# Implementation and Evaluation of the K-Nearest Neighbors Algorithm

## 1. Introduction

The K-Nearest Neighbors (k-NN) algorithm is a simple, non-parametric, and lazy learning method widely used for classification and regression. It classifies a new data point by identifying the `k` nearest neighbors among the training data based on a distance metric and assigning the majority class label of those neighbors to the test point.

The primary objectives of this assignment are:

1. To implement the k-NN algorithm from scratch without relying on external libraries.
2. To preprocess and classify data from the `PlayTennis` dataset using k-NN.
3. To evaluate the algorithm's performance using leave-one-out cross-validation and analyze the results.

---

## 2. Methodology

### Data Preparation

The `PlayTennis` dataset includes weather-related features (`Outlook`, `Temperature`, `Humidity`, `Wind`) and a target label (`PlayTennis`) that indicates whether tennis was played on a given day. An additional `Day` column was present in the dataset but was removed as it was irrelevant to the classification task.

Categorical features (`Outlook`, `Temperature`, etc.) were one-hot encoded to prepare them for distance-based computations. For example: - The `Outlook` feature was transformed into `Outlook_Overcast`, `Outlook_Rain`, and `Outlook_Sunny`. - This encoding ensures that each category is represented as binary values (0 or 1), making them compatible with distance metrics.

### Implementation of the k-NN Classifier

**Preprocessing** The `_preprocess_data` method converts categorical features in the training data into one-hot encoded vectors. Similarly, `_preprocess_test_data` ensures that test data is aligned with the training data's encoding structure by adding missing columns with default values.

**Distance Metrics** Two distance metrics were implemented:

1. **Manhattan Distance**: Manhattan(x,y) = abs(x1 - y1) + abs(x2 - y2) + abs(x3 - y3) ... abs(xi - yi)

2. **Euclidean Distance**: Euclidean(x,y) = ((x1 - y1)^2 + (x2 - y2)^2 + (x3 - y3)^2 + ... + (xi - yi)^2) ^ (1/2)

These metrics determine the "closeness" of a test point to its neighbors.

**Prediction** For a given test point:

1. The distances to all training points are computed.
2. The `k` nearest neighbors are selected based on these distances.
3. The majority class label among the neighbors is assigned to the test point.

**Cross-Validation** Leave-one-out cross-validation was employed to evaluate the algorithm. For each iteration, one instance was used as the test data while the rest formed the training set. The process was repeated for all instances, and the results were aggregated.

**Challenges**

1. **Handling Categorical Data**: One-hot encoding was used to convert categorical features into binary vectors. Ensuring alignment between training and test data was crucial.
2. **Tie-Breaking for Even `k`**: The algorithm enforces the use of odd values for `k` to prevent ties during classification.

---

## 3. Results

### Confusion Matrix and Accuracy for Different Values of k

The performance of the k-NN classifier was evaluated using leave-one-out cross-validation with different values of `k`. Below are the results:

**k = 1**

| Metric | Value |
|---|---|
| True Positives (TP) | 3 |
| True Negatives (TN) | 3 |
| False Positives (FP) | 2 |
| False Negatives (FN) | 6 |
| **Accuracy** | 0.43 |

**k = 3**

| Metric | Value |
|---|---|
| True Positives (TP) | 4 |
| True Negatives (TN) | 3 |
| False Positives (FP) | 2 |
| False Negatives (FN) | 5 |
| **Accuracy** | 0.50 |

**k = 5**

| Metric | Value |
|---|---|
| True Positives (TP) | 5 |
| True Negatives (TN) | 1 |
| False Positives (FP) | 4 |
| False Negatives (FN) | 4 |
| **Accuracy** | 0.43 |

**k = 7**

| Metric | Value |
|---|---|
| True Positives (TP) | 9 |
| True Negatives (TN) | 0 |
| False Positives (FP) | 5 |
| False Negatives (FN) | 0 |
| **Accuracy** | 0.64 |

---

## 4. Discussion

### Analysis of Results

The results show that the accuracy of the k-NN classifier depends heavily on the choice of `k`. Smaller values like `k=1` result in low accuracy (43%) due to sensitivity to noise and outliers, while larger values like `k=7` achieve higher accuracy (64%) by reducing noise impact but introducing bias toward the majority class.

### Reasons for Misclassifications

1. **Feature Overlap**: Several data points shared similar feature values, leading to ambiguous classifications.
2. **Bias towards Yes**: There are many more yes class attributes in the target column.
3. **Small Dataset**: The small size of the dataset limited the model's ability to generalize.

**Limitations**

1. The performance is sensitive to the choice of `k` and the distance metric.
2. The one-hot encoding approach may not always capture the relationships between categorical variables effectively.

---

## 5. Conclusion

The k-NN classifier was implemented and evaluated on the PlayTennis dataset, achieving a highest accuracy of 64% with k=7. The results highlighted the trade-offs between small and large k values, with larger values reducing sensitivity to outliers but introducing bias toward the majority class. The project demonstrated the importance of preprocessing, such as one-hot encoding, and the challenges of working with small datasets.