Middle East Technical University | Department of Computer Engineering

CENG 414

Introduction to Data Mining

Spring' 2023-2024
Take-Home Exam 2

Emre Külah
kulah@ceng.metu.edu.tr
Due date: Apr 5, 2024, Friday, 23:59

kNN Applications

1 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple yet effective algorithm used for classification and regression tasks in machine learning. The main idea behind KNN is to predict the class or value of a data point by looking at its 'k' nearest neighbors in the feature space. In classification tasks, the class label of the majority of the nearest neighbors is assigned to the data point being predicted. In regression tasks, the average or weighted average of the values of the 'k' nearest neighbors is taken as the prediction.

The choice of 'k' determines the number of neighbors to consider when making predictions. A smaller 'k' value makes the model more sensitive to noise, while a larger 'k' value makes it smoother but may miss local patterns. KNN is a non-parametric method, meaning it does not make assumptions about the underlying distribution of the data.

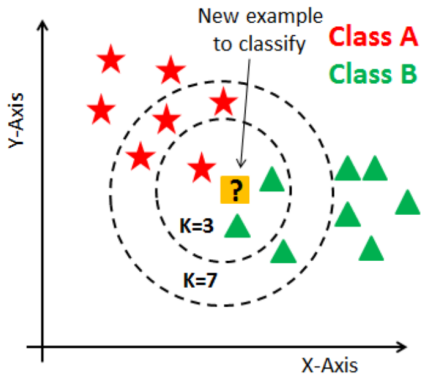

Figure 1: Illustration of the K-Nearest Neighbors (KNN) algorithm.

2 Tasks

2.1 kNN Implementation

In this part of the exam, you are required to implement a k-nearest neighbors (KNN) algorithm. KNN is a simple, yet powerful, algorithm used for classification and regression tasks. It works by finding the 'k' nearest neighbors of a given data point in the feature space and using their labels to make predictions.

2.1.1 Parameters

The following parameters will be used in test cases with different combinations:

- **Distances:** Specifies the distance metric to be used for calculating distances between data points. You can choose from Euclidean, Manhattan, and Chebyshev distances.
- **Re-training:** A boolean value indicating whether re-training will be enabled. Re-training involves incorporating a new test sample into the training set with its labeled classification, ensuring it influences the classification of future samples.

- **k:** An integer value specifying the number of nearest neighbors to consider when making predictions. It should be within the range [3, 10].
- **Distance Threshold:** A floating-point value indicating the distance threshold. If provided, only data points within this threshold will be considered as neighbors. If not provided (None), all data points will be considered.
- **Weighted Voting:** A boolean value indicating whether to use weighted voting when making predictions. If set to True, closer neighbors will have a greater influence on the prediction.

$$w = \frac{1}{d(x_q, x_i)^2} \tag{1}$$

where x is the position, q is the query and i is one of the samples.

2.1.2 Function Signature

```
knn(
    existing_data: pd.DataFrame, test_data: pd.DataFrame,
    k: int, distance_method: str, re_training: bool,
    distance_threshold: float, weighted_voting: bool
)

Return: Return the predictions as an iterable, preferably in the form of a Series object.
```

2.2 Missing Feature Prediction

In this part, you are given a test set with labels, but one feature is missing for each sample. Your task is to predict the missing feature value for each sample such that it corresponds to the given label. You should avoid using brute force algorithms for finding appropriate missing values.

2.2.1 Parameters

The following parameters will be used in test cases with different combinations:

- **Distances:** Specifies the distance metric to be used for calculating distances between data points. You can choose from Euclidean and Manhattan distances.
- **k:** An integer value specifying the number of nearest neighbors to consider when making predictions. It should be within the range [3, 10].
- **Distance Threshold:** A floating-point value indicating the distance threshold. If provided, only data points within this threshold will be considered as neighbors. If not provided (None), all data points will be considered.
- **Weighted Voting:** A boolean value indicating whether to use weighted voting when making predictions. If set to True, closer neighbors will have a greater influence on the prediction.

$$w = \frac{1}{d(x_q, x_i)^2} \tag{2}$$

where x is the position, q is the query and i is one of the samples.

2.2.2 Example

In the dataframe below, each sample is characterized by three features, but one feature is missing for each sample. However, their corresponding labels are provided. Your algorithm's task is to determine suitable values for these missing features so that they match the given labels.

Feature 1	Feature 2	Feature 3	Label
2.5	NaN	4.0	0
NaN	3.0	5.0	1
1.0	2.0	NaN	0

To evaluate your algorithm, the following steps will be taken:

- Execute your algorithm to determine values for the missing features.
- Remove the labels from the dataframe.
- Employ our own KNN implementation to predict the labels.
- Compare the predicted labels with the actual labels.

2.2.3 Function Signature

```
fill_missing_features(
    existing_data: pd.DataFrame, test_data: pd.DataFrame,
    k: int, distance_method: str, distance_threshold: float, weighted_voting: bool
)

Return the filled DataFrame of the provided test set.
```

Note: In this task, avoid employing brute force algorithms for finding appropriate values for missing features.

Note: Re-training will not be utilized in this part. Consequently, your outputs will solely impact the respective sample.

2.3 Dataset

The dataset comprises 30 numeric features and 1 label. The labels are binary, represented by 0 and 1. The training set consists of 398 samples, while the test set contains 85 samples. These files are provided in OdtuClass as "train.csv" for the training set, "test.csv" for the regular test set, and "test_with_missing.csv" for the version of the test dataset with randomly removed features for each sample. While testing your implementations, omit the labels for the first task, and for the second task, utilize the "test_with_missing.csv" file.

	label	f1	f2	...	f28	f29	f30
count	398.000000	398.000000	398.000000	...	398.000000	398.000000	398.000000
mean	0.351759	14.064324	19.346935	...	0.112582	0.290708	0.083921
std	0.478120	3.538799	4.274047	...	0.066433	0.063718	0.018293
min	0.000000	6.981000	10.380000	...	0.000000	0.156500	0.055040
25%	0.000000	11.622500	16.330000	...	0.063532	0.250425	0.071827
50%	0.000000	13.270000	18.895000	...	0.096655	0.283100	0.080075
75%	1.000000	15.740000	21.715000	...	0.156375	0.319525	0.092053
max	1.000000	27.220000	39.280000	...	0.291000	0.663800	0.207500

Figure 2: Description of the training set using Pandas.

2.4 Submission

Your submission should include a Python file named "hw2.py". This file must contain the following two functions:

- knn
- fill_missing_features

The provided Python file will be imported into a test script, where your functions will be executed with predefined test cases. Use given parameter names in the task details, do not change function signature.

3 Regulations

- **Programming Language:** Python
- **Packages:**
  You are only allowed to use the following packages: *"pandas"* and *"numpy"*.
- **Submission:**
  You will submit your homework via the OdtuClass system. **Late submissions are strictly not allowed**; the deadline will not be extended under any circumstances, so **please refrain from requesting any deadline extensions**.
- **Evaluation:** Your code will be evaluated using several test datasets. To ensure consistency, please refrain from shuffling the provided test datasets, as this may lead to different outputs for re-training examples.
- **Cheating: We have a zero-tolerance policy for cheating**. Any individuals found to be engaged in cheating will be subject to punishment in accordance with university regulations, and their work will receive a score of 0. Sharing code between students or utilizing third-party code is strictly prohibited. Even if you use only a "part" of code from another source, this still constitutes cheating. Please be aware that sophisticated tools exist to detect similarities between codes. Therefore, attempting to modify code obtained from elsewhere will not prevent detection.