Practical Examples: Supervised & Unsupervised Learning, Dimensionality Reduction

Lida Kanari

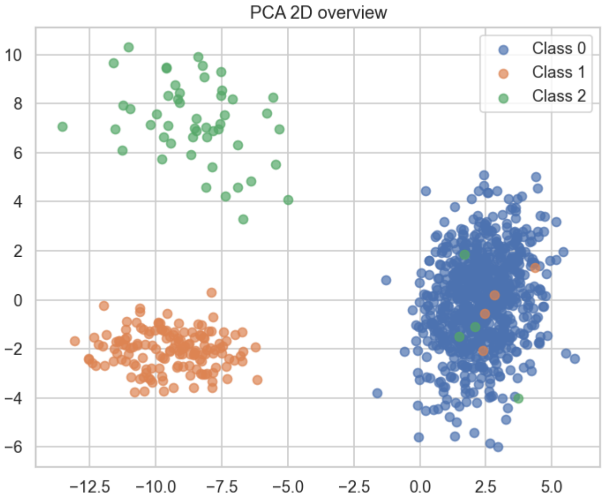
Mathematical Institute
University of Oxford

Introduction to Machine Learning, November 2025

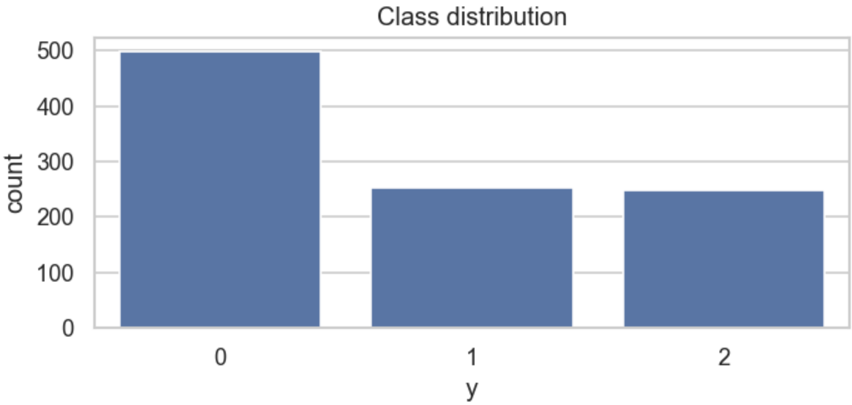
UNIVERSITY OF OXFORD
Mathematical Institute

Table of Contents

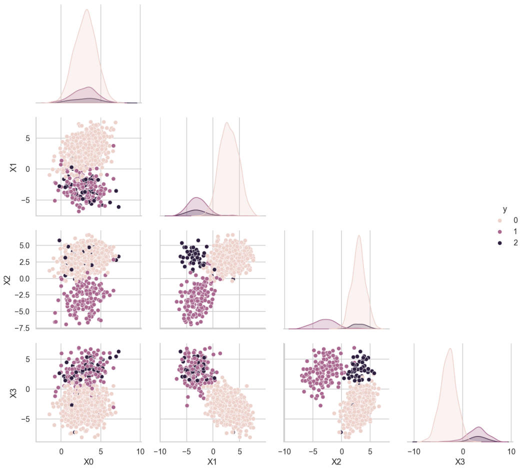
Dataset A


PCA 2D overview
Class 0
Class 1
Class 2
10
8
6
4
2
0
−2
−4
−6
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0

