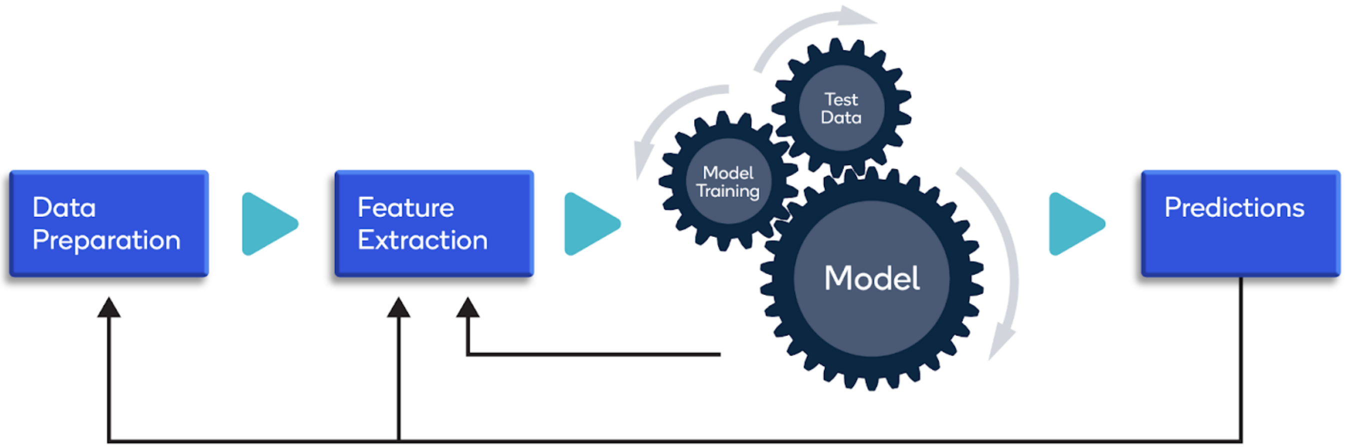
CCS2213: Machine Learning



Topic 2

Machine Learning – Process & Metric

Assoc. Prof. Dr Umi Kalsom Yusof
SCHOOL OF COMPUTING & INFORMATICS
AlBukhary International University(AIU)

Methodologies for Machine Learning

scikit learn machine learning in Python WEKA

Fayyad's Knowledge Discovery in Databases (KDD)

