# 8 KMeans

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

from sklearn.datasets import load\_wine

from sklearn.metrics import completeness\_score, silhouette\_score, calinski\_harabasz\_score

wine = load\_wine()

X = pd.DataFrame(wine.data, columns=wine.feature\_names)

y = wine.target

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

k = 3

kmeans = KMeans(n\_clusters=k, n\_init=10, random\_state=42)

kmeans.fit(X\_scaled)

centroids = kmeans.cluster\_centers\_

labels = kmeans.labels\_

completeness = completeness\_score(y, labels)

silhouette\_avg = silhouette\_score(X\_scaled, labels)

calinski\_harabasz = calinski\_harabasz\_score(X\_scaled, labels)

print(f'Silhouette Coefficient: {silhouette\_avg:.2f}')

print(f'Calinski-Harabasz Index: {calinski\_harabasz:.2f}')

print(f'Completeness: {completeness:.2f}')

# 9 Hierarchical Clustering (Iris dataset)

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.datasets import load\_iris

from sklearn.metrics import completeness\_score, silhouette\_score, calinski\_harabasz\_score

import scipy.cluster.hierarchy as sch

import matplotlib.pyplot as plt

iris = load\_iris()

X = pd.DataFrame(iris.data, columns=iris.feature\_names)

y = iris.target

data = pd.concat([X, pd.Series(y, name='species')], axis=1)

sample = data.groupby('species').apply(lambda x: x.sample(10, random\_state=42)).reset\_index(drop=True)

X\_sample = sample.drop(columns='species')

y\_sample = sample['species']

scaler = StandardScaler()

X\_sample\_scaled = scaler.fit\_transform(X\_sample)

linked = sch.linkage(X\_sample\_scaled, method='ward')

num\_clusters = 3

labels = sch.fcluster(linked, num\_clusters, criterion='maxclust')

completeness = completeness\_score(y\_sample, labels)

silhouette = silhouette\_score(X\_sample\_scaled, labels)

calinski\_harabasz = calinski\_harabasz\_score(X\_sample\_scaled, labels)

print(f"Number of clusters: {num\_clusters}")

print(f"Completeness Score: {completeness:.2f}")

print(f"Silhouette Score: {silhouette:.2f}")

print(f"Calinski-Harabasz Score: {calinski\_harabasz:.2f}")

species\_colors = {i: color for i, color in enumerate(plt.cm.tab10(np.linspace(0, 1, len(np.unique(y\_sample)))))}

plt.figure(figsize=(12, 8))

dendrogram = sch.dendrogram(

linked,

orientation='top',

labels=y\_sample.values,

distance\_sort='descending',

show\_leaf\_counts=True

)

plt.title('Dendrogram of Hierarchical Clustering on Sample')

plt.xlabel('Sample Index / Species')

plt.ylabel('Euclidean Distance')

plt.show()

#7 Decision Tree (Breast Cancer dataset)

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, classification\_report

from sklearn import tree

import matplotlib.pyplot as plt

data = load\_breast\_cancer()

X = data.data

y = data.target

print("Feature names:", data.feature\_names)

print("Class names:", data.target\_names)

print("First two rows of the dataset:")

print(X[:2])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

clf = DecisionTreeClassifier()

clf.fit(X\_train, y\_train)

y\_pred = clf.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("Classification Report:")

print(classification\_report(y\_test, y\_pred, target\_names=data.target\_names))

plt.figure(figsize=(20,10))

tree.plot\_tree(clf, feature\_names=data.feature\_names, class\_names=data.target\_names, filled=True)

plt.show()

#6 Logistic Regression (Breast Cancer dataset)

import pandas as pd

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_curve, roc\_auc\_score

import matplotlib.pyplot as plt

cancer = load\_breast\_cancer()

X = pd.DataFrame(cancer.data, columns=cancer.feature\_names)

y = cancer.target

print("First few rows of the Breast Cancer dataset:")

print(X.head(1))

print("\nTarget variable distribution:")

print(pd.Series(y).value\_counts())

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

model = LogisticRegression(max\_iter=10000)

model.fit(X\_train\_scaled, y\_train)

y\_pred = model.predict(X\_test\_scaled)

y\_prob = model.predict\_proba(X\_test\_scaled)[:, 1]

print("\nModel Evaluation:")

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.2f}")

print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

fpr, tpr, thresholds = roc\_curve(y\_test, y\_prob)

auc = roc\_auc\_score(y\_test, y\_prob)

plt.figure()

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC)')

plt.legend(loc="lower right")

plt.show()

#5 SVM

# Import necessary libraries

import pandas as pd

from sklearn.datasets import load\_wine

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score, f1\_score

# Load the wine dataset from sklearn

wine = load\_wine()

X = pd.DataFrame(wine.data, columns=wine.feature\_names)

y = pd.Series(wine.target)

# Display basic dataset info

print("Dataset shape:", X.shape)

print("\nFeature names:", wine.feature\_names)

print("\nTarget classes:", wine.target\_names)

# Standardize features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Split into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X\_scaled, y, test\_size=0.3, random\_state=42

)

# Initialize and train the SVM model

model = SVC(kernel='rbf', gamma='scale', C=1.0, class\_weight='balanced', probability=True)

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Calculate and print accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"\nAccuracy: {accuracy:.4f}")

# Confusion Matrix

print("\nConfusion Matrix:")

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

# Precision, Recall, F1 Score

precision = precision\_score(y\_test, y\_pred, average=None)

recall = recall\_score(y\_test, y\_pred, average=None)

f1 = f1\_score(y\_test, y\_pred, average=None)

print("\nPrecision for each class:")

for i, p in enumerate(precision):

print(f"Class {wine.target\_names[i]}: {p:.4f}")

print("\nRecall for each class:")

for i, r in enumerate(recall):

print(f"Class {wine.target\_names[i]}: {r:.4f}")

print("\nF1 Score for each class:")

for i, f in enumerate(f1):

print(f"Class {wine.target\_names[i]}: {f:.4f}")