Marmara University - Faculty of Engineering

Department of Computer Engineering

Intro to Machine Learning (Fall 2024)

Submit Date: 01/12/2024.

K-Nearest-Neighborhood-Classifier

Student Number (ID)	Name	Surname
150120998	Abdelrahman	Zahran

Sections Of the Report: -

1. Introduction

The K-Nearest Neighbor (k-NN) algorithm is a fundamental instance-based learning technique in machine learning. It classifies a given instance by examining the labels of its k-nearest neighbors in the feature space. The algorithm relies on a distance metric to measure similarity, making it versatile and applicable to various data types. However, its simplicity also makes it sensitive to the choice of k, the size of the dataset, and the quality of feature representation.

This assignment aimed to implement the k-NN algorithm from scratch using Python, applying it to the "Play Tennis" dataset, a widely used toy dataset in machine learning. Key objectives were:

- To gain hands-on experience with implementing machine learning algorithms without relying on pre-built libraries for classification.
- To understand and analyze the impact of data preprocessing, distance metrics, and hyperparameters on the performance of the k-NN classifier.
- To evaluate the classifier's performance using rigorous testing methods such as Leave-One-Out Cross-Validation (LOOCV).

2. Methodology

2.1 Data Preparation

The "Play Tennis" dataset consists of 14 instances, each representing weather conditions and a target variable indicating whether tennis was played. The dataset includes the following features:

2. Dataset Table

The "Play Tennis" dataset contains 14 instances of weather conditions and a target variable ("PlayTennis") indicating whether tennis was played. It is commonly used for illustrative purposes in machine learning due to its simplicity and mix of categorical features.

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

- **Outlook**: Sunny, Overcast, Rain
- **Temperature**: Hot, Mild, Cool
- **Humidity**: High, Normal
- **Wind**: Weak, Strong
- **Play Tennis (Target)**: Yes, No

Steps in Data Preparation:

1. **Loading the Dataset**:
   - The dataset was provided in JSON format and loaded using Python's json module into a Pandas DataFrame.
2. **Encoding Categorical Features**:
   - Features such as "Outlook" and "Temperature" were transformed into numeric values using one-hot encoding to make them suitable for numerical computations.

   - The target variable, "Play Tennis," was encoded as 1 for "Yes" and 0 for "No."

3. **Feature Normalization**:
   - To ensure fair comparison across features, all feature values were normalized to the range [0, 1]. This step eliminated biases arising from differing value ranges, such as "Temperature" and "Wind."
4. **Exploration and Summary**:
   - The dataset was displayed in a tabular format, and basic statistics, such as class distributions, were computed.

2.2 Implementation Details

The k-NN classifier was implemented with the following capabilities:

1. **Distance Metrics**:
   - **Euclidean Distance**: Measures the straight-line distance between two points in the feature space.
   - **Manhattan Distance**: Calculates the sum of absolute differences across dimensions.
   - **Cosine Similarity**: Measures the cosine of the angle between two vectors, assessing their similarity.
2. **Training Phase**:
   - As a lazy learning algorithm, the training phase of k-NN simply involved storing the dataset in memory. No explicit model was built.
3. **Classification Phase**:
   - For a given test instance, distances to all training instances were computed using the selected metric.
   - Neighbors were sorted by distance, and the majority class among the top k neighbors determined the predicted label.
4. **Evaluation**:
   - Two modes of evaluation were implemented:
     - **LOOCV**: Each instance was tested independently, with all other instances used for training.
     - **Standard Evaluation**: The entire dataset was used for both training and testing.

2.3 Handling Challenges

1. **Zero Probabilities in Cosine Similarity**:
   - Instances with zero magnitudes were assigned a default similarity value to avoid division by zero.
2. **Small Dataset Size**:
   - LOOCV was employed to maximize data utility, as the dataset contained only 14 instances.
3. **Hyperparameter Sensitivity**:
   - The value of k was taken as input from the user. For this assignment, k=3 was chosen for evaluation, balancing simplicity and performance.

3. Results – For (k = 3)

3.1 Leave-One-Out Cross-Validation (LOOCV)

Overall Accuracy: 57%

The LOOCV method rigorously tested the classifier's ability to generalize. Each instance was classified independently, with all other instances serving as the training set. This evaluation revealed challenges in handling overlapping feature values.

Metric	***Value***
Accuracy	0.57
Precision	0.64
Recall	0.78
F1 Score	0.70

Confusion Matrix

	Predicted Yes	***Predicted No***
Actual Yes	7	2
Actual No	4	1

3.2 Standard Evaluation

Overall Accuracy: 79%

In this mode, the classifier was evaluated on the same dataset used for training. While this approach demonstrated higher accuracy, it lacked the rigor of testing on unseen data.

Metric	***Value***
Accuracy	0.79
Precision	0.75
Recall	1.00
F1 Score	0.86

Confusion Matrix

	Predicted Yes	***Predicted No***
Actual Yes	9	0
Actual No	3	2

---

4. Discussion

4.1 Analysis of Results

- **LOOCV**:
  - Highlighted the classifier's difficulty with instances near decision boundaries.
  - The relatively low precision (0.64) indicated overprediction of positive labels.
- **Standard Evaluation**:
  - Achieved perfect recall, reflecting the classifier's ability to correctly predict all positive instances.
  - The inflated accuracy (79%) underscored the importance of evaluating on unseen data.

4.2 Misclassification Insights

- **Overlapping Features**:
    - Instances with similar feature values but different labels (e.g., "Sunny" weather) caused confusion.
- **Class Imbalance**:
    - Slight imbalance in the dataset (9 positive vs. 5 negative instances) influenced predictions.

4.3 Limitations

1. **Small Dataset**:
    - The dataset's small size made it challenging to establish robust decision boundaries.
2. **Sensitivity to Hyperparameters**:
    - The value of k and the choice of distance metric significantly impacted performance.
3. **Computational Cost**:
    - As k-NN computes distances to all training instances, its computational cost grows with dataset size.

---

5. Conclusion

This assignment demonstrated the implementation and evaluation of the k-NN algorithm from scratch. Key findings included:

- The importance of data preprocessing (e.g., encoding and normalization) in ensuring reliable distance calculations.
- The influence of hyperparameters such as k and distance metrics on classifier performance.
- The need for larger datasets and rigorous evaluation methods to assess generalization capabilities accurately.

The implementation provided valuable insights into the strengths and limitations of instance-based learning, laying the foundation for exploring more advanced algorithms.

---

6. References

- Lecture Notes on k-NN Algorithm.
- Ethem Alpaydin, *Introduction to Machine Learning*.
- Python documentation for NumPy and pandas.