TERADATA Raising Intelligence

IBM SPSS

sas

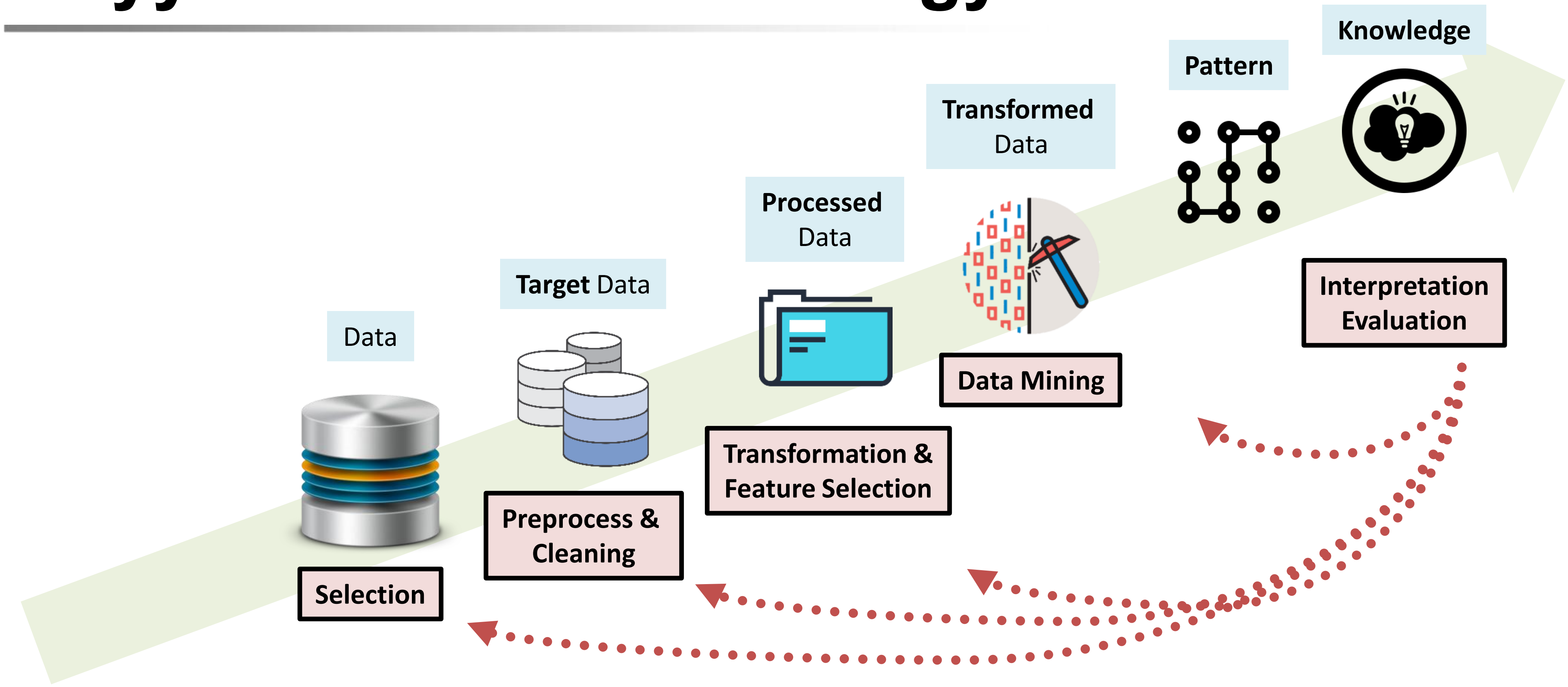
Data Mining Methodologies

rapidminer

Sample, Explore, Modify, Model, Assess (SEMMA)

CRoss-Industry Standard Process for Data Mining (CRISP-DM)

Fayyad’s KDD Methodology

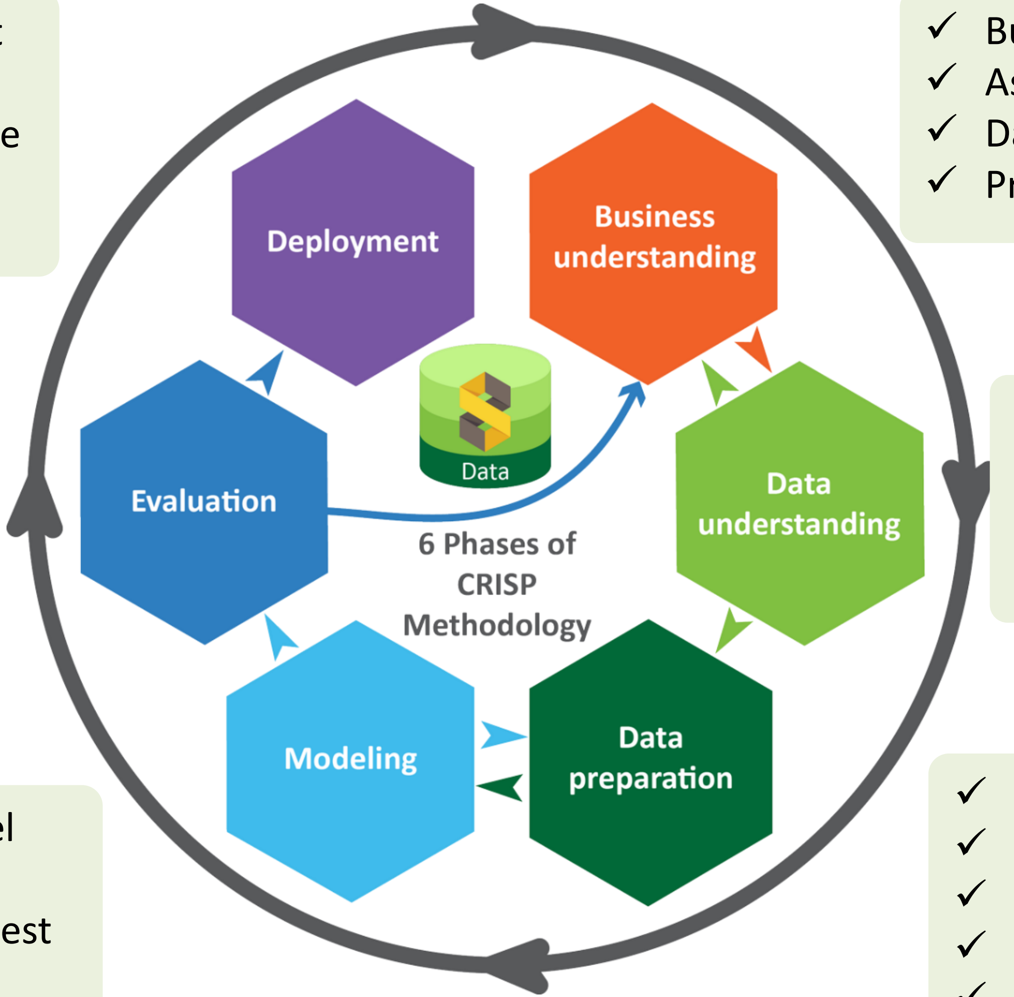


CRISP-DM Methodology

- ✓ **Plan** deployment
- ✓ **Plan** monitoring
- ✓ **Plan** maintenance
- ✓ **Final** report
- ✓ **Review** project

- ✓ Business **objective**
- ✓ Assess **situation**
- ✓ Data mining **goals**
- ✓ Project **plan**

- ✓ **Evaluate** results
- ✓ **Review** process
- ✓ **Determine** next steps



- ✓ **Collect** data
- ✓ **Describe** data
- ✓ **Explore** data
- ✓ **Verify** data quality

- ✓ **Select** model
- ✓ **Techniques**
- ✓ **Design** the test
- ✓ **Build** model
- ✓ **Assess** model

