Microsoft: DAT210x Programming with Python for Data Science

Bookmarks

- Start Here
- 1. The Big Picture
- 2. Data And Features
- 3. Exploring Data
- 4. Transforming Data
- **5. Data Modeling**

Lecture: Clustering
Quiz

Lab: Clustering
Lab

Lecture: Splitting Data
Quiz

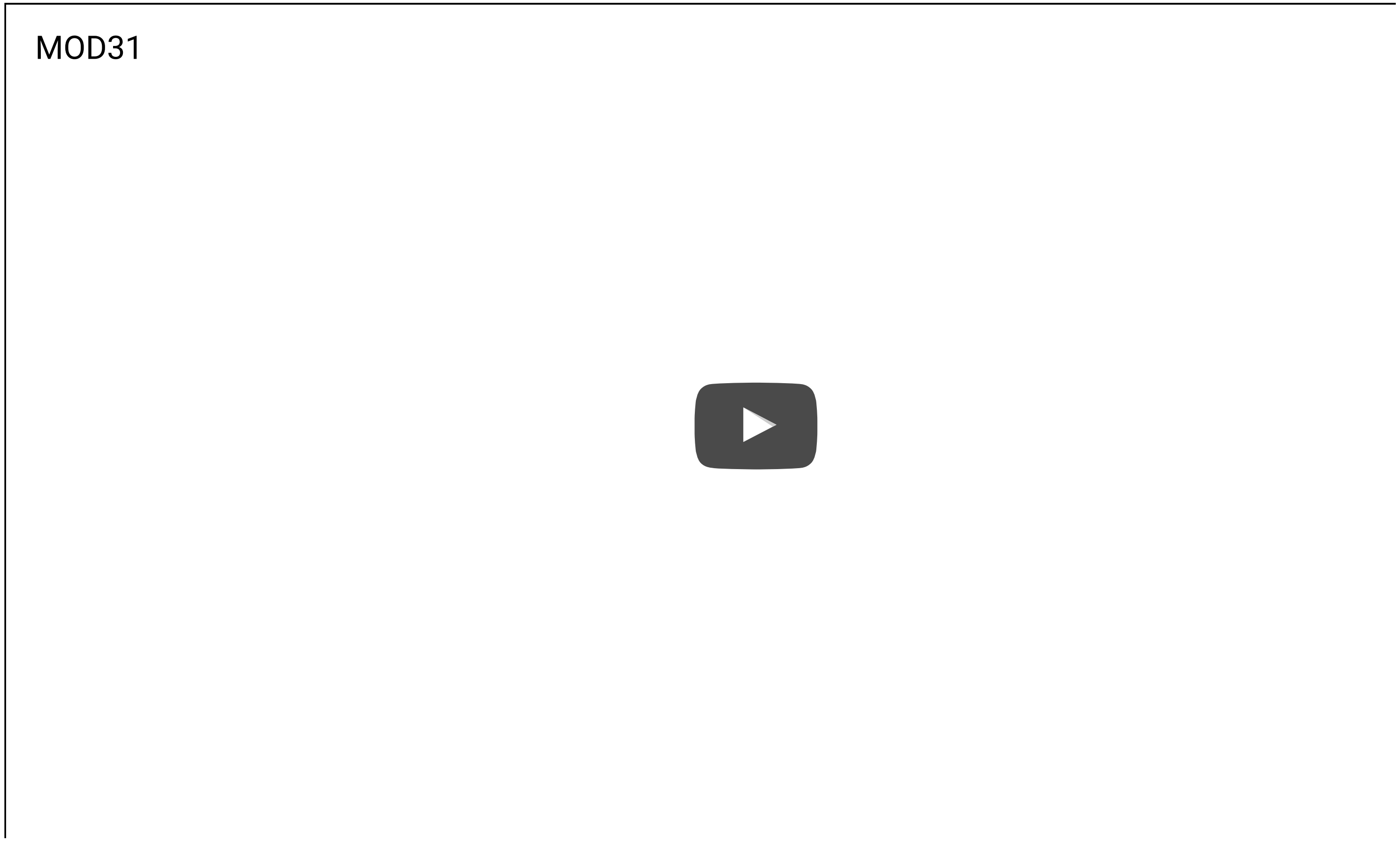
Lecture: K-Nearest Neighbors
Quiz

Lab: K-Nearest Neighbors

5. Data Modeling > Lecture: K-Nearest Neighbors > Video

Bookmark

How Does K-Neighbors Work?



