LIBRARIES:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.linear\_model import LinearRegression, Ridge, LogisticRegression

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

from sklearn.metrics import mean\_squared\_error, r2\_score, accuracy\_score, classification\_report, confusion\_matrix

from sklearn.preprocessing import PolynomialFeatures

from sklearn.pipeline import make\_pipeline

from sklearn.naive\_bayes import GaussianNB

import seaborn as sns

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.cluster import AgglomerativeClustering, KMeans

from scipy.cluster.hierarchy import dendrogram, linkage

from sklearn.decomposition import PCA

from sklearn.metrics import silhouette\_score

LOAD DATASET:

Load Iris dataset

iris = load\_iris()

X = iris.data[:, 0].reshape(-1, 1) # Sepal length (1st feature)

y = iris.data[:, 2].reshape(-1, 1)

TRAIN-TEST SPLIT:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

SIMPLE LINEAR REGRESSION

model = LinearRegression()

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

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

plt.scatter(X\_test, y\_test, color='green', label='Actual')

plt.plot(X\_test, y\_pred, color='red', label='Predicted')

plt.xlabel('Sepal Length (cm)')

plt.ylabel('Petal Length (cm)')

plt.title('Linear Regression on Iris Dataset')

plt.legend()

plt.show()

MULTIPLE LINEAR REGRESSION

iris = load\_iris()

X = iris.data[:, [0, 1, 3]] # Sepal Length, Sepal Width, Petal Width

y = iris.data[:, 2]

model = LinearRegression()

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

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

plt.scatter(y\_test, y\_pred, color='purple')

plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], 'r--')

plt.xlabel('Actual Petal Length')

plt.ylabel('Predicted Petal Length')

plt.title('Actual vs Predicted Petal Length')

plt.grid(True)

plt.show()

POLYNOMIAL REGRESSION

degree = 3

model = make\_pipeline(PolynomialFeatures(degree), LinearRegression())

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

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

plt.scatter(X, y, color='green', label='Actual Data')

plt.plot(X\_plot, y\_plot, color='red', linewidth=2, label=f'Degree {degree} Fit')

plt.xlabel('Sepal Length (cm)')

plt.ylabel('Petal Length (cm)')

plt.title(f'Polynomial Regression (Degree {degree}) on Iris')

plt.legend()

plt.grid(True)

plt.show()

RIDGE REGRESSION

model = Ridge(alpha=0.1)

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

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

EVALUATE

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

NAÏVE BAYES

model = GaussianNB()

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

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

DECISION TREE

model = DecisionTreeClassifier(criterion='gini', max\_depth=3, random\_state=42)

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

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

KNN

k = 5

model = KNeighborsClassifier(n\_neighbors=k)

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

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

LOGISTIC REGRESSION

model = LogisticRegression(max\_iter=200, multi\_class='multinomial', solver='lbfgs')

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

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

SVM

model = SVC(kernel='rbf', C=1.0, gamma='scale') # Try 'linear', 'poly', or 'rbf'

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

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

EVALUATE

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

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

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

# Plot confusion matrix

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=target\_names, yticklabels=target\_names) plt.xlabel("Predicted Labels")

plt.ylabel("True Labels")

plt.title("Confusion Matrix")

plt.show()

K MEANS CLUSTERING

kmeans = KMeans(n\_clusters=3, random\_state=42)

y\_kmeans = kmeans.fit\_predict(X)

# Evaluate the clustering performance using silhouette score

sil\_score = silhouette\_score(X, y\_kmeans)

print(f"Silhouette Score: {sil\_score:.2f}")

# Visualize the clusters using PCA (to reduce to 2D for plotting)

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X)

# Plotting the clusters

plt.figure(figsize=(8, 6))

plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=y\_kmeans, cmap='viridis', marker='o', edgecolor='k')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], c='red', marker='x', s=200, label="Centroids")

plt.title('K-Means Clustering (Iris Dataset) with PCA Projection')

plt.xlabel('PCA 1')

plt.ylabel('PCA 2')

plt.legend()

plt.show()

HIERARCHICAL

# Perform Hierarchical Clustering (Agglomerative)

agg\_clust = AgglomerativeClustering(n\_clusters=3, linkage='ward')

y\_agg\_clust = agg\_clust.fit\_predict(X)

# Plot the Dendrogram

linked = linkage(X, 'ward') # 'ward' minimizes variance within clusters

plt.figure(figsize=(10, 7))

dendrogram(linked, labels=iris.target\_names[y], orientation='top', distance\_sort='descending', show\_leaf\_counts=True)

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('Sample Index or (Cluster Size)')

plt.ylabel('Distance')

plt.show()