Clustering and Classification in Machine Learning

in Python

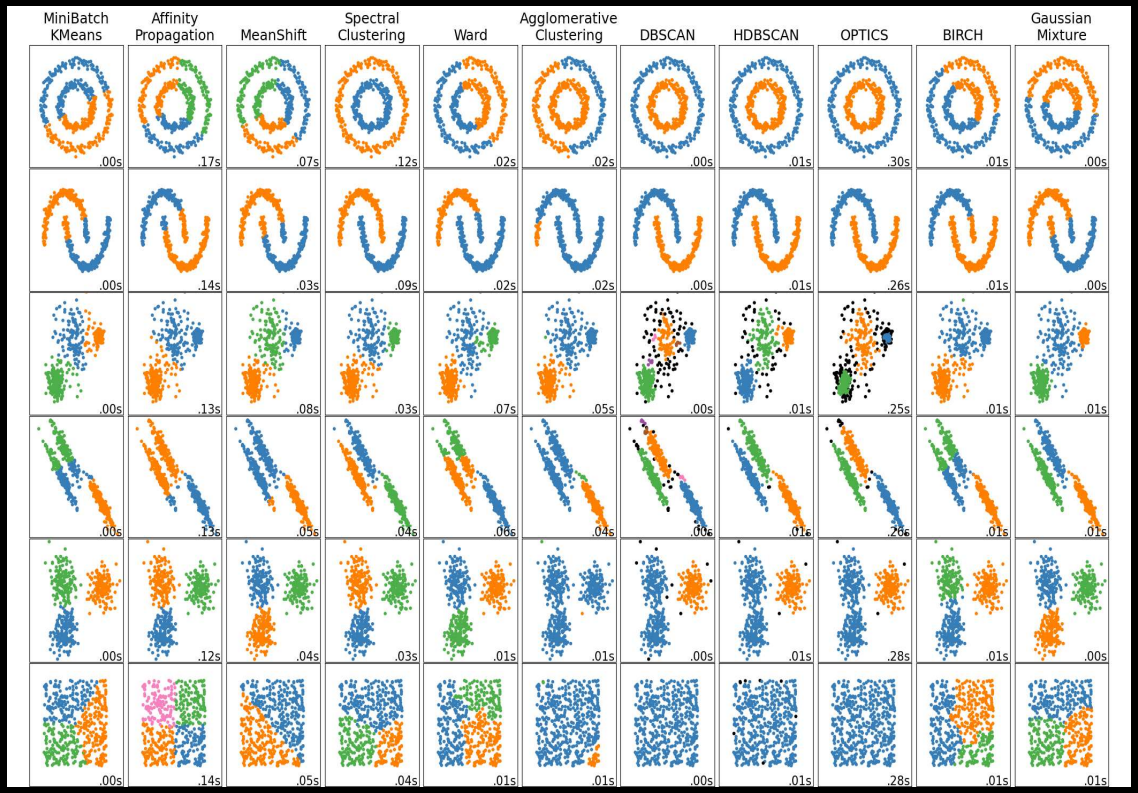
Swathi MVS

Topics

1. Clustering
2. Classification
3. Clustering vs. Classification
4. Application of Clustering
5. Applications of Classification

Clustering

- Imagine - big bag of different colored marbles
- Clustering - sorting marbles and categorizing them into groups based on colors and patterns
- Marbles in same group are more alike, and those in different groups are less alike
- Organize and understand the marbles without knowing their names or labels in advance.
- In Python, clustering algorithms sort data points automatically
- Goal: discover hidden patterns or groups in data



Clustering

Clustering Methods:

- Density-Based Methods

  consider the clusters as the dense region having some similarities and differences from the lower dense region of the space

  - *OPTICS* , *DBSCAN*
- Hierarchical Based Methods

  clusters formed in this method form a tree-type structure based on the hierarchy. New clusters are formed using the previously formed one.

