



# Lecture: Machine Learning for Data Science

Winter semester 2021/22

## Lectures 6: Classification (Evaluation)

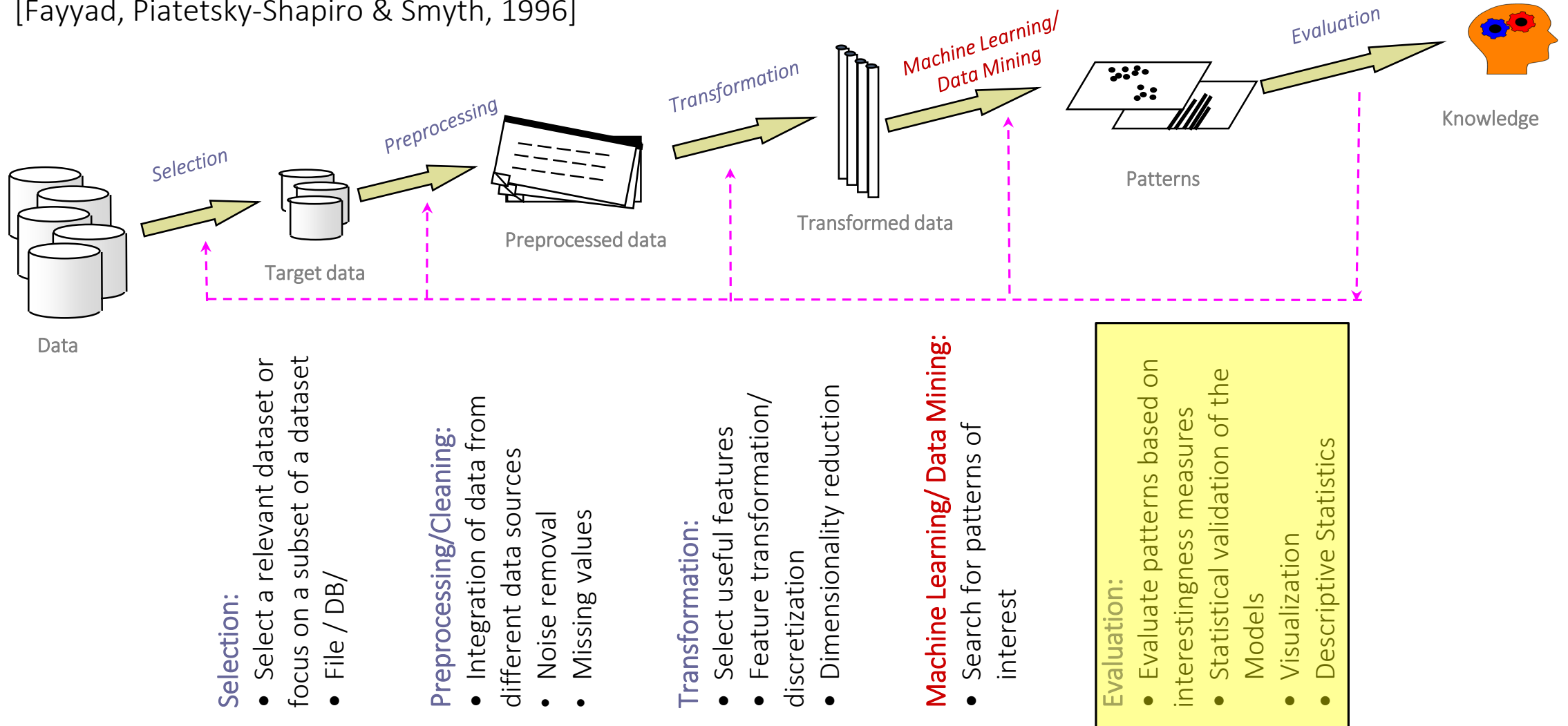
Prof. Dr. Eirini Ntoutsi

# Outline

- Evaluation of classifiers
- Evaluation measures for predictive performance
- Evaluation setup
- Beyond predictive performance evaluation
- Things you should know from this lecture & reading material

# The KDD process

[Fayyad, Piatetsky-Shapiro & Smyth, 1996]



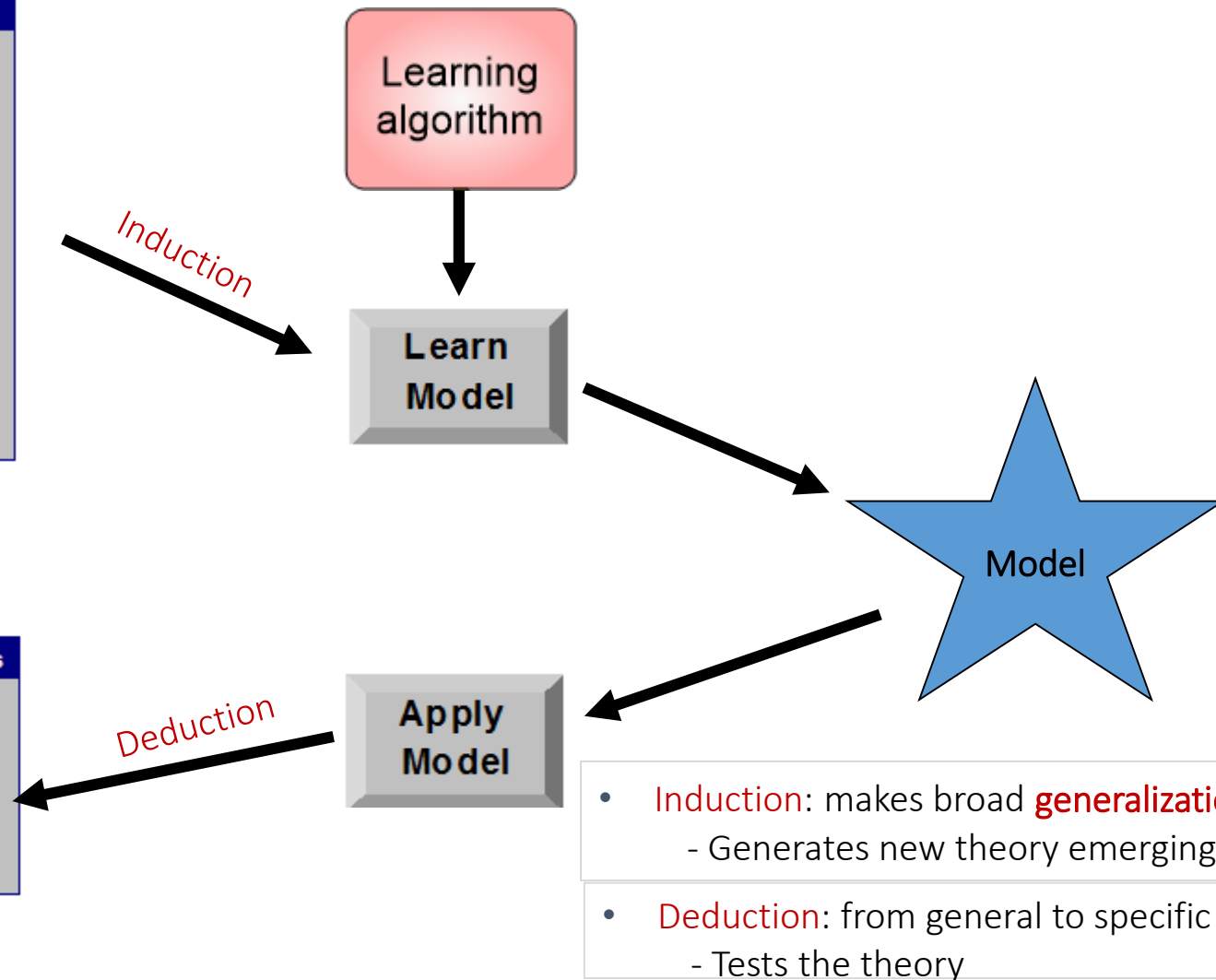
# General approach for building a classifier

