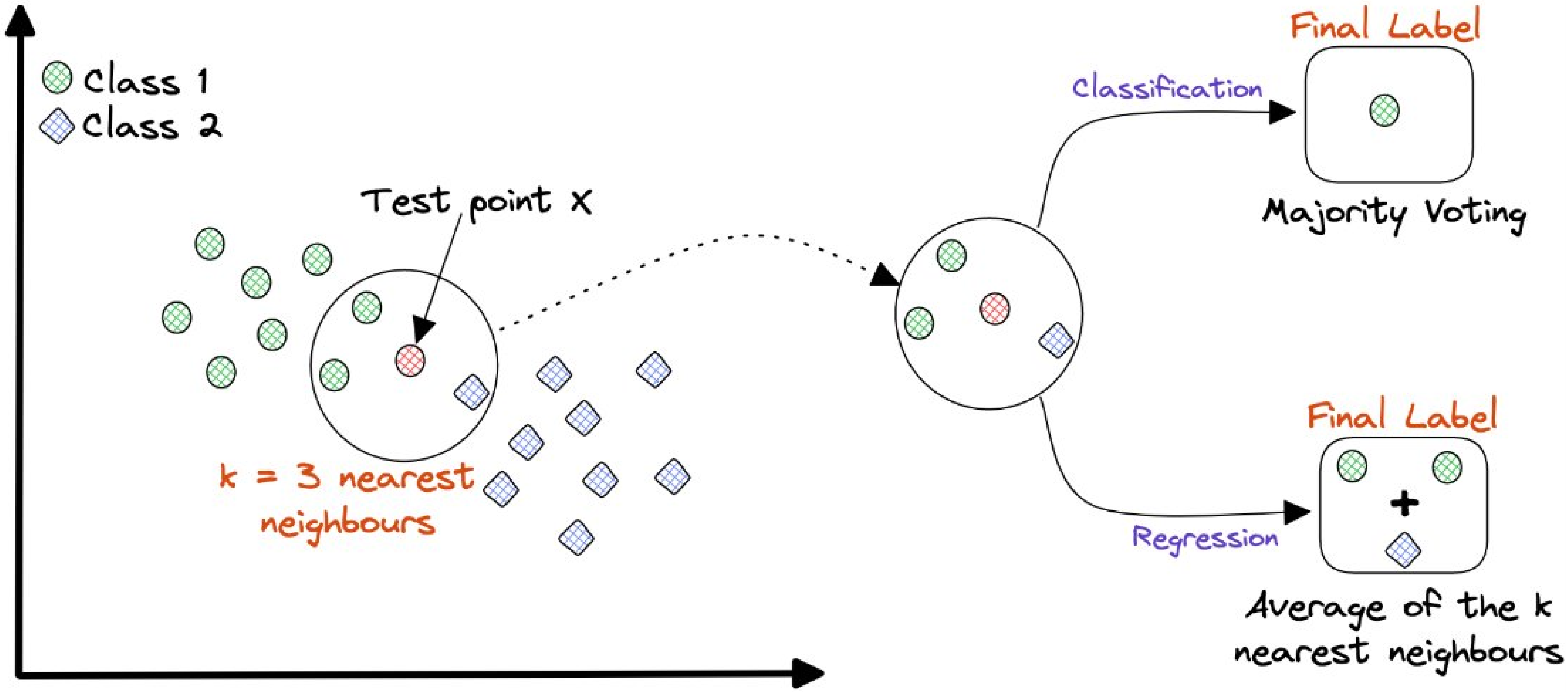
KNN:K Nearest Neighbours



@akshay_pachaar

KNN involves 4 simple steps:

1️⃣ Choose the value of K

2️⃣ Calculate the distance between the new & existing data points

3️⃣ Find K closest data points (neighbours)

4️⃣ For classification, vote for the majority class, for regression,compute the average.