  - **Agglomerative** (bottom-up *approach*)
  - **Divisive** (top-down *approach*
- Partitioning Methods
- Grid-based Methods

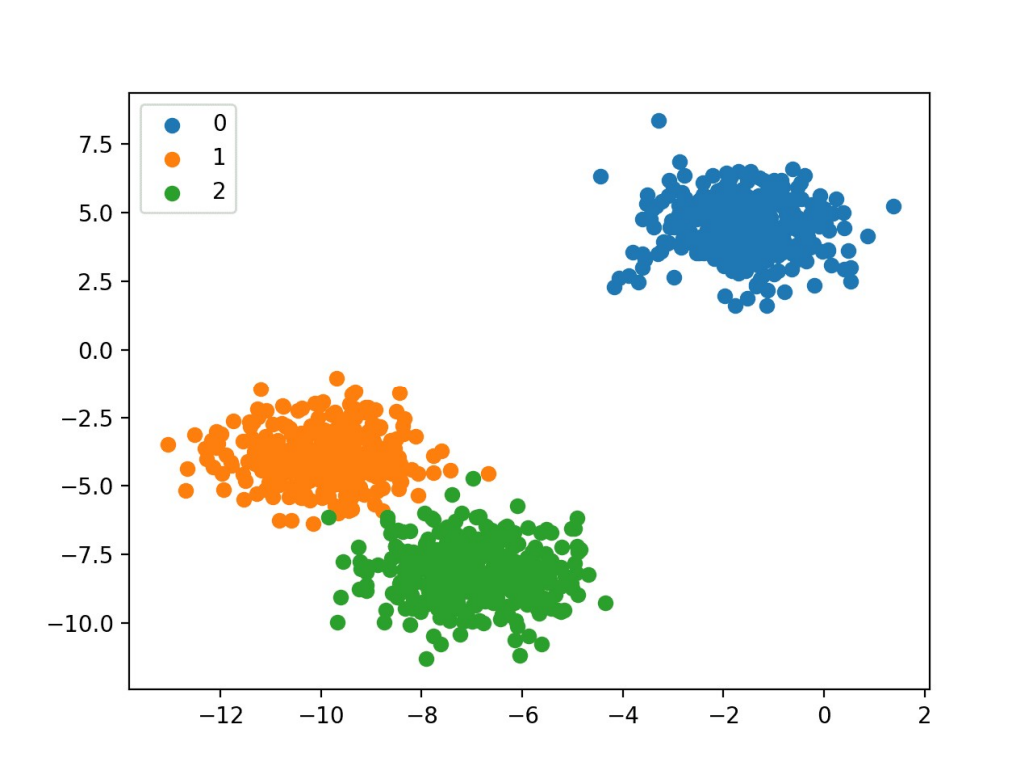
Classification

- Imagine - big pile of fruits, such as blueberries, oranges, and grapes
- Differentiate between fruits

- Provide computer examples of each fruit, with colors, shapes, and sizes.
- Show computer a new fruit – not seen before

- Uses previous knowledge to decide if it's a blueberry, orange, or grape.
- Classification in Python make computer a smart fruit identifier

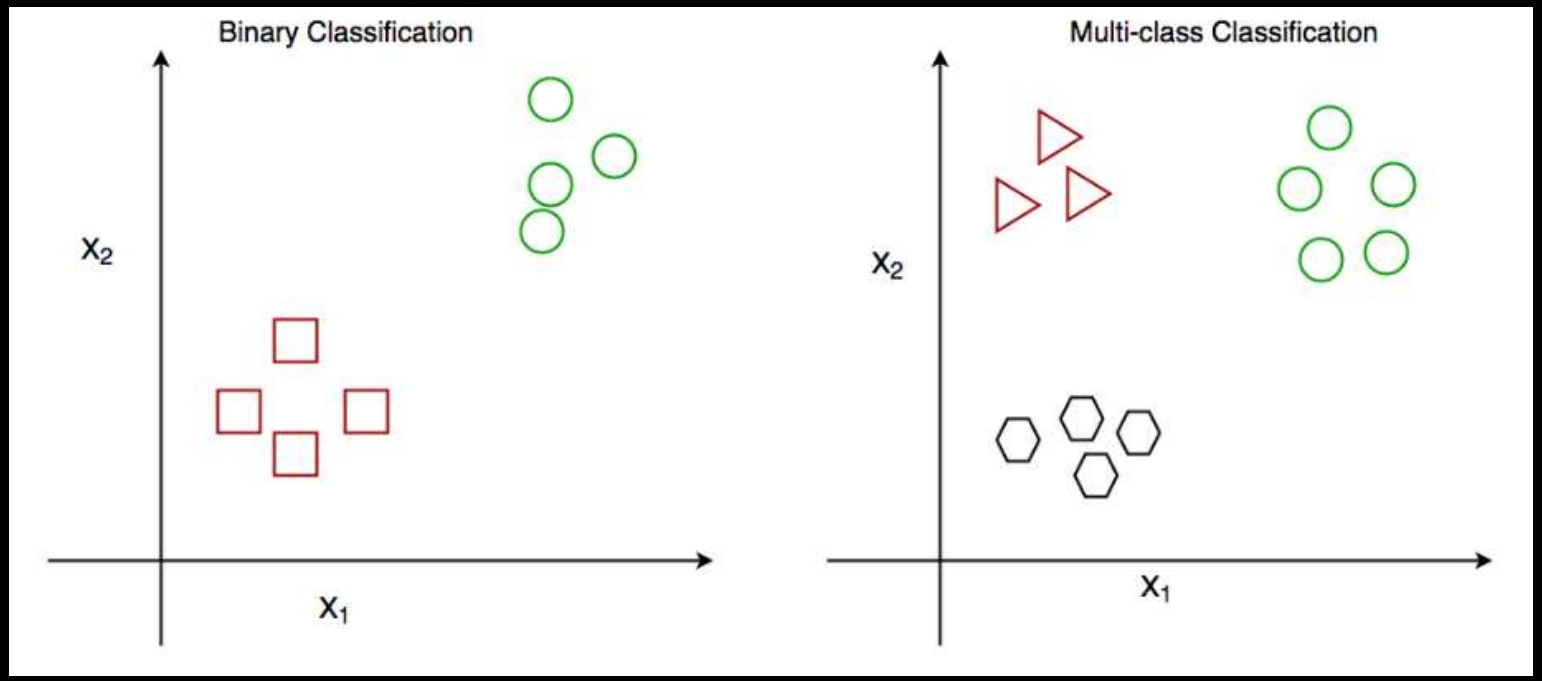
- Make decisions based on characteristics learned from the examples provided.
- Goal: recognize and label things automatically



