Experiment/Practical 5 K-Nearest Neighbors (KNN)

Title: Implementation of Implementation of K-Nearest Neighbors (KNN) for Classification

Aim: To apply the K-Nearest Neighbors (KNN) algorithm for classification and to understand its mechanism and performance.

Objective: Students will learn

- Implementation of the KNN classification algorithm on the given dataset(s).
- To evaluate and understand the effect of different values of K (the number of neighbors) on model performance.
- To visualize and interpret the results effectively.

Problem statement

Use the given dataset(s) to demonstrate the application of K-Nearest Neighbors (KNN) classification. The task is to identify the class of the query instance using k-nearest neighbors.

Explanation/Stepwise Procedure/ Algorithm:

- Give a brief description of K-Nearest Neighbours

  K-Nearest Neighbours (KNN) is a simple, non-parametric classification algorithm used in machine learning. It classifies a data point based on the majority class among its 'k' closest neighbors in the feature space, typically measured via distance metrics like Euclidean distance. KNN can also be used for regression tasks by averaging the values of the nearest neighbors. It's intuitive and easy to implement but can be computationally expensive, especially with large datasets.

- Give mathematical formulation of K-Nearest Neighbours

  The fundamental principle behind k-NN assumes that similar data points exist in close proximity to each other. Here is a step-by-step breakdown of how the algorithm operates:

  Data Representation: Assume we have a dataset with features (attributes) and corresponding labels (for supervised learning tasks).

  Distance Calculation: For a given data point (or query point) that needs to be classified or predicted, the algorithm calculates the distance between this point and every other point in the dataset. The most common distance metrics used are Euclidean distance, Manhattan distance, minkowski distance.

$$Manhattan\ Distance = \sum_{i=1}^{d} |x_{1i} - x_{2i}|$$

$$Euclidean\ Distance = \left(\sum_{i=1}^{d} (x_{1i} - x_{2i})^2\right)^{\frac{1}{2}}$$

$$Minkowski\ Distance = \left(\sum_{i=1}^{d} |x_{1i} - x_{2i}|^p\right)^{\frac{1}{p}}$$

3. Finding Neighbors: Once distances are calculated, the algorithm identifies the k nearest neighbors to the query point.

4. Classification or Regression:

For classification, the algorithm assigns the query point to the class most common among its k nearest neighbors (using majority voting).

For regression, the algorithm predicts the average of the k nearest neighbors' target values.

5. Choosing the Value of k: The value of k is a crucial parameter (hyperparameter) in k-NN. A smaller value of k leads to more complex models (potentially overfitting), whereas a larger value of k makes the model simpler but may miss local patterns. Choosing the right value of k often involves experimentation and validation techniques like cross-validation.

- Write the importance of K-Nearest Neighbours

1. Simplicity: Easy to understand and implement; intuitive mechanism based on proximity.

2. Versatility: Applicable for both classification and regression tasks.

3. Non-parametric: Does not assume any underlying data distribution, useful for complex patterns.

4. No Training Phase: Instantly usable without a separate training step, storing the dataset for predictions.

5. Dynamic Adaptability: Can easily include new data points without needing retraining.

6. Good for Small Datasets: Performs well on smaller datasets, reducing the risk of overfitting.

7. Multi-class Support: Naturally handles multiclass classification problems without alterations.

8. Flexible Distance Metrics: Allows customization of distance calculations (e.g., Euclidean, Manhattan).

9. Baseline Model: Provides a straightforward benchmark for comparing more complex algorithms.

10. Computational Considerations: Sensitive to larger datasets in terms of computation and memory; must consider dimensionality issues (curse of dimensionality).

- Mention applications of K-Nearest Neighbours.

K-Nearest Neighbours (KNN) has a wide range of applications across various domains. Here are some notable applications:

1. Image Recognition: KNN can be used for classifying images based on their features, such as in facial recognition and object detection.

2. Recommendation Systems: KNN helps in recommending products or services by finding similar users or items based on preferences and behaviors.

3. Medical Diagnosis: In healthcare, KNN can aid in diagnosing diseases by comparing patient symptoms and medical history with those of existing cases.

4. Customer Segmentation: Businesses can group customers based on purchasing behavior for targeted marketing and personalized services.

