Machine Learning I

Data Mining and Data integration in Biomedicine
Master in Bioinformatics

Janet Piñero
Medbioinformatics Solutions SL
2025-2026

Outline

What is machine learning

Classification

- K nearest neighbor
- Decision Trees

Hands-on session

References

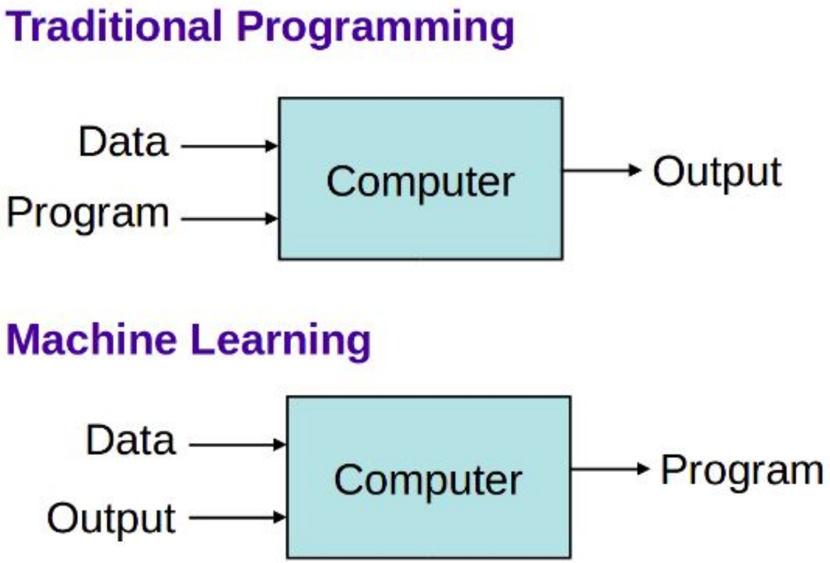
What is machine learning

"Field of study that gives computers the ability to learn without being explicitly programmed"
Arthur Samuel (1959)

a subset of Artificial Intelligence (AI) in the field of computer science that often uses statistical techniques to give computers the ability to “learn” (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed.

A well posed learning problem: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." Tom Michel (1999)

What is machine learning



Traditional Programming

Data

Program

Computer

Output

Machine Learning

Data

Output

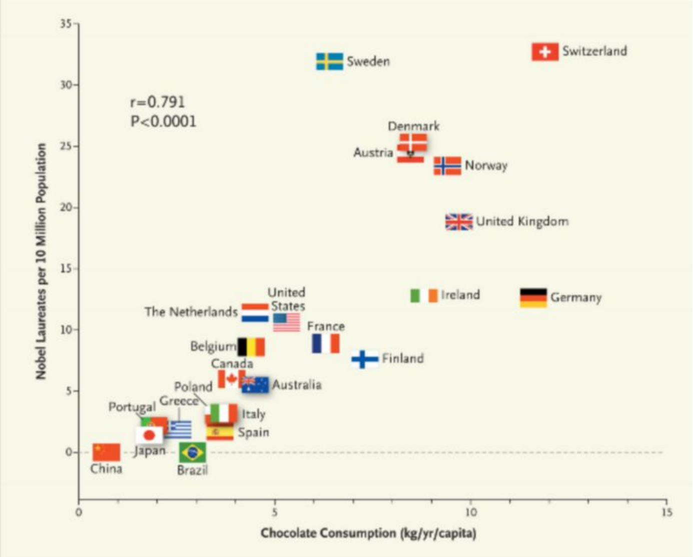
Computer

Program

Components of a predictor

question -> input data -> features -> algorithm -> parameters -> evaluation

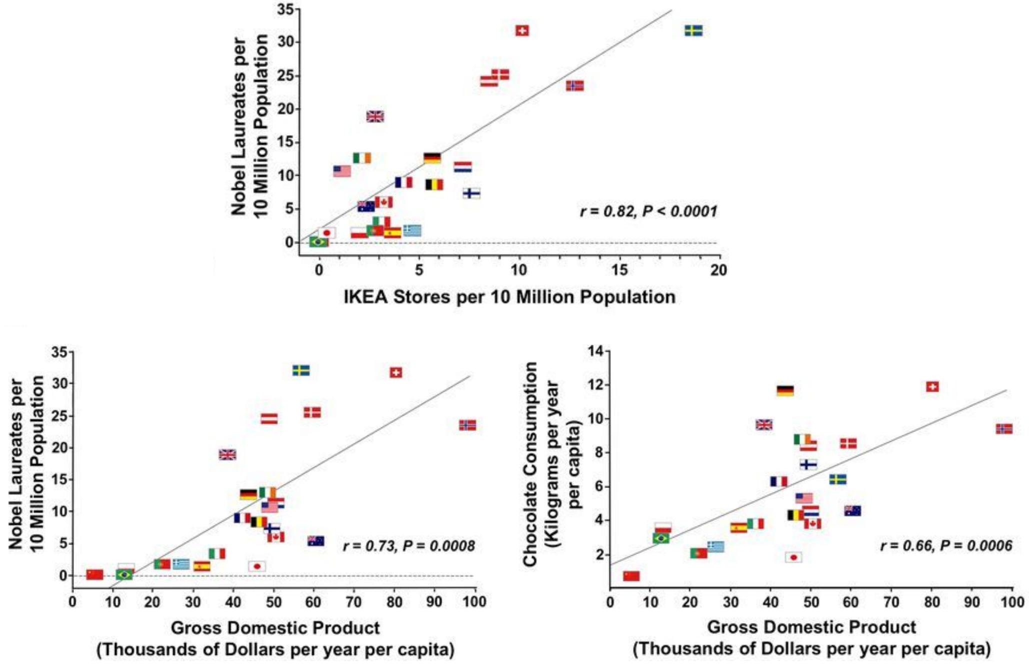
Unrelated data is the most common mistake



- There was a close, significant linear correlation (r=0.791, P<0.0001) between chocolate consumption per capita and the number of Nobel laureates per 10 million persons in a total of 23 countries
- When recalculated with the exclusion of Sweden, the correlation coefficient increased to 0.862.
- Switzerland was the top performer in terms of both the number of Nobel laureates and chocolate consumption.
- The slope of the regression line allows us to estimate that it would take about 0.4 kg of chocolate per capita per year to increase the number of Nobel laureates in a given country by 1.

http://www.nejm.org/doi/full/10.1056/NEJMon1211064

Unrelated data is the most common mistake



https://doi.org/10.3945/jn.113.174813

Components of a predictor

question -> **input data** -> features -> algorithm -> parameters -> evaluation

Often more data > better models

Data relevant for the question is the most important step!

Components of a predictor

question -> input data -> **features** -> algorithm -> parameters -> evaluation

Properties of good features

Lead to data compression

Retain relevant information

Are created based on expert application knowledge

Common mistakes

Trying to automate feature selection **(automate with care)**

Not paying attention to data-specific quirks

Throwing away information unnecessarily

Components of a predictor

question -> input data -> features -> **algorithm** -> parameters -> evaluation

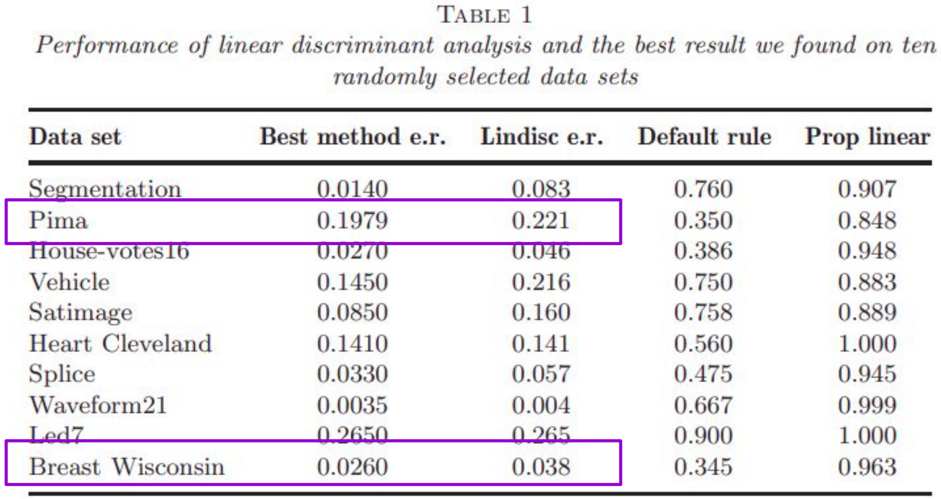


TABLE 1

Performance of linear discriminant analysis and the best result we found on ten randomly selected data sets