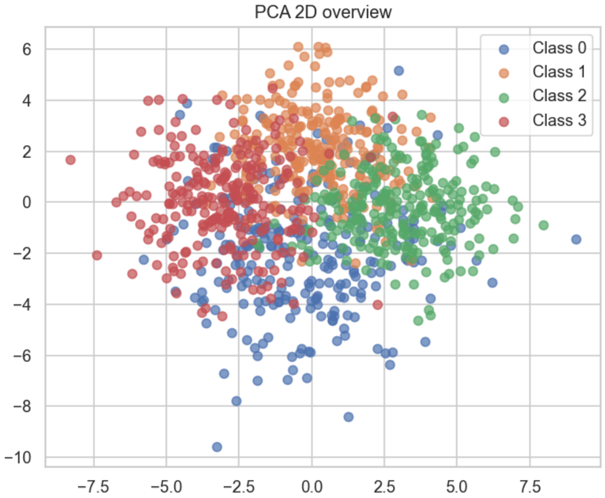
Dataset A



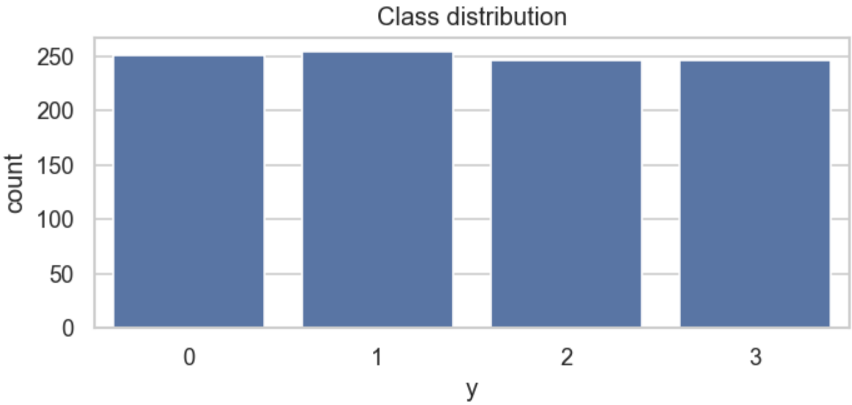
Dataset A



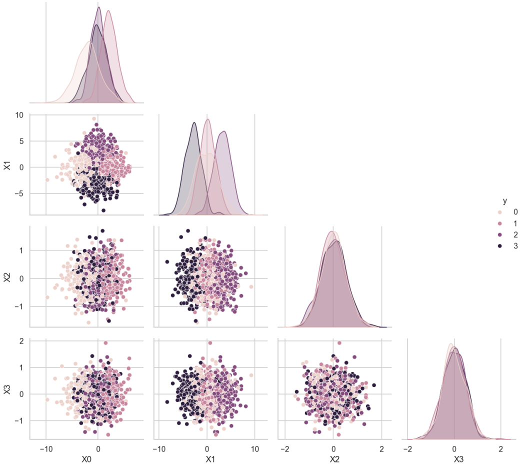
Dataset B


PCA 2D overview
Class 0
Class 1
Class 2
Class 3
6
4
2
0
−2
−4
−6
−8
−10
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

Dataset B

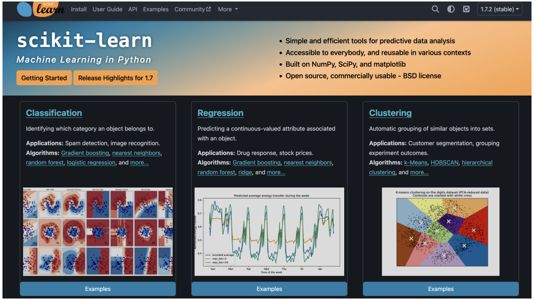


Dataset B



Scikit - Learn

https://scikit-learn.org/stable/index.html



Packages you will need for the practical lecture

- numpy
- pandas
- seaborn
- sklearn
- matplotlib
- umap

Dataset Generation

Dataset A: simple three class dataset

3]:

```
# Generate dataset
n_samples = 1000
n_features = 10
weights = [0.8, 0.15, 0.05]

X, y = make_classification(n_samples=n_samples,
                           n_features=n_features,
                           n_informative=8,
                           n_redundant=0,
                           n_repeated=0,
                           n_classes=3,
                           n_clusters_per_class=1,
                           weights=weights,
                           class_sep=3.0,
                           flip_y=0.01,
                           random_state=RND)

feature_names = [f'X{i}' for i in range(X.shape[1])]
df = pd.DataFrame(X, columns=feature_names)
df['y'] = y
```

Part 3: Dimensionality Reduction Algorithms

Part 3: Dimensionality Reduction Algorithms

Oxford Mathematical Institute

Summary Table

Algorithm	Type	Linear / Non-linear	Preserves Local Structure	Preserves Global Structure	Key Parameters	Main Use Case	Link
PCA	Principal components	✅ Linear	⚪ Partial	✅ Strong	`n_components`	Noise reduction, variance analysis	Docs
ICA	Independent components	✅ Linear	⚪ Partial	⚪ Moderate	`n_components`	Source separation, signal analysis	Docs
t-SNE	Probabilistic	❌ Non-linear	✅ Excellent	⚪ Weak	`n_components`, `perplexity`, `learning_rate`	Visualizing high-dimensional clusters	Docs
UMAP	Graph-based	❌ Non-linear	✅ Excellent	✅ Good	`n_neighbors`, `min_dist`, `n_components`	Fast, interpretable 2D embeddings	Docs
Isomap	Geodesic	❌ Non-linear	✅ Good	✅ Good	`n_neighbors`, `n_components`, `metric`	Unfolding curved manifolds	Docs
MDS	Distance-based	❌ Non-linear	⚪ Partial	✅ Strong	`n_components`, `metric`, `max_iter`, `dissimilarity`	Preserving pairwise distances	Docs

Mathematical Institute

3.1 PCA (Principal Component Analysis)

scikit-learn docs: https://scikit-learn.org/stable/modules/decomposition.html#pca

[153]:
```
pca2 = PCA(n_components=2, random_state=RND).fit(X)
X_pca2 = pca2.transform(X)
explained = pca2.explained_variance_ratio_
```

Principal Component Analysis (PCA)

Type: Linear Projection
Concept: Finds orthogonal directions (principal components) that capture the maximum variance in the data.
Strengths:

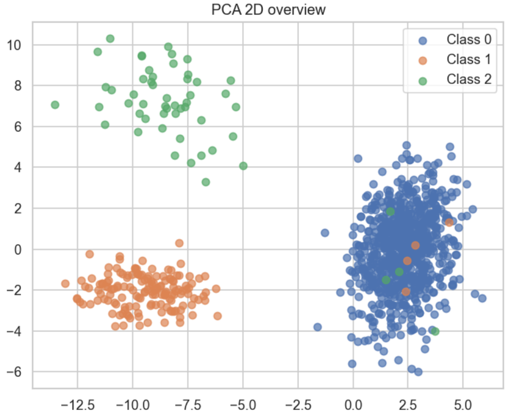
- Fast, interpretable, widely used.
- Provides insight into feature variance and structure.

