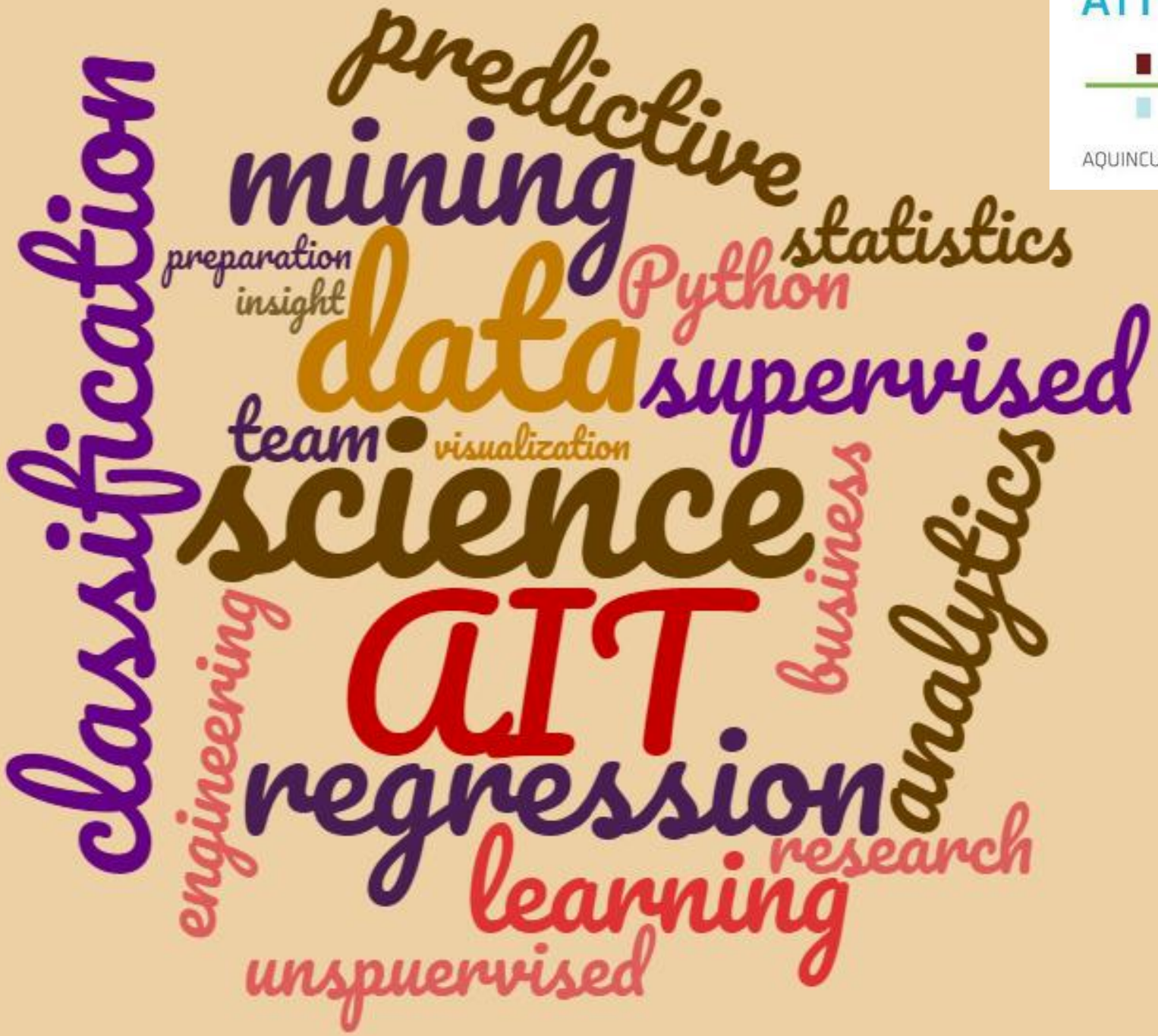
Data Science

March 7, 2023
Overfitting, model evaluation

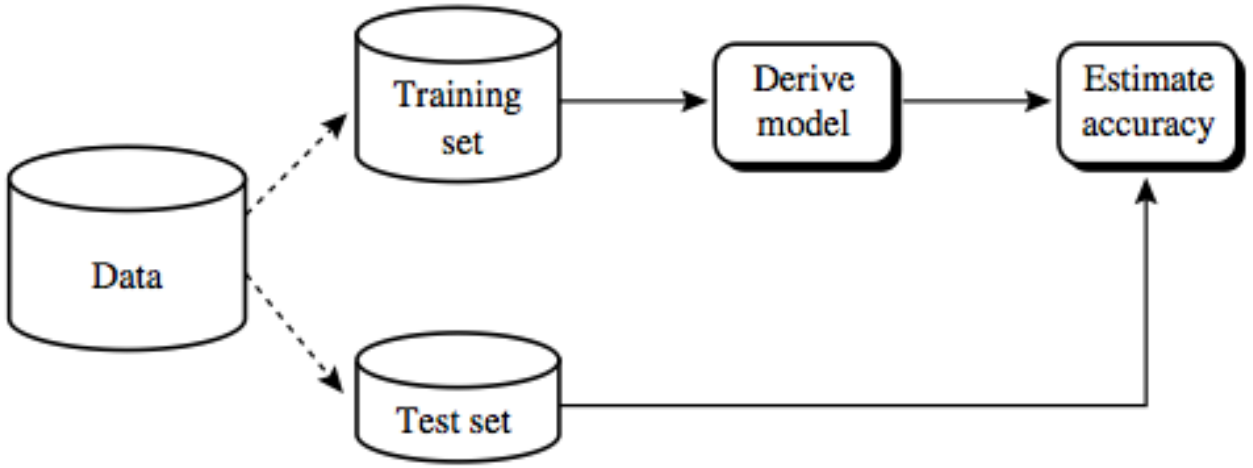


AIT-BUDAPEST
AQUINCUM INSTITUTE OF TECHNOLOGY

Dr. Roland Molontay

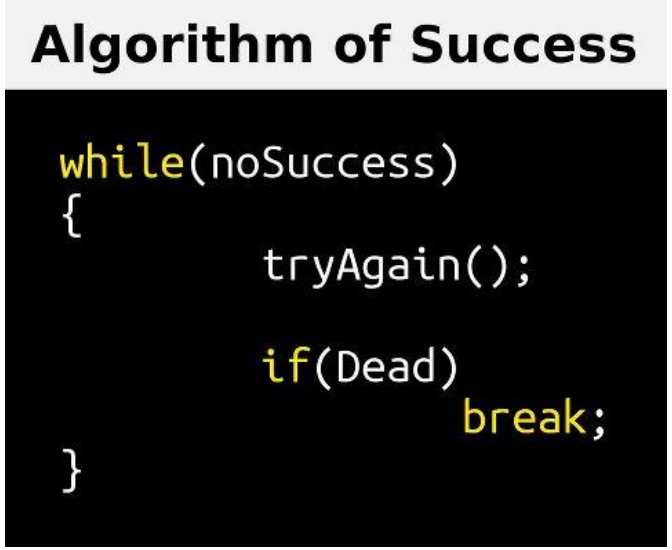
Generalization ability

- Purpose: to build a model that predicts the target variable well in general not just on the available data set ➔ good generalization ability
- Dataset is divided into two (or three) parts
- Cross validation
- To evaluate models, a numerical „goodness” notion is needed



Which model is the best?

- Scenario:
  - We aim to solve a supervised learning task
  - We have training data
  - We aim to give prediction on a target variable
  - We have candidate algorithms (models) to solve the problem
- Question:
  - How to choose which model (algorithm) to use?



Training and test set

First approach: split the data set into two parts