- ✓ **Select** data
- ✓ **Clean** data
- ✓ **Construct** data
- ✓ **Integrate** data
- ✓ **Format** data

SEMMA Methodology

SAMPLE
- Input Data
- Sampling
- Data Partition

EXPLORE
- Distributions
- Multiplot
- Insight
- Association
- Variable selection

MODIFY
- Transform variable
- Filter outliers
- Clustering
- SOM/ Kohonen

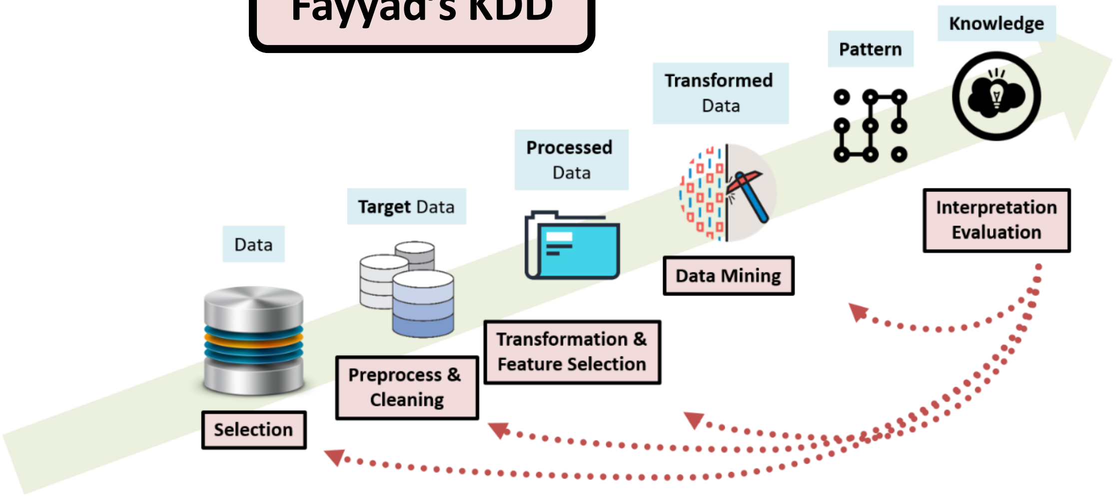
MODEL
- Regression
- Tree
- Neural Network
- Ensemble

