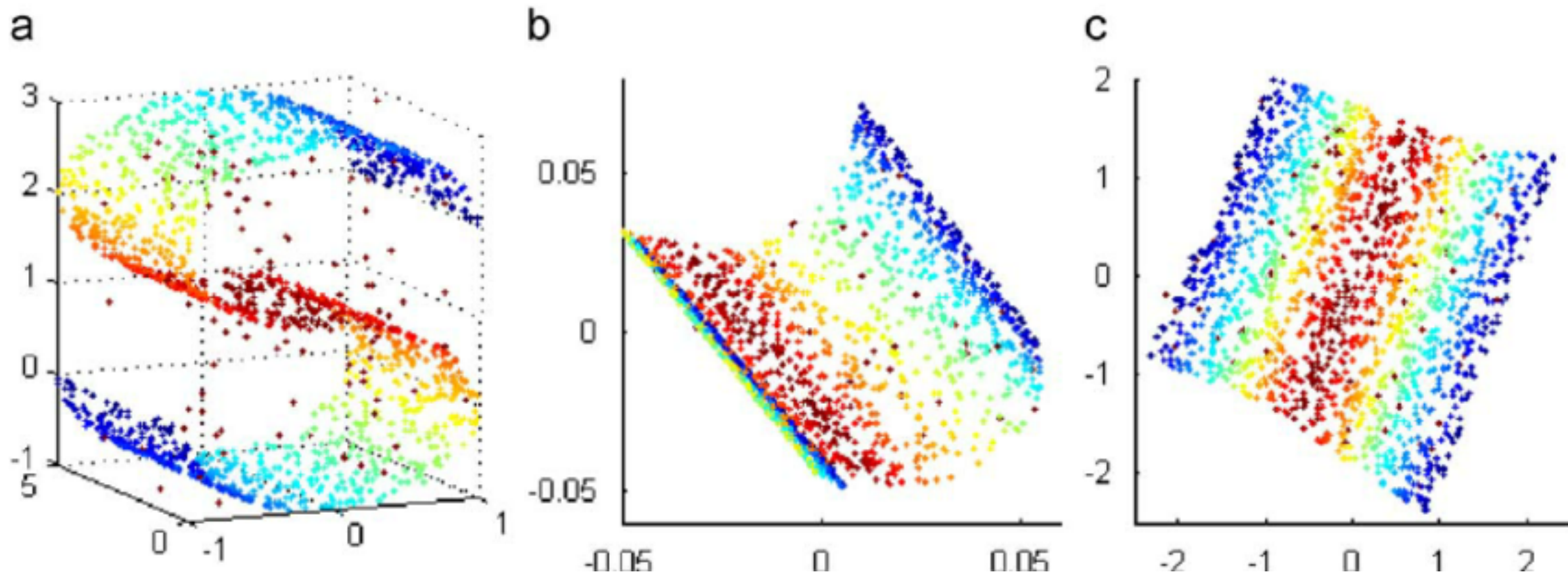# Dimensionality reduction

AI FUNDAMENTALS

Nemanja Radojkovic
Senior Data Scientist

Definition

"Dimensionality reduction is the process of reducing the number of variables under consideration by obtaining a set of principal variables."

Why?

Pro's

- Reduce overfitting

- Obtain independent features

- Lower computational intensity

- Enable visualization

Con's

- Compression => Loss of information => loss of performance

Types

Feature selection (B ? A)

- Selecting a **subset** of existing features, based on predictive power

- **Non-trivial problem:** Looking for the best "team of features", not individually best features!

Feature extraction (B ? A)

- Transforming and combining existing features into new ones.

- Linear or non-linear **projections**.

Common algorithms

Linear (faster, deterministic)

- Principal Component Analysis (PCA)

```
from sklearn.decomposition \
  import PCA
```

- Latent Dirichlet Allocation

```
from sklearn.decomposition \
  import LatentDirichletAllocation
```

Non-linear (slower, non-deterministic)

- Isomap

```
from sklearn.manifold import Isomap
```

- t-distributed Stochastic Neighbor Embedding (t-SNE)

```
from sklearn.manifold import TSNE
```
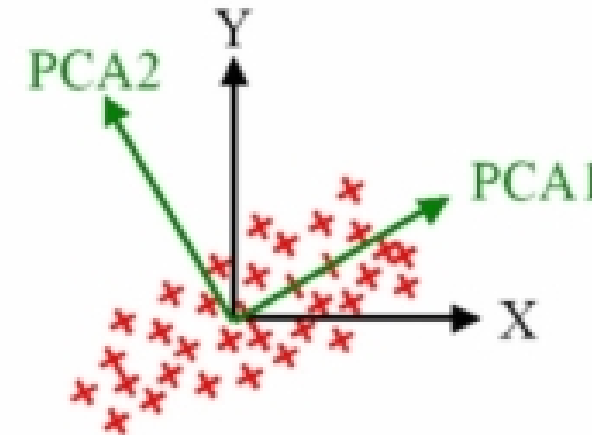
Principal Component Analysis (PCA)

Family: Linear methods.

Intuition:

- **Principal components** are directions of highest variability in data.

- **Reduction** = keeping only top #N principal components.

Assumption: Normal distribution of data.

Caveat: Very sensitive to outliers.

Code example:

```python
from sklearn.decomposition import PCA

pca = PCA(n_dimensions=3)

X_reduced = pca.fit_transform(X)
```
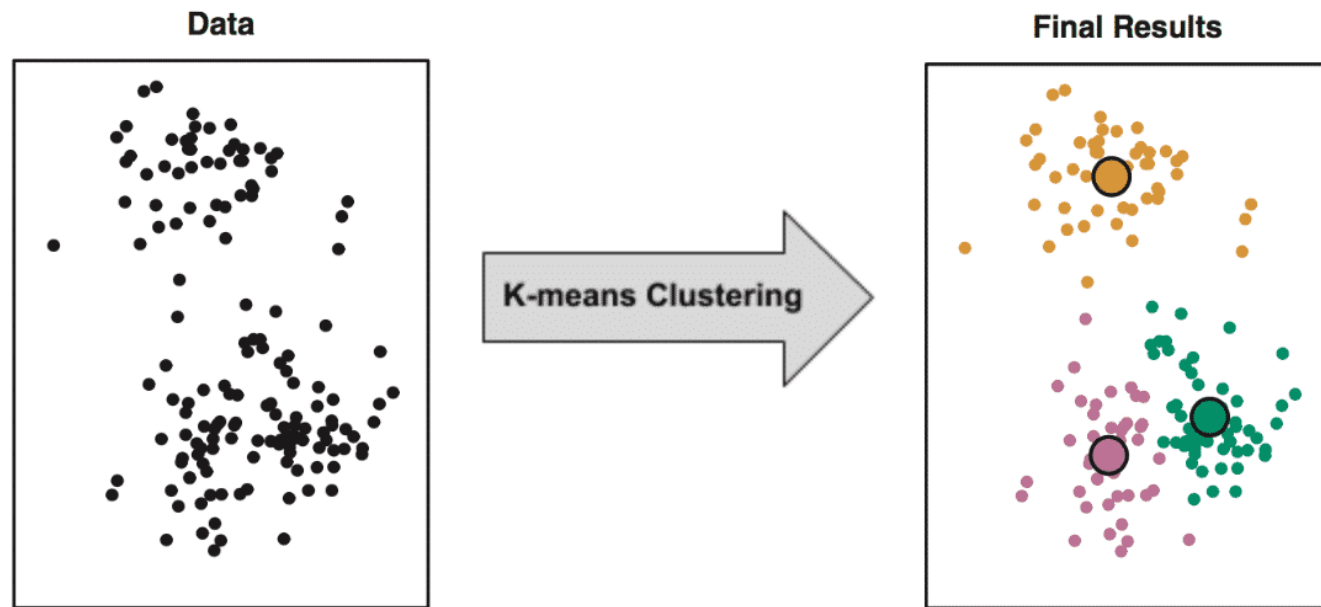
Use it wisely!