- Training set: fitting the model (i.e. optimizing its parameters) on the training set in such a way that it has a good performance on the training set and has a good generalization ability
- Test set: we test the model performance on data that were not seen by the model before
  - We choose the model that has the best performance on the test set

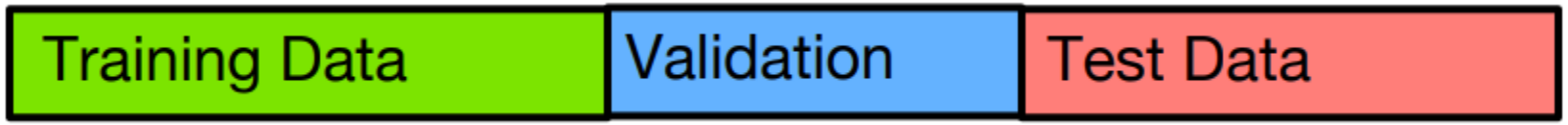
Problems:

- It overestimates the test performance of the best model („lucky” model)
- The algorithm never has access to the test data

Using validation set

Second approach: split the data into three parts

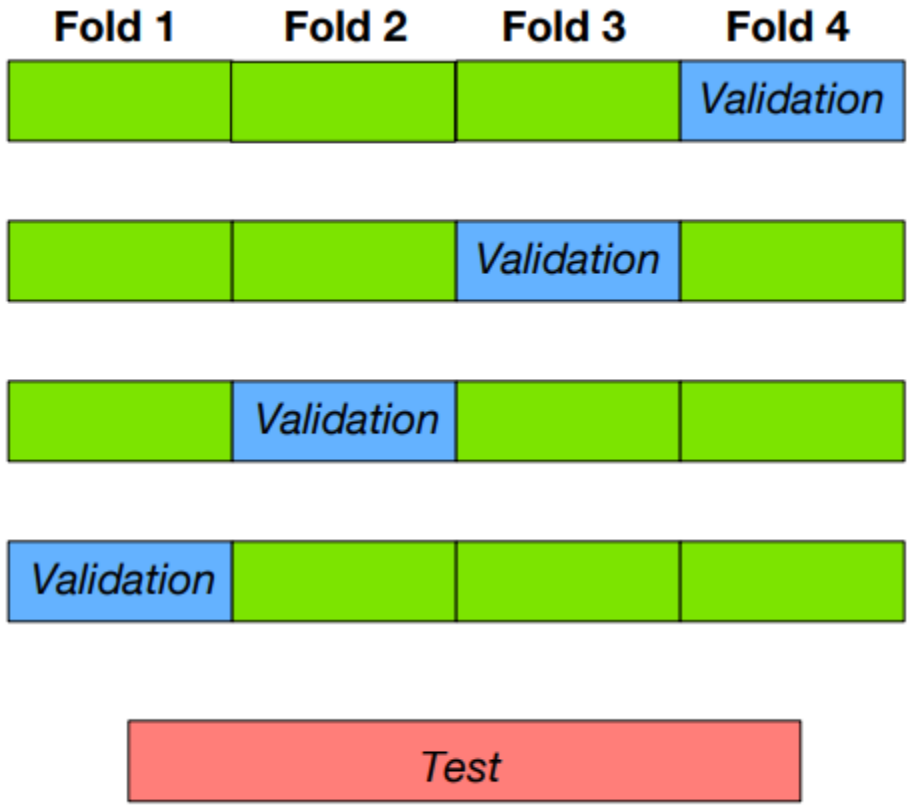
- Training, validation and test data
- Fitting the models on the training data
- Evaluating the models on the validation data
  - Choosing the best performing model
- The chosen model is also evaluated on the test data



- Problems:
  - Wasting training data (neither the validation set nor the test set can be used for training)

