MACHINE LEARNING WITH PYTHON FOR SPACE WEATHER APPLICATIONS

by ITU Upper Atmosphere and Space Weather Laboratory

Lecture 3: Classification and Clustering Techniques

Instructor: Dogacan Su Ozturk

Contact: dsozturk@alaska.edu

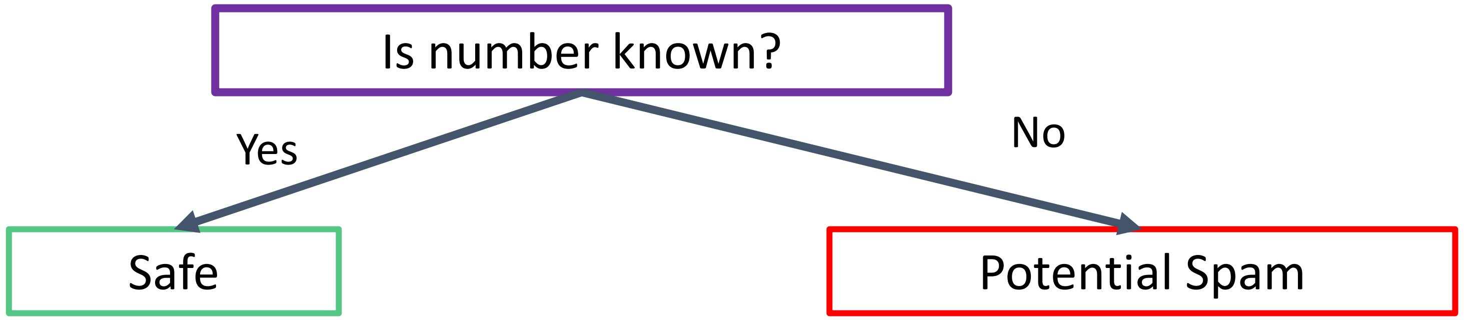
11-12 May 2022 at Upper Atmosphere and Space Weather Laboratory

What is classification?

- Classification is one of the most common supervised learning techniques.
- It can be simplified as “predicting classes/categories”.
- A classification application consists of the following:
    - Classifier: Algorithm chosen for the task
    - Classification model: The model that predicts the class
    - Feature: Descriptors of the data set that leads to distinct classes
    - Binary classification: Classification task with two outcomes
    - Multi-class Classification: Each sample belongs to only one class
    - Multi-label Classification: A sample can be assigned to set of classes
    - Target: The class
    - Evaluation: Evaluation of model’s prediction capability
- When you are training your machine learning algorithms you have to pick whether problem at hand is a classification or regression problem. It is not easy as it sounds.

Binary Classification

- Classification with only 2 classes is called binary classification.
- Can you think of some examples?
- What have we learnt so far to help with Binary Classification?



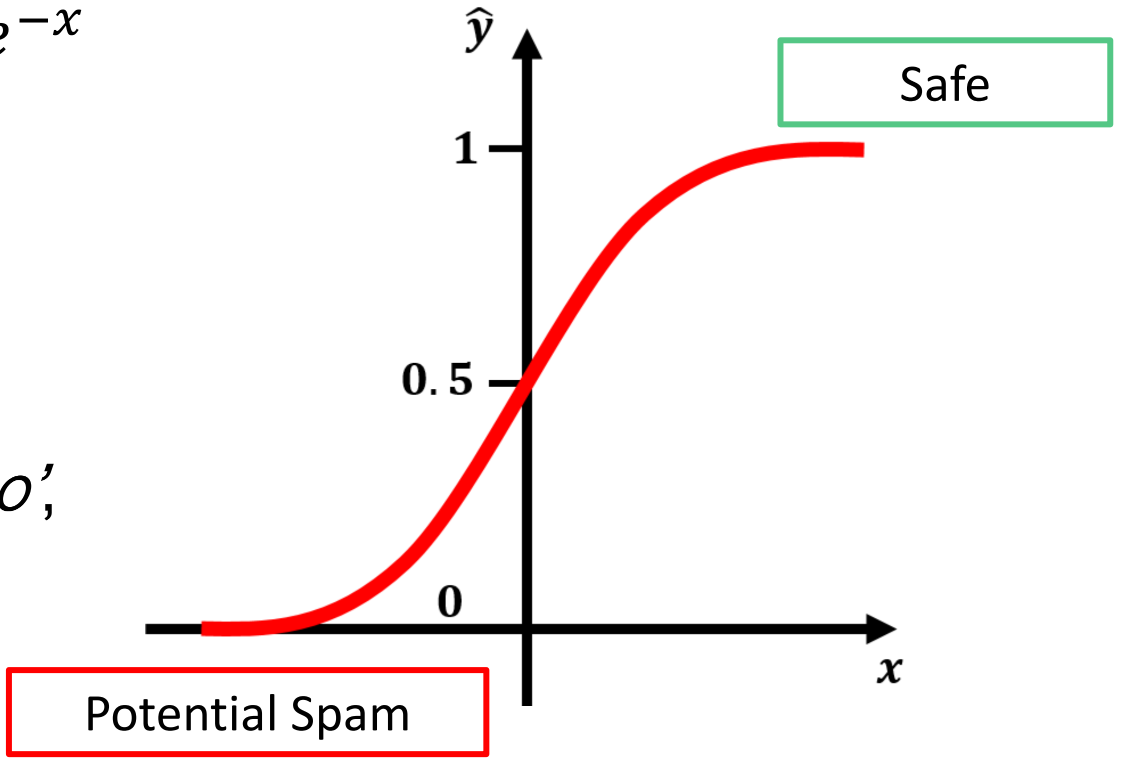
1. Methods: Logistic Regression

Logistic regression is commonly used to provide a probability (pass/fail, win/lose). It is the binary classification analog of linear regression.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

