K-Nearest Neighbors Classification
An Applied Simple Nonparametric Approach

Report by Salvador Durán, Leonardo Pasotti, Inés Pestarino, Mea Railo & Pietro Trimboli

Introduction
Nonparametric statistics are statistical methods that use minimal assumptions about the data's distribution. One prominent example is the k-nearest neighbors (k-NN) classifier, a flexible method that is used for classifying data. Rather than constructing a model, k-NN works with the available data. To classify a new observation, k-NN identifies the k nearest samples from the training set and assigns the new point to the class that is most common within those neighbors. We approach this report with an application-focused perspective, demonstrating how changing the parameter k affects classification outcomes across 2D, 3D, and higher-dimensional datasets.

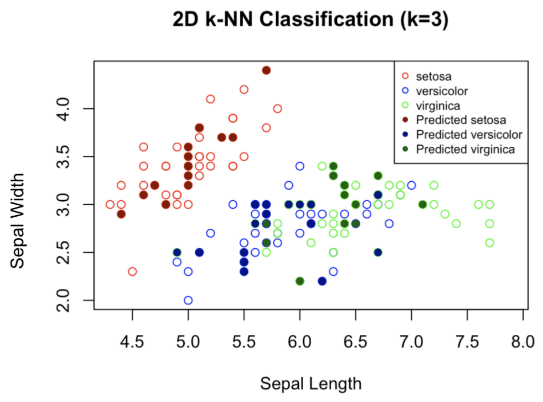
Main Use
k-NN is mostly applied in classification problems, including image recognition or medical diagnosis. The algorithm functions by computing the distance between a new observation and existing data points in the dataset. The algorithm selects the k closest neighbors (observations) and the new point is assigned with the most frequent class among them.

Methodology
Distance, hence closeness, is crucial in determining the k nearest training points among the xi training points for the K-NN algorithm. As stated, the algorithm identifies the training points with the lowest distances d (x, xi) with respect to the query point x0 (which is then assigned to a so-called majority class). Indeed, for simplicity of computation, the metric distance mostly employed is the Euclidean distance (equal to sqrt(Σ (xi$_2$ - xi$_1$)^2), for i = 1 to n), which is appropriate for quantitative feature space (as used by the KNN library in the R code for this report).

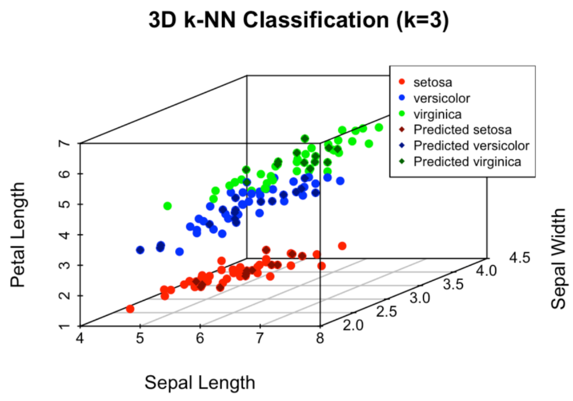
The key to evaluate the model complexity of the k-NN algorithm (in order to minimise the test error) is by measuring the bias-variance tradeoff through choosing different values of the parameter k (number of training points selected); in fact, selecting a low parameter k implies a low bias and high variance (therefore varying upon various training sets), while instead a high parameter chosen leads to high bias and low variance (since the algorithm is necessarily less affected by specific training samples, adapting itself closely to the training data).

Application
k-Nearest Neighbors Classification of the Iris Dataset
We will now implement k-NN on the Iris dataset, which is a classification benchmark consisting of sepal and petal dimensions of three iris species (setosa, versicolor, virginica). Our goal is to demonstrate the effect of feature selection and dimensionality on classification accuracy. The effect of feature selection and dimensionality on k-NN classification may be explained with a simple analogy. Assume predicting a student's final mark. If we look only at attendance and homework completion (similar to consideration of Sepal Length and Sepal Width in the Iris dataset), we may not make a good prediction because of the absence of the understanding of the concepts assessed through the features. Incorporating test scores would likely boost the prediction accuracy. In the same sense, whilst using only Sepal Length and Sepal Width, we most likely would not be able to distinguish species with high accuracy in the Iris dataset.

In trying to answer this question, we performed an initial exploration of the Iris dataset. Using a train-test split (70-30), we generated 2D and 3D visualizations of the data, and found that increasing feature space, especially with useful features, greatly improves classification performance. As is visible in the 2D plot (left image), the classification accuracy with only Sepal Length and Sepal Width is very low due to the high overlap between species. However, as illustrated in the 3D plot (right image), the inclusion of Petal Length significantly improves the separation of species. This shows how import feature selection is in k-NN classification.