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training data

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Future unseen instances



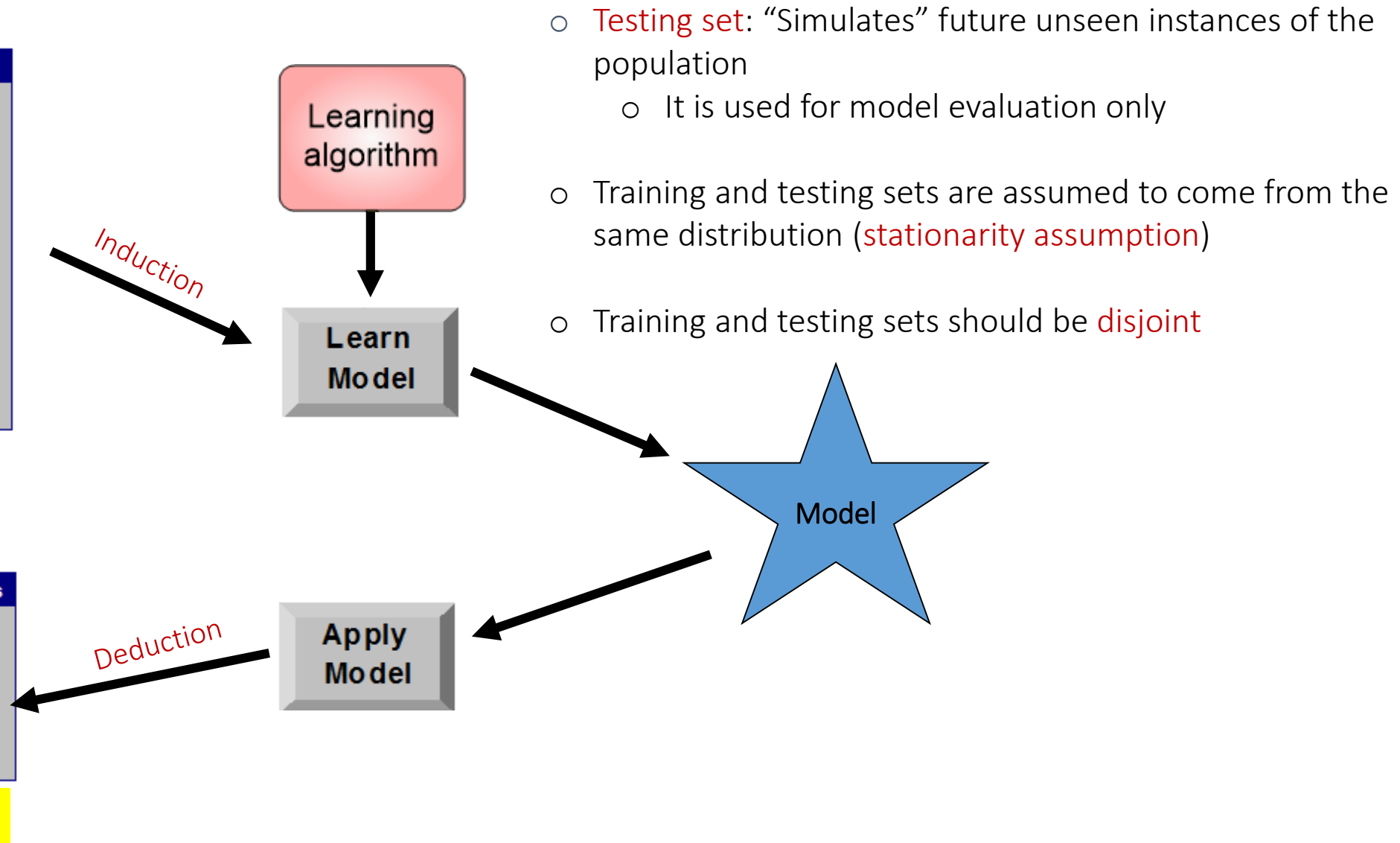
# General approach for building a classifier

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training data

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test set

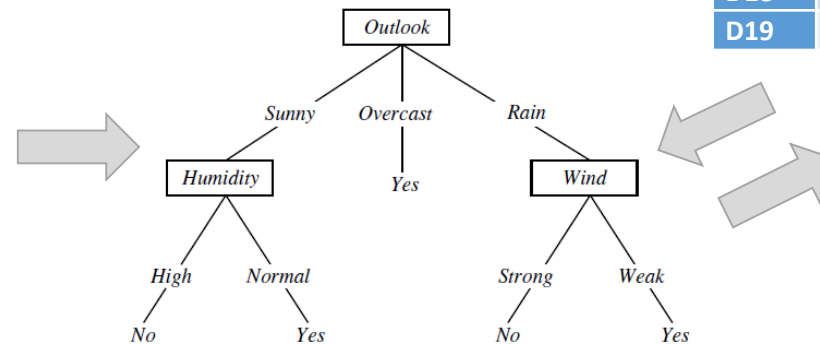


# Evaluation of classifiers

- The quality of a classifier is evaluated over the **test set**
  - For each instance in the test set, we know its true class label
  - We compare the predicted class (by some classifier) with the true class of the test instances

**Training set**

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



**Test set**

Day	Outlook	Temperature	Humidity	Wind	<i>true class label</i> Play Tennis	Prediction
D16	Overcast	Cool	Normal	Weak	Yes	Yes
D17	Overcast	High	Normal	Weak	Yes	No
D18	Sunny	Hot	Normal	Weak	No	No
D19	Overcast	Cool	Normal	Weak	No	Yes

*predicted class labels*

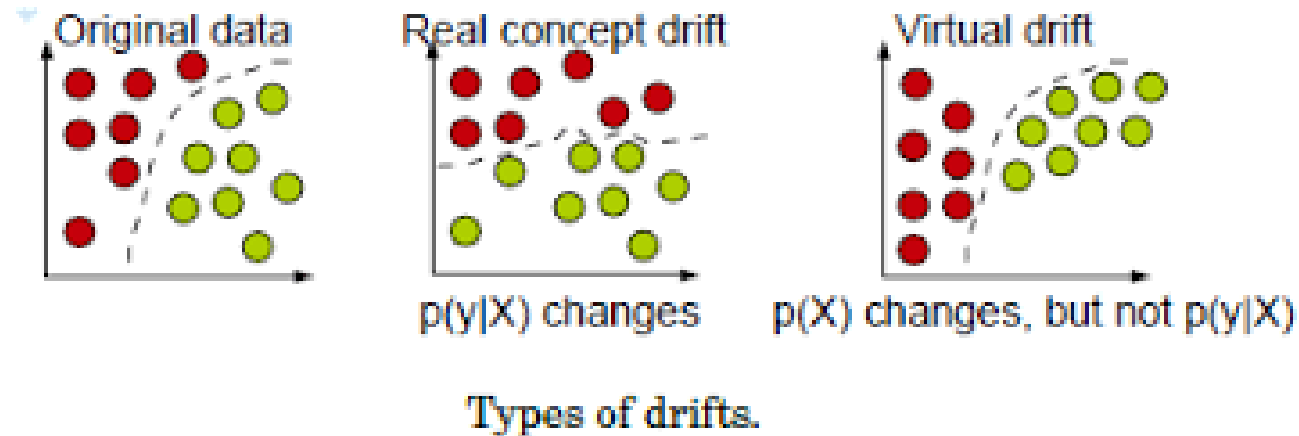
# Fundamental assumptions on the test set

- Training and testing sets come from the same distribution



Can you think of an application where this assumption is violated?

