K-Nearest Neighbors (KNN) Algorithm in Machine Learning

Understanding KNN, Its Working, Importance, Applications & Implementation

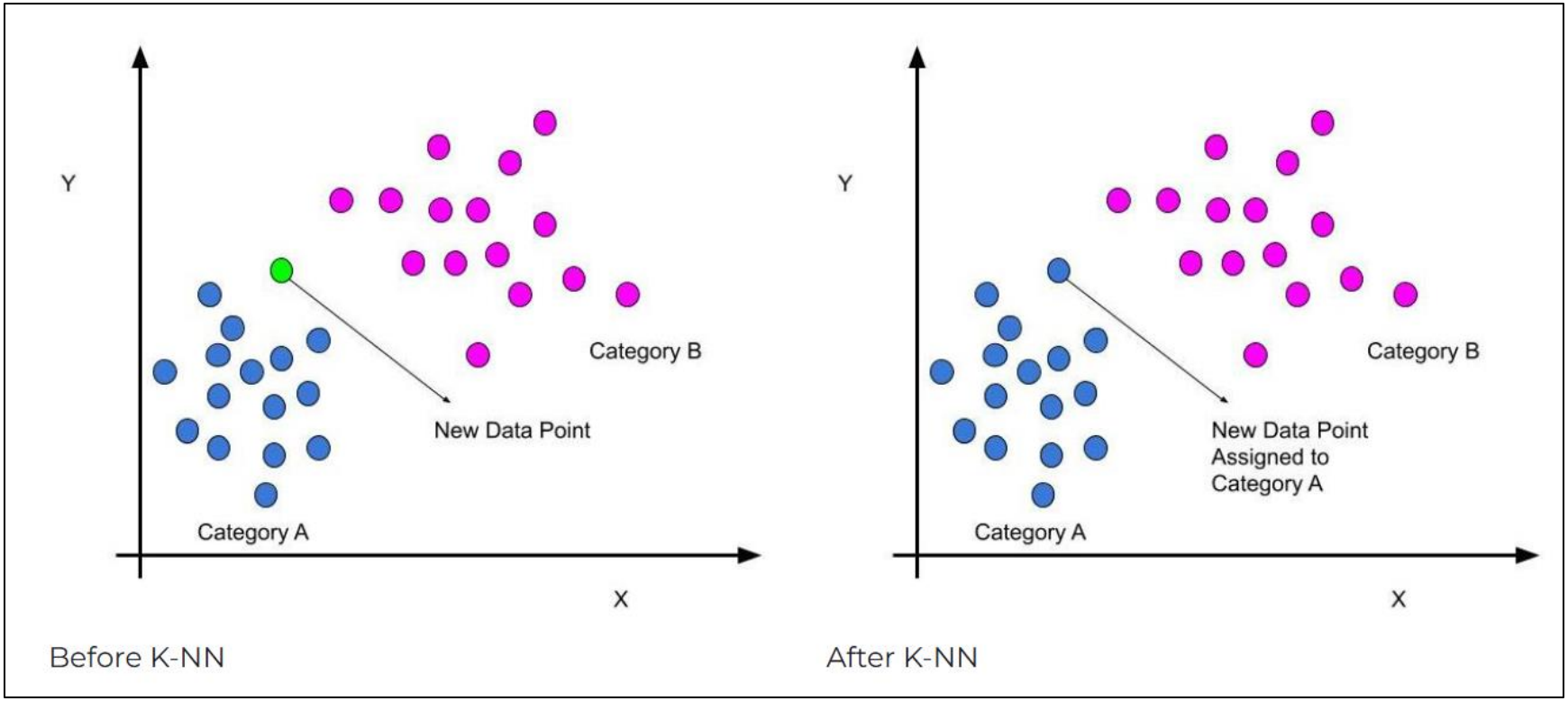
Introduction to KNN

- KNN is a supervised learning algorithm used for classification and regression.
- It is non-parametric and instance-based (lazy learning).
- It classifies new data based on similarity with existing labeled data.

How KNN Works?

1. Choose the number of neighbors (K).
2. Calculate the distance between the new data point and existing points.
3. Identify the K nearest neighbors.
4. For classification: Assign the majority class.
5. For regression: Compute the average of K nearest neighbors.

Before and After KNN



Distance Metrics in KNN

- Euclidean Distance (most common metric)
- Manhattan Distance
- Minkowski Distance
- Hamming Distance (for categorical data)

Choosing the Right K-Value

- Small K: More sensitive to noise, may cause overfitting.
- Large K: More generalized but may cause underfitting.
- Use cross-validation to find the optimal K.

Benefits of KNN

- Simple and easy to implement.
- No training phase (lazy learning).
- Works well for small datasets.
- Handles multi-class classification effectively.

Challenges of KNN

- Slow for large datasets (distance computation).
- Sensitive to irrelevant or unscaled features.
- Requires proper feature scaling (Normalization/Standardization).

Real-World Applications of KNN

- **Medical Diagnosis: Identifying diseases based on symptoms.**
- **Image Recognition: Handwritten digit classification.**
- **Recommendation Systems: Product recommendations.**
- **Anomaly Detection: Fraud detection in banking.**

Implementing KNN in Python

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

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

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train KNN model
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# Make predictions and evaluate
y_pred = knn.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
```

Summary

- KNN is a simple and powerful algorithm.
- Works well for small datasets but struggles with large datasets.
- Requires careful choice of K and feature scaling.
- Used in diverse real-world applications.

THANKYOU !!