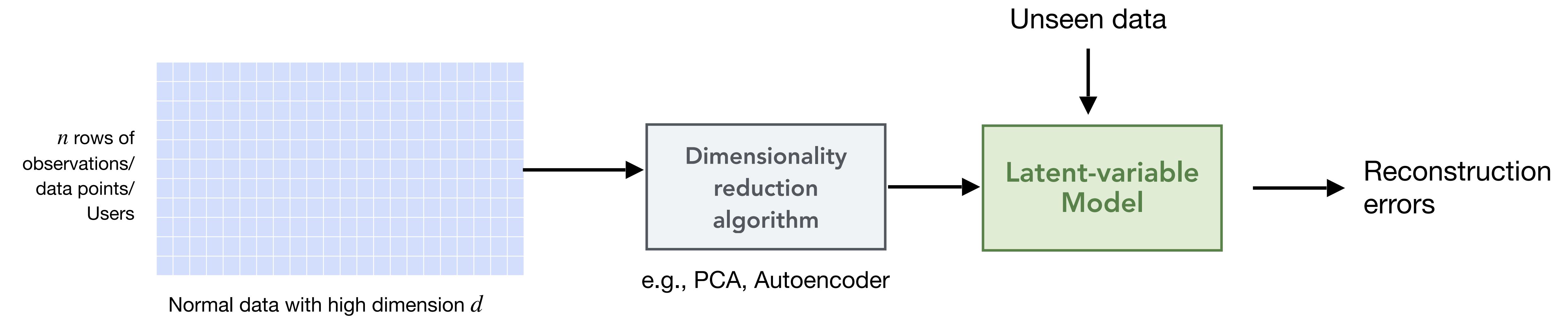


Autoencoder for Dimensionality Reduction

The network can be shallow or deep, depending on # hidden layers in the encoder/decoder.



Latent-Variable-based Approach





Anomaly Detection Lab

(Group of 4 to 6)

Part I

From the diamond dataset (sheet 'Diamond' in anomaly-detection.xlsx), use the columns carat, depth, and price to determine the number of unusual diamonds from the dataset based on the MHD at 1% threshold.

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48
0.24	Very Good	I	VVS1	62.3	57	336	3.95	3.98	2.47
0.26	Very Good	H	SI1	61.9	55	337	4.07	4.11	2.53
0.22	Fair	E	VS2	65.1	61	337	3.87	3.78	2.49
0.23	Very Good	H	VS1	59.4	61	338	4	4.05	2.39
0.3	Good	J	SI1	64	55	339	4.25	4.28	2.73
0.23	Ideal	J	VS1	62.8	56	340	3.93	3.9	2.46
0.22	Premium	F	SI1	60.4	61	342	3.88	3.84	2.33
0.31	Ideal	J	SI2	62.2	54	344	4.35	4.37	2.71
0.2	Premium	E	SI2	60.2	62	345	3.79	3.75	2.27
0.32	Premium	E	I1	60.9	58	345	4.38	4.42	2.68
0.3	Ideal	I	SI2	62	54	348	4.31	4.34	2.68
0.3	Good	J	SI1	63.4	54	351	4.23	4.29	2.7
0.3	Good	J	SI1	63.8	56	351	4.23	4.26	2.71



Part II

The bearing dataset ('BearingTest.csv') contains the time series data from four sensors.

- ◆ Use data from '2004-02-12 11:02:39' to '2004-02-13 23:52:39' as the train data and from '2004-02-13 23:52:39' as the test data.
- ◆ Apply PCA with two PCs to the training data to obtain the latent-variable model.
- ◆ Plot the histogram of mean absolute error (MAE) from reconstruction errors of individual training data points and set the anomaly threshold to 105% of the maximum MAE.
- ◆ Visualize the MAE of train data and test data over time and identify the time when the model detects anomalies.

