k-Nearest Neighbor Learning

DS2500: Intermediate Programming with Data

What is machine learning?

- Can we really make our machines (computers) learn?
- "Secret sauce" is **data, and lots of it**
- **Rather than programming expertise into our applications, we program them to learn from data**
- Build working machine-learning models then use them to make **remarkably accurate predictions**

Prediction

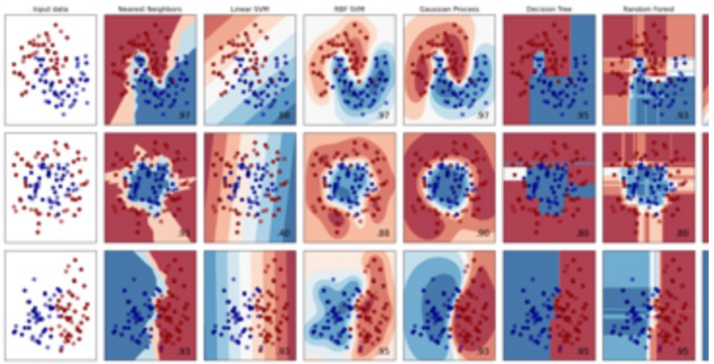
- Improve **weather forecasting** to save lives, minimize injuries and property damage
- Improve **cancer diagnoses** and **treatment regimens** to save lives
- Improve **business forecasts** to maximize profits and secure people's jobs
- **Detect fraudulent credit-card purchases** and **insurance claims**
- Predict **customer "churn"**, what prices houses are likely to sell for, ticket sales of new movies, and anticipated revenue of new products and services
- Predict the **best strategies for coaches and players** to use to win more games and championships
- All of these kinds of predictions are happening today with machine learning.

Machine Learning Approaches

Classification

Identifying which category an object belongs to.

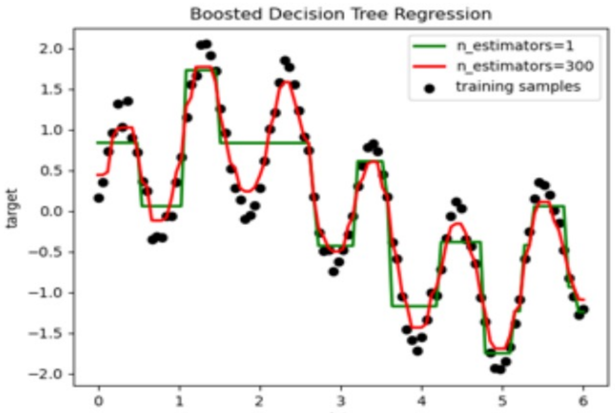
Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

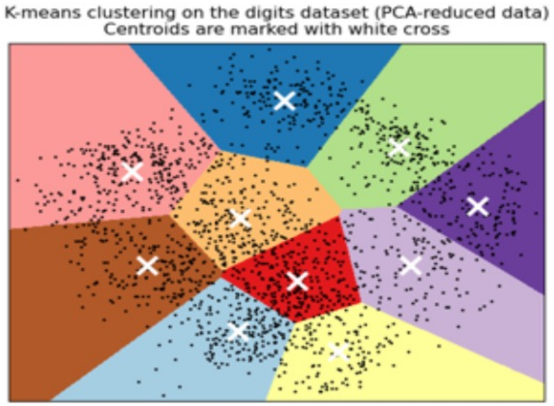
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...



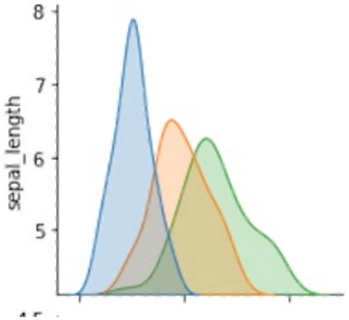
Popular Machine Learning Applications

Anomaly detection	Data mining social media (like Facebook, Twitter, LinkedIn)	Predict mortgage loan defaults
Chatbots	Detecting objects in scenes	Natural language translation (English to Spanish, French to Japanese, etc.)
Classifying emails as spam or not spam	Detecting patterns in data	Recommender systems ("people who bought this product also bought…")
Classifying news articles as sports, financial, politics, etc.	Diagnostic medicine	Self-Driving cars (more generally, autonomous vehicles)
Computer vision and image classification	Facial recognition	Sentiment analysis (like classifying movie reviews as positive, negative or neutral)
Credit-card fraud detection	Handwriting recognition	Spam filtering
Customer churn prediction	Insurance fraud detection	Time series predictions like stock-price forecasting and weather forecasting
Data compression	Intrusion detection in computer networks	Voice recognition
Data exploration	Marketing: Divide customers into clusters	

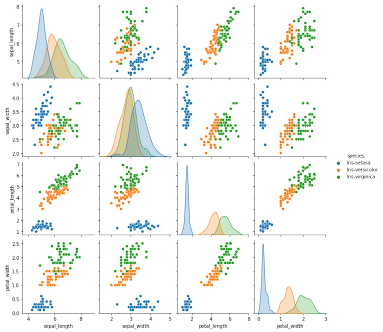
Machine Learning Recipe

```
5.1,3.8,1.6,0.2,Iris-setosa
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
```

load data

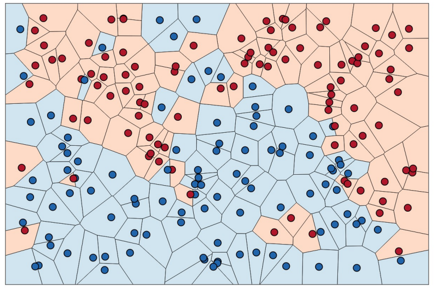


explore



transform

training & test split



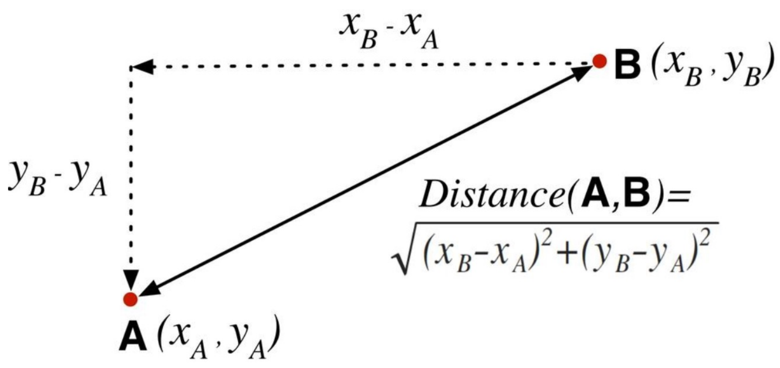
model building

Some frequently used distance functions.	
Camberra: $d(x,y)=\sum_{i=1}^{m}\frac{\lvert x_i-y_i\rvert}{\lvert x_i+y_i\rvert}$ (2)	Euclidean: $d(x,y)=\sqrt{\sum_{i=1}^{m}(x_i-y_i)^2}$ (5)
Minkowsky: $d(x,y)=\left(\sum_{i=1}^{m}\lvert x_i-y_i\rvert^r\right)^{1/r}$ (3)	Manhattan / city - block: $d(x,y)=\sum_{i=1}^{m}\lvert x_i-y_i\rvert$ (6)
Chebychev: $d(x,y)=\max_{i=1}^{m}\lvert x_i-y_i\rvert$ (4)	

model tuning

Euclidean Distance

Attribute	Person A	Person B
Age	23	40
Years at current address	2	10
Residential status (1=Owner, 2=Renter, 3=Other)	2	1