- The **data stationarity assumption**



- Testing instances are not seen during training

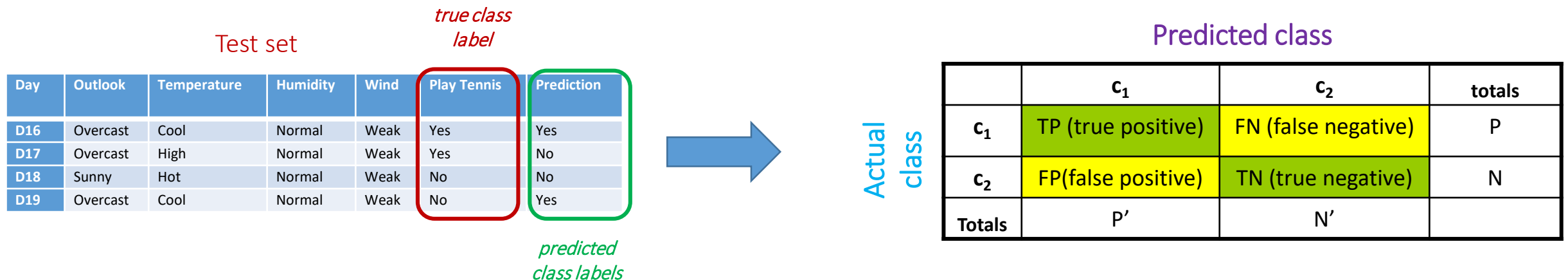
- the training and testing sets are **disjoint**



What if they are not disjoint? How this would affect evaluation?

# Confusion matrix

- A useful tool for analyzing how well a classifier performs is the confusion matrix
- The confusion matrix is built from the test set and the predictions of the classifier
- For an  $m$ -class problem, the matrix is of size  $m \times m$
- Terminology
  - **Positive tuples**: tuples of the main class of interest
  - **Negative tuples**: all other tuples
- An example for a binary classification problem  $Y=\{c_1, c_2\}$ :





# Outline

- Evaluation of classifiers
- Evaluation measures for predictive performance
- Evaluation setup
- Beyond predictive performance evaluation
- Things you should know from this lecture & reading material

# Classifier evaluation measures

- Accuracy
- Error rate
- Sensitivity
- Specificity
- Precision
- Recall
- F-measure
- $F_\beta$ -measure
- ...

# Classifier evaluation measures


		Predicted class		
		c <sub>1</sub>	c <sub>2</sub>	totals
Actual class\	c <sub>1</sub>	TP (true positive)	FN (false negative)	P
	c <sub>2</sub>	FP(false positive)	TN (true negative)	N
	Totals	P'	N'	

- **Accuracy/ Recognition rate**: % of test set instances correctly classified

$$accuracy(M) = \frac{TP + TN}{P + N}$$

- **Error rate/ Missclassification rate**: % of test set instances misclassified ( $1 - accuracy(M)$ )

$$error\_rate(M) = \frac{FP + FN}{P + N}$$

- Example: What is the accuracy and error rate of the following model? 

		Predicted class		
		classes	buy_computer = yes	buy_computer = no
Actual class	buy_computer = yes	6954	46	7000
	buy_computer = no	412	2588	3000
	total	7366	2634	10000

Accuracy(M)=95.42%

Error\_rate(M)=4.58%

- These measures are more effective when the class distribution is relatively balanced

# Limitations of accuracy and error rate

- Consider a binary classification problem  $Y=\{c_1, c_2\}$
- Consider a test set of 10000 instances
  - 9990 instances belong to class  $c_1$
  - 10 instances belong to class  $c_2$
- Assuming a model  $M$  that predicts everything to be of class  $c_1$
- What is the accuracy of this model?
  - $\text{Accuracy}(M) = 9990/10000 = 99.9 \%$
  - Is this a good model?
  - No!
- Accuracy is misleading because the model does not detect any  $c_2$  instances.
- !!! Accuracy and error rate are more effective when the **class distribution is relatively balanced**

# Classifier evaluation measures

		Predicted class		
		c <sub>1</sub>	c <sub>2</sub>	totals
Actual class	c <sub>1</sub>	TP (true positive)	FN (false negative)	P
	c <sub>2</sub>	FP(false positive)	TN (true negative)	N
	Totals	P'	N'	

- If classes are **imbalanced**:
- **Sensitivity/ True positive rate/ recall**: % of positive tuples that are correctly classified

$$TPR(M) = sensitivity(M) = recall(M) = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- **Specificity/ True negative rate** : % of negative tuples that are correctly classified

$$TNR(M) = specificity(M) = \frac{TN}{N} = \frac{TN}{TN + FP}$$

- Example: What is the sensitivity and specificity of the following model? **?**

		Predicted class		
Actual class	classes	buy_computer = yes	buy_computer = no	total
	buy_computer = yes	6954	46	7000
	buy_computer = no	412	2588	3000
	total	7366	2634	10000

→ Accuracy(M)=95.42%

→ sensitivity(M)=99.34%

→ specificity(M)=86.27%

# Classifier evaluation measures

		Predicted class		
		c <sub>1</sub>	c <sub>2</sub>	totals
Actual class\	c <sub>1</sub>	TP (true positive)	FN (false negative)	P
	c <sub>2</sub>	FP(false positive)	TN (true negative)	N
	Totals	P'	N'	

Lets take a closer look to the positive class

- **Sensitivity/ True positive rate/ recall** : % of positive tuples classified as positive

$$recall(M) = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- **Precision**: % of tuples classified as positive which are actually positive

$$precision(M) = \frac{TP}{P'} = \frac{TP}{TP + FP}$$

- Recall vs precision

- Precision biased towards TP and FP ; Recall biased towards TP and FN
- Higher precision → lower FP; Higher recall → lower FN

- Example: What is the precision and recall of the following model?



		Predicted class			
		buy_computer = yes	buy_computer = no	total	Accuracy (%)
Actual class	buy_computer = yes	6954	46	7000	99.34
	buy_computer = no	412	2588	3000	86.27
	total	7366	2634	10000	95.42

→ recall(M)=99.34%

→ precision(M)=94.41%

# Classifier evaluation measures

		Predicted class		
Actual class\		c <sub>1</sub>	c <sub>2</sub>	totals
	c <sub>1</sub>	TP (true positive)	FN (false negative)	P
	c <sub>2</sub>	FP(false positive)	TN (true negative)	N
	Totals	P'	N'	

- F-measure/ F1 score/F-score combines both

$$F(M) = \frac{2 * precision(M) * recall(M)}{precision(M) + recall(M)}$$

It is the harmonic mean of precision and recall

- For our example,  $F(M) = 2 * 94.41\% * 99.34\% / (94.41\% + 99.34\%) = 96.81\%$

- $F_{\beta}$ -measure is a weighted measure of precision and recall

$$F_{\beta}(M) = \frac{(1 + \beta^2) * precision(M) * recall(M)}{\beta^2 * precision(M) + recall(M)}$$