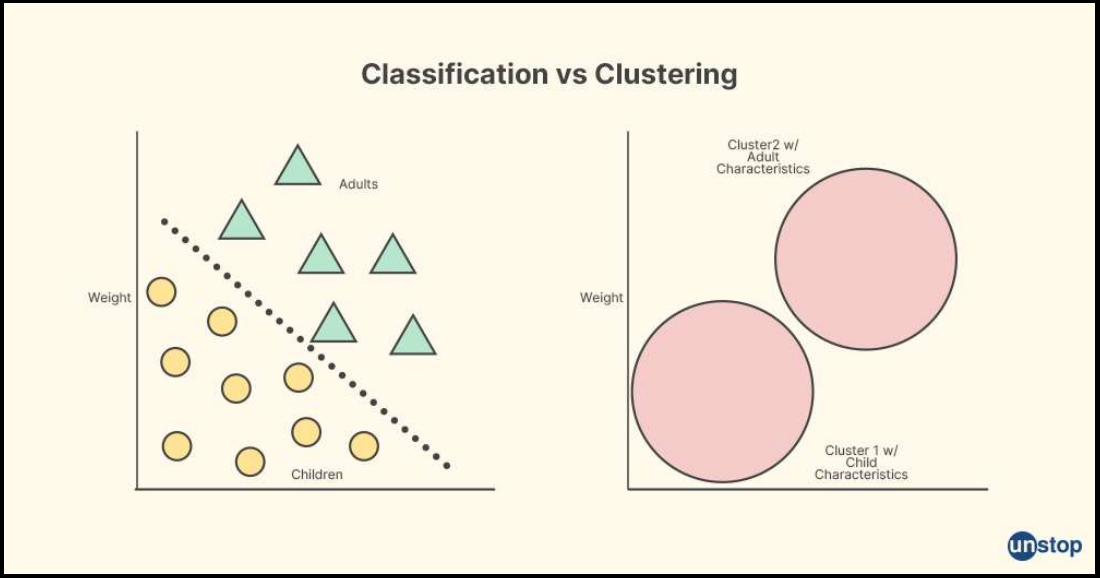
Classification

1. **Binary Classification:** Classify the input into one of two classes or categories.
   Example – On the basis of the given health conditions of a person, we have to determine whether the person has a certain disease or not.
2. **Multiclass Classification**: Classify the input into one of several classes or categories.
   For Example – On the basis of data about different species of flowers, we have to determine which specie our observation belongs to.