5. Anomaly Detection: KNN can identify unusual data points by measuring their distance from typical cases, useful in fraud detection and network security.

6. Text Classification: In natural language processing, KNN can classify documents or categorize sentiments based on text features.

7. Financial Applications: KNN can be employed for credit scoring, evaluating the likelihood of loan defaults by comparing borrowers with similar profiles.

8. Handwriting Recognition: KNN is used to classify handwritten characters or digits based on feature extraction from images.

- Brief explanation of performance metrics (e.g., accuracy, precision, recall, f1-score, confusion matrix, cross-validation)

Performance Metrics in Machine Learning

1. Accuracy
   Accuracy measures the proportion of correctly classified instances out of the total instances. It is useful when the dataset is balanced but can be misleading if the classes are imbalanced.

   Formula:

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

2. Precision
   Precision evaluates how many of the predicted positive instances were actually positive. It is important in scenarios where false positives are costly.

   Formula:

$$Precision = \frac{TP}{TP + FP}$$

3. Recall (Sensitivity or True Positive Rate)
   Recall measures how many actual positive instances were correctly predicted. It is crucial when missing a positive instance is costly, such as in medical diagnoses.

   Formula:

$$Recall = \frac{TP}{TP + FN}$$

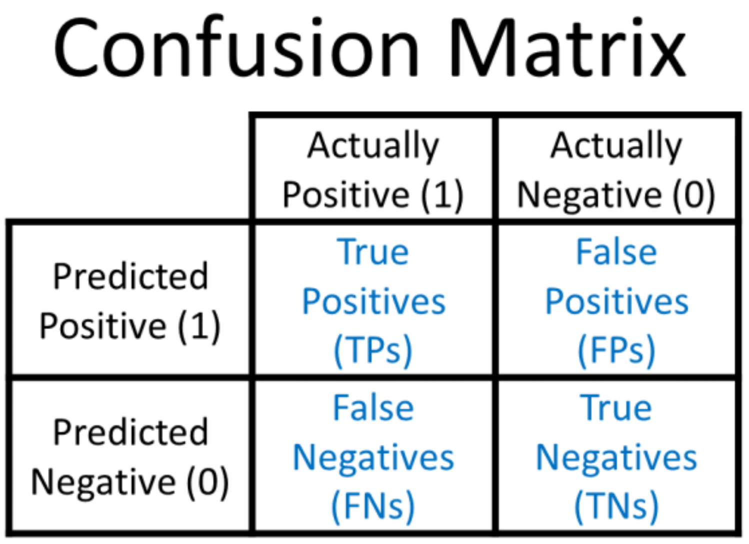
4. F1-Score
   The F1-score is the harmonic mean of precision and recall, providing a balanced measure when there is an uneven class distribution.

   Formula:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

5. Confusion Matrix
   A confusion matrix is a table used to evaluate the performance of a classification model by showing the counts of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

   Actual Positive - Predicted Positive (TP), Predicted Negative (FN)
   Actual Negative - Predicted Positive (FP), Predicted Negative (TN)

