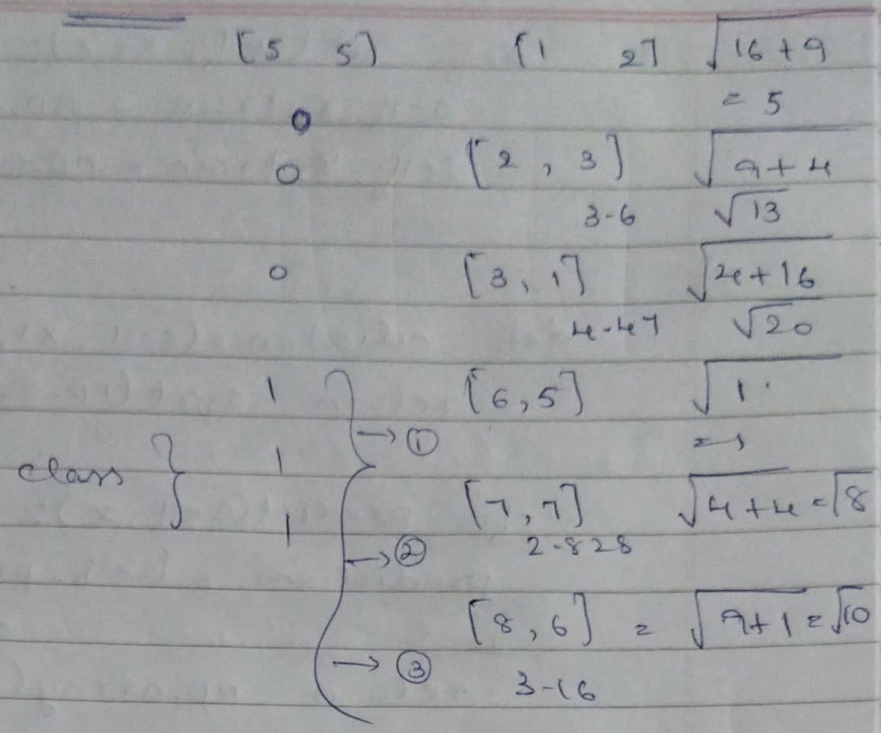
7/4/2025

KNN



KNN is a supervised learning algorithm used for classification and regression.

steps:-

- Choose the value of K (number of neighbour)
- calculate the distance (euclidean) between the new_data and training samples
- Sort and pick the k-nearest neighbour.
- Take a majority vote (for classification)
- Assign the most frequent class as the predicted label.

```
import numpy as np
from collections import Counter

class KNN:
    def __init__(self, k=3):
        self.k = k
```

```
def fit(self, x, y):
    self.x_train = np.array(x)
    self.y_train = np.array(y)

def edistance(self, x1, x2):
    return sqrt(np.sum((x1-x2)**2))

def predict(self, x):
    predictions = [self._predict(x) for
                   x in X]
    return np.array(predictions)

def predict(self, x)
    d = [self.edistance(x, x_train) for
         x_train in self.x_train]

    k_i = np.argsort(d)[:self.k]

    k_labels = [self.y_train[i] for i in
                k_indices]

    most_common = Counter(k_nearest_
                  labels).
                  most_comm(1)

    return most_common[0][0]

x, y = load_iris_data()
train_size = int(0.8 * len(x))
x_train, x_test = x[:train_size], x[train_size:]
```

y_train, y_test = y[:trainsize], y[trainsize:]

km = KNN(k=3)
knn.fit(x_train, y_train)
predicts = knn.predict(x_test)

* Support Vector Machine. (SVM)

Steps :-

1) Input the data
   - Feature matrix X consisting of (samples x features)
   - Target labels y (multi-class 0, 1, 2)
   - Regularization constant c
   - max_iteration

2) Split the Dataset into training set (70%.) and testing_set (30%.)

3) Initialize the SVM classifier
   set the following parameters.
   C : regularization constant
   max_iter: Number of training iterations
   Kernel - linear defined $K(x \cdot x') = x \cdot x'$

4) One vs Rest Training strategy
   For each class c in the set of unique classes: convert labels into Binary format

$$y_{binary} = \begin{cases} 1 & \text{if } y = c \\ -1 & \text{otherwise} \end{cases}$$

- Train a binary SVM classifier using the Sequential Minimal optimization

5) Binary SVM Training

Initialize

$\alpha = 0$ Langrange multiplier

$\beta = 0$ Bias term

Repeat for max_iter: iterations

For each training sample i:

- Randomly select another index
- compute prediction errors $j \neq i$

E_i and E_j

- save old values α_i, α_j
- compute bounds L, H

If $L == H$ continue

- compute

$$\eta = 2K(x_i, x_j) - K(x_i, x_i) - K(x_j, x_j)$$

If $\eta \geq 0$ skip update

- Update α_j

$$\alpha_j = \alpha_j + y_j \frac{(E_i - E_j)}{\eta}$$

- clip $\alpha_j \in [L, H]$
- Update α_i
- Compute β

e. Prediction Phase

For each test sample x

- For each trained binary classifier:
  - Compute decision score

$$f(x) = \sum_j \alpha_j y_j K(x_i, x_j) + b$$

- Store the score
- Predict the class with max decision score

f) Evaluation

- Compare predicted labels y_{pred} with true labels y_{test}
- calculate accuracy

$$\text{Accuracy} = \frac{\text{No of correct predictions}}{\text{No of samples.}} \times 100$$

7/4