CSE4288

Introduction to Machine Learning

K-Nearest Neighbor Classifier

Mohamad Nael Ayoubi

150120997

• Introduction

The k-Nearest Neighbor (k-NN) Algorithm: The k-Nearest Neighbor (k-NN) algorithm is a simple, yet powerful supervised learning method widely used for classification and regression tasks. It is a lazy learner, meaning it does not build an explicit model during the training phase. Instead, it stores the training data and makes predictions based on the majority class of the k closest neighbors in feature space. All instances correspond to points in the n-D space. The closeness is determined by a distance metric, such as Euclidean or Manhattan distance.

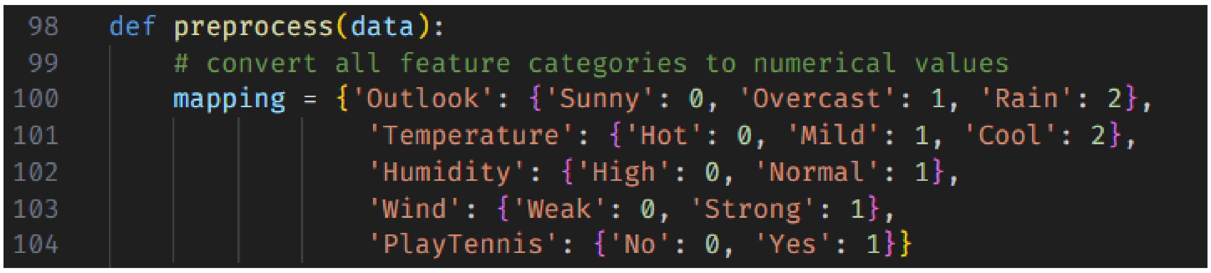
The objective of this assignment is to understand and implement the k-NN algorithm from scratch without using machine learning libraries like scikit-learn. The tasks include:

1. Preparing the Play Tennis dataset.
2. Implementing the k-NN algorithm for classification.
3. Evaluating its performance using metrics such as accuracy and a confusion matrix.
4. Analyzing the results and identifying any challenges during implementation.

• Methodology

1. Data Preparation:

The Play Tennis dataset consists of categorical features (e.g., Outlook, Temperature, Humidity, Wind) and a target variable (PlayTennis), which were converted to numeric form using label encoding. As shown below:



```python
def preprocess(data):
    # convert all feature categories to numerical values
    mapping = {'Outlook': {'Sunny': 0, 'Overcast': 1, 'Rain': 2},
               'Temperature': {'Hot': 0, 'Mild': 1, 'Cool': 2},
               'Humidity': {'High': 0, 'Normal': 1},
               'Wind': {'Weak': 0, 'Strong': 1},
               'PlayTennis': {'No': 0, 'Yes': 1}}
```

The dataset was saved in JSON format (dataset.json).

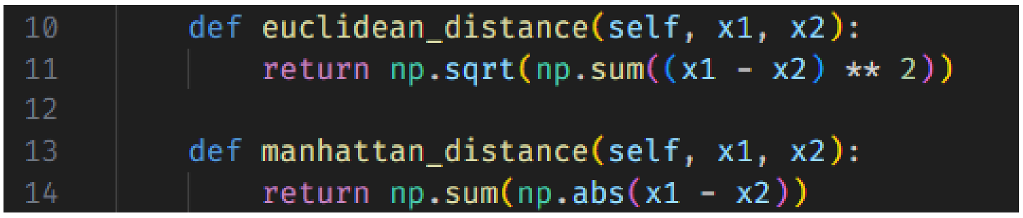
2. Implementation Details:

The kNN_classifier class was developed with:

- Support for both Euclidean and Manhattan distance metrics.
- Hyperparameter k, allowing user input to control the number of neighbors.
- Predictions based on majority voting among the k nearest neighbors.