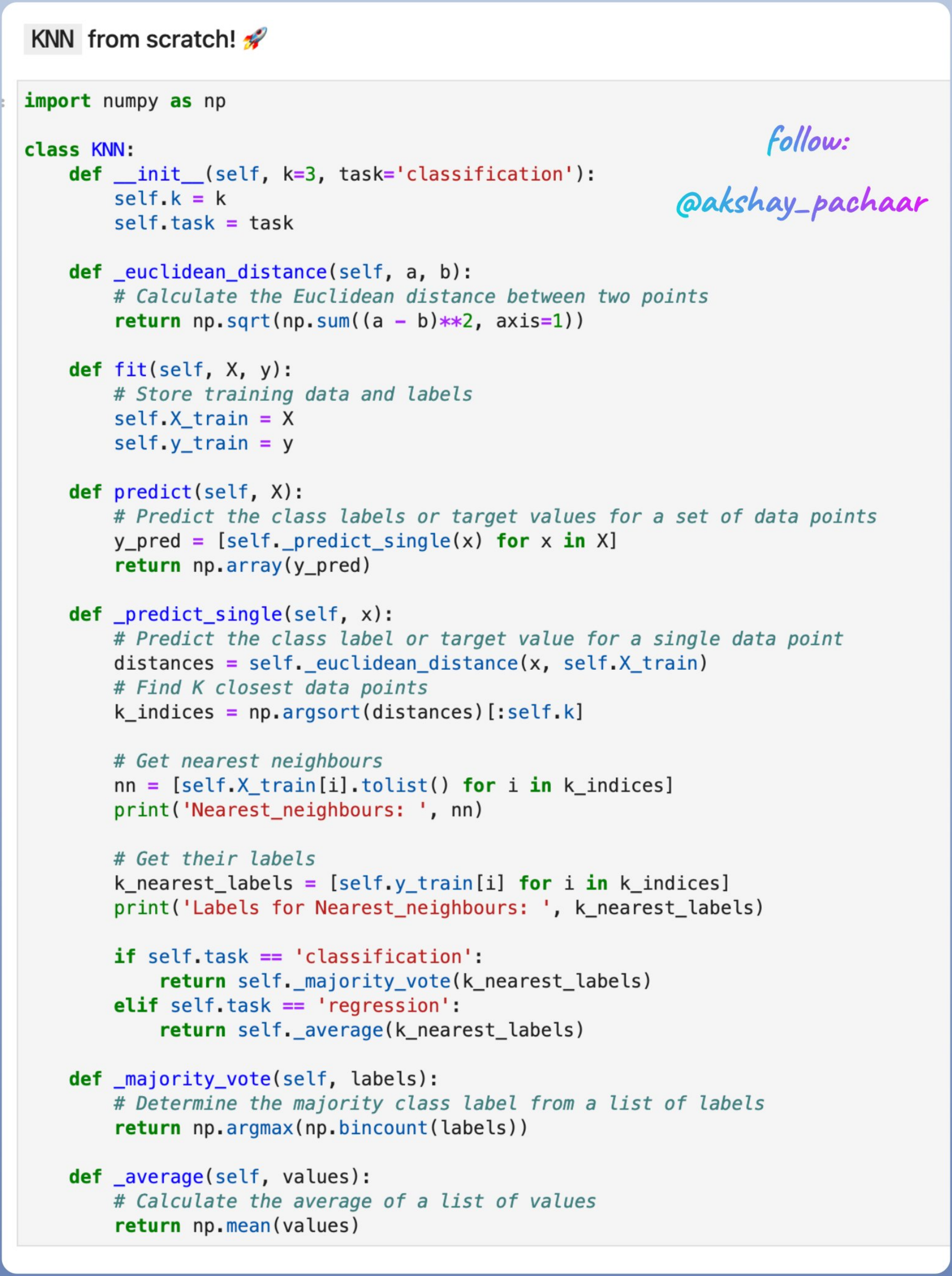
Swipe 👉

@akshay_pachaar

Here's a didactic implementation of KNN from scratch!

Seeing things in code makes our understanding more concrete! 👊

Swipe 👉



KNN from scratch! 🚀

follow:

@akshay_pachaar

```
import numpy as np

class KNN:
    def __init__(self, k=3, task='classification'):
        self.k = k
        self.task = task

    def _euclidean_distance(self, a, b):
        # Calculate the Euclidean distance between two points
        return np.sqrt(np.sum((a - b)**2, axis=1))

    def fit(self, X, y):
        # Store training data and labels
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        # Predict the class labels or target values for a set of data points
        y_pred = [self._predict_single(x) for x in X]
        return np.array(y_pred)

    def _predict_single(self, x):
        # Predict the class label or target value for a single data point
        distances = self._euclidean_distance(x, self.X_train)
        # Find K closest data points
        k_indices = np.argsort(distances)[:self.k]

        # Get nearest neighbours
        nn = [self.X_train[i].tolist() for i in k_indices]
        print('Nearest_neighbours: ', nn)

        # Get their labels
        k_nearest_labels = [self.y_train[i] for i in k_indices]
        print('Labels for Nearest_neighbours: ', k_nearest_labels)

        if self.task == 'classification':
            return self._majority_vote(k_nearest_labels)
        elif self.task == 'regression':
            return self._average(k_nearest_labels)

    def _majority_vote(self, labels):
        # Determine the majority class label from a list of labels
        return np.argmax(np.bincount(labels))

    def _average(self, values):
        # Calculate the average of a list of values
        return np.mean(values)
```