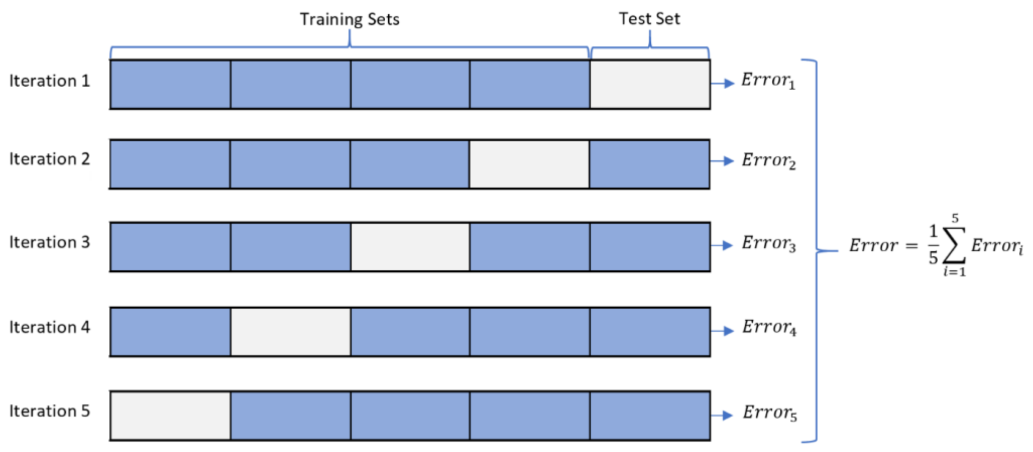


Confusion Matrix

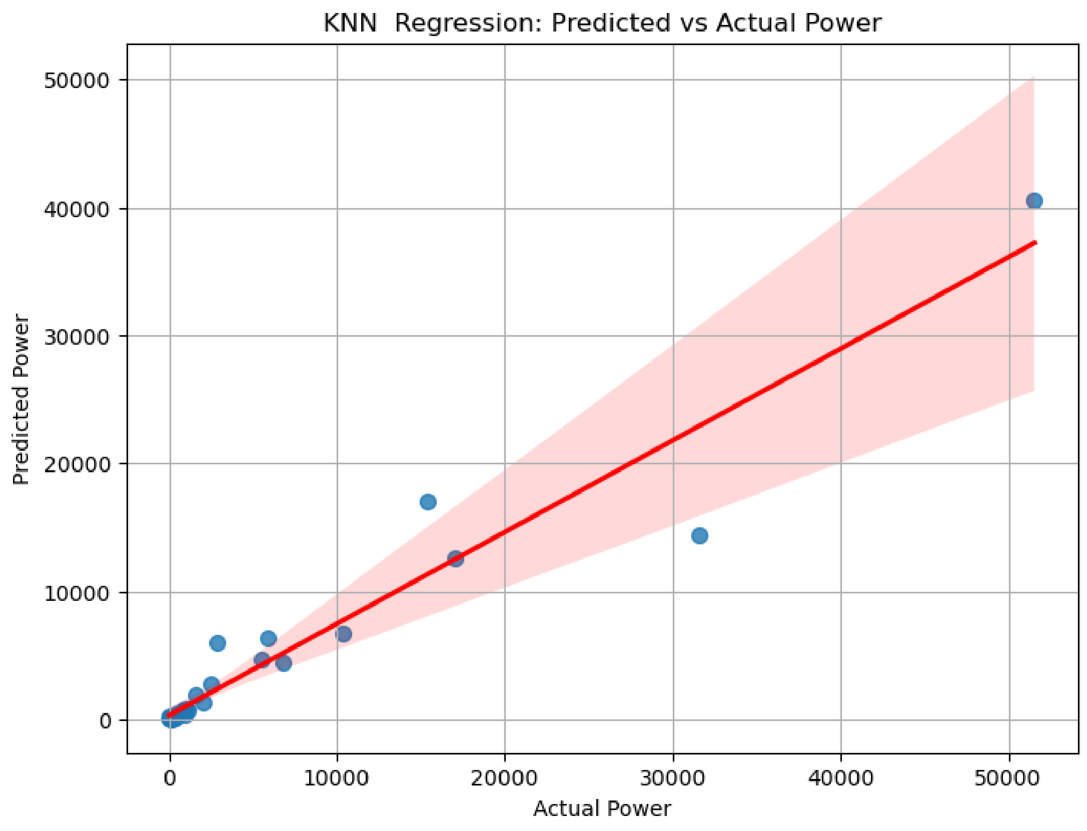
	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

