K-Nearest Neighbours and the Curse of Dimensionality

Student Name: Pavani Oruganti
Student ID: 24082686
GitHub: https://github.com/pavani-lucky/knn-curse-of-dimensionality

---

1. Introduction

K-Nearest Neighbours (KNN) is a simple yet powerful machine-learning algorithm that predicts the label of a new point by looking at the labels of nearby training examples in feature space. Because it relies directly on distances, it can model complex, non-linear decision boundaries without explicitly learning parameters, but its performance depends heavily on the geometry of the data and the choice of distance metric.

This tutorial investigates how KNN behaves on a low-dimensional toy dataset and how its behaviour changes when many irrelevant features are added, a situation known as the curse of dimensionality. Through controlled experiments and visualisations, the tutorial shows how KNN performance degrades in high-dimensional spaces, and how feature scaling and dimensionality reduction can mitigate these effects.

---

2. KNN intuition

KNN treats classification as a voting problem in feature space: for a new point x, it finds the k training points with smallest Euclidean distance $d(x, xi)$ and predicts the majority label among those neighbours. In two dimensions, Euclidean distance between two points $x = (x1, x2)$ and $z = (z1, z2)$ is $d(x, z) = (x1 - z1)2 + (x2 - z2)2$, and this generalises to higher dimensions by summing over more coordinates.

Because the prediction is based only on local neighbourhoods, KNN can follow curved class boundaries and adapt to complex patterns, but it is sensitive to noise, to the scale of each feature, and to how “meaningful” distances remain as dimensionality increases. The hyperparameter k controls model complexity: small k leads to highly flexible but noisy decision boundaries, while large k produces smoother, higher-bias models.

3. Dataset: make_moons

3.1 Data description

The experiments use the `make_moons` dataset from scikit-learn, which generates two interleaving half-moon shapes for binary classification. The feature matrix X has shape $(500, 2)$, meaning there are 500 observations and two numerical features per observation, while the target vector y has shape $(500,)$, providing a binary class label for each sample. Example feature vectors such as $[0.83, -0.30]$ and $[1.09, 0.90]$ paired with labels $[1, 0, 1, 0, 0]$ confirm that this is a two-class problem in a two-dimensional Euclidean space.

3.2 Visual pattern of the classes

Figure 1 shows the scatter plot of the two features coloured by class label. Points form two noisy, interleaving moon-shaped clusters, with different classes lying close to each other along a curved boundary. This non-linear structure makes the dataset a good testbed for KNN, because linear models would struggle to separate the classes without additional feature engineering.

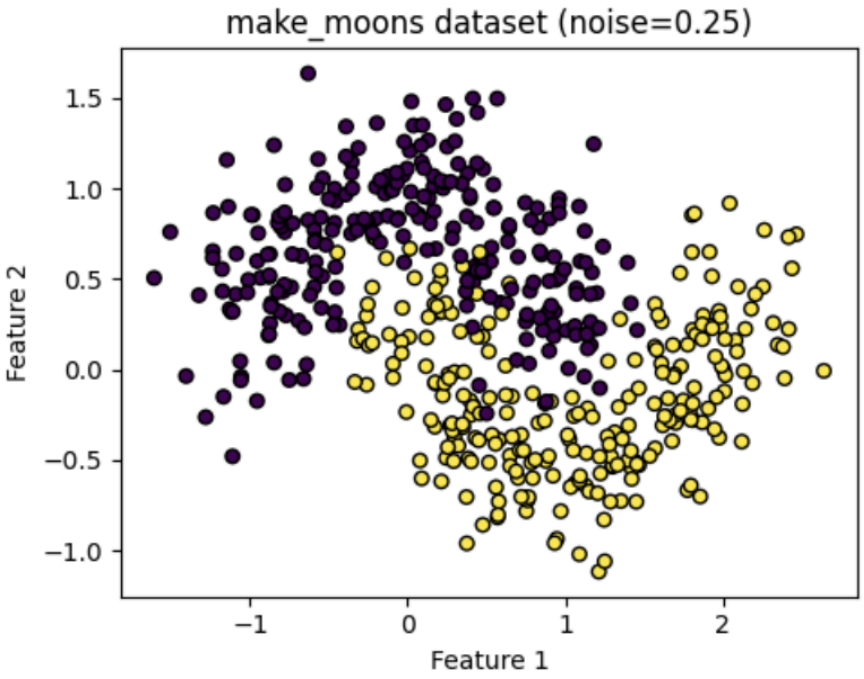

Figure 1 : scatter plot of the raw `make_moons` dataset .