Limitations:

- Captures only linear relationships.
- Sensitive to feature scaling.

Key Parameters: `n_components`, `svd_solver`, `whiten`.
Reference: scikit-learn – PCA

Principal Component Analysis (PCA)



Independent Component Analysis (ICA)

Mathematical Institute

Type: Linear (Statistical Independence)
Concept: Decomposes data into statistically independent components.
Strengths:

- Identifies independent hidden factors.

Limitations:

- Sensitive to noise and scaling.
- Results depend on initialization.

Key Parameters: `n_components`, `max_iter`, `tol`.
Reference: scikit-learn – ICA

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Oxford Mathematical Institute

Type: Non-linear / Probabilistic Manifold Learning
Concept: Preserves local structure by modeling pairwise similarities in high- and low-dimensional space.
Strengths:

- Good for visualizing non-linear clusters.

Limitations:

- Computationally heavy.
- Non-deterministic results.

Key Parameters: `n_components`, `perplexity`.
Reference: scikit-learn – tSNE

Uniform Manifold Approximation and Projection (UMAP)

Type: Non-linear / Graph-based
Concept: Builds a weighted graph and learns a low-dimensional embedding that tried to preserve local and global structures.
Strengths:

- Faster than t-SNE.

Limitations:

- Non-deterministic results.

Key Parameters: `n_neighbors`, `min_dist`, `metric`.
Reference: umap-learn

Isomap (Isometric Mapping)

Mathematical Institute

Type: Non-linear / Geodesic
Concept: Preserves geodesic distances computed along a neighborhood graph.
Strengths:

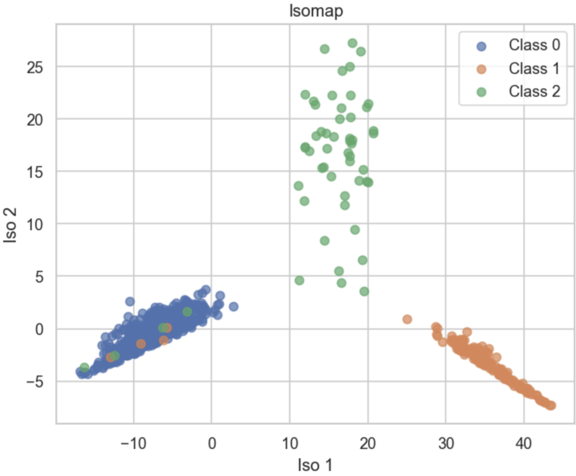
- Captures curved manifolds like the Swiss roll.

Limitations:

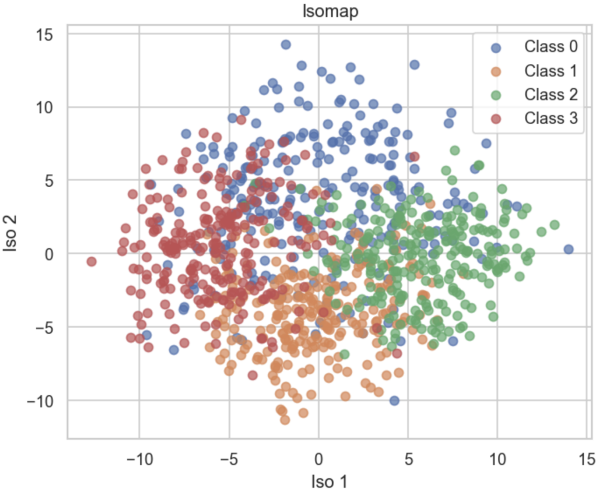
- Sensitive to noise.

Key Parameters: `n_neighbors`, `n_components`, `metric`.
Reference: scikit-learn – Isomap

Isomap


Isomap
Class 0
Class 1
Class 2
Iso 2
Iso 1
25
20
15
10
5
0
−5
−10
0
10
20
30
40

Isomap


Isomap
Class 0
Class 1
Class 2
Class 3
Iso 2
Iso 1
15
10
5
0
−5
−10
−10
−5
0
5
10
15

Multidimensional Scaling (MDS)

Mathematical Institute

Type: Non-linear / Distance-preserving
Concept: Embeds points to preserve pairwise distances as faithfully as possible.
Strengths:

- Works with arbitrary distance metrics.

Limitations:

- Computationally expensive; can get stuck in local minima.

Key Parameters: `n_components`, `metric`, `max_iter`, `dissimilarity`.
Reference: scikit-learn – MDS

Dimensionality Reduction — Summary

- Linear methods (PCA, ICA) are fast and interpretable.
- Non-linear methods (t-SNE, UMAP, Isomap, MDS) uncover complex manifolds.
- It may be useful to scale features before projection.

