Experiment No. 6

Title: Implement K Nearest Neighbour (KNN) classification

Objectives:

1. To understand and implement the KNN classification algorithm.
2. To classify data points based on similarity to their nearest neighbors.
3. To evaluate the model using accuracy and other performance metrics.

Problem Statement:

Given a dataset with labeled instances, the objective is to classify new, unseen instances by identifying the KKK nearest neighbors and assigning the most common label among them. The classification should be performed efficiently, and the choice of KKK should be optimized to avoid underfitting or overfitting.

Outcomes:

1. A trained KNN model capable of classifying new data points.
2. Performance evaluation using metrics like accuracy.
3. Visualization of decision boundaries.

Tools Required: 4GB RAM, Anaconda, Notebook

Theory:

KNN is a non-parametric, instance-based learning algorithm. Given a data point:

1. Compute the distance between the new point and all existing labeled points in the dataset.
2. Select the KKK nearest points.
3. Assign the most common label among these neighbors to the new point.

Common distance metrics:

- Euclidean Distance:
  $d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$
- Manhattan Distance:

$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|$

- Minkowski Distance:

$d(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^p\right)^{1/p}$

The choice of KKK is crucial:

- Small KKK → Sensitive to noise, overfitting risk.
- Large KKK → Smoother decision boundary, potential underfitting.

Algorithm:

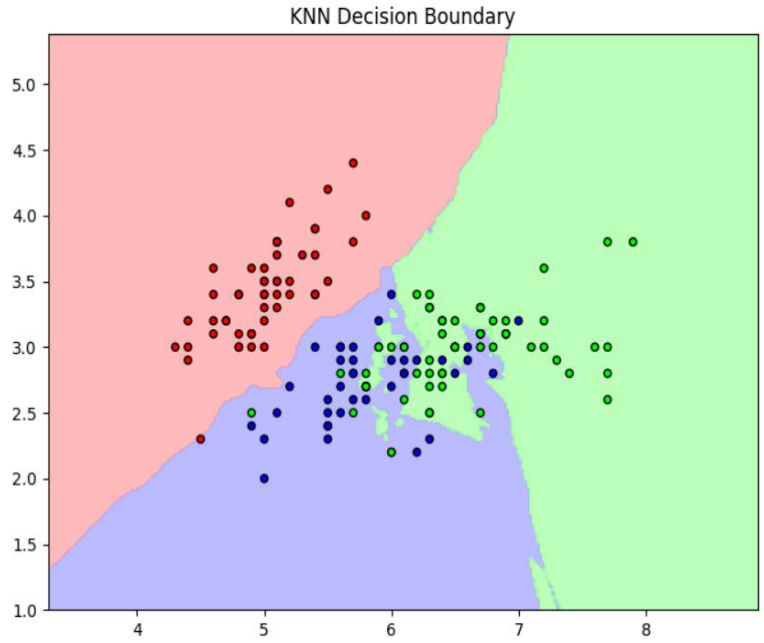
Steps:

1. Load and preprocess the dataset.
2. Choose the value of KKK.
3. Compute the distance between the test instance and all training instances.
4. Identify the KKK nearest neighbors.
5. Assign the most frequent class label among the neighbors.
6. Evaluate the model using a test set.

Source Code:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
data = load_iris()
X, y = data.data, data.target
X_reduced = X[:, :2]
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_reduced, y)
def plot_decision_boundary(X, y, model):
    cmap_light = ListedColormap(['#FFAAAA', '#AAAAFF', '#AAFFAA'])
    cmap_bold = ListedColormap(['#FF0000', '#0000FF', '#00FF00'])
    h = .02
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.figure(figsize=(8, 6))
    plt.contourf(xx, yy, Z, cmap=cmap_light, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold, edgecolor='k', s=20)
    plt.title("KNN Decision Boundary")
    plt.show()
plot_decision_boundary(X_reduced, y, knn)
```

Output:



Conclusion:

_____

_____

_____

_____