- Common values for  $\beta$ :

- $\beta=1 \rightarrow F_1$

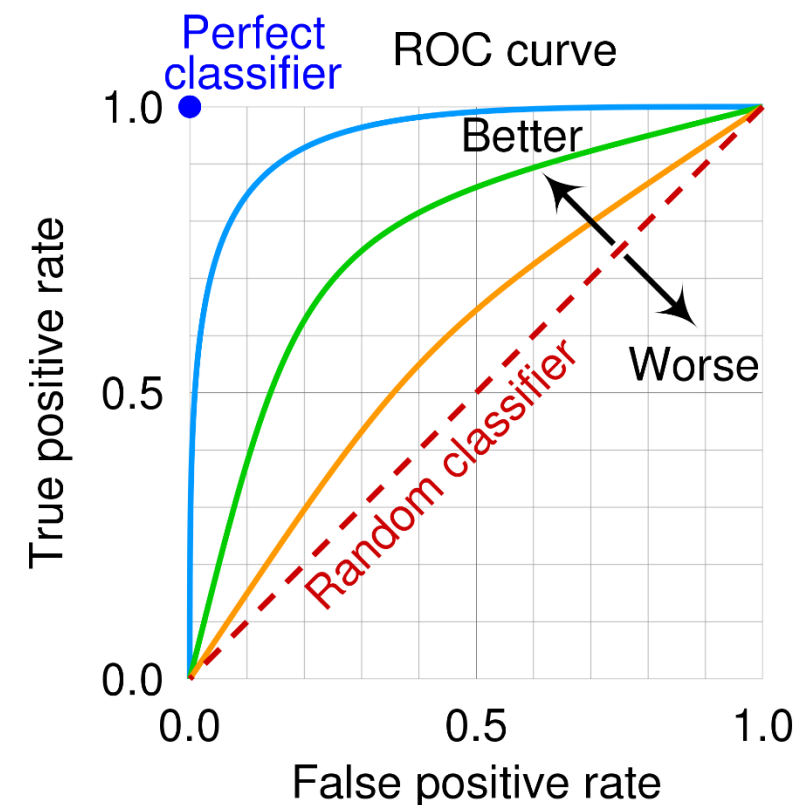
- $\beta=0.5$

More on harmonic mean:  
<http://mathworld.wolfram.com/HarmonicMean.html>

# Classifier evaluation measures: ROC curve

Actual class \ Predicted class	Predicted class		totals
	c <sub>1</sub>	c <sub>2</sub>	
	c <sub>1</sub>	c <sub>2</sub>	
c <sub>1</sub>	TP (true positive)	FN (false negative)	P
c <sub>2</sub>	FP (false positive)	TN (true negative)	N
Totals	P'	N'	

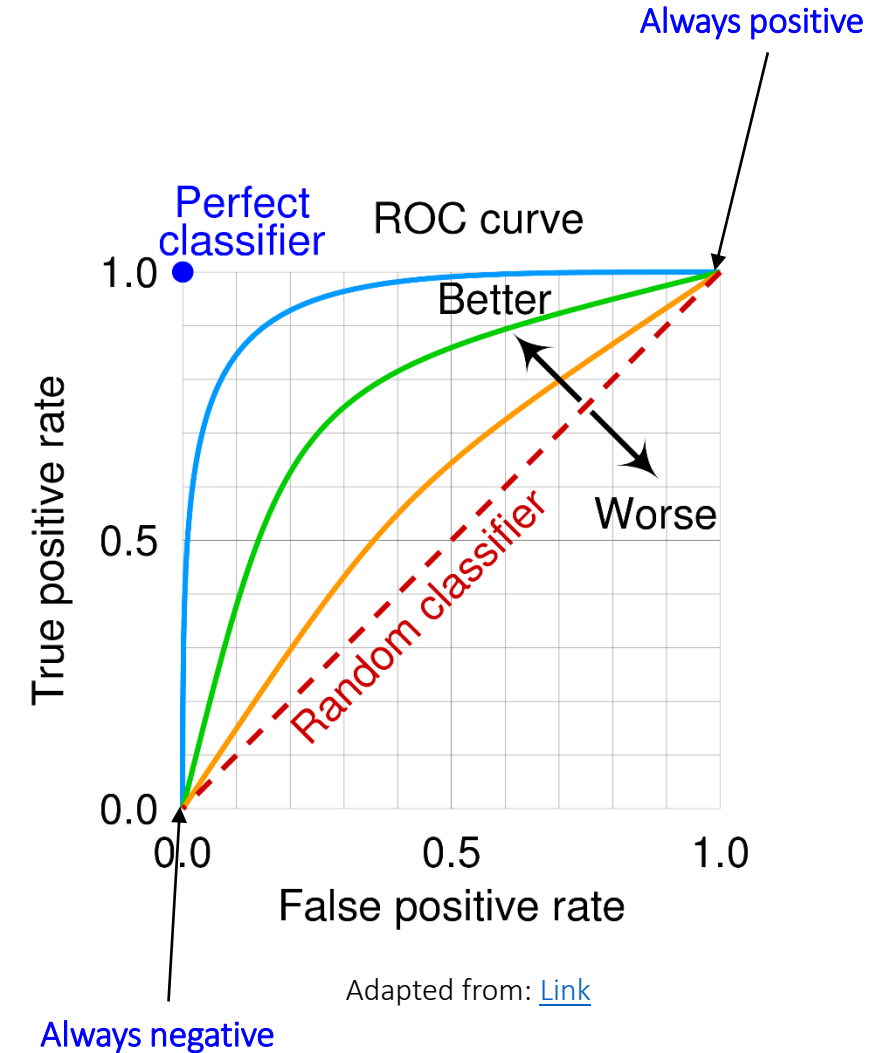
- ROC: Receiver Operating Characteristic curve
  - a graphical plot for inspecting the performance of a binary classifier
  - It compares the rates at which the classifier is making correct decisions (true positives) with making wrong decisions (false positives)
- Used for evaluating classifier's performance and for comparing the relative performance among different classifiers.
- On the Y axis the **TPR (True Positive Rate)** is depicted
  - TPR: % of positive tuples that are correctly recognized (as positive)
    - $TPR(M) = sensitivity(M) = recall(M) = \frac{TP}{P} = \frac{TP}{TP+FN}$
- On the X axis the **FPR (False Positive Rate)** is depicted
  - FPR: % of negative tuples that are incorrectly classified
    - $FPR(M) = \frac{FP}{N} = \frac{FP}{FP+TN} = 1 - specificity(M)$





# Classifier evaluation measures: ROC curve

- **Perfect classification:**
  - $\text{TPR} = 1.0$  (no false negatives)
  - $\text{FPR} = 0.0$  (no false positives)
- **Always positive classification:**
  - $\text{TPR} = 1.0$
  - $\text{FPR} = 1.0$
- **Always negative classification:**
  - $\text{TPR} = 0.0$
  - $\text{FPR} = 0.0$
- **Random classification** (diagonal = line of no-discrimination):
  - $\text{TPR} = 0.5$
  - $\text{FPR} = 0.5$
- **In practice:**
  - Points above the diagonal represent good classification results (better than random)
  - Points below the line represent poor results (worse than random)
- The steepness of the curve is also important since ideally we want to maximize TPR while minimizing FPR