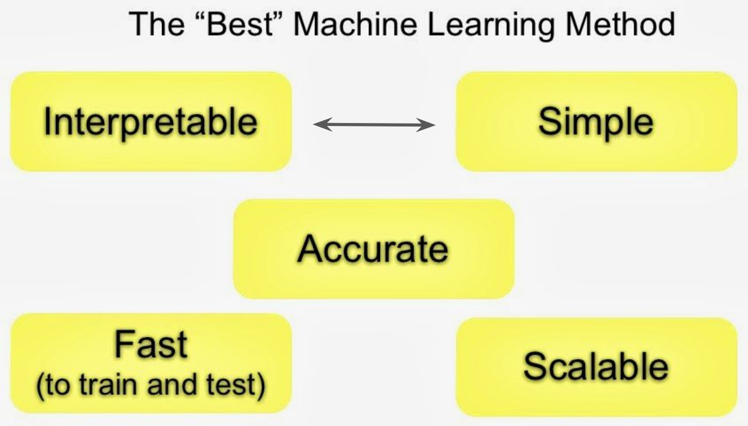
Data set	Best method e.r.	Lindisc e.r.	Default rule	Prop linear
Segmentation	0.0140	0.083	0.760	0.907
Pima	0.1979	0.221	0.350	0.848
House-votes16	0.0270	0.046	0.386	0.948
Vehicle	0.1450	0.216	0.750	0.883
Satimage	0.0850	0.160	0.758	0.889
Heart Cleveland	0.1410	0.141	0.560	1.000
Splice	0.0330	0.057	0.475	0.945
Waveform21	0.0035	0.004	0.667	0.999
Led7	0.2650	0.265	0.900	1.000
Breast Wisconsin	0.0260	0.038	0.345	0.963

e.r. -> error rate
Default rule -> assigning every point to the majority class

https://arxiv.org/pdf/math/0606441.pdf

Components of a predictor

question -> input data -> features -> **algorithm** -> parameters -> evaluation



Relative order of importance

question > data > features > algorithms

Supervised vs Unsupervised learning

- Supervised Learning:
  - Use a data set X to predict or detect association with a response y
  - Teach the computer how to do something, then let it use its new found knowledge to do it
    - ■ Regression
    - ■ Classification
- Unsupervised Learning:
  - Discover the signal in X; no y is available
  - Let the computer learn how to do something, and use this to determine structure and patterns in data
    - ■ Clustering
    - ■ PCA

Steps in supervised machine learning

Pre-processing data: Data cleaning, normalization and data transformation procedures.

Training and test data split: Decide which strategy you want to use for evaluation purposes. You need to use a test set to evaluate your model later on.

Training the model: This is where your choice of supervised learning algorithm becomes relevant. "Training" generally means your data set is used in optimization of the loss function to find parameters for f(x).

Estimating performance of the model: This is about which metrics to use to evaluate performance and how to calculate those metrics.

Model tuning and selection: Trying different parameters and selecting the best model.

The machine learning process

A major goal of the machine learning process is to find an **algorithm** f(X) that most accurately predicts future values (Ŷ) based on a set of features (X).

The algorithm not only fits well past data, but more importantly, predicts a future outcome accurately. This is called the ***generalizability*** of the algorithm.

To assess this, it is crucial how data is divided.

To provide an accurate understanding of the generalizability of the final optimal model, data is split into training and test data sets:

- **Training set**: these data are used to develop feature sets, train algorithms, tune hyperparameters, compare models, and all of the other activities required to choose a final model.
- **Test set**: having chosen a final model, these data are used to estimate an unbiased assessment of the model's performance (*generalization error*).

It is critical that the test set not be used prior to selecting your final model. Assessing results on the test set prior to final model selection biases the model selection process since the testing data will have become part of the model development process.

Selecting the algorithm

While certain algorithms tend to perform better than others with certain types of data, no single algorithm will always outperform all others on all problems. This concept is called the ***no free lunch theorem***: you don't get something for nothing; you need to put some effort into working out the best algorithm for your particular problem.

Data scientists typically choose a few algorithms they know tend to work well for the type of data and problem they are working on, and see which algorithm generates the best-performing model.

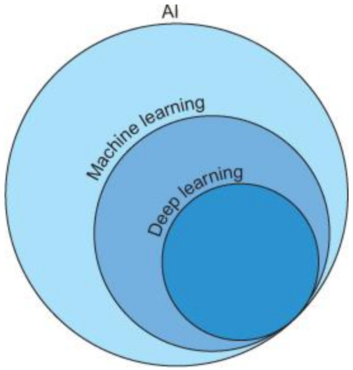
The theorem states that all optimization algorithms perform equally well when their performance is averaged across all possible problems.

Flavors of Supervised Learning

- Regression: Predict a quantitative response, such as blood pressure, cholesterol level, tumor size
- Classification: Predict a categorical response, such as tumor versus normal tissue, heart disease versus no heart disease, subtype of glioblastoma
- Other Flavors: survival analysis, and more . . .

Note on deep learning

DMI



Very popular in the last 5-10 years:

- models with outstanding performance.
- computational power

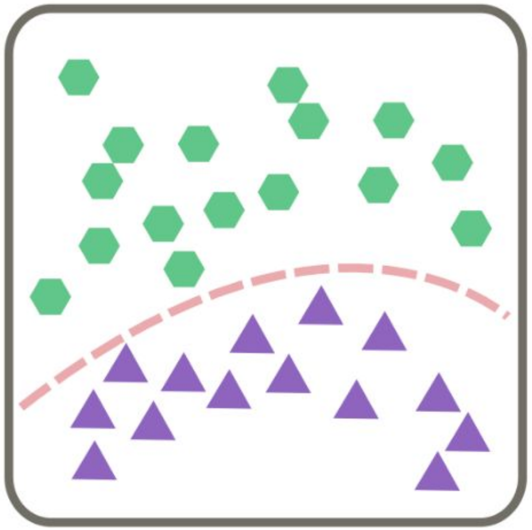
BUT

- Require more data
- Long time (hours or even days!) to train.

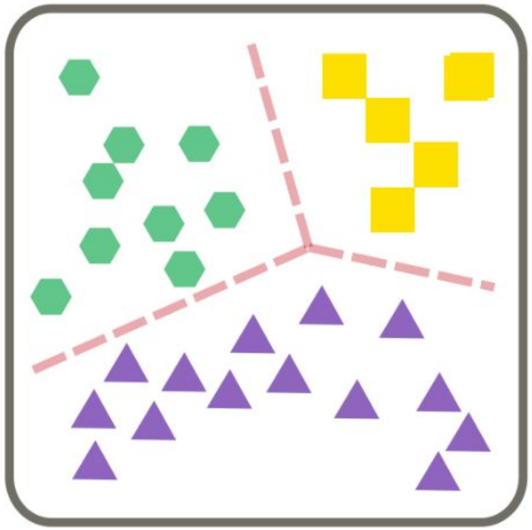
- Computationally expensive (a lot of computing power)
- The rules are less interpretable. By their nature, deep learning models favor performance over model interpretability.

Deep learning algorithms are particularly good at tasks involving complex data, such as image classification, NLP and audio transcription.

Classification



Binary classification



Multi-class classification

Classification vs. Numeric Prediction

Classification

Predict categorical class labels (discrete or nominal)

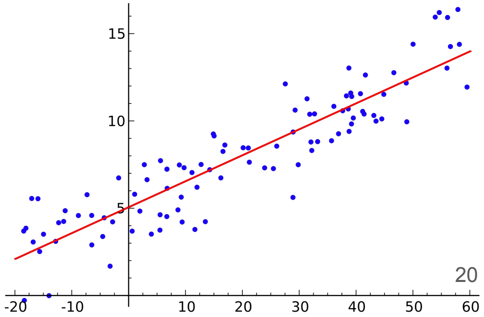
- ○ Construct a model based on the training set and the **class labels** (the values in a classifying attribute) and use it in classifying new data

Numeric prediction

Model continuous-valued functions (i.e., predict unknown or missing values)

Typical applications of classification

- ○ Medical diagnosis: if a tumor is cancerous or benign
- ○ Disease vs non-disease
- ○ Predicting embryonic cell states



Classification—Model Construction, Validation and Testing

Model construction

- Each sample is assumed to belong to a predefined class (shown by the **class label**)
- The set of samples used for model construction is **training set**
- Model: Represented as decision trees, rules, mathematical formulas, or other forms

Model Validation and Testing:

- **Test:** Estimate accuracy of the model
  - The known label of test sample is compared with the classified result from the model
  - *Accuracy:* % of test set samples that are correctly classified by the model
  - Test set is independent of training set
- **Validation**: If *the test set* is used to select or refine models, it is called **validation** (or development) **(test) set**

Model Deployment: If the accuracy is acceptable, use the model to classify new data

Classification: Definition

- Given a collection of records (training set)

– Each record contains a set of attributes, one of the attributes is the class.

