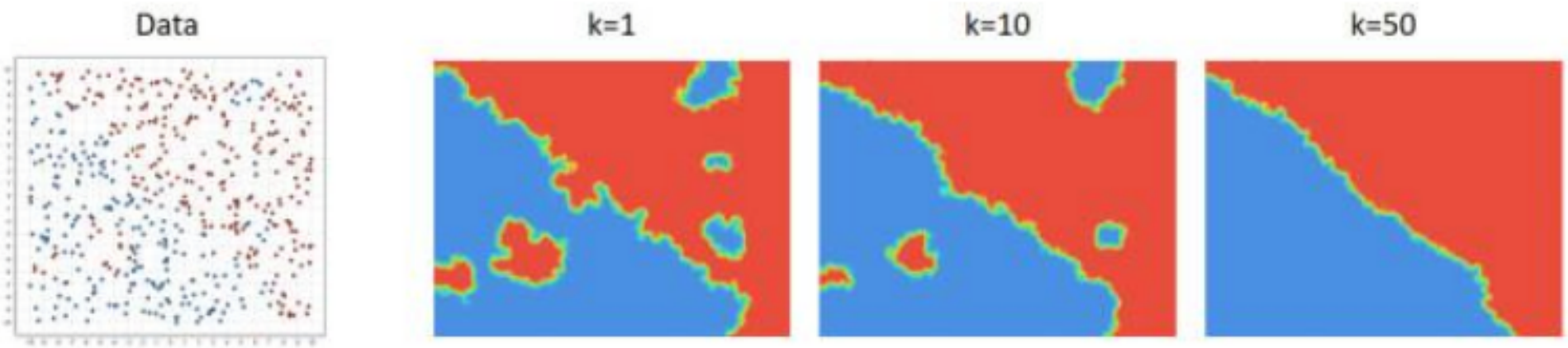
Lazy Learners (or Learning from Your Neighbors)

Ronnie Alves (alvesrco@gmail.com)
https://sites.google.com/site/alvesrco/



https://www.kdnuggets.com/2017/09/rapidminer-k-nearest-neighbors-laziest-machine-learning-technique.html

Machine Leaning: The Basis

- Understand the domain and goals
- Data integration, selection and cleaning
- Learning models
- Interpreting results, aligned with the goals
- Deploying models

LEARNING = REPRESENTATION + EVALUATION + OPTIMIZATION

Deploy a score!

Learning classification models

Sequence analysis

Tally, a scoring tool to set boundary between repetitive and non-repetitive protein sequences

François D. Richard[1,2], Ronnie Alves[2], and Andrey V. Kajava[1,2,*]

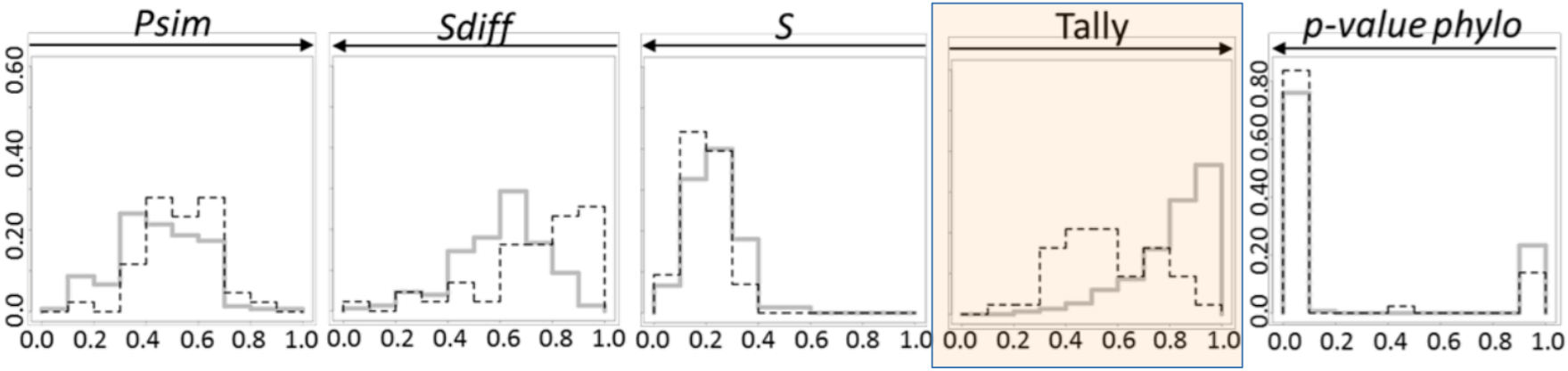
[1] Centre de Recherches de Biochimie Macromoléculaire, UMR 5237 CNRS, Université Montpellier 1919 Route de Mende, 34293 Montpellier, Cedex 5, France

[2] Institut de Biologie Computationnelle (IBC), 860 rue St Priest, Bâtiment 5 - CC05019, 34095, Montpellier, France