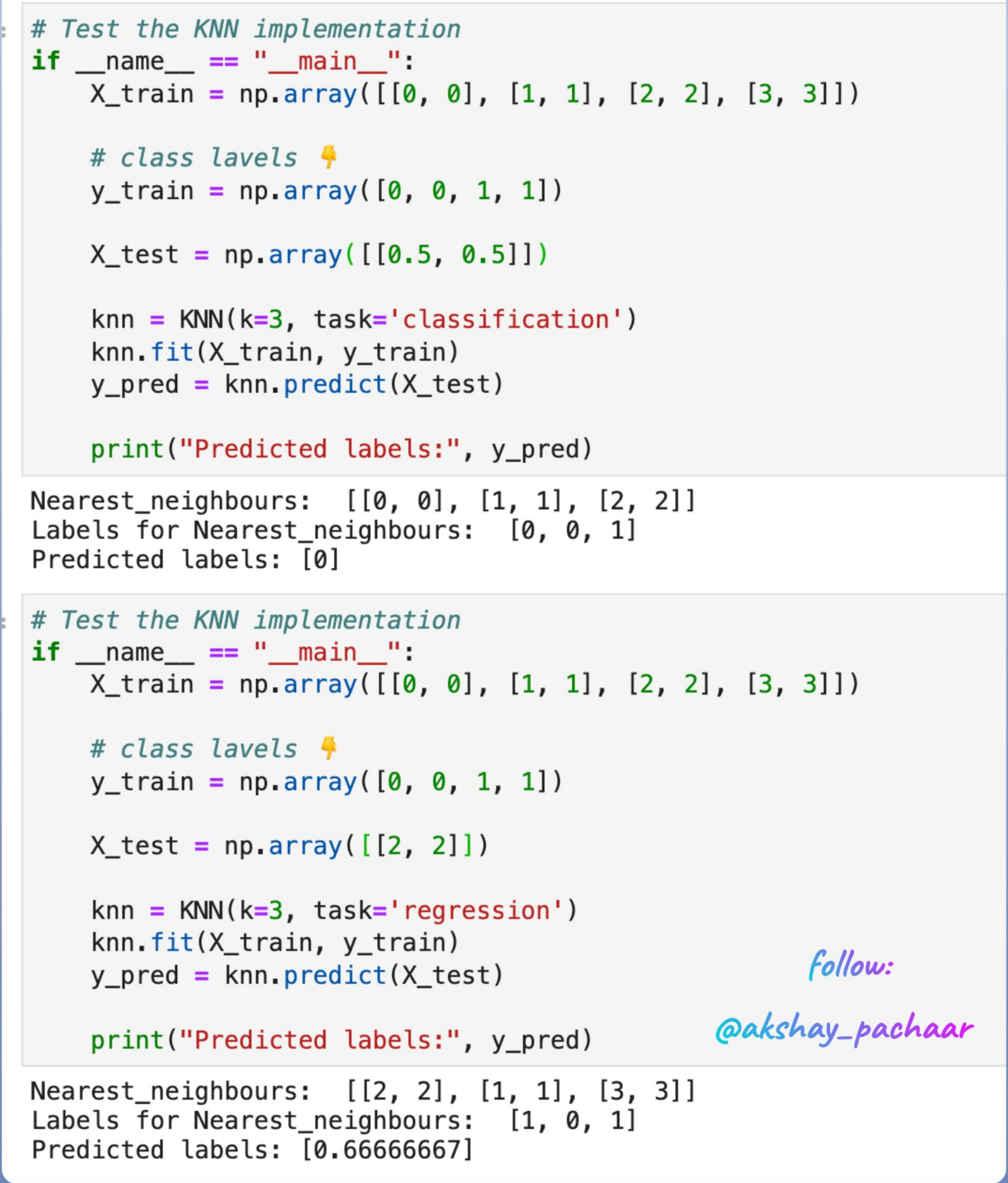
Let's test the above implementation with two examples:

- Classification
- Regression

Swipe 👉

@akshay_pachaar

Let's test it for regression & Classification 🚀



```
# Test the KNN implementation
if __name__ == "__main__":
    X_train = np.array([[0, 0], [1, 1], [2, 2], [3, 3]])

    # class lavels 👇
    y_train = np.array([0, 0, 1, 1])

    X_test = np.array([[0.5, 0.5]])

    knn = KNN(k=3, task='classification')
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)

    print("Predicted labels:", y_pred)
```

```
Nearest_neighbours:  [[0, 0], [1, 1], [2, 2]]
Labels for Nearest_neighbours:  [0, 0, 1]
Predicted labels: [0]
```

```
# Test the KNN implementation
if __name__ == "__main__":
    X_train = np.array([[0, 0], [1, 1], [2, 2], [3, 3]])

    # class lavels 👇
    y_train = np.array([0, 0, 1, 1])

    X_test = np.array([[2, 2]])

    knn = KNN(k=3, task='regression')
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)

    print("Predicted labels:", y_pred)
```

follow:

@akshay_pachaar

```
Nearest_neighbours:  [[2, 2], [1, 1], [3, 3]]
Labels for Nearest_neighbours:  [1, 0, 1]
Predicted labels: [0.66666667]
```

That's a wrap!

If you interested in:

- Python 🐍
- Data Science 📈
- Machine Learning 🤖
- MLOps 🛠️
- NLP 🗣️
- Computer Vision 🎥
- LLMs 🧠

Follow me on LinkedIn ✔
Everyday, I share tutorials on above topics!

Cheers!! 🙂

@akshay_pachaar