- Find a model for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.

– A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model

and test set used to validate

Classification Techniques

- Logistic Regression
- Decision Tree
- Naïve Bayes
- Rule-based Classification
- K Nearest Neighbor
- Support Vector Machines
- Ensemble methods

K-Nearest Neighbors

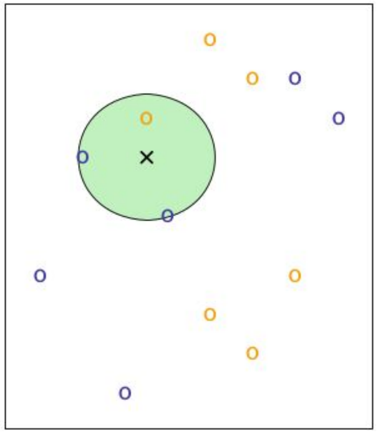
DMI

- K-nearest neighbor (KNN) is a very simple algorithm in which each observation is predicted based on its “similarity” to other observations.

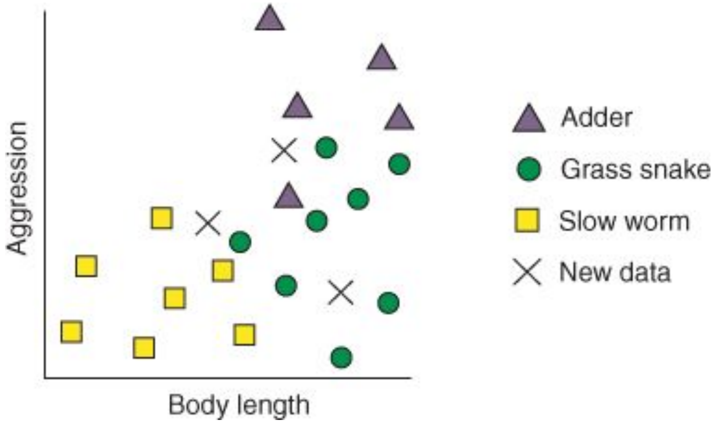
- The most common distance measures are the Euclidean and Manhattan distance metrics

- The KNN algorithm identifies **k** observations that are “similar” or nearest to the new record being predicted and then uses the average response value (regression) or the most common class (classification) of those k observations as the predicted output.

-



K-Nearest Neighbors

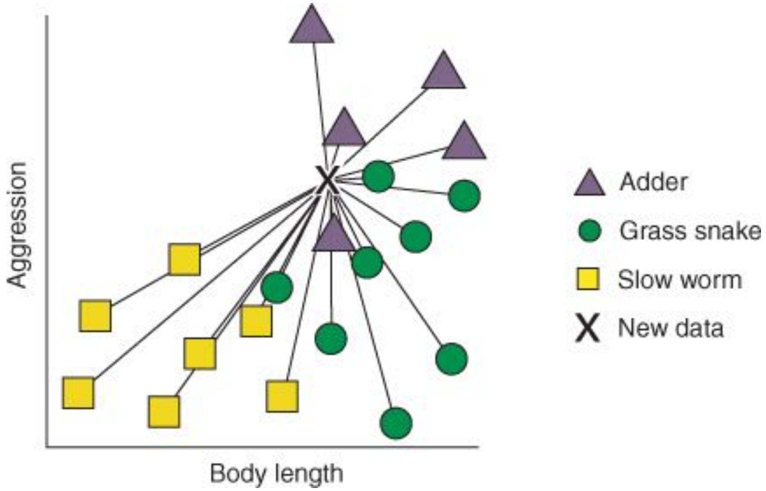


Body length and aggression of reptiles.

Labeled cases for adders, grass snakes, and slow worms are indicated by their shape.

New, unlabeled data are shown by black crosses.

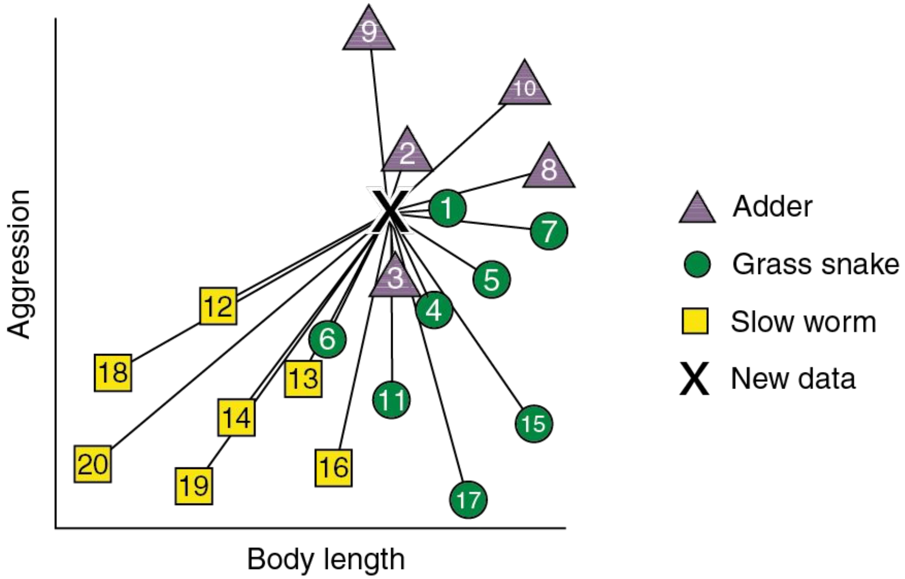
K-Nearest Neighbors



The first step of the kNN algorithm: calculating distance.

The lines represent the distance between one of the unlabeled cases (the cross) and each of the labeled cases.

K-Nearest Neighbors

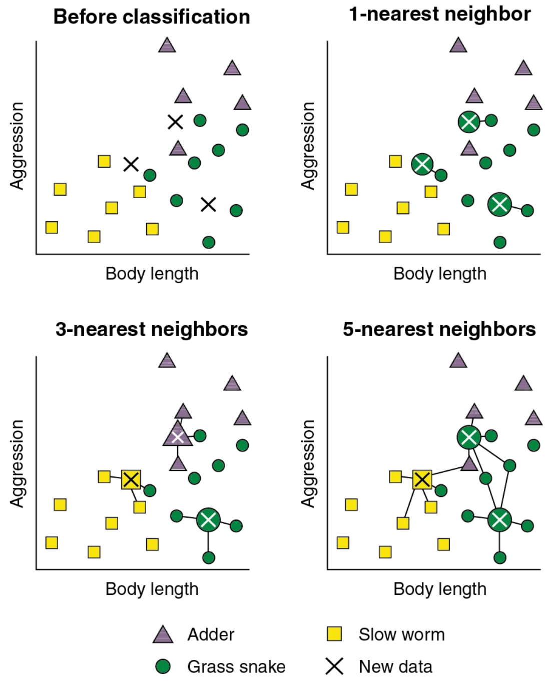


The second step of the kNN algorithm: ranking the neighbors.

The lines represent the distance between one of the unlabeled cases (the cross) and each of the labeled cases.

The numbers represent the ranked distance between the unlabeled case (the cross) and each labeled case (1 = closest).

K-Nearest Neighbors



The final step of the kNN algorithm: identifying the k-nearest neighbors and taking the majority vote.

Lines connect the unlabeled data with their one, three, and five nearest neighbors.

The majority vote in each scenario is indicated by the shape drawn under each cross.

used for both classification and regression problems

K-Nearest Neighbors

- ✓ Different features might have different scales. For example, we can have a measure of pain scaling from 1 to 10 or 1 to 100.
- ✓ Some similarity or distances measures assume the same measuring unit in all feature dimensions.
- ✓ This requires that the data may need to be transferred into the same scale.
- ✓ Re-scaling can make each feature contribute to the distance in a relatively equal manner, avoiding potential bias.

