High Level K-Nearest Neighbors(HL-KNN): Modified Version

JULIA UMMA REDDY   
Computer Science, CalState University- EastBAy  
Hayward, USA  
jummareddy@horizon.csueastbasy.edu

***Abstract*— High Level K Nearest Neighbors (HL-KNN) machine learning model for classification analysis was introduced in September 2023 by Dr. Elife Ozturk Kiyak, Dr. Bita Ghasemkhani and Dr. Derya Birant. This algorithm is based on the K-Nearest Algorithm(KNN) and addresses the noise problem that has been known to reduce classification prediction accuracy in KNN. HL-KNN considers two sets of neighbors for a given test sample. First is the low-level neighborhood and the second is the high-level neighborhood. Low-level neighborhoods’ are the direct k neighbors(N) of the test sample and high-level neighborhoods’ are the K neighbors of each of these N neighbors. Then classifies the test samples’ class based on the total number of classes of both the levels combined. This paper proposes modified versions of HL-KNN. There are two versions : First is that the algorithm doesn’t get the high-level neighbors if all the low-level neighbors have the same class. Second is that the algorithm gets high-level neighbors for only those low-level neighbors that have the class with maximum count. This approach reduces the number of times the euclidean distance is calculated for each of the sample tests. And has shown improvement in time taken by the algorithm with same(or almost close) accuracy predicted by HL-KNN. Tested the new version algorithm with different size data sets[7].**

***Keywords—K-Nearest Neighbors, Euclidean Distance***

# Introduction

Machine Learning(ML) is a domain that introduces prediction of future events based on available historical data. Not only prediction but detection of anomalies for abnormal events and also clustering unsupervised data.

There are many classification algorithms in ML that have major impact in data analysis and K-NN has been one of the reputed algorithms for accurate prediction of data class.HL-KNN[1] is another algorithm that was introduced in 2023 as an improvement of KNN. The algorithm proposed in this paper is based on KNN and HL-KNN algorithms.

The proposed method in this paper attempts to improve the HL-KNN mainly due to its advantages such as its effectiveness and easy implementation. The benefits also include ease of interpretation of the results. Also it predicts both discrete and categorical classification, it is best used for both regression and classification problems. Further, it supports independence of data distribution and incremental data easily using local neighborhoods and no pre-training of new data.

The main contributions of the proposed algorithm are:

1. A machine learning method that is an extension of HL-KNN
2. Increase the performance of the algorithm by decreasing the computation time (the number of times euclideans distance calculations performed) in predicting the classification.
3. The experimental results show the improvement for several datasets in comparison to the HL-KNN algorithm.

# Related work

## Review of KNN and HL-KNN

Both KNN and HL-KNN[1] have noteworthy performance in accuracy of predicting class of given data. In KNN, the K value is considered as the square root of the number of test samples. And the classification of the test sample is based on the maximum number of neighbors with a particular class. However, its classification accuracy is affected when there are noise samples near the test data. This is where HL-KNN comes into play. HL-KNN has been successful in predicting more accurately even with the noise samples around the test data by considering k-neighbors of each of the k-nearest neighbors of the test sample. This approach addresses the noise problem very well.

There are other models of k-NNl like k-min-max-sum (k-MMS)[5] that calculate the remoteness of a neighbors with each of the other neighbors and k-Nearest Centroid Neighbors(k-NCN)[6] where all the centroid neighbors are calculated and then the class is given base on voting of majority of neighbors around that centroid. But the proposed algorithm is solely looking at k nearest neighbors of the given test data.

The proposed method in this paper is an improvement of HL-KNN attempting to reduce the computation time of the algorithm given the same hardware and software specification for both the algorithms.

## Review of noise removal methods based on ENN-Rule

There have been many editing methods that attempt to remove the noise samples from the dataset. Wilson’s Edited Nearest Neighbor has been the most well known algorithm for this purpose. It considers all the misclassified data as noise and removes them. The other various variants of ENN[2](All-KNN, RENN, ENNProb, ENNth)[3][4] have worked on removing the noise samples by iterating through the training set with different k values, iterating through until the majority of the k neighbors have the same class, editing based on probability estimations etc.

