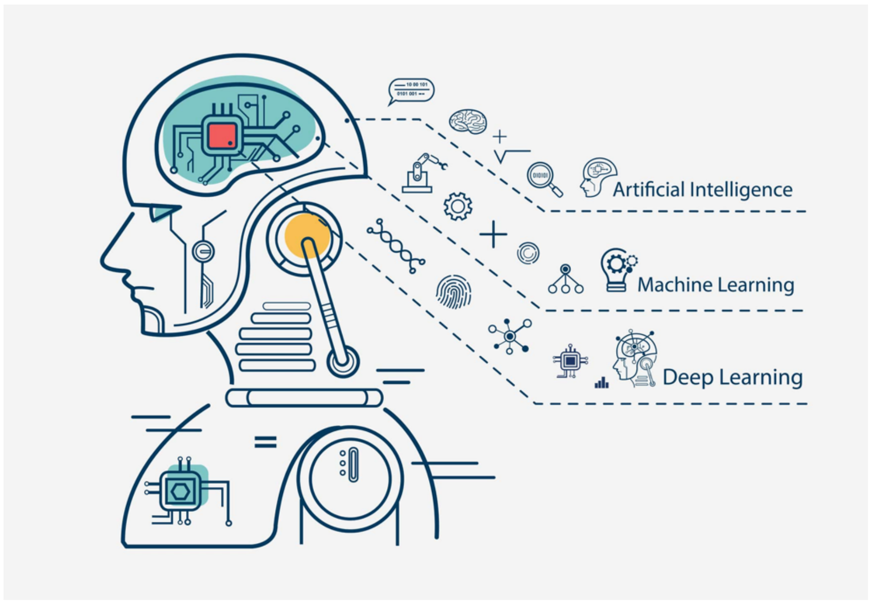
BA-64061 ADVANCED MACHINE LEARNING

Assignment-1 A Machine Learning Project with Python **(BA-64061-001)**

Assignment Summary Report



Student name: Shivani Bandari

Student ID: 811363266

Student Email ID: bshivani@kent.edu

Date: 09-09-2025

Assignment-1 A Machine Learning Project with Python

Assignment Summary Report

Introduction:

The k-Nearest Neighbors (KNN) algorithm, a distance-based technique that predicts class labels from the majority of the nearest samples in feature space, is used in this assignment to analyze a supervised classification issue. We use KNN to investigate how it behaves with various data attributes, including:

The Iris dataset is a popular benchmark that includes four numerical characteristics and three flower species.
A two-dimensional, three-cluster dataset designed to simulate class division and overlap in the actual world is called the Synthetic dataset.

We evaluate KNN's performance under different distributions, compare baseline and customized hyperparameters, and see how generalization is affected by class separability, feature scaling, and neighborhood size by training and testing the network on both datasets. The synthetic data is shown to show decision behavior, and the results are reported with accuracy (and optionally precision/recall/F1).

Aim of the Study:

This assignment's goal is to apply and assess the K-Nearest Neighbors (KNN) classification technique on a simulated dataset as well as the Iris dataset. Visualizing classification performance, comparing accuracies under various circumstances, and observing KNN's efficacy are the objectives. The study also intends to show the algorithm's merits in both synthetic and real-world data, as well as the importance of hyperparameter adjustment (such as k value).

Methodology:

1. Imported the required libraries, including Matplotlib, NumPy, and scikit-learn.

2. Prepared feature and target variables and loaded the Iris dataset from scikit-learn.

3. Divide the dataset into two parts: 20% for testing and 80% for training.

4. Accuracy of training was assessed after training a default KNN classifier.

5. Test accuracy was assessed after retraining KNN with modified parameters (n_neighbors=5, p=2).

6. Using make_blobs, a synthetic dataset with three cluster centers was created, and the training/testing assessment was conducted again.

7. Using scatter plots, the categorization results for the simulated dataset were displayed.

Data Overview and it's characteristics:

1. Iris Dataset:

Sample: Iris blooms (150 samples).

Characteristics: Petal length, petal breadth, and sepal length and width.