Received on XXXXX; revised on XXXXX; accepted on XXXXX

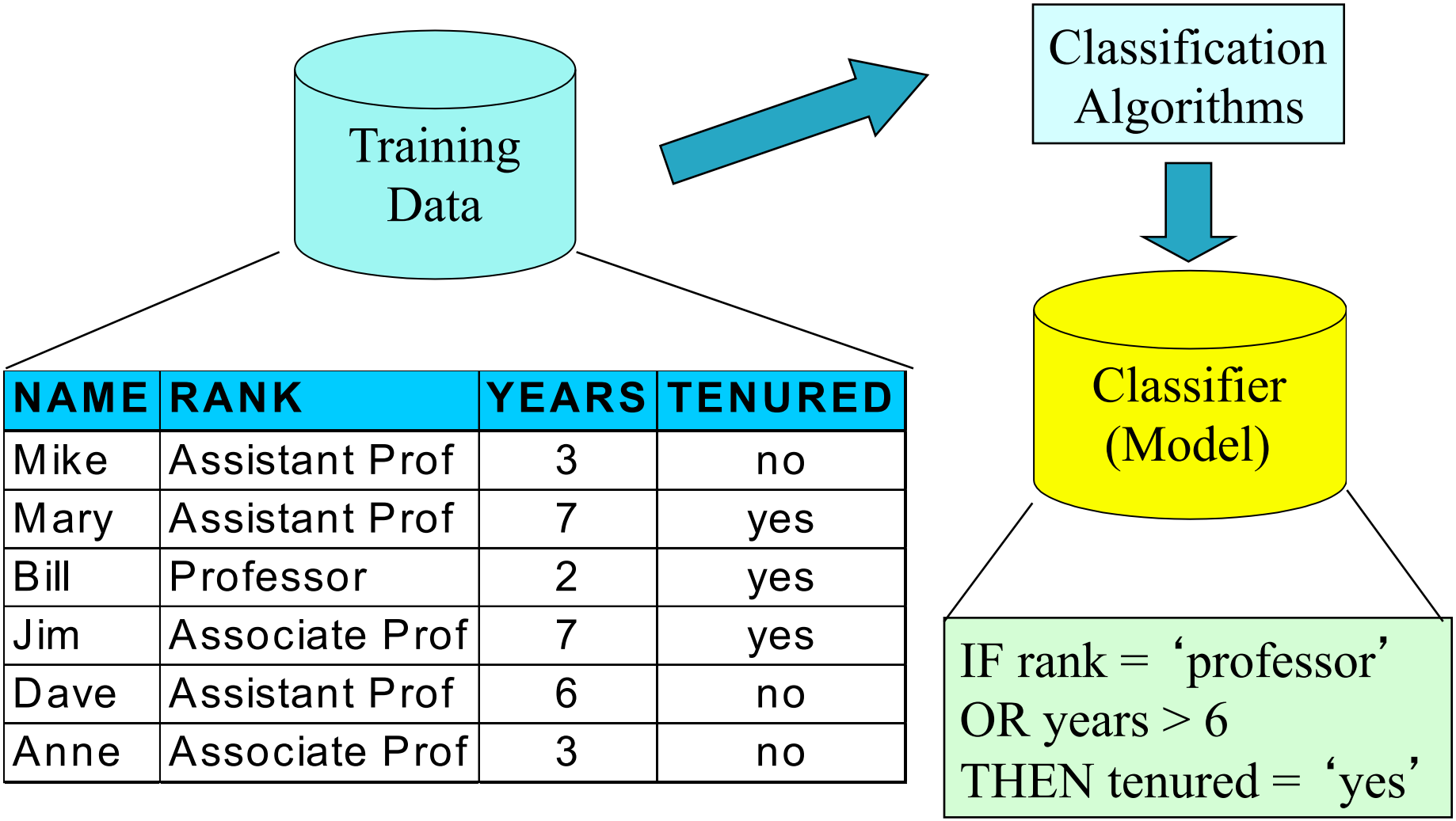
Associate Editor: XXXXXXX

Domain > Data !



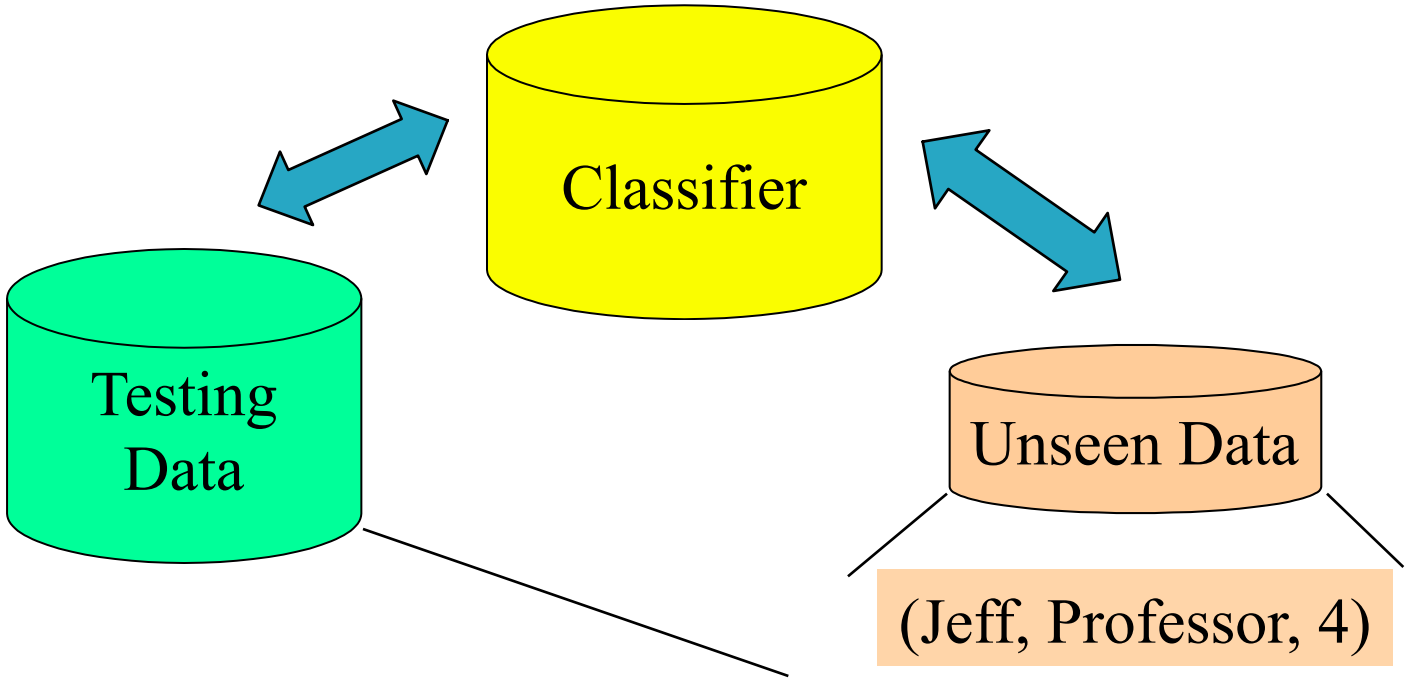
Goal: to distinguish between the sequences that contain highly imperfect TRs and the aperiodic sequences without TRs

Process (1): Model Construction



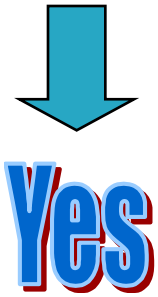
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Tenured?