KNNs: Choosing k

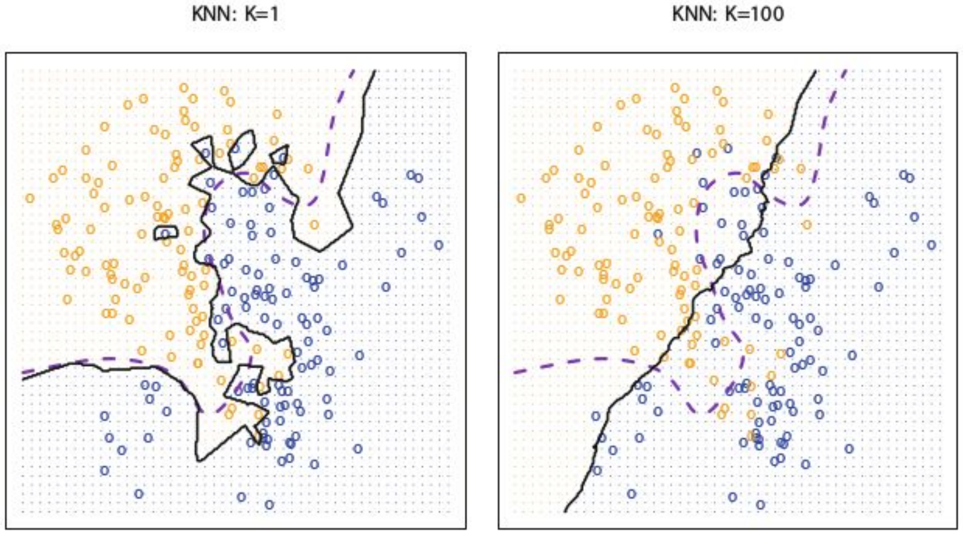
The performance of KNNs is very sensitive to the choice of k:

- In general, low values of k typically overfit and large values often underfit.
- At the extremes, when k=1 , we base our prediction on a single observation that has the closest distance measure
- When k=n , we are simply using the average (regression) or most common class (classification) across all training samples as our predicted value.

There is no general rule about the best k as it depends greatly on the nature of the data. For high signal data with very few noisy (irrelevant) features, smaller values of k tend to work best. As more irrelevant features are involved, larger values of k are required to smooth out the noise.

When using KNN for classification, it is best to assess odd numbers for k to avoid ties in the event there is equal proportion of response levels

KNNs: Choosing k



A comparison of the KNN decision boundaries (solid black curves) obtained using K = 1 and K = 100

KNNs

KNNs are a very simplistic, and intuitive, algorithm that can provide average to decent predictive power, especially when the response is dependent on the local structure of the features.

A major drawback of KNNs is their computation time, which increases by $n \times p$ for each observation.

Although KNNs rarely provide the best predictive performance, they have many benefits, for example, in feature engineering and in data cleaning and preprocessing (missing data imputation)

Predicting with trees

Key ideas

- Iteratively split variables into groups
- Evaluate "homogeneity" within each group
- Split again if necessary

Pros:

- Easy to interpret
- Better performance in nonlinear settings

Cons:

- Without pruning/cross-validation can lead to overfitting
- Harder to estimate uncertainty
- Results may be variable

Tree-based models are a class of nonparametric algorithms that work by partitioning the feature space into a number of smaller (non-overlapping) regions with similar response values using a set of splitting rules.

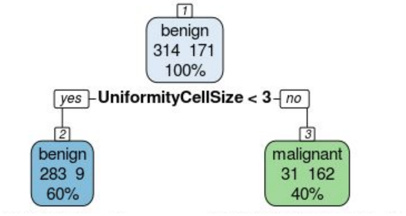
Example: the breast cancer dataset

Sample number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
1057013	8	4	5	1	2	4	7	3	1	Malignant
1096800	6	6	6	9	6	2	7	8	1	Benign
1183246	1	1	1	1	1	2	2	1	1	Benign
1184840	1	1	3	1	2	2	2	1	1	Benign
1193683	1	1	2	1	3	3	1	1	1	Benign

1193698	1	1	2	1	3	3	1	1	1	?

https://www.kaggle.com/datasets/yasserh/breast-cancer-dataset

Example Tree



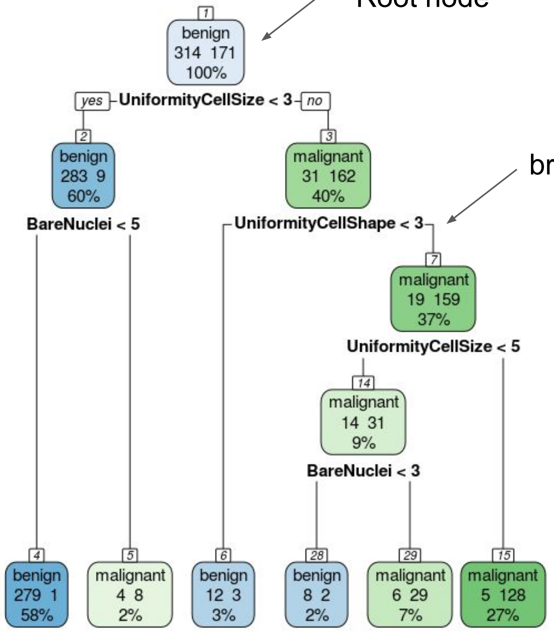
Each node shows

- the predicted class (benign or malignant)
- the number of observations per class
- the percentage of observations with respect to the total

Basic algorithm
- Start with all variables in one group
- Find the variable/split that best separates the outcomes
- Divide the data into two groups ("leaves") on that split ("node")
- Within each split, find the best variable/split that separates the outcomes
- Continue until the groups are too small or sufficiently "pure"

Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses

Example Tree



Root node

branch

leaf

Each node shows

- the predicted class (benign or malignant)
- the number of observations per class
- the percentage of observations with respect to the total

Basic algorithm

- Start with all variables in one group
- Find the variable/split that best separates the outcomes
- Divide the data into two groups ("leaves") on that split ("node")
- Within each split, find the best variable/split that separates the outcomes
- Continue until the groups are too small or sufficiently "pure"

Measures of impurity

Number of times that class k appears in leaf m

It is the **probability of being wrong** if we predict the majority class.

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \ in \ Leaf \ m} 1(y_i = k)$$

N is the number of total objects in leaf m

Misclassification Error:

$$1 - \hat{p}_{mk(m)}; k(m) = \mathrm{most; common; k}$$

- 0 = perfect purity
- 0.5 = no purity

Gini index:

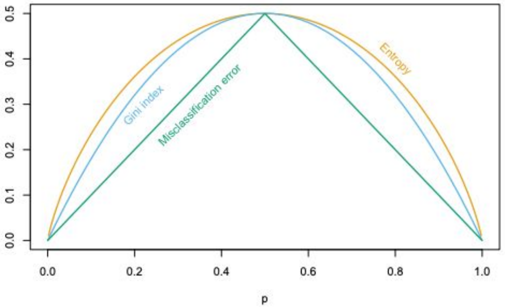
how likely it is that two randomly chosen observations from a node belong to different classes,

$$\sum_{k \neq k'} \hat{p}_{mk} \times \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^{K} p_{mk}^2$$

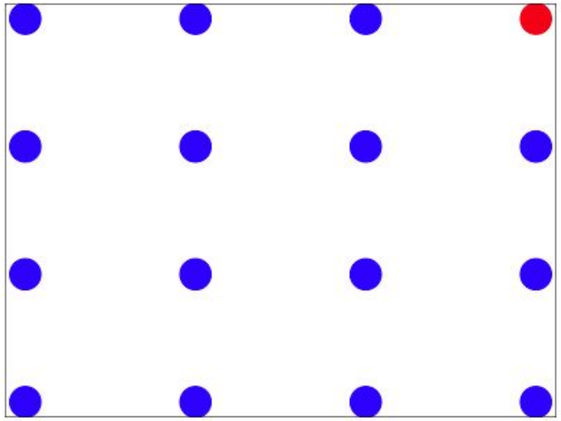
the sum of the squared probabilities of each class