Dimensionality Reduction — Summary

Oxford Mathematical Institute

Summary Table

Algorithm	Type	Linear / Non-linear	Preserves Local Structure	Preserves Global Structure	Key Parameters	Main Use Case	Link
PCA	Principal components	✅ Linear	⚪ Partial	✅ Strong	`n_components`	Noise reduction, variance analysis	Docs
ICA	Independent components	✅ Linear	⚪ Partial	⚪ Moderate	`n_components`	Source separation, signal analysis	Docs
t-SNE	Probabilistic	❌ Non-linear	✅ Excellent	⚪ Weak	`n_components`, `perplexity`, `learning_rate`	Visualizing high-dimensional clusters	Docs
UMAP	Graph-based	❌ Non-linear	✅ Excellent	✅ Good	`n_neighbors`, `min_dist`, `n_components`	Fast, interpretable 2D embeddings	Docs
Isomap	Geodesic	❌ Non-linear	✅ Good	✅ Good	`n_neighbors`, `n_components`, `metric`	Unfolding curved manifolds	Docs
MDS	Distance-based	❌ Non-linear	⚪ Partial	✅ Strong	`n_components`, `metric`, `max_iter`, `dissimilarity`	Preserving pairwise distances	Docs

Part 4: Classification Algorithms

Part 4: Classification Algorithms

Mathematical Institute

Summary

Algorithm	Type	Linear	Probabilistic	Handles Non-linearity	Key Strength	Link
Gaussian Naive Bayes	Probabilistic	✅ Yes	✅ Yes	❌ No	Very fast baseline	Docs
LDA	Linear Generative	✅ Yes	✅ Yes	❌ No	Simple & interpretable	Docs
QDA	Quadratic Generative	❌ No	✅ Yes	✅ Yes	Captures curved boundaries	Docs
Logistic Regression	Linear Discriminative	✅ Yes	✅ Yes	❌ No	Robust baseline classifier	Docs
SVM	Margin-based	(depends on kernel)	❌ No	✅ Yes (with kernel)	Strong generalization	Docs
k-NN	Instance-based	❌ No	❌ No	✅ Yes	Simple and non-parametric	Docs
Decision Trees	Tree-based	❌ No	✅ (optional)	✅ Yes	Interpretable structure	Docs
Random Forests	Ensemble	❌ No	✅ (probabilities via votes)	✅ Yes	Robust ensemble learner	Docs

Part 4: Classification Algorithms

Gaussian Naive Bayes (GaussianNB)

sklearn: https://scikit-learn.org/stable/modules/naive_bayes.html

```
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()

try:
    clf.fit(X_train_s, y_train)
    y_pred = clf.predict(X_test_s)
    res_gnb = classification_scores(y_test, y_pred)
    print('GaussianNB metrics:')
    for k in ['accuracy','balanced_accuracy','precision_macro','recall_macro','f1_macro']:
        v = res_gnb.get(k, np.nan)
        try:
            print(f'  {k}: {v:.4f}')
        except Exception:
            print(f'  {k}: {v}')

    plot_preds_pca(y_pred, 'GaussianNB predictions', X_test_pca)
    if res_lda['confusion_matrix'] is not None:
        plot_confusion(res_gnb['confusion_matrix'])

except Exception as e:
    print('GaussianNB failed:', e)
```

Gaussian Naive Bayes

Mathematical Institute

Type: Probabilistic (Generative)

Concept: Assumes features are conditionally independent given the class and normally distributed.

Strengths:

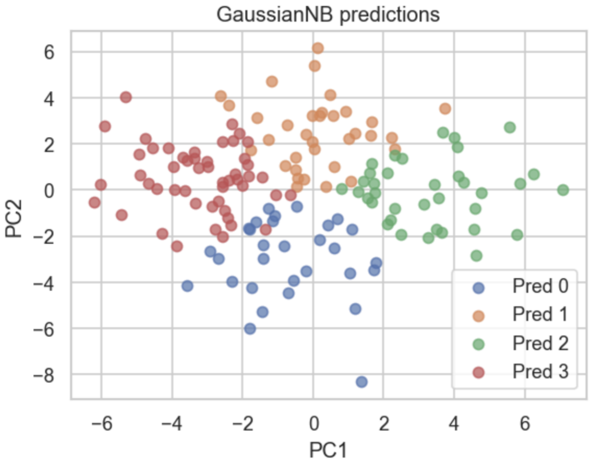
- Very fast and simple; good baseline for many problems.
- Works well with small training sets and high-dimensional data.

Limitations:

- Independence assumption is often violated in practice.
- Performance suffers when features are strongly correlated.

Reference: scikit-learn – Naive Bayes

Gaussian Naive Bayes (example)



Linear Discriminant Analysis (LDA)

Oxford Mathematical Institute

Type: Linear (Generative)
Concept: Models each class with a Gaussian distribution and assumes identical covariance matrices; finds a linear boundary maximizing class separability.
Strengths:

- Interpretable; effective for linearly separable problems.
- Provides probabilistic outputs.

Limitations:

- Assumes normally distributed features and equal covariances across classes.