But the proposed method in this paper does not remove

the noise samples but improves the time consumption of the HL-KNN algorithm by reducing the computation steps..

# Materials and Methods

This section explains the HL-KNN modified method. The figure1 shows the general structure of the proposed method. The first stage is data collection and data cleaning. Here data from various sources is collected and cleaned up. Cleaning the data is checking if there are any missing values and replacing them with average calculated values.

The second stage is data preprocessing, where the distribution of the data and the dependance of each of the features on the classification is studied. corr0 is a method that calculates the dependencies of each of the features of the dataset with the classification.

The third stage is the machine learning stage, here the working of the proposed algorithm is discussed.

1. Get the k-nearest neighbors (Low-Level neighborhood) of the test data by calculating the Euclidean distance of the test data with each of the training data.
2. Checked if all the k-neighbors (N) have the same class.
3. If yes, then don't get k-neighbors for any of the neighbors in N and classify the test data the same as the class of neighbors in N.
4. But if all the neighbors in N do not have the same class then get the majority of neighbors class.
5. Get k-nearest neighbors for the neighbors (High Level neighborhood) with majority class only.
6. Classify the test sample based on the class of the majority of neighbors in the high level neighborhood. (Voting system)

The k nearest neighbors for each test data are calculated as below:

1. For each of the test data, calculate the distance of that test data from each of the training data.
2. Then sort the distances calculated in ascending order
3. Consider the top 7 distances as the distances of k-nearest neighbors
4. The distances are calculated using Euclidean distance

*sqrt(sum(square(training features-test data features))) (1)*

The fourth stage shows the accuracy and time efficiency of the predicted class of the test samples. Accuracy is calculated by dividing the total number of correctly labeled samples by the total number of test samples and multiplied by 100.

(*Correctly Labeled / Total test samples)\* 100 (2)*

Then the time efficiency is calculated in seconds by starting a time-start function before calling the proposed algorithm and calling the time-stop function soon after the algorithm is invoked.

*Finish Time - Start Time (3)*

## Formal Description

Let Dtest be a test dataset and x an element of Dtest.

Nlow-level is the set of low-level neighborhoods’ and Nhigh-level is the set of high-level neighborhoods’

Low-Level Neighborhood: For a given test sample, k-nearest neighbors are selected based on the euclidean’s distance. HL-KNN takes K to be 7. Hence k is considered as 7 in this paper as well.

High-Level Neighborhood: For all the k-nearest neighbors, consider only the ones that are with the majority class as S. Get the k-nearest neighbors for these S data.

No high-level neighborhood is computed if all the low-level neighbors have the same class.

Algorithm:

Begin

for each x in DTest do

get k-Nearest neighbors Nlow-level

if all neighbors in Nlow-level have

the same class, then class(x) =

class(Nlow-level)

else

get k-nearest neighbor Nhigh-level of each of (Nlow-level) with majority

class

end if-else

end for-each

end

## Hardware and Software specifications:

## C. Pictorial Explanation

This section has four figures, which cover all possible cases with noise in the training data set.

k is considered as 3 in all the four figures for the sake of simplicity in pictorial representation. X in the figures is the test sample.

