BIOS 635: Bias and variance trade-off, Classification, K-nearest neighbor

Kevin Donovan

1/26/2021

Review

- Homework 1 assigned, due on 1/28 at 11PM through GitHub Classroom
- Article Evaluation 1 assigned, due on 2/2 through GitHub Classroom
- Office Hours: Wednesday 10-11AM
- Last week: discussed supervised and unsupervised learning, curse of dimensionality, evaluating model performance

Supervised Learning

- Features:

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \dots \\ X_p \end{pmatrix}$$

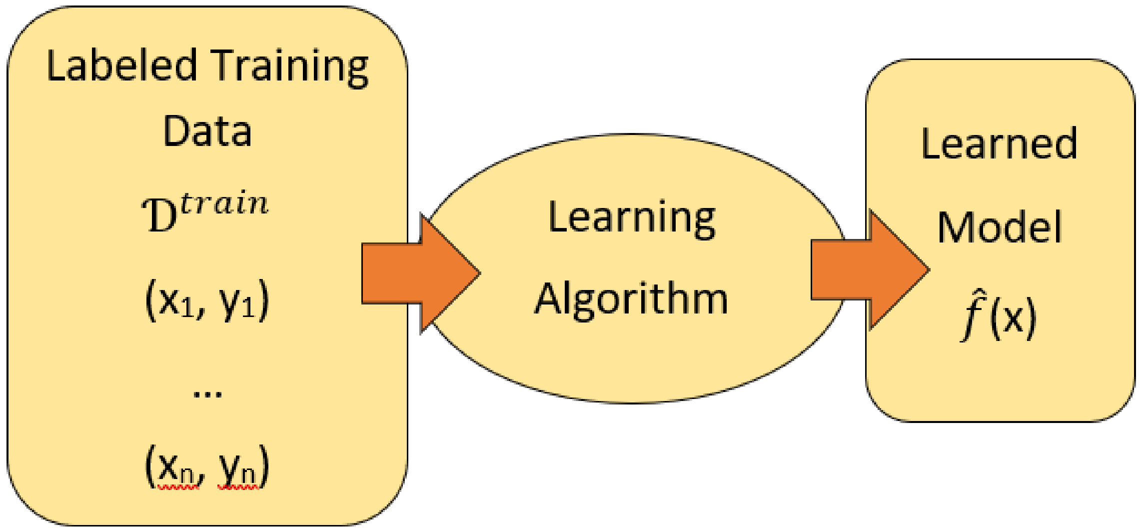
- Model:

$$Y = f(X) + \epsilon$$

- Goals

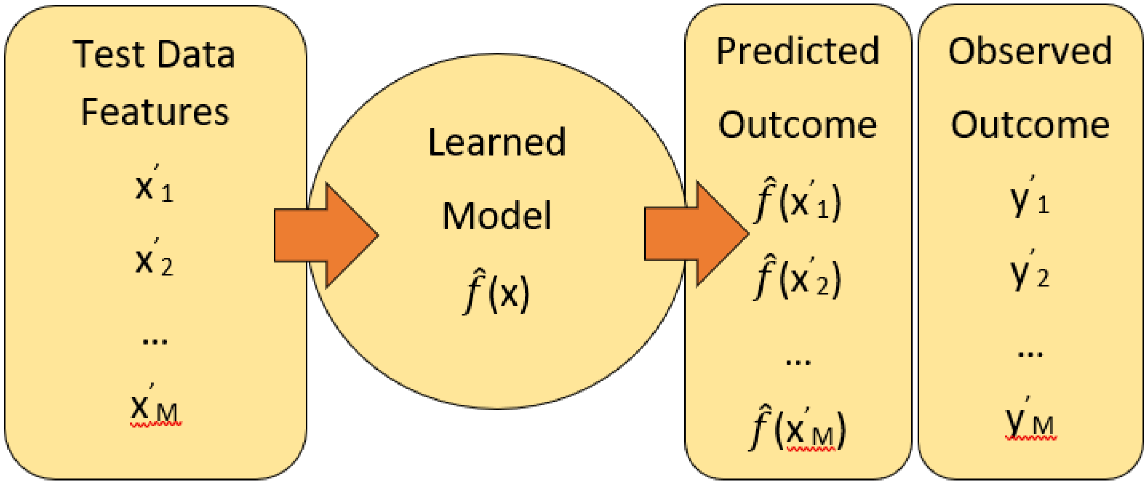
  - *Define* $f(X)$*; e.g.* $f(X) = E(Y|X = x)$
  - *Model and estimate* $f(X)$*, denoted* $\hat{f}(x)$
  - *Define metric to evaluate estimated model; e.g.* $M\hat{S}E(x) = E[(Y - \hat{f}(X))^2|X = x]$

