R.V.Raghunandan

BL.EN.U4AIE23147

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.neighbors import KNeighborsClassifier

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

np.random.seed(42)

X = np.random.rand(100, 5) \* 10 # Features

y = np.random.choice([0, 1], 100) # Class labels

A1: . Evaluate the intraclass spread and interclass distances between the classes in your dataset. If your data deals with multiple classes, you can take any two classes. Steps below (refer below diagram for understanding):

• Calculate the mean for each class (also called as class centroid) (Suggestion: You may use numpy.mean() function for finding the average vector for all vectors in a given class. Please define the axis property appropriately to use this function.

EX: feat\_vecs.mean(axis=0))

• Calculate spread (standard deviation) for each class (Suggestion: You may use numpy.std() function for finding the standard deviation vector for all vectors in a given class. Please define the axis property appropriately to use this function.)

• Calculate the distance between mean vectors between classes (Suggestion: numpy.linalg.norm(centroid1 – centroid2) gives the Euclidean distance between two centroids.)

Soln:

class\_0 = X[y == 0]

class\_1 = X[y == 1]

centroid\_0 = np.mean(class\_0, axis=0)

centroid\_1 = np.mean(class\_1, axis=0)

spread\_0 = np.std(class\_0, axis=0)

spread\_1 = np.std(class\_1, axis=0)

distance\_between\_centroids = np.linalg.norm(centroid\_0 - centroid\_1)

A2: Take any feature from your dataset. Observe the density pattern for that feature by plotting the histogram. Use buckets (data in ranges) for histogram generation and study. Calculate the mean and variance from the available data. (Suggestion: numpy.histogram()gives the histogram data. Plot of histogram may be achieved with matplotlib.pyplot.hist())

Soln:

feature\_index = 2

plt.hist(X[:, feature\_index], bins=10, alpha=0.7, color='blue', edgecolor='black')

plt.xlabel("Feature Value")

plt.ylabel("Frequency")

plt.title("Histogram of Feature")

plt.show()

A3: Take any two feature vectors from your dataset. Calculate the Minkwoski distance with r from 1 to 10. Make a plot of the distance and observe the nature of this graph.  
  
Soln:

minkowski\_distances = [np.linalg.norm(X[0] - X[1], ord=r) for r in range(1, 11)]

plt.plot(range(1, 11), minkowski\_distances, marker='o')

plt.xlabel("r value")

plt.ylabel("Minkowski Distance")

plt.title("Minkowski Distance for Different r values")

plt.show()

A4: . Divide dataset in your project into two parts – train & test set. To accomplish this, use the train test\_split() function available in SciKit. See below sample code for help:  
  
Soln:

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

A5: Train a kNN classifier (k =3) using the training set obtained from above exercise. Following code for help:

>>> import numpy as np

>>> from sklearn.neighbors import KNeighborsClassifier

>>> neigh = KNeighborsClassifier(n\_neighbors=3)

>>> neigh.fit(X, y)

Soln:

neigh = KNeighborsClassifier(n\_neighbors=3)

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

A6: Test the accuracy of the kNN using the test set obtained from above exercise. Following code for help.

>>> neigh.score(X\_test, y\_test)

This code shall generate an accuracy report for you. Please study the report and understand it.

Soln:

accuracy = neigh.score(X\_test, y\_test)

A7: Use the predict() function to study the prediction behavior of the classifier for test vectors.

>>> neigh.predict(X\_test)

Perform classification for a given vector using neigh.predict(<>). This shall produce the class of the test vector (test\_vect is any feature vector from your test set).

Soln:

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

A8: Make k = 1 to implement NN classifier and compare the results with kNN (k = 3). Vary k from 1 to 11 and make an accuracy plot.  
  
Soln:

k\_values = range(1, 12)

accuracies = []

for k in k\_values:

knn = KNeighborsClassifier(n\_neighbors=k)

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

accuracies.append(knn.score(X\_test, y\_test))

plt.plot(k\_values, accuracies, marker='o')

plt.xlabel("k value")

plt.ylabel("Accuracy")

plt.title("k-NN Accuracy for Different k Values")

plt.show()

A9: Please evaluate confusion matrix for your classification problem. From confusion matrix, the other performance metrics such as precision, recall and F1-Score measures for both training and test data. Based on your observations, infer the models learning outcome (underfit / regularfit / overfit).  
  
Soln:

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

report = classification\_report(y\_test, y\_pred)

accuracy, conf\_matrix, report