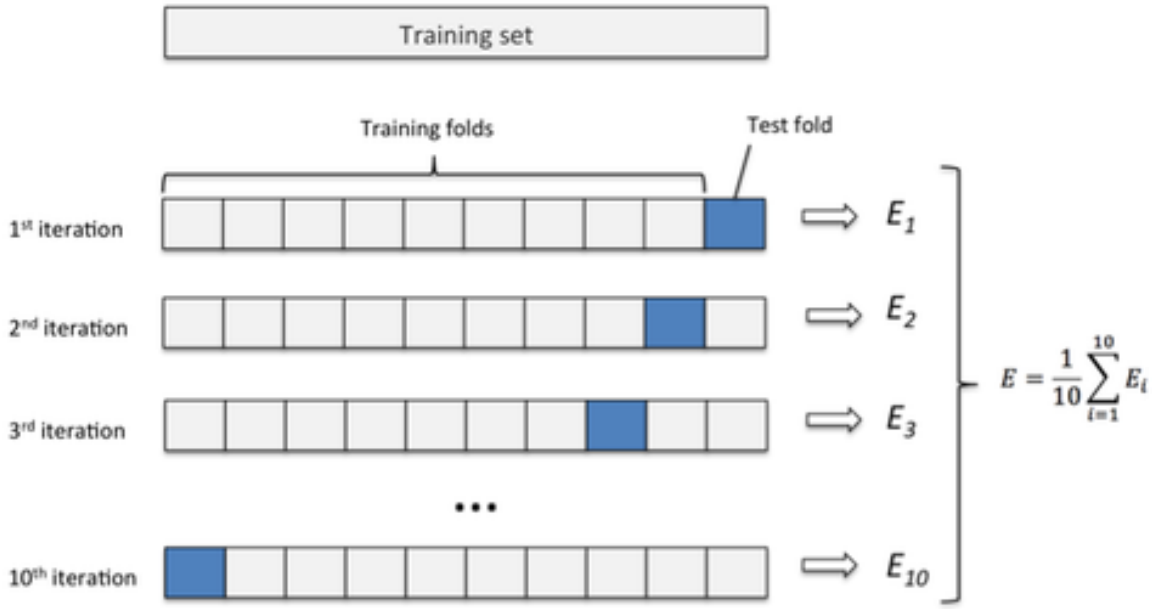
Cross validation

- Split the training data for *K* equal parts (folds)
- Fit the models on *K-1* parts and evaluate them on the rest
  - Do it in all possible *K*-ways
- We choose the model according to the best average performance of the K evaluations
- The chosen model can be evaluated again on a separate test set



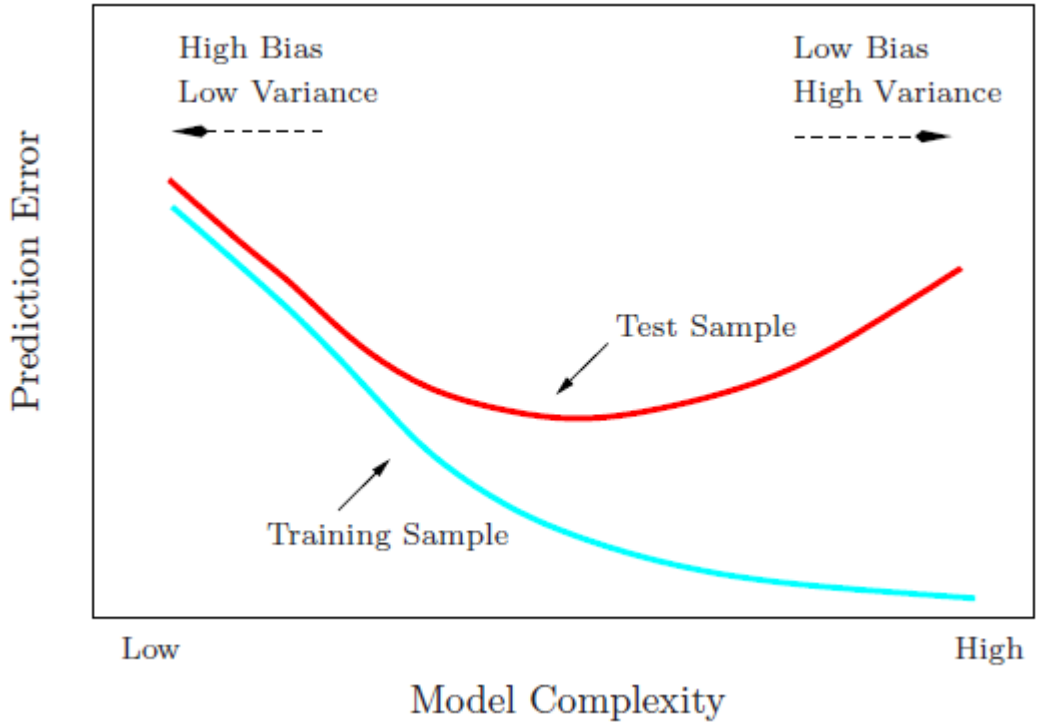
K-fold cross validation

- How to choose *K*?
  - If *K* is too small: the model is fitted on a small data set that leads to high bias
  - If *K* is too large: Few data points are used for evaluation, the performance estimation is noisy that leads to high variance
    - *K=N*: „leave-one-out” cross-validation
  - In practice: 5- or 10- fold cross validation is typical
- The learning algorithm has to be run *K* times but it can be parallelized