Supervised Learning



- Give learner training data
- Learner returns model $\hat{f}(x)$

Supervised Learning



- Give test data estimated model $\hat{f}(x)$
- Compare predicted outcomes from $\hat{f}(x)$ with observed

Mean Squared Error Decomposition

Recall: MSE for estimate at $X = x$ can be decomposed into

$$MSE_{\hat{f}}(x) = E[(Y - \hat{f}(X))^2 | X = x] = [f(x) - \hat{f}(x)]^2 + Var(\epsilon)$$

Consider taking expectation marginally (i.e., across Y and X).

Can show

$$E[(Y - \hat{f}(X))^2] = E_x[\text{bias}(\hat{f}(x))^2] + E_x[\text{Var}(\hat{f}(x))] + \text{Var}(\epsilon)$$

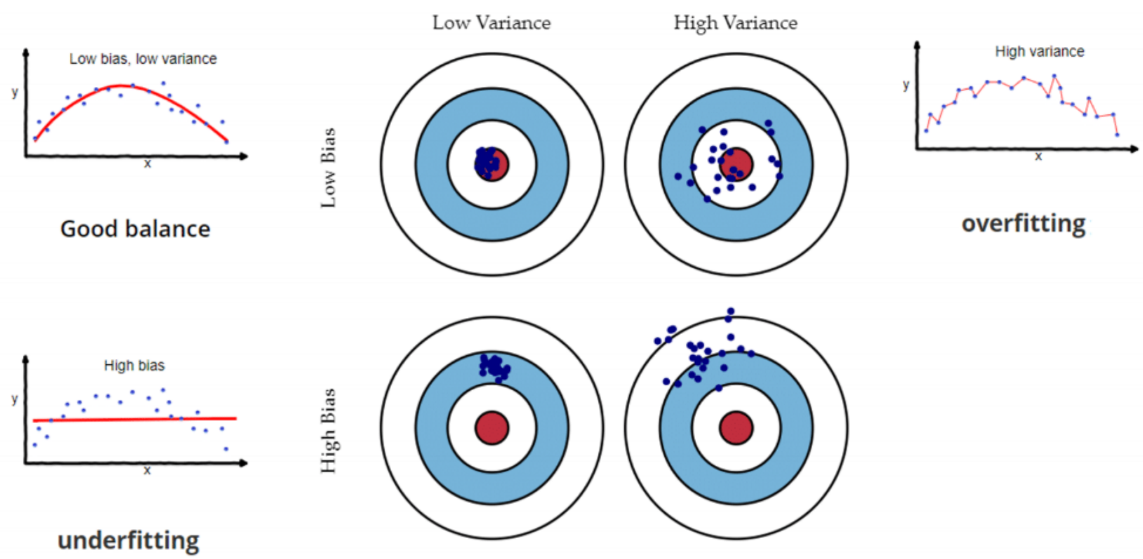
where $\text{bias}(\hat{f}(x)) = \text{E}[\hat{f}(x)] - f(x)$