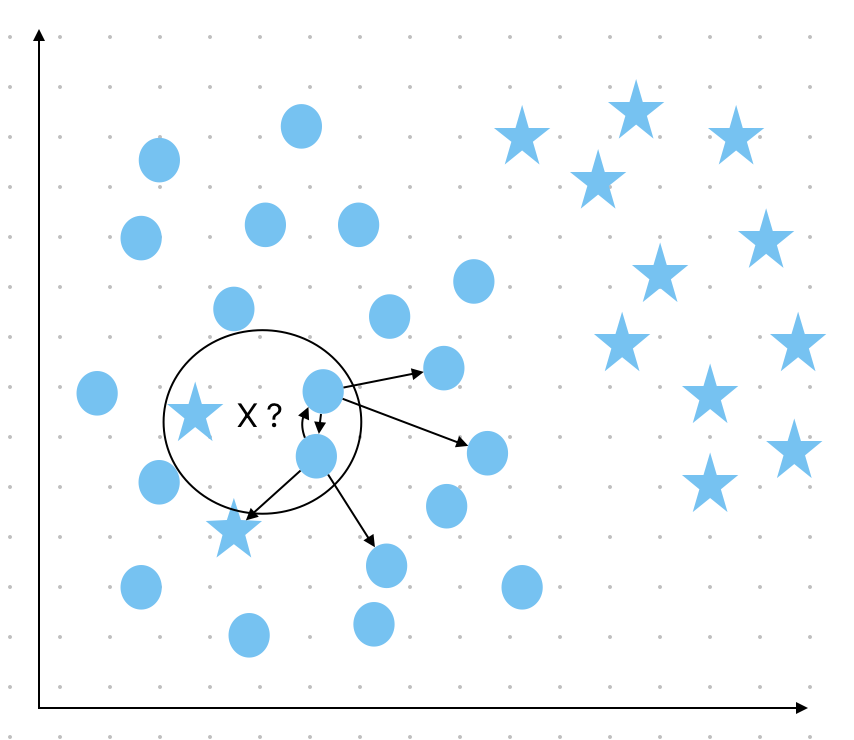
The black circle represents the low-level neighborhood and the dotted circle represents the high-level neighborhood.

In Figure1, the 3 nearest neighbors for X are shown inside a black circle. The ‘blue circles’ are more than the ‘stars’. Getting neighbors of each of the blue circles shows X will be classified as a ‘blue circle’.

Similarly in Figure 2, though the stars are maximum in the low-level neighborhood, the k-neighbors of each of the stars are blue-circles. Hence X will be classified as a circle.

Figure 3 shows a border line condition where X will be classified based on majority in the low-level neighborhood class.

Figure 4 shows all the neighbors in the low-level are of the same class, and according to the proposed algorithm in this paper, the high-level neighbors are not calculated for such a case. Hence X will be classified as a circle.

 Figure 1

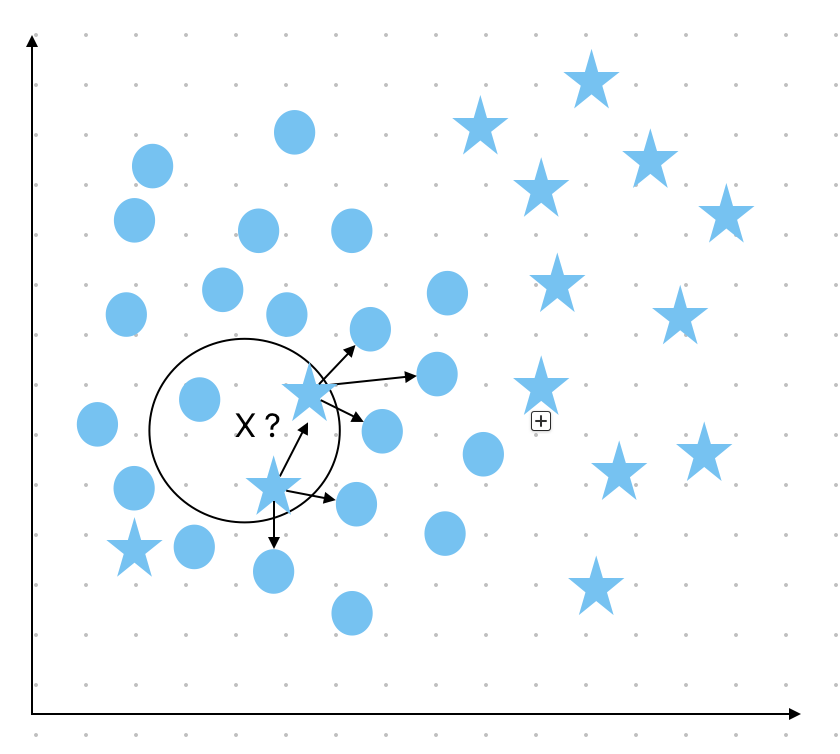
It is clearly seen in all the first three cases(shown in first three figures) , getting neighbors of the ones that were minority in low-level neighborhoods would not have changed the final classification of X(see Figure 5).