Process (3): Model Evaluation

Holdout method

- Given data is randomly partitioned into two independent sets
  - **Training** set (e.g., **2/3**) for model construction
  - **Test** set (e.g., **1/3**) for accuracy estimation
- Random sampling: a variation of holdout
  - Repeat holdout k times, accuracy = avg. of the accuracies obtained

Cross-validation (*k*-fold, where k = 10 is most popular)

- Randomly partition the data into ***k mutually exclusive* subsets**, each approximately equal size
- At *i*-th iteration, use D_i as test set and others as training set
- Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
- ***Stratified cross-validation***: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

Process (3): Model Evaluation

Confusion Matrix:

Actual class\Predicted class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

Given *m* classes, an entry, $\boldsymbol{CM_{i,j}}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*

May have extra rows/columns to provide totals

Process (3): Model Evaluation

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

Classifier Accuracy, or recognition rate: percentage of test set tuples that are correctly classified

Accuracy = (TP + TN)/All

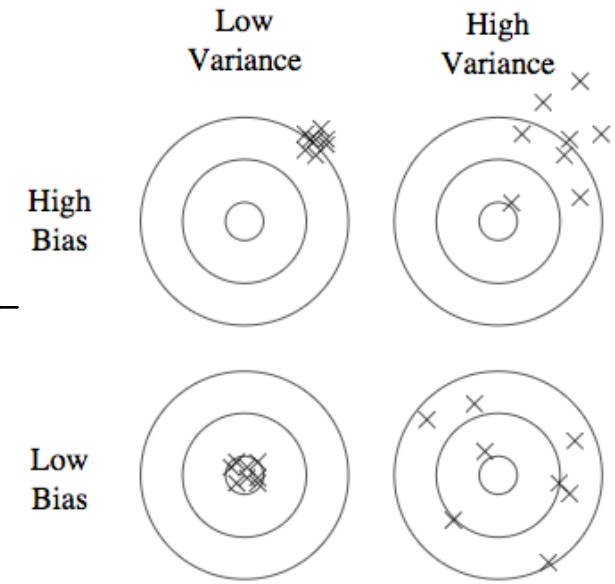
Error rate: *1 – accuracy*, or

Error rate = (FP + FN)/All

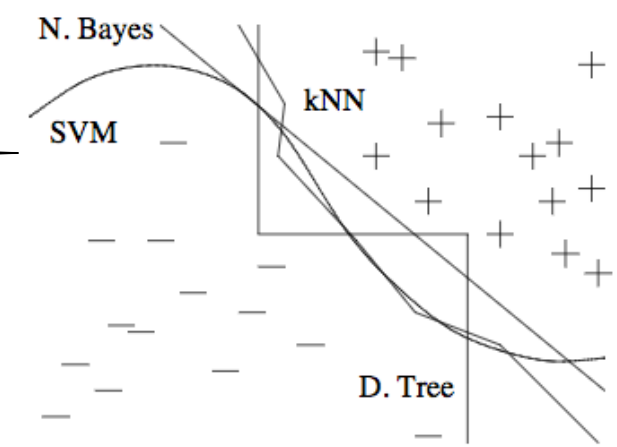
- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

Bias and Variance

Bias is a learner's tendency to consistently learn the same wrong thing.
Variance is the tendency to learn random things irrespective of the real signal.