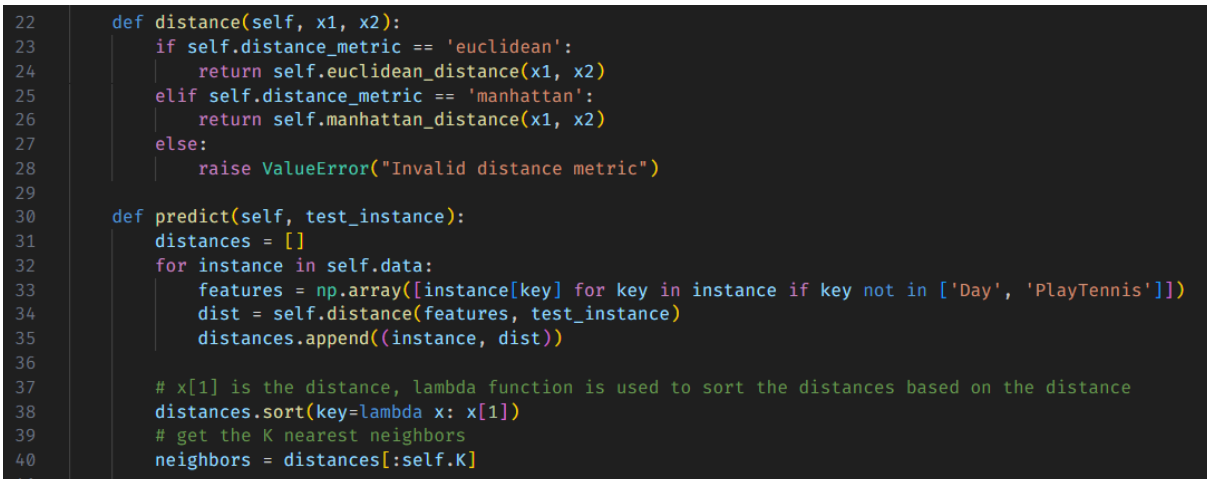
Lastly, its performance was evaluated using accuracy and Confusion Matrix.

How distance and predictions has been calculated is shown below:



```python
    def euclidean_distance(self, x1, x2):
        return np.sqrt(np.sum((x1 - x2) ** 2))

    def manhattan_distance(self, x1, x2):
        return np.sum(np.abs(x1 - x2))
```



```python
    def distance(self, x1, x2):
        if self.distance_metric == 'euclidean':
            return self.euclidean_distance(x1, x2)
        elif self.distance_metric == 'manhattan':
            return self.manhattan_distance(x1, x2)
        else:
            raise ValueError("Invalid distance metric")

    def predict(self, test_instance):
        distances = []
        for instance in self.data:
            features = np.array([instance[key] for key in instance if key not in ['Day', 'PlayTennis']])
            dist = self.distance(features, test_instance)
            distances.append((instance, dist))

        # x[1] is the distance, lambda function is used to sort the distances based on the distance
        distances.sort(key=lambda x: x[1])
        # get the K nearest neighbors
        neighbors = distances[:self.K]
```

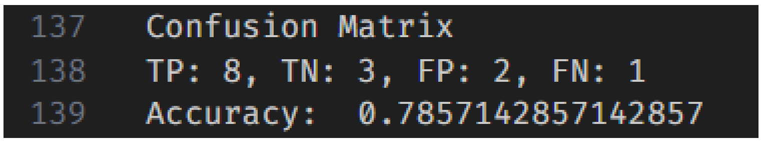
3. Challenges:

Categorical-to-numeric conversion required careful preprocessing to avoid misalignment between features.

- **Results**

For K = 3 and Euclidean distance picked as a metric I got the following results using the same dataset for testing.

- Accuracy of 0.79 is achieved
- Confusion matrix is shown below



```
Confusion Matrix
TP: 8, TN: 3, FP: 2, FN: 1
Accuracy:  0.7857142857142857
```

- **Discussion**

1. Analysis of Results:
    - The high accuracy indicates that the k-NN algorithm performed well on the Play Tennis dataset, likely due to the small size and clear separability of the data.
    - Misclassifications (FP and FN) could arise from: Insufficient neighbor information (k value too small). Simplistic distance metrics not capturing nuanced patterns in the data.
2. Limitations of the Implementation:
    - Scalability: The algorithm processes every data point in the training set for each prediction, making it inefficient for large datasets
    - Categorical Feature Encoding: Label encoding may not capture the ordinal relationships between categories effectively.

- **Conclusion**

The k-NN algorithm's simplicity makes it a good starting point for understanding instance-based learning and lazy classifiers. Proper preprocessing and careful parameter tuning (such as choosing the right value of k) significantly impact the algorithm's performance.

Overfitting and Underfitting in k-NN: Overfitting: When k is set to a very low value (e.g., k=1), the model may overfit to the training data. It becomes highly sensitive to noise and specific data points, leading to poor generalization on unseen data. This happens because the prediction is based on a single closest neighbor, which might not represent the overall distribution. Underfitting: Conversely, when k is too large, the model may underfit. It considers too many neighbors, potentially including data points that are not relevant to the test instance.

- **References**

[1] Classification Basic slides given in lecture.

[2] Introduction to Machine Learning by Ethem Alpaydin