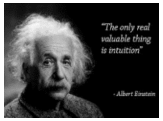
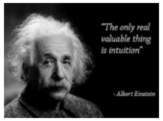
K Nearest Neighbors



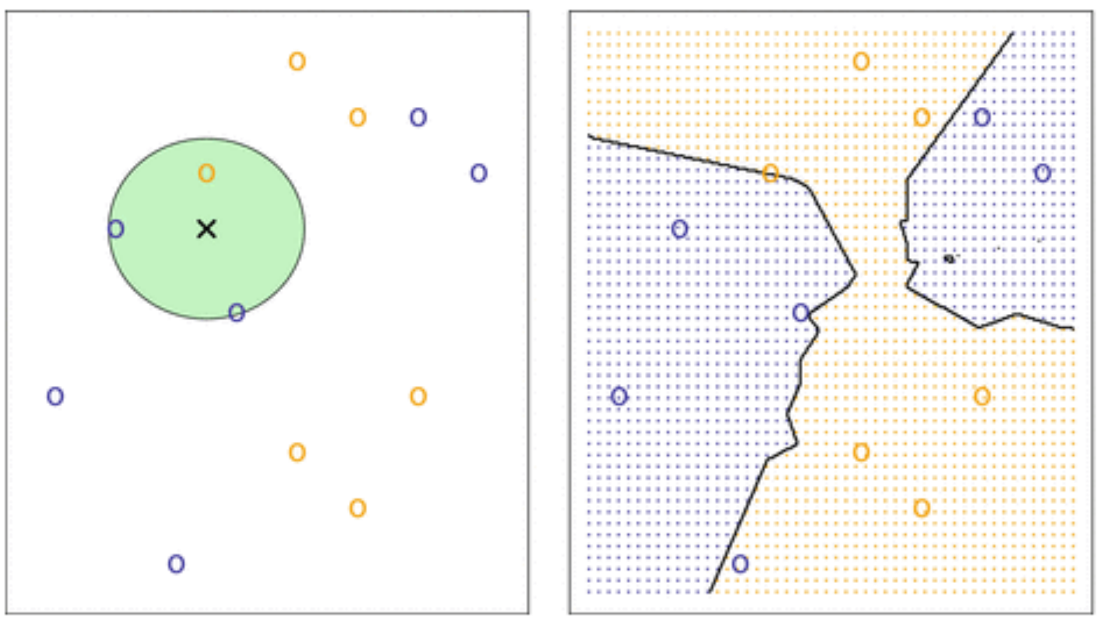
Non-Parametric Models: Intuition

- A **"parametric"** model is one in which we make **assumptions** about the **functional form** of the model (e.g., a model is linear, errors are normally distributed, etc.).
- These assumptions **help** us define parameters (e.g., means, variance, coefficients) to **simplify** the model and its solutions.
- But parameters also **constrain** the solution to those that conform to these parameters (e.g., data points never touch the regression line)
- In contrast, a "non-parametric" model does **not** make such **assumptions** and the models simply try to **get** as **close** as possible to the **data points**.
- For example, if you find **10 teenagers** in a **store** in the mall, you may be predicting that the next customer to walk in will be a teenager, without resorting to regression models or normal distributions.