- 0 = perfect purity
- 0.5 = no purity

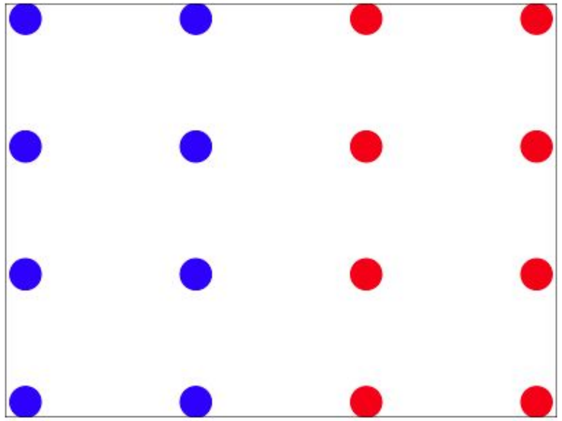
For 2 classes



Measures of impurity

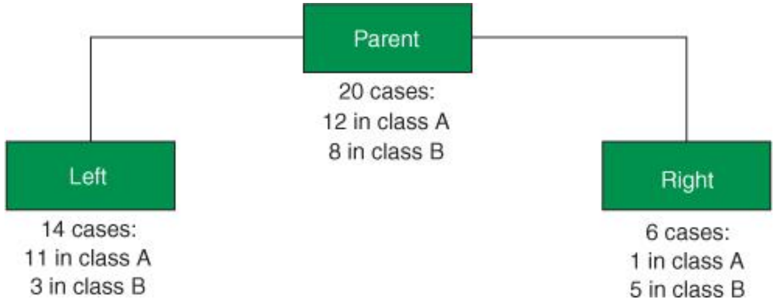


- **Misclassification:** $1/16 = 0.06$
- **Gini:** $1 - [(1/16)^2 + (15/16)^2] = 0.12$



- **Misclassification:** $8/16 = 0.5$
- **Gini:** $1 - [(8/16)^2 + (8/16)^2] = 0.5$

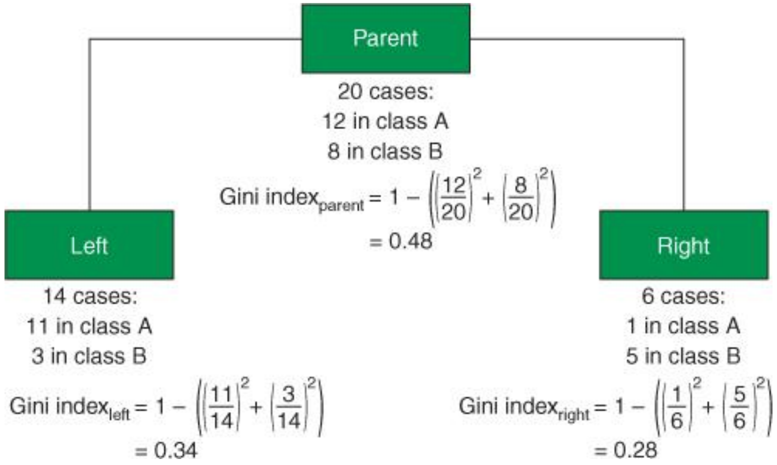
Predicting with trees



the Gini gain of this split is the difference between the Gini index of the parent node and the Gini index of the split.

$$\text{Gini index} = 1 - (p(A)^2 + p(B)^2)$$

Predicting with trees



Parent

20 cases:
12 in class A
8 in class B

$$\text{Gini index}_{\text{parent}} = 1 - \left(\left(\frac{12}{20}\right)^2 + \left(\frac{8}{20}\right)^2\right) = 0.48$$

Left

14 cases:
11 in class A
3 in class B

$$\text{Gini index}_{\text{left}} = 1 - \left(\left(\frac{11}{14}\right)^2 + \left(\frac{3}{14}\right)^2\right) = 0.34$$

Right

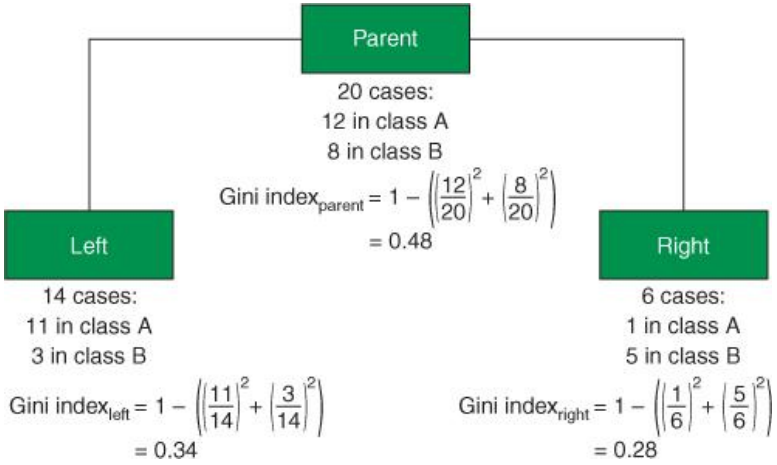
6 cases:
1 in class A
5 in class B

$$\text{Gini index}_{\text{right}} = 1 - \left(\left(\frac{1}{6}\right)^2 + \left(\frac{5}{6}\right)^2\right) = 0.28$$

$$\text{Gini index} = 1 - (p(A)^2 + p(B)^2)$$

The Gini gain at a particular node is calculated for each predictor variable, and the predictor that generates the largest Gini gain is used to split that node. This process is repeated for every node as the tree grows

Predicting with trees



Parent

20 cases:
12 in class A
8 in class B

$$\text{Gini index}_{\text{parent}} = 1 - \left(\left(\frac{12}{20}\right)^2 + \left(\frac{8}{20}\right)^2\right) = 0.48$$

Left

14 cases:
11 in class A
3 in class B

$$\text{Gini index}_{\text{left}} = 1 - \left(\left(\frac{11}{14}\right)^2 + \left(\frac{3}{14}\right)^2\right) = 0.34$$

Right

6 cases:
1 in class A
5 in class B

$$\text{Gini index}_{\text{right}} = 1 - \left(\left(\frac{1}{6}\right)^2 + \left(\frac{5}{6}\right)^2\right) = 0.28$$

$$\text{Gini index}_{\text{split}} = p(\text{left}) \times \text{Gini index}_{\text{left}} + p(\text{right}) \times \text{Gini index}_{\text{right}}$$

$$\text{Gini index}_{\text{split}} = \frac{14}{20} \times 0.34 + \frac{6}{20} \times 0.28 = 0.32$$

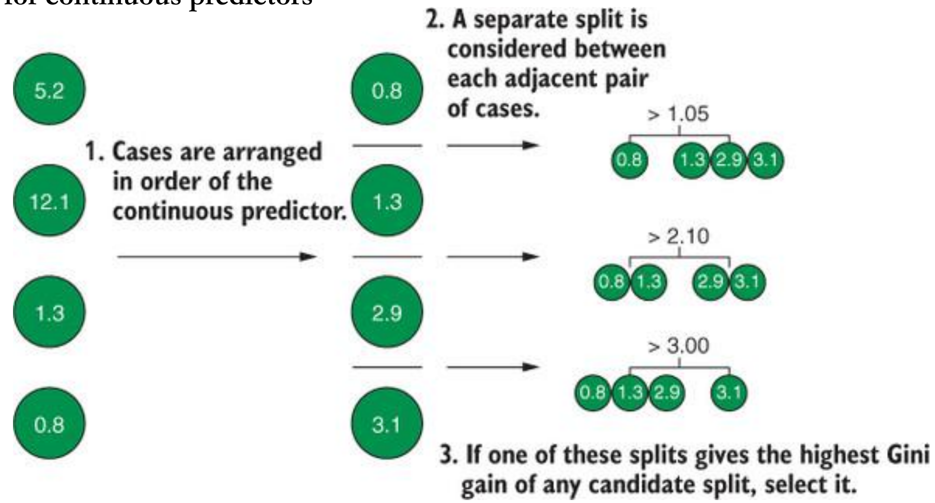
Gini gain = 0.48 – 0.32 = 0.16

$$\text{Gini index} = 1 - (p(A)^2 + p(B)^2)$$

The Gini gain at a particular node is calculated for each predictor variable, and the predictor that generates the largest Gini gain is used to split that node. This process is repeated for every node as the tree grows