ASSESS
- Assessment
- Score
- Report

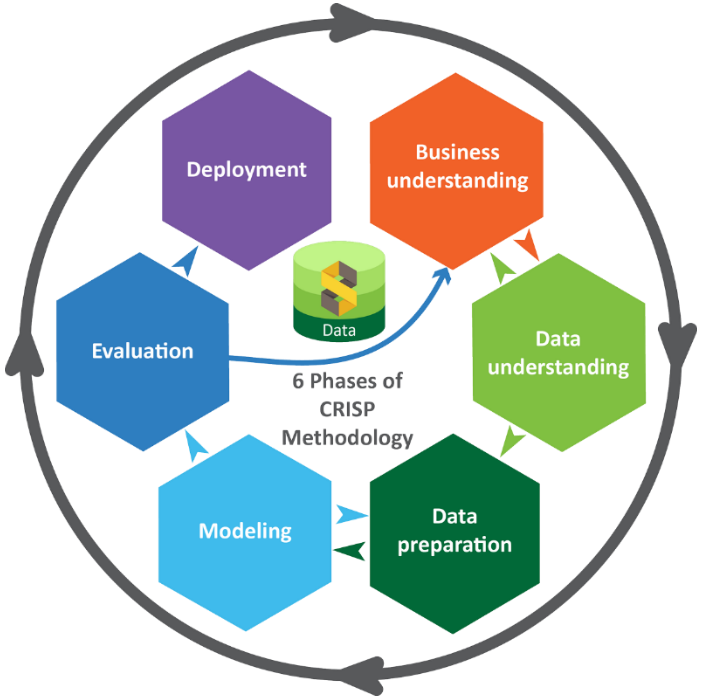
sas Enterprise Mining Environment

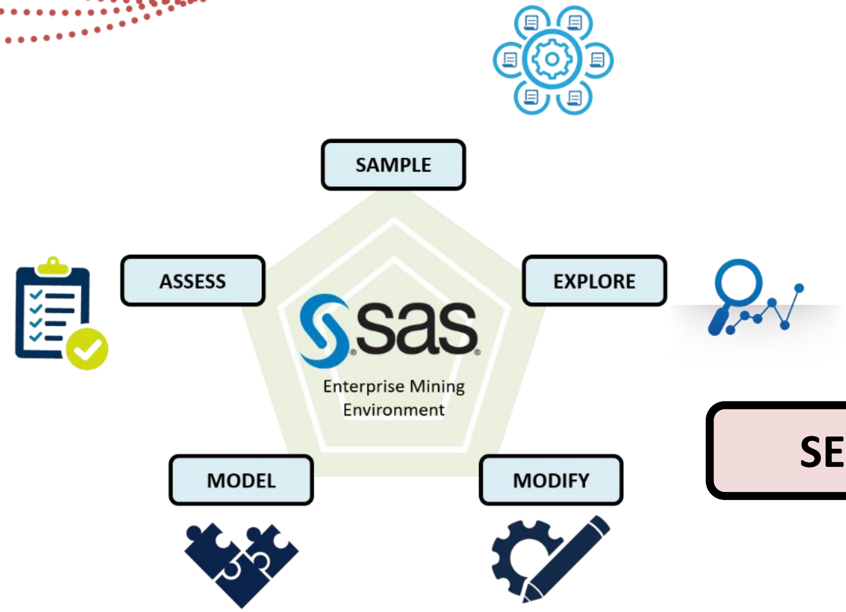
A Comparison

Fayyad’s KDD



CRISP-DM





SEMMA

Phases & Generic Tasks

Business Understanding → Data Understanding → Data Preparation → Modeling → Evaluation → Deployment

Business Understanding
- Determine Business Objectives
- Assess Situation
- Determine Data Mining Goals
- Produce Project Plan

Business Understanding
- ❑ Project **objectives**
- ❑ Requirements from a **business perspective**
- ❑ Convert **knowledge** into a **data mining problem definition**
- ❑ Design **plan** to **achieve objectives**

Data Understanding
- Collect Initial Data
- Describe Data
- Explore Data
- Verify Data Quality

Data Understanding
- ❑ Initial **data collection**
- ❑ Identify data **quality** problems
- ❑ Discover **insights**
- ❑ Detect **interesting subsets**
- ❑ Form **hypotheses** for **hidden** information.

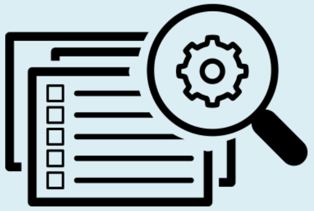
Motivation

It is a ***data partitioning strategy*** so that you can effectively use your dataset to build ***a more generalized model***. The main intention of doing any kind of machine learning is to develop a more generalized model which can perform well on ***unseen data***.

Important to evaluate classifier's generalization performance:

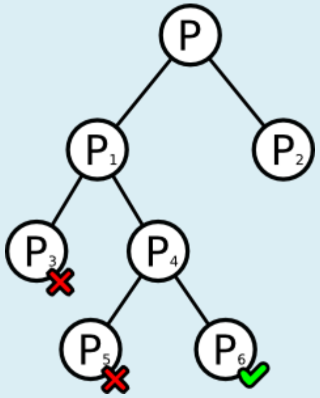
Determine whether to **employ** the **classifier**

Example:
Learning the **effectiveness** of medical treatments from a **limited-size data**, it is important to **estimate** the **accuracy** of the **classifiers**