Lab

▶ 0:00 / 0:00 1.0x

Download video Download transcript .srt

You've actually already seen the major portion of the K-Neighbor algorithm in action as an interim step in Isomap's process. Isomap used K-Neighbors in an unsupervised way to build a neighborhood map. The K-Neighbors classifier takes that a step further by applying more logic to actually label new and never-before-seen records.

K-Nearest Neighbors works by first simply storing all of your training data samples.

Then in the future, when you attempt to check the classification of a new, never-before seen sample, it finds the nearest "K" number of samples to it from within your training data. You must have numeric features in order for 'nearest' to be meaningful. There are other methods you can use for categorical features. For example you can use bag of words to vectorize your data. Even so, you may want to experiment with other methods, such as as cosine similarity instead. But at the end of the day, SciKit-Learn's K-Nearest Neighbors only supports numeric features, so you'll have to do whatever has to be done to get your data into that format before proceeding. The distance will be measures as a standard Euclidean

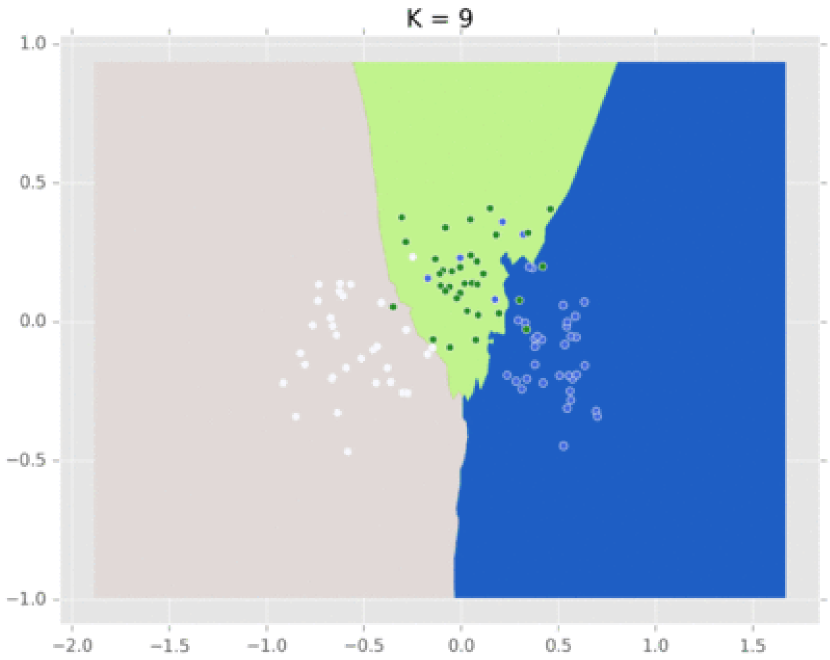
With the nearest neighbors found, K-Neighbors looks at their classes and takes a mode vote to assign a label to the new data point. Further extensions of K-Neighbors can take into account the distance to the samples to weigh their voting power. Each new prediction or classification made, the algorithm has to again find the nearest neighbors to that sample in order to call a vote for it. This process is where a majority of the time is spent, so instead of using brute force to search the training data as if it were stored in a list, tree structures are used instead to optimize the search times. Due to this, the number of *classes* in dataset doesn't have a bearing on its execution speed. Only the number of records in your training data set.

Decision Boundaries

A unique feature of supervised classification algorithms are their decision boundaries, or more generally, their n-dimensional decision surface. These boundaries are somewhat similar to the event horizon on a black hole: whenever performing classification, there exist a threshold or region where if superseded, will result in your sample being assigned that class.

The decision surface isn't always spherical. In fact, it can take many different types of shapes depending on the algorithm that generated it. Some decision boundaries very linear and take the form of a hyperplane. Others are twisted and contorted and look like strung up swiss cheese. In the following labs, you'll dedicate some time to becoming familiar with the decision surface options for each classification method you cover.

For K-Neighbors, generally the higher your "K" value, the smoother and less jittery your decision surface becomes. Higher K values also result in your model providing probabilistic information about the ratio of samples per each class. There is a tradeoff though, as higher K values mean the algorithm is less sensitive to local fluctuations since farther samples are taken into account. This causes it to only model the overall classification function without much attention to detail, and increases the computational complexity of the classification.


K = 9
1.0
0.5
0.0
−0.5
−1.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5