


Práctica: K-Nearest-Neighbor

1st Samed Rouven Vossberg 

Karlsruhe Institute of Technology

Karlsruhe, Germany

Charité – Universitätsmedizin Berlin

Movement Disorder and Neuromodulation Unit

Berlin, Germany

Universidad Nacional Autónoma de México

Mexico City, Mexico

samedvossberg@gmail.com

Abstract—This paper presents a comprehensive exploration of the k -Nearest Neighbors (kNN) algorithm for both classification and regression tasks, using three datasets to illustrate its versatility: *Glass*, *Pima Indians Diabetes*, and *Automobile Price*. We compare different normalization strategies (min-max and standardization) and show how they can significantly affect performance, particularly when features vary in scale or when outliers are present. Furthermore, we investigate the impact of tuning the Minkowski distance exponent, highlighting how emphasis on larger coordinate differences can either improve or degrade accuracy, depending on the data distribution. We also extend the traditional kNN approach from classification to regression, demonstrating how prediction by averaging neighbor values can be evaluated through the mean absolute error metric. The experimental results confirm that careful choice of k , distance exponent, and normalization technique can markedly influence model performance, thereby underscoring the importance of adapting hyperparameters and preprocessing steps to the characteristics of the datasets.

Index Terms—Data Exploration, Machine Learning, UNAM

I. INTRODUCTION

Machine Learning (ML) is a programming technique that allows machines to learn without needing explicit programming at each step. ML algorithms are divided into three main categories: supervised learning, unsupervised learning, and reinforcement learning. Here, we focus on a type of supervised learning, which uses labeled data to make predictions. This learning process also compares the output with the correct and intended output to identify errors. The main focus of this work, the K-Nearest Neighbors (KNN) algorithm, classifies new data according to the most frequent category among its K nearest neighbors. [1]

II. RELATED WORK

The k -Nearest Neighbors (kNN) classifier is among the most intuitive and widely applied machine learning algorithms for both classification and regression tasks [2], [3]. In its simplest form, kNN assigns a label to a query instance based on the labels of its k nearest neighbors in the training set, typically determined by a distance metric such as Euclidean distance. Despite its conceptual simplicity, kNN can achieve strong performance in numerous application domains, particularly when appropriate normalization and distance metrics are employed.

In this work, we consider three distinct datasets to study the performance of kNN . The first one is the **Glass** dataset [4], originally motivated by criminological investigations. It contains measurements (e.g., refractive index and chemical compositions) that allow identifying different types of glass (e.g., window glass, headlamps, tableware). Because the goal is to predict the glass type, this dataset is framed as a classification problem.

The second dataset is the well-known **Pima Indians Diabetes** dataset [5], which comprises clinical variables such as plasma glucose concentration, diastolic blood pressure, body mass index, and so on. The task is to predict whether an individual is “tested positive” or “tested negative” for diabetes, making it a binary classification problem. This dataset serves as a standard benchmark to evaluate classification algorithms, including kNN .

Finally, the **Automobile-** or **Auto price** dataset [6] is employed as a regression problem. Here, various continuous attributes of cars (e.g., engine size, horsepower, and curb weight) are used to predict the car’s price. This dataset illustrates how kNN can be adapted beyond classification, relying on local averaging of neighboring points’ target values for prediction.

Together, these datasets demonstrate the versatility of kNN for both classification and regression. In the sections that follow, we build upon these foundations to explore the impact of normalization techniques, the choice of distance metrics, and other design decisions on kNN performance.

III. METHODOLOGY

The three datasets were processed as DataFrames using the Pandas library. The K-Nearest-Neighbor algorithm was studied through the kNN class found in the Jupyter notebook provided with the *Glass* and *Pima Indians Diabetes* datasets. Test and training data were defined using *train_test_split* from the sklearn library, and the class was tested for a set of k values in the range $K = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31]$, obtaining the algorithm’s accuracy for each value. Using the same range for k , accuracy was obtained for both the test and training data, but this time normalized through the *normalize*

method within the kNN class. The accuracy of the kNN class was also tested with the *Glass* dataset with its normalized attributes, varying the Minkowski distance exponent in the range $\text{Exp} = [1, 2, 3, 4, 5, 10, 20, 50, 100, 1000, 5000, 10000]$. For this, the provided script from the Jupyter Notebook was also used.