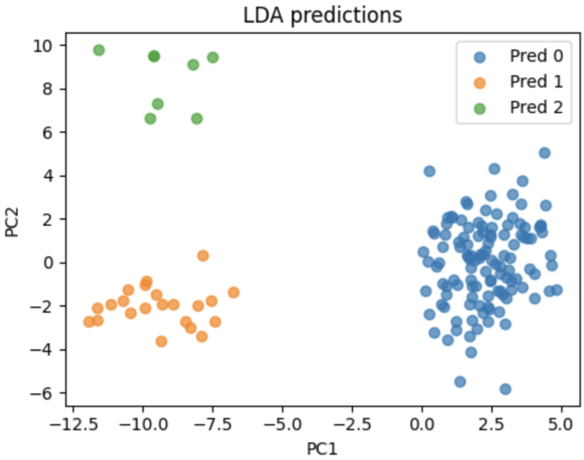
Reference: scikit-learn – LDA/QDA

Linear Discriminant Analysis (LDA)

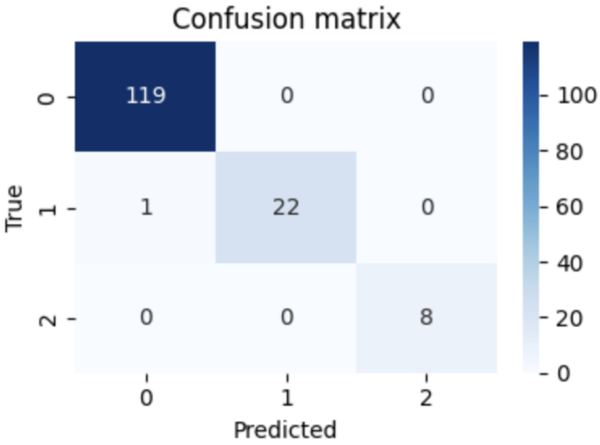
Mathematical Institute

```
LDA metrics:
  accuracy: 0.9933
  balanced_accuracy: 0.9855
  precision_macro: 0.9972
  recall_macro: 0.9855
  f1_macro: 0.9912
```

Linear Discriminant Analysis (LDA)



Linear Discriminant Analysis (LDA)



Quadratic Discriminant Analysis (QDA)

Type: Quadratic (Generative)
Concept: Like LDA but allows each class to have its own covariance — yields quadratic decision boundaries.
Strengths:

- Can model non-linear boundaries between classes.

Limitations:

- Estimates many parameters (covariances); prone to overfitting on small datasets.

Reference: scikit-learn – LDA/QDA

Logistic Regression

Type: Linear (Discriminative)
Concept: Models class probability using the logistic function applied to a linear combination of features.
Strengths:

- Interpretable coefficients, outputs probabilities, robust baseline.

Limitations:

- Only linear decision boundary unless features are transformed or interactions added.

Reference: scikit-learn — Logistic Regression

Support Vector Machine (SVM)

Type: Margin-based (Kernel)
Concept: Finds a hyperplane that maximizes margin between classes; kernels allow non-linear boundaries.
Strengths:

- Effective in high-dimensional spaces; flexible via kernels.

Limitations:

- Sensitive to kernel choice and regularization; can be slow on large datasets.

Key Parameters: `kernel` (linear, rbf, poly).
Reference: scikit-learn – SVM

k-Nearest Neighbors (k-NN)

Type: Instance-based / Non-parametric
Concept: Classifies a sample by majority vote among its k nearest neighbors.
Strengths:

- Simple, non-parametric; adapts to complex boundaries.

Limitations:

- Prediction cost scales with dataset size; requires appropriate scaling and distance metric.

Reference: scikit-learn – Neighbors

Decision Trees

Type: Tree-based
Concept: Recursively splits the feature space to maximize classification accuracy.
Strengths:

- Interpretable and easy to visualize.

Limitations:

- Prone to overfitting if not pruned.

Reference: scikit-learn – Decision Trees

Random Forests

Type: Ensemble of Trees

Concept: Aggregates many decision trees trained on bootstrapped samples and random feature subsets.

Strengths:

- Robust, reduces overfitting compared to single trees; often strong predictive performance.

Limitations:

- Less interpretable than random trees; can be slower and memory-intensive.

Reference: scikit-learn – Random Forests

Classification — Summary

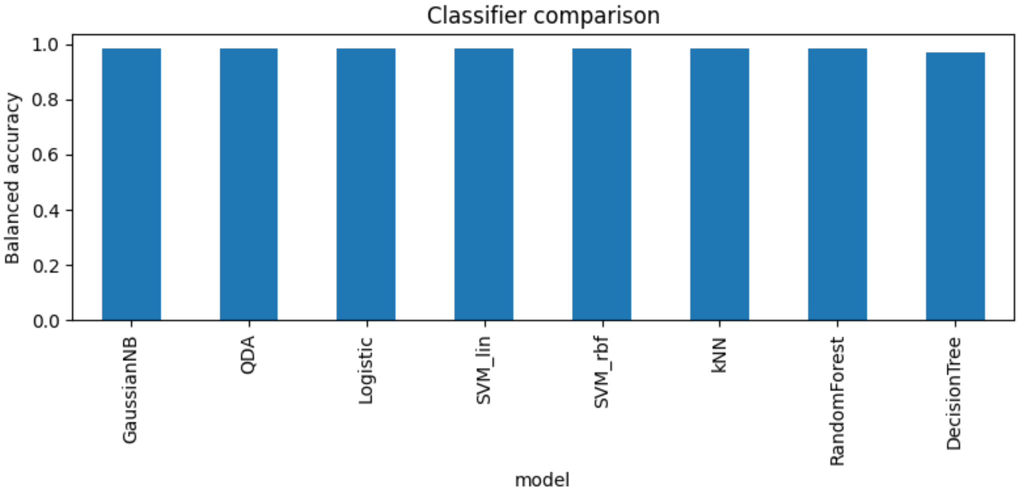
- **Generative methods** (Naive Bayes, LDA, QDA) model class-conditional distributions.
- **Discriminative linear methods** (Logistic Regression) model boundaries directly.
- **Kernel methods** (SVM) handle non-linearities more efficiently.
- **Tree-based methods** (Decision Tree, Random Forest) handle non-linearities more efficiently.
- **Evaluation** with confusion matrices, precision/recall/F1, and unbalanced accuracy.

Classification — Summary

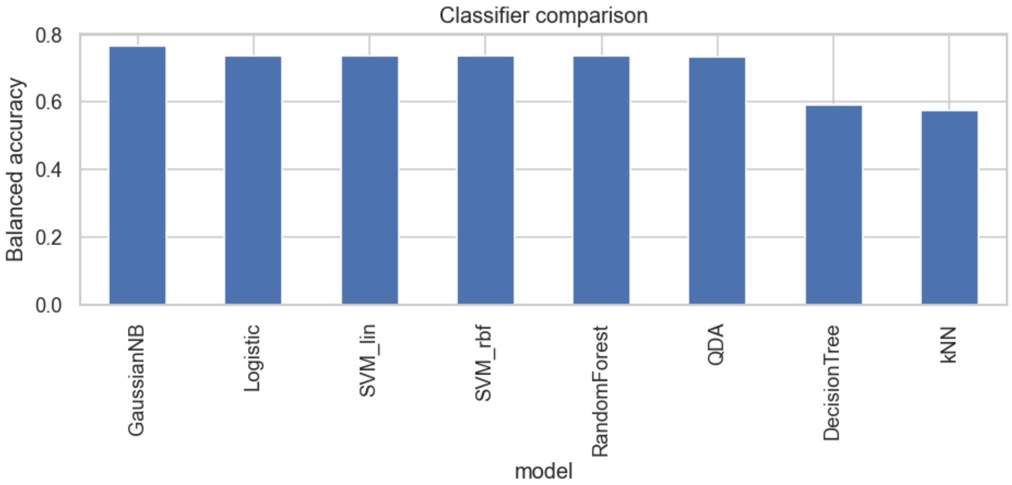
Summary

Algorithm	Type	Linear	Probabilistic	Handles Non-linearity	Key Strength	Link
Gaussian Naive Bayes	Probabilistic	✅ Yes	✅ Yes	❌ No	Very fast baseline	Docs
LDA	Linear Generative	✅ Yes	✅ Yes	❌ No	Simple & interpretable	Docs
QDA	Quadratic Generative	❌ No	✅ Yes	✅ Yes	Captures curved boundaries	Docs
Logistic Regression	Linear Discriminative	✅ Yes	✅ Yes	❌ No	Robust baseline classifier	Docs
SVM	Margin-based	(depends on kernel)	❌ No	✅ Yes (with kernel)	Strong generalization	Docs
k-NN	Instance-based	❌ No	❌ No	✅ Yes	Simple and non-parametric	Docs
Decision Trees	Tree-based	❌ No	✅ (optional)	✅ Yes	Interpretable structure	Docs
Random Forests	Ensemble	❌ No	✅ (probabilities via votes)	✅ Yes	Robust ensemble learner	Docs

Classification — Summary



Classification — Summary



Part 5: Clustering Algorithms

Part 5: Clustering Algorithms

University of Oxford Mathematical Institute

Summary

Algorithm	Type	Needs k	Handles Noise	Suitable For	Link
K-Means	Centroid-based	✅ Yes	❌ No	Spherical clusters	Docs
Agglomerative	Hierarchical	✅ Yes	❌ No	Hierarchical structure	Docs
DBSCAN	Density-based	❌ No	✅ Yes	Arbitrary shapes	Docs
Spectral	Spectral	✅ Yes	❌ No	Non-convex shapes	Docs
BIRCH	Hierarchical (tree-based)	✅ Yes	❌ Limited	Large datasets / streaming	Docs
Affinity Propagation	Message-passing	❌ No	❌ Limited	Automatically finding exemplars	Docs
Gaussian Mixture Models	Probabilistic / Model-based	✅ Yes	❌ Limited	Overlapping / elliptical clusters	Docs

Part 5: Clustering Algorithms

K-Means

sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

```
from sklearn.cluster import KMeans

clf = KMeans(n_clusters=len(np.unique(y_train)), random_state=42)

try:
    clf.fit(X)
    y_pred = clf.predict(X)
    res_kmeans = unsupervised_clustering_metrics(X, y_pred, y_true=y)
```

K-Means

Type: Centroid-based
Concept: Iteratively assigns points to nearest centroid and updates centroids to minimize within-cluster variance.
Strengths:

- Fast and efficient for spherical, well-separated clusters.