4. KNN on the original 2D data

4.1 Train–test split

The dataset is split into 350 training samples and 150 test samples using a 70/30 stratified split so that class proportions remain consistent across both sets. Stratification is important in binary classification because it prevents one class from being under-represented in the test set, which would distort the estimate of generalisation performance.

4.2 Baseline KNN with $k = 5$

A baseline KNN classifier with $k = 5k$ is trained on the 2D training data. On the held-out test set, this model achieves an accuracy of 0.953, correctly classifying 95.3% of test points. This suggests that a small neighbourhood already captures much of the curved class structure while averaging over enough neighbours to be reasonably robust to noise.

Figure 2 shows the decision boundary learned by the baseline KNN model on the training data. The boundary closely follows the crescent shapes of the moons, wrapping around clusters of same-class points and placing most misclassified points in the noisy overlap region between the classes.

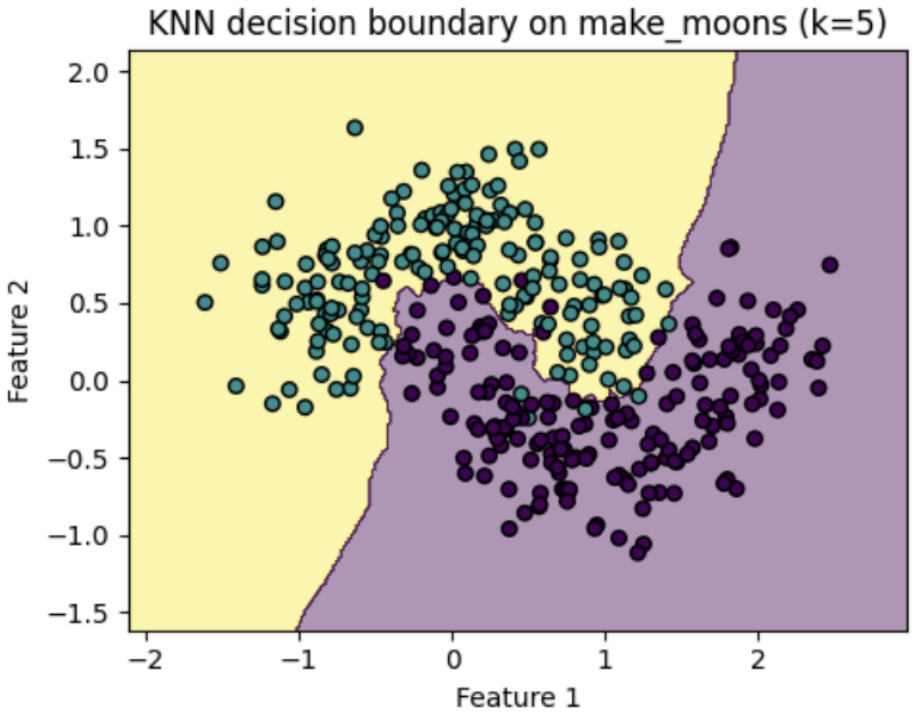

Figure 2 : decision boundary for KNN with $k = 5k$.

4.3 Effect of the number of neighbours

To study the effect of k , models are trained for $k \in \{1, 3, 5, 11, 21, 51\}$ in and their accuracies are measured on both train and test sets. When $k = 1$The model perfectly fits the training data (train accuracy 1.000) but test accuracy drops to 0.933, indicating overfitting to individual noisy points. As k increases to values between 3 and 21, training accuracy stabilises around 0.957–0.960 while test accuracy peaks at 0.960 for $k = 21$, representing a good bias–variance trade-off, whereas for very large k (51) both accuracies decline slightly due to underfitting.

Figure 3 plots train and test accuracy as functions of k, clearly visualising this transition from overfitting at small k to underfitting at very large k.