$$d(A, B) = \sqrt{(23-40)^2 + (2-10)^2 + (2-1)^2} \approx 18.8$$

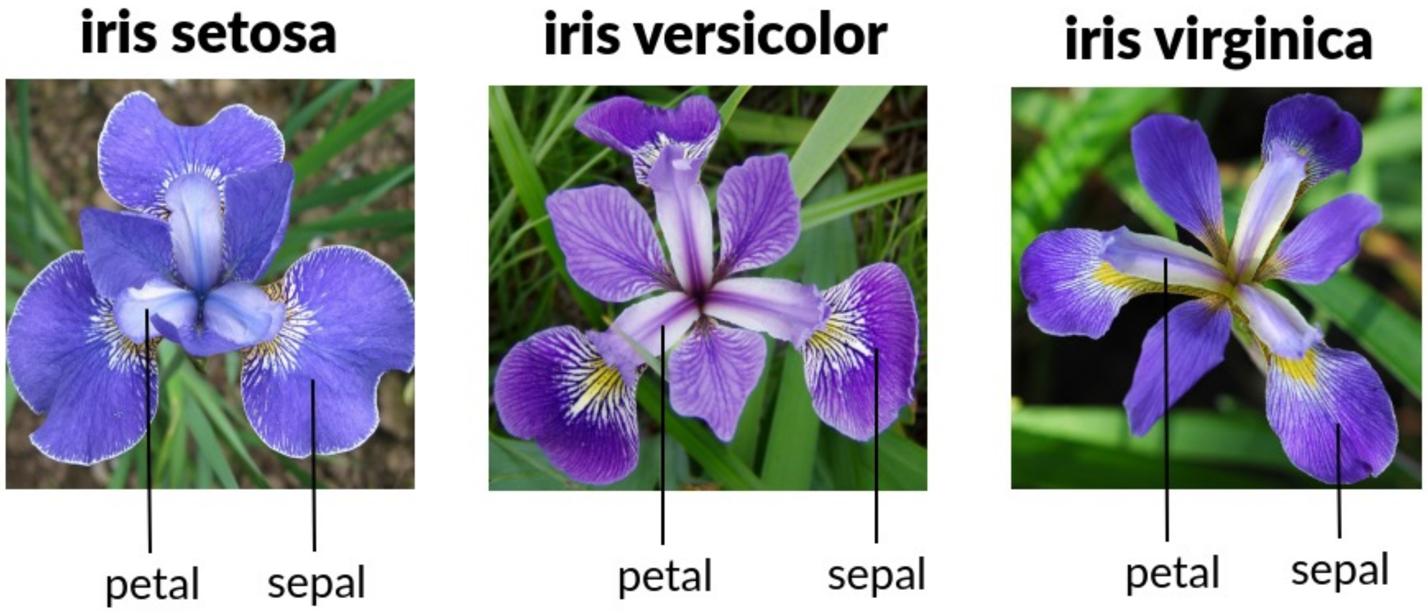
Distance metrics

Some frequently used distance functions.	
Camberra : $$d(x,y)=\sum_{i=1}^{m}\frac{\lvert x_i-y_i\rvert}{\lvert x_i+y_i\rvert} \quad (2)$$	Euclidean : $$d(x,y)=\sqrt{\sum_{i=1}^{m}(x_i-y_i)^2} \quad (5)$$
Minkowsky : $$d(x,y)=\left(\sum_{i=1}^{m}\lvert x_i-y_i\rvert^r\right)^{1/r} \quad (3)$$	Manhattan / city - block : $$d(x,y)=\sum_{i=1}^{m}\lvert x_i-y_i\rvert \quad (6)$$
Chebychev : $$d(x,y)=\max_{i=1}^{m}\lvert x_i-y_i\rvert \quad (4)$$	

Euclidean Distance on Iris Dataset

$$\text{diff} = \sqrt{(\Delta Slen)^2 + (\Delta Swid)^2 + (\Delta Plen)^2 + (\Delta Pwid)^2}$$

Slen = Sepal Length
Swid = Sepal Width
Plen = Petal Length
Pwid = Petal Width



A more general distance measure

$$\text{diff} = w_1|\Delta A_1|^r + w_2|\Delta A_2|^r + \ldots + w_n|\Delta A_n|^r$$

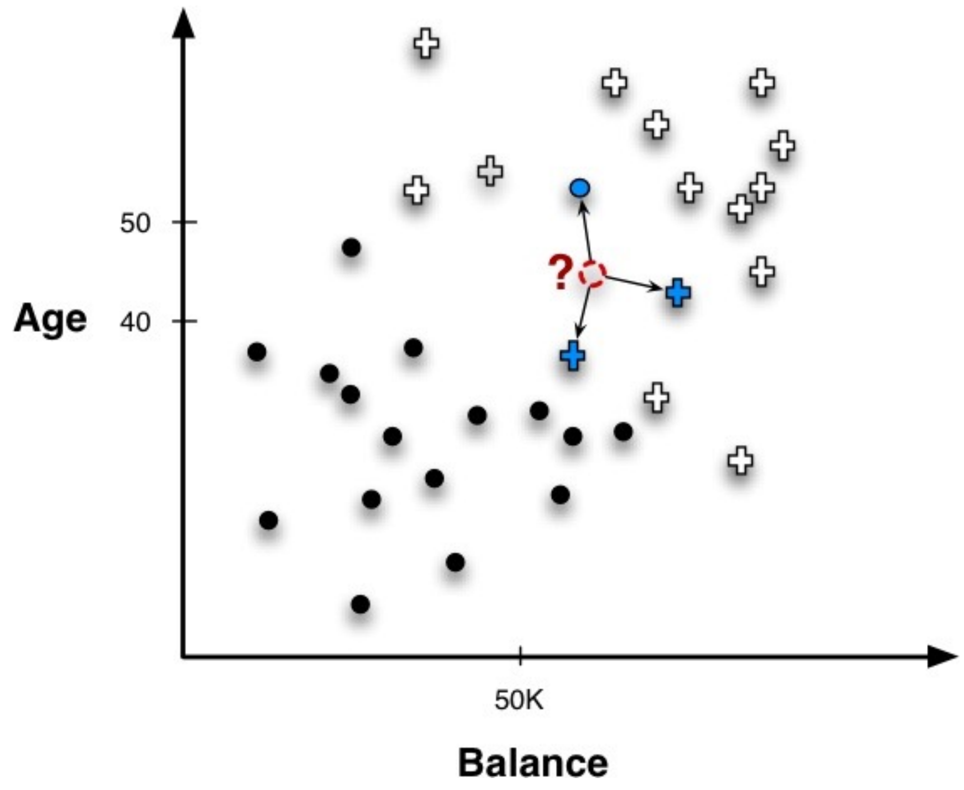
where

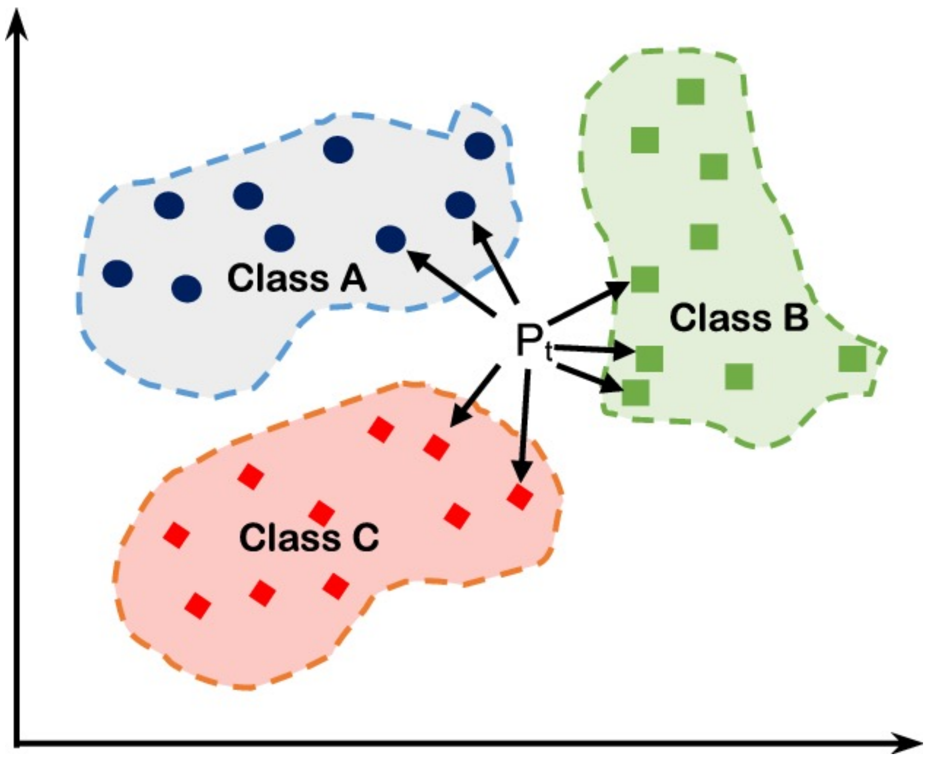
ΔA_i is the difference with respect to feature i,

w_k is a feature weighting factor (0.0 to 1.0), and

r is an exponent (> 0.0)