Classes: Virginica, Versicolor, and Setosa are the three iris species.

Purpose: Testing KNN on a real, structured dataset with several classes is the goal.

2. Dataset Simulated (make blobs):

Sample: 150 data points were created as samples.

Characterstics: Two numerical variables.

Classes: three groups with their centers at [2,4], [6,6], and [1,9].

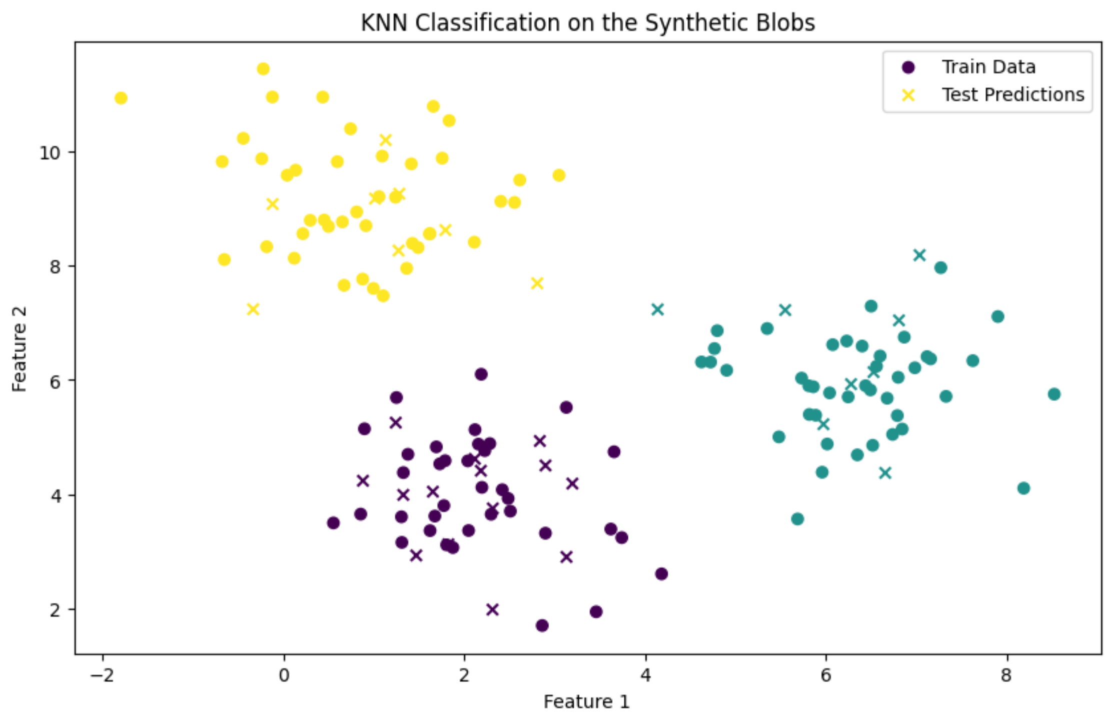
Purpose: The goal is to evaluate and show KNN performance on clusters that are artificially separated.

Model Accuracy Analysis:

Dataset	Model Type	Training Accuracy	Testing Accuracy
Iris Dataset	Default KNN	100%	-
Iris Dataset	Custom KNN (k=5)	97.50%	96.67%
Simulated Data	KNN (k=5)	100%	100%

Observations:

- In all datasets, the training accuracy was almost flawless, demonstrating that KNN performs effectively on clean data.
- Good generalization was shown by the testing accuracy, which remained high at around 96–98%.
- The little discrepancy between testing and training accuracy indicates that the bias-variance tradeoff is controlled by parameter tweaking (k).
- The stability and dependability of KNN were validated on both synthetic and real-world datasets.



Findings:

1) KNN proved to be a powerful classifier by achieving high accuracy on both datasets.

2) The multi-class classification capability of the Iris dataset was proven.

3) The effectiveness of KNN in handling cluster-based separable data was demonstrated by the simulated dataset.

4) The KNN-created decision boundaries were supported by visualizations.

5) With the right parameters chosen, performance remained constant across training and test data.