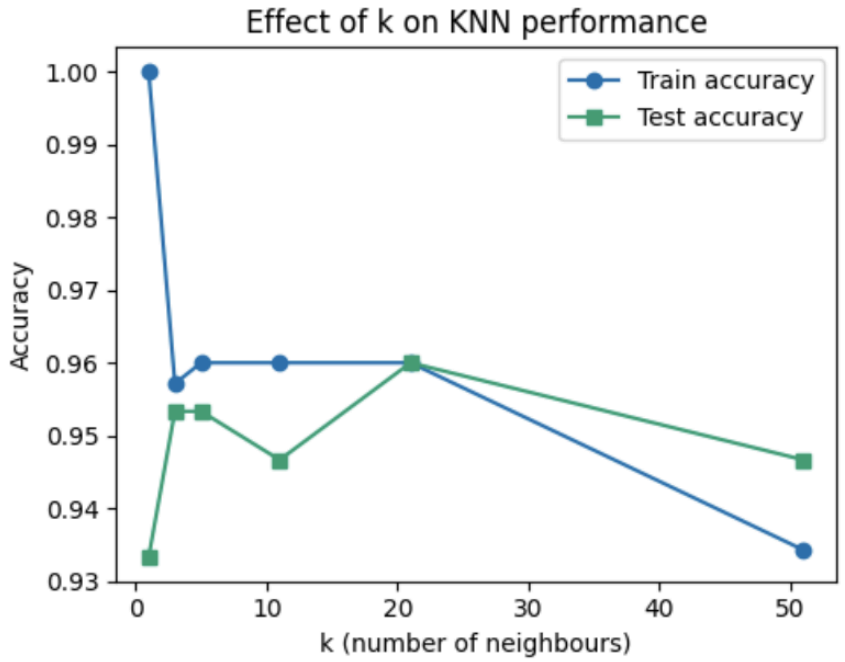

Figure 3 : accuracy vs k curve .

---

5. Curse of dimensionality experiment

5.1 Constructing a high-dimensional dataset

To emulate high-dimensional data with many irrelevant features, 48 additional noise dimensions are appended to each 2D sample by drawing independent Gaussian noise and concatenating it to the original features. The resulting feature matrix has shape $(500, 50)$and applying the same 70/30 stratified split produces training and test sets of shapes $(350, 50)$ and $(150, 50)$ respectively, where most dimensions carry no information about the class label.

5.2 Performance in 2D versus 50D

KNN is re-evaluated on both the original 2D data and the 50D noisy data for the same set of k values. In 2D, test accuracy remains high across k, ranging from 0.933 at $k = 1$ to a peak of 0.960 at $k = 21$, consistent with the earlier experiment. In 50D, however, test accuracies fall dramatically: they range from 0.587 at $k = 1$ up to only 0.740 at best, and never approach the 2D performance.

Figure 4 summarises this comparison: the 2D accuracy curve stays above 0.93 for all k, while the 50D curve lies much lower and is more sensitive to the choice of neighbourhood size. This degradation demonstrates the curse of dimensionality for distance-based methods: adding many irrelevant features makes distances less informative, so “nearest” neighbours are no longer truly similar in terms of the underlying classes.

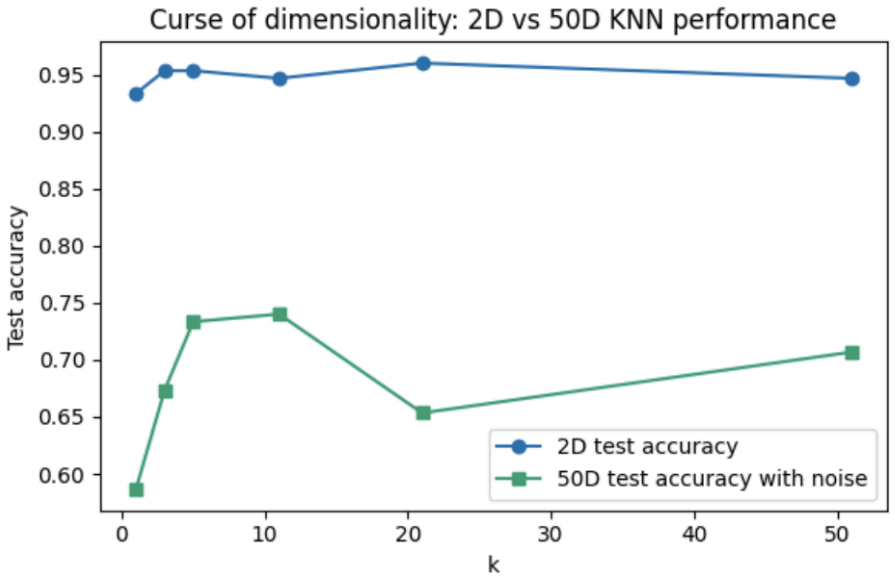

Figure 4 : 2D vs 50D test accuracy plot.

5.3 Distance concentration

To better understand why performance worsens, the ratio between each point’s nearest and farthest neighbour distances is computed for samples in the 2D and 50D training sets. In 2D, most ratios are close to zero, meaning the nearest neighbour is much closer than the farthest one and the notion of “nearest” is meaningful. In 50D, the ratios cluster tightly between approximately 0.55 and 0.65, indicating that nearest and farthest neighbours are at similar distances; this phenomenon is known as distance concentration and is a key aspect of the curse of dimensionality.

Figure 5 displays histograms of the nearest-to-farthest distance ratios in 2D and 50D. The stark difference between the spread in 2D and the narrow peak in 50D provides geometric evidence for why KNN struggles when many noisy features are present.

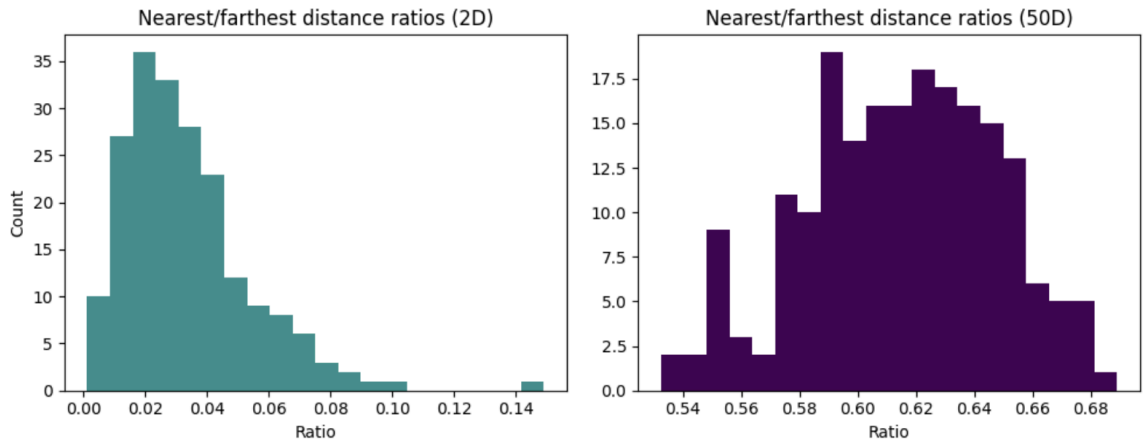

Figure 5 : histograms of distance ratios in 2D and 50D .

---

6. Mitigation strategies: scaling and PCA

6.1 Feature scaling in 50D

Since KNN relies directly on distances, feature scaling is a standard recommendation to ensure that all dimensions contribute comparably. Standardisation (zero mean, unit variance) is applied to the 50D features, and KNN is re-evaluated for the same range of kkk. Without scaling, test accuracies lie between 0.587 and 0.740, whereas after scaling they rise to the range 0.613–0.773, with the best result of 0.773 around $k = 11$ or $k = 51$. This confirms that scaling partially restores the discriminative power of distances even in the presence of many noisy dimensions. *Figure 6* compares unscaled and scaled 50D accuracies as functions of k, highlighting consistent improvements from standardisation across all neighbourhood sizes.

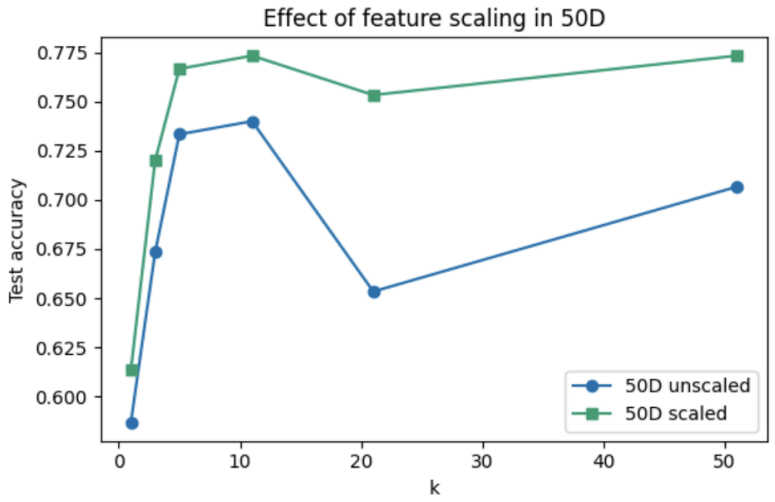

Figure 6 : unscaled vs scaled 50D test accuracy plot .

6.2 Dimensionality reduction with PCA

Principal Component Analysis (PCA) is then applied to the scaled 50D data to project it into a two-dimensional subspace that retains the directions of highest variance. Running KNN on these two principal components yields test accuracies between 0.567 and 0.673 across the tested k values: this is better than the original unscaled 50D model and closer to the scaled 50D performance, while operating in a much lower-dimensional space.