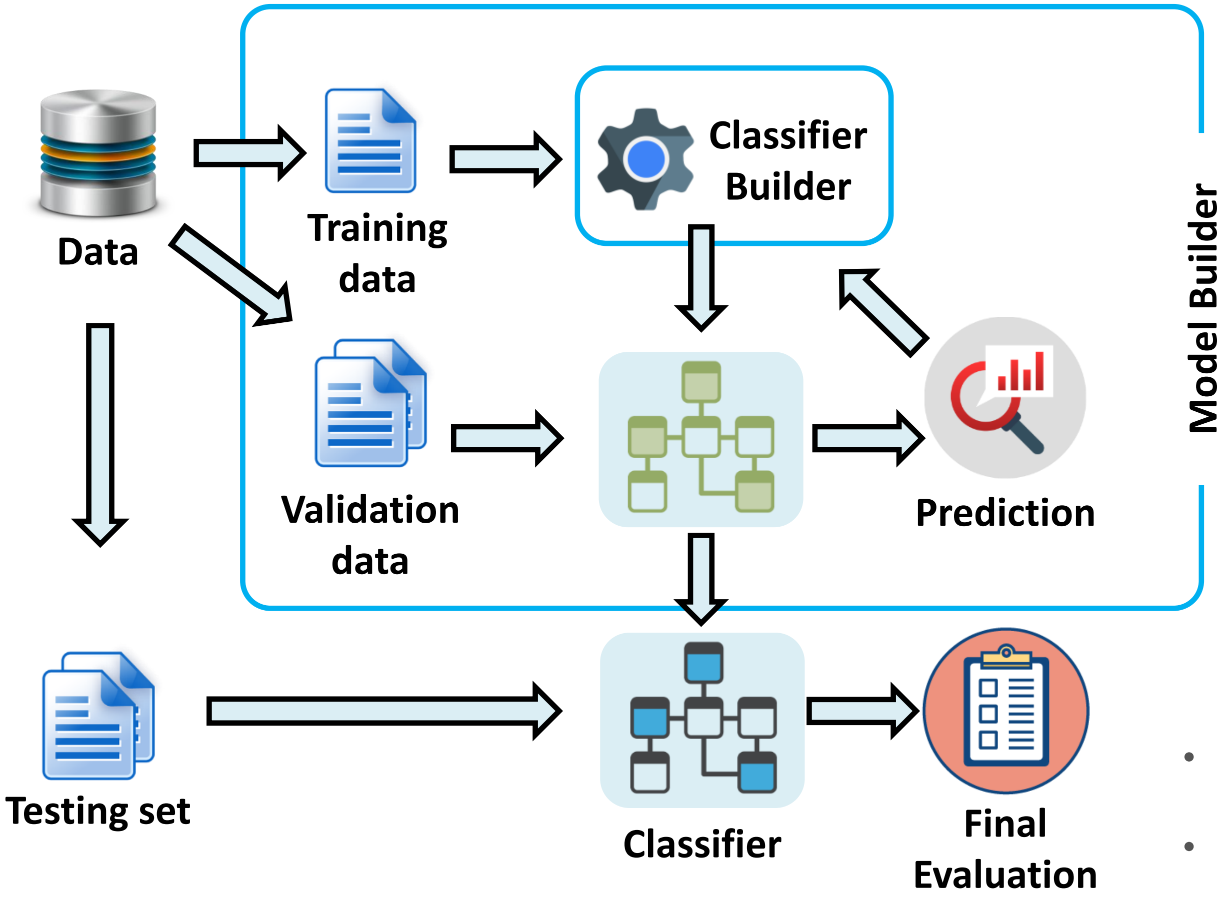


Optimize the **classifier**

Example:
when **post-pruning** decision trees we must **evaluate** the **accuracy** of the decision trees on each **pruning step**



Classification:
Train, Validation, Test Split



The test data can't be used for parameter tuning!

Making The Most of The Data

Once evaluation is **complete**, **all the data** can be used to **build** the **final classifier**.

The **larger** the **training data** the **better** the classifier (but returns diminish).

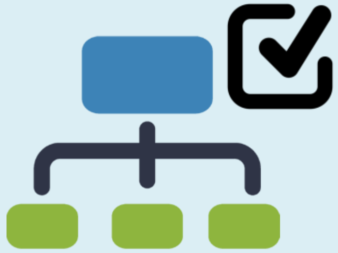
The **larger** the **test data** the more **accurate** the error estimate.

- ***Training set***: *A set of examples used for learning, that is to fit the parameters of the classifier.*
- ***Validation set***: *A set of examples used to tune the parameters of a classifier, for example to choose the number of hidden units in a neural network.*
- ***Test set:*** *A set of examples used only to assess the performance of a fully-specified classifier.*

For evaluating a model’s performance
and hyperparameter tuning

Model evaluation techniques:

- **Training** data;
- **Independent** test data;
- **Hold-out** method;
- **k-fold** cross-validation method;
- **Leave-one-out** method;
- **Bootstrap** method;
- and many more…



Model Performance
Evaluation Techniques

Training Data

Training Set → Classifier → Training Set

The accuracy/metric estimates on the training data are not good indicators of performance on future data

New data will probably not exactly the same as the training data!

This measure the degree of classifier's overfitting (or underfitting).

Independent Test Data

Training Set → Classifier → Test Set

- Used when we have plenty of data
- Natural way of forming training and test data

Example:

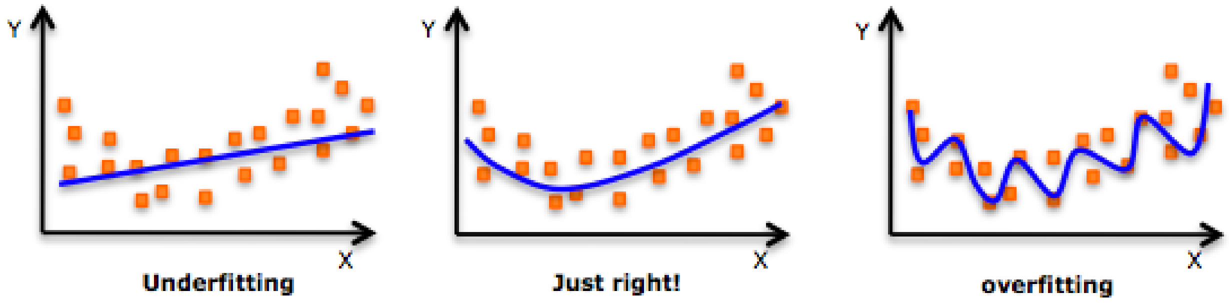
- Quinlan in 1987 reported experiments in a medical domain
- Trained on data from 1985
- Tested on data from 1986.

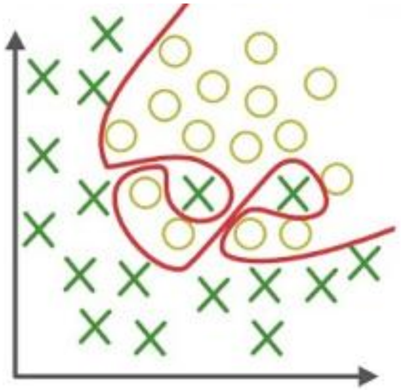
Overfitting & Underfitting

What is Overfitting?