Predicting with trees

split for continuous predictors

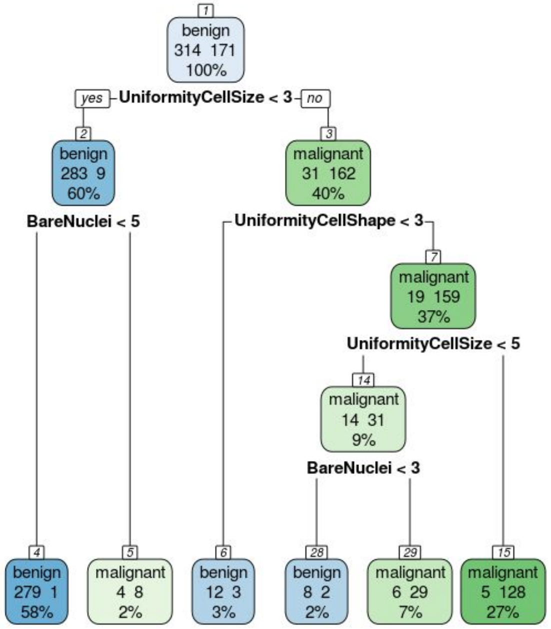


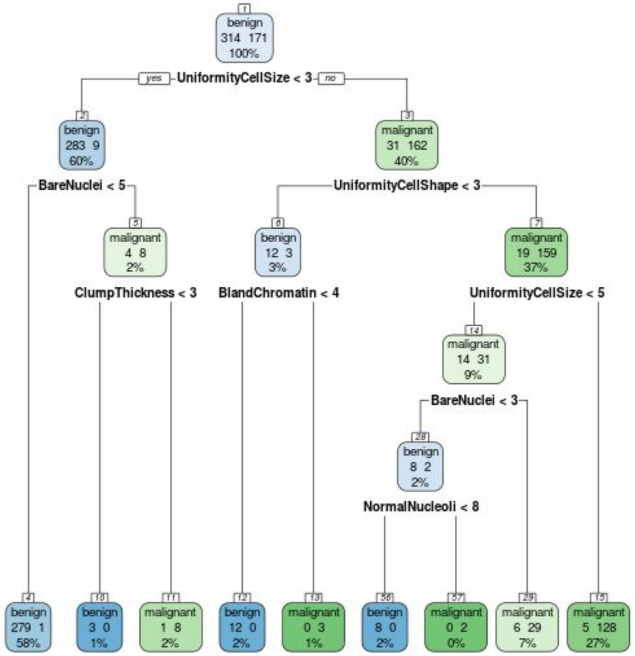
1. Cases are arranged in order of the continuous predictor.

2. A separate split is considered between each adjacent pair of cases.

3. If one of these splits gives the highest Gini gain of any candidate split, select it.

How deep?


1
benign
314 171
100%
yes
UniformityCellSize < 3
no
2
benign
283 9
60%
BareNuclei < 5
3
malignant
31 162
40%
UniformityCellShape < 3
7
malignant
19 159
37%
UniformityCellSize < 5
14
malignant
14 31
9%
BareNuclei < 3
4
benign
279 1
58%
5
malignant
4 8
2%
6
benign
12 3
3%
28
benign
8 2
2%
29
malignant
6 29
7%
15
malignant
5 128
27%


1
benign
314 171
100%
yes
UniformityCellSize < 3
no
2
benign
283 9
60%
BareNuclei < 5
5
malignant
4 8
2%
ClumpThickness < 3
3
malignant
31 162
40%
UniformityCellShape < 3
6
benign
12 3
3%
BlandChromatin < 4
7
malignant
19 159
37%
UniformityCellSize < 5
14
malignant
14 31
9%
BareNuclei < 3
28
benign
8 2
2%
NormalNucleoli < 8
4
benign
279 1
58%
10
benign
3 0
1%
11
malignant
1 8
2%
12
benign
12 0
2%
13
malignant
0 3
1%
56
benign
8 0
2%
57
malignant
0 2
0%
29
malignant
6 29
7%
15
malignant
5 128
27%

Model Selection for Decision Trees

Pre-Pruning (Early Stopping Rule)

– Stop the algorithm before it becomes a fully-grown tree

– Typical stopping conditions for a node:

Stop if all instances belong to the same class

Stop if all the attribute values are the same

– More restrictive conditions:

Stop if number of instances is less than some user-specified threshold

Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

Stop if estimated generalization error falls below certain threshold

Model Selection for Decision Trees

Post-Pruning

– Grow decision tree to its entirety

– Subtree replacement

  Trim the nodes of the decision tree in a bottom-up fashion

  If generalization error improves after trimming, replace sub-tree by a leaf node

  Class label of leaf node is determined from majority class of instances in the sub-tree

Decision trees

Strengths

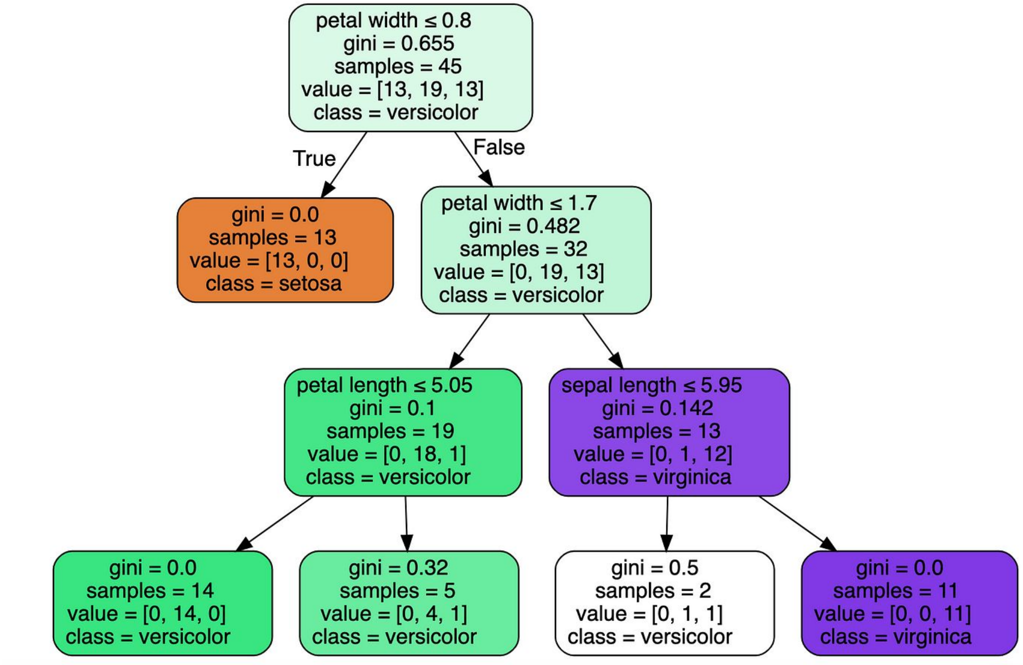
- ✓ Easy to Understand.
- ✓ Useful in Data exploration. Decision tree is one of the fastest way to identify most significant variables and relation between two or more variables.
- ✓ Less data cleaning required. It requires less data cleaning compared to some other modeling techniques. It is not influenced by outliers and missing values to a fair degree.
- ✓ Data type is not a constraint. It can handle both numerical and categorical variables.
- ✓ Decision tree is considered to be a non-parametric method. This means that decision trees have no assumptions about the space distribution and the classifier structure.

Decision trees

Weaknesses

- ✓ Overfitting: This problem gets solved by setting constraints on model parameters and pruning
- ✓ trees can be very non-robust: a small change in the data can cause a large change in the final estimated tree.
- ✓ individual decision trees generally do not often achieve state-of-the-art predictive accuracy.

Multiclass decision tree



How do we know how good our model is?