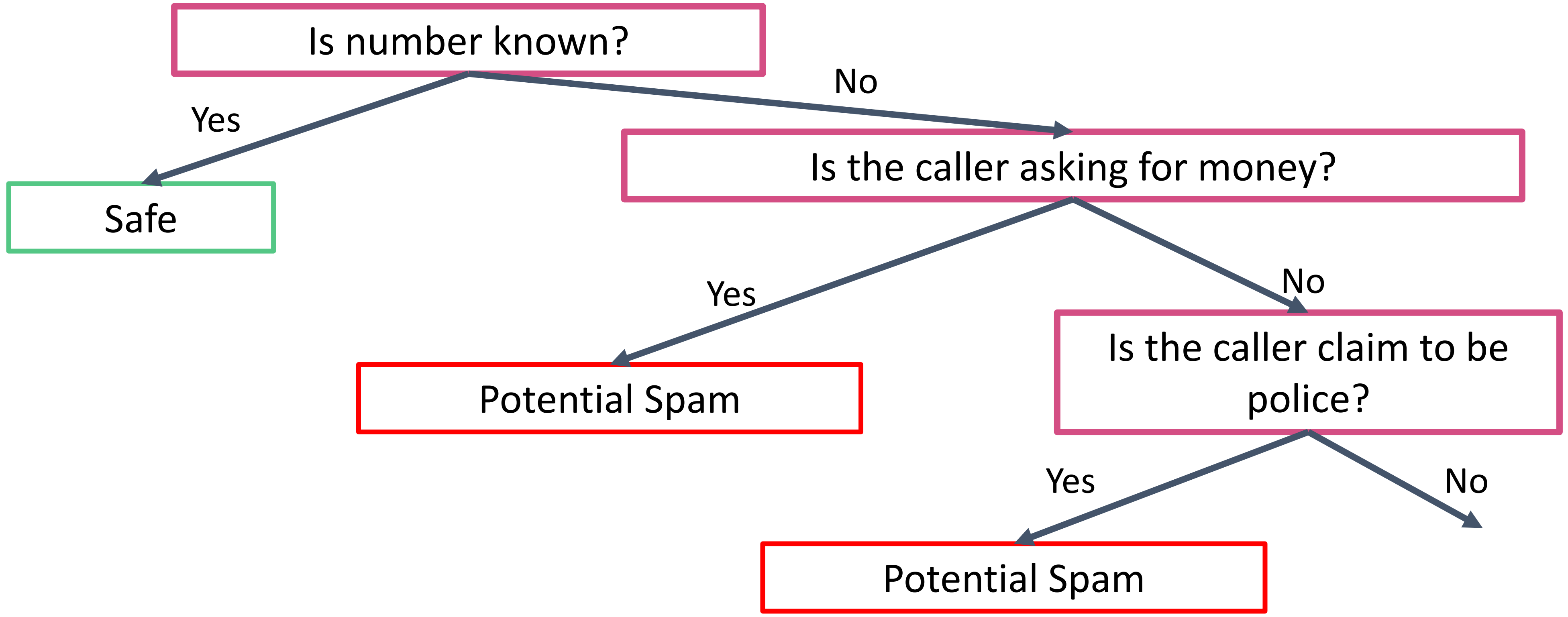
```
from sklearn.linear_model import LogisticRegression
```

class sklearn.linear_model.LogisticRegression(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*)



1. Methods: Decision Trees and Random Forest

We have learnt about using decision trees and random forests for linear regression. They can also be used for classification.


Is number known?
Yes
No
Safe
Is the caller asking for money?
Yes
No
Potential Spam
Is the caller claim to be police?
Yes
No
Potential Spam

1. Methods: Decision Trees and Random Forest

We have learnt about using decision trees and random forests for linear regression. They can also be used for classification.

- from sklearn.tree import DecisionTreeClassifier

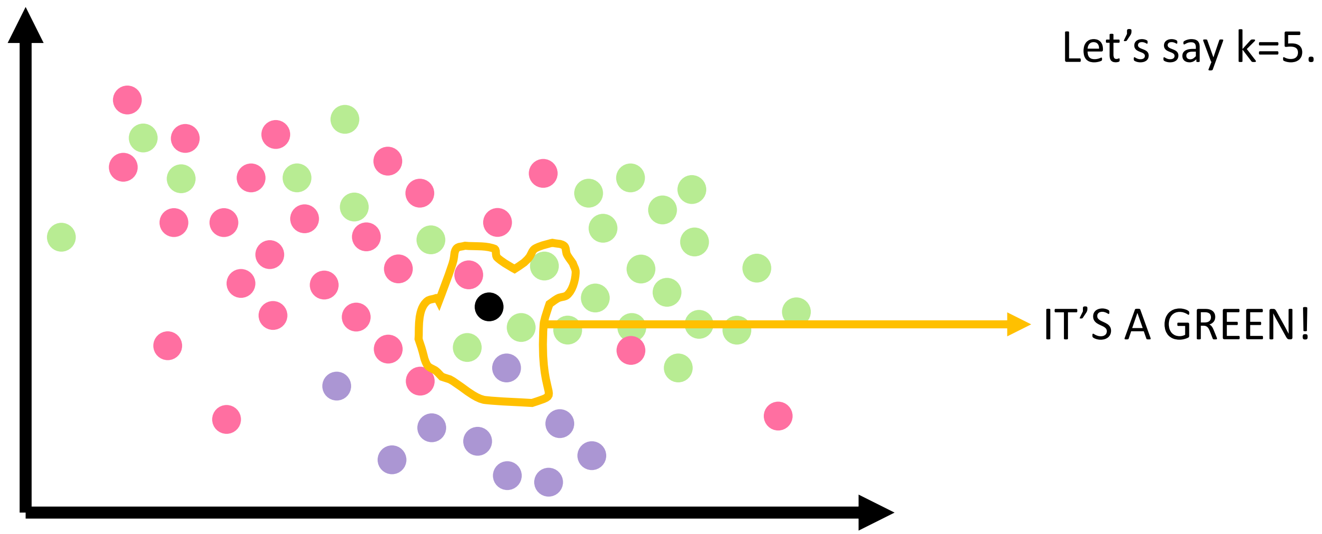
  class sklearn.tree.DecisionTreeClassifier(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0*)

- from sklearn.ensemble import RandomForestClassifier

  class sklearn.ensemble.RandomForestClassifier(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*)

1. Methods: k-Nearest Neighbours

- Depending on the k number of neighbouring point labels from training data, kNN algorithms predicts a label for the point.



- from sklearn.neighbors import KNeighborsClassifier

  class sklearn.neighbors.KNeighborsClassifier(*n_neighbors=5, *, weights='uniform'*, *algorithm='auto'*, *leaf_size=30, p=2, metric='minkowski'*, *metric_params=None, n_jobs=None, **kwargs*)

Evaluation of Classification

- How do we evaluate classification? How is it different than Regression?