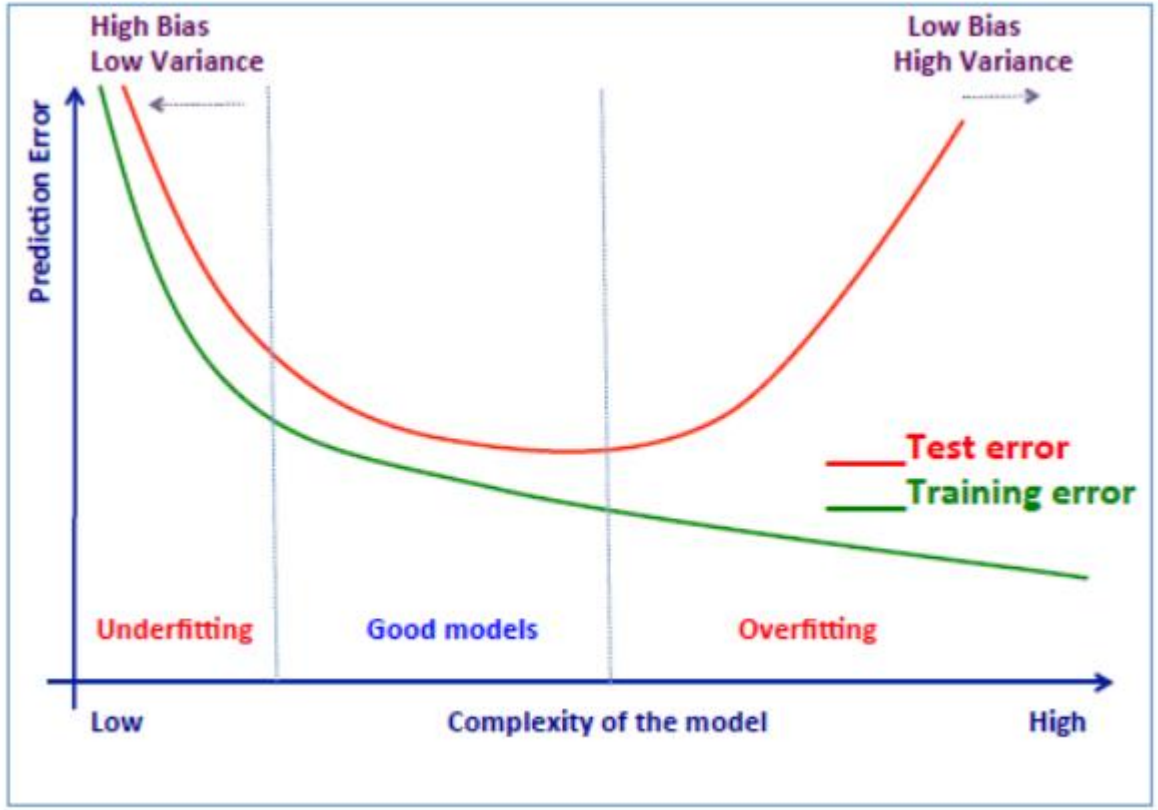
Underfitting - overfitting

- Underfitting: the model is not sophisticated enough to make good predictions
  - Error rate is high both on training data and on test data
  - We should build a more complex model (maybe more training data is needed)
- Overfitting: the model corresponds too closely to a particular set of data and doesn't generalize well
  - Reduce the complexity of the model



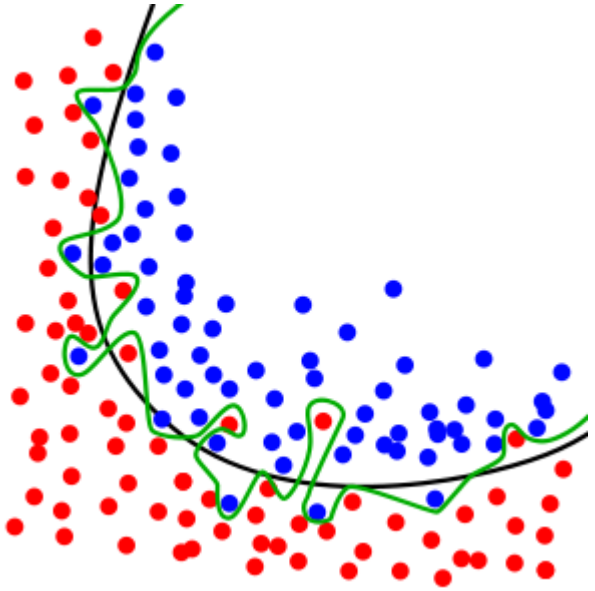
Under- and overfitting

- We say that model *M* underfits, if ∃ *M'* such that:
  - $error_{training}(M) > error_{training}(M')$
  - $error_{test}(M) > error_{test}(M')$
- We say that model M overfits if ∃ M' such that
  - $error_{training}(M) < error_{training}(M')$
  - $error_{test}(M) > error_{test}(M')$



Causes of overfitting

- Indirect cause: the model is too sophisticated, too complex, contains more parameters than can be justified by the data
- Direct cause:
  - Non-representative data (great number of special cases)
  - Noisy data
  - We choose from too many models: the best will have a good performance on the data set that we evaluate on with no warranty that it will have a similarly good performance on other data



Occam's razor

The principle is attributed to the medieval philosopher William of Occam

- Simpler solutions are more likely to be correct than complex ones
- Among competing hypothesis one should select the solution with the fewest assumption
- The principle is also used in machine learning: Build models in such a way that it penalizes too complex models

Confusion matrix

- Confusion matrix for binary classification problems:

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Performance indicators for binary classification

- Accuracy: $A = \frac{TP + TN}{TP + TN + FP + FN}$
  - It can be weighted as well
  - Not a good measure for unbalanced classes
- Precision: $P = \frac{TP}{TP + FP}$
  - The ratio of correctly predicted positive observations to the total predicted positive observations
- Recall (sensitivity, TPR): $R = \frac{TP}{TP + FN}$
  - The ratio of correctly predicted positive observations to the all observations in actual positive class
- F1 score (F-measure): $F = \frac{2 \cdot P \cdot R}{P + R} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$
  - Harmonic mean of Precision and Recall

Further performance indicators

		Predicted condition			
	Total population	Predicted Condition positive	Predicted Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
True condition	condition positive	**True positive**	**False Negative** (Type II error)	True positive rate (TPR), Sensitivity, Recall $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$
	condition negative	**False Positive** (Type I error)	**True negative**	False positive rate (FPR), Fall-out $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$
	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

Problem

The following table summarizes a data set with three attributes (A, B, C) and two class labels (+, –). Build a two-level decision tree.

1. Use misclassification error as inhomogenity measure. Calculate the gains for each attribute. Which attribute gives the best split?
2. Repeat the previous step for the two children of the root node. Which nodes should be split, and which is the second splitting attribute?
3. Calculate Accuracy, Error rate, Precision, Recall, and F-measure!
4. Choose C as the first splitting attribute and continue building the tree! How a tree of depth 2 would look like in that case?