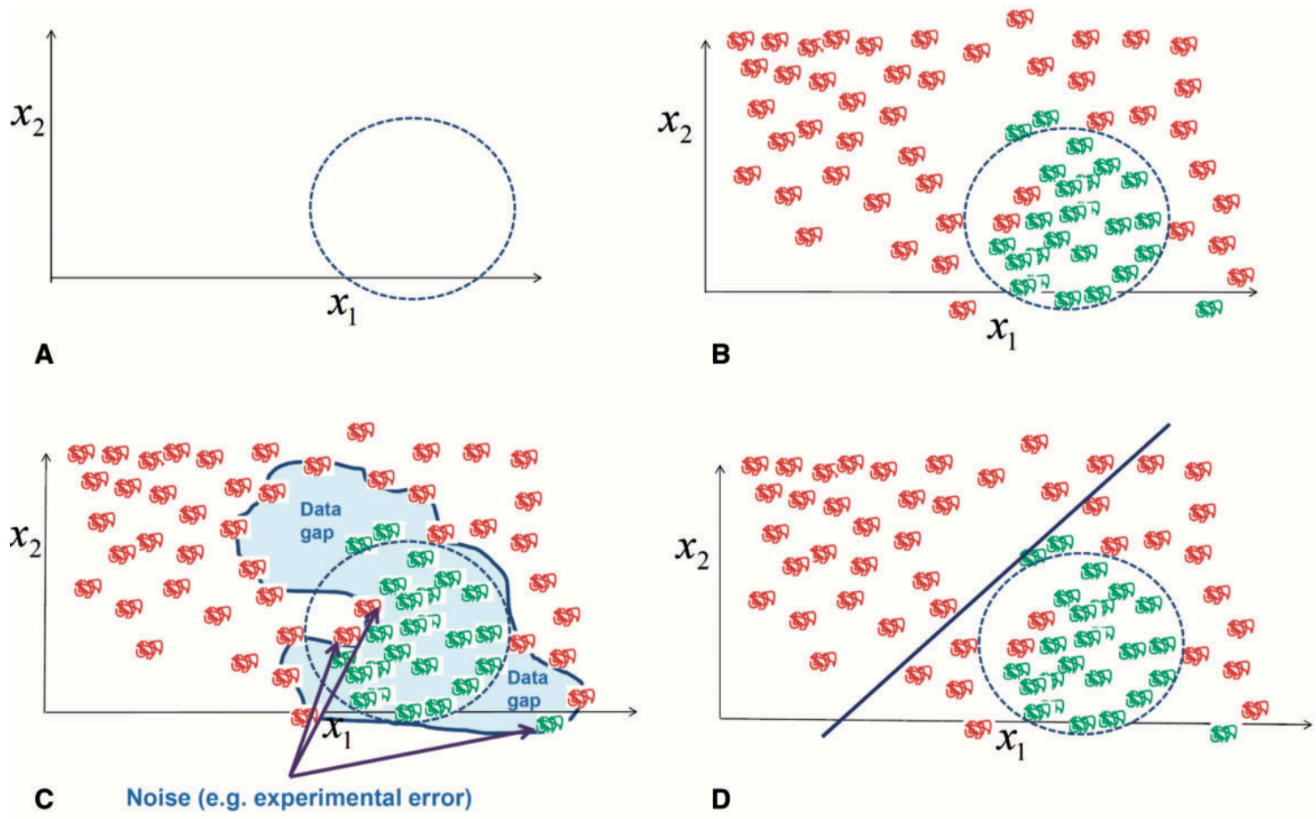


Very different decision boundaries can yield similar class predictions.



A more powerful learner is not necessarily better than a less powerful one

Over-fitting vs Under-fitting



ELLABORATE A PROPER VALIDATION SCHEME TO BEST UNDERSTAND BIAS AND VARIANCE.

Lazy Learner: Instance-Based Methods

Instance-based learning:

Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified

Typical approaches

***k*-nearest neighbor approach**

- Instances represented as points in a Euclidean space.

Locally weighted regression

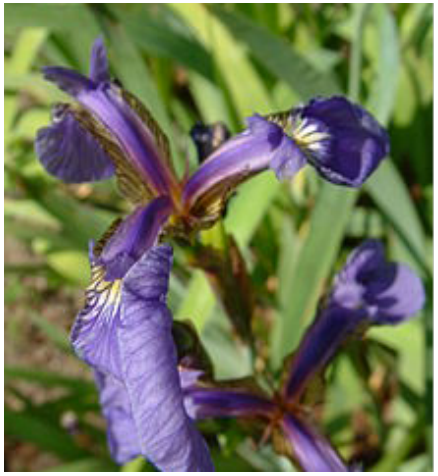
- Constructs local approximation

Case-based reasoning

- Uses symbolic representations and knowledge-based inference

Lazy Learner: Iris Data set

The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis.[1]



setosa

virginica

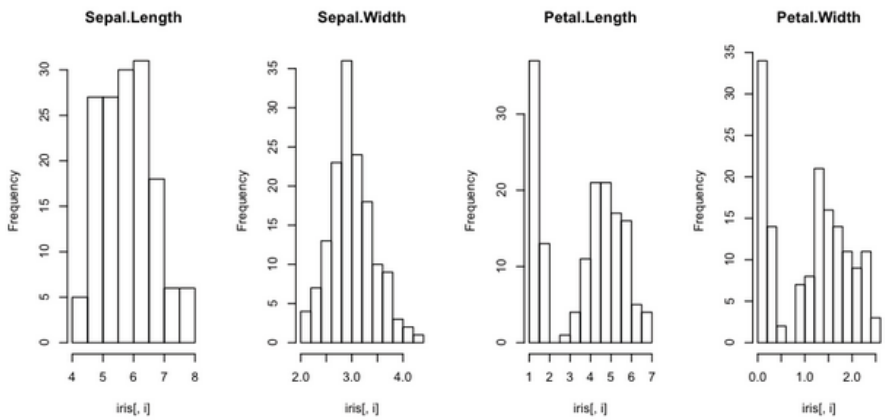
versicolor

The data set consists of **50 samples from each of three species** of Iris (Iris setosa, Iris virginica and Iris versicolor).

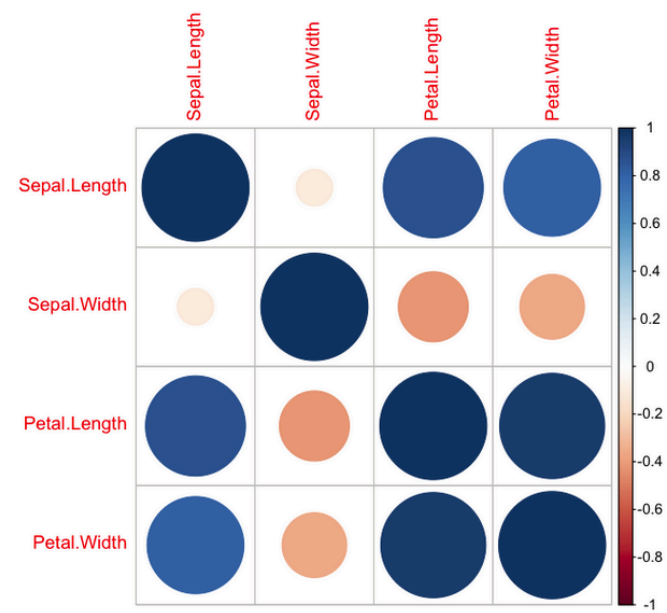
Four features were measured from each sample:
the length and the width of the sepals and petals, in centimetres.