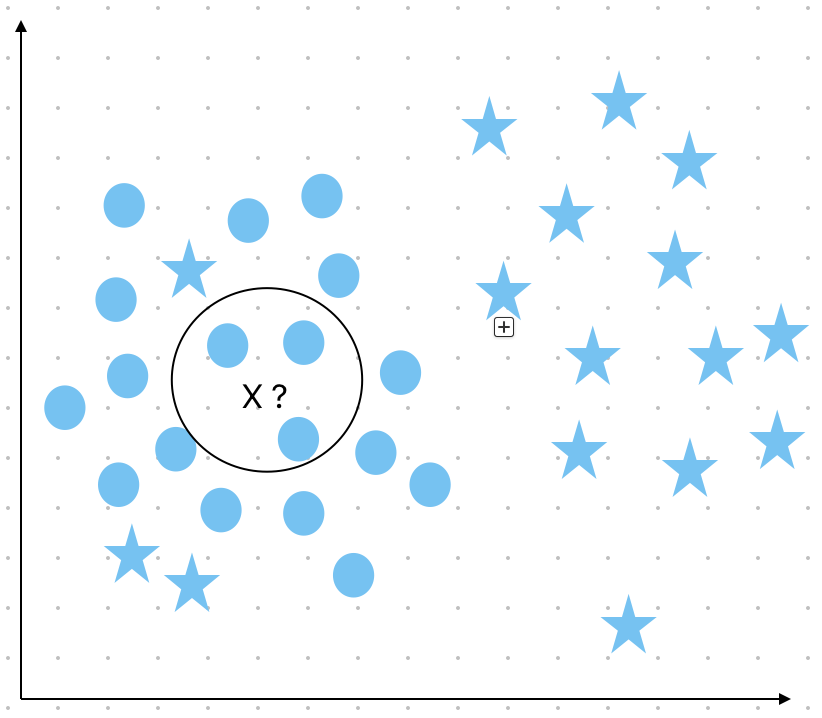
It is also clear that when all the neighbors in the minority class have the same class, then getting the neighbors of each of the low-level neighbors will yield the same result, X as a circle.

For example in Figure 1, the star was the minority class and k neighbors of the star are blue-circles. And so counting the occurrence of each class in both low-level and high-level neighborhoods together would yield the answer blue-circle for X.

Hence, the proposed algorithm runs faster than HL-KNN as it does not calculate distances for all the 7 neighbors of the low-level neighbors.

The experimental results in the next section prove that the proposed algorithm runs faster than HL-KNN.