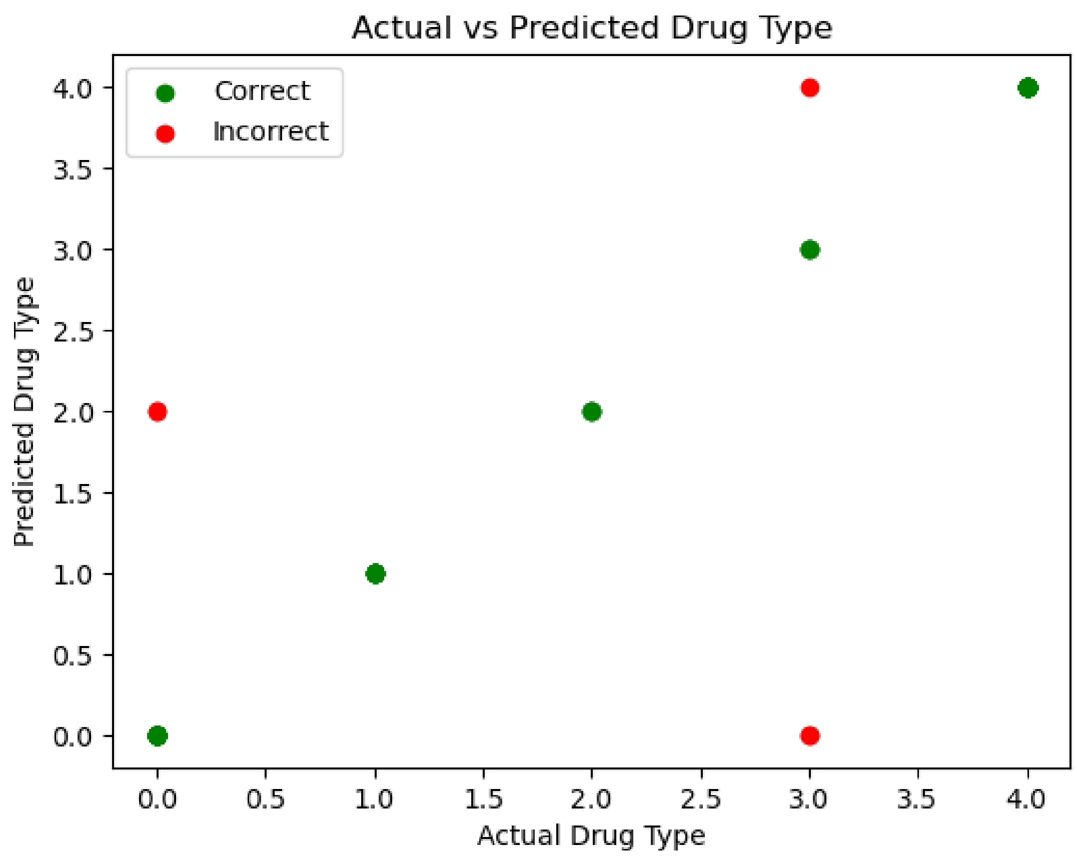
6. Cross-Validation
   Cross-validation is a technique for assessing the performance of a model by splitting the dataset into multiple training and testing subsets. The most common method is k-fold cross-validation, where the dataset is divided into k subsets, and the model is trained k times, each time using a different subset as the test set.

   This technique helps in reducing overfitting and provides a more reliable measure of a model's generalization ability.



- ***Add necessary figure(s)/Diagram(s)***





Input & Output:

Input:

- Dataset: A labeled dataset with features and a target

  Practice dataset 1 KNN regression Hydropower_Consumption.csv

  Practice dataset 2 KNNClassifier drug_classification.csv

- User Input: The number of neighbors K can be selected for optimal performance.

Output:

- Predictions: The predicted class labels for test data.
- Model Evaluation Metrics: Accuracy, confusion matrix, and other metrics as needed.
- Visualizations: Plots showing decision boundaries, confusion matrix, and performance for different values of K.

Conclusion:

Impact of K:

Impact of K in KNN for Drug Classification

In the drug classification study, the choice of K significantly impacts the model’s accuracy and generalization. A small K (e.g., K=1) results in a highly sensitive model prone to overfitting, as it closely follows the training data and is affected by noise. On the other hand, a large K (e.g., K=21, as selected through GridSearchCV) helps in smoothing decision boundaries and reducing variance while maintaining a good classification performance. The best model achieved an accuracy of 91.67% using the Manhattan distance metric and weighted voting.

Impact of K in KNN for Hydropower Consumption Prediction

For KNN regression in hydropower consumption prediction, a small K (e.g., K=3) captures local fluctuations well but is susceptible to noise, leading to higher variance. A larger K (e.g., K=12, as selected in the study) smooths predictions by considering more neighbors, reducing sensitivity to outliers while slightly increasing bias. The optimal model achieved an R^2 score of 0.87, balancing bias and variance effectively.