Lazy Learner: Iris Data set

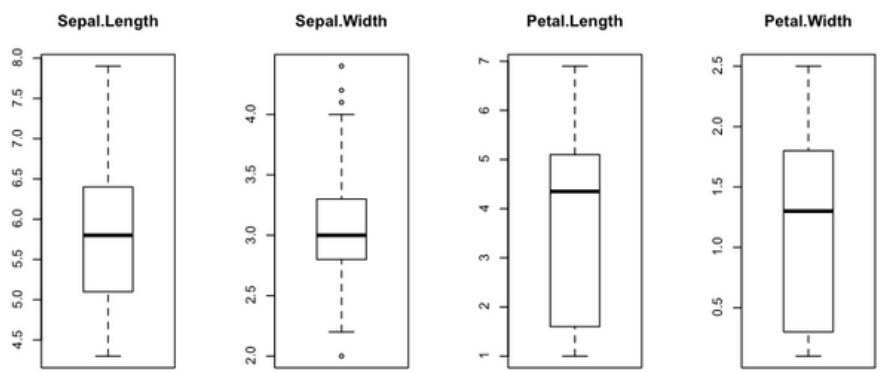
Histograms



Correlation plot



Boxplots

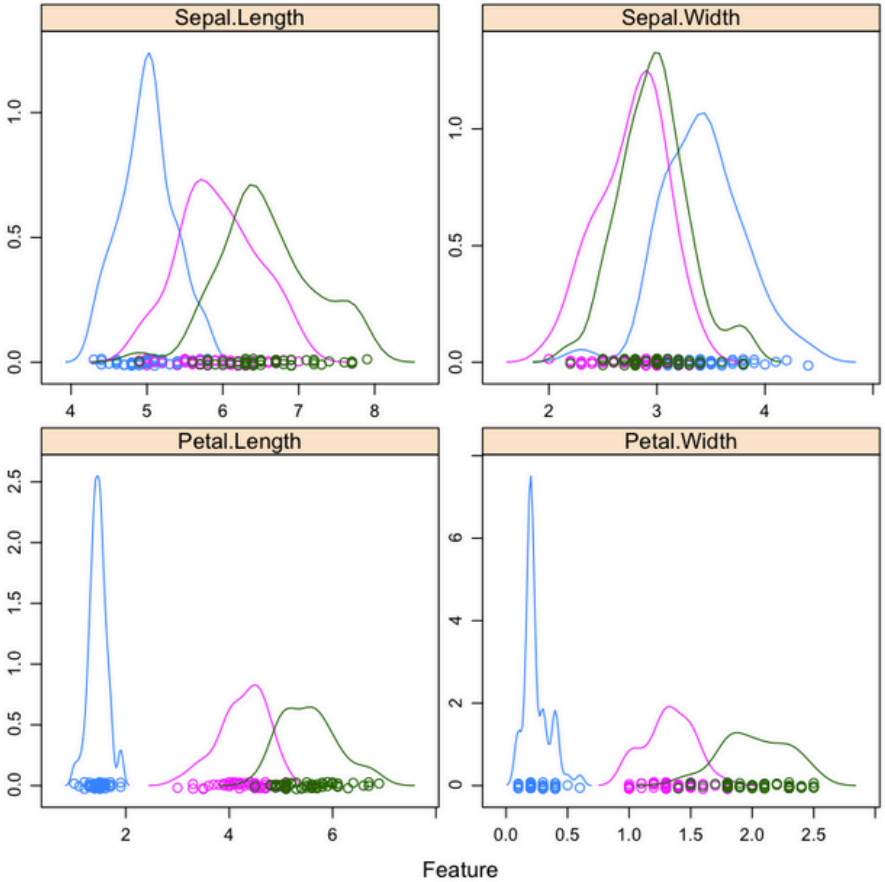


The data set consists of **50 samples from each of three species** of Iris (Iris setosa, Iris virginica and Iris versicolor).

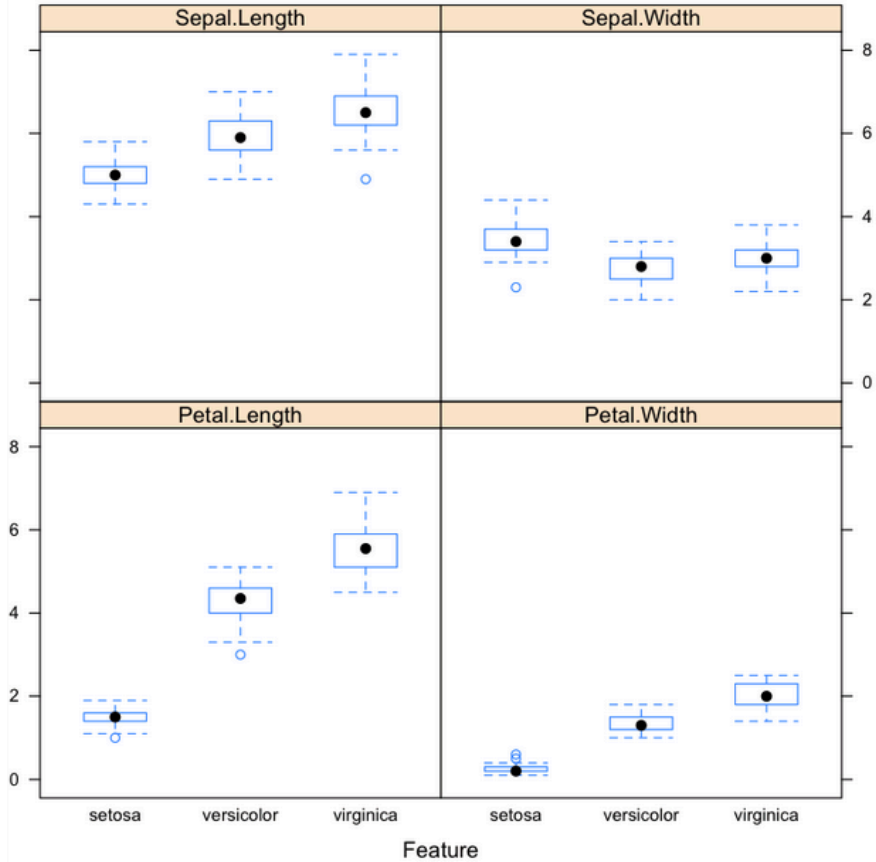
Four features were measured from each sample:
the length and the width of the sepals and petals, in centimetres.