 Figure 2

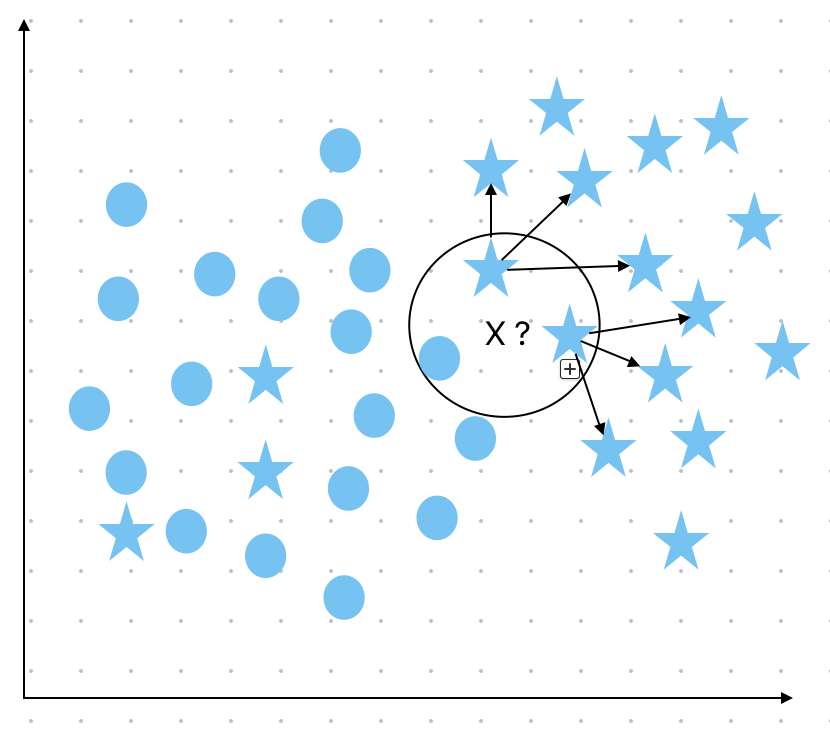
 Figure 4

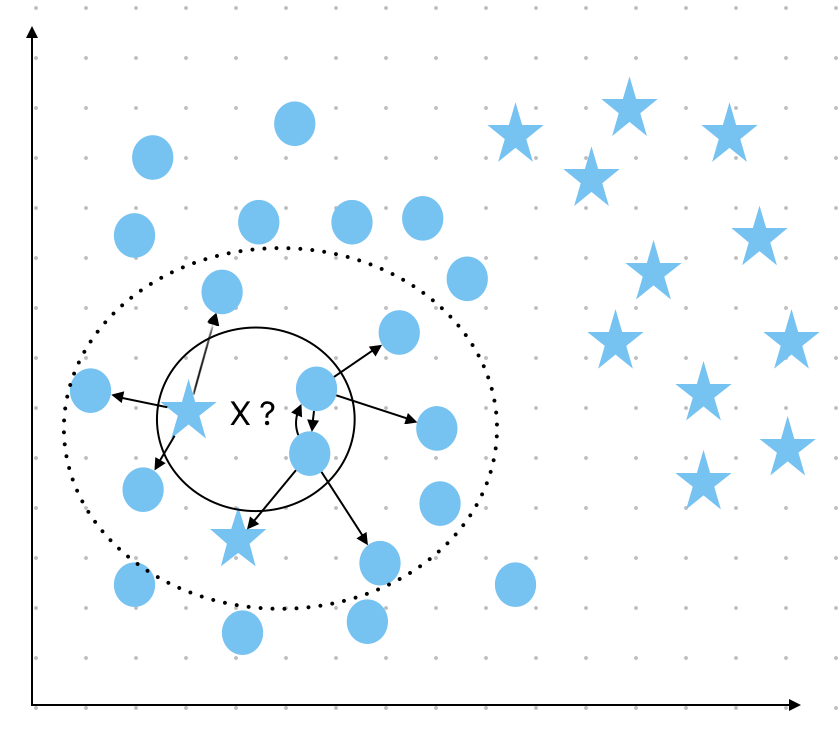
# Experimental Studies

The effectiveness of the improved version of the algorithm was tested with three data sets. All the three data sets were tested with both HL-KNN algorithm and with the modified versions of HL-KNN algorithm. There are two modified versions, Version 1 does not calculate high-level neighbors if all the neighbors in low-level have the same class. Version 2 calculates the high-level neighbors of only majority class neighbors in the low-level neighborhood. (Version 2 includes Version 1 changes as well)

The hardware and software specifications are Apple M2 Pro processor, 32 GB memory, Jupyter Notebook with Python version 3.

The test is conducted in three phases. Baseline is testing the time and accuracy of the HL-KNN algorithm. Stage1 is testing the time and accuracy of the modified version 1. And Stage2 is testing the time and accuracy of the modified version 2.