AI FUNDAMENTALS

Clustering

AI FUNDAMENTALS

Nemanja Radojkovic
Senior Data Scientist

What is clustering?

Data → K-means Clustering → Final Results

Cluster = Group of entities or events sharing similar attributes.

Clustering (AI) = The process of applying Machine Learning algorithms for automatic discovery of clusters.

Popular clustering algorithms

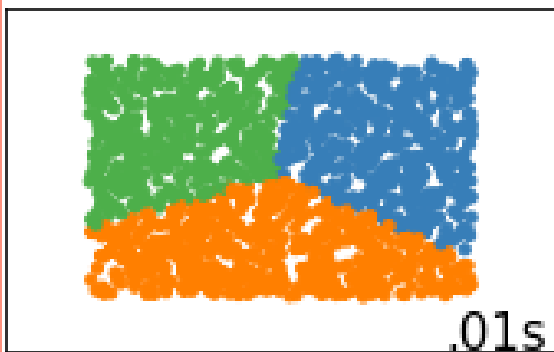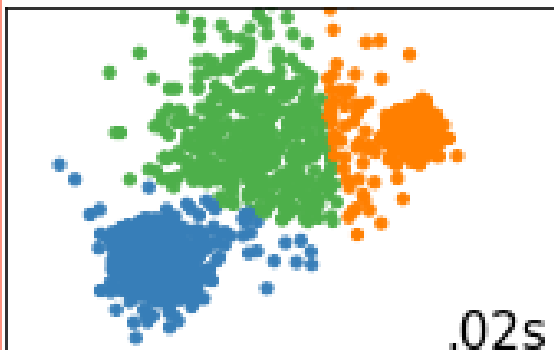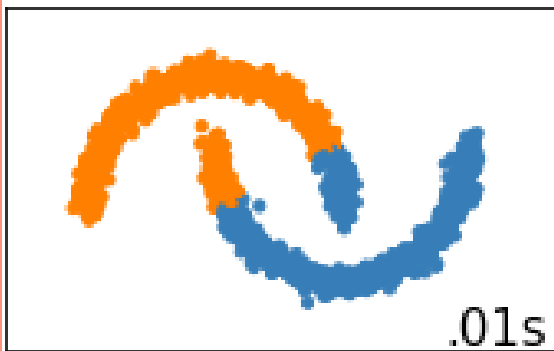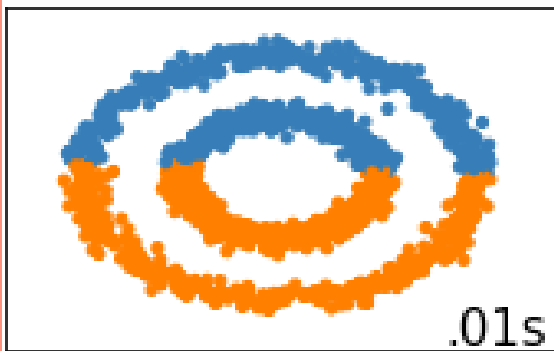## KMeans clustering

```python
from sklearn.cluster import KMeans
```

Spectral clustering

```python
from sklearn.cluster import SpectralClustering
```
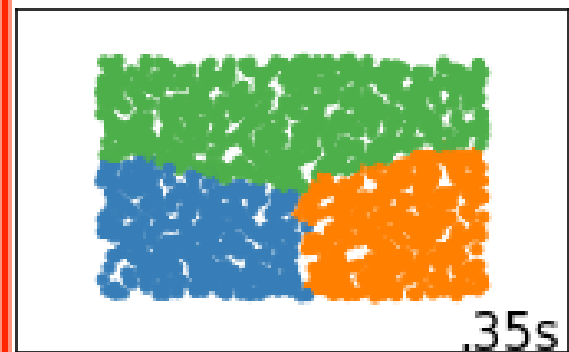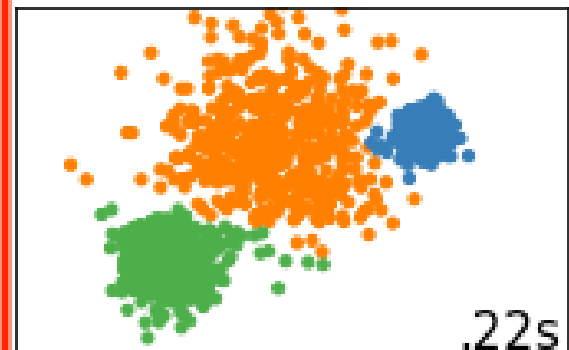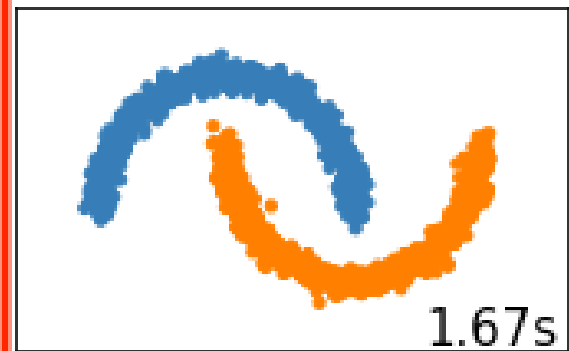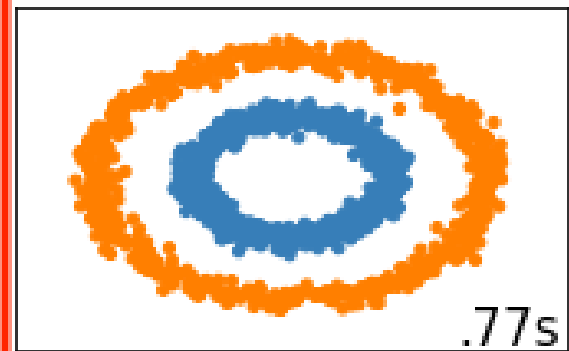
DBSCAN

```python
from sklearn.cluster import DBSCAN
```

