

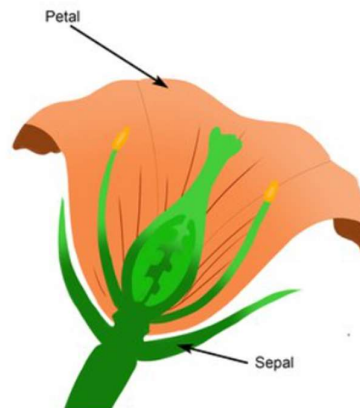
IC 272 - Data Science III

Assignment 2: Principal Component Analysis (PCA) and K-Nearest Neighbour (KNN)

Deadline: September 15, 2024: 23.59 Hr.

Dataset Description:

You are given a CSV file "**iris.csv**" containing the measurements of three types of Iris flowers:



Independent variables/ Attributes/ Features:

- i. "SepalLengthCm": Sepal length in cm.
- ii. "SepalWidthCm": Sepal width in cm.
- iii. "PetalLengthCm": Petal length in cm.
- iv. "PetalWidthCm": Petal width in cm.

Dependent variable/ Target Attribute/ Class:

- i. "species": Type of the flower corresponding to a set of measurements.

Problem Statements:


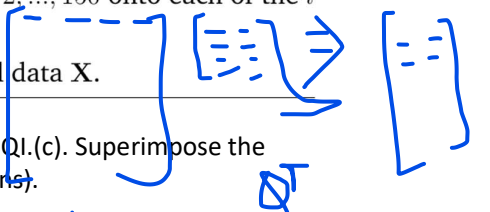
I. Read the file using Pandas and do the following:

- a. Extract the attributes as one matrix and the target attribute as an array, called “true class labels” or y .
- b. Replace the outliers (if at all any) in the attributes with the median of the respective attributes. Let's call this outlier corrected data as X .
- c. Reduce the dimension of X through Principal Component Analysis (PCA) by implementing Algorithm 1 in Python:

Algorithm 1 Data Dimension Reduction using Principal Component Analysis (PCA)

Require: Data matrix X having $N = 150$ number of samples. Each sample is of dimension $d = 4$.

Ensure: The reduced-dimensional data - dimension is reduced from $d = 4$ to $l = 2$.

1. Subtract the respective mean from each attribute (dimension) in data samples (tuples). Let's call the mean subtracted data as \tilde{X} .
2. Compute the correlation matrix $C = \tilde{X}^T \tilde{X}$. 4×4 
3. Perform eigen analysis of C using `numpy.linalg.eig`.
4. Order the eigenvalues (λ 's) of C such that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_4$.
5. Consider $l = 2$ eigenvectors (q 's) corresponding to the l significant eigenvalues and create a matrix Q of size 2×4 .
6. Take the original data X . Project each original sample x_i , $i = 1, 2, \dots, 150$ onto each of the l directions performing the following operation: $\hat{X} = QX$. 
7. Return the dimension-reduced version $\hat{X} \in \mathbb{R}^{150 \times 2}$ of the original data X .
- d. Draw a scatter plot of the dimension-reduced data generated in Q1.(c). Superimpose the eigen directions with proper scaling (use in-built plotting functions).
- e. Reconstruct the original data from dimension-reduced data generated in Q1.(c) by implementing Algorithm 2 in Python:

Algorithm 2 Reconstruction of Data from Dimension-reduced Data

Require: 1. Dimension-reduced data matrix \hat{X} having $N = 150$ number of samples, each of dimension $l = 2$.

2. Eigenvector matrix Q .

Ensure: The reconstructed original data $\hat{X} \in \mathbb{R}^{150 \times 4}$.

- 1: Approximate each original sample x_i , $i = 1, 2, \dots, 150$ by performing the following operation:
 $\hat{X} = \hat{X}Q$.
- 2: Return the reconstructed original data \hat{X} of size 150×4 .

-
- f. Compute the RMSE between the original data (X) and its reconstruction (\hat{X}) using your implementation in Assignment 1.

II. Consider the dimension-reduced data generated in QI.(c) and do the following:

- a. Build a K-Nearest Neighbour (KNN) classifier by implementing Algorithm 3 in Python:

Algorithm 3 K-Nearest Neighbour (KNN)

Require: 1. Dimension-reduced data matrix $\hat{\mathbf{X}}$ having $N = 150$ number of samples, each of dimension $l = 2$.

2. True class label vector \mathbf{y} .

Ensure: A matrix containing the true class label (actual) and the predicted class label (predicted) of each test sample \mathbf{x}_i^{test} .

- 1: Split data $\hat{\mathbf{X}}$ into training set and test set using
`sklearn.model_selection.train_test_split($\hat{\mathbf{X}}$, \mathbf{y} , random_state=104,
test_size=0.20, shuffle=True).`
- 2: **for** each test sample \mathbf{X}_i^{test} in the test set **do**
- 3: **for** each training sample \mathbf{X}_j^{train} in the training set **do**
- 4: Compute the Euclidean distance between \mathbf{X}_i^{test} and \mathbf{X}_j^{train} as

$$dist = \sqrt{\sum_{l=1}^2 (\mathbf{X}_{il}^{test} - \mathbf{X}_{jl}^{train})^2}$$

- 5: **end for**
- 6: Sort the training examples in ascending order of the distance to \mathbf{X}_i^{test} (use in-built sorting function).
- 7: Choose the first $K = 5$ examples in the sorted list.
- 8: Assign \mathbf{X}_i^{test} to the most common class among its 5 neighbours.
- 9: Save the actual and predicted class labels of \mathbf{X}_i^{test} .
- 10: **end for**
- 11: Return the matrix of actual and predicted class labels of 150×2 samples.
-

- b. Compute the confusion matrix using the function from `sklearn.metrics`. Create an interpretable visual display of the confusion matrix using the function from `sklearn.metrics` and plot it using `matplotlib.pyplot`.