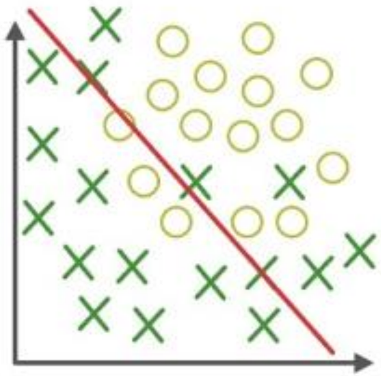
Overfitting: A model that models the training data too well.

Underfitting: A model that can neither model the training data nor generalize to new data





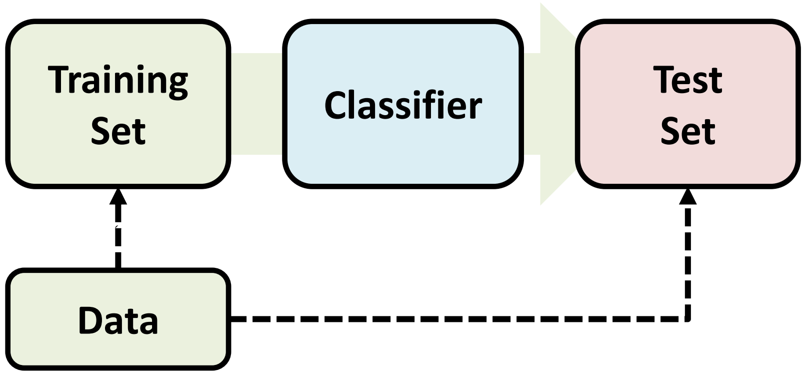
Over-fitting
(forcefitting--too good to be true)



Under-fitting
(too simple to explain the variance)

Hold-out Method

Hold-out

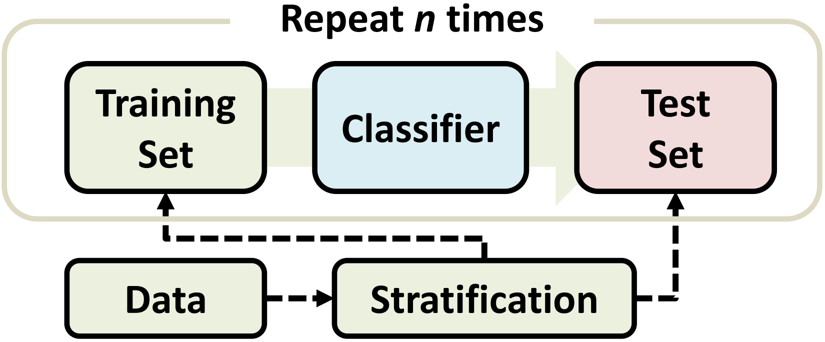


The method **splits** the data into **training** data and **test** data (usually **2/3 for train**, **1/3 for test**).

Then, **build** a classifier using the train data and **test** it using the test data.

Used when there are **thousands** of **instances** & several **hundred instances** from each **class**.

Repeated Hold-out



Holdout estimate can be made **more reliable** by **repeating** the process with **different subsamples**.

In each iteration, a certain **proportion** is **randomly** selected for **training** (possibly with **stratification**).

The **error rates** on the **different** iterations are **averaged** to **yield** an **overall error rate**.

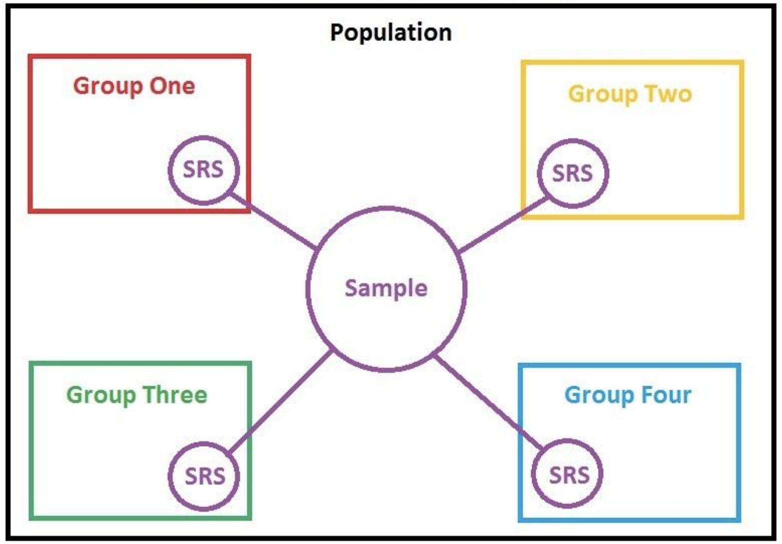
Stratification

The holdout method reserves a certain amount for testing and uses the remainder for training.

For "unbalanced" datasets, samples might not be representative (Few instances or none of some classes).

Stratified sample
An advanced version of balancing the data where each class is represented with approximately equal proportions

Stratified Random Sampling



Stratification is the process of dividing members of the population into homogeneous subgroups before sampling.

The strata should be mutually exclusive

Every element in the population must be assigned to only one stratum.