MiniBatchKMeans	SpectralClustering	DBSCAN
.01s	.77s	.01s
.01s	1.67s	.01s
.02s	.22s	.01s
.01s	.35s	.01s

MiniBatchKMeans | SpectralClustering | DBSCAN
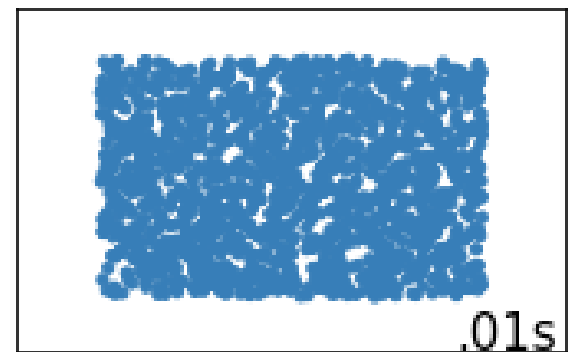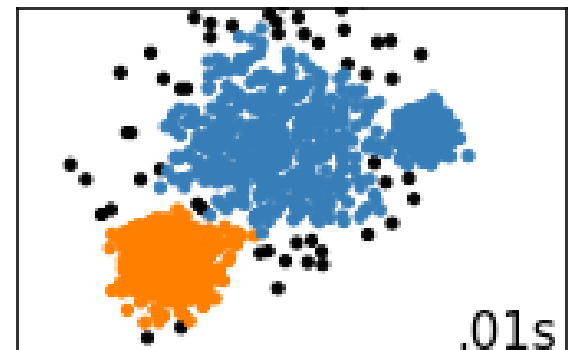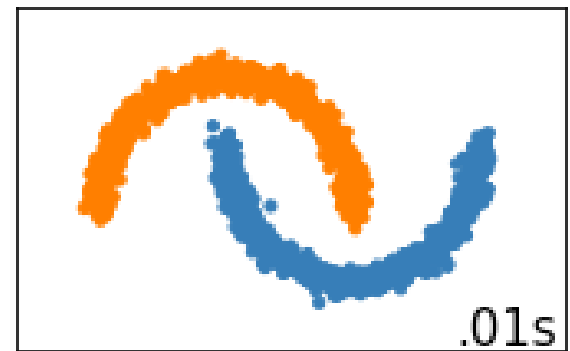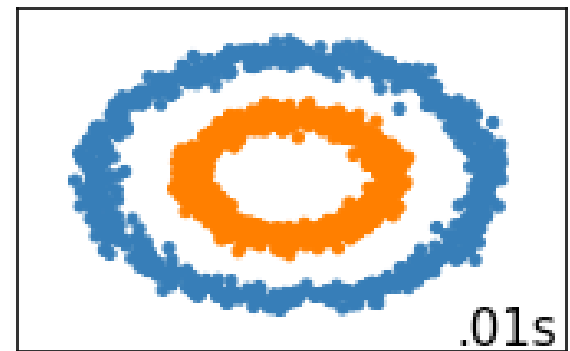