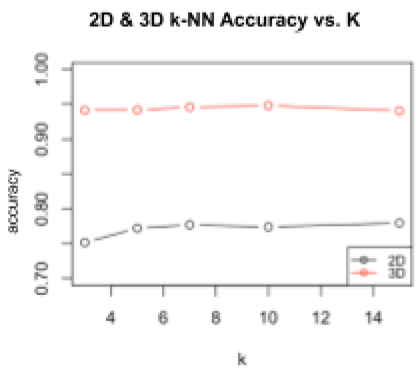


2D Accuracy (k=3): 0.733

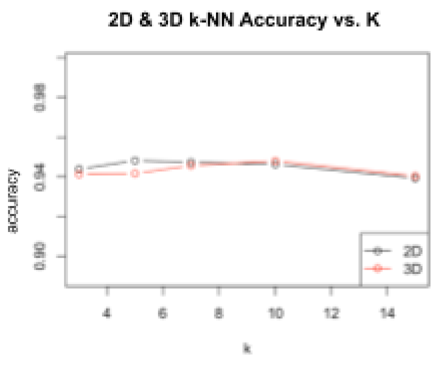


3D Accuracy (k=3): 0.978

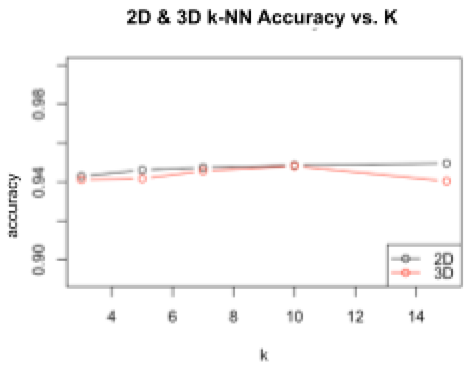
Then, to further investigate the impact of dimensionality and the choice of k, we conducted a simulation study using the Iris dataset with 10,000 trials, comparing 2D (Sepal Length & Sepal Width) performance against that of three 3D feature combinations. The simulation results, shown in the 2D k-NN Accuracy vs. k plots on the right, show that the selection of k affects the classification accuracy in the lower 2D feature space more than in the higher 3D feature space. In particular, the 2D accuracy plots of Sepal Length and Sepal Width demonstrate greater variability at different k values. On the other hand, the 3D accuracy plots including Petal Length show relatively stable performance with varying k. Adding a relevant dimension and a feature such as Petal Length does improve classification accuracy and reduces sensitivity to the choice of k. It is important to note that generally though, adding more dimensions brings problems, as we can see in the next section.



Sepal.Length & Sepal.Width



Sepal.Length & Petal.Length



Sepal.Width & Petal.Length

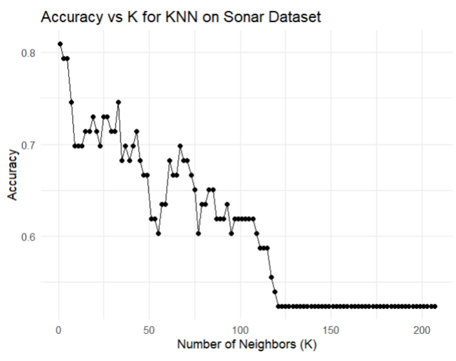
Value of k in the Sonar Dataset

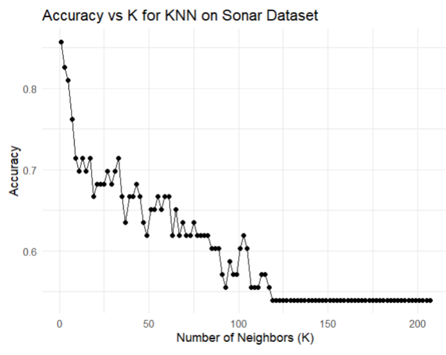
Nearest neighbour classification greatly depends on the value of k, which represents the amount of closest neighbours that are considered when analyzing each datapoint. As the dimension increases, it becomes even more crucial to select an optimal k: the "curse of dimensionality" (k-NN struggles with bigger databases) in this context states that in higher dimensions, the sparse data and unreliable distances make it harder to identify patterns or build coherent models. An interesting and insightful database to grasp the relevance of the k choice and its impact on the model's precision is the "Sonar" dataset, made up by 208 observations, each consisting of 60 continuous variables representing the energy of sonar signals reflected at many angles, and one binary variable which classifies the item either as a mine (M) or rock (R). In this context, our applicational goal is to create a model which is capable of discriminating which type (M or R) a certain datapoint is.