2. Evaluation: Accuracy Score

Accuracy score is the Number of Correct Predictions/Total Number of Predictions.

```
from sklearn.metrics import accuracy_score
```

sklearn.metrics.accuracy_score(*y_true, y_pred, *, normalize=True, sample_weight=None*)

2. Evaluation: Confusion Matrix

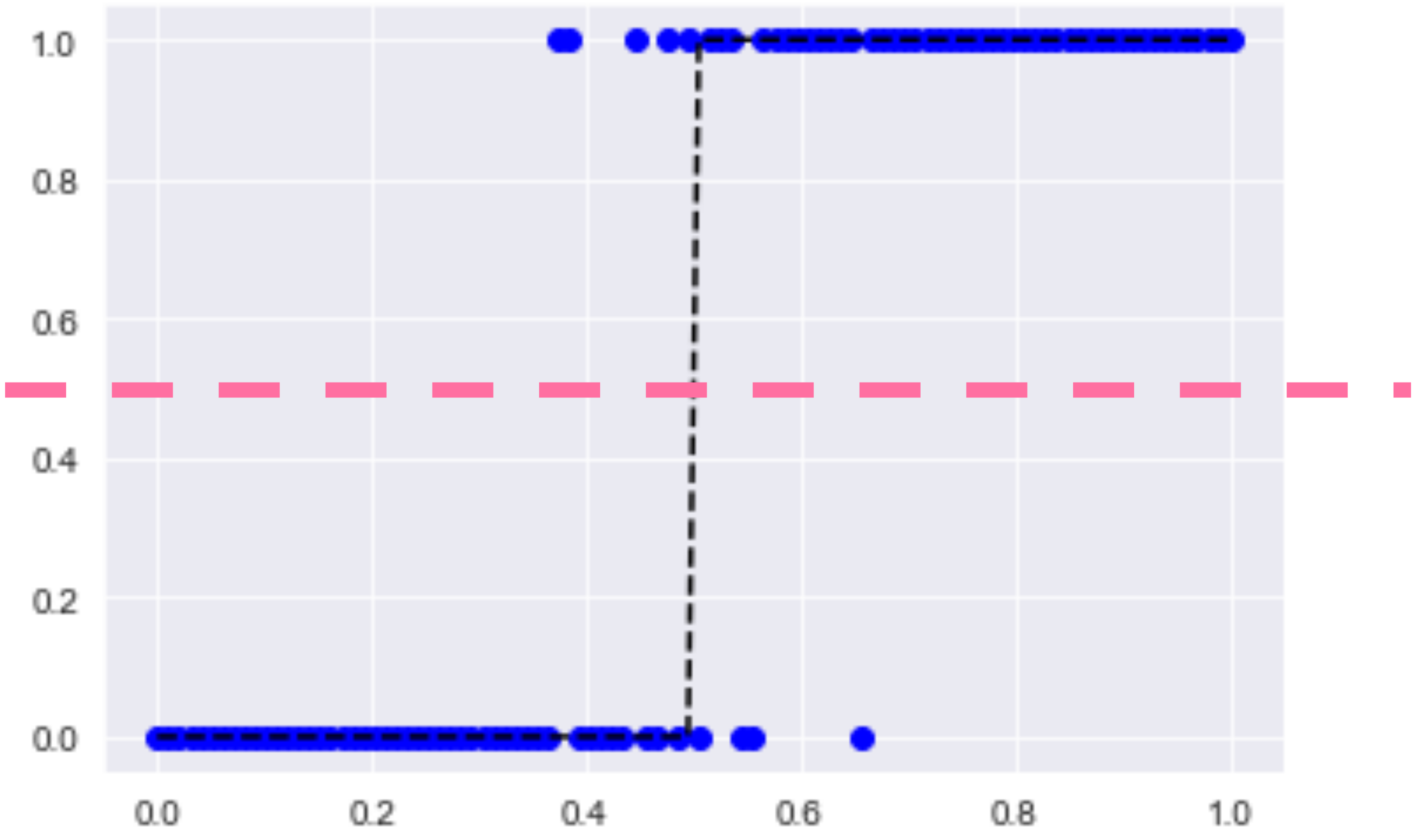
- There are four possible outcome of predicting classes.
  - True positive - Predict 1 when the actual class is 1.
  - False positive - Predict 1 when the actual class is 0.
  - True negative - Predict 0 when the actual class is 0.
  - False negative - Predict 0 when the actual class is 1.
- Accuracy metric alone can not account for the false positive and negative.

TN	FP
FN	TP

- from sklearn.metrics import confusion_matrix

  sklearn.metrics.confusion_matrix(*y_true*, *y_pred*, ***, *labels=None*, *sample_weight=None*, *normalize=None*)