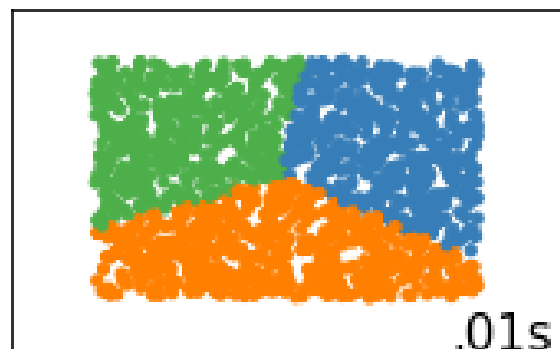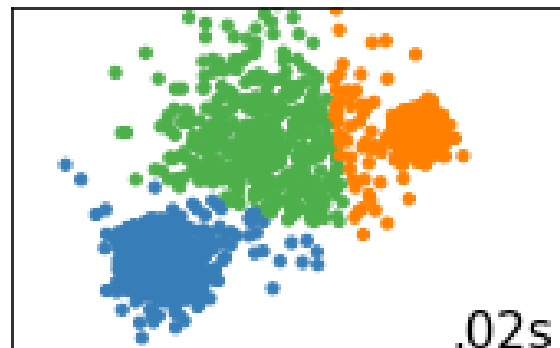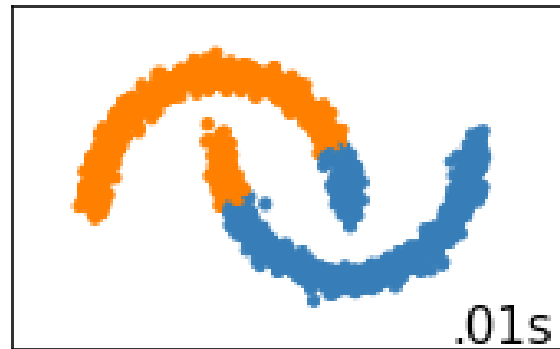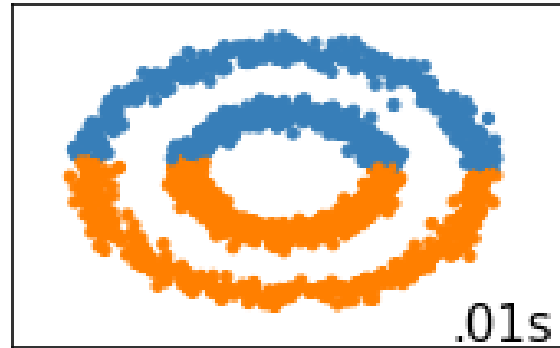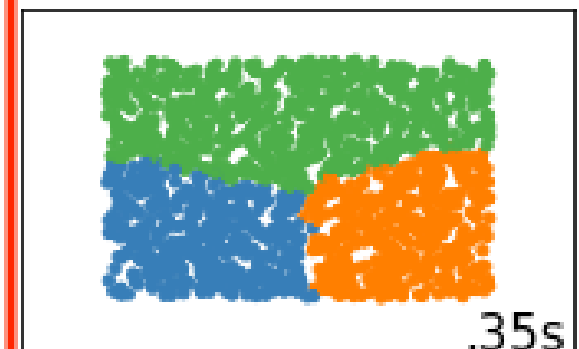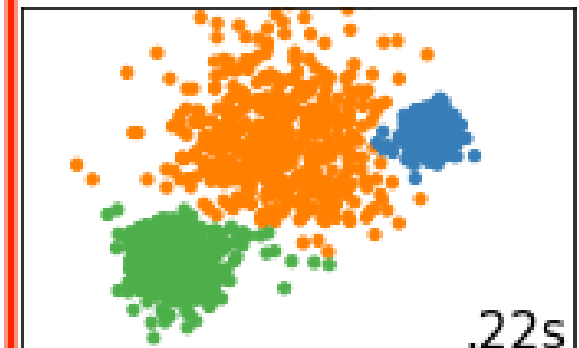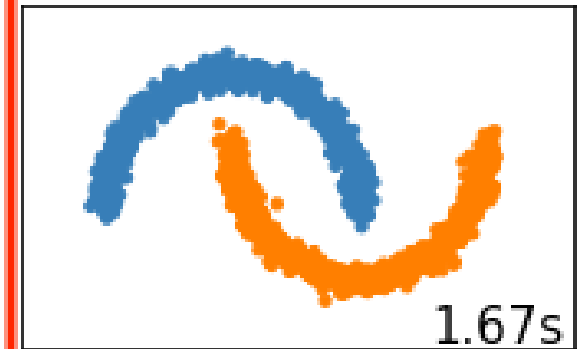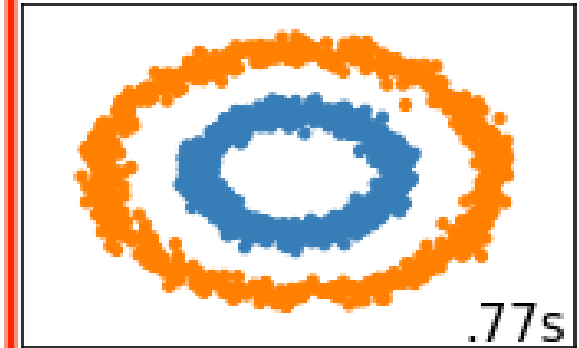