For data preprocessing, two methods of scaling were compared:

- **Min–Max Normalization (our *normalize* method):**

$$x_{\text{minmax}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

which rescales each feature to the range $[0, 1]$.

- **Standardization (using *preprocessing.scale*):**

$$x_{\text{standard}} = \frac{x - \mu}{\sigma}$$

which transforms each feature so that it has mean 0 and standard deviation 1.

The *getClassProbs* method was added to the kNN class. In this method, for each row in the testing DataFrame, a regression value equal to the mean of the values in the training label set of the k nearest neighbors to the instance in the training set is computed. In other words, it takes a row from the testing set and assigns it the probability of belonging to each class in the training set, which is obtained by calculating the frequency of each class and then is represented as a DataFrame.

IV. RESULTS AND DISCUSSION

A. Impact of Data Normalization on kNN Classifier Performance

The two graphs in Figure 1 show a comparison of training and test accuracy of the kNN classifier for the *Diabetes* and the *Glass* datasets. The difference in each plot is the way the training and test data were preprocessed:

- *No Normalization* (raw features),
- *Min–Max Normalization* (rescaling each feature to $[0, 1]$),
- *Standardization* (mean = 0, standard deviation = 1) using the scikit-learn *scale* function.

a) *Glass Dataset.*: From the right-hand plot, we can see that the results drastically improve when normalizing the data, nearly regardless of the value of k . When using standardization, there is a slight improvement overall; however, especially as k gets larger, the advantage of standardization seems to diminish. One likely reason for these improvements is that the *Glass* dataset contains features on quite different scales. Since kNN relies on distance computations, bringing all features onto a comparable scale can lead to more meaningful distance measurements. In addition, min–max normalization can be especially helpful if certain features have dominant ranges that would otherwise overwhelm the distance metric. Standardization can also help, but its effects may fade at higher k , because larger neighborhood sizes sometimes reduce the sensitivity to exact distance magnitudes.

b) *Diabetes Dataset.*: By contrast, the left-hand plot indicates interesting behavior for the *Diabetes* dataset. We observe worse results under min–max normalization for values of $k < 5$. However, for $k > 5$, the normalized data outperform or at least match the other methods, and the accuracy curve is more consistent. A possible explanation is that for very small k , the distance metric becomes hypersensitive to small variations introduced by min–max scaling, particularly if there are outliers or skewed features. As k grows, these small-scale variations matter less, allowing normalized data to perform better.

When using standardization, we do not see the early drop in accuracy for $k < 5$. The accuracy generally remains above 70% for almost all k values, and even increases slightly with larger k . Standardization tends to reduce the impact of outliers compared to min–max (by shifting and scaling based on mean and standard deviation), which may explain why there is no initial accuracy penalty at lower k .

c) Possible Reasons for Observed Differences.:

- **Feature Scales & Outliers:** The degree to which features differ in scale or contain outliers heavily influences distance-based algorithms like kNN. Min–max normalization can compress large outliers into a narrow range, whereas standardization spreads data according to standard deviations.
- **Sensitivity at Small vs. Large k :** When k is very small, the classifier is highly sensitive to local distance relationships, so normalization can either help or hurt depending on how the feature distributions shift. As k grows, individual data point distances become less critical, potentially diminishing the differences between normalization methods.
- **Dataset Characteristics:** The *Glass* dataset, with potentially more disparate feature scales, gains more from normalization. The *Diabetes* dataset may have tighter feature ranges or fewer extremes, so standardization proves more consistently beneficial across all k .

Overall, these experiments illustrate that normalization strategies can significantly affect kNN performance, but the exact outcome often depends on the interplay between dataset characteristics (scales, outliers, and feature distributions) and the choice of neighborhood size k .