An example: the breast cancer dataset

Sample number	True Class	Predicted Class
1057013	Malignant	Malignant
1096800	Benign	Benign
1183246	Benign	Malignant
1184840	Benign	Benign
1193683	Benign	Benign

Metrics for Performance Evaluation

- Confusion Matrix:

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10

		Predicted Class	
		disease	Non-disease
Actual Class	disease	0	10
	Non-disease	0	9990

- If model predicts everything to be class 0, accuracy is 99.9 %

Accuracy is misleading because model does not detect any class 1 example

Cost-Sensitive Measures

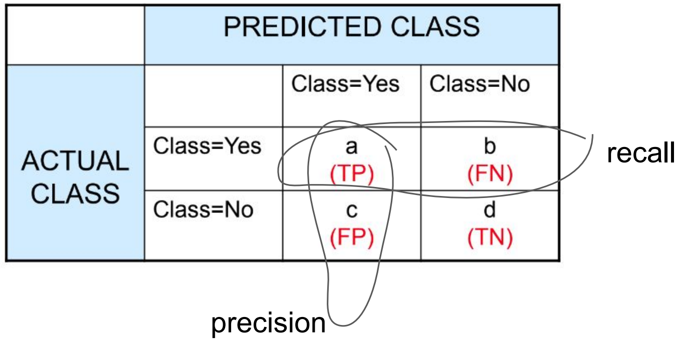
$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F - measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

$$\text{Precision} = \frac{tp}{tp+fp}$$

$$\text{Recall} = \frac{tp}{tp+fn}$$



		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

Receiver Operating Characteristic (ROC) curve

In binary classification you are predicting one of two categories

- Alive/dead
- Disease/Healthy

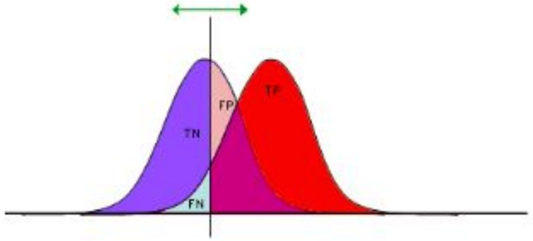
But your predictions are often quantitative:

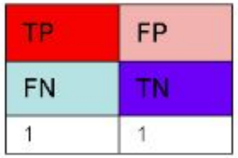
- Probability of being alive

The cutoff you choose gives different results

Receiver Operating Characteristic (ROC) curve

- AUC = 0.5: random guessing
- AUC = 1: perfect classifier
- In general AUC of above 0.8 considered "good"

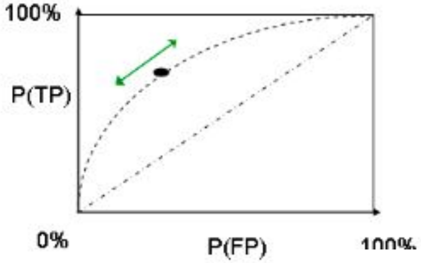




TP	FP
FN	TN
1	1

True positive rate (recall, sensitivity)

$$TPR = \frac{TP}{TP + FN}$$

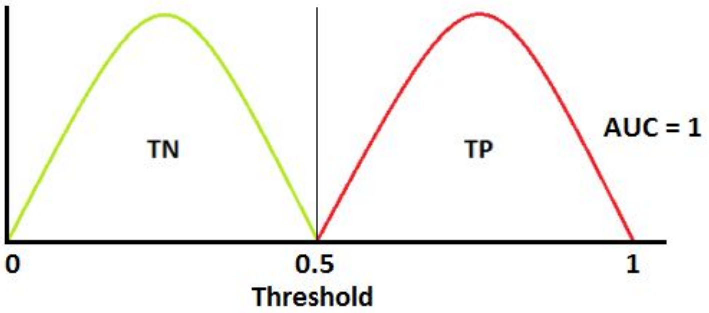


AUC: Area under the curve

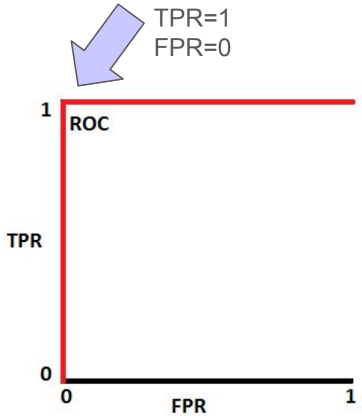
False Positive Rate $FPR = \frac{FP}{FP + TN}$

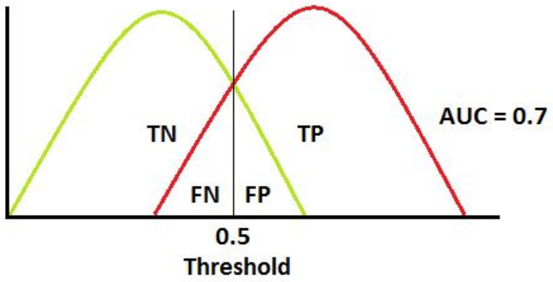
probability of false alarm, FPR= 1-specificity

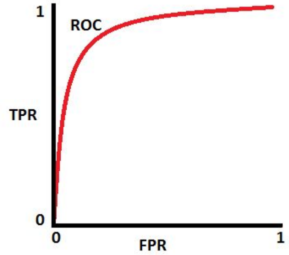
ROC curve: what is good?



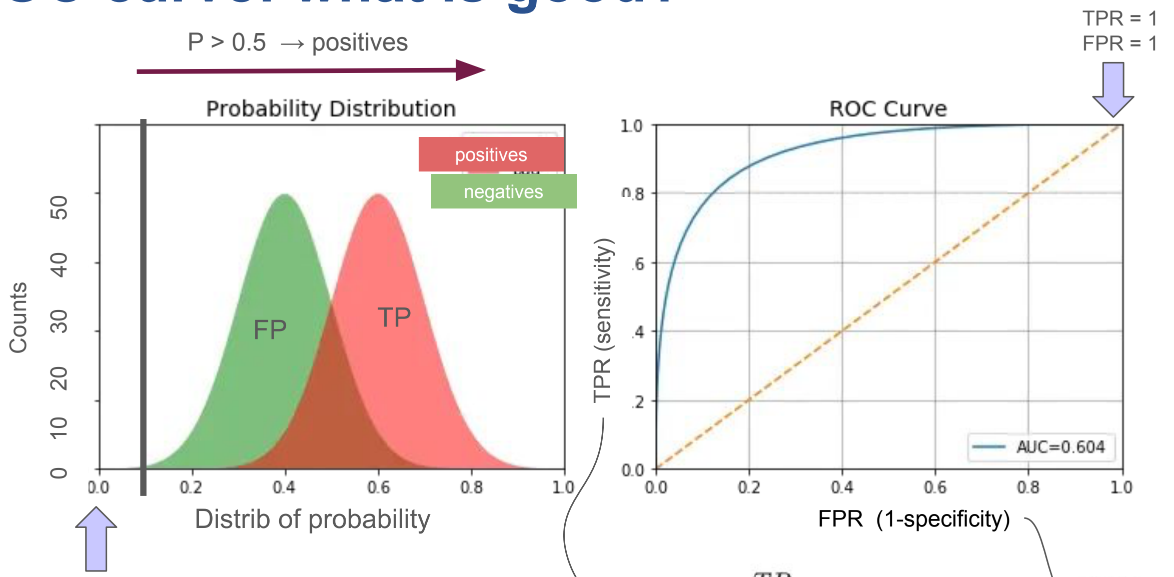
P=0.5
TPR=1
FPR=0







ROC curve: what is good?



$$TPR = \frac{TP}{TP + FN}$$

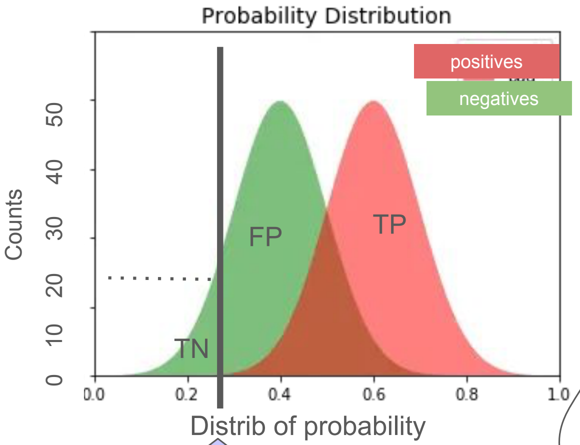
$$FPR = \frac{FP}{FP + TN}$$

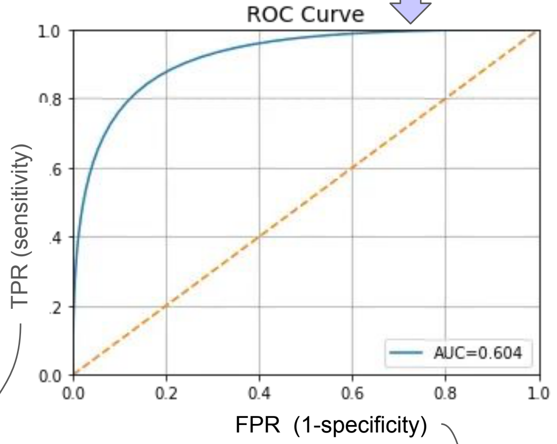
positives
100% TP
0 FN
TPR =1
Sensitivity=1

negative
100% FP
0 TN
FPR =1
Specificity=0

ROC curve: what is good?