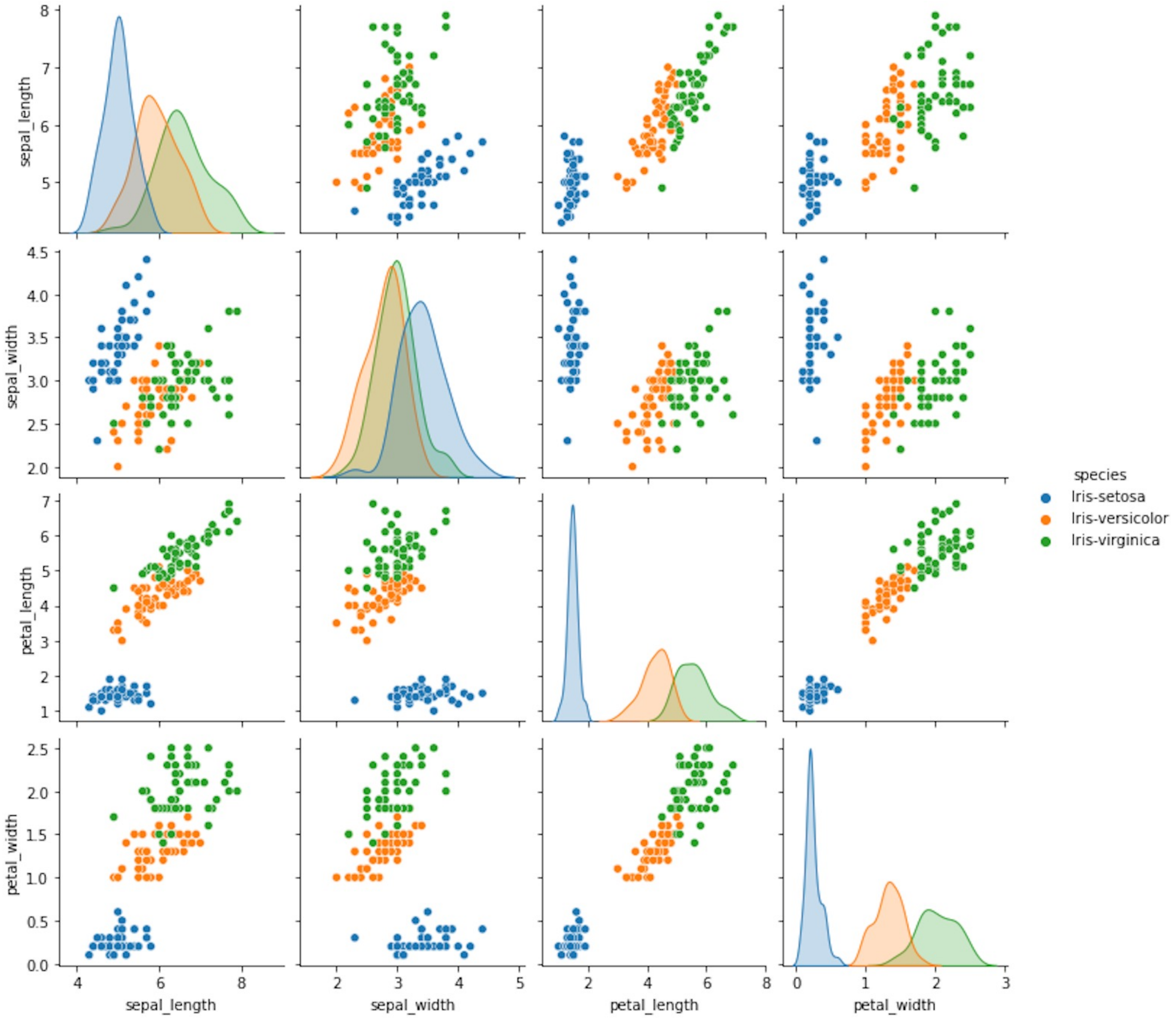
Multiple classes



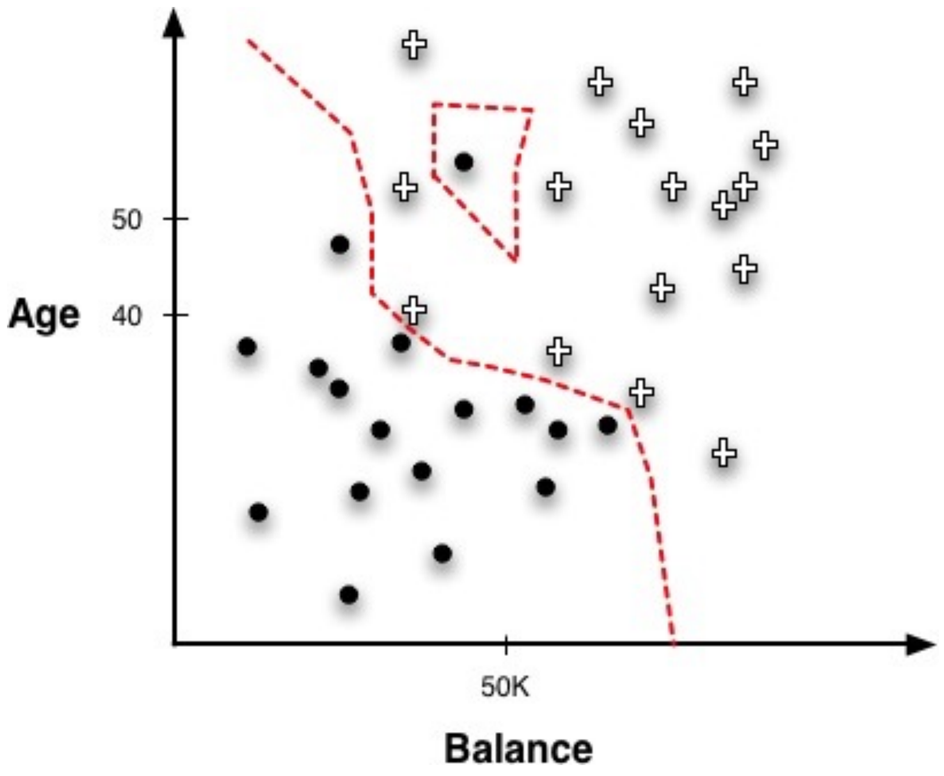


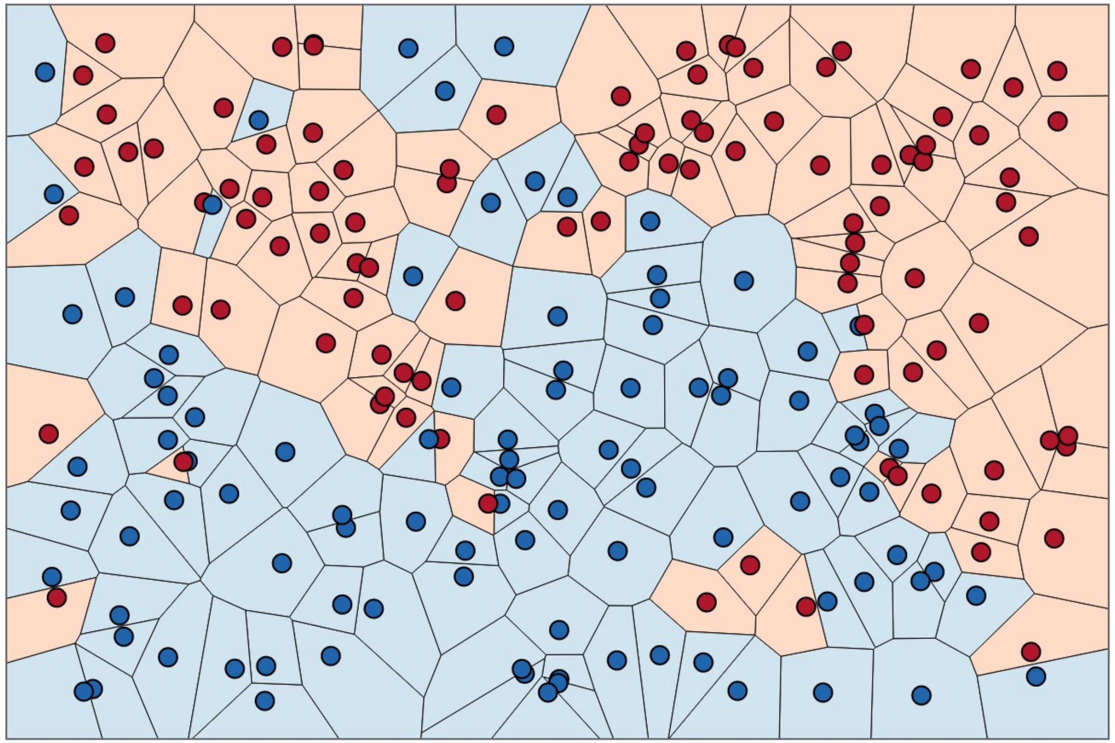
Source: https://laptrinhx.com/k-nearest-neighbors-unlocked-454254569/