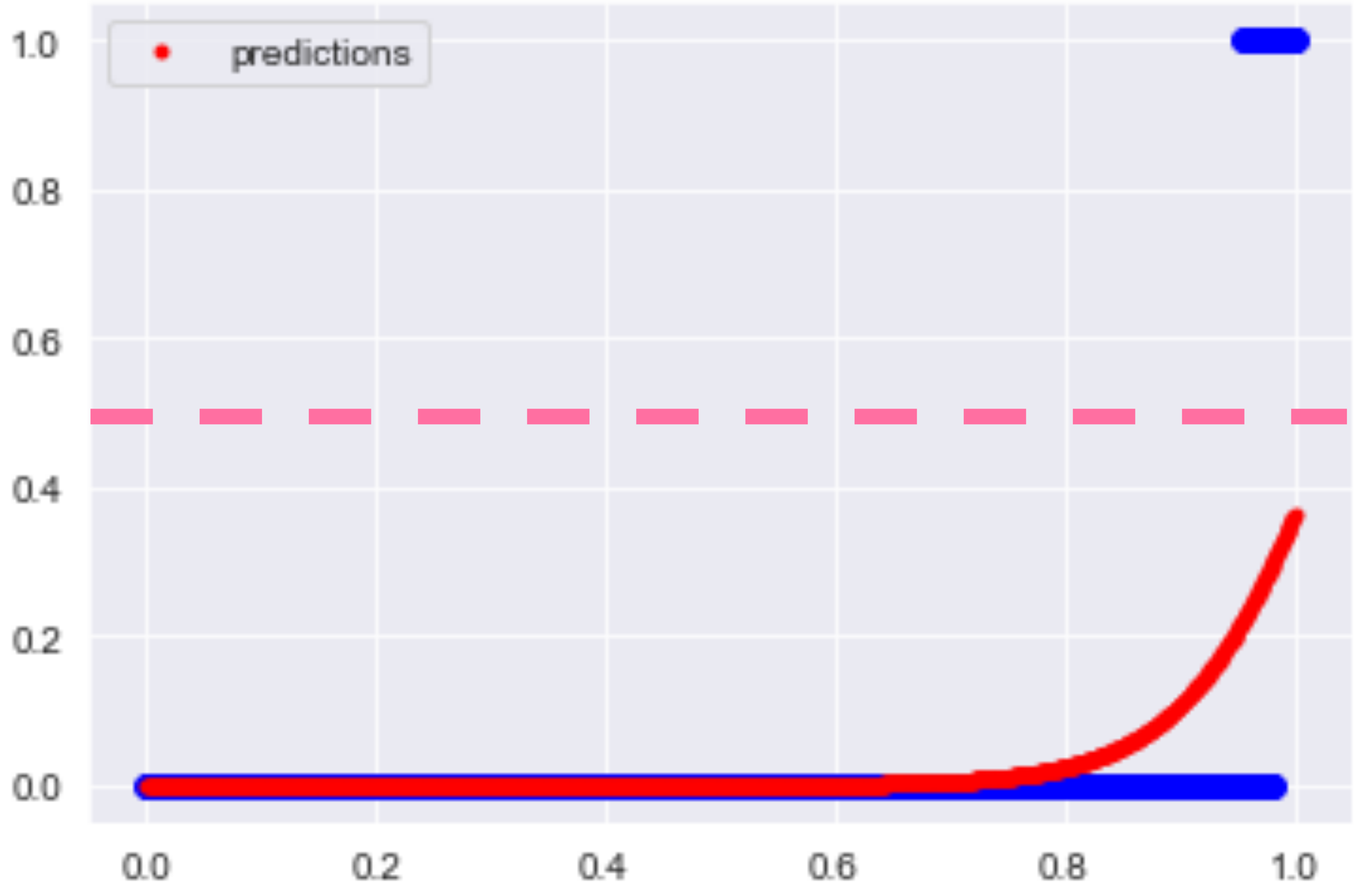
2. Evaluation: Prediction Threshold



Threshold is 0.5 by default in binary classification and also across sklearn implementations.

But what happens when we have an imbalanced class?

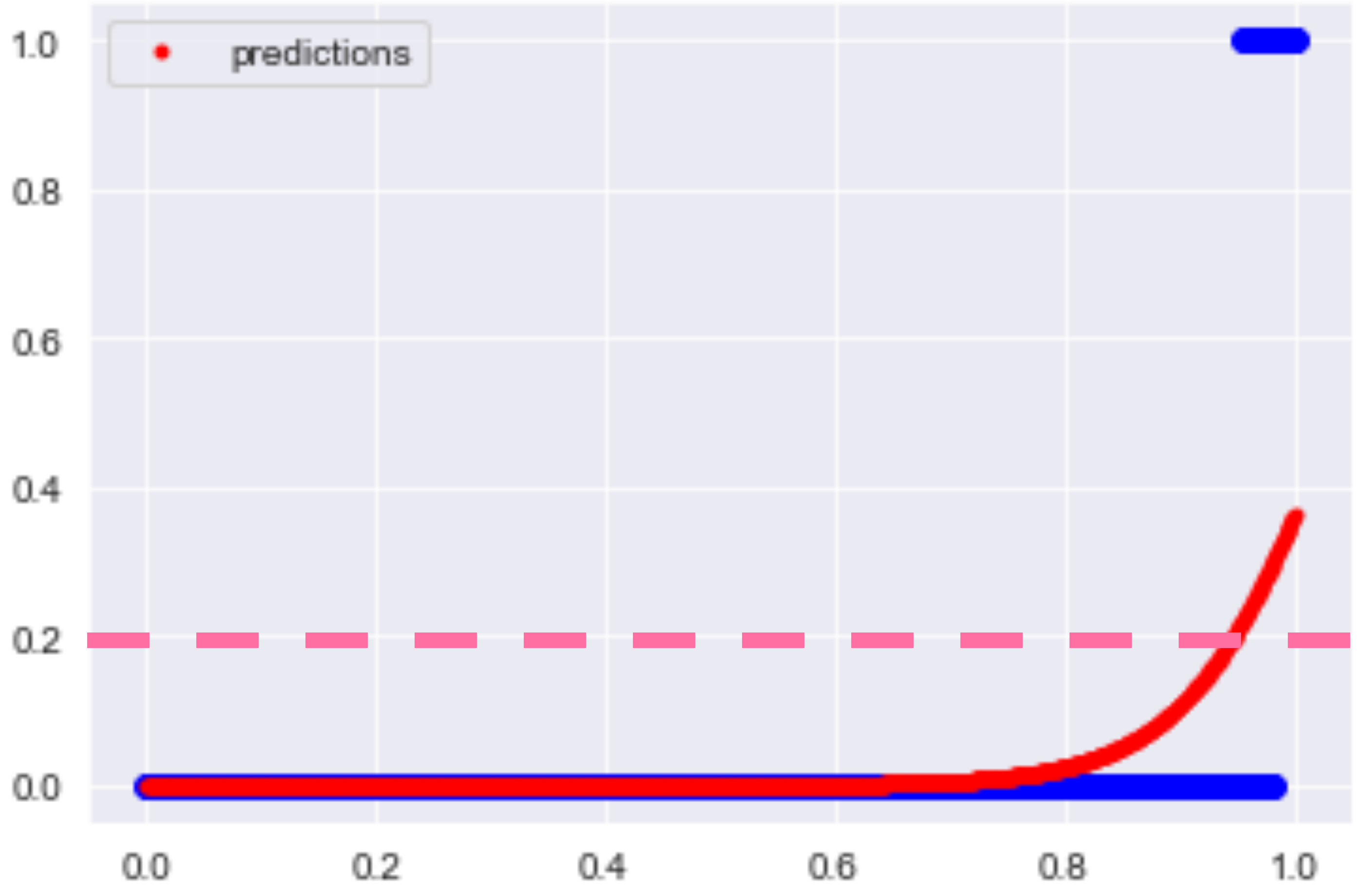
2. Evaluation: Prediction Threshold



- What if we have an unevenly distributed classification problem?
- How will the threshold affect the results?

95	**0**
5	0

2. Evaluation: Prediction Threshold



- What if we have an unevenly distributed classification problem?
- How will the threshold affect the results?

95	**0**
1	4

2. Evaluation: Precision-recall

- Precision is defined as TP/(TP+FP).
- Recall is defined as TP/(TP+FN).
- Precision-recall is a measure of the “success” of prediction. Here, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned.

from sklearn.metrics import precision_recall_curve

sklearn.metrics.precision_recall_curve(*y_true*, *probas_pred*, ***, *pos_label=None*, *sample_weight=None*)

from sklearn.metrics import plot_precision_recall_curve

from sklearn.metrics import PrecisionRecallDisplay

class sklearn.metrics.PrecisionRecallDisplay(precision, recall, *, average_precision=None, estimator_name=None, pos_label=None)[source]

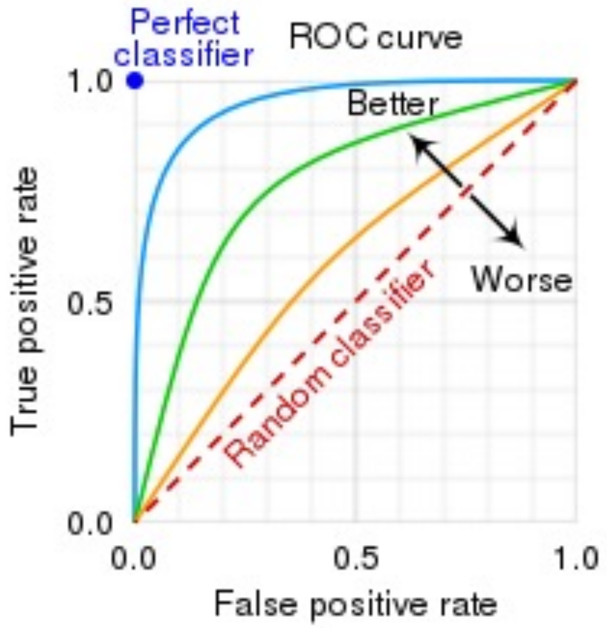
2. Evaluation: ROC Curve

- Receiver Operating Characteristic Curve is the graph that shows the performance of a binary classifier. TP vs FP curve at various threshold settings.
- AUROC: area under ROC curve is a reliable metric for binary classification. It shows the probability that a random positive class observation ranks higher than a random negative class observation.

```
from sklearn.metrics import roc_curve, roc_auc_score
```

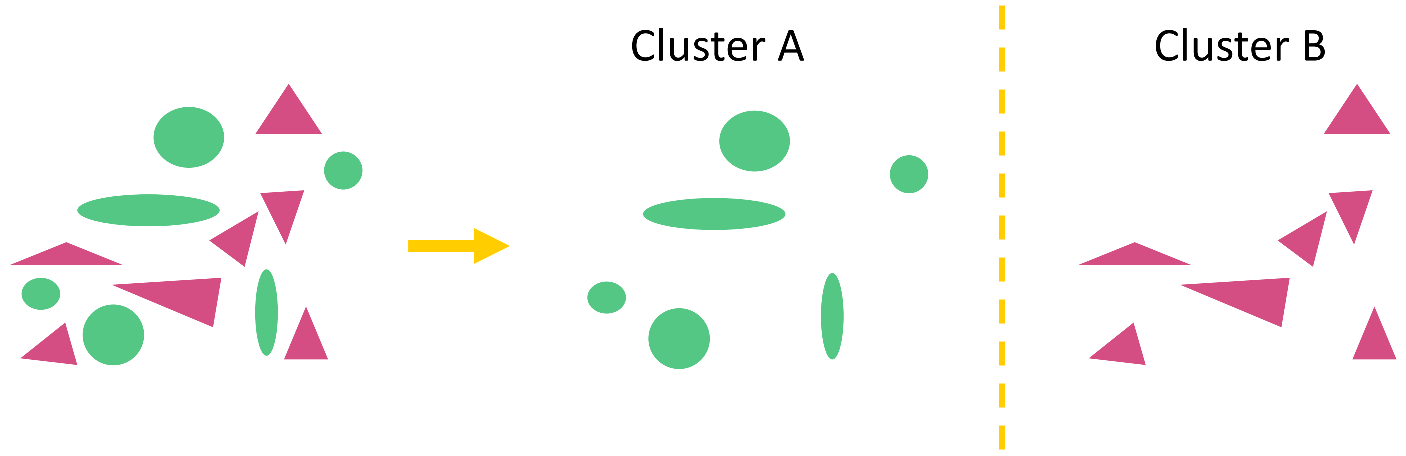
sklearn.metrics.roc_curve(*y_true, y_score, *, pos_label=None, sample_weight=None, drop_intermediate=True*)

sklearn.metrics.roc_auc_score(*y_true, y_score, *, average='macro', sample_weight=None, max_fpr=None, multi_class='raise', labels=None*)

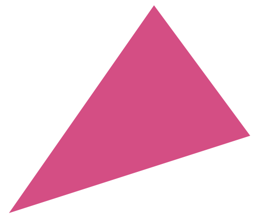


What is clustering?

Clustering is the identifying of similar instances and assigning of samples to different clusters.



And deciding where the next sample goes.

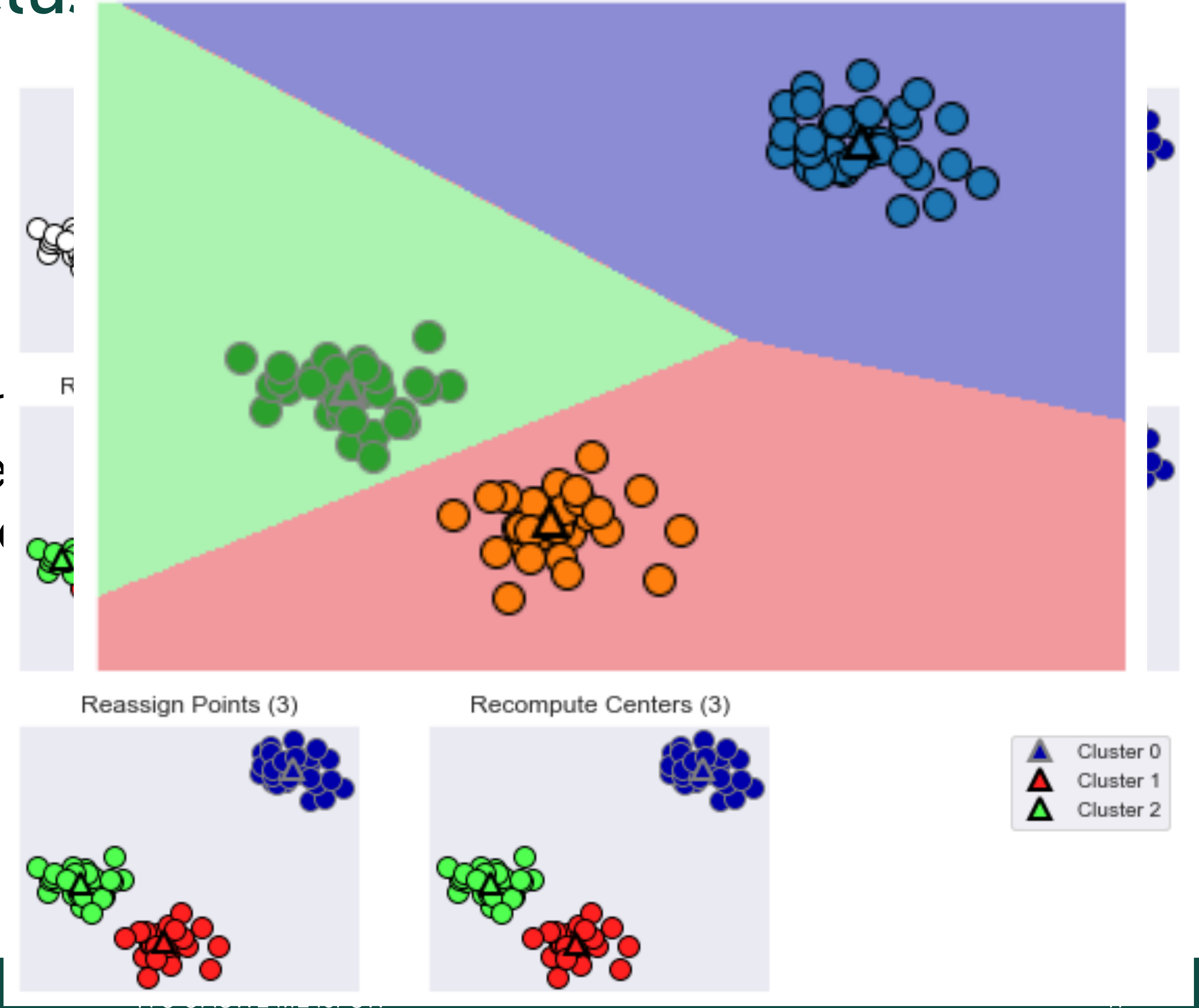


Clustering can be used for:

- Segmentation
- Data exploration
- Dimensionality reduction
- Anomaly detection
- Semi-supervised learning

1. Methods: KMeans Clus

- As we have learnt in Classification, k-means is a simple way of clustering samples.
- In clustering, k-means assigr a sample to a cluster and the recalculates the cluster cent in each step.

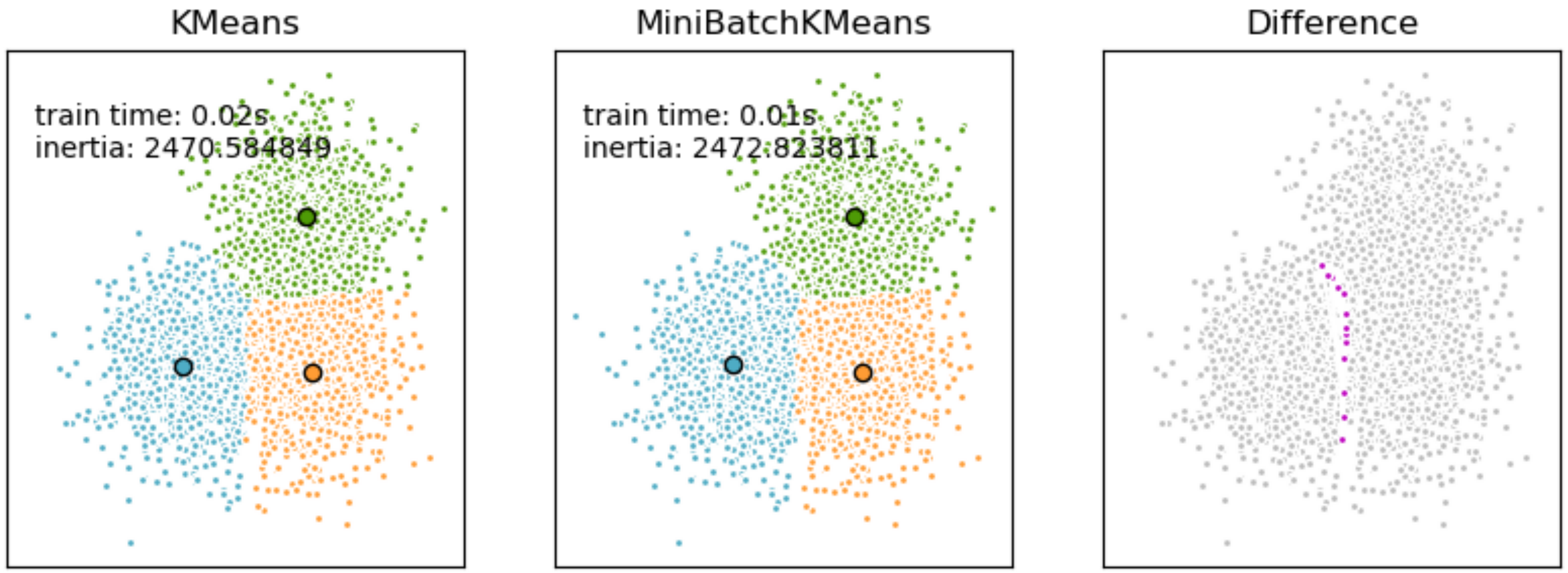


1. Methods: KMeans Clustering

- As we have learnt in Classification, k-means is a simple way of clustering samples.
- from sklearn.cluster import KMeans
  - *class* sklearn.cluster.KMeans(*n_clusters=8, *, init='k-means++', n_init=10, max_iter=300, tol=0.0001, precompute_distances='deprecated', verbose=0, random_state=None, copy_x=True, n_jobs='deprecated', algorithm='auto'*)
  - kmeans.cluster_centers_
  - kmeans.labels_
  - kmeans.intertia_
  - kmeans.n_iter_

1. Methods: Mini Batch KMeans Clustering

- Almost like an ensemble model, Mini Batch K-means works with a small sample size to reduce computational time.
- from sklearn.cluster import MiniBatchKMeans
  - *class sklearn.cluster.MiniBatchKMeans(n_clusters=8, *, init='k-means++', max_iter=100, batch_size=1024, verbose=0, compute_labels=True, random_state=None, tol=0.0, max_no_improvement=10, init_size=None, n_init=3, reassignment_ratio=0.01)*
  - mini_kmeans.cluster_centers_
  - mini_ kmeans.labels_
  - mini_ kmeans.intertia_
  - mini_ kmeans.n_iter_



Evaluation of Clustering: Inertia

- KMeans clusters samples by n groups of equal variance, trying to minimize a value called inertia.

$$\sum_{i=0}^{n} min\left(\left\|x_i - \mu_j\right\|^2\right)$$

x_i: sample location

μ_j: mean of the samples in cluster (center/centroids)

- Inertia responds poorly to elongated clusters.
- Lower values are better and zero is optimal.

Evaluation of Clustering: Silhouette Score

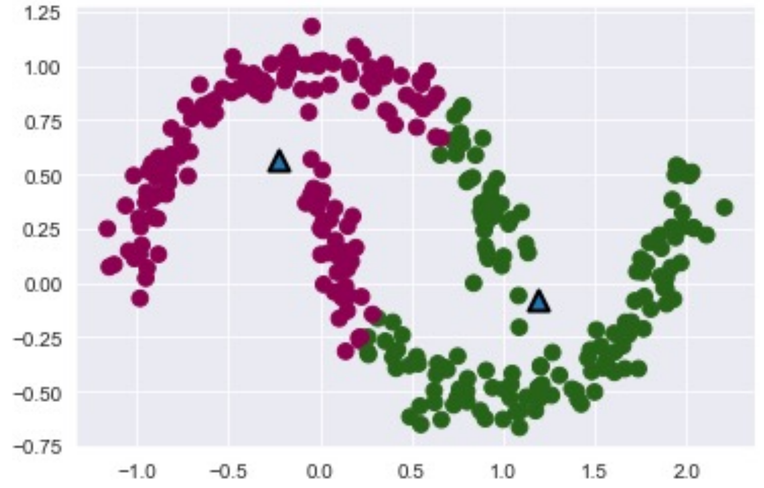
$$S = \frac{(b - a)}{\max(b - a)}$$

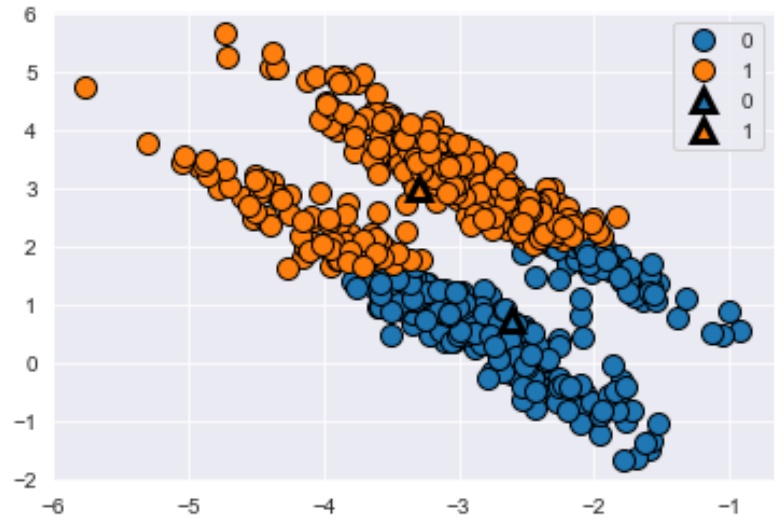
a: average distance between one data point and all other points in the same cluster.
b: average distance between one data point and all other points in the nearest cluster.

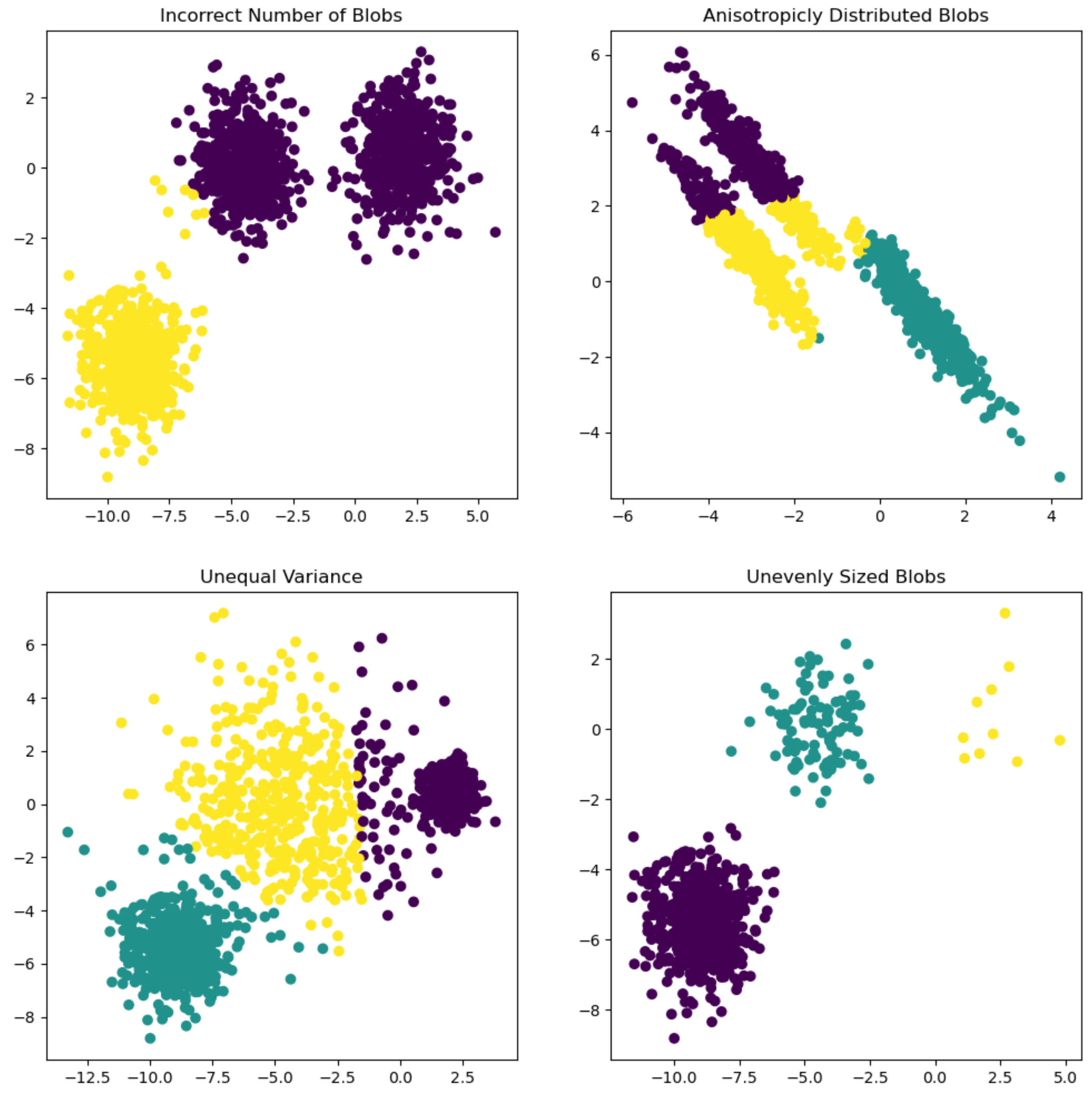
- Silhoutte Coefficient is bounded between -1 and 1, the first being the worst and the latter being the best case scenario.
- A value of 0 indicates overlap between different clusters.

KMeans Failures

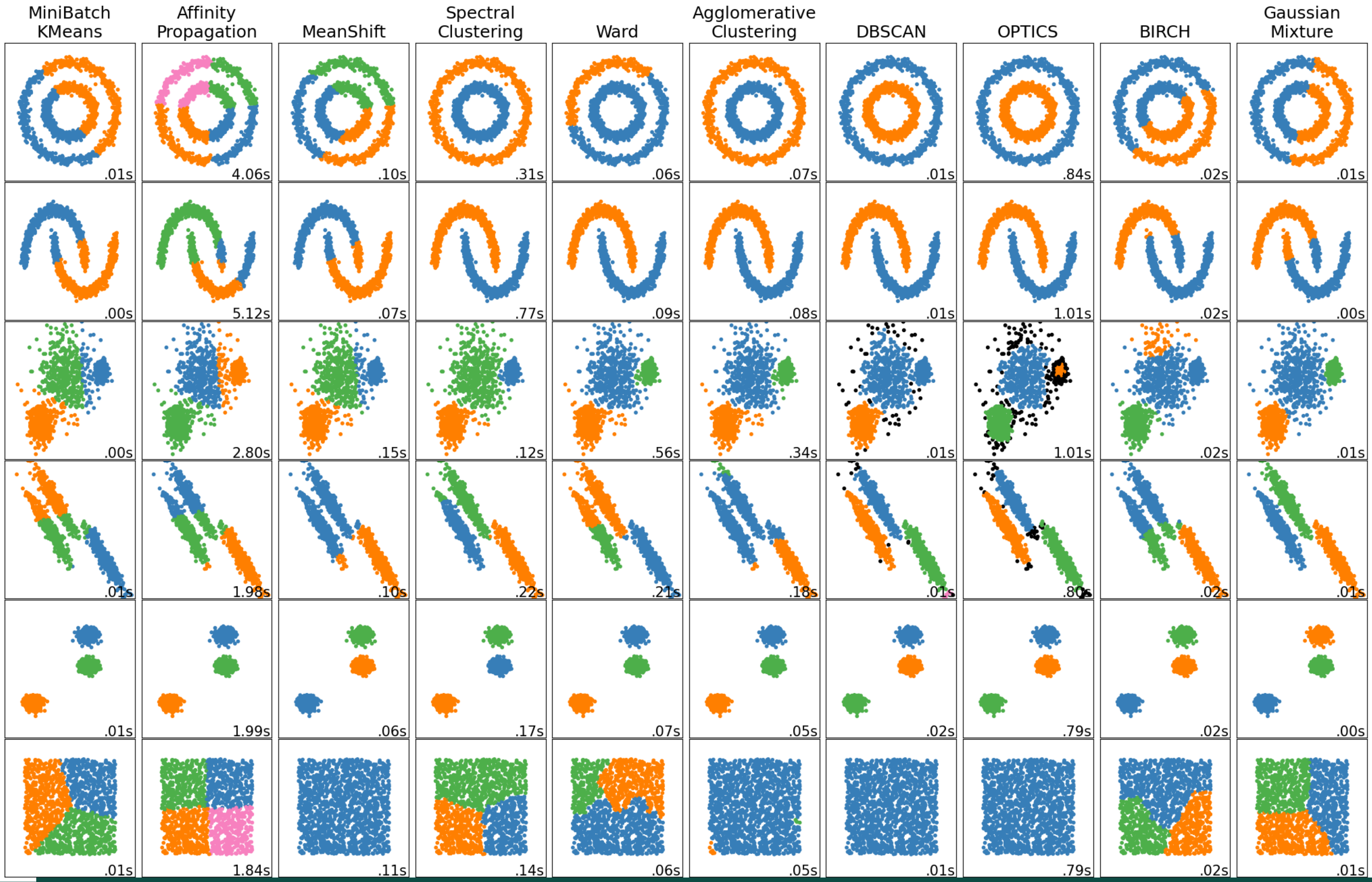
When does KMeans fail?







Comparison of Clustering Algorithm Performance



There seems to be two emerging behaviour between different clustering algorithms.

1. Methods: DBSCAN: Density Based Spatial Clustering of Applications with Noise

- The main benefit of this algorithm is that a priori knowledge of clusters is not needed.
- DBSCAN works by identifying points in ”crowded” regions.
- Points inside these dense regions in feature space are named as core samples.
- DBSCAN works with 2 samples: min_samples and eps. If an instance has at least min_samples in its ε neighborhood it is a core sample.
- from sklearn.cluster import DBSCAN

  class sklearn.cluster.DBSCAN(*eps=0.5, *, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*)
- Once DBSCAN is fitted, it can be used to train a KNN too.

Supervised vs Unsupervised Learning

Parameter	Supervised Learning	Unsupervised Learning
Feature selection	Features and targets	Entire data set
Learning from	Labelled data	Entire data set
ML algorithms	Linear and logistic regressions, decision trees, classification, neural networks, etc.	K-means, DBSCAN, etc.
Splitting	Training (+validation) + test set	Entire data set
Accuracy	Controlled through metrics	Trial and error
Result	Number of classes known	Number of classes unknown
Drawback	Creating classes/labels difficult	Getting precise information is difficult

Semi-supervised learning: Train with unlabeled data in conjunction with a small set of labeled data.