Figure 7 compares test accuracies for scaled 50D KNN and PCA-to-2D KNN. Although PCA does not fully recover the original 2D accuracy, it demonstrates that dimensionality reduction can counteract some effects of the curse by discarding noisy directions and concentrating information into fewer, more meaningful components.

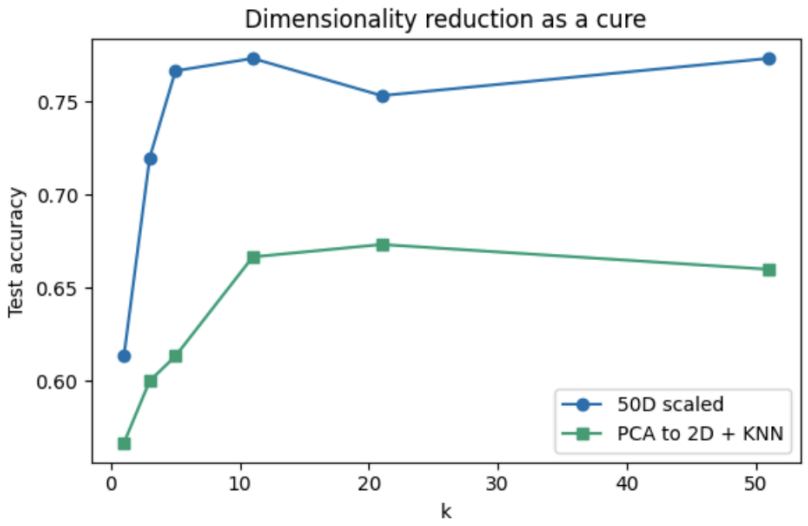

Figure 7 : scaled 50D vs PCA-2D test accuracy plot .

---

7. Practical guidance and ethical considerations

The experiments suggest several practical lessons for using KNN. First, KNN is a strong baseline for low-dimensional, well-scaled data, particularly when a moderate value of k balances noise reduction and boundary flexibility. Second, in higher dimensions or when many irrelevant features are present, performance can deteriorate sharply due to distance concentration, and mitigation strategies such as feature selection, scaling, and dimensionality reduction become essential.

In real applications, KNN is often used on data containing sensitive attributes such as location, income, or demographic factors. Because it bases predictions on “similar” neighbours, any historical bias encoded in these features can propagate into the model’s decisions, potentially disadvantaging certain groups. Careful consideration of which features to include, how to weight them in distance calculations, and how to evaluate fairness is therefore important when deploying KNN in domains like healthcare, credit scoring, or criminal justice.

---

8. Conclusion

This tutorial demonstrated how KNN performs on a simple two-dimensional `make_moons` dataset and how its behaviour changes when the data is embedded in a higher-dimensional space with many noisy features. In 2D, KNN achieved high accuracy and flexible decision boundaries, with the best generalisation around $k = 21$, while in 50D its accuracy dropped substantially due to the curse of dimensionality and distance concentration. Feature scaling and PCA were shown to partially recover performance by making distances more comparable across features and by projecting the data into a lower-dimensional subspace that filters out noise.

Overall, the results highlight that although KNN is conceptually simple, its success depends critically on dimensionality, feature scaling, and thoughtful preprocessing; understanding these factors allows practitioners to use KNN effectively and to recognise when other models or dimensionality reduction techniques are required.

---

9. References

- Scikit-learn developers. “KNeighborsClassifier — scikit-learn documentation.”KNeighborsClassifier — scikit-learn 1.8.0 documentation
- Scikit-learn developers. “make_moons — scikit-learn documentation.”make_moons — scikit-learn 1.7.2 documentation
- Cornell University. “Lecture 2: k-nearest neighbors / Curse of Dimensionality.”Lecture 2: k-nearest neighbors / Curse of Dimensionality
- GeeksforGeeks. “How does KNN work for high dimensional data?”How does KNN work for high dimensional data? - GeeksforGeeks
- ArXiv. “Interpreting the Curse of Dimensionality from Distance ...”Interpreting the Curse of Dimensionality from Distance Concentration and Manifold Effect
- Wikipedia. “Curse of dimensionality.”Curse of dimensionality - Wikipedia
- Scikit-learn. “Importance of Feature Scaling.”Importance of Feature Scaling — scikit-learn 1.7.2 documentation
- Additional tutorials on KNN classification with scikit-learn.9. k-Nearest-Neighbor Classifier with sklearn | Machine Learning