The strata should be collectively exhaustive

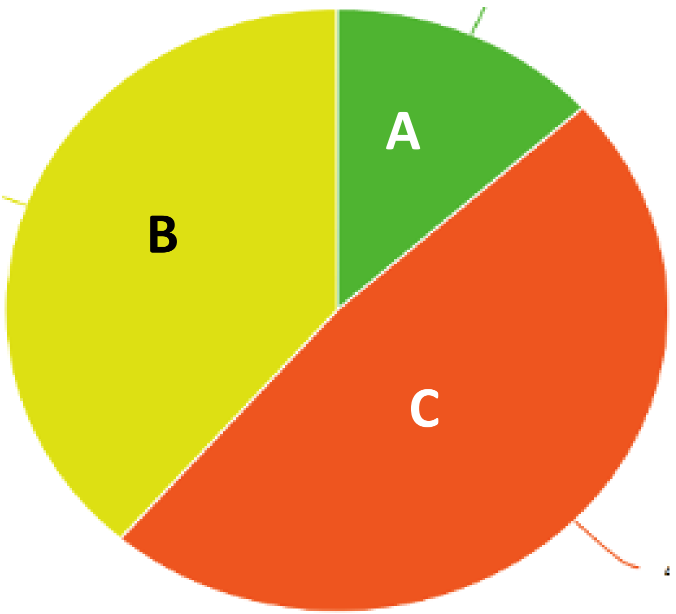
- No population element can be excluded.
- Simple random sampling is applied or;
- Systematic sampling is applied

This often improves the representativeness of the sample by reducing sampling error.

Stratification

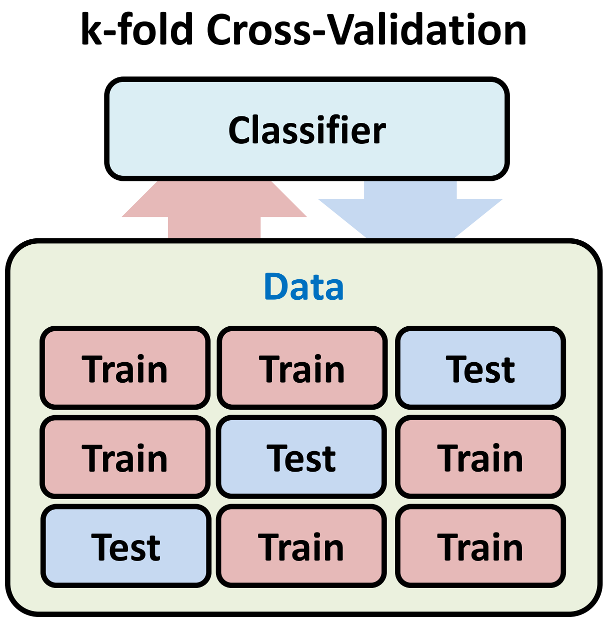
Example

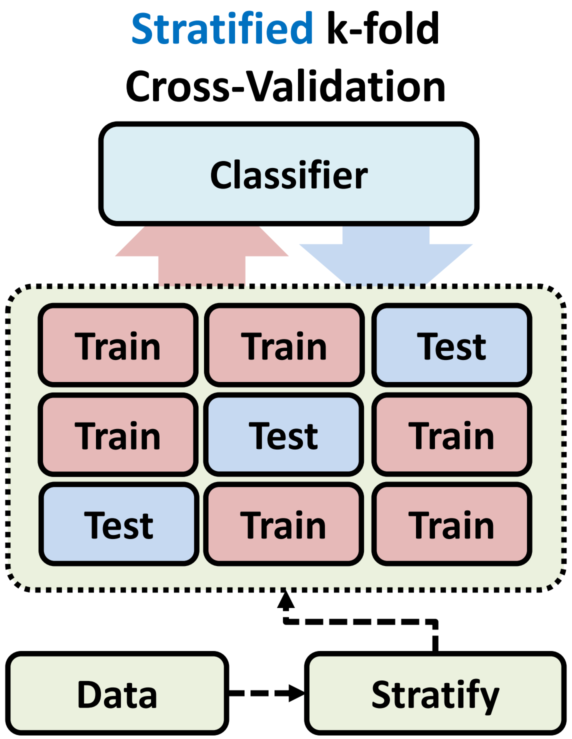
- Need to estimate average number of votes for each candidate in an election. The country has 3 towns:
  - Town A has 1 million factory workers
  - Town B has 2 million office workers
  - Town C has 3 million retirees.