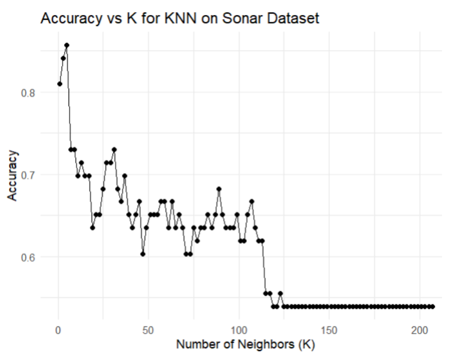
We proceed by selecting an interval of candidate k values (we choose only odd values for k because the classification is determined by a "majority vote" among the k nearest neighbors, hence odd numbers avoid ties completely), which will then be plotted to see which value is the most adequate for selection. This time, we can't create a simple 2d/3d scatterplot, as the 60-dimensional space isn't visualizable without dimensionality reduction (which would be a different theme), but mathematics teaches us that the Euclidean distance (in this case we use it to find the k-NN of each testing datapoint) can still be applied, maintaining the same order size in terms of computational cost ($O(n)$, although n can be multiplied by a large constant). In the application, we use a seed to give uniqueness to the analyzed sample (usually different results from different seeds are also compared). Such seed will randomize which data points go to the "train" and which go to the "test" subsets: the first one is going to be used to let the model understand the underlying patterns and relationships in the data, the second one is going to be subsequently analyzed, and its binary label will be predicted. In R, this happens with the help of the knn() function (coming from the "class" library), which creates a prediction after getting the "train" subset's continuous variables and label, the 60 floats of each "test" datapoint, and k as inputs, outputting either M or R as an educated prediction.

Each k value is assigned an accuracy score equal to the ratio between the correctly guessed data points and the total data points observed, but simply choosing the k value with the highest accuracy would be incorrect, as we need to consider overfitting and underfitting. The first phenomena can happen for very small k values: in this case, although accuracy might be very high (as a matter of fact, in the Sonar database $k = 1$ has the highest average accuracy across seeds), the sample is too sensitive to noise. We can't rely merely on an outlier-riddled batch, as it would be like asking how many cookies are in a jar to too few people to have a precise mean (or median for that matter). On the flipside, underfitting typically refers to the effect of having a k value which is too high. In this case, the predictions are too general, as insightful patterns are not picked up. What causes this phenomenon is the excessive smoothing of data (removing variance): it's like asking 1000 kids who can barely count to guess how many cookies are in a jar. It might seem smart to "maximize" accuracy with a k which approaches the total number of training points, so that every information available is used, however this leads to the predictive process becoming a "popularity contest", as the accuracy will be equal to the relative frequency of the most common class (in our case, M or R). This is why the proposed code also portrays very high k values: to show that after the point where k is greater than the number of objects not in the most popular class, the accuracy equals the relative frequency of the most popular class, making a straight line on the graph. Ultimately, the best k reflects the dataset's structure and must be guided by both accuracy and domain insight, all while balancing bias and variance.

The following figures portray what was explained in this paragraph, with 3 different seeds.







Conclusion

k-NN is a nonparametric classification method that relies on the majority vote of nearby data points rather than building a predictive model. Its performance is sensitive to the choice of k, feature selection, and dataset dimensionality, as seen in the application concerning the Iris dataset (to illustrate how adding meaningful features improves accuracy and reduces sensitivity to k), and the high-dimensional Sonar dataset (which demonstrates the challenges of choosing k in sparse spaces due to the curse of dimensionality). Small k can overfit, while large k risks underfitting. Overall, tuning k is a balance between bias and variance and must be guided by both statistical and domain knowledge.

Although k-NN proves to be great for smaller datasets thanks to its simplicity, it still lacks scalability and robustness, and it also needs thoughtful preprocessing and a very precise tuning. Its alternatives can be better or worse, depending on the application: a good choice would be to add a weight to each k-NN, where closer neighbors contribute more to the vote (for example by inverse distance). Notable nonparametric counterparts include decision trees, random forests, and kernel-based support vector machines. Together, these methods offer diverse tools for tackling classification problems without rigid assumptions, but as always, a good statistician or data analyst is aware of the context, and will choose the most appropriate solution to any problem.

Main references:

1. James, G., Witten, D., Hastie, T., & Tibshirani, R. An Introduction to Statistical Learning. Springer.

2. Kumari, A., & Singla, R. (2022). A Review on Analysis of K-Nearest Neighbor Classification Machine Learning Algorithms Based on Supervised Learning.

Secondary reference: Raschka, S. (2018). KNN Notes - STAT 479

Links:

https://www.stat.berkeley.edu/~rabbee/s154/ISLR_First_Printing.pdf

https://www.researchgate.net/publication/362383623_A_Review_on_Analysis_of_K-Nearest_Neighbor_Classification_Machine_Learning_Algorithms_based_on_Supervised_Learning

https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf