K Nearest Neighbors Classifier

What is KNN?

K Nearest Neighbors (KNN) classifier is a **supervised**, **non-parametric** machine learning model.

Supervised: A KNN classifier uses labeled input and output data in order to detect the relationships between the input and output results.

Non-parametric: No assumptions about the functional form of the model are made.

KNN can also be categorized as a **deterministic**, **discriminative**, and **local** classifier.

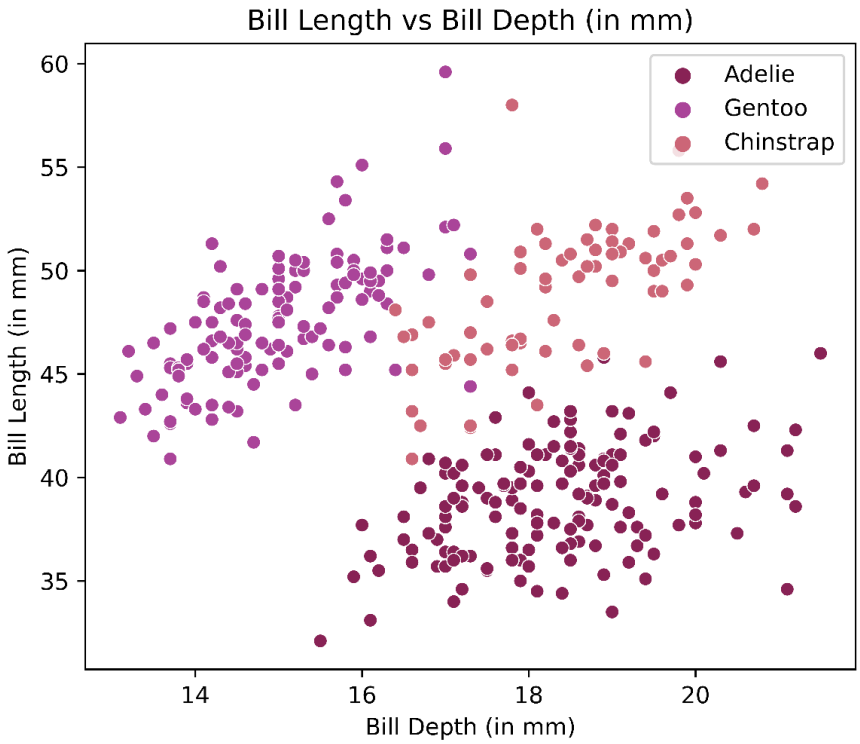
Deterministic: KNN assigns a discrete-value label for each data instance. So in terms of prediction, the actual prediction is going to be a label or a class like 'yes' or 'no'.

Discriminative: The classifier does not try to replicate the data generation process. Instead, it predicts class labels without explicitly describing the distribution of each class label.

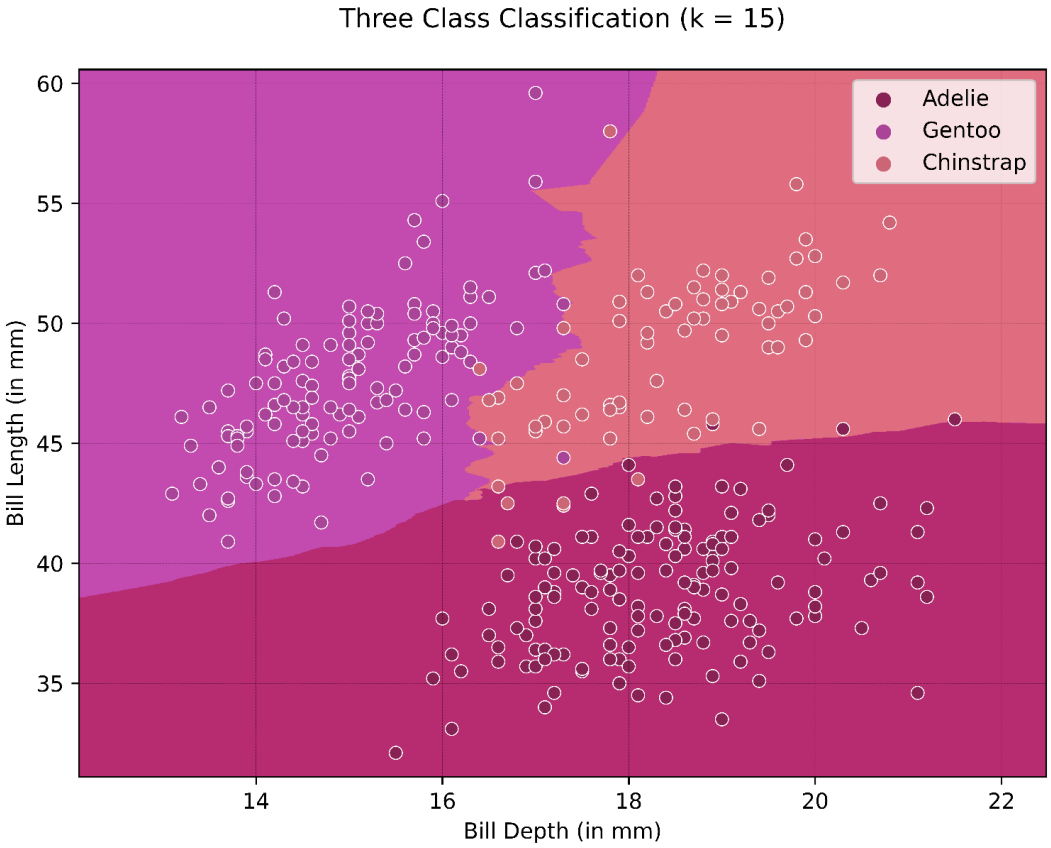
Local: KNN does not take a single model and fit it to the entire dataset. Rather, it takes the data, partitions it into smaller regions, and then fits the model based on the data in that region. For this reason, KNN is sometimes referred to as "model free".

How Does it Work?

Let's use an example to understand how the KNN classifier works. We will work with a dataset on penguins which includes a variable recording 3 penguin species as well as other features. For simplicity however, let's assume this dataset has only two variables (bill length and bill depth) in addition to species.



Now let's say that we have to classify a penguin with a bill depth of 16mm and bill length of 40mm - the grid lines in the graph below should help you visualize where the point should be. If we have specified the number of neighbors (k) as 15, KNN will calculate the distance (Euclidean, Manhattan etc.) of this point from all other points and determine the fifteen nearest neighbors. It will then assign the point to the class to which the majority of the nearest neighbors belong. This can be visualized easily using decision boundaries. Decision boundaries separate the feature space into different regions which are then used to classify new points. They are useful because they help us visualize how data instances would be classified for the entire feature space.

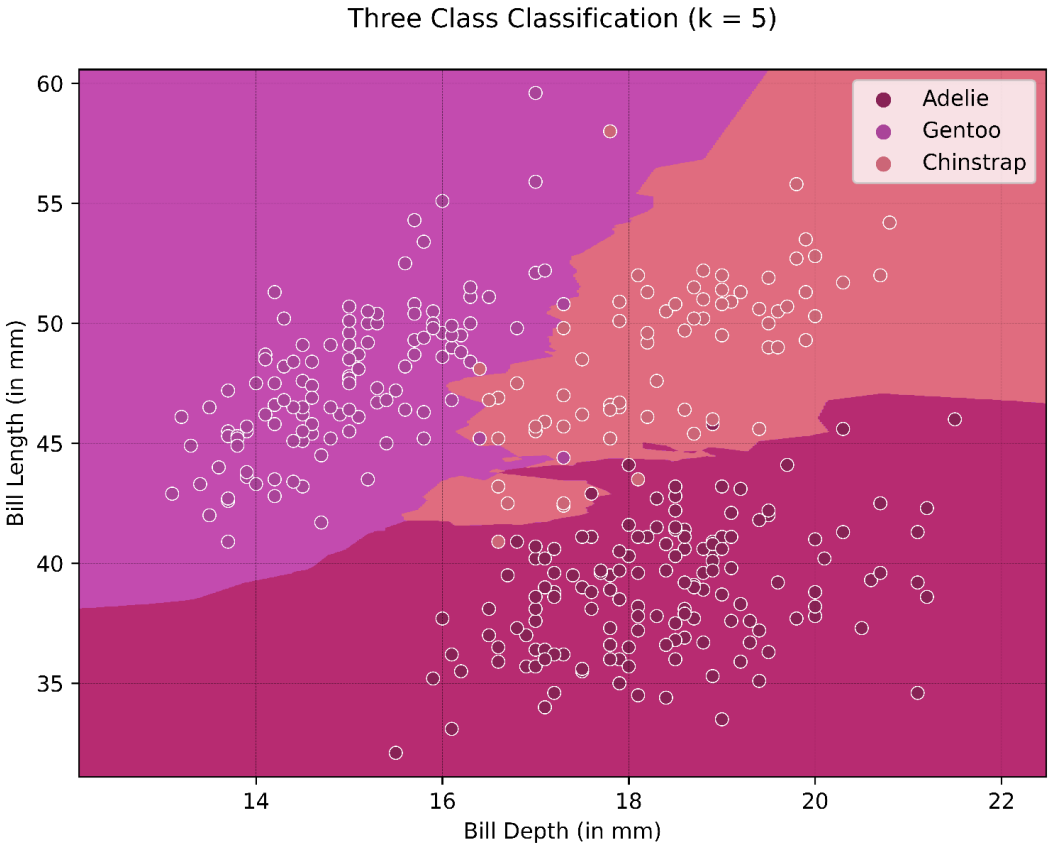


The colors for the graph are obtained from an accessible palette by Paul Tol.

Looking at the decision boundaries above, we can clearly see that our KNN classifier (with k=15) would classify the penguin with a bill depth of 16mm and bill length of 40mm as Adelie. We can also notice misclassifications – there are instances that the KNN classifier would assign to the wrong class.

Decision boundaries change with the distance and weight functions used, as well as the number of neighbors (k) specified. In these examples, Euclidean distance and a uniform weight function