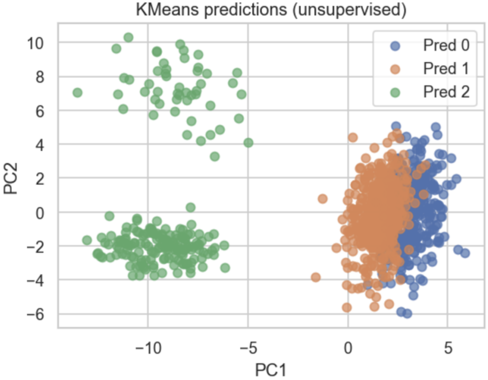
Limitations:

- Requires specifying number of clusters k; sensitive to initialization and outliers.

Key Parameters: `n_clusters`, `n_init`.
Reference: scikit-learn – KMeans

K-Means

```
KMeans unsupervised metrics:
  silhouette_score: 0.1988
  calinski_harabasz_score: 535.5923
  davies_bouldin_score: 1.6674
  mutual_info_score: 0.4683
  rand_score: 0.6651
  homogeneity_score: 0.7428
  fowlkes_mallows_score: 0.6996
  completeness_score: 0.4449
```



Agglomerative Clustering

Mathematical Institute

Type: Hierarchical (Bottom-up)
Concept: Start with each point as a cluster and iteratively merge the closest clusters per chosen linkage.
Strengths:

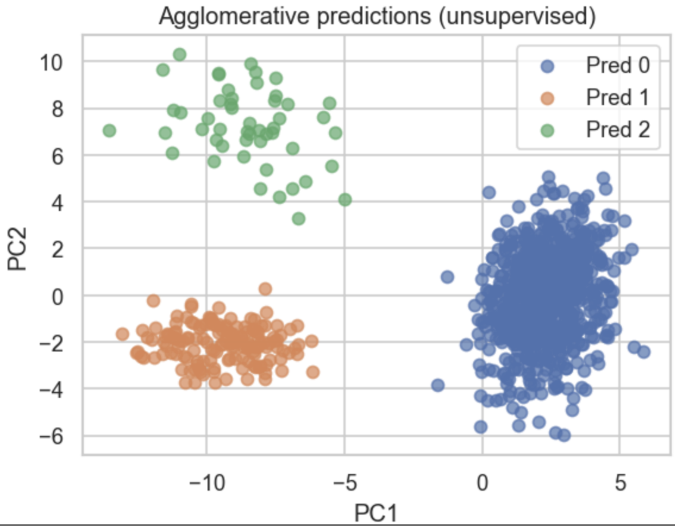
- Produces a dendrogram; flexible linkage criteria.

Limitations:

- n^3 - worst-case complexity; not ideal for very large datasets.

Key Parameters: `n_clusters`, `linkage` (ward, complete, average), `affinity`.
Reference: scikit-learn – AgglomerativeClustering

Agglomerative Clustering



DBSCAN

Type: Density-based
Concept: Forms clusters as high-density regions separated by low-density areas; labels low-density points as outliers.
Strengths:

- Detects arbitrary-shaped clusters and outliers; does not require specifying k.

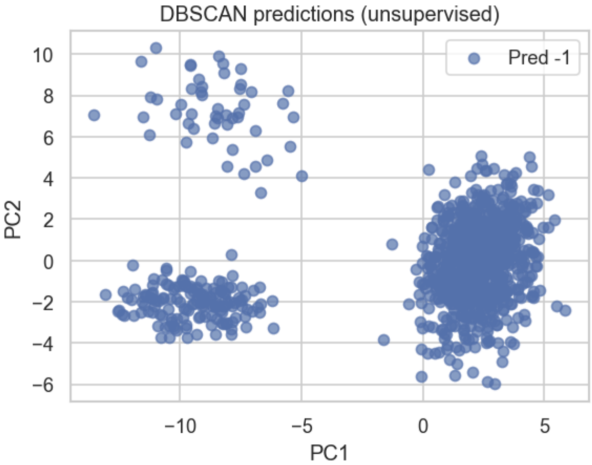
Limitations:

- Sensitive to parameter selection, struggles with varying densities.

Key Parameters: `eps`, `min_samples`.
Reference: scikit-learn – DBSCAN

DBSCAN

Mathematical Institute



Spectral Clustering

Mathematical Institute

Type: Graph / Spectral
Concept: Uses eigenvectors of a graph Laplacian built from data graph, then clusters in the spectral embedding.
Strengths:

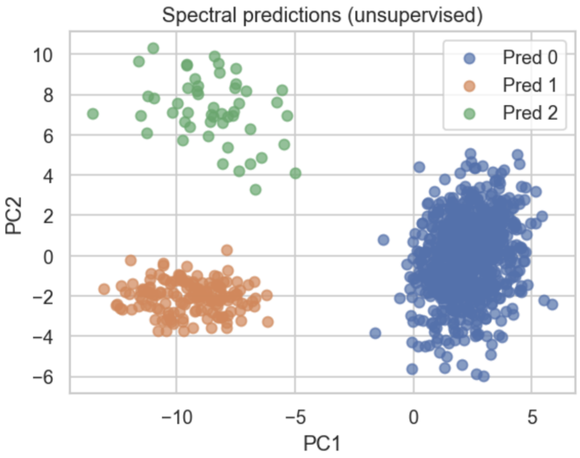
- Effective for non-convex clusters; leverages graph structure.

Limitations:

- Computationally expensive; sensitive to affinity construction.

Key Parameters: `n_clusters`, `affinity` (rbf, nearest_neighbors).
Reference: scikit-learn – SpectralClustering

Spectral Clustering



BIRCH

Mathematical Institute

Type: Hierarchical, Tree-based
Concept: Incrementally builds a CF-tree that summarizes data and clusters using limited memory.
Strengths:

- Scales well to large datasets; useful for streaming data.

Limitations:

- Sensitive to `threshold`; not ideal for arbitrary-shaped clusters.

Key Parameters: `threshold`, `branching_factor`, `n_clusters`.
Reference: scikit-learn – Birch

Affinity Propagation

Mathematical Institute

Type: Message-passing / Graph-based
Concept: Exchanges messages between points to identify exemplars; clusters form around exemplars.
Strengths:

- Automatically selects number of clusters and exemplar points.

Limitations:

- High memory/time costs; sensitive to preference parameter.

Key Parameters: `damping`, `preference`, `max_iter`.
Reference: scikit-learn – AffinityPropagation

Gaussian Mixture Models (GMM)

Mathematical Institute

Type: Probabilistic / Model-based
Concept: Models data as a set of Gaussian distributions, each representing a cluster. Assigns soft probabilities rather than hard labels.
Strengths:

- Handles overlapping clusters; captures elliptical shapes.
- Provides probabilistic cluster membership.

Limitations:

- Requires specifying the number of components k; assumes Gaussian-shaped clusters; may converge to local minimum.

Key Parameters: `n_components`, `covariance_type`.
Reference: scikit-learn – GaussianMixture

Clustering — Summary

- **Centroid-based:** K-Means: fast, simple, assumes spherical clusters.
- **Hierarchical:** Agglomerative, BIRCH: reveals structure, scales to large datasets.
- **Density-based:** DBSCAN: identifies arbitrary shapes and noise.
- **Spectral/Graph-based:** Spectral Clustering, Affinity Propagation: captures complex relationships.
- **Probabilistic:** Gaussian Mixture Models: elliptical clusters.

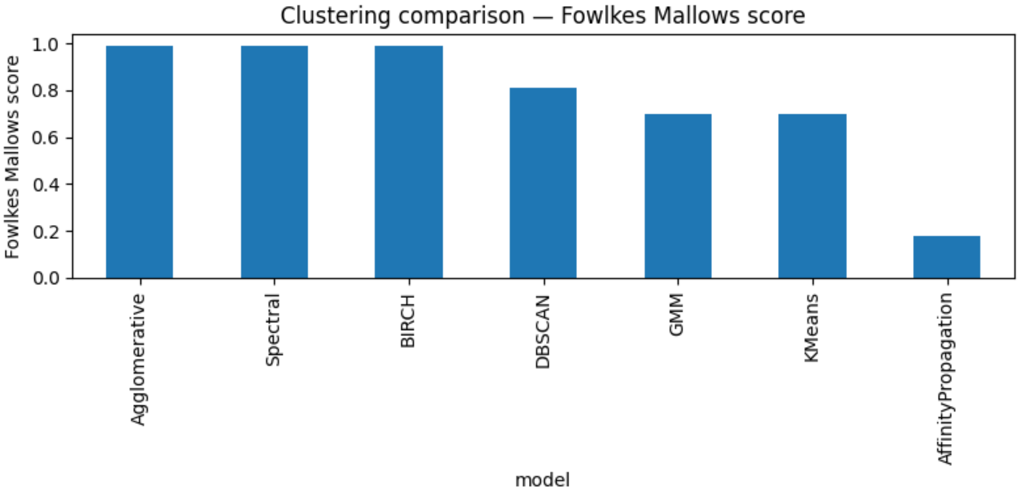
Clustering — Summary

Mathematical Institute

Summary

Algorithm	Type	Needs k	Handles Noise	Suitable For	Link
K-Means	Centroid-based	✅ Yes	❌ No	Spherical clusters	Docs
Agglomerative	Hierarchical	✅ Yes	❌ No	Hierarchical structure	Docs
DBSCAN	Density-based	❌ No	✅ Yes	Arbitrary shapes	Docs
Spectral	Spectral	✅ Yes	❌ No	Non-convex shapes	Docs
BIRCH	Hierarchical (tree-based)	✅ Yes	❌ Limited	Large datasets / streaming	Docs
Affinity Propagation	Message-passing	❌ No	❌ Limited	Automatically finding exemplars	Docs
Gaussian Mixture Models	Probabilistic / Model-based	✅ Yes	❌ Limited	Overlapping / elliptical clusters	Docs

Clustering — Summary



Thank you! Questions?