Why 1-NN makes mistakes on the iris dataset

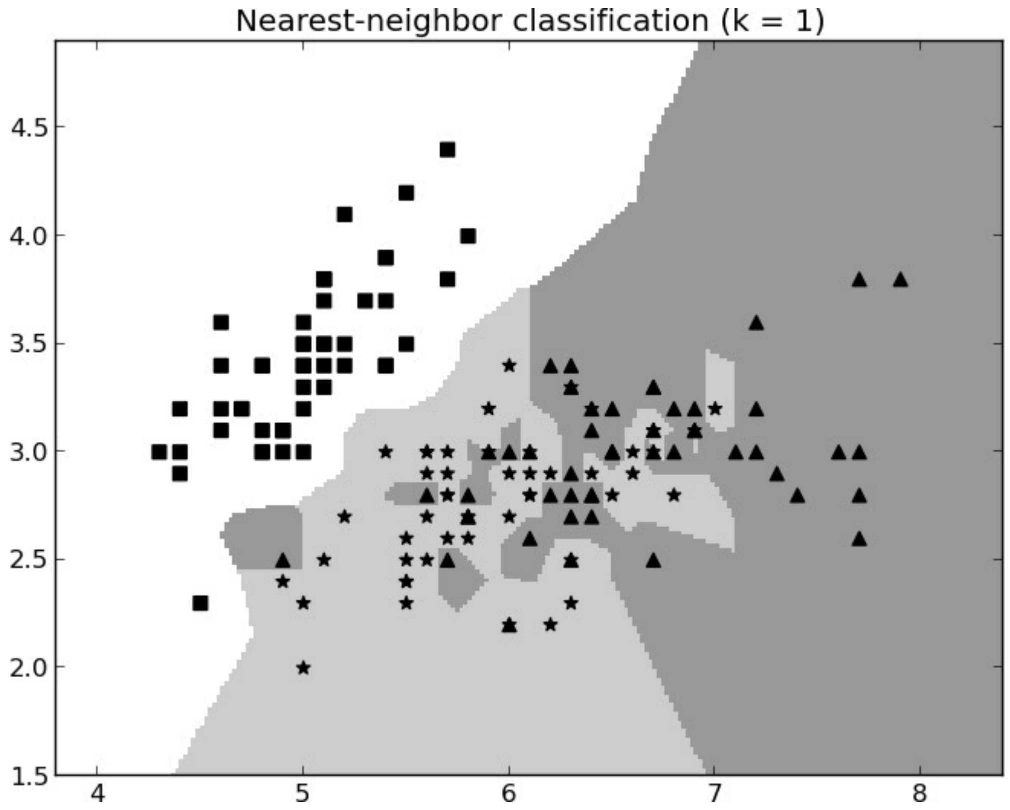


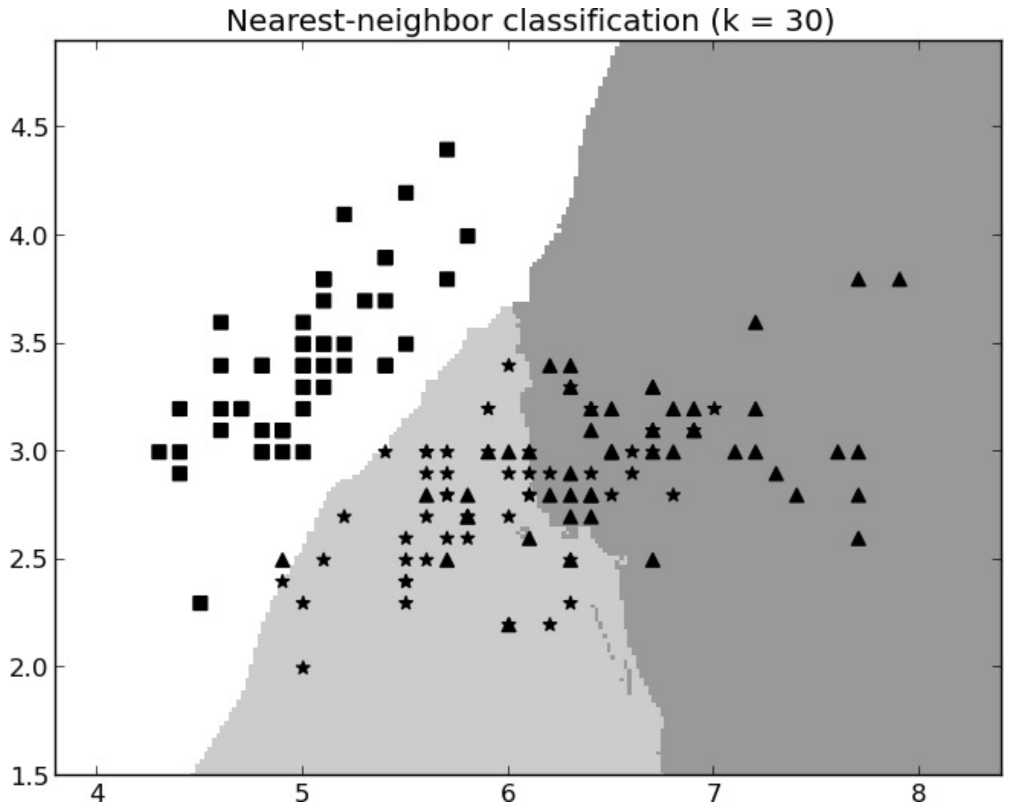
Overfitting





Justifying the k-value





Intelligibility

"The movie Billy Elliot was recommended based on your interest in Amadeus, The Constant Gardener and Little Miss Sunshine"

Amazon: Customers with similar searches purchased….
Amazon: Related to Items You've Viewed…..

We declined your mortgage application because you remind us of the Smiths and the Mitchells, who both defaulted."

Efficiency Issues

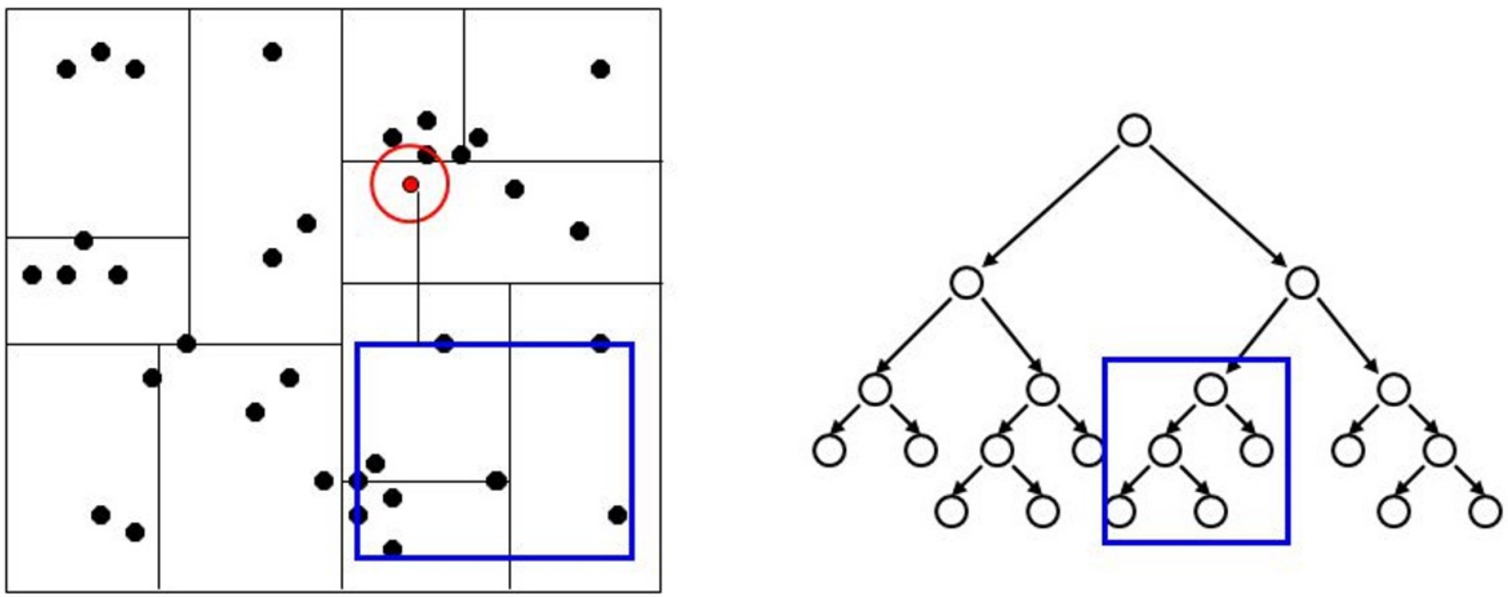
Training: very fast (simply store the instances)

Classification: Finding k nearest neighbors requires computing distance to each instance, and sorting → **O**(n log n) for each instance.

