#5 Writing our First Classifier

L K-Nearest Neighbours (KNN)

Date | Page No

Step-1 => Comment out imports

We will comment to classifier which we imported during writing pipeline.

Step-2 => Implment class for classifier

```
Class ScrappyKNN():
    ~~def fit(self, x-train, y-train):~~
```

Step-3 => Understand the Interface [~~under~~ What method we need to implement]

```
S5  def fit(Self, X-train, y-train):
        self.X_train = X_train
        Self.Y_train = Y_train

S3  def Predict (Self, x_test):
        Predictions = [ ]
        for each row in x_test:
            label = Self.closest(row)
            predictions. append(label)
        return Predictions.
```

Step4: Get pipeline working

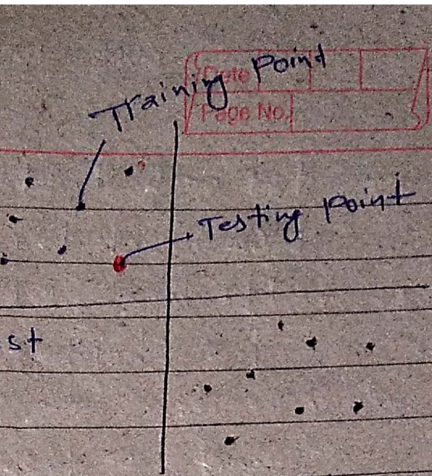
L accuracy [33%] for Random classifier

Calculate the distance from test point to 1st training point ←

```
def closest(self, row):
    best_dist = euc(row, self.X_train[0])
    best_index = 0
    for i in range(1, len(self. train)):
        dist = euc (row, self. X-train[i])
        if dist < best_dist:
            best_dist = dist
            best_index = i
    return self. y-train[(best_index)]
```

REST CODE WILL BE SAME AS PREVIOUS CLASS.

Step 5 → Intro to K-NN



For given testing Point, the algorithm finds the closest training Point.

predictions → The testing Point is then predicted to have the Same label as its nearest neighbr.

Role of K → If there is ties in distance or to make predictions more robust, K come into Play. K Represent the number of neighbours to consider when making a Prediction.

For example → K=3, the classifier looks at three Closest Point. But for simplicity we take K=1.

Step 6 → Measure distance

- To determine the "closest ~~distance~~" neighbour, a distance metric is required. ~~The video introduce~~
- Use Euclidean distance → Measure St. line distance b/w two Points $A^2+B^2=C^2$. It Can be used by implementing scipy library.

$$d(a,b) = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

it work Same for all the visualization 1D - 2D - 3D ----

```
from scipy.spatial import distance
def euc(a,b):
    return distance.euclidean(a,b)
```

a is point from training data
b is point from testing data

Step 7→ Implement nearest neighbor algorithm.

- For each test point, the classifier Calculates the Euclidean distance to all training Points.
- It then iterate through these distance to find the shortest one.
- variables are maintained to track the shortest_dist found so far and the index of the training Point corresponding to that shortest dist.

→ finally using the Index, the label of the closest training example is retrieved and returned as the prediction for this test point.

Step-8 Run Pipeline

↳ Accy got 90% +