Bias-variance trade-off

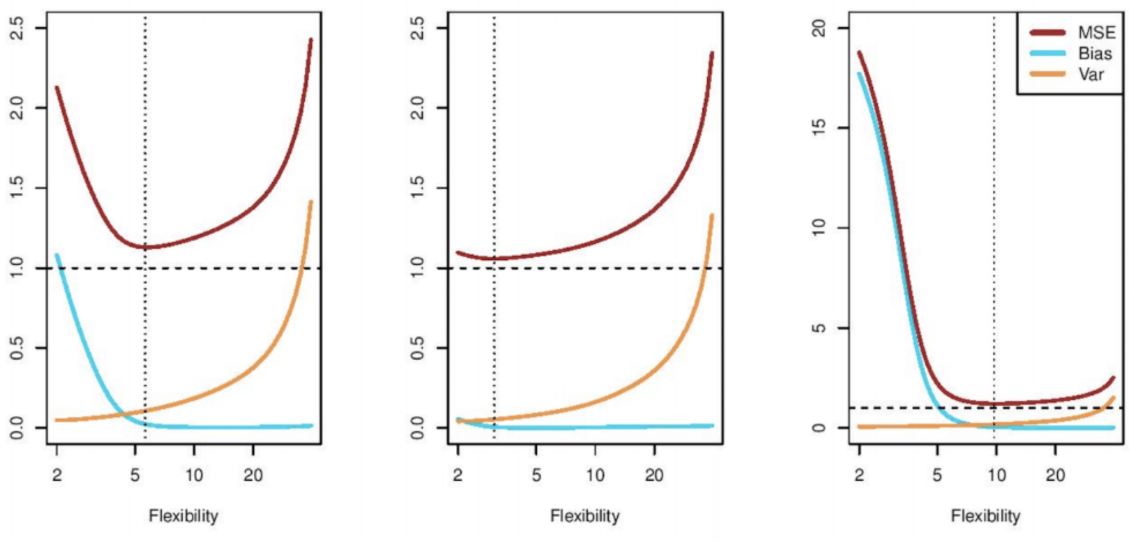
Above means bias **and** variance of model increases expected model error

Creates tradeoff:

- Higher complexity: **decreased** bias but **increased** variance
- Lower complexity: **increased** bias but **decreased** variance



Bias-variance trade-off



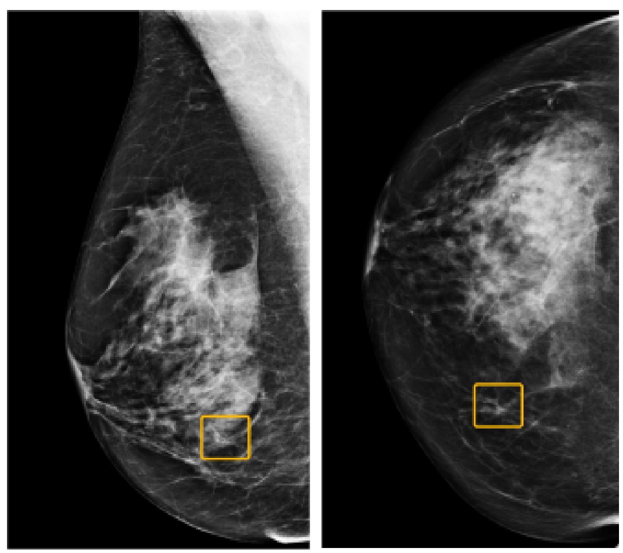
Classification

Suppose instead response Y is categorical

e.g. cancer stage is one of $C = (0, 1, 2, 3, 4)$ where 0 indicates cancer-free

Goals:

- Build classifier $\hat{f}(X)$ that maps a category from C to future observation X
- Assess uncertainty in each classification
- Understand roles of predictions $X = (X_1, X_2, \ldots, X_p)$



Classification

What to model?:

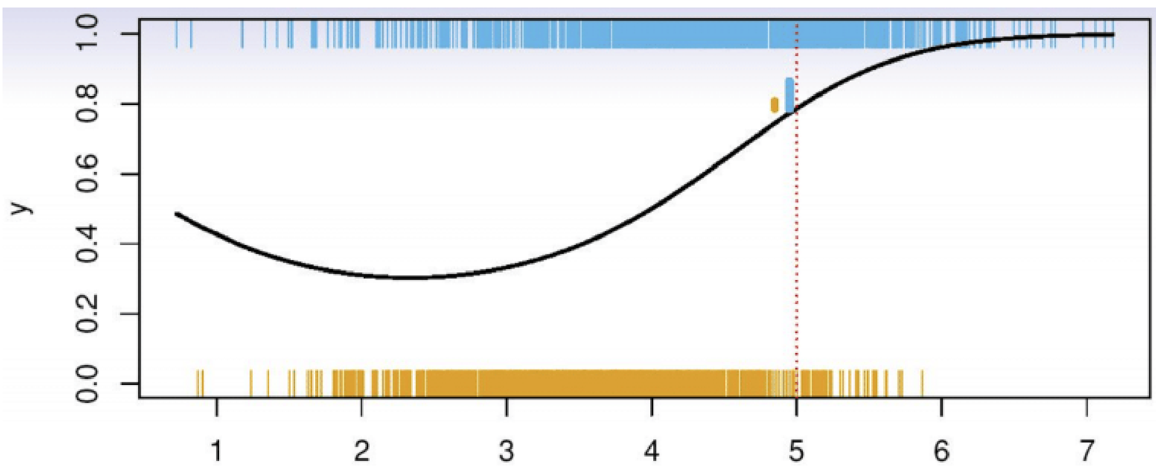
Let $p_k(x) = \Pr(Y = k|X = x)$, $k = 1, 2, \ldots, K$