Lazy Learner: Iris Data set

Density distribution by classes



Box plots by classes



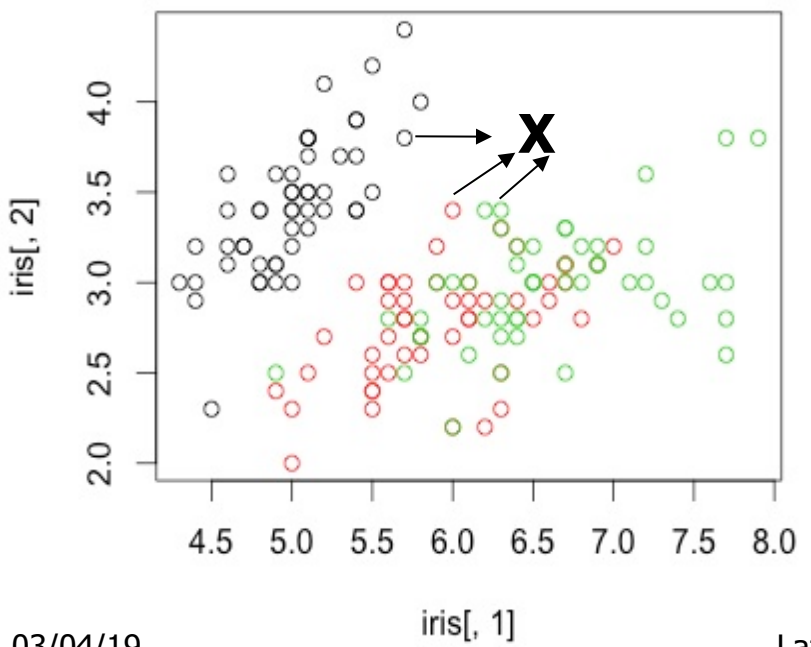
The data set consists of **50 samples from each of three species** of Iris (Iris setosa, Iris virginica and Iris versicolor).

Four features were measured from each sample:
the length and the width of the sepals and petals, in centimetres.

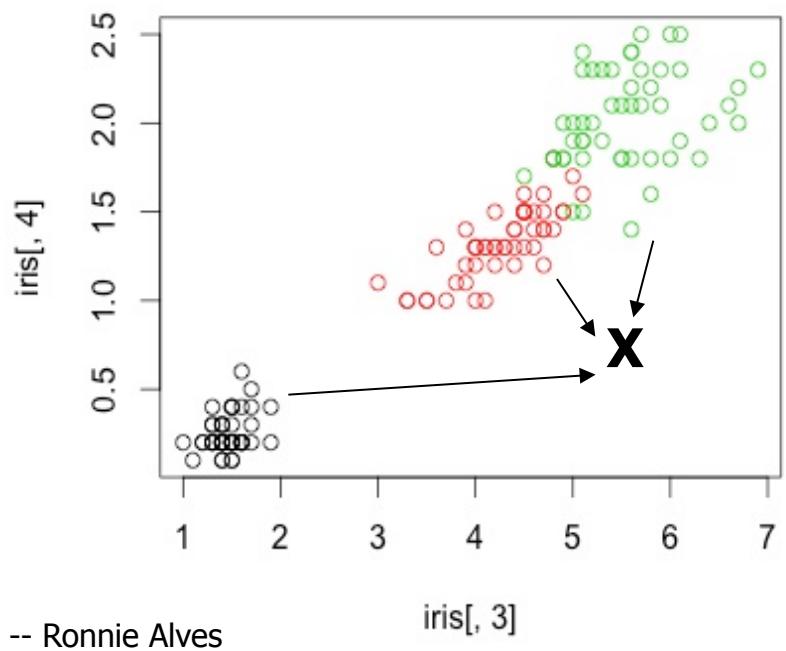
Lazy Learner: Instance-Based Methods

What kind of flower (X) is this?
i) setosa, (ii) versicolor, iii) virginica

Sepals



Petals



Lazy Learner: Nearest-neighbor classifiers

Fix and Hodges introduced a **non-parametric method** for pattern classification that has since become known the k-nearest neighbor rule (1951).

Nearest Neighbors have been used in statistical estimation and **pattern recognition** already in the beginning of 1970's (non-parametric techniques).

Dynamic Memory: A theory of Reminding and Learning in Computer and People (Schank, 1982).

Nearest-neighbor classifiers are based on **learning by analogy**, that is, by comparing a given test tuple with training tuples that are similar to it.

Lazy Learner: Nearest-neighbor classifiers

The training tuples are described by n attributes.

Each tuple represents a point in a n-dimensional pattern space. Thus, all of training tuples are stored in a n-dimensional pattern space.

When given an unknown tuple, a **k-nearest-neighbor classifier** searches the pattern space for the training **tuples that are <u>closest</u> to the unknown tuple**.

These **k training tuples** are the **k "nearest neighbors"** of the unknown tuple.

Lazy Learner: Closeness

Closeness is defined in terms of a distance metric, such as **Euclidean distance.**

Instance x (often called a feature vector)

<a1(x), a2(x), …, an(x)>

Distance between two instances xi and xj

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

Lazy Learner: Discrete Valued Target Function

Training algorithm:

For each training example <x, class(x)>, add the example to the list *Training*

Classification algorithm:

Let V ={v1, …, vl} be a set of classes

Given a query instance xq to be classified

Let X={x1, …, xk} denote the k instances from Training that are nearest to xq

$$\forall i:1\ldots l,\ \ vote_i=\{x\in X\,|\,class(x)=v_i\}$$

Return vi such that **|votei| is largest**

Lazy Learner: Continuous valued target function

Algorithm computes the **mean value of the k nearest** training tuples rather than the most common value

Replace fine line in previous algorithm with

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

class(xq)

Discussion on the *k*-NN Algorithm

k-NN for <u>real-valued prediction</u> for a given unknown tuple

Returns the mean values of the *k* nearest neighbors

<u>Distance-weighted</u> nearest neighbor algorithm

Weight the contribution of each of the *k* neighbors according to their distance to the query x_q