TPR = 1
FPR < 1





positives
100% TP
0 FN
TPR =1

negative
225 FP
25 TN
FPR =225/250

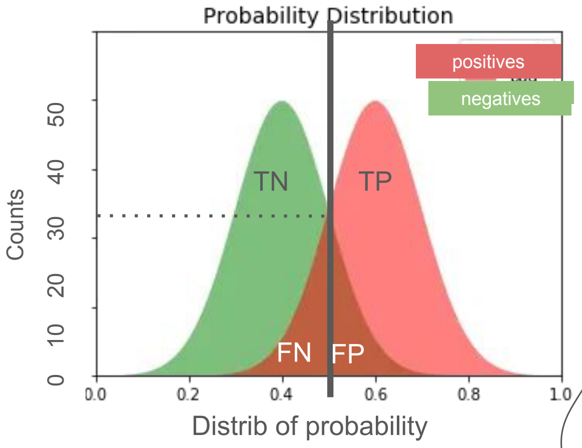
$$TPR = \frac{TP}{TP + FN}$$

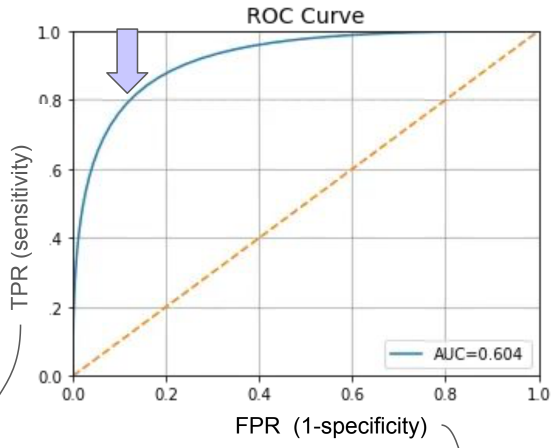
$$FPR = \frac{FP}{FP + TN}$$

$$Specif \; = \; \frac{TN}{TN+FP}$$

ROC curve: what is good?

TPR < 1
FPR < 1





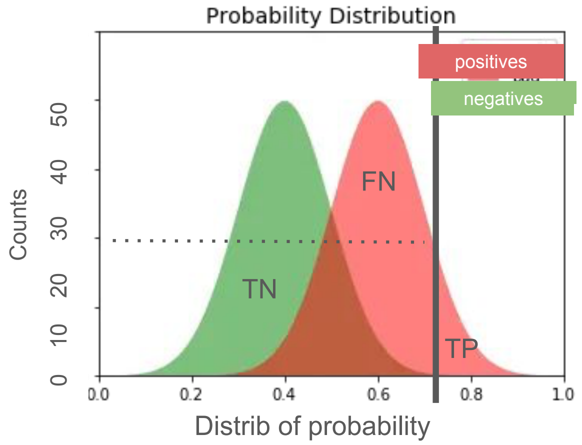
$$TPR = \frac{TP}{TP + FN}$$

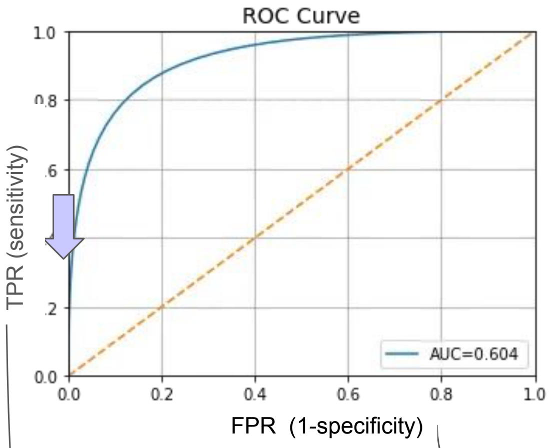
$$FPR = \frac{FP}{FP + TN}$$

positives	negative
220 TP	30 FP
30 FN	220 TN
TPR =220/250	FPR =30/250

ROC curve: what is good?

TPR < 1
FPR = 0





positives
30 TP
220 FN
TPR =30/250

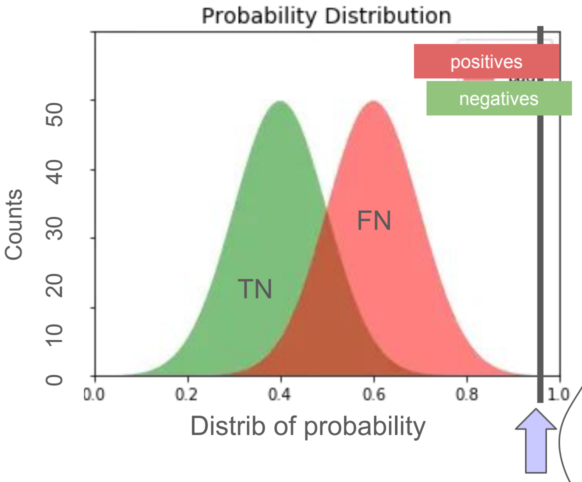
negative
0 FP
100% TN
FPR =0

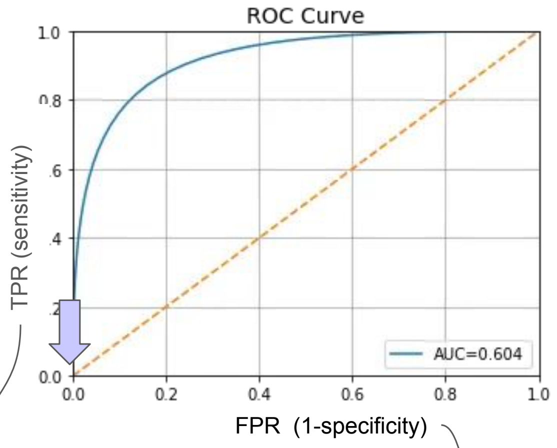
$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

ROC curve: what is good?

TPR = 0
FPR = 0





$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

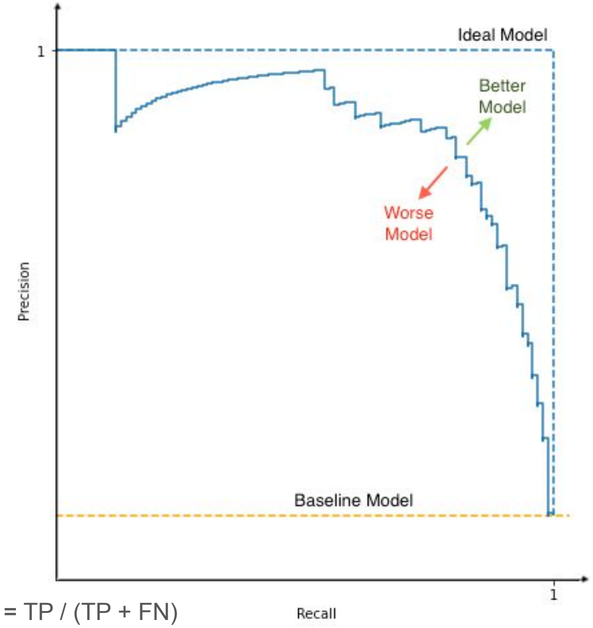
$$Specif = \frac{TN}{TN+FP}$$

positives
0% TP
100% FN
TPR =0
Sensitivity=0

negative
0 FP
100% TN
FPR =0
Specificity=1

Precision-Recall curves

Precision = TP / (TP + FP)



Recall = TP / (TP + FN)

Precision-Recall curves are often considered more informative than ROC curves when dealing with unbalanced classes. This is because Precision and Recall focus on the performance of a classifier in the positive class, which can be crucial when one class is significantly smaller than the other.

The tidymodels framework

DMI

- ✓ **parsnip** for model definition
- ✓ **recipes** for data preprocessing and feature engineering
- ✓ **rsample** to resample data (useful for cross-validation)
- ✓ **yardstick** to evaluate model performance
- ✓ **dials** to define tuning parameters of your models
- ✓ **tune** for model tuning
- ✓ **workflows** which allows you to bundle everything together

Hands on session

Go to this page and accept the assignment

Bibliography

Statistics versus machine learning. Nat Methods 15, 233–234 (2018).

https://doi.org/10.1038/nmeth.4642

Elements of Statistical Learning, Data Mining, Inference, and Prediction

Trevor Hastie, Robert Tibshirani, Jerome Friedman

An Introduction to Statistical Learning, with Applications in R

Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani