KNN NOTES BY SOHAM

Now lets start with KNN:

First and foremost it is a supervised learning algorithm that can perform both regression and classification.

K-Nearest Neighbors (KNN) Overview

K-Nearest Neighbors (KNN) is a simple, intuitive, and versatile algorithm used in machine learning for both classification and regression tasks. Itâ€™s a type of instance-based learning, where the model makes predictions based on the closest training examples in the feature space.

How KNN Works

1. Choose the Number of Neighbors (k):

- The number of neighbors k is a user-defined parameter. It determines how many neighbouring data points to consider when making a prediction.
- Common practice is to try several values of k and select the one that works best for your data.

2. Compute Distances:

- Calculate the distance between the query point (the data point you want to classify or predict) and all points in the training dataset.
- Common distance metrics include:

- Euclidean Distance: $d(p, q) = \sqrt{\sum (p_i - q_i)^2}$
- Manhattan Distance: $d(p, q) = \sum |p_i - q_i|$
- Minkowski Distance: A generalization of both Euclidean and Manhattan distances.

3. Identify Neighbors:

- Identify the k data points in the training set that are closest to the query point based on the computed distances.
-

4. Make a Prediction:

- **Classification:** For classification tasks, the output is the class that is most frequent among the k nearest neighbors (majority voting).
- **Regression:** For regression tasks, the output is the average of the values of the k nearest neighbors.

Example

KNN NOTES BY SOHAM

Imagine you have a dataset of animals with features like height and weight, and you want to classify a new animal as either a dog or a cat.

1. **Choose** k

   Suppose k=3
2. **Compute Distances**: Calculate the distance from the new animal to all other animals in the dataset.
3. **Identify Neighbors**: Find the 3 animals in the dataset that are closest to the new animal.
4. **Make a Prediction**: Look at the classes of these 3 neighbors. If 2 out of 3 are dogs and 1 is a cat, predict that the new animal is a dog.
5.

Advantages of KNN

1. **Simplicity**: KNN is easy to understand and implement.
2. **No Training Phase**: KNN is a lazy learner, meaning it doesn't involve any model training. It simply stores the training data and makes predictions by comparing the query point to the stored data.
3. **Versatility**: Can be used for both classification and regression tasks.

Improving KNN

1. **Feature Scaling**: Standardize or normalize the feature values so that all features contribute equally to the distance calculations.
2. **Dimensionality Reduction**: Use techniques like PCA (Principal Component Analysis) to reduce the number of features and mitigate the curse of dimensionality.
3. **Weighted KNN**: Assign weights to the neighbors based on their distance to the query point, giving closer neighbors more influence on the prediction.

SCIKIT LEARN CODE FOR KNN

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load dataset
data = load_iris()
X = data.data
y = data.target

# Split dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize KNN classifier with 5 neighbors
knn = KNeighborsClassifier(n_neighbors=5)

# Fit the model
knn.fit(X_train, y_train)

# Make predictions
y_pred = knn.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```