- Give **greater weight to closer neighbors**

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

Discussion on the *k*-NN Algorithm

Robust to **noisy** data by averaging *k*-nearest neighbors

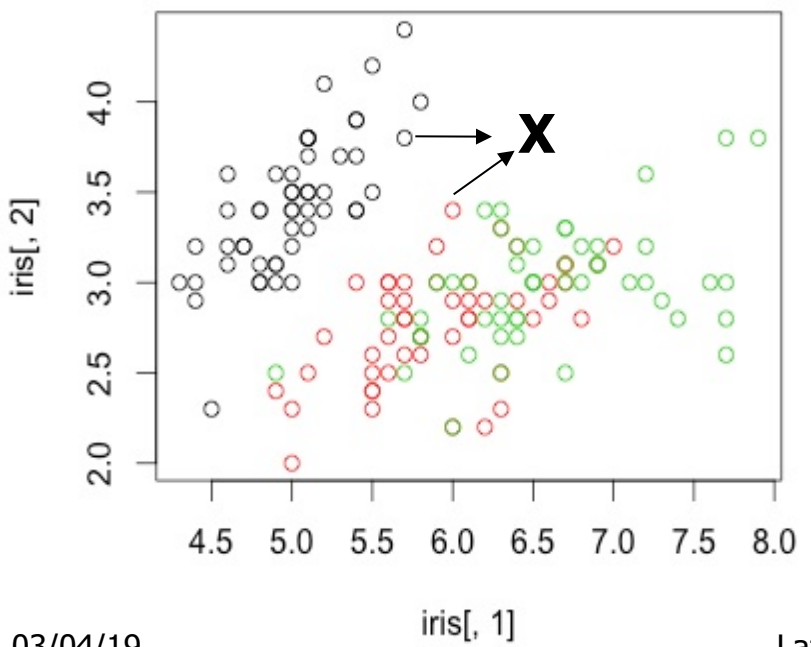
Curse of dimensionality: distance between neighbors could be dominated by **irrelevant attributes**

To overcome it, elimination of the least relevant attributes, **dimensionality reduction**

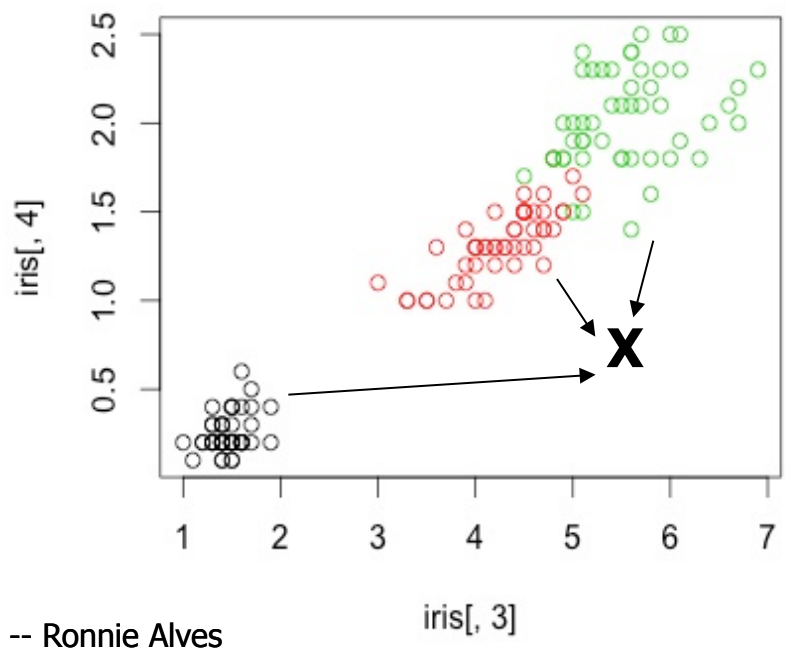
Lazy Learner: hands on KNN <R>

What kind of flower is this?
i) setosa, (ii) versicolor, iii) virginica

Sepals



Petals



Lazy Learner: 1-NN

library(class)

summary(iris)

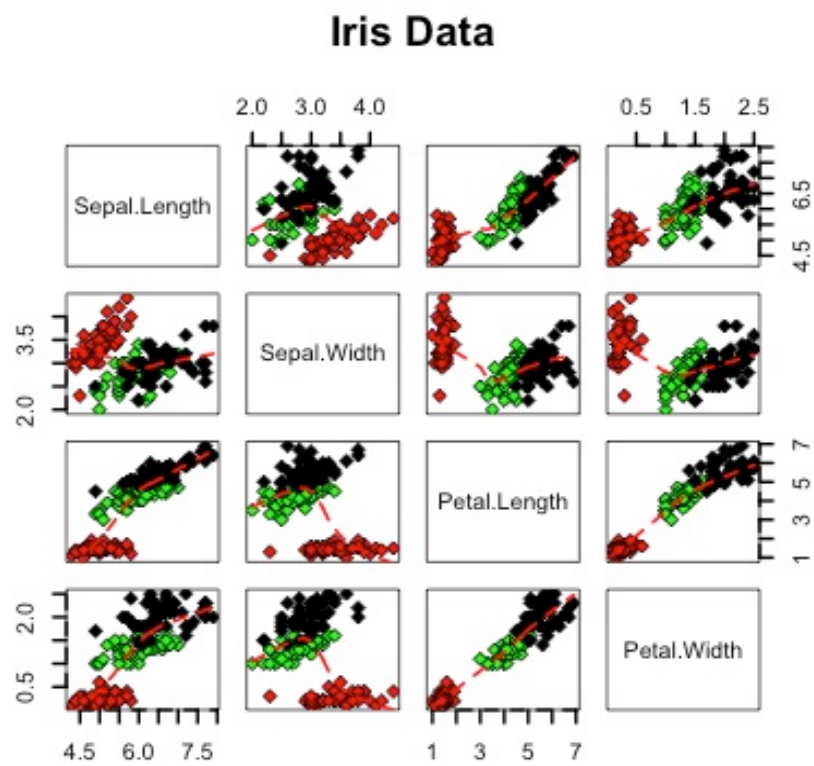
```
 Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species  
Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50  
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50  
Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50  
Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199                  
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800                  
Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500                  
```

head(iris)

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

Lazy Learner: 1-NN

```
pairs(iris[1:4], main="Iris Data", panel=panel.smooth, pch=23,
bg=c("red","green","black")[unclass(iris$Species)],lwd=2,lty=2)
```



What pairs produce the best decision boundaries?