B. Effect of Increasing the Minkowski Exponent in kNN

In addition to adjusting the number of neighbors, one can also tune the exponent of the Minkowski distance in the kNN algorithm. By default, $\text{exp} = 2$ yields the Euclidean distance, whereas $\text{exp} = 1$ corresponds to the Manhattan distance. As the exponent grows larger (e.g., 10, 100, 1000), the distance measure increasingly emphasizes the largest coordinate difference between two instances, effectively approaching the so-called Chebyshev distance in the limit of $\text{exp} \rightarrow \infty$.

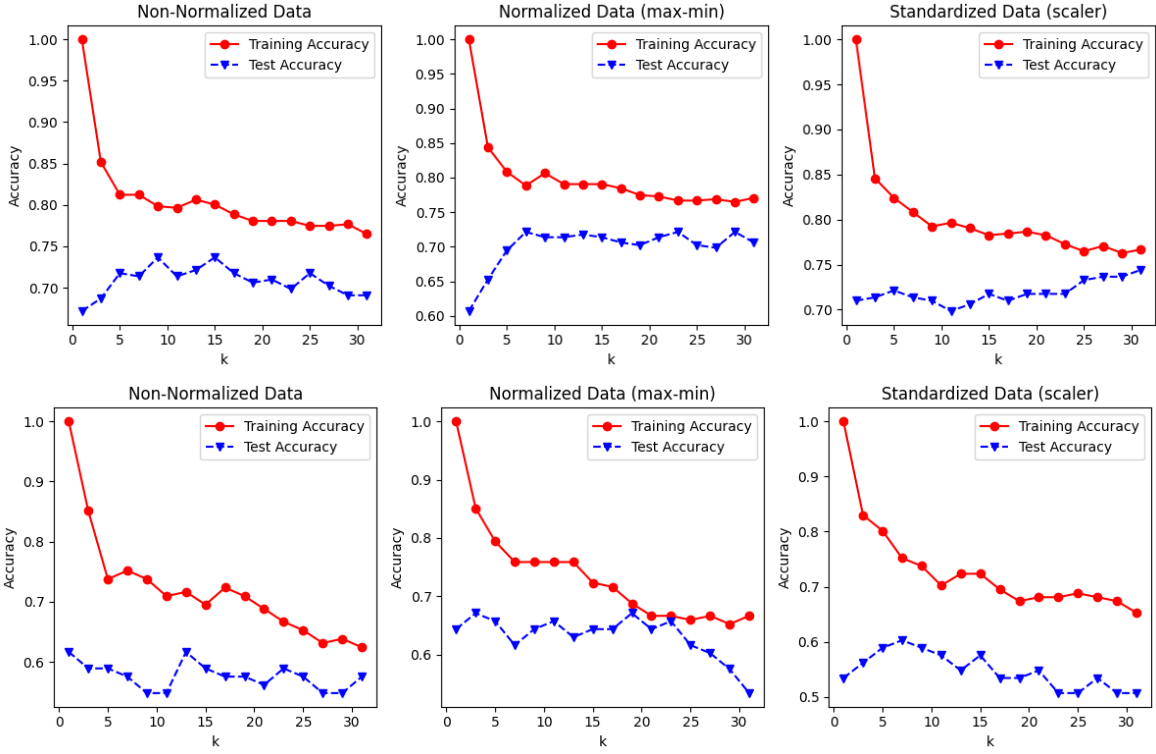


Fig. 1. Comparison of training and test accuracy of the kNN classifier for the Diabetes dataset (*first*) and the Glass dataset (*second*), using three different preprocessing strategies: no normalization, min-max normalization, and standardization.

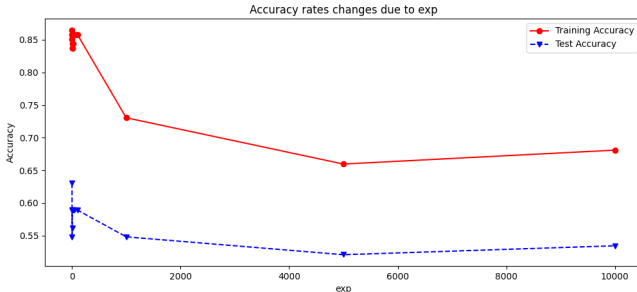


Fig. 2. Training and test accuracy of the kNN classifier on the *Glass* dataset as the Minkowski distance exponent, *exp*, increases.