Classification



Clustering vs. Classification



Clustering vs. Classification

	Classification	Clustering
Objective	To assign pre-defined classes or labels to instances	To group similar instances based on similarities
Purpose	Predicting the class or label of unseen instances	Discovering inherent patterns or structures
Supervision	Supervised learning	Unsupervised learning
Training	Requires labeled data for training	Does not require labeled data
Output	Class or label assignments	Cluster assignments
Example	Predicting whether an email is spam or not	Grouping customers based on purchasing behavior

Classification

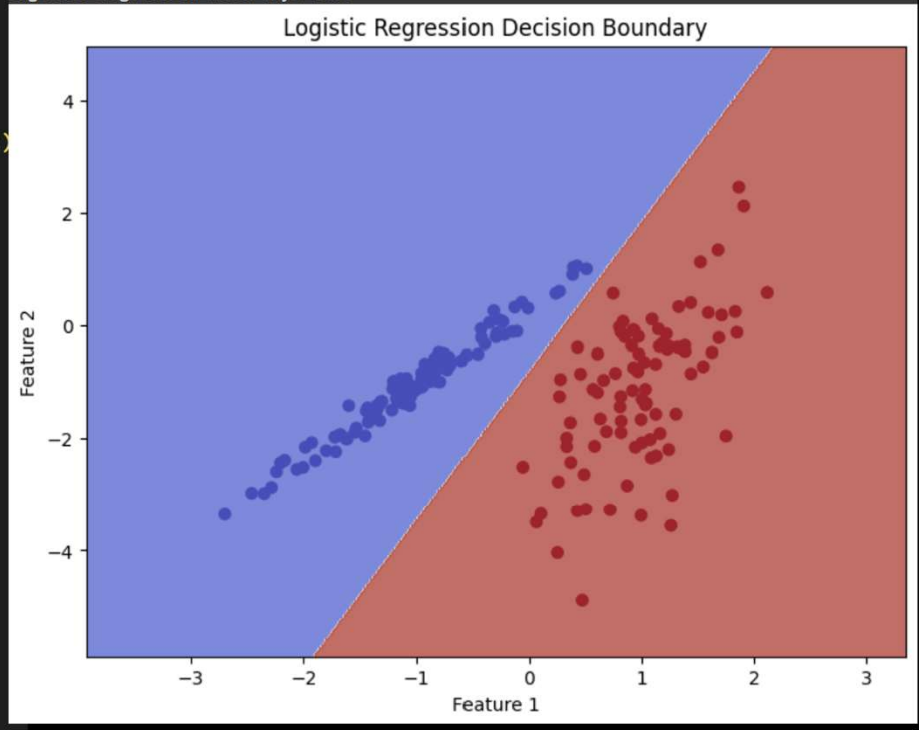
1. **Logistic Regression**:
   Logistic regression is a linear classification algorithm used for binary and multiclass classification tasks. It models the probability of an instance belonging to a particular class.

2. **Support Vector Machines (SVM)**:
   SVMs are used for binary classification tasks and aim to find a hyperplane that maximizes the margin between classes.

Classification – Logistic Regression

```python
#Logistic Regeression
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
# Generate synthetic dataset
X, y = make_classification(n_samples=1000, n_features=2, n_informative=2,
                           n_redundant=0, n_clusters_per_class=1, random_state=4)
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Create and train the Logistic Regression model
logistic_regression = LogisticRegression()
logistic_regression.fit(X_train, y_train)
# Make predictions on the test data
y_pred = logistic_regression.predict(X_test)
# Calculate and print the accuracy
accuracy = np.mean(y_pred == y_test)
print(f"Logistic Regression Accuracy: {accuracy:.2f}")
# Plot the decision boundary
plt.figure(figsize=(8, 6))
h = .02  # Step size in the mesh
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = logistic_regression.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.coolwarm)
# Plot the data points
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=plt.cm.coolwarm)
plt.title("Logistic Regression Decision Boundary")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```

```
Logistic Regression Accuracy: 1.00
```



Clustering

1. **K-Means Clustering**:
   K-Means is one of the most popular clustering algorithms. It partitions data into K clusters, with each data point assigned to the nearest cluster center.

2. **Hierarchical Clustering**:
   Hierarchical clustering builds a tree-like structure of clusters, either bottom-up (agglomerative) or top-down (divisive), allowing for different levels of granularity in cluster assignments.

Clustering – K-Means

```
# k-means clustering
from numpy import unique
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import KMeans
from matplotlib import pyplot
# define dataset
X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2,
                           n_redundant=0, n_clusters_per_class=1, random_state=4)
# define the model
model = KMeans(n_clusters=2)
# fit the model
model.fit(X)
# assign a cluster to each example
yhat = model.predict(X)
# retrieve unique clusters
clusters = unique(yhat)
# create scatter plot for samples from each cluster
for cluster in clusters:
 # get row indexes for samples with this cluster
 row_ix = where(yhat == cluster)
 # create scatter of these samples
 pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
  warnings.warn(
```