# An example: 3 classifiers

- ROC curve is used to evaluate the performance of different classifiers

True	Predicted	
	pos	neg
pos	40	60
neg	30	70

## Classifier 1

TP = 0.4

FP = 0.3

True	Predicted	
	pos	neg
pos	70	30
neg	50	50

## Classifier 2

TP = 0.7

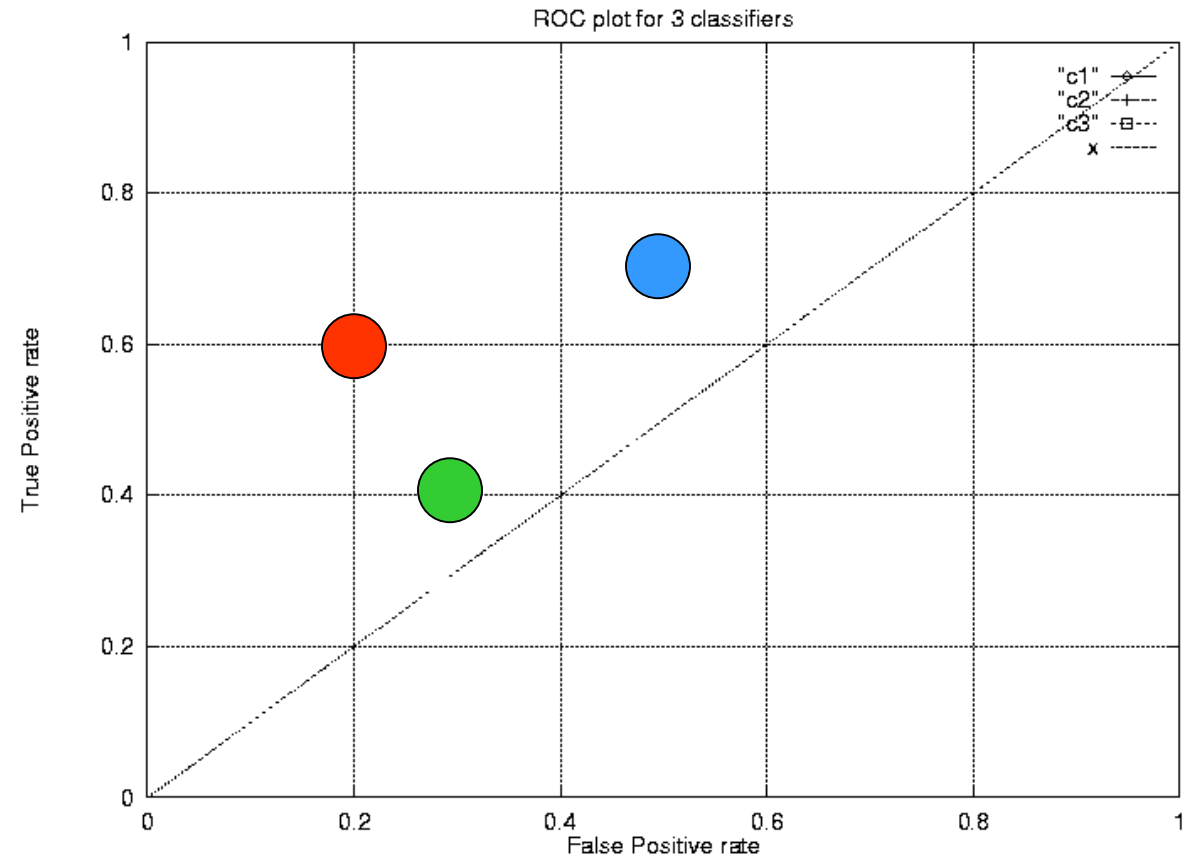
FP = 0.5

True	Predicted	
	pos	neg
pos	60	40
neg	20	80

## Classifier 3

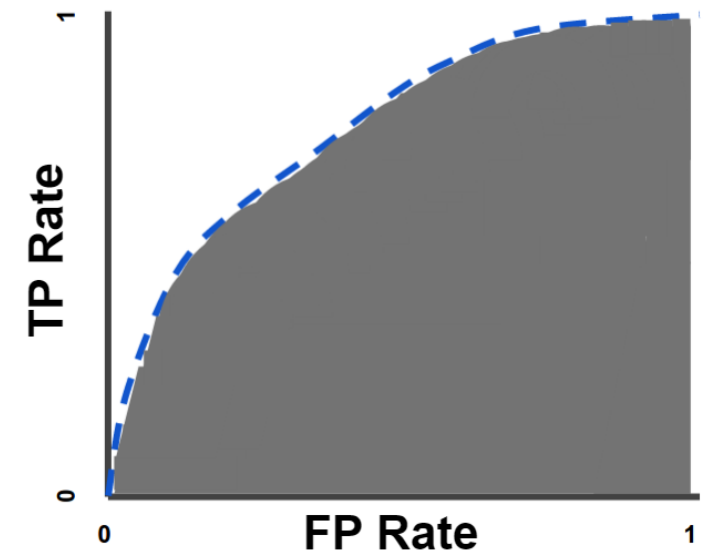
TP = 0.6

FP = 0.2



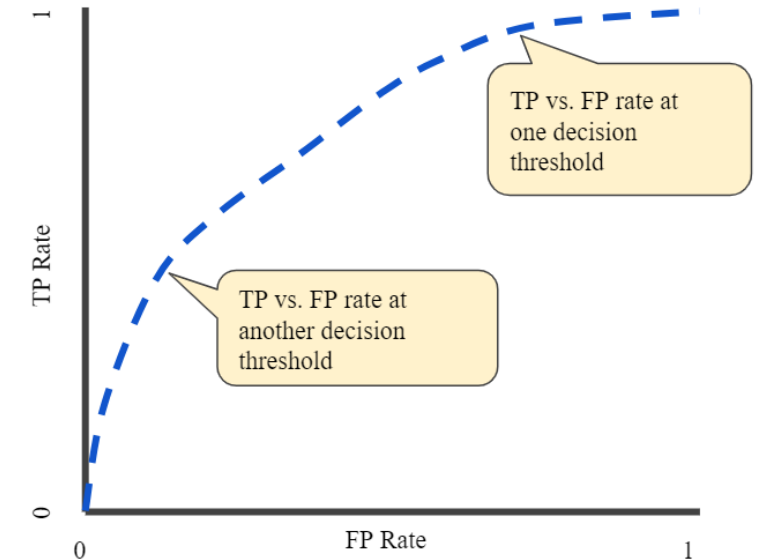
# Area under the curve (AUC)

- Area Under the Curve (AUC)
  - measures the entire two-dimensional area underneath the entire ROC curve
  - Provides an aggregated measure of the performance
- The higher the AUC the better the performance of the model



# Evaluation of classifiers: ROC curve

- How to create a ROC curve?
- Classifiers like a decision trees produces a single confusion matrix → 1 ROC point.
- Classifiers like neural networks and Naïve Bayes naturally yield an instance probability or score a numeric value that represents the degree to which an instance is a member of a class.
- Such a scoring classifier can be used with a threshold to produce a discrete (binary) classifier: if the classifier output is above the threshold, the classifier outputs class Yes, else class No.
- Each threshold value then produces a different confusion matrix and corresponds to a different point in the ROC space.
- If the classifier has a “sensitivity” parameter, varying it produces a series of ROC points (different confusion matrices).
- Alternatively, a series of ROC points can be generated by varying the class ratio in the training set.