Reduced to **O**(kn) ~ **O**(n) if k is small using selection by partial sorting.

```
function select(list[1..n], k)
    for i from 1 to k
        minIndex = i
        minValue = list[i]
        for j from i+1 to n do
            if list[j] < minValue then
                minIndex = j
                minValue = list[j]
                swap list[i] and list[minIndex]
    return list[k]
```

KD-Trees



Using the distance bounds and the bounds of the data below each node, we can prune parts of the tree that could NOT include the nearest neighbor.

Heterogeneous Attributes

Attribute	Person A	Person B
Sex	Male	Female
Age	23	40
Years at current address	2	10
Residential status (1=Owner, 2=Renter, 3=Other)	2	1
Income	50,000	90,000

Handling categorical attributes:

1. Ignore?
2. Convert to numbers?
3. Cosine similarity?

The k-NN algorithm

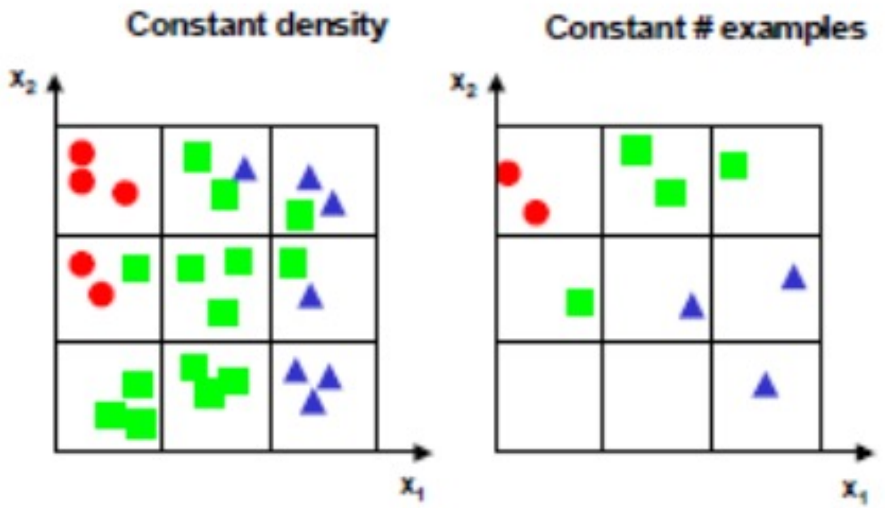
Algorithm:

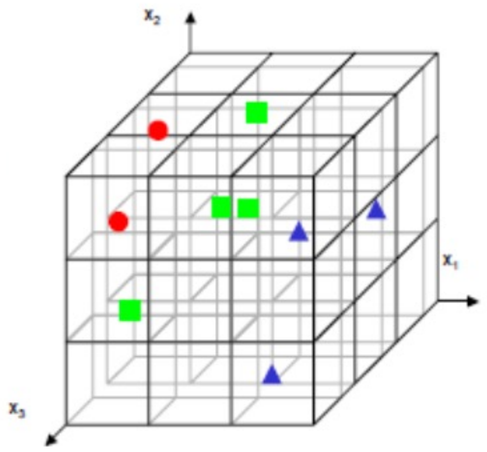
Suppose you have a training data of size D with d samples, and t is a new test sample.

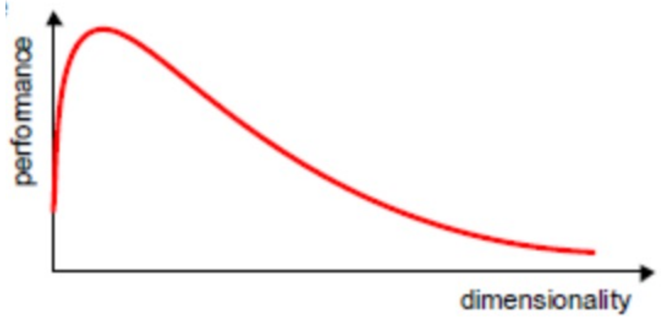
1. Select a value of K, which is the value of the nearest neighbors to be used in the computation of the algorithm.
2. For i=0 to d; find the distance of the test point from every point in the training data set.
3. Make a set S of the K smallest distances.
4. Return the majority label from set S.

Curse of Dimensionality

Since all attributes contribute to the distance calculations, instance similarity can be confused and misled by the presence of too many irrelevant attributes.







Precision and Recall / Sensitivity and Specificity

10 photos, 7 dogs

Perfect Classification:

		True/Actual	
		Positive (🐶)	Negative
Predicted	Positive (🐶)	7	0
	Negative	0	3

Source: https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2

Imperfect Classification

		True/Actual	
		Positive (🐶)	Negative
Predicted	Positive (🐶)	5 (TP)	1 (FP) Type I Error
	Negative	2 (FN) Type II Error	2 (TN)

Accuracy: What proportion of photos are correctly classified?

Accuracy = (TP + TN) / (TP + TN + FP + FN) = 7 / 10 = 70.0%

Imperfect Classification: Sensitivity

Sensitivity: Identify all the dogs!

		True/Actual	
		Positive (🐶)	Negative
Predicted	Positive (🐶)	5 (TP)	1 (FP) Type I Error
	Negative	2 (FN) Type II Error	2 (TN)

Sensitivity: What proportion of **dogs (*positives*)** are identified?

Sensitivity *= TP / (TP + FN) = TP / T = 5 / (5 + 2) = 71.4%*

a.k.a. ***Recall***

Imperfect Classification: Specificity

Specificity: Don't classify cats as dogs!

		True/Actual	
		Positive (🐶)	Negative
Predicted	Positive (🐶)	5 (TP)	1 (FP) Type I Error
	Negative	2 (FN) Type II Error	2 (TN)

What proportion of the **cats** (negatives) are identified?

Specificity *= TN / (TN + FP) = 2 / (1 + 2) = 66.7%*

Specificity = 1 – False Positive Rate

Imperfect Classification

Precision: Identify dogs and *only* dogs

		True/Actual	
		Positive ( )	Negative
Predicted	Positive ( )	5 (TP)	1 (FP) Type I Error
	Negative	2 (FN) Type II Error	2 (TN)

Precision: What proportion of ***predicted dogs (positives)*** are truly dogs?

Precision = TP / (TP + FP) = TP / P = 5 / (5 + 1) = 83.3%

Multiple classes

		True/Actual		
		Cat	Fish	Hen
Predicted	Cat	4	6	3
	Fish	1	2	0
	Hen	1	2	6

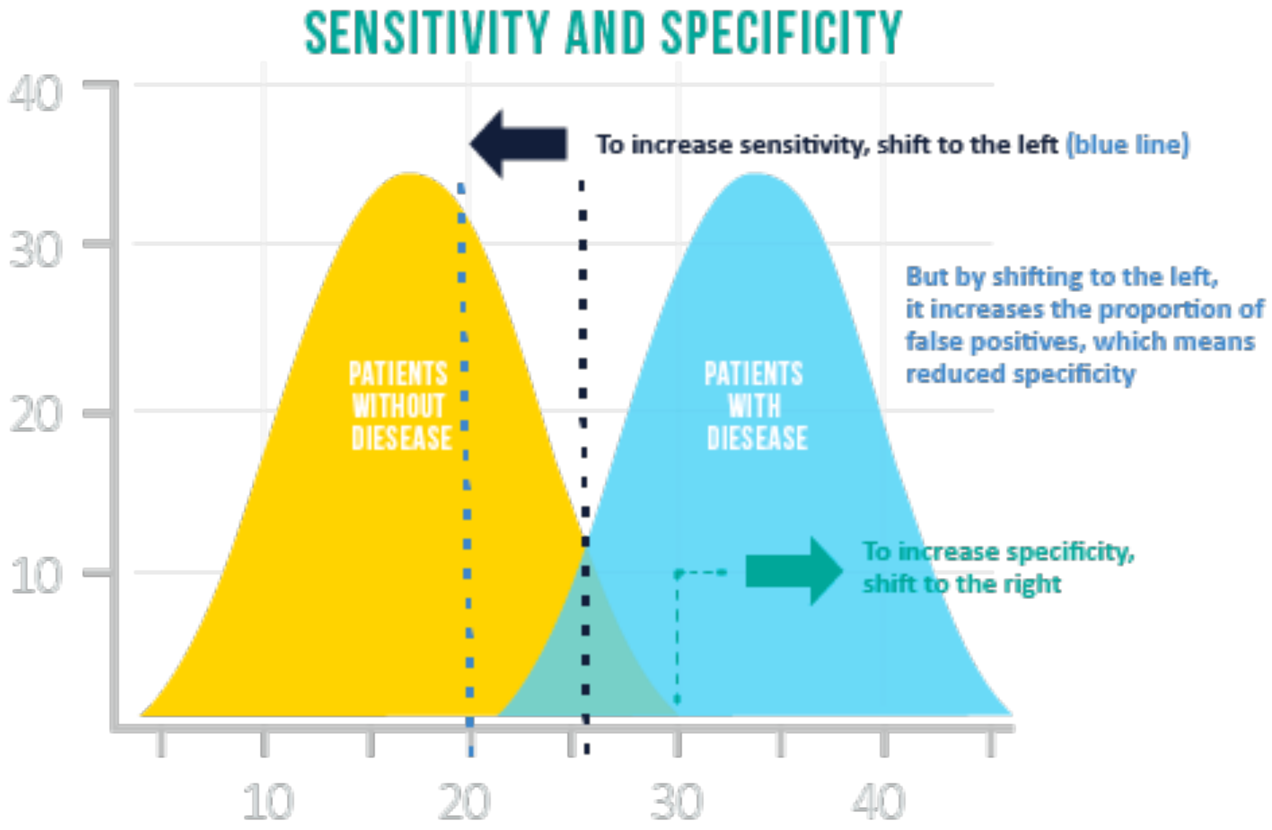
	precision	recall	f1-score
Cat	0.308	0.667	0.421
Fish	0.667	0.200	0.308
Hen	0.667	0.667	0.667