 Figure 3

 Figure 5

1. Time and Accuracy for each of the Datasets

| Dataset (size) | ***Metrics*** | ***Baseline*** | ***Stage1*** | ***Performance Change*** | ***Stage2*** | ***Performance Change*** |
| --- | --- | --- | --- | --- | --- | --- |
| Iris.csv (150 rows) | Time (sec) | 11.5 | 10.84 | 5.7% improved | 10.65 | 7..3% improved |
| Accuracy(%) | 100 | 100 | No change | 100 | No change |
| ILPD.data (582 rows) | Time (sec) | 153.51 | 164.05 | 6.9% increase | 114.3 | 25% improved |
| Accuracy(%) | 68.376 | 68.37 | No change | 67.52 | 1.2% decrease |
| Diabetes.csv  (768 row) | Time (sec) | 257.59 | 231.69 | 10% improved | 195.356 | 24.1% improved |
| Accuracy(%) | 77.5 | 77.92 | No change | 75.9 | 2.0% decrease |
| BankNoteAuthenticatoin.txt (1372 rows) | Time (sec) | 653.25 | 700.47 | 7.2% increase | 153.51 | 76% decrease |
| Accuracy(%) | 99.27 | 99.27 | No change | 68.37 | 31.1% decrease |
| DryBeans.csv  (2722 rows) | Time (sec) | Hardware support | Hardware support | \_ | 2041.54 | \_ |
| Accuracy(%) | Hardware support | Hardware support | \_ | 58.348 | \_ |

The table 1 shows time and accuracy for testing different datasets with each of the versions of HL-KNN.

Tested the HL-KNN algorithm for five datasets of different data sizes. The below was observed:

1. For datasets that are below 1000 rows in size, there has been a significant decrease in time taken to compute classification of test data with minimal or change in accuracy.

(i) Time taken(version1): improved by 10% compared to HL-KNN

(ii) Accuracy(version1): changed by average of 5% compared to HL-KNN

(iii) Time taken(version2): improved by 25% compared to HL-KNN

(iv) Accuracy(version2): changed by average of 2% compared to HL-KNN

1. For datasets that are above 1000 rows in size, there has not been much time improvement and decrease in accuracy.

The test results show that the improved versions of the HL-KNN work very well for small datasets(<1000 rows in size)

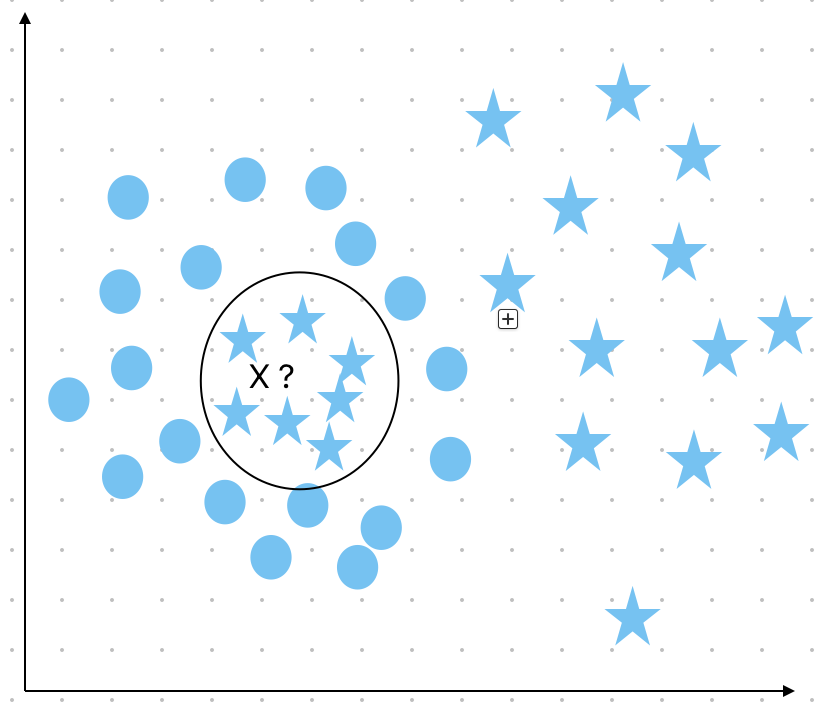
# Conclusion and Future Work

The proposed algorithm does show that there is a significant amount of reduction in time consumption of the algorithm, there is still scope of improvement in the algorithm for getting higher accuracy with reduction in time consumption.

