Hasan Pekedis 150120068

Implementation of a Simple k-Nearest Neighbor (k-NN) Classifier in Python

1. Introduction

K-Nearest Neighbors (k-NN) Algorithm:

The K-Nearest Neighbors (k-NN) algorithm is a simple, yet powerful, supervised machine learning technique primarily used for classification and regression tasks. The algorithm works by identifying the k closest training examples (neighbors) to a test instance and making predictions based on the majority class (for classification) or the average value (for regression) of these neighbors. In the k-NN algorithm, distance metrics such as Euclidean distance or Manhattan distance are commonly used to measure how similar or different two instances are.

k-NN is often preferred for its simplicity and effectiveness, particularly when the decision boundaries between classes are not well-defined and the dataset is not very large. However, it can become computationally expensive as the dataset grows in size due to the need to compute distances between the test instance and every training instance.

Objectives of the Assignment

The primary objective of this assignment is to implement and evaluate a K-Nearest Neighbors (k-NN) classifier to predict the target variable PlayTennis based on weather-related features. These features include Outlook, Temperature, Humidity, and Wind. By applying k-NN, we aim to understand how the algorithm works and how its performance can be influenced by various factors, such as the choice of distance metric and the value of the parameter k.

2. Methodology

Data Preparation

To prepare the dataset for the K-Nearest Neighbors (k-NN) classification task, the following steps were taken:

1. Loading the Data: The dataset, which contains information about weather conditions and whether tennis was played (PlayTennis), was loaded from a JSON file into a pandas DataFrame.
2. Handling Categorical Data: The dataset included categorical features such as Outlook, Temperature, Humidity, and Wind. These features were converted into numeric values using one-hot encoding.

3. Splitting the Data: The dataset was split into two parts: the training data and the test data. The training data was used to build the k-NN model, while the test data was used for evaluating the model's performance.
4. Normalization: Since the k-NN algorithm relies on distance calculations between instances, it is essential to ensure that all features are on a similar scale.

Implementation of K-Nearest Neighbors (k-NN) Classifier

The K-Nearest Neighbors classifier was implemented using the following steps:

1. Distance Calculation: The first step in the k-NN algorithm is to compute the distance between the test instance and each training instance. The Euclidean distance and Manhattan distance formulas used for this part.
2. k-NN Prediction: For each test instance, the k-NN classifier calculates the distances between the test instance and all instances in the training data. The algorithm then selects the k closest instances and assigns the predicted label based on a majority vote from these neighbors. The label that appears most frequently among the k nearest neighbors is the predicted label for the test instance. If there is a tie in the votes, the classifier returns the label of the closest neighbor.
3. Evaluation: The classifier was evaluated using Leave-One-Out Cross-Validation (LOOCV). In LOOCV, each instance in the dataset is used as a test case, while the remaining instances are used as the training data. This process ensures that every instance in the dataset is tested, providing a comprehensive evaluation of the model's performance. The accuracy was calculated as the ratio of correct predictions to the total number of instances.

Handling Challenges

During the implementation of the k-NN classifier, a few challenges were encountered and addressed:

1. Handling Categorical Features: Categorical features in the dataset, such as Outlook and Wind, posed a challenge because the k-NN algorithm requires numerical input.
2. Zero Probabilities in k-NN: A common issue in k-NN, especially in small datasets, is the possibility of zero probability situations. This can occur if the majority class among the k nearest neighbors is the same class as the test instance. However, since k-NN relies on distance, it is also possible that the neighbors are far apart, leading to less reliable predictions.
3. Distance Metric Selection: The selection of the appropriate distance metric (Euclidean vs. Manhattan) is crucial in k-NN. While the Euclidean distance is commonly used and effective in many cases, the Manhattan distance was also implemented to compare its performance. The choice of distance metric can significantly impact the model's

predictions, especially when the feature space has varying scales or when the relationship between features is more linear (as opposed to radial in the case of Euclidean distance).

3. Results

The performance of the K-Nearest Neighbors (k-NN) classifier was evaluated using a confusion matrix and accuracy metric. Below is a detailed explanation of the evaluation results:

Confusion Matrix

The confusion matrix for the k-NN classifier is presented as follows:

TP: 0, FP: 0

FN: 8, TN: 6

Accuracy: 0.42857142857142855

Analysis of Results

The accuracy of 42.86% indicates that the model is performing poorly. While the model does correctly predict some instances where tennis is not played (True Negatives), it fails to predict instances where tennis is actually played (True Positives are 0).

4. Discussion

Analysis of the Results

The results indicate that the k-Nearest Neighbors (k-NN) classifier did not perform well on the dataset, achieving an accuracy of only 42.86%. This suggests that the model is not effectively capturing the relationships in the data to distinguish between instances where tennis is played (positive class) and where it is not played (negative class).

- The True Positives (TP) count is zero, meaning the model never correctly predicted that tennis would be played.
- The True Negatives (TN) count of 6 indicates that the model was able to correctly identify some instances where tennis was not played.

- However, the False Negatives (FN) count of 8 is concerning, as it suggests that the model frequently failed to identify instances where tennis was actually played, misclassifying them as negative cases.

Possible Reasons for Misclassifications

1. **Choice of Features**:
    - The dataset includes categorical features such as 'Outlook', 'Temperature', 'Humidity', and 'Wind', which were one-hot encoded. While one-hot encoding is a valid technique, it may not always capture the relationships between categories well, especially when there are interactions between features that are not well-represented in this encoding.
2. **Distance Metric (Euclidean and Manhattan)**:
    - The **Euclidean distance** used for calculating the distance between instances may not be the most appropriate for categorical data. Euclidean distance is more suited for continuous numerical data, as it assumes that differences in the values of features are meaningful and can be directly subtracted.
3. **Choice of k**:
    - The choice of **k** (the number of neighbors) can significantly impact the performance of the k-NN classifier. If the value of k is too low, the model may be too sensitive to outliers or noise in the data, leading to overfitting and poor generalization.
4. **Imbalanced Classes**:
    - If the dataset is imbalanced, where one class (e.g., 'PlayTennis' = No) has more instances than the other class ('PlayTennis' = Yes), the model may be biased towards predicting the majority class.

Limitations of the Implementation

1. **Limited Feature Engineering**:
    - The implementation relies solely on the basic categorical features without much exploration of additional feature engineering techniques.
2. **Use of One-Hot Encoding**:
    - While one-hot encoding is a common technique for categorical variables, it may not always be the most efficient way to represent categorical data, especially when there are many categories.
3. **Distance Calculation Limitations**:
    - As mentioned earlier, the use of Euclidean and Manhattan distances may not be ideal for categorical data.
4. **Small Dataset Size**:
    - The dataset used for training the model may be too small to fully capture the complexity of the problem. Machine learning algorithms like k-NN generally perform better with larger datasets.

5. Conclusion

In this project, I implemented a k-Nearest Neighbors (k-NN) classifier to predict whether tennis would be played based on weather conditions. The model achieved an accuracy of 42.86%, with no true positives, indicating poor performance. The results highlighted several key issues, including the choice of features, the distance metrics used, and the imbalanced dataset. We learned that k-NN's effectiveness heavily relies on appropriate data preprocessing, feature selection, and tuning hyperparameters. To improve the model, exploring better distance metrics, handling imbalanced classes, and optimizing the model parameters are necessary steps.

6. References:

- StatQuest: K-nearest neighbors, Clearly Explained Youtube Video - https://www.youtube.com/watch?v=HVXime0nQeI&t=147s