A	B	C	Number of instances	
			class: +	class: –
T	T	T	5	0
F	T	T	0	20
T	F	T	20	0
F	F	T	0	5
T	T	F	0	0
F	T	F	25	0
T	F	F	0	0
F	F	F	0	25

Binary classification using confidence scores

- Until now we emphasized that the classifier predicts the label (0/1)
- In most cases we can supplement the prediction with a probability score (confidence score)
  - For *k*NN algorithm: it is the ratio of observations with label 1 among the *k* nearest neighbor of the data point
    - If the we use the weighted version, then the weights are taken into consideration as well
  - For decision tree: the ratio of observations with label 1 among the observations in the leaf of the data point
- Usually we predict label 1 if the score is above 0.5
  - The threshold can be set other values, it depends if we value Type 1 error / Type 2 error more

ROC

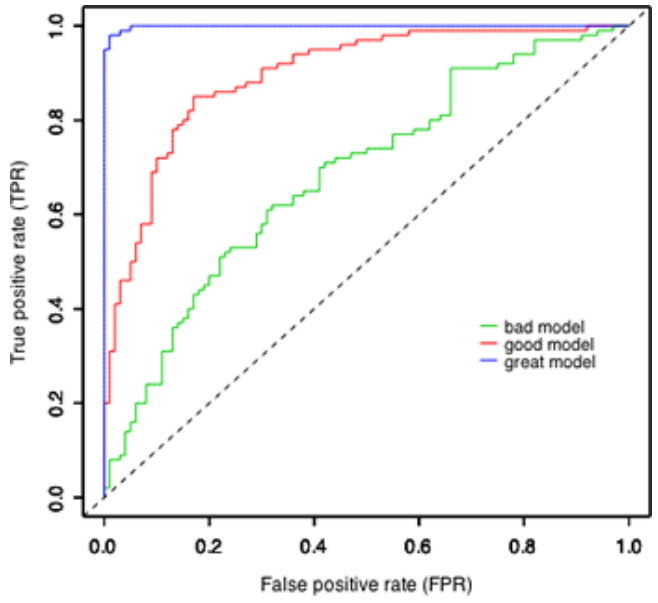
- **R**eceiver **O**perating **C**haracteristic
  - It has its origins in signal processing (1950s)
- Sort the test data according to their probability scores
  - Above a certain threshold we predict 1 (below that we predict 0)
- For all possible threshold values we determine the FPR and TPR rates
- ROC curve: plotting TPR against FRP at various threshold settings

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

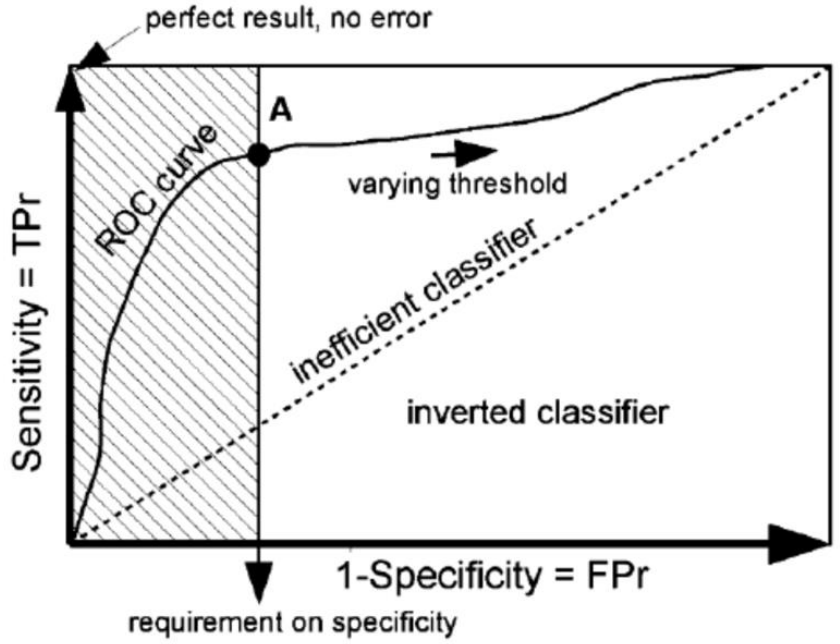
$$FPR = \frac{FP}{TN + FP}$$

ROC curve

- (1,1): every instance is classified as positive
- (0,0): every instance is classified as negative
- (0,1): perfect classifier

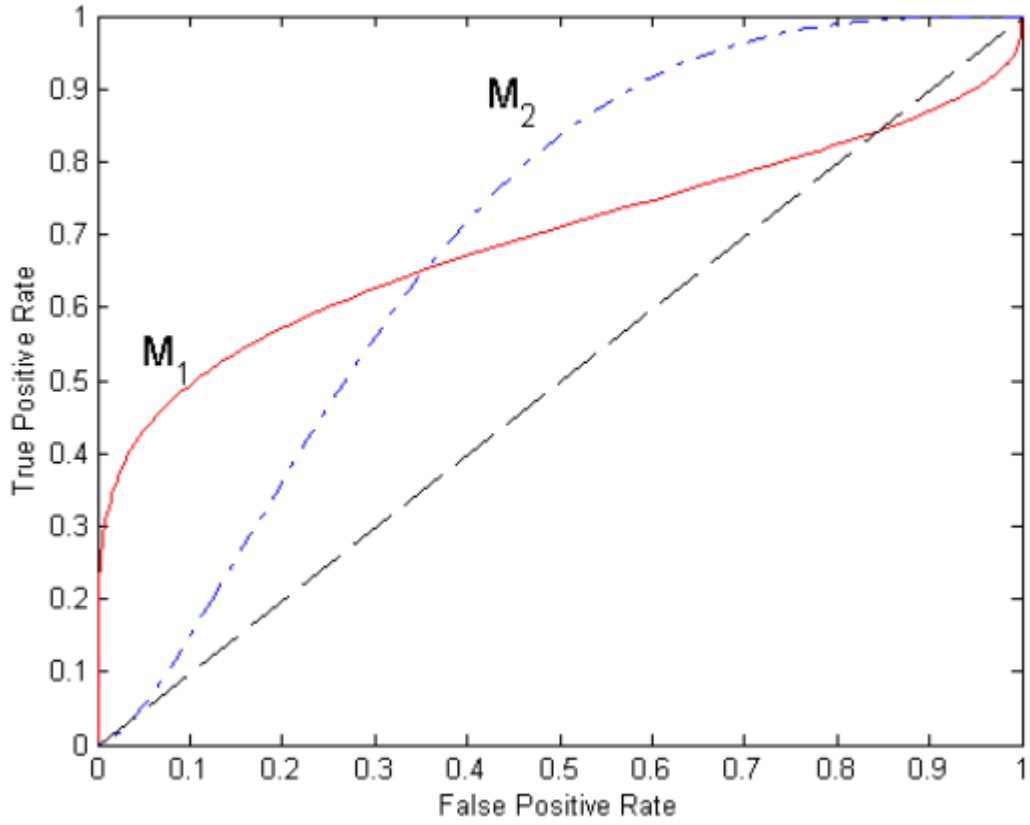


- Diagonal line (y=x): random classifier
  - Below the diagonal: even worse than random classifier
  - The predicted labels should be inverted



ROC for comparing models

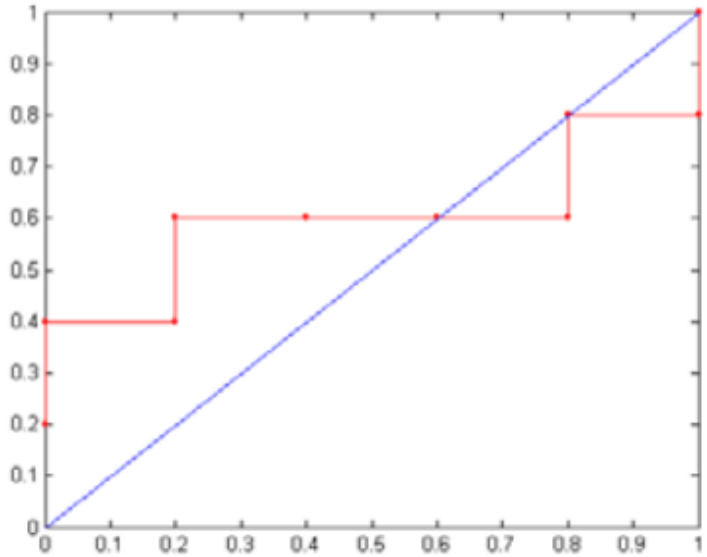
- None of the models can be considered consistently better than the other
  - M_1 is better for small FPR values
  - M_2 is better for large FPR values



Designing the ROC curve

	Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=		0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
	TP	5	4	4	3	3	3	3	2	2	1	0
	FP	5	5	4	4	3	2	1	1	0	0	0
	TN	0	0	1	1	2	3	4	4	5	5	5
	FN	0	1	1	2	2	2	2	3	3	4	5
→	TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
→	FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

ROC Curve:

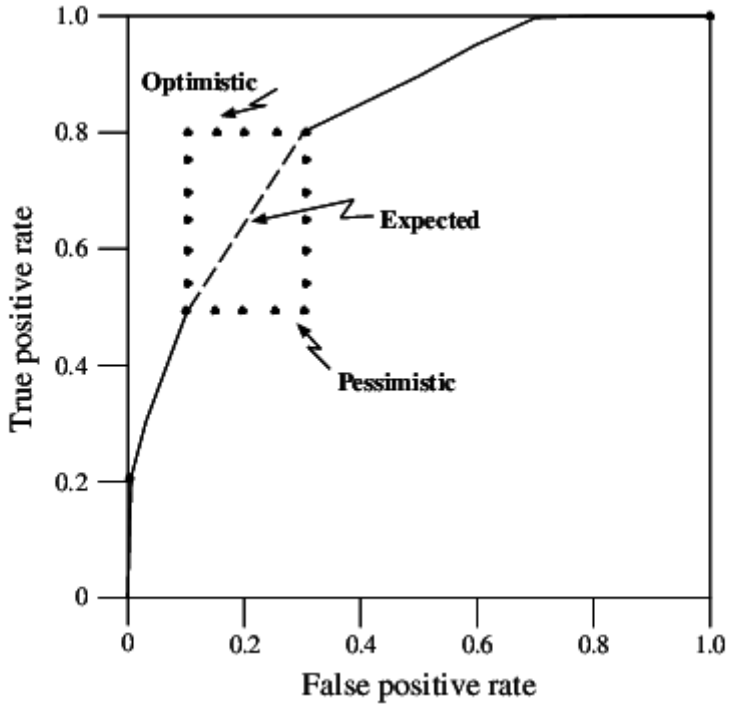


$$TPR = \frac{TP}{TP + FN}$$

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

ROC – equal probability scores with different labels

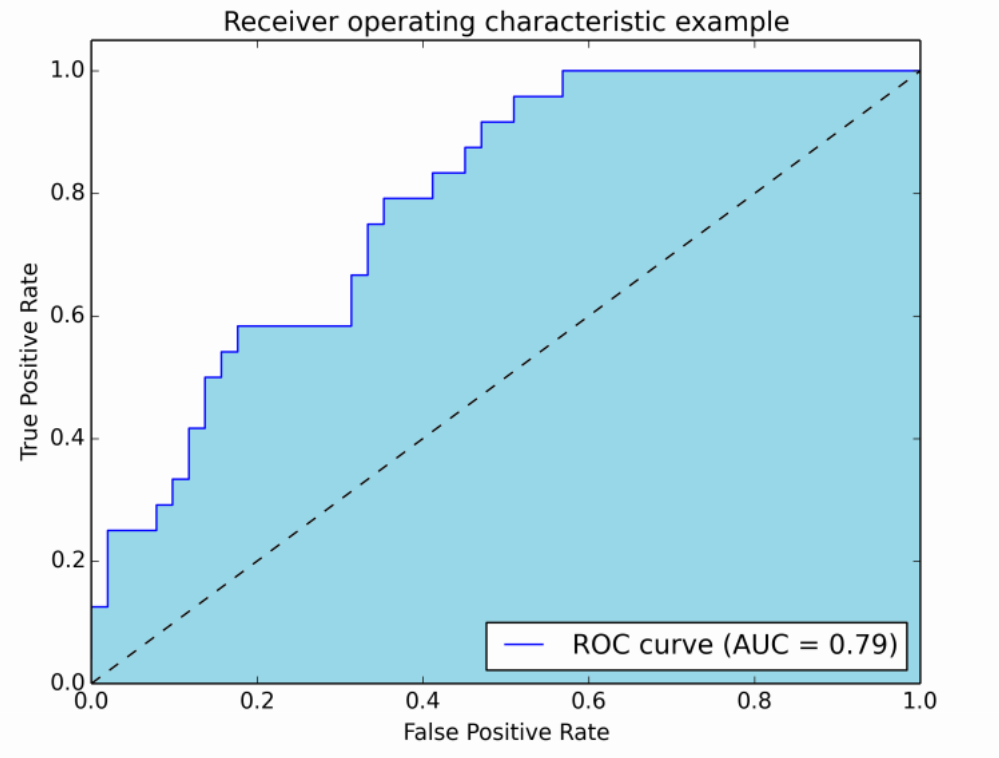
- It may happen that records with different labels have equal probability scores
  - In this case the ROC curve has diagonal segments

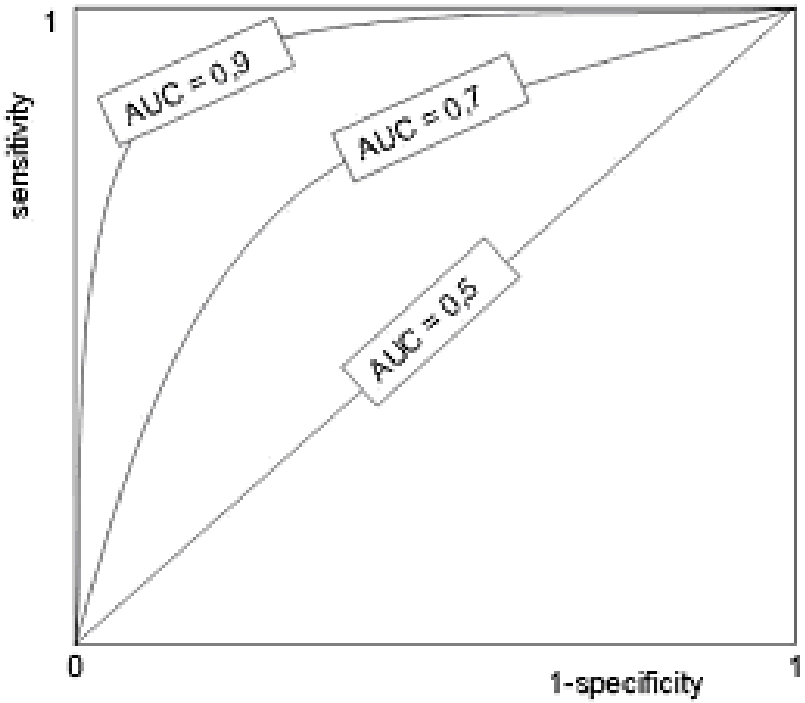


AUC

- **A**rea **U**nder ROC **C**urve
  - Its maximal value: 1 (perfect classifier)
  - For random classifier: 0.5
- It is the probability that a randomly-chosen positive record is ranked more highly than a randomly chosen negative record
- It is only defined for binary classification
- It is independent from the threshold – more stable performance indicator