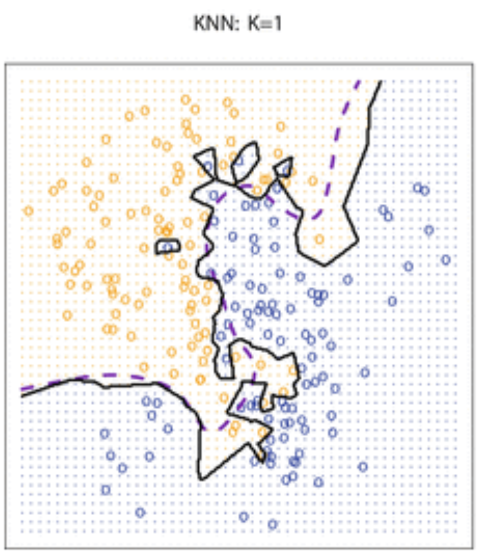
K Nearest Neighbors (KNN): Intuition

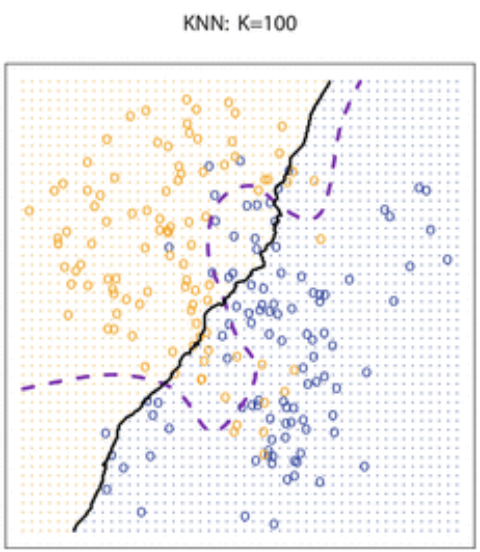
- KNN is a non-parametric method most suitable for data in which we have very little knowledge about its distribution
- KNN simply classifies a test observation based on the classification of the K training closest training points.
- The resulting classification boundary for a K=3 KNN is illustrated below

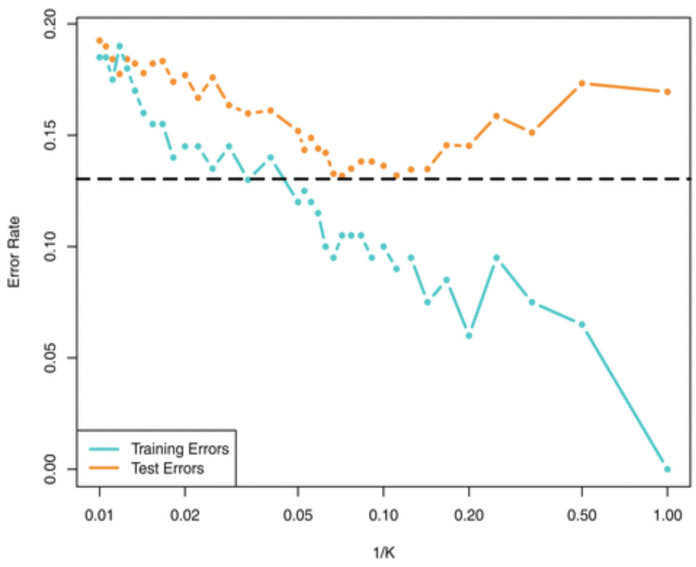


Selecting K

- The solid line in the diagrams below illustrate **KNN** classification boundaries for ***K = 1*** and ***K = 100***
- As the left diagrams show, a **larger *K*** produce **smoother** classification **boundaries**; a **smaller *K*** produce **better training accuracy**, which naturally does **not** ensure **test accuracy**.
- As the diagram on the right illustrates, **training error** tends to **decline** when ***K*** increases (1/K decreases); but the **test error bottoms out** around ***K = 10*** (*1/K = 0.10*)







R Tips

`knn(){class}` → "K-Nearest Neighbors" function in the {class} package to fit KNN models. The `{class}` package has functions to fit various classification models.

One nice feature of the `knn()` function is that it does estimation and prediction in one step. You first need to define the training and test data sets:

`train.x=cbind(x1,x2,etc.)[train,]` → Use the column bind function to create a predictor variable matrix; `[train,]` is the record index defined to select the training data set (see Machine Learning lecture)

`test.x=cbind(x1,x2,etc.)[!train,]` → `[!train,]` Is the record index vector for all records not (!) in the train set, which we use for the test set.

`knn.pred=knn(train.X,test.X,y,k=1)` → y is the classification variable; `k=1` specifies 1 neighbor in the model; use `k=n` to specify `n` neighbors.

Kogod School
of
Business