MiniBatchKMeans | SpectralClustering | DBSCAN
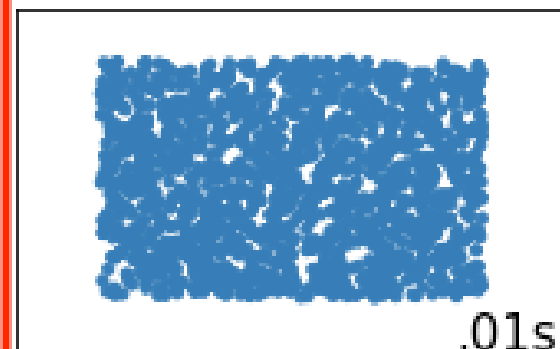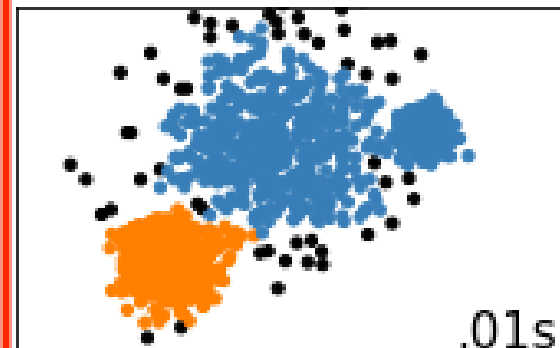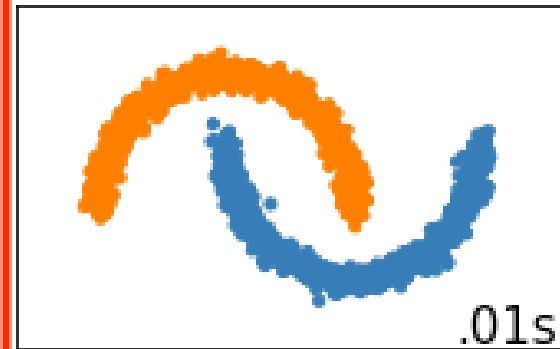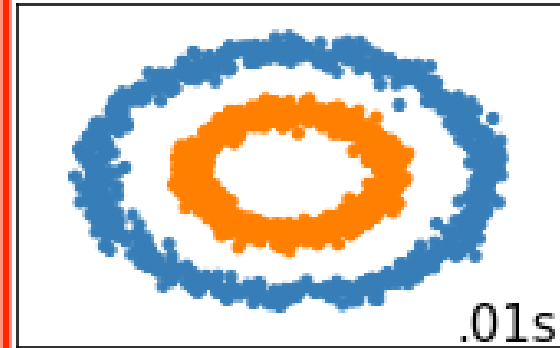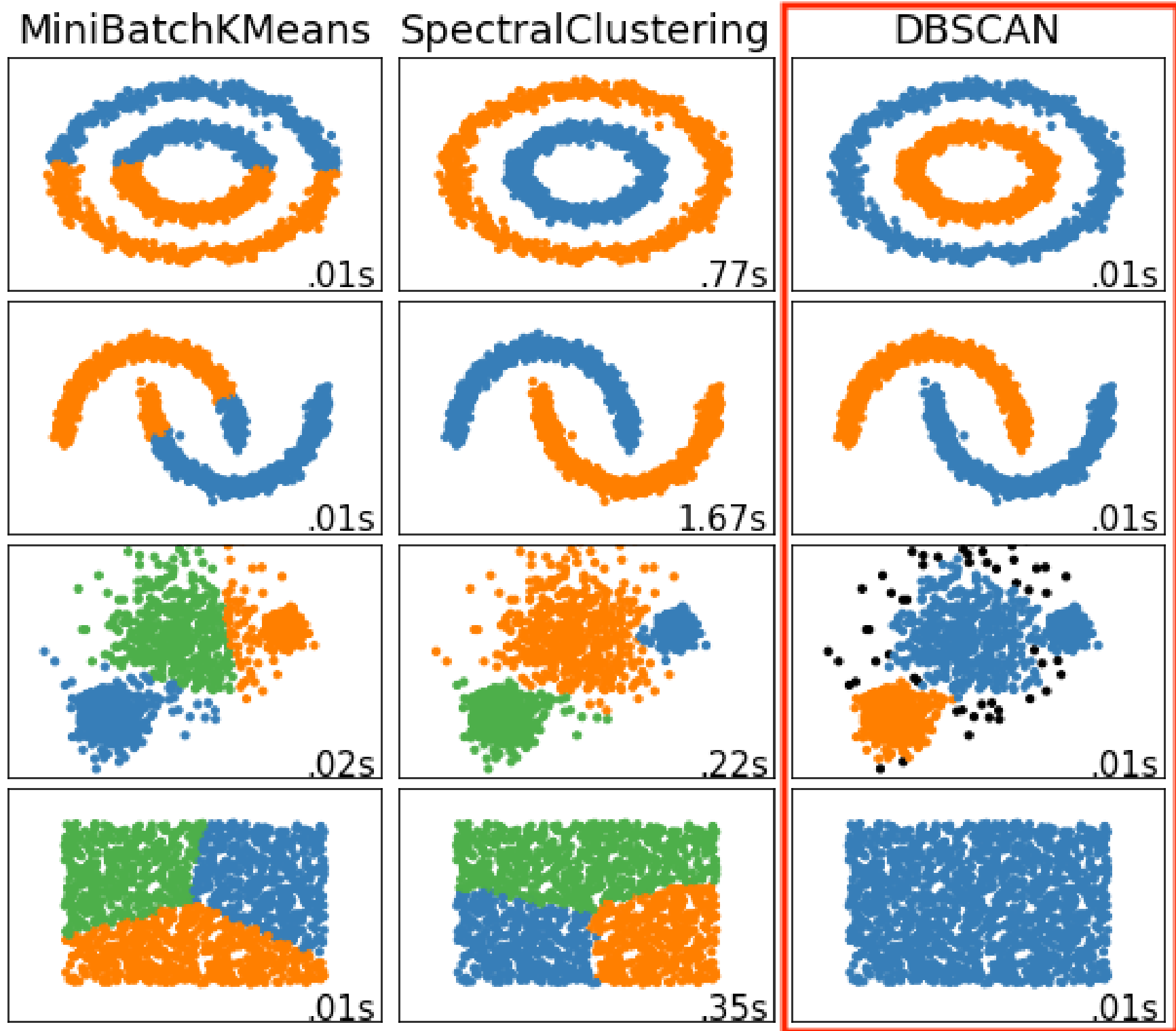
How many clusters do I have?
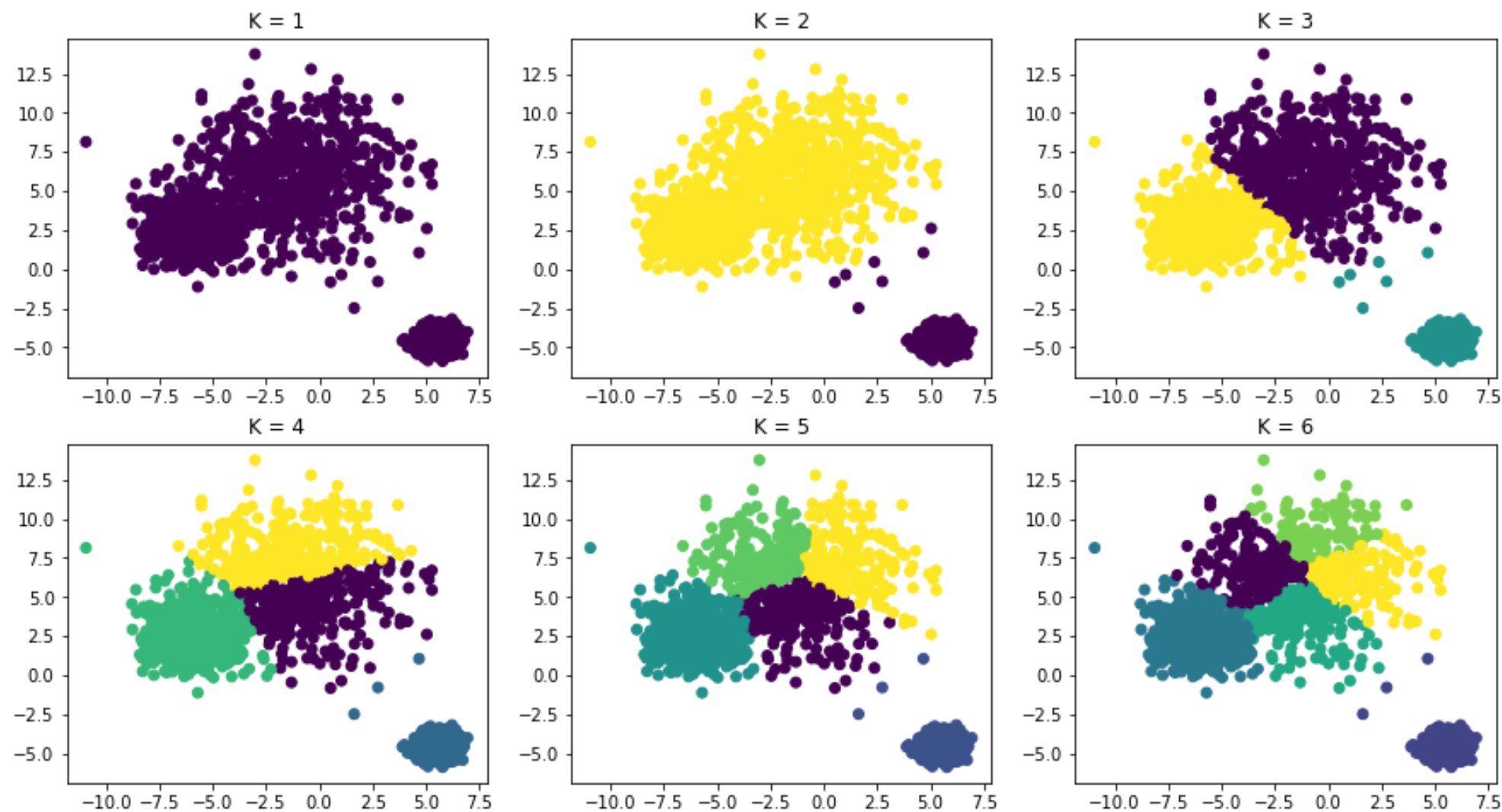
–> **Elbow method!**

