Practical 15: Simple K-Nearest Neighbors (KNN) Algorithm

In [6]:

```
import numpy as np
import pandas as pd
from collections import Counter
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Fix SSL certificate issues if needed
import ssl
ssl._create_default_https_context = ssl._create_unverified_context

# Avoid warnings
import warnings
warnings.filterwarnings('ignore')
```

In [7]:

```
class SimpleKNN:
    def __init__(self, k=3):
        self.k = k
        self.X_train = None
        self.y_train = None

    def fit(self, X, y):
        """Store training data"""
        self.X_train = np.array(X)
        self.y_train = np.array(y)
        return self

    def predict(self, X):
        """Make predictions for all samples in X"""
        return np.array([self._predict_single(x) for x in X])

    def _predict_single(self, x):
        """Predict class for a single sample"""
        # Calculate distances between x and all training samples
        distances = [np.sqrt(np.sum((x - x_train)**2)) for x_train in self.X

        # Get indices of k nearest neighbors
        k_indices = np.argsort(distances)[:self.k]

        # Get most common class among k neighbors
        k_nearest_labels = self.y_train[k_indices]
        most_common = Counter(k_nearest_labels).most_common(1)
        return most_common[0][0]
```

In [8]:

```
# Load Iris dataset
iris = load_iris()
```

```python
X = iris.data
y = iris.target

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rar

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Print dataset info
print(f"Iris Dataset: {X.shape[0]} samples, {X.shape[1]} features")
print(f"Classes: {iris.target_names}")
print(f"Features: {iris.feature_names}")
```

```
Iris Dataset: 150 samples, 4 features
Classes: ['setosa' 'versicolor' 'virginica']
Features: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'pe
tal width (cm)']
```

In [9]:

```python
# Test with different k values
print("Results using our Simple KNN:")
print("-" * 30)
print(f"{'k':<5} {'Accuracy':<10}")
print("-" * 30)

for k in [1, 3, 5, 7]:
    # Create and train our model
    knn = SimpleKNN(k=k)
    knn.fit(X_train_scaled, y_train)

    # Make predictions
    y_pred = knn.predict(X_test_scaled)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f"{k:<5} {accuracy:.4f}")

print("-" * 30)
```

```
Results using our Simple KNN:
------------------------------
k     Accuracy
------------------------------
1     0.9778
3     1.0000
5     1.0000
7     1.0000
------------------------------
```

In [10]:

```python
# Test with scikit-learn's KNN
print("Results using Scikit-learn's KNN:")
print("-" * 30)
print(f"{'k':<5} {'Accuracy':<10}")
print("-" * 30)
```

```python
for k in [1, 3, 5, 7]:
    # Create and train sklearn model
    sklearn_knn = KNeighborsClassifier(n_neighbors=k)
    sklearn_knn.fit(X_train_scaled, y_train)

    # Make predictions
    y_pred = sklearn_knn.predict(X_test_scaled)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f"{k:<5} {accuracy:.4f}")
```

```
Results using Scikit-learn's KNN:
------------------------------
k     Accuracy
------------------------------
1     0.9778
3     1.0000
5     1.0000
7     1.0000
```

This notebook was converted with convert.ploomber.io