Cat: Precision = 4 / 13 = 0.308, Recall = 4 / 6 = 0.667

Fish: Precision = 2 / 3 = 0.667, Recall = 2 / 10 = 0.200

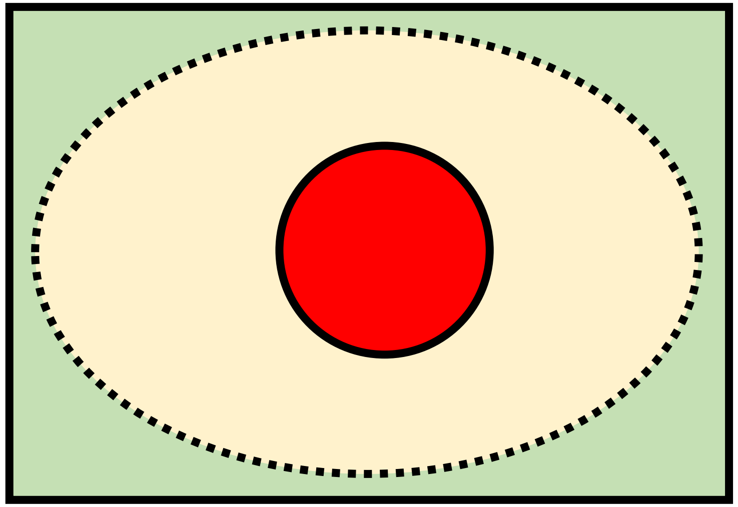
Hen: Precision = 6 / 9 = 0.667, Recall = 6 / 9 = 0.667

Medical Testing



How does this impact patient care and public policy?

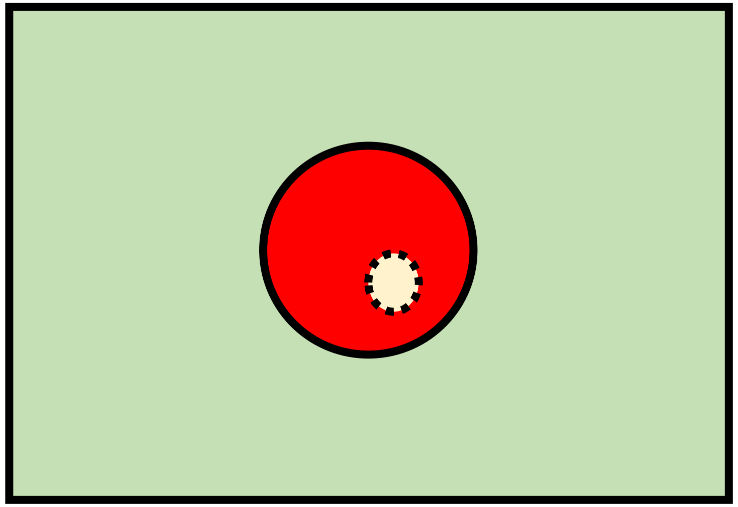
Recall/Precision and Sensitivity/Specificity



Perfect recall / sensitivity (TP/P):
Everyone truly sick is detected.

Poor precision (TP/(TP+FP)):
Few of the positives are really sick.

Poor specificity (TN/N):
Many healthy people are test positive.

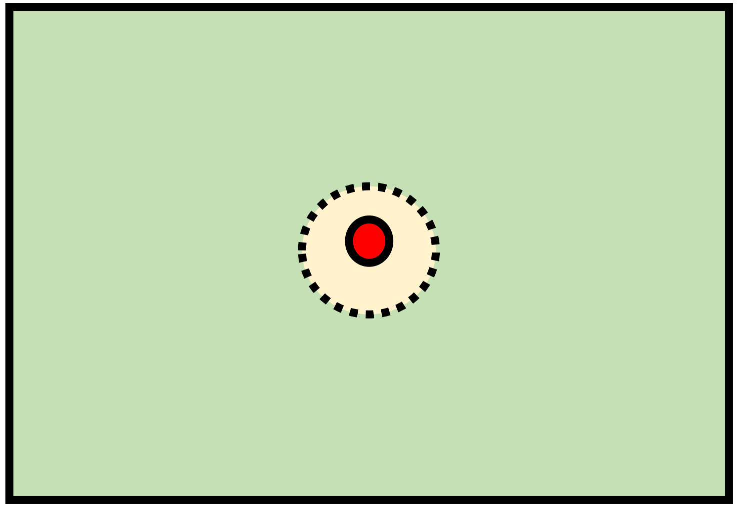


Poor recall / sensitivity:
Many sick people aren’t detected.

Perfect precision:
Every positive is truly sick.

Perfect specificity:
All healthy people test negative.

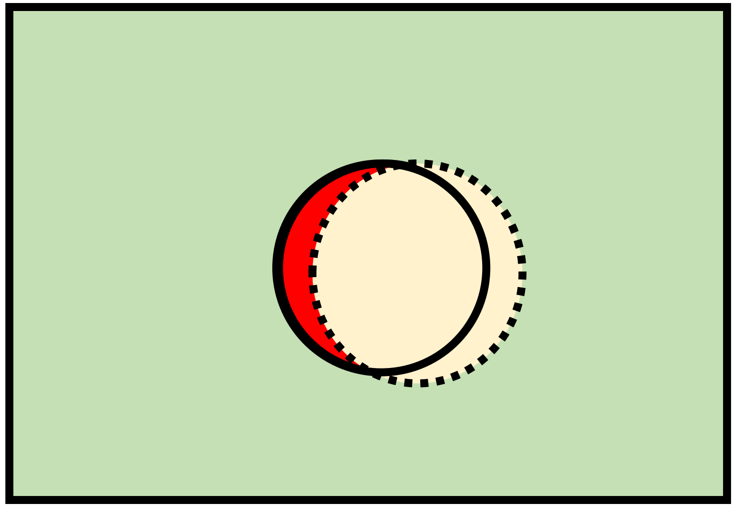
Recall/Precision and Sensitivity/Specificity



Perfect recall / sensitivity (TP/P):
Everyone truly sick is detected.

Poor precision (TP/(TP+FP)):
Few of the positives are really sick.

High specificity (TN/N):
Most healthy people are test negative.



Good recall / sensitivity:
Most sick people are detected positive.

Good precision:
Most positives are *really* sick.

Good specificity:
Most healthy people test negative

Precision or Recall? It depends.

Covid-19 Testing:

⭐ High recall / sensitivity → We detect most patients with Covid (**screening**)
High precision → A high proportion of positives are truly sick
⭐ High specificity → A high proportion of negatives are indeed healthy (**diagnosis**)

Movie Recommendations:

High recall / sensitivity → We recommend most movies a user would like
⭐ High precision → Most recommendations are indeed liked
High specificity → Most movies not recommended would not be liked

Search Engines / Information Retrieval

High recall / sensitivity → Find all relevant information
⭐ High precision → Most hits are relevant
High specificity → Most ignored information is not relevant

F1 - Score

Classifier A: precision > recall
Classifier B: recall > precision

Which is better?