How many clusters do I have?

Sum of squared distances vs Number of clusters
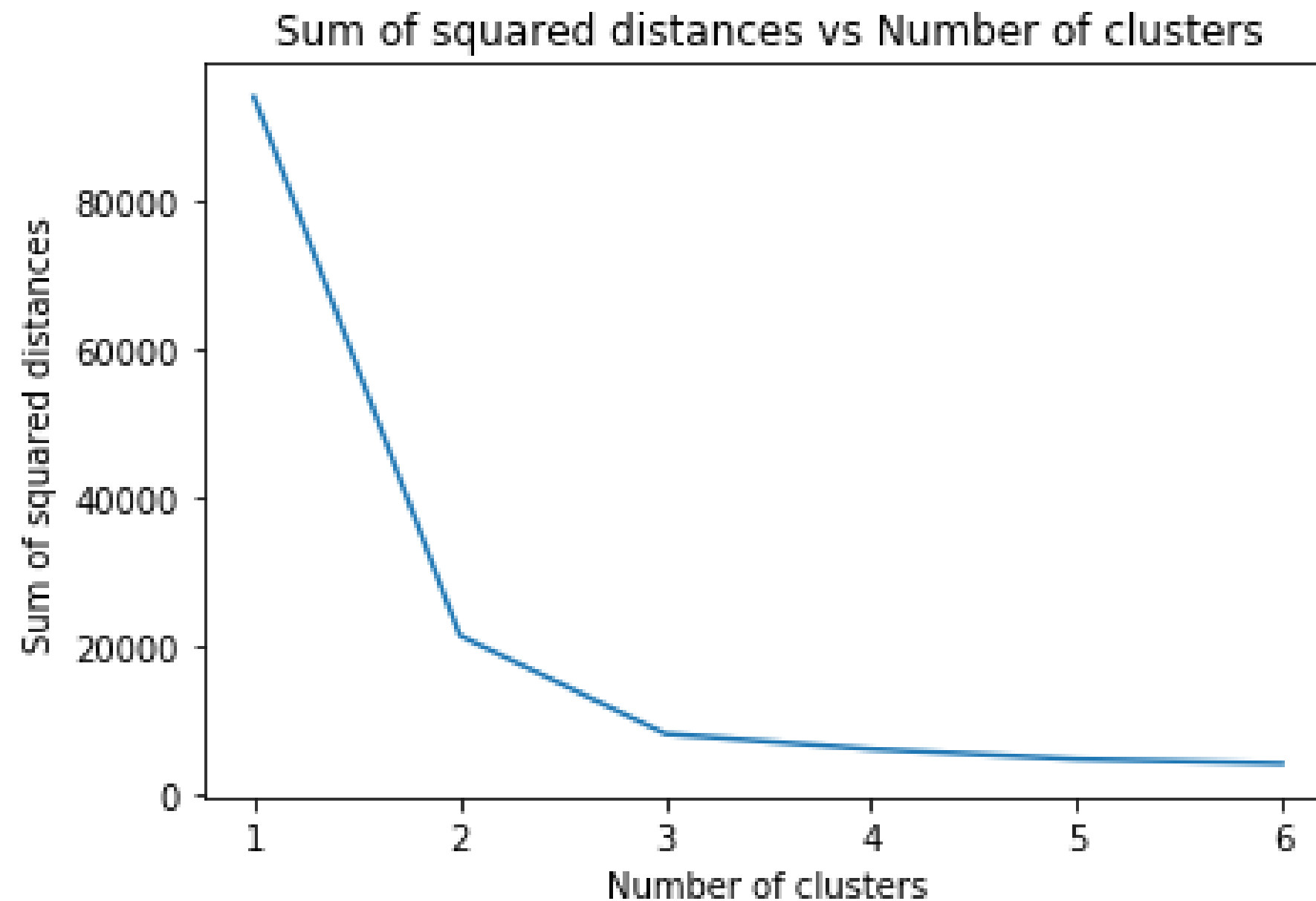
Cluster analysis and tuning

Unsupervised (no "ground truth", no expectations)

- Variance Ratio Criterion: `sklearn.metrics.calinski_harabaz_score`
  - *"What is the average distance of each point to the center of the cluster AND what is the distance between the clusters?"*

- Silhouette score: `sklearn.metrics.silhouette_score`
  - *"How close is each point to its own cluster VS how close it is to the others?"*

Supervised ("ground truth"/expectations provided)

- Mutual information (MI) criterion: `sklearn.metrics.mutual_info_score`

- Homogeneity score: `sklearn.metrics.homogeneity_score`

Explore, experiment and tune!

AI FUNDAMENTALS

Anomaly detection

AI FUNDAMENTALS

Nemanja Radojkovic
Senior Data Scientist

Definition and use cases

- Detecting unusual entities or events.

- Hard to define what's odd, but possible to define what's normal.