Figure 2 illustrates the behavior of both the training and test accuracy as *exp* is varied. We observe:

- **Initial Drop in Accuracy:** Transitioning from low exponents (e.g., 1–2) to large exponents leads to a noticeable decline in test accuracy. This is because the distance measure starts to be dominated by whichever attribute has the largest absolute difference, thus potentially reducing the value of more “balanced” distance contributions from other features.
- **Partial Convergence:** As *exp* becomes very large (> 1000), the training and test accuracy curves level off, indicating that the distance metric has converged in practice to a form where only the “worst-case” feature difference matters. In some cases, this can degrade classification

performance if that single largest difference is not the most meaningful for discrimination.

- **Overall Effect on kNN:** Because kNN is a distance-based technique, changes to how distance is measured can substantially alter neighbor rankings. Larger exponents shift emphasis to outliers or dominant features, and depending on the dataset’s structure, this may result in lower accuracy or overfitting.

Hence, while Euclidean distance ($\text{exp} = 2$) and Manhattan distance ($\text{exp} = 1$) are most common in kNN, experimenting with the exponent can sometimes yield improvements, especially for certain data distributions. However, extreme values of *exp* often yield diminishing returns or even harm performance, as shown by the convergence of accuracy to suboptimal levels in Figure 2.

C. Estimation of the posterior class probabilities for all test instances

In Tables 1 to 3, the results of calculating the probability that each element of the dataset belongs to each class while varying the parameter *k* are shown. In Table 1, three neighbors are considered, and therefore, the probability of belonging to a class can be 0%, 33.3%, 66.6%, or 100%. There are also many classes to which an element has a null probability of belonging, as the distance to the non-nearest neighbors is very large.

In the following tables, the value of *k* is significantly increased, and it is clearly observed that as more neighbors are

considered, the probabilities of elements belonging to multiple classes increase considerably, as there are fewer null values and a more balanced proportion of probabilities.

It is also noticed that in all the tables, it is observed that the probabilities of the first four elements belonging to the last class are null, which could indicate that the elements of this class are more isolated from these, which makes sense as it is the class with the fewest elements.

TABLE I
HEAD OF THE DATAFRAME OBTAINED WITH *getClassProbs* WITH $k = 3$

	0	1	2	3	4	5
0	0.666667	0.333333	NaN	NaN	NaN	NaN
1	1.000000	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	0.666667	0.333333	NaN	NaN
3	0.333333	NaN	NaN	NaN	0.666667	NaN

TABLE II
HEAD OF THE DATAFRAME OBTAINED WITH *getClassProbs* WITH $k = 21$

	0	1	2	3	4	5
0	0.619048	0.285714	0.095238	NaN	NaN	NaN
1	0.904762	NaN	0.095238	NaN	NaN	NaN
2	NaN	0.095238	0.476190	0.285714	0.142857	NaN
3	0.333333	0.1428	0.523810	NaN	NaN	NaN

TABLE III
HEAD OF THE DATAFRAME OBTAINED WITH *getClassProbs* WITH $k = 31$

	0	1	2	3	4	5
0	0.548387	0.225806	0.193548	0.032258	NaN	NaN
1	0.709677	0.161290	0.096774	0.032258	NaN	NaN
2	0.032258	0.419355	0.096774	0.354839	0.096774	NaN
3	0.322581	0.483871	0.193548	NaN	NaN	NaN

D. kNN Regression on the *autoprice* Dataset

For regression problems, our *kNN* class offers the *getPrediction* method, which computes a continuous prediction for each test instance by averaging the training *y*-values of the *k* nearest neighbors. Specifically, for each test sample, the algorithm calculates Minkowski distances to all training points, selects the *k* closest ones, and then returns the mean of their corresponding target values as the prediction. This allows us to apply the same *kNN* framework for both classification and regression tasks, distinguishing them through how the final output is aggregated (majority vote for classification vs. average for regression).