# Outline

- Evaluation of classifiers
- Evaluation measures for predictive performance
- Evaluation setup
- Beyond predictive performance evaluation
- Things you should know from this lecture & reading material

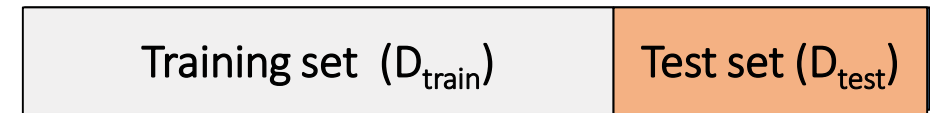
# Evaluation setup

- How to create the training and test sets out of a dataset?
  - We don't want to make unreasonable assumptions about our population
- Many approaches
  - Holdout
  - Cross-validation
  - Bootstrap
  - ....

# Evaluation setup

- 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
- (+) It takes no longer to compute
- (-) it depends on how data are divided
- (-) You have less data for training, problematic for small datasets.

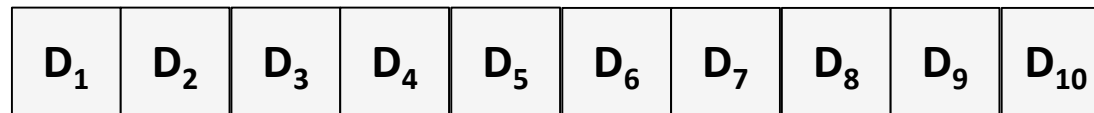


- Random sampling: a variation of holdout

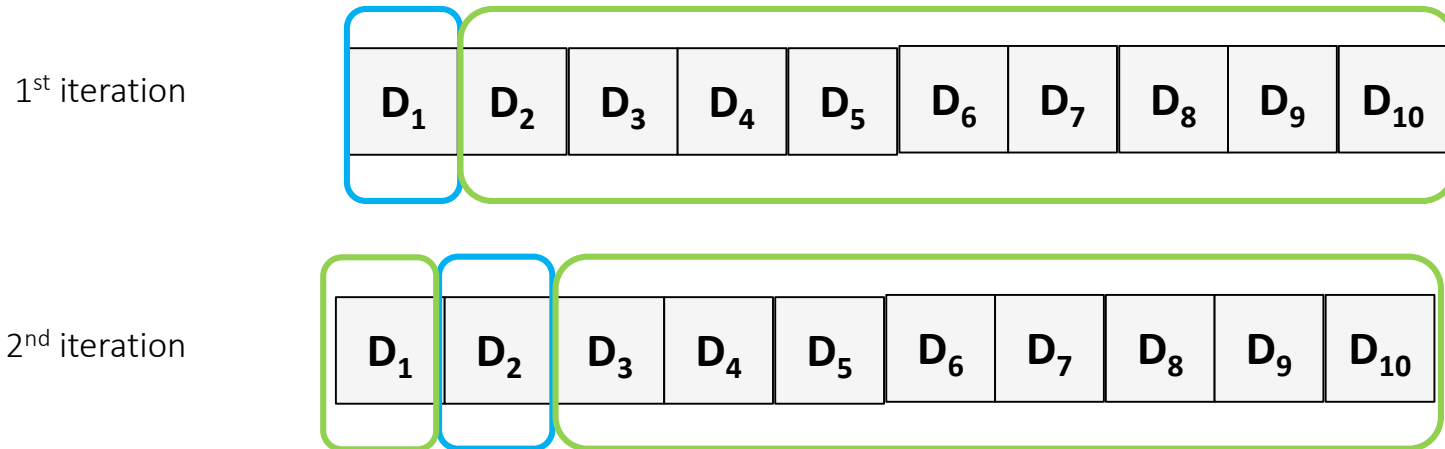
- Repeat holdout  $k$  times, accuracy is the avg accuracy obtained

# Evaluation setup

- **Cross-validation** ( $k$ -fold cross validation,  $k = 10$  usually)
  - Randomly partition the data into  $k$  mutually exclusive subsets  $D_1, \dots, D_k$  each approximately equal size



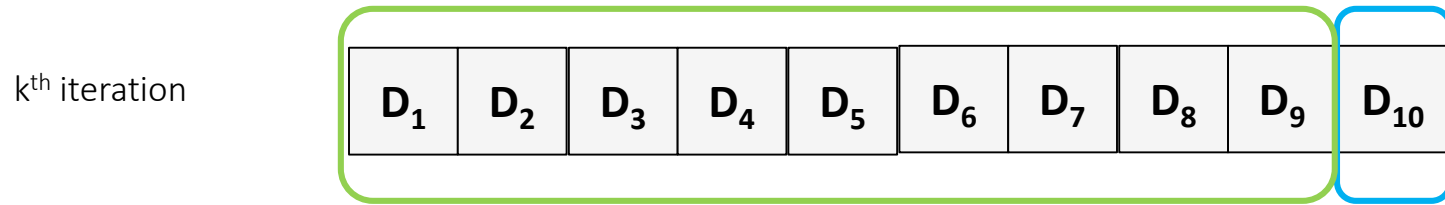
- Training and testing is performed  $k$  times
  - At the  $i$ -th iteration, use  $D_i$  as test set and rest as training set





# Evaluation setup

- **Cross-validation** ( $k$ -fold cross validation,  $k = 10$  usually) cont'

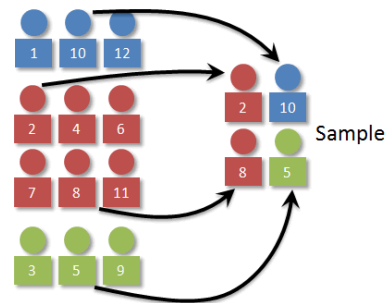


- Each point is in a test set 1 time and in a training set  $k-1$  times
  - Accuracy is the avg accuracy over all iterations
  - (+) Does not rely so much on how data are divided
  - (-) The algorithm should re-run from scratch  $k$  times
- **Leave-one-out:**  $k$ -folds with  $k = \text{\#of tuples}$ , so only one sample is used as a test set at a time;
    - for small sized data

# Evaluation setup

- **Stratified sampling** vs **random sampling**

- Stratified sampling creates a mini-reproduction of the population in terms of the class labels. E.g., if 25% of the population belongs to the class “blue”, 25% to class “green” and 50% to class “red” then 25% of the sample is drawn randomly from class “blue”, 25% from class “green” and 50% from class “red”.



Source: [link](#)

- **Stratified cross-validation**: folds are stratified so that class distribution in each fold is approximately the same as that in the initial data
  - **Stratified 10 fold cross-validation** is recommended!!!

# Evaluation setup

- **Bootstrap**: Samples the given training data uniformly with replacement
  - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
  - Works well with small data sets
- Several bootstrap methods, and a common one is **.632 bootstrap**
  - Suppose we are given a dataset of  $n$  tuples. The data set is sampled  $n$  times, with replacement, resulting in a training set of  $n$  samples (known also as **bootstrap sample**)
  - The data tuples that did not make it into the training set end up forming the test set.
  - Each sample has a probability  $1/n$  of being selected and  $(1-1/n)$  of not being chosen.
  - We repeat  $n$  times, so the probability for a tuple to not be chosen during the whole period is  $(1-1/n)^n$ .
    - For large  $n$ :
$$\left(1 - \frac{1}{n}\right)^n \approx e^{-1} \approx 0.368$$
  - So on average, 36.8% of the tuples will not be selected for training and thereby end up in the test set; the remaining 63.2% will form the train test

# Evaluation setup

- Repeat the sampling procedure  $k$  times  $\rightarrow k$  bootstrap datasets
- What is the overall accuracy of the model?
  - Note that in each bootstrap round, the training and testing sets might overlap (due to sampling with replacement), so the accuracy on the bootstrap test set might be upward biased
  - To correct for this problem, Efron proposed the 0.632 estimate.

$$acc_{boot}(M) = \frac{1}{k} \sum_{i=1}^k (0.632 \times acc(M_i)_{testSet_i} + 0.368 \times acc(M_i)_D)$$

Accuracy of the model obtained by bootstrap sample  $i$  when it is applied on test set  $i$ .