- **Use cases**
  - Credit card fraud detection

  - Network security monitoring

  - Heart-rate monitoring

Approaches: Thresholding
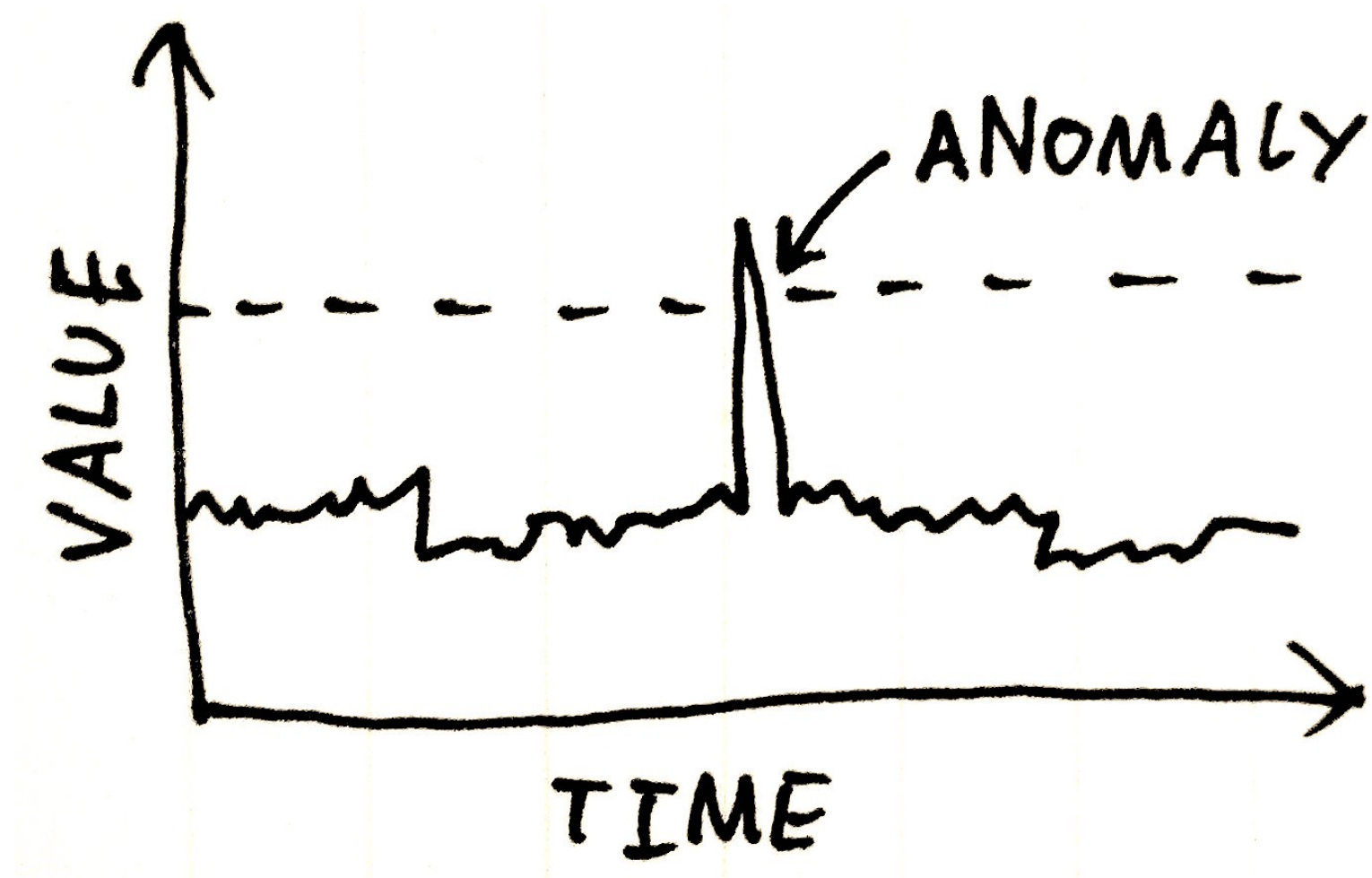
Approaches: Rate of change
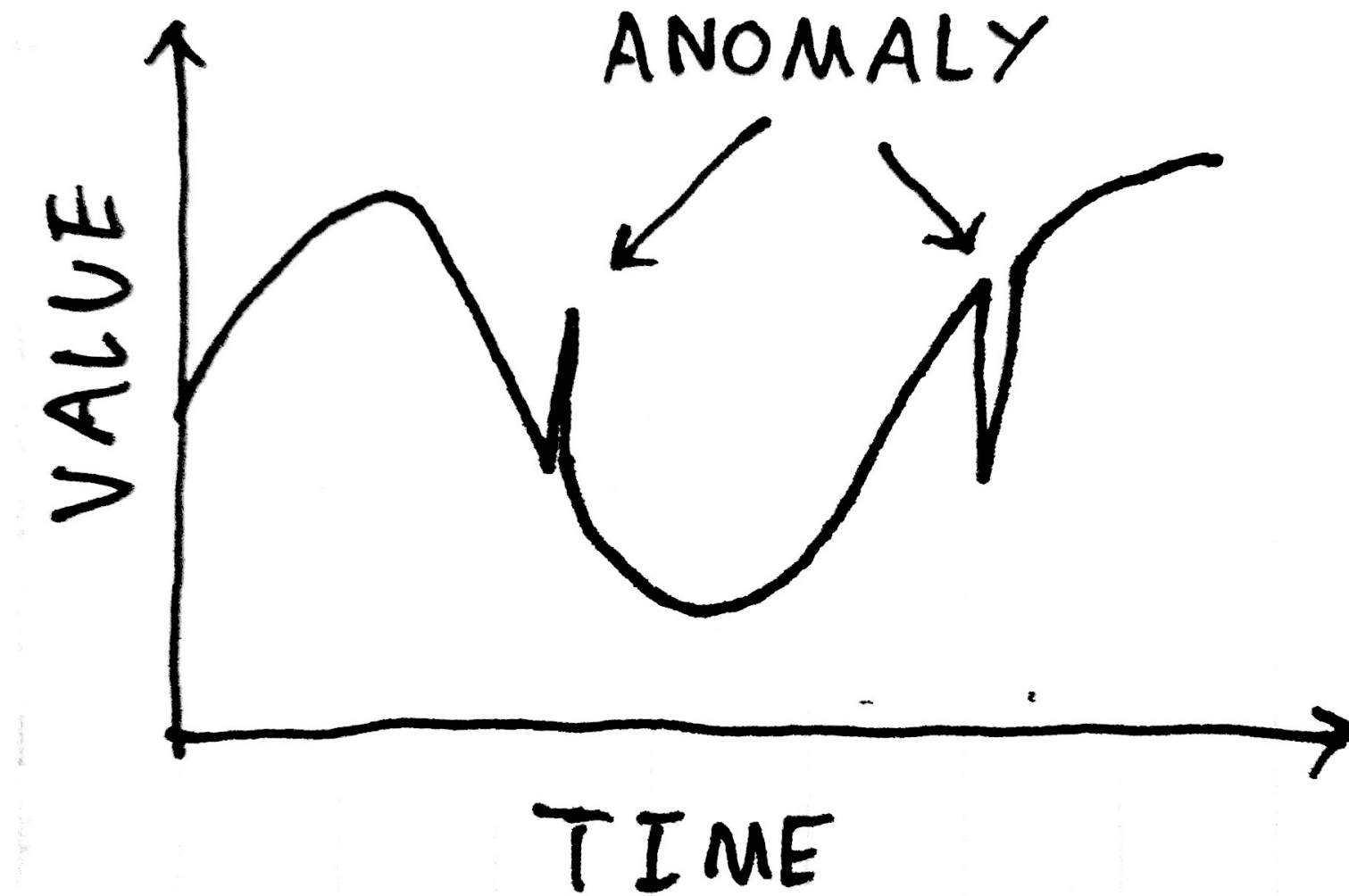
Approaches: Shape monitoring
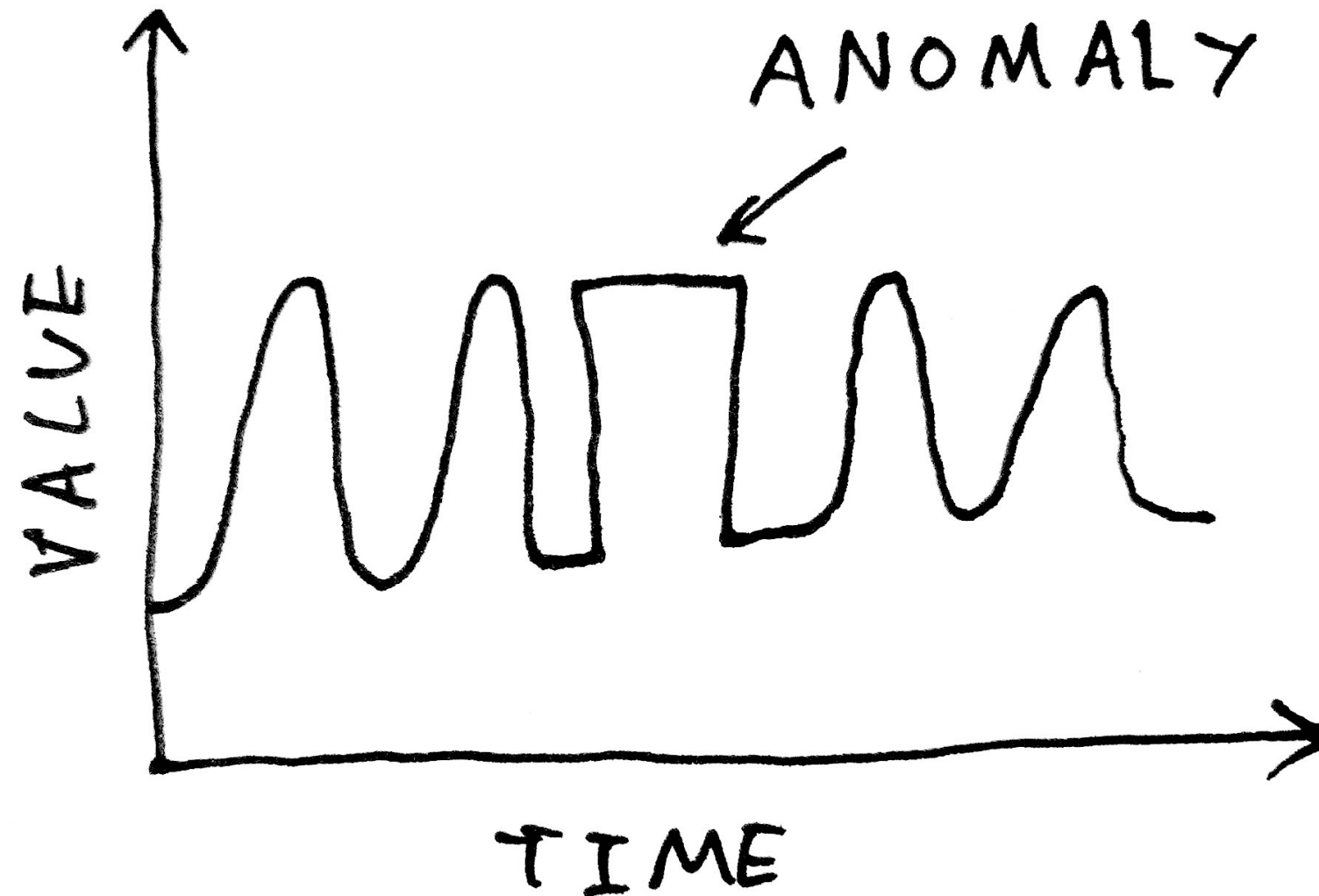
Algorithms