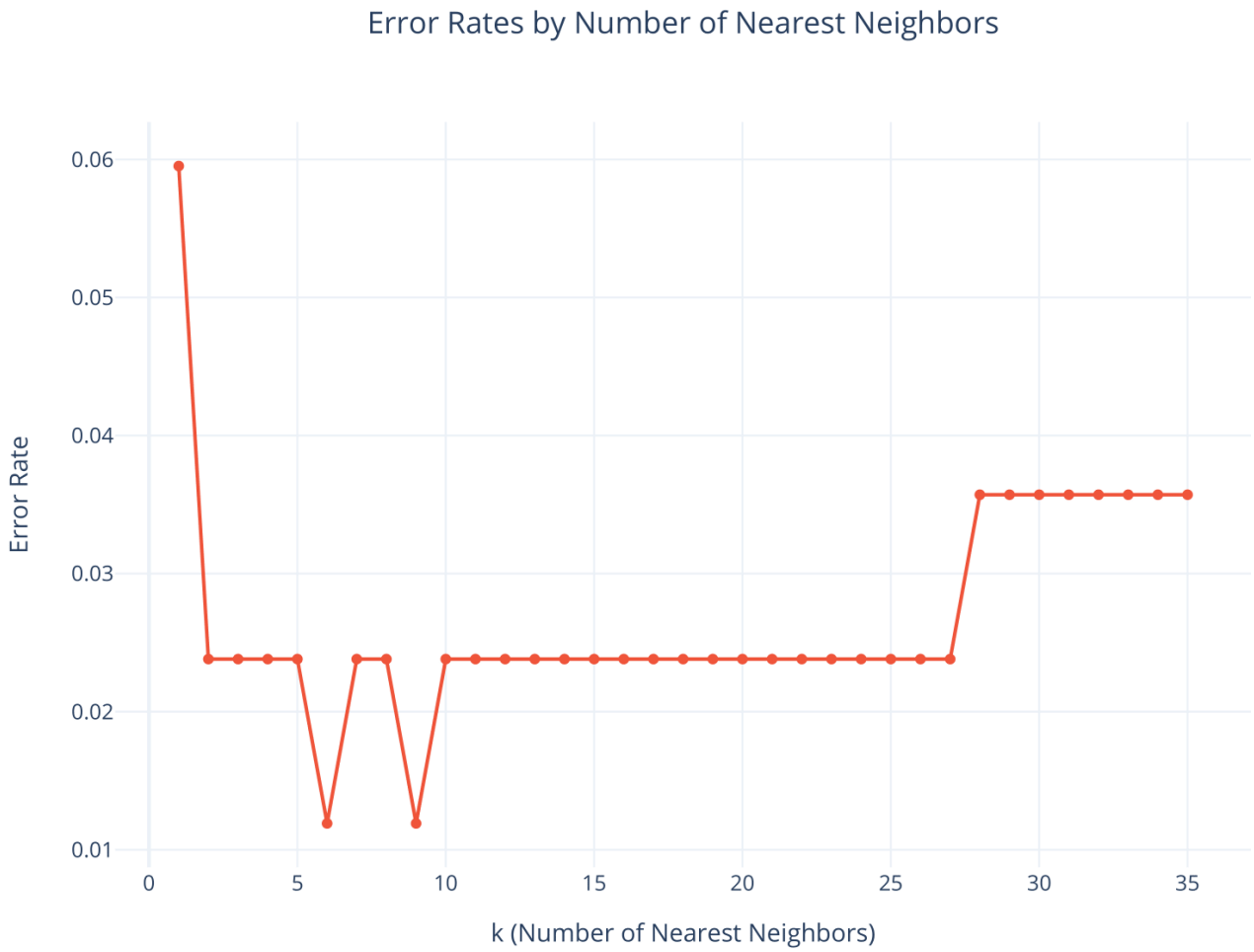
(that assigns an equal weight to all neighbors) are being used. Let's see how the decision boundaries change if we set k to 5 instead of 15.



The decision boundaries are less smooth when k is set to 5. If we set k to 1, every training example will have its own neighborhood, while setting k to the number of instances would make the entire feature space a single neighborhood. With so many options of setting the number of neighbors, how do we find the optimal number i.e. the optimal k?

Finding the Optimal Number of Neighbors

One approach is to determine the error rate for each value of k and choose the number of neighbors that correspond to the lowest error rate. Let's try this with our dataset and visualize the mean error for the predicted values of our test data. Looking at the graph below, we can see that the mean error is the closest to zero when k is either 6 or 9.



Bonus: If you are interested in predicting the confidence scores of the KNN classifier for each point in the test data, you can access the Jupyter Notebook here and play around with the interactive plot (the notebook also contains code for all of the graphs in this report).

References

Palmer Station Antarctica LTER and K. Gorman. 2020. Structural size measurements…, 2007-2009 ver 5. Environmental Data Initiative. (Accessed 2021-11-22). https://doi.org/10.6073/pasta/98b16d7d563f265cb52372c8ca99e60f.

Marine Carpuat. Classification with Nearest Neighbors. http://www.cs.umd.edu/class/spring2017/cmsc422/slides0101/lecture04.pdf.

Scikit-learn. Nearest Neighbors Classification. https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html#sphx-glr-auto-examples-neighbors-plot-classification-py.

Scott Robinson. K-Nearest Neighbors Algorithm in Python and Scikit-Learn. https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/.