Figure 3 presents the mean absolute error (MAE) of *kNN* regression across various *k* values on the *autoprice* dataset, where *class* is a continuous price attribute. We compare:

- **Raw Data:** Features are used as given (no scaling).
- **Normalized Data:** Each feature is scaled to the range $[0, 1]$ using min-max normalization.

The results show that normalization often produces *higher* MAE, particularly for smaller *k*, indicating that scaling can also worsen distance-based methods. However, depending on the dataset and preprocessing steps, scaling could also enhance these methods by ensuring that all features contribute more equally to neighbor searches. As *k* increases, the performance difference between raw and normalized data becomes somewhat narrower; however, non-normalized data generally maintains an edge or at least stays comparable in performance. Overall, the figure illustrates how *kNN* regression performance is sensitive to both the number of neighbors *k* and the presence of feature scaling.

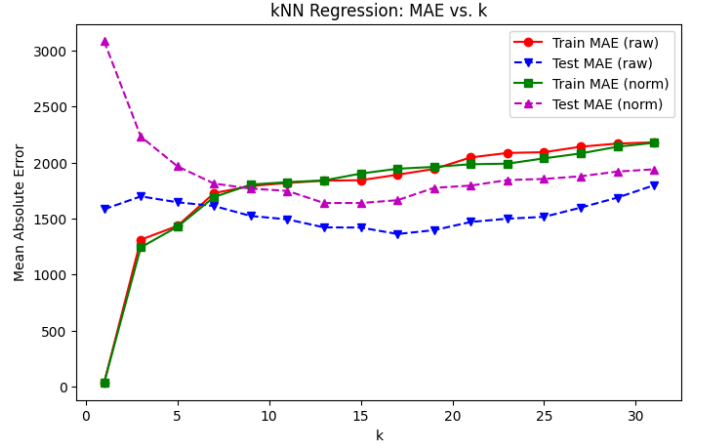


Fig. 3. *kNN* Regression on the *autoprice* dataset: Training and test set mean absolute error (MAE) for multiple values of *k*. We compare raw data (no normalization) and min-max normalized data.

V. CONCLUSION

The results from the tests with the Glass and Diabetes datasets highlight the importance of data normalization in *kNN* classifier performance, while also showing that its impact varies depending on the dataset characteristics and the values of *k* and *exp*. For the Glass dataset, normalization, especially Min-Max normalization, significantly improved classifier performance due to disparities in feature scales, allowing for more equitable distance comparisons. However, for higher values of *k*, the benefits of standardization diminished, suggesting that larger neighborhood sizes reduce the sensitivity to distance variations.

In contrast, the Diabetes dataset displayed more complex behavior, where Min-Max normalization initially negatively affected performance for small *k*, likely due to the classifier's hypersensitivity to small variations in scaled data, particularly with outliers or skewed features. However, as *k* increased, normalized data outperformed or at least matched other methods. Standardization showed more consistent results across all *k* values, possibly due to its ability to mitigate outlier impacts.

REFERENCES

- [1] H. Vaishnav and A. Choudhary, "Evolutional study on knn and k-means algorithms," in *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, vol. 1, 2024, pp. 1–4.
- [2] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.
- [3] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.
- [4] B. German, "Glass Identification," UCI Machine Learning Repository, 1987, DOI: <https://doi.org/10.24432/C5WW2P>.
- [5] V. Chang, J. Bailey, Q. A. Xu, and Z. Sun, "Pima indians diabetes mellitus classification based on machine learning (ml) algorithms," *Neural Computing and Applications*, vol. 35, no. 22, pp. 16 157–16 173, 2023.
- [6] J. Schlimmer, "Automobile," UCI Machine Learning Repository, 1985, DOI: <https://doi.org/10.24432/C5B01C>.