Accuracy of the model obtained by bootstrap sample  $i$  when it is applied over all training data  $D$  (training error)

- Therefore this part contributes with a 0.632 weight (Recall that the average number of distinct instances in each bootstrap round is 63.2%)

# The role of the validation set

- If **model selection** and **error estimates** are to be computed simultaneously, the data should be divided into three **disjoint** datasets

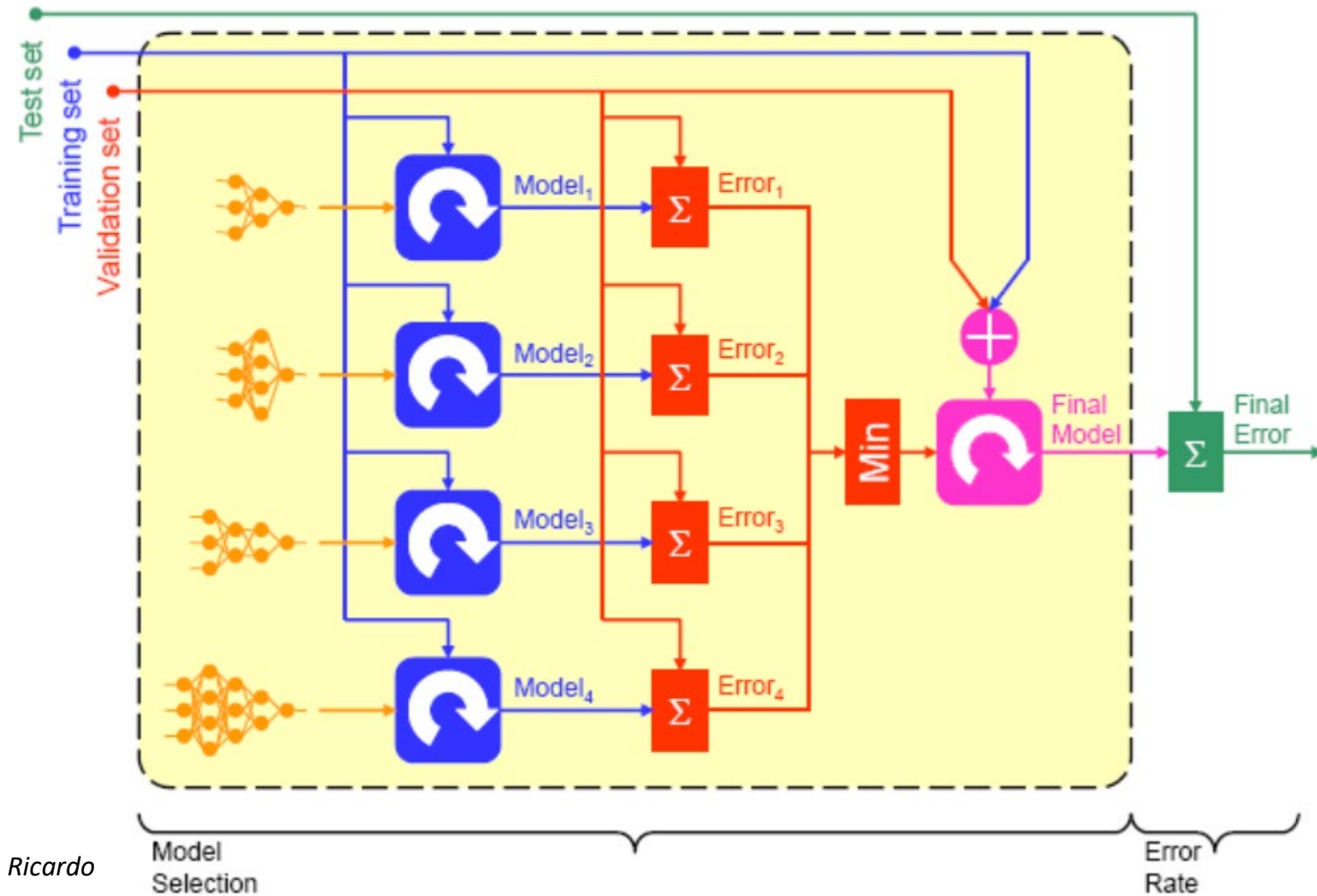


- **Training set** is a set of examples used for learning a model.
  - **Validation set** is a set of examples that can help to tune model parameters (e.g., selecting  $k$  in kNN).
  - **Test set** is used only to assess the performance of the final model and provide an estimation of the generalization error.
- 
- Why separate test and validation sets?
    - The error estimate of the final model on validation data will be biased (smaller than the true error rate) since the validation set is used to select the final model.
    - After assessing the final model on the test set, you **MUST NOT** tune the model any further.

# The role of the validation set

- Procedure outline (for holdout)
  1. Divide the available data into training, validation and test set.
  2. Select parameters
  3. Train the model using the training set
  4. Evaluate the model using the validation set
  5. Repeat steps 2 through 4 using different parameters
  6. Select the best parameters and train the final model using data from both the training and validation sets
  7. Assess this final model using the test set
- If cross validation or bootstrap are used, steps 3,4 have to be repeated for each of the  $k$  folds.

# Train, Validation and Test in practice



Source: CSCE 666 Pattern Analysis | Ricardo Gutierrez-Osuna | CSE@TAMU

# Classifier evaluation summary

- Evaluation measures
  - accuracy, error rate, sensitivity, specificity, precision,  $F$ -score,  $F_\beta$ ..., ROC
- Train – test splitting
  - Holdout, cross-validation, bootstrap,...
- Validation set for parameter selection
- Other parameters that might influence our decision on which model is best for our problem
  - Speed (construction time, usage time)
  - Robustness to noise, outliers and missing values
  - Scalability for large data sets
  - Interpretability (by humans)
  - Fairness w.r.t. some protected attribute(s)
  - ...

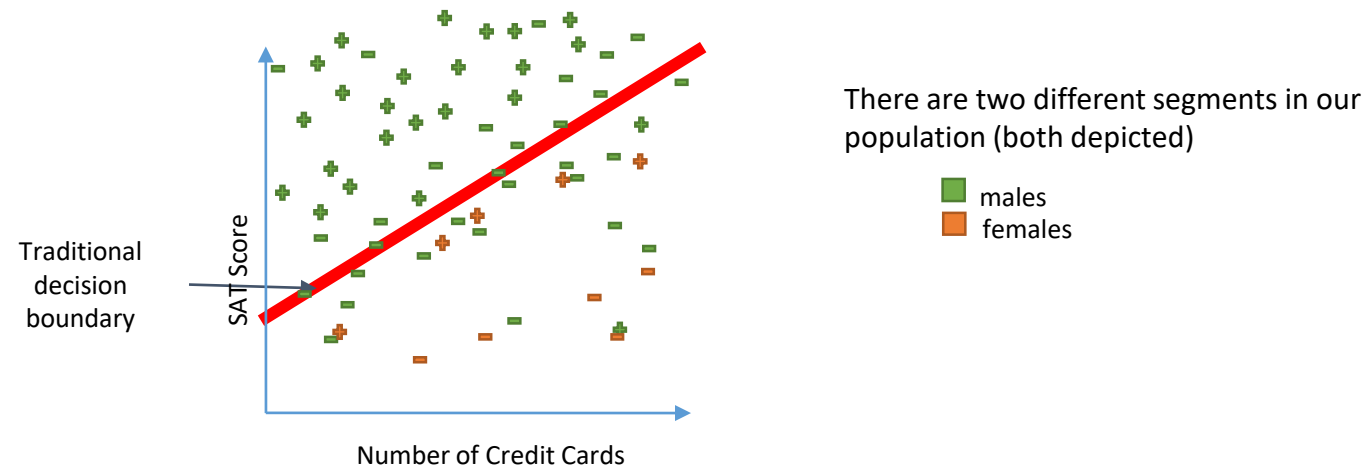


# Outline

- Evaluation of classifiers
- Evaluation measures for predictive performance
- Evaluation setup
- Beyond predictive performance evaluation
  - Things you should know from this lecture & reading material

# Fairness-aware evaluation

- Consider the following binary classification problem with classes:  $\{+, -\}$ . Consider also a binary protected attribute like gender {males, females}

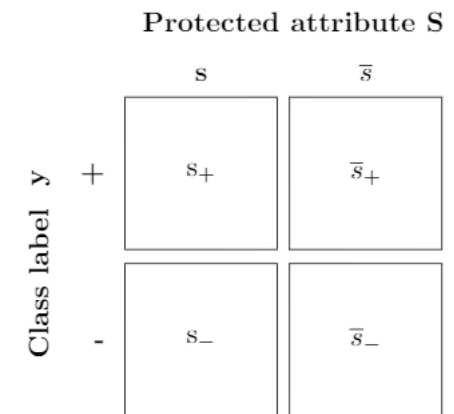


- The goal of a traditional classifier is to find the hypothesis (parameters of the line) that minimizes the empirical error.
  - This might incur discrimination (all female instances are rejected in our example)
  - How can we evaluate the fairness-related behavior of a model?

# Typical (batch) fairness-aware learning setup

- **Input:**  $D$  = training dataset
  - $F$ : set of non-protected attributes
  - $S$ : (typically: binary, single) protected attribute
    - $s$  ( $\bar{s}$ ): protected (non-protected) group
  - $y$  = (typically: binary) class attribute  $\{+, -\}$  (+ for accepted, - for rejected)

	<b>F1</b>	<b>F2</b>	<b>S</b>	<b>y</b>
<b>User<sub>1</sub></b>	$f_{11}$	$f_{12}$	$s$	+
<b>User<sub>2</sub></b>	$f_{21}$			-
<b>User<sub>3</sub></b>	$f_{31}$	$f_{23}$	$s$	+
...	...	...	...	...
<b>User<sub>n</sub></b>	$f_{n1}$			+



# Measuring (un)fairness: some measures

	F1	F2	S	y	$\hat{y}$
User <sub>1</sub>	f <sub>11</sub>	f <sub>12</sub>	S	+	-
User <sub>2</sub>	f <sub>21</sub>			-	+
...	...	...	...	...	...
User <sub>n</sub>	f <sub>n1</sub>			+	+

- **Statistical parity**: If subjects in both protected and unprotected groups should have equal probability of being assigned to the positive class

$$P(\hat{y} = + | S = s) = P(\hat{y} = + | S = \bar{s})$$

- **Equal opportunity**: There should be no difference in model's prediction errors regarding the positive class

$$P(\hat{y} \neq y | S = s_+) = P(\hat{y} \neq y | S = \bar{s}_+)$$

- **Disparate Mistreatment**: There should be no difference in model's prediction errors between protected and non-protected groups for both classes

$$\delta FNR = P(\hat{y} \neq y | S = s_+) - P(\hat{y} \neq y | S = \bar{s}_+)$$

$$\delta FPR = P(\hat{y} \neq y | S = s_-) - P(\hat{y} \neq y | S = \bar{s}_-)$$

$$\text{Disparate Mistreatment} = |\delta FNR| + |\delta FPR|$$

# Outline

- Evaluation of classifiers
- Evaluation measures for predictive performance
- Evaluation setup
- Beyond predictive performance evaluation
- Things you should know from this lecture & reading material

# Overview and Reading

- Overview

- Evaluation measures
- Evaluation setup
- Beyond predictive performance measures

- Reading

- Chapter 11: Model selection and validation, Understanding Machine Learning book by Shai Shalev-Schwartz and Shai Ben-David
- Chapter 22, Classifier evaluation, Zaki and Meyra book
- An introduction to ROC analysis , Tom Fawcet [link](#)

# Hands on experience



- Consider some imbalanced dataset, for example:
  - [Bank dataset](#)
    - Bank dataset is related to direct marketing campaigns of a Portuguese banking institution. The task is to determine if a person subscribes to the product (bank term deposit). As positive class we consider people who subscribed to a term deposit. Class imbalance ratio, positive: negative ratio = 11%:89%
  - [KDD census income dataset](#)
    - It contains demographic data from the U.S. and the task is to predict whether the annual income of a person will exceed 50K dollars. Severe class imbalance: positive to negative ratio of 1:15 (exact ratio 6%:94%).
- The classifiers we discussed thus far assumed the population is balanced
- How would you change them to take into account the imbalance of the dataset?
- Evaluate the performance of different interventions using proper evaluation measures (obviously, accuracy is not a good indicator)

# Acknowledgements

- The slides are based on
  - ❑ KDD I lecture at LMU Munich (Johannes Aßfalg, Christian Böhm, Karsten Borgwardt, Martin Ester, Eshref Januzaj, Karin Kailing, Peer Kröger, Eirini Ntoutsi, Jörg Sander, Matthias Schubert, Arthur Zimek, Andreas Züfle)
  - ❑ Introduction to Data Mining book slides at <http://www-users.cs.umn.edu/~kumar/dmbook/>
  - ❑ Pedro Domingos Machine Lecture course slides at the University of Washington
  - ❑ Machine Learning book by T. Mitchel slides at <http://www.cs.cmu.edu/~tom/mlbook-chapter-slides.html>
  - ❑ (DTs) J. Fürnkranz slides from TU Darmstadt (<https://www.ke.tu-darmstadt.de/lehre/archiv/ws0809/mlbm/>)
  - ❑ Thank you to all TAs contributing to their improvement, namely Vasileios Iosifidis, Damianos Melidis, Tai Le Quy, Han Tran.



Thank you

Questions/Feedback/Wishes?

# Acknowledgements

- The slides are based on
  - KDD I lecture at LMU Munich (Johannes Aßfalg, Christian Böhm, Karsten Borgwardt, Martin Ester, Eshref Januzaj, Karin Kailing, Peer Kröger, Eirini Ntoutsi, Jörg Sander, Matthias Schubert, Arthur Zimek, Andreas Züfle)
  - Introduction to Data Mining book slides at <http://www-users.cs.umn.edu/~kumar/dmbook/>
  - Pedro Domingos Machine Lecture course slides at the University of Washington
  - Machine Learning book by T. Mitchel slides at <http://www.cs.cmu.edu/~tom/mlbook-chapter-slides.html>
  - Arthur Zimek DMML lecture at SDU
  - Thank you to all TAs contributing to their improvement, namely Vasileios Iosifidis, Damianos Melidis, Tai Le Quy, Han Tran.