Denoted as the **conditional class probabilities** at x

If these are known, can define classifier at x by

$f(x) = j$ if $p_j(x) = \max[p_1(x), \ldots, p_K(x)]$

Denoted as the **Bayes optimal classifier** at x



Classification metrics

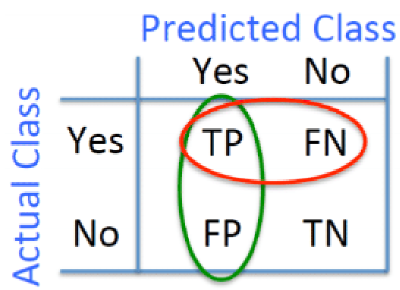
Basic:

$$\text{accuracy} = \frac{\text{\# correct predictions}}{\text{\# test instances}}$$

$$\text{error} = 1 - \text{accuracy}$$

These are in general **not sufficient** (why?)

Confusion Matrix



$$\text{accuracy} = \frac{TP + TN}{P + N}$$

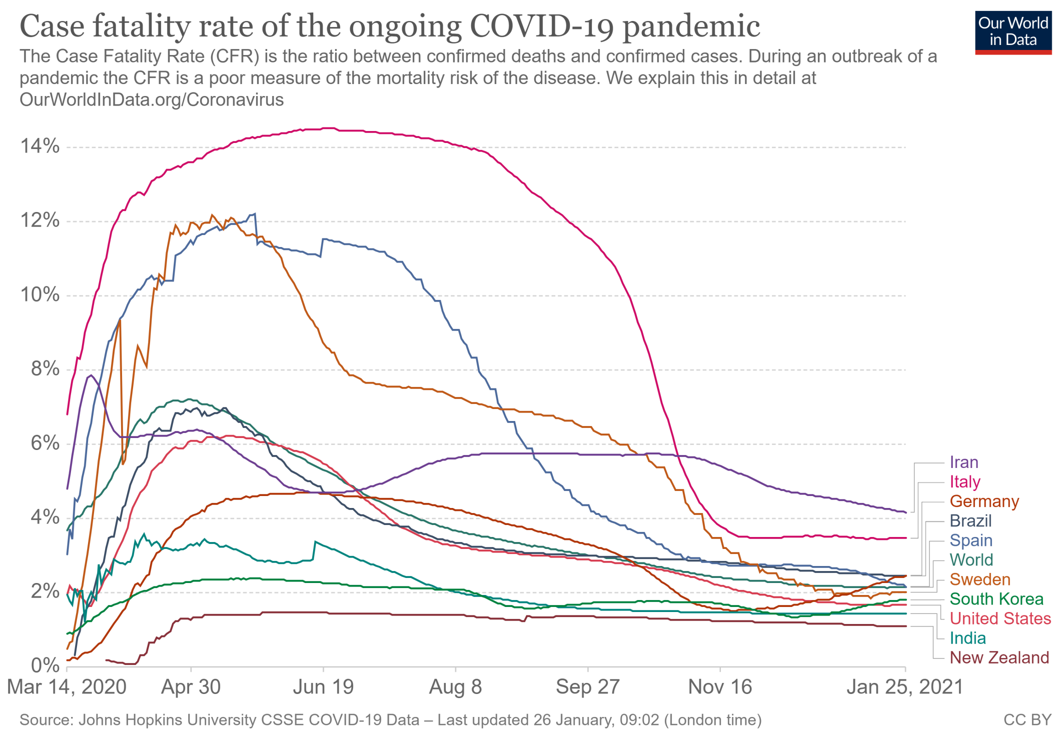
- Want to correctly identify positive **and** negative instances accurately
- For positive instances, have the following metrics:

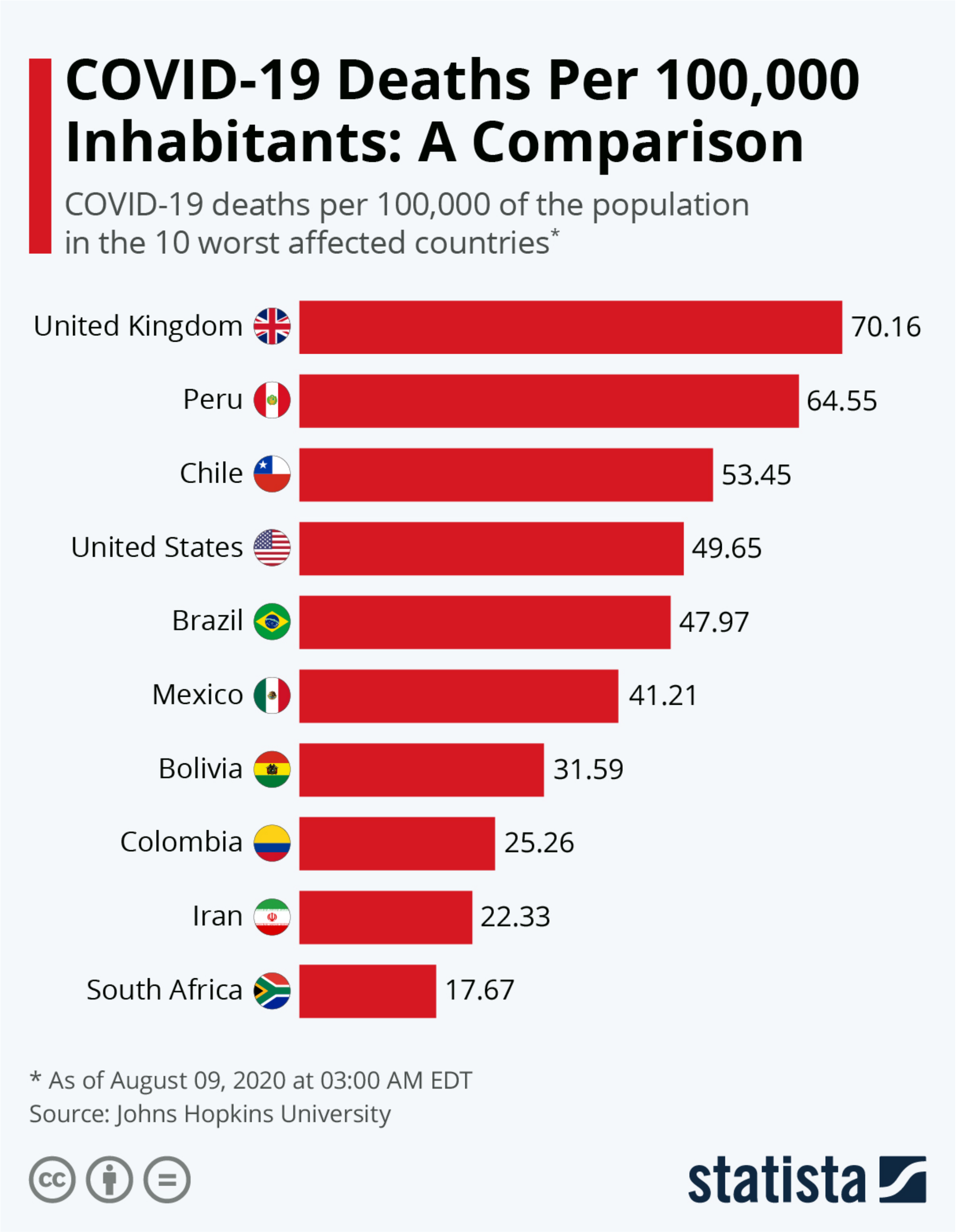
$$PPV = \frac{TP}{TP + FP} \qquad Sensitivity = \frac{TP}{TP + FN} \qquad F_1 = \frac{2}{\frac{1}{PPV} + \frac{1}{Sensitvity}}$$

Confusion Matrix

Example

During the COVID-19 pandemic, different metrics to quantify risk:

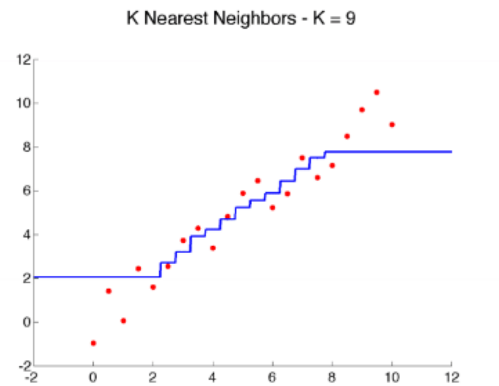


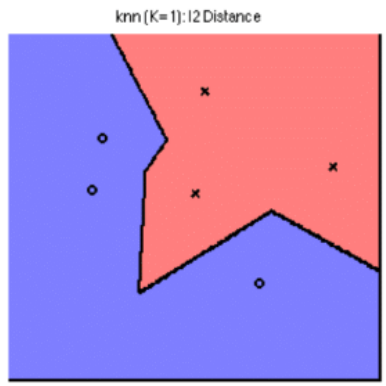

COVID-19 Deaths Per 100,000 Inhabitants: A Comparison
COVID-19 deaths per 100,000 of the population in the 10 worst affected countries*
United Kingdom 70.16
Peru 64.55
Chile 53.45
United States 49.65
Brazil 47.97
Mexico 41.21
Bolivia 31.59
Colombia 25.26
Iran 22.33
South Africa 17.67
* As of August 09, 2020 at 03:00 AM EDT
Source: Johns Hopkins University
statista

K-nearest Neighbor (KNN)

Simple and **flexible** algorithm:

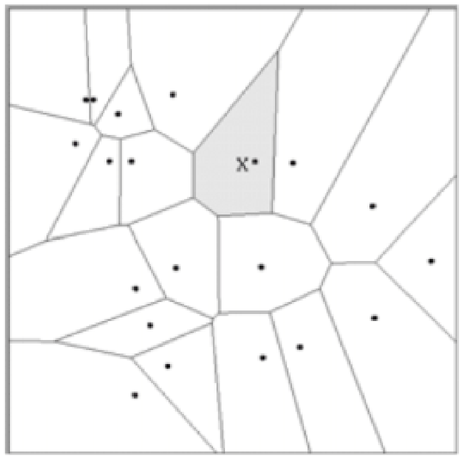
1. Given training data $D = \{\mathbf{x}_i, y_i\}$, distance function $d(\cdot, \cdot)$ and input $\mathbf{x}$
2. Find $\{j_1, \ldots, j_K\}$ closest examples with respect to $d(\mathbf{x}, \cdot)$
   - (regression) if $y \in \mathbf{R}$, return average: $\frac{1}{K}\sum_{k=1}^{K} y_{j_k}$
   - (classification) if $y \in \pm 1$, return majority: $sign(\sum_{k=1}^{K} y_{j_k})$



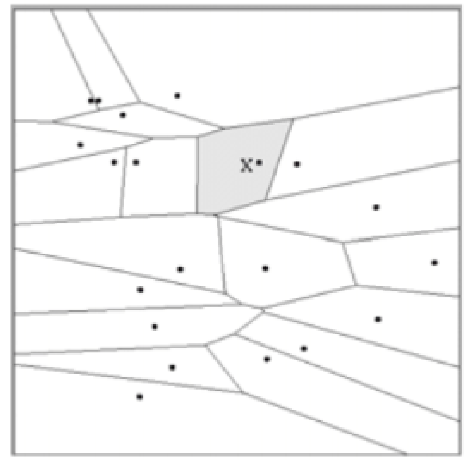


Distance Metrics

- Different metrics can dramatically change neighborhoods:



Dist(**a,b**) $=(a_1 - b_1)^2 + (a_2 - b_2)^2$



Dist(**a,b**) $=(a_1 - b_1)^2 + (3a_2 - 3b_2)^2$

- Standard Euclidean distance metric:
  - *2D:* $\mathrm{Dist}(a, b) = \sqrt{[(a_1 - b_1)^2 + (a_2 - b_2)^2]}$
  - *Multdim:* $\mathrm{Dist}(a, b) = \sqrt{\sum (a_i - b_i)^2}$