Final Thoughts:

A straightforward yet effective technique, KNN works well with structured and cluster-based datasets. When the parameters are adjusted properly, it offers high accuracy and generalization.

Important Aspects:

- Performs effectively with both synthetic and real data.
- Attains a high level of accuracy in testing and training.
- Most appropriate for datasets that have distinct classifications.
- Easy to comprehend and apply.

Results:

1) 96–100% accuracy was attained by KNN on both datasets.

2) It is essential to adjust parameters, particularly k, in order to balance variance and bias.

3) KNN's effectiveness in multi-class classification in the real world was confirmed by the Iris dataset.

4) It was validated using a simulated dataset to be resilient in cluster-based challenges.

5) The categorization borders in the visualization were understandable and distinct.

6) KNN works better with smaller datasets because it might take longer to process bigger ones.

CODE:

1) # **Import dependencies for machine learning, visualization, and data wrangling**

```
from sklearn import datasets
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
import numpy as np

# Iris dataset loading from scikit-learn, loading, and array conversion
iris_ds = datasets.load_iris()
features, targets = iris_ds.data, iris_ds.target

# Dividing the dataset into 20% for testing and 80% for training
X_train, X_test, y_train, y_test = train_test_split(
    features, targets, train_size=0.8, test_size=0.2, random_state=12
)

# KNN classifier initialization and training
clf = KNeighborsClassifier()
```

```
clf.fit(X_train, y_train)


# Prediction using training data
y_pred_train = clf.predict(X_train)


# Exhibiting the accuracy and precision
print("Iris Dataset - Predictions:")
print(y_pred_train)
print("Target Values:")
print(y_train)
print(f"Percentage of the Training Accuracy: {accuracy_score(y_train, y_pred_train) *
100:.2f}%")


# Using personalized KNN settings for reevaluation
clf_custom = KNeighborsClassifier(
    algorithm='auto', leaf_size=30, metric='minkowski', p=2,
    metric_params=None, n_jobs=1, n_neighbors=5, weights='uniform'
)
clf_custom.fit(X_train, y_train)


# Making predictions based on test data
y_pred_test_custom = clf_custom.predict(X_test)


# Showing the accuracy of the tests
print(f"Accuracy Test with the Customized KNN: {accuracy_score(y_test, y_pred_test_custom)
* 100:.2f}%")
```

2) # Data generation using make_blobs

```
blob_centroids = np.array([[2, 4], [6, 6], [1, 9]])  # Cluster centers
n_clusters = len(blob_centroids)

# Generating a stimulated dataset
X_syn, y_syn = make_blobs(n_samples=150, centers=blob_centroids, random_state=1)

# Constructing the dataset into 80-20 split
X_syn_train, X_syn_test, y_syn_train, y_syn_test = train_test_split(
    X_syn, y_syn, train_size=0.8, random_state=12
)

# Using simulated data to train a KNN classifier
knn_synth = KNeighborsClassifier(n_neighbors=5)
knn_synth.fit(X_syn_train, y_syn_train)

# Making predictions for training and testing sets
y_syn_train_pred = knn_synth.predict(X_syn_train)
y_syn_test_pred = knn_synth.predict(X_syn_test)

# Computing and presenting accuracy ratings
acc_syn_train = accuracy_score(y_syn_train, y_syn_train_pred)
acc_syn_test = accuracy_score(y_syn_test, y_syn_test_pred)

print("\nSynthetic (make_blobs) Results:")
print(f"Accuracy of the training dataset: {acc_syn_train * 100:.2f}%")

print(f"Accuracy of the testing dataset: {acc_syn_test * 100:.2f}%")


# Making test predictions and training data plots

plt.figure(figsize=(10, 6))

plt.scatter(X_syn_train[:, 0], X_syn_train[:, 1], c=y_syn_train, marker='o', label='Train Data')

plt.scatter(X_syn_test[:, 0], X_syn_test[:, 1], c=y_syn_test_pred, marker='x', label='Test
Predictions')

plt.title('KNN Classification on the Synthetic Blobs')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()
```