The F1-Score is a way of combining precision and recall into a single overall score:

$$\textbf{F1-score} = 2 \times (\text{precision} \times \text{recall})/(\text{precision} + \text{recall})$$

Combining F1-Scores

		True/Actual		
		Cat (🐱)	Fish (🐟)	Hen (🐔)
Predicted	Cat (🐱)	4	6	3
	Fish (🐟)	1	2	0
	Hen (🐔)	1	2	6

Class	Precision	Recall	F1-score
Cat	30.8%	66.7%	42.1%
Fish	66.7%	20.0%	30.8%
Hen	66.7%	66.7%	66.7%

$$\text{Macro-F1} = (42.1\% + 30.8\% + 66.7\%) / 3 = 46.5\%$$

$$\text{Weighted-F1} = (6 \times 42.1\% + 10 \times 30.8\% + 9 \times 66.7\%) / 25 = 46.4\%$$

Iris Classification Report

	Setosa	Virginica	Versicolor
Setosa	50	0	0
Virginica	0	47	3
Versicolor	0	3	47

```
Classification Report:
              precision    recall  f1-score   support

      setosa      1.000     1.000     1.000        50
  versicolor      0.940     0.940     0.940        50
   virginica      0.940     0.940     0.940        50

    accuracy                          0.960       150
   macro avg      0.960     0.960     0.960       150
weighted avg      0.960     0.960     0.960       150
```

Limitations of F1-Score

Main problem: The F1-Score gives equal weight to Precision and Recall

A more general F score, F_β, that uses a positive real factor β, where β is chosen such that recall is considered β times as important as precision, is:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

General Distance Measure

$$\text{diff} = w_1|\Delta A_1|^r + w_2|\Delta A_2|^r + \ldots + w_n|\Delta A_n|^r$$

where

ΔA_i is the difference with respect to feature i,

w_k is a feature weighting factor (0.0 to 1.0), and

r is an exponent (> 0.0)

Can we evolve an optimal distance metric?

$$\text{diff} = w_1|\Delta A_1|^r + w_2|\Delta A_2|^r + \ldots + w_n|\Delta A_n|^r$$

where

ΔA_i is the difference with respect to feature i,

w_k is a feature weighting factor (0.0 to 1.0), and

r is an exponent (> 0.0)

A **solution** consists of a specification of the distance metric parameters: *$w_1 \ldots w_n$, r, k (# of nearest neighbors).* So ***n+2*** parameters altogether.

We *evaluate* the solution by testing it against a dataset and finding:

- 1 - classification accuracy (error rate)
- 1 - macro-F1 / weighted-F1
- # of weights > 0
- 1 - macro-precision / weighted-precision?
- 1 - macro-recall / weighted-recall?

Tweaking / modifying a solution

Mutate:

weights				r	k
1.0	1.0	1.0	1.0	2	1

1.0	1.0	0.95	1.0	2	1

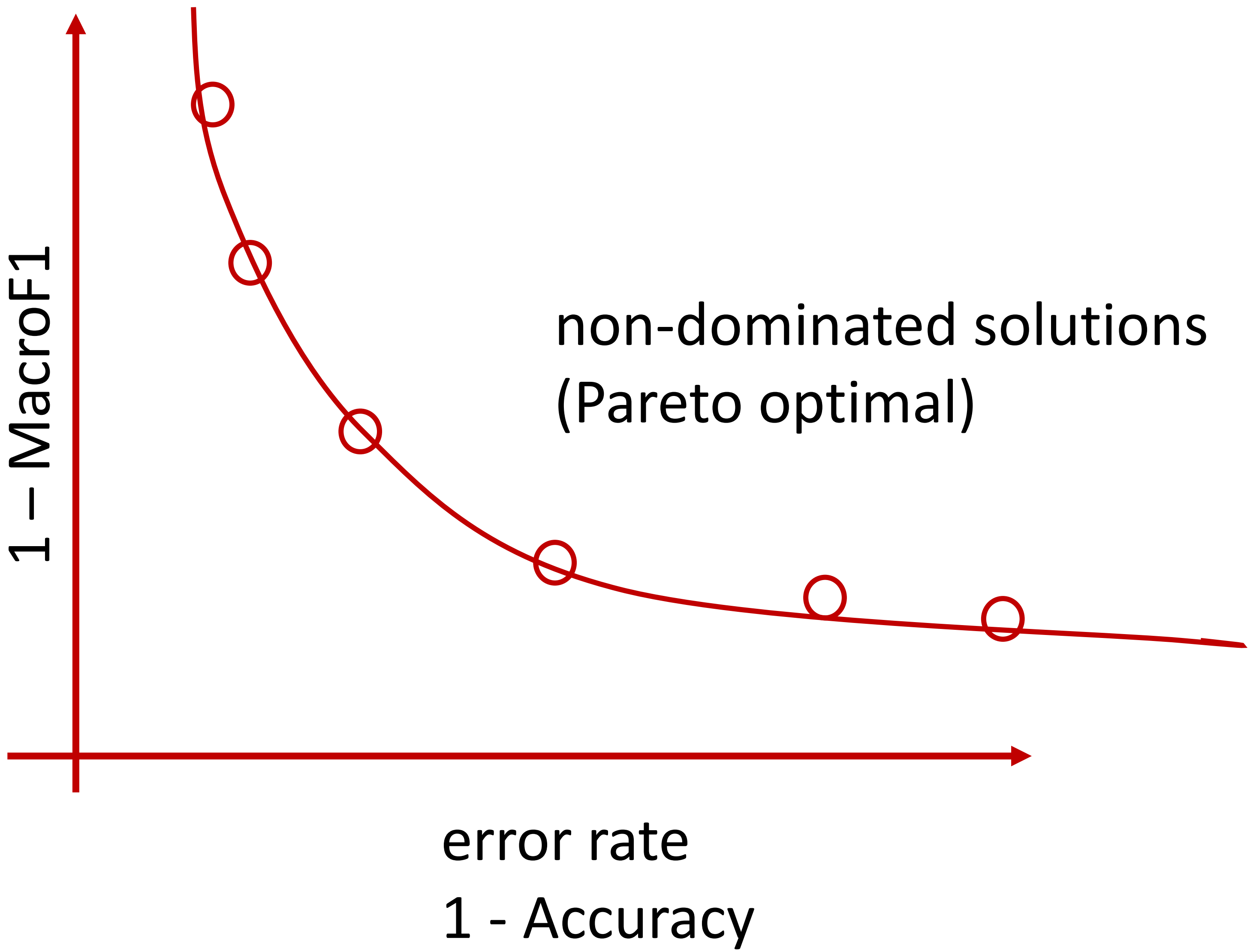
Crossover:

0.20	0.90	1.00	0.45	2	5

0.38	0.29	1.00	0.93	1.5	3

0.20	0.90	1.00	0.93	1.5	3		0.38	0.29	1.00	0.45	2	5

Tradeoffs


1 – MacroF1
non-dominated solutions
(Pareto optimal)
error rate
1 - Accuracy

Tweaking / modifying a solution

weights				r	k
1.0	1.0	1.0	1.0	2	1

Mutate:

1.0	1.0	0.95	1.0	2	1

0.20	0.90	1.00	0.45	2	5

0.38	0.29	1.00	0.93	1.5	3

Crossover:

0.20	0.90	1.00	0.93	1.5	3

0.38	0.29	1.00	0.45	2	5

Predicting recurrence of breast cancer

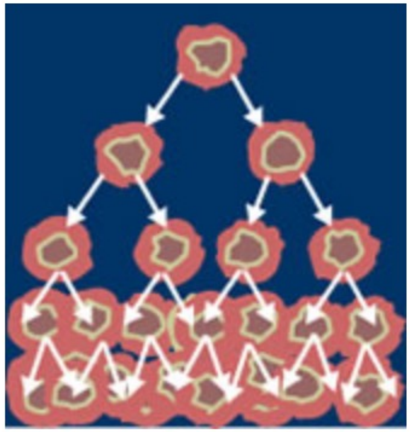
UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Breast Cancer Wisconsin (Prognostic) Data Set