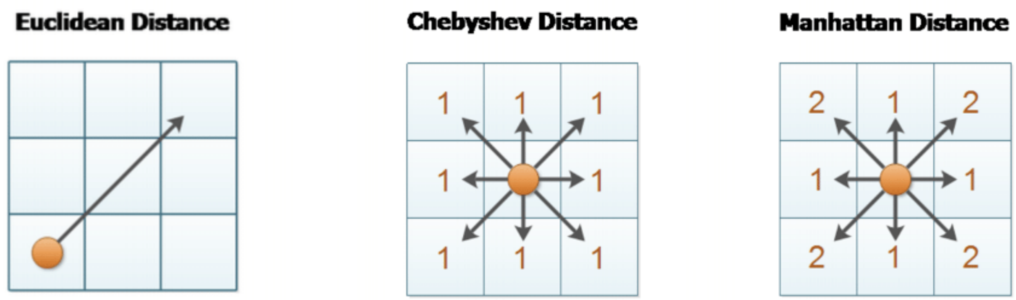
Distance Metrics

$$\mathrm{Dist}(a,b) = \left(\sum_i |a_i - b_i|^p\right)^{1/p}$$

p = 1, Manhattan Distance

p = 2, Euclidean Distance

p = ∞, Chebychev Distance



$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ $\max(|x_1 - x_2|, |y_1 - y_2|)$ $|x_1 - x_2| + |y_1 - y_2|$

KNN Examples

1. Regression

- Suppose we want to predict the number of wins for a NBA team

```
## k-Nearest Neighbors
##
## 30 samples
##  2 predictor
##
## Pre-processing: centered (2), scaled (2)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 30, 30, 30, 30, 30, 30, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    5   4.585041  0.8918501   3.700367
##    7   4.845494  0.8979390   3.972826
##    9   5.044867  0.8978793   4.163350
##   11   5.408245  0.9069323   4.400432
##   13   6.119488  0.8964707   4.995359
##   15   6.908745  0.8930579   5.724303
##   17   7.706909  0.8906881   6.415243
##   19   8.613891  0.8744222   7.197625
##   21   9.406709  0.8592141   7.902947
##   23  10.066420  0.8578698   8.499900
##   25  10.842491  0.7900220   9.237286
##   27  11.829461  0.7020191  10.111490
##   29  12.466729  0.6103163  10.753089
##   31  12.744056        NaN  11.034206
##   33  12.744056        NaN  11.034206
##   35  12.744056        NaN  11.034206
##   37  12.744056        NaN  11.034206
##   39  12.744056        NaN  11.034206
```

```
##   41  12.744056        NaN  11.034206
##   43  12.744056        NaN  11.034206
## 
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 5.
```

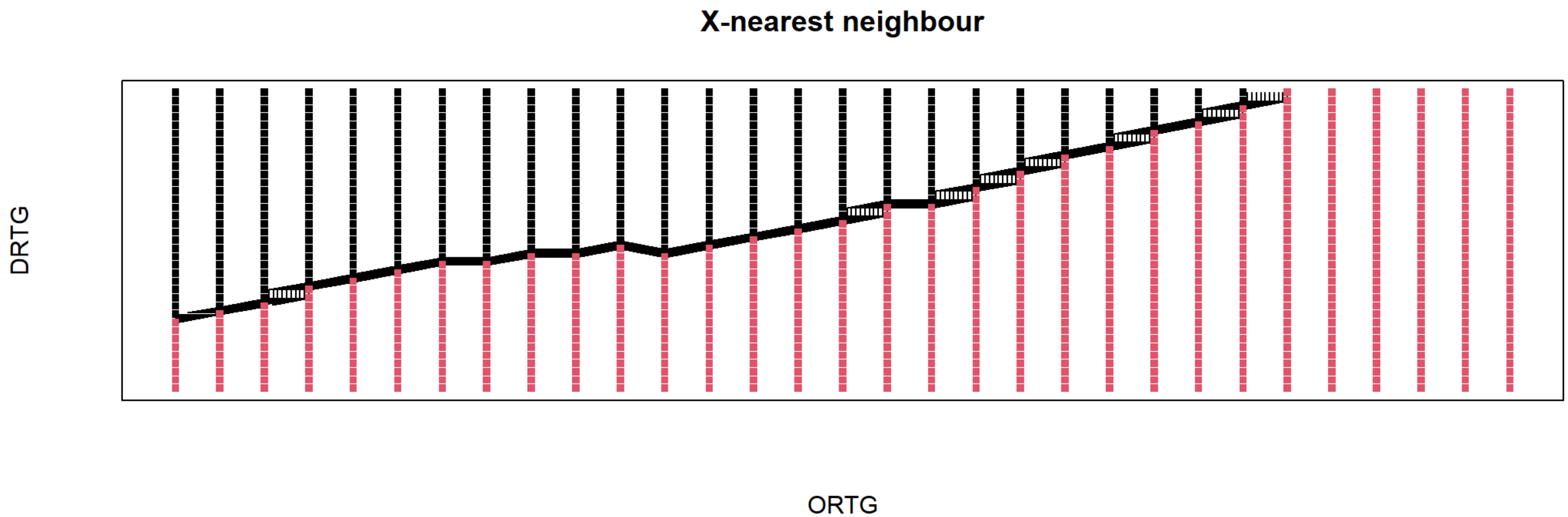
KNN Examples

2. Classification

- Suppose we want to predict whether a NBA team made the playoffs

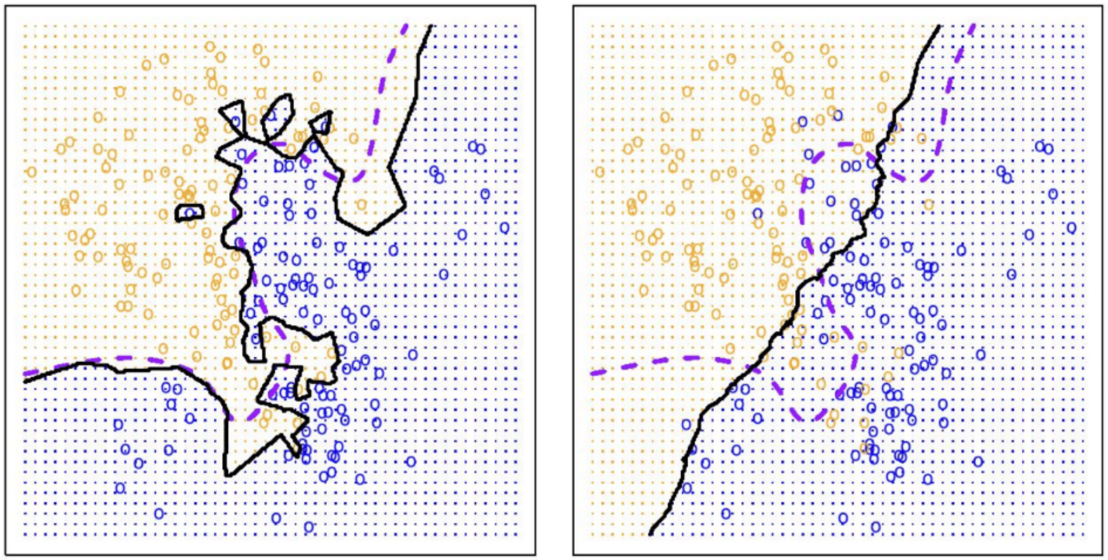
```
## k-Nearest Neighbors
##
## 30 samples
##  2 predictor
##  2 classes: 'no', 'yes'
##
## Pre-processing: centered (2), scaled (2)
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 30, 30, 30, 30, 30, 30, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8625986  0.7151352
##    7  0.8640695  0.7320667
##    9  0.8463797  0.6966267
##   11  0.8348877  0.6892596
##   13  0.8283119  0.6807602
##   15  0.8381627  0.7011166
##   17  0.7941796  0.6165274
##   19  0.7428680  0.5514878
##   21  0.6846963  0.4460085
##   23  0.6910537  0.4625074
##   25  0.5882703  0.2921750
##   27  0.4964599  0.1476145
##   29  0.4181164  0.0000000
##   31  0.4181164  0.0000000
##   33  0.4181164  0.0000000
##   35  0.4181164  0.0000000
##   37  0.4181164  0.0000000
```

```
##   39  0.4181164  0.0000000
##   41  0.4181164  0.0000000
##   43  0.4181164  0.0000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```



KNN Examples

KNN: K=1 KNN: K=100



Train and Test Error