The future changes to the algorithm will be as listed below:

1. When all the neighbors in the low-level neighborhood(N) have the same class, then calculate high-level neighbors for four distant neighbors in N. This will eliminate wrong classification due to noise as well reduce the time compared to HL-KNN

Figure 6 clearly explains the wrong prediction due to the presence of noise. considering k as 3 instead of 7 for simplicity.

 Figure 6

1. When the number of neighbors calculated in version 2 have a tie in class, then get the high-level neighbors for the rest of the low-level neighbors.

For example, there are 4 neighbors in the low-level neighborhood with a majority. Calculating 7 neighbors of each of the 4 will give 28 high-level neighbors. Since 28 is an even number there is a possibility of getting a tie in the number of class occurrences. (See Fig 7)

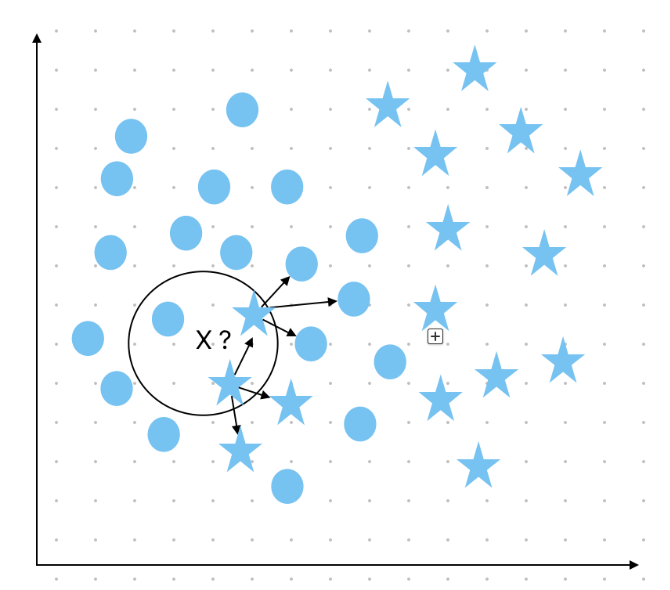


Figure 7: k is taken as 3 for pictorial simplicity.

##### References

1. E. O. Kiayak, B. Ghasemkhani and D. Berant, “High-Level K-Nearest Neighbors: A Supervised Machine Learning Model for Classification Analysis”, Advances in Data Science: Methods, Systems, and Applications, Electronics 2023, 12(18),3828.

<https://doi.org/10.3390/electronics12183828>

1. S. Ougiaroglou and G Evangelidis, “Dealing with Noisy data in context of k-NN Classification '', Conference: BCI 2015 At: Craiova, Romania, September 2015, DOI: 10.1145/2801081.2801116.

<https://www.researchgate.net/publication/291342831_Dealing_with_noisy_data_in_the_context_of_k-NN_Classification>

1. P. A. Devijver and J. Kittler, “On the edited nearest neighbor rule”, In Proceedings of the Fifth International Conference on Pattern Recognition, The Institute of Electrical and Electronics Engineers, 1980
2. Tomek, “An experiment with the edited nearest-neighbor rule”, IEEE Transactions on Systems, Man, and Cybernetics, 6:448–452, 1976
3. F. V´azquez, J. S. S´anchez, and F. Pla, “A stochastic approach to wilson’s editing algorithm”. In Proceedings of the Second Iberian conference on Pattern Recognition and Image Analysis - Volume Part II, IbPRIA’05, pages 35–42, Berlin, Heidelberg, 2005. Springer-Verlag
4. J.N.Mazón, L. Micó; F. Moreno-Seco, F, “New Neighborhood Based Classification Rules for Metric Spaces and Their Use in Ensemble
5. Datasets:Iris.csv(150 rows), IndianLiverPatientData(ILPD) (582 rows), Diabetes.csv(768 rows), BankNoteAuthentication.txt(1372 rows) and DryBeans.csv(2722 rows) Datasets taken from ([https://archive.ics.uci.edu](https://archive.ics.uci.edu/))