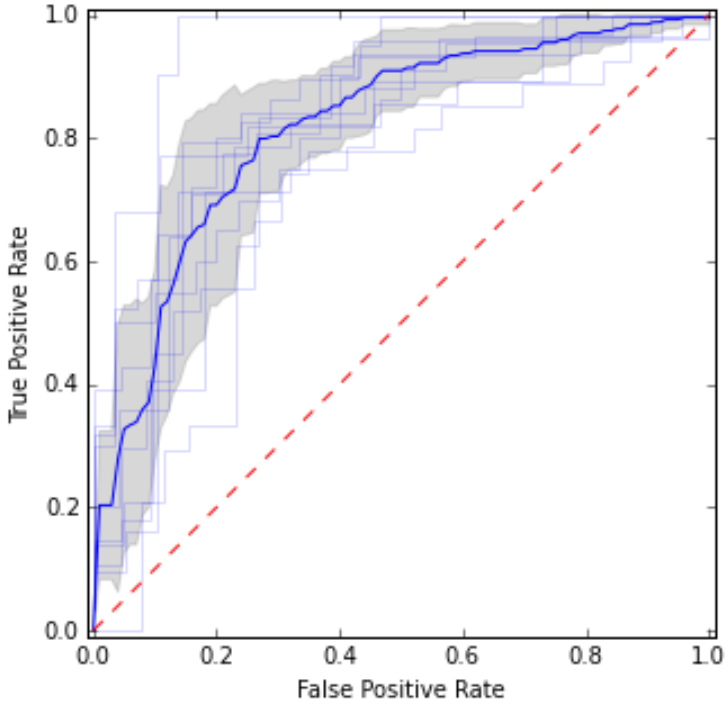
ROC, AUC

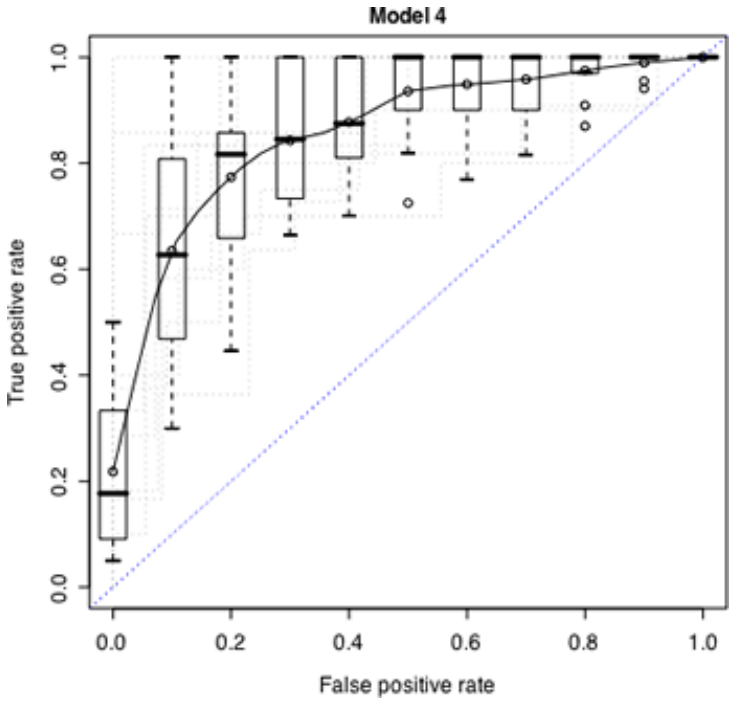

Receiver operating characteristic example
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.79)


1
sensitivity
AUC = 0.9
AUC = 0.7
AUC = 0.5
0
1-specificity
1

Cross-validation with ROC, AUC

- If we cross-validate the data we have more validation sets, so ROC can be determined each time
  - Mean values and variation can be calculated





ROC for multi-class classification

- There are various generalizations of the ROC concept for multi-class classification
  - No canonical approach
  - We omit this here

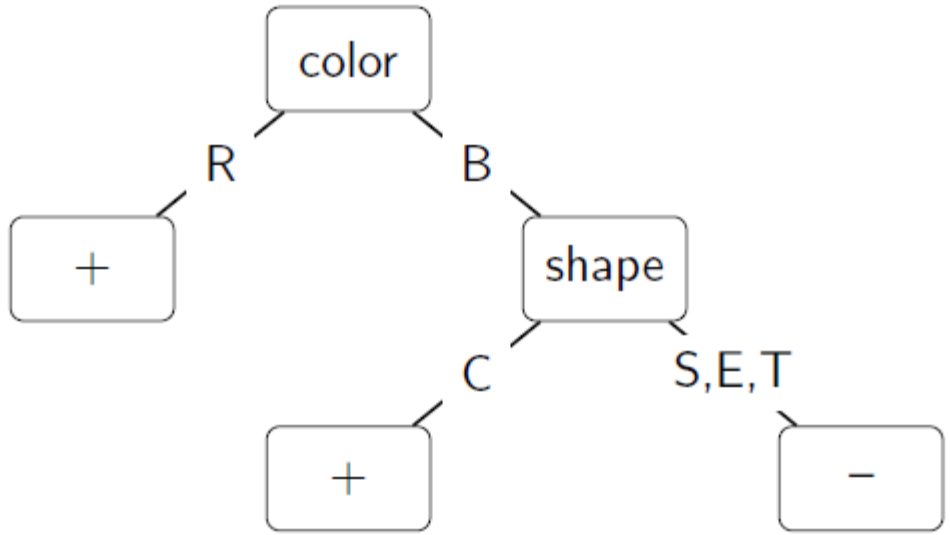
Problem

Construct the ROC curve of the following classifier and calculate the AUC. How do you interpret the AUC score? What would you suggest in terms of results? In the table confidence scores increase from left to right.

Label:	+	+	−	+	−	−	+	+	−	+	
TP											
FN											
TN											
FP											
TPR											
FPR											

You can see a schematic diagram of a possible decision tree built on training data below.

1. Determine the confidence scores (ratio of positive observations) of the leaves based on the training data (train1, train2, ..., train7).
2. Sort the confidence scores of first three test instances (test1, test2, test3) in ascending order.
3. Construct an ROC curve using the first three instances of the test data (test1, test2, test3) and calculate the AUC.
4. Construct the ROC curve after adding two new test data (test4, test5). If more instances have the same confidence scores ROC curve may change diagonally!



ID	Shape	Color	Size	Class
train1	S	R	L	+
train2	C	R	H	+
train3	C	B	H	+
train4	T	R	L	+
train5	S	B	M	−
train6	E	B	L	−
train7	C	R	M	−

ID	Shape	Color	Size	Class
test1	C	R	H	+
test2	C	B	L	−
test3	E	B	H	−
test4	C	R	L	+
test5	E	R	H	−

Data leakage

- The use of information in the model training process which would not be expected to be available at prediction time ➔ predictive metrics overestimate the model's utility when run in a production environment
  - Feature leakage
    - The inclusion of columns that will not be available when the model is used for predictions, and result in leakage if included when the model is trained
      - E.g.: "MonthlySalary" column when predicting "YearlySalary"; or "MinutesLate" when predicting "IsLate"; or more subtly "NumOfLatePayments" when predicting "ShouldGiveLoan"
  - Premature featuerization (e.g feature scaling parameters calculation before train-test split)
  - Duplicate rows between train/validation/test (e.g. oversampling before train-test split)
  - Group leakage
    - E.g. 3 X-ray images per patient, all images of a patient should be in the same split, therefore the model cannot memorize the patients instead of learning to recognize covid-19
  - Time leakage
    - Splitting a time-series dataset randomly instead of newer data in test set

Acknowledgement

- András Benczúr, Róbert Pálovics, SZTAKI-AIT, DM1-2
- Krisztián Buza, MTA-BME, VISZJV68
- Bálint Daróczy, SZTAKI-BME, VISZAMA01
- Judit Csima, BME, VISZM185
- Gábor Horváth, Péter Antal, BME, VIMMD294, VIMIA313
- Lukács András, ELTE, MM1C1AB6E
- Tim Kraska, Brown University, CS195
- Dan Potter, Carsten Binnig, Eli Upfal, Brown University, CS1951A
- Erik Sudderth, Brown University, CS142
- Joe Blitzstein, Hanspeter Pfister, Verena Kaynig-Fittkau, Harvard University, CS109
- Rajan Patel, Stanford University, STAT202
- Andrew Ng, John Duchi, Stanford University, CS229