Robust covariance (assumes normal distribution)

```
from sklearn.covariance import EllipticEnvelope
```

Isolation Forest (powerful, but more computationally demanding)

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

One-Class SVM (sensitive to outliers, many false negatives)

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

| | Robust covariance | One-Class SVM | Isolation Forest |
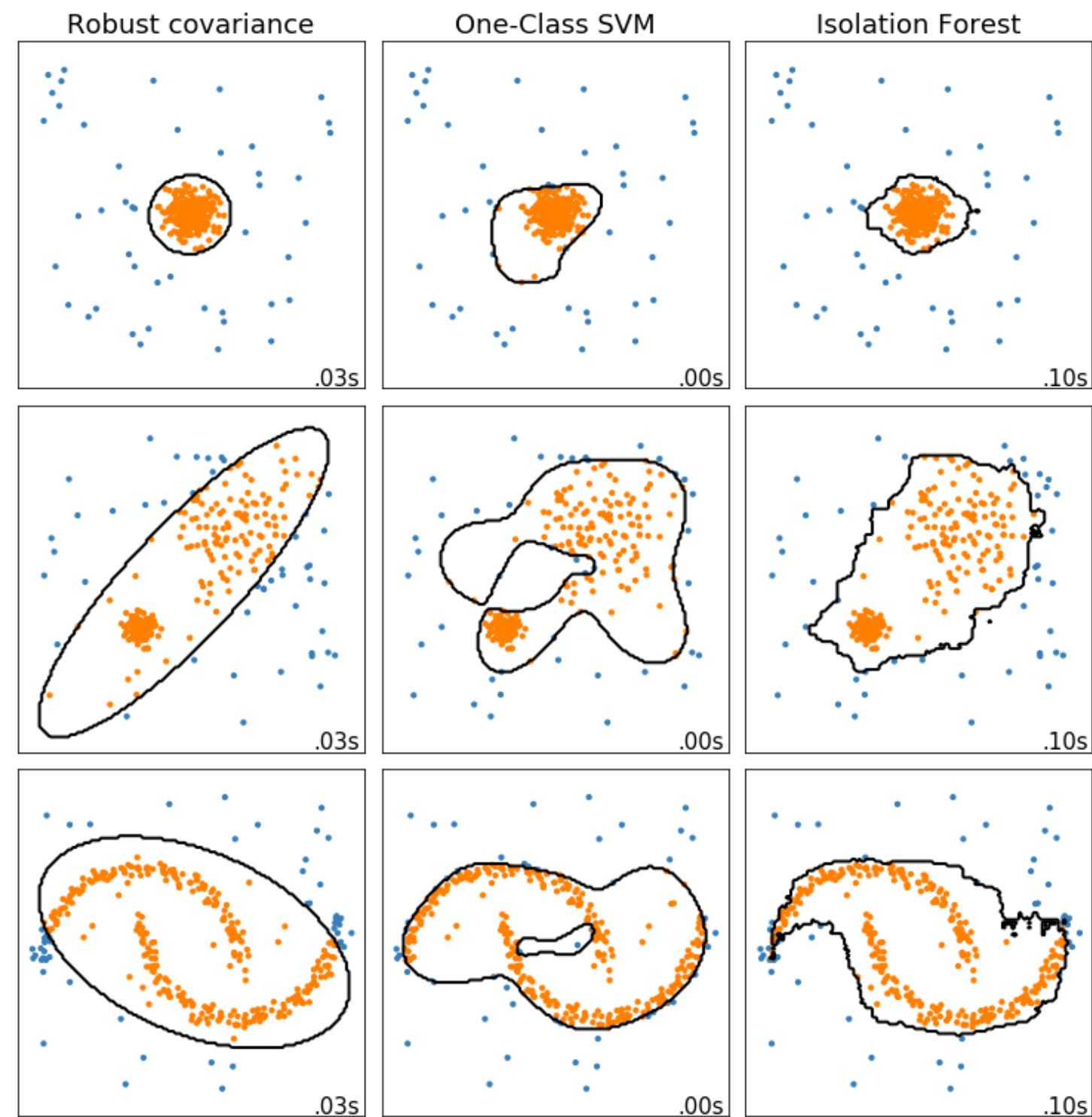
Training and testing

Example: Isolation Forest

```python
from sklearn.ensemble import IsolationForest


algorithm = IsolationForest()


# Fit the model
algorithm.fit(X)


# Apply the model and detect the outliers
results = algorithm.predict(X)
```

Evaluation

```python
from sklearn.metrics \
import (confusion_matrix,
        precision_score,
        recall_score)


confusion_matrix(y_true, y_predicted)
```

Precision = How many of the anomalies I have detected are TRUE anomalies?

Recall = How many of the TRUE anomalies I have managed to detect?

Example: Arrhythmia detection

Want to learn more?

AI FUNDAMENTALS

Selecting the right model

AI FUNDAMENTALS

Nemanja Radojkovic
Senior Data Scientist

Model-to-problem fit

Type of Learning

- **Target variable defined & known? => Supervised.**
  - Classification?

  - Regression

- **No target variable, exploration? => Unsupervised.**
  - Dimensionality Reduction?

  - Clustering?

  - Anomaly Detection?

Defining the priorities

Interpretable models

- Linear regression (Linear, Logistic, Lasso, Ridge)

- Decision Trees

Well performing models

- Tree ensembles (Random Forests, Gradient Boosted Trees)

- Support Vector Machines

- Artificial Neural Networks

Simplicity first!

Using multiple metrics

Satisfying metrics

- Cut-off criteria that every candidate model needs to meet.

- Multiple satisfying metrics possible (e.g. minimum accuracy, maximum execution time, etc)

Optimizing metrics

- Illustrates the ultimate business priority (e.g. "minimize false positives", "maximize recall")

- "There can be only one"

Final model:

- Passes the bar on all satisfying metrics and has the best score on the optimization metric.

Interpretation

Global

- *"What are the general decision-making rules of this model?"*

- Common approaches:
  - Decision tree visualization

  - Feature importance plot

Local

- *"Why was this specific example classified in this way?"*

- LIME algorithm (Local Interpretable Model-Agnostic Explanations)

Model selection and interpretation

AI FUNDAMENTALS