Download: Data Folder, Data Set Description



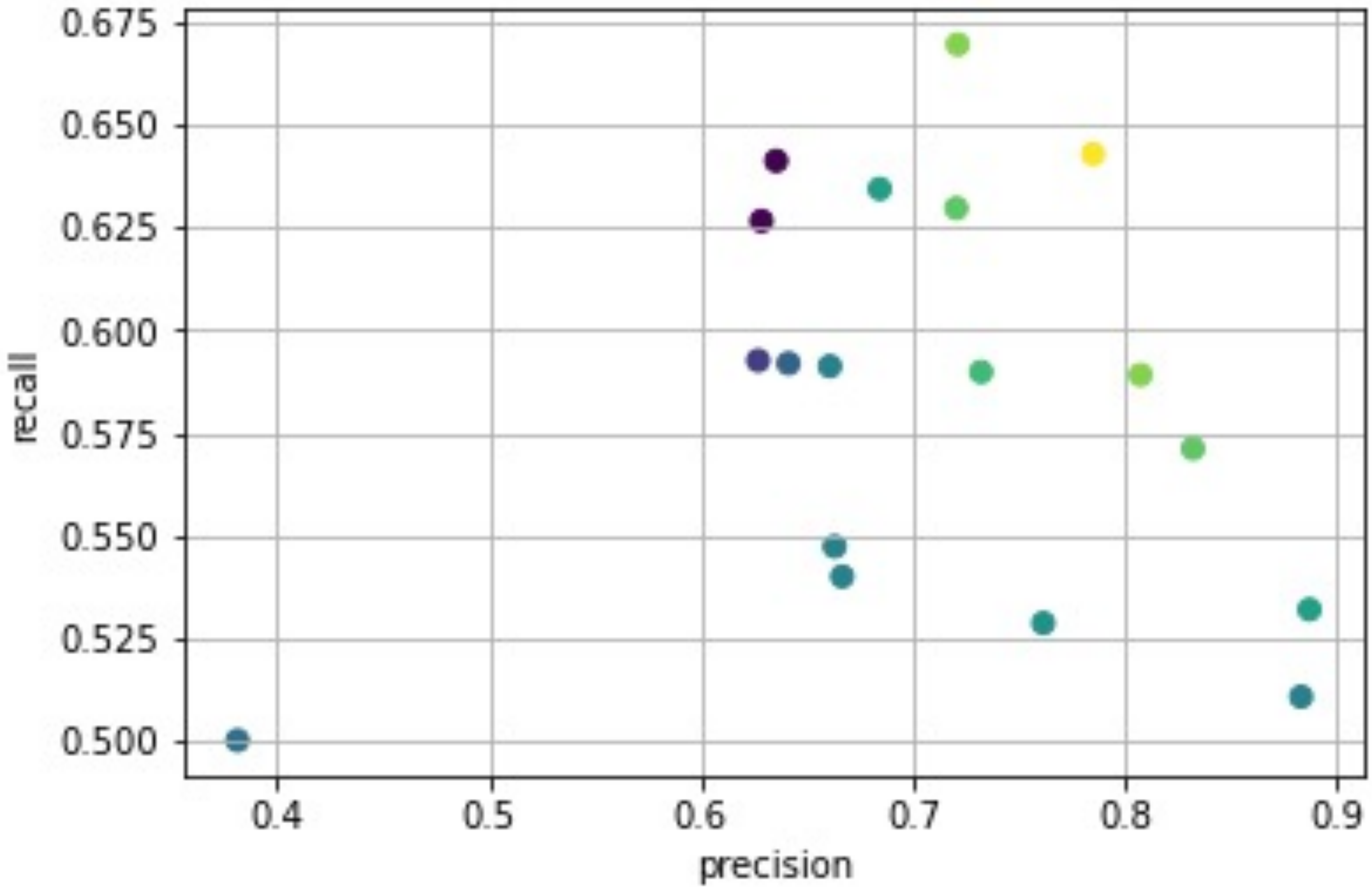
Abstract: Prognostic Wisconsin Breast Cancer Database

Data Set Characteristics:	Multivariate	**Number of Instances:**	198	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	34	**Date Donated**	1995-12-01
Associated Tasks:	Classification, Regression	**Missing Values?**	Yes	**Number of Web Hits:**	224238

		predicted	
		N	R
actual	N	[[151	0]
	R	[44	3]]

	precision	recall	f1-score	support
N	0.77436	1.00000	0.87283	151
R	1.00000	0.06383	0.12000	47
accuracy			0.77778	198
macro avg	0.88718	0.53191	0.49642	198
weighted avg	0.82792	0.77778	0.69413	198

discarded	accuracy	recall	precision	weights	R	K
32	0.76263	0.50000	0.38131	00000000000000000000000000000000	2.0085	3
31	0.74747	0.59264	0.62700	00000000000000000000000000000010	2.1085	4
31	0.73232	0.62667	0.62858	00000000000000000000000000000010	2.1085	1
30	0.73232	0.64133	0.63554	00000000010000000000000000000010	1.0264	2
31	0.75758	0.59194	0.64133	00000000000000000000001000000000	2.0085	5
31	0.76768	0.59124	0.66087	00000000000100000000000000000000	2.6851	5
31	0.76768	0.54727	0.66310	00000000000000010000000000000000	2.0085	8
31	0.76768	0.53995	0.66667	00000000000000010000000000000000	2.0085	10
30	0.77778	0.63449	0.68437	00000000010000000000000000000010	1.0537	3
30	0.79293	0.62977	0.72055	00000001000000000000000000000100	1.1662	5
29	0.79798	0.66972	0.72121	00000000010000000000000001000010	1.7124	3
30	0.78788	0.58983	0.73224	00000000010000000000000000000010	1.4039	7
31	0.77273	0.52860	0.76160	00000000000100000000000000000000	2.0223	22
29	0.81313	0.64302	0.78511	00000101000000000000000000000100	1.0179	9
30	0.79798	0.58912	0.80749	00000001010000000000000000000000	2.9913	7
30	0.79293	0.57116	0.83224	00000000010000000000000000000010	2.7197	8
31	0.76768	0.51064	0.88325	00000000000000000010000000000000	2.2906	17
30	0.77778	0.53191	0.88718	00100000000000000000010000000000	2.1623	19


0.675
0.650
0.625
0.600
0.575
0.550
0.525
0.500
recall
0.4
0.5
0.6
0.7
0.8
0.9
precision

```
          predicted
            N     R
actual  N [[138  13]
        R  [ 27  20]]
```

R=1.7, K=3, Features = 3

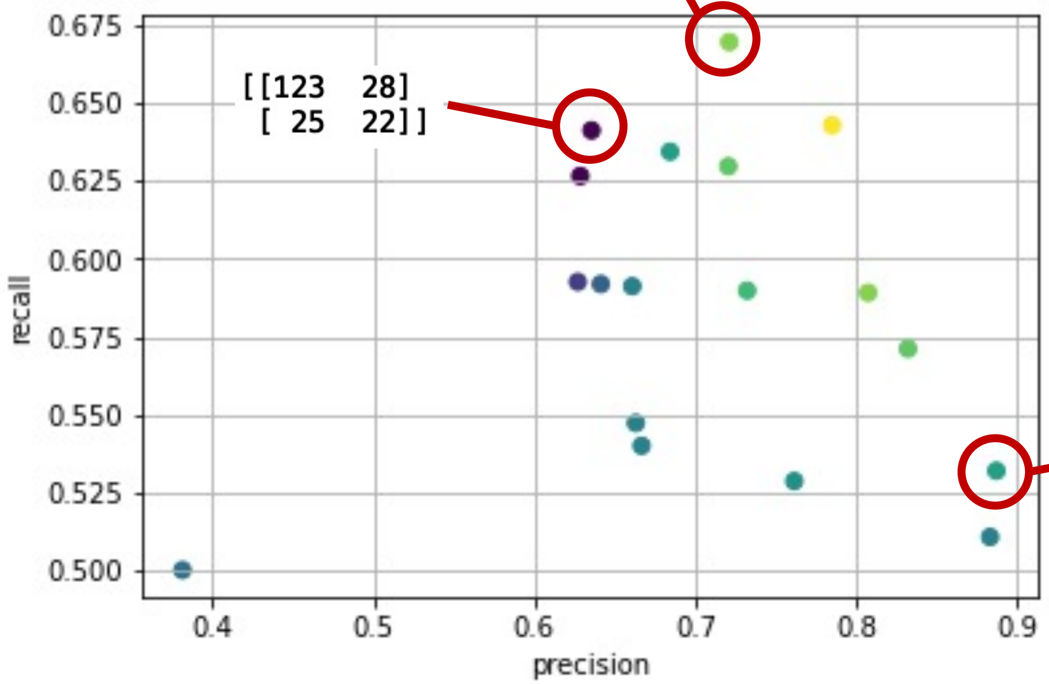
```
              precision    recall  f1-score   support

           N    0.83636   0.91391   0.87342       151
           R    0.60606   0.42553   0.50000        47

    accuracy                        0.79798       198
   macro avg    0.72121   0.66972   0.68671       198
weighted avg    0.78170   0.79798   0.78478       198
```

Recall: What fraction of the recurring cancers are we detecting (predicted to recur)? (20 / 47 = 0.42553)

Precision: What fraction of those cancers predicted to recur actually recurred? (20 / 33 = 0.60606)




```
          predicted
            N     R
actual  N [[151   0]
        R  [ 44   3]]
```

```
              precision    recall  f1-score   support

           N    0.77436   1.00000   0.87283       151
           R    1.00000   0.06383   0.12000        47

    accuracy                        0.77778       198
   macro avg    0.88718   0.53191   0.49642       198
weighted avg    0.82792   0.77778   0.69413       198
```

R=2.3, K=17, Features = 1