Lazy Learner: 1-NN

```
testidx <- which(1:length(iris[,1])%%5 == 0)

iristrain <- iris[-testidx,]
iristest <- iris[testidx,]

train_input <- as.matrix(iristrain[,-5])
train_output <- as.vector(iristrain[,5])
test_input <- as.matrix(iristest[,-5])
test_output <- as.vector(iristest[,5])
```

120 training tuples

30 test tuples

feature vectors

class vector

Lazy Learner: 1-NN

```
prediction <- knn(train_input, test_input,
                  train_output, k=1)
```

KNN function

```
table(prediction, test_output)
```

Confusion matrix

```
          test_output
prediction   setosa versicolor virginica
  setosa       10         0        0
  versicolor    0        10        1
  virginica     0         0        9
```

Misclassification error Rate = 0.03

```
misclass <- (nrow(test_input)-sum(diag(table(prediction, test_output))))/
nrow(test_input)
```

Lazy Learner: How can I determine a good number of neighbors?

Algorithm:

Starting with k = 1, we use a test set to estimate the misclassification error rate.

Repeat this processes n times, incrementing k to allow one more neighbor.

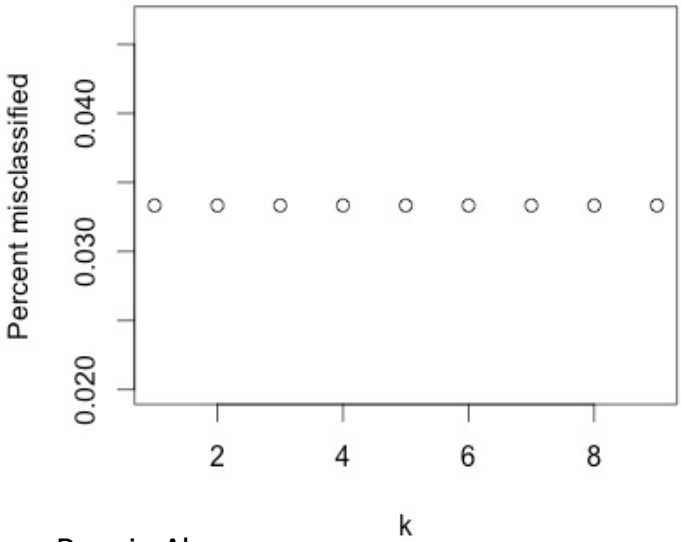
Return k with lowest error rate.

The large the number of of training tuples is, the larger the value of k will be.

If the number of points is fairly large, then the error rate of Nearest Neighbor is less that twice the Bayes error rate

Lazy Learner: How can I determine a good number of neighbors?

```
k=c(1:9)
p=rep(0,9)
summary=cbind(k,p)
colnames(summary)=c("k","Percent misclassified")
for(i in 1:9){
  result=knn(train_input, test_input, train_output, k=i)
  summary[i,2]=(nrow(test_input)-sum(diag(table(prediction, test_output))))/
nrow(test_input)
}
plot(summary)
```



Lazy Learner: Distance-based comparison

Nearest-neighbor classifiers intrinsically assign equal weight to each attribute. Thus, they can suffer from poor accuracy when given noisy or irrelevant attributes.

The choice of a distance metric can be critical.

TASK 1:

List other distance measurements and potential scenario of application (practical examples).

Lazy Learner: Complexity

If D is a training database of |D| tuples and k = 1,

then **O(|D|)** comparisons are required in order to classify a given test tuple.

By presorting and arranging the stored tuples into search trees, the number of comparisons can be reduced to

O(log(|D|)

TASK-2:

Parallel implementation can reduce the running time to

O(1). List other techniques which could speed up knn running time.

Lazy Learner: Performance assesment

Explore kNN with the Wine data set:
http://archive.ics.uci.edu/ml/datasets/Wine

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

TASK-3:

Explore a optimum k value for the wine data set through a five-fold cross validation training.

Lazy Learner: Summary

Top 10 algorithms in data mining

Xindong Wu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang · Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng · Bing Liu · Philip S. Yu · Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg

Received: 9 July 2007 / Revised: 28 September 2007 / Accepted: 8 October 2007
Published online: 4 December 2007

kNN

Abstract This paper presents the top 10 data mining algorithms identified by the IEEE International Conference on Data Mining (ICDM) in December 2006: C4.5, *k*-Means, SVM, Apriori, EM, PageRank, AdaBoost, *k*NN, Naive Bayes, and CART. These top 10 algorithms are among the most influential data mining algorithms in the research community. With each algorithm, we provide a description of the algorithm, discuss the impact of the algorithm, and review current and further research on the algorithm. These 10 algorithms cover classification,

Lazy Learner: Summary

KNN is conceptually simple, yet able to solve complex problems

Can work with relatively little information

Learning is simple (no learning at all!)

Memory and CPU cost

Feature selection problem, Sensitive to representation

Kaggle inClass Prediction Competition 2019

Start -> 03/04/2019
End -> 19/06/2019

https://www.kaggle.com/t/7df13aa49435466898c6867484a644e0

1st Competition (8 teams, 163 submissions)

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲1	**PogMasters**			0.76571	44	4y
2	▼1	**Equipe Graduação**			0.76517	7	4y
3	—	**Parallax** → baseline			0.74248	82	4y

2nd Competition (12 teams, 401 submissions)

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	—	**PPGCC2016** → best			0.84232	54	1y
2	▲2	**Diemisom Melo, Bruno Conte ...**			0.78339	71	1y
3	—	**RMC**			0.78109	66	1y

3rd Competition (9 teams, 569 submissions)

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	—	**ML_RULES**			0.80399	63	9mo
2	▲5	**Like_Wine**			0.76462	88	9mo
3	—	**BioScripters**			0.76407	102	9mo