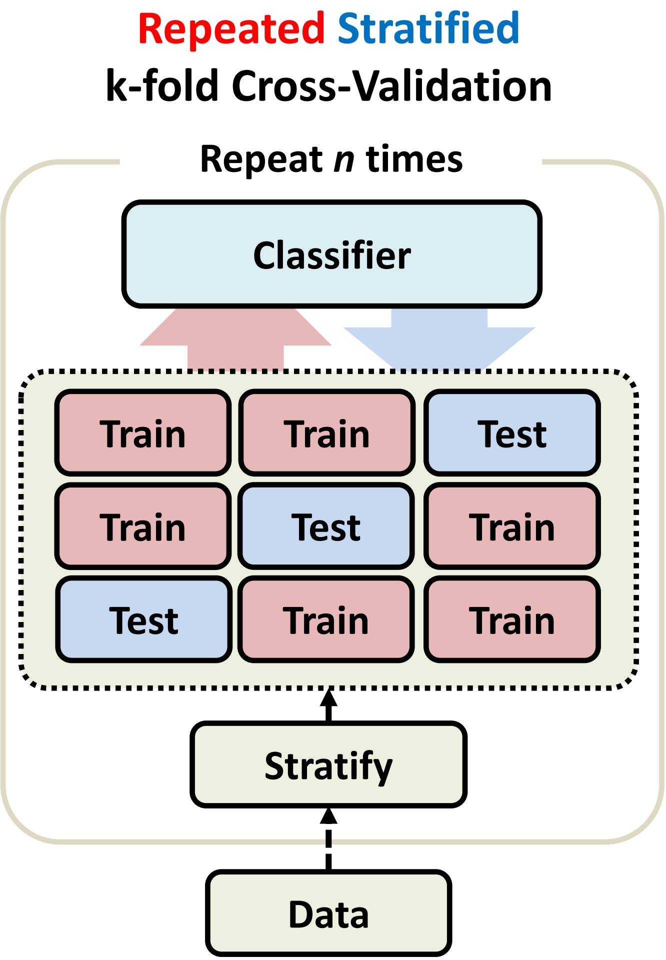


- A **random sample of size 60** over entire population will have some chance of:
  - The random sample not well balanced across these towns
  - Bias causing a significant error in estimation.
- **Random sample** of 10, 20 and 30 from Town A, B and C respectively
- Produce a smaller error in estimation for the same total size of sample.

k-Fold Cross-Validation







- Avoids overlapping test sets data is equally split into *k* subsets
- Some subset for testing, remainder for training.
- Recommended k = 10 (best choice)
- Results are averaged (reduces the estimate's variance)
- Standard: Stratified *k*-fold cross-validation.
- Even better: Repeated stratified *k*-fold cross-validation.

Leave-One-Out Cross-Validation

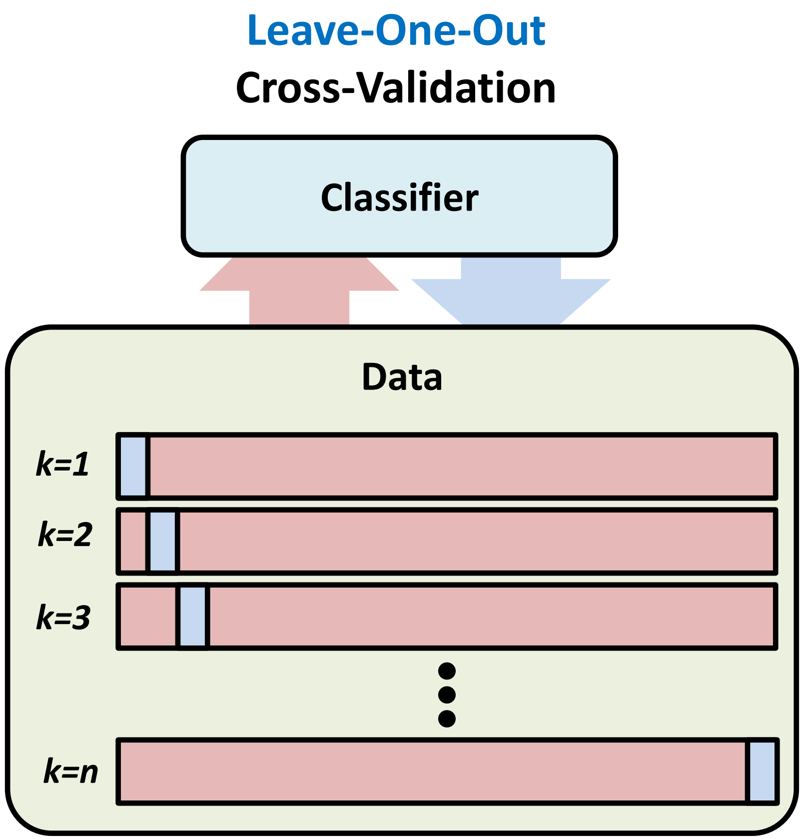
An extreme form of cross-validation

- Set number of folds to number of training instances;
- N-1 training instances, 1 test instance, and build classifier N times.

Make best use of the data

Involves no random sub-sampling

Very computationally expensive



Classification measures

- Accuracy is only one measure (error = 1 - accuracy).
- Accuracy is not suitable in some applications.
- In classification involving skewed or highly imbalanced data, e.g., network intrusion and financial fraud detections, we are interested only in the minority class.
  - High accuracy does not mean any intrusion is detected.
  - E.g., 1% intrusion. Achieve 99% accuracy by doing nothing.
- The class of interest is commonly called the **positive class**, and the rest **negative classes.**

How to evaluate the Classifier's Generalization Performance?

- We test a classifier on some test set
- We derive at the end the following *confusion matrix*:

		Predicted class	
		Pos	Neg	
Actual class	Pos	*TP*	*FN*	*P*
	Neg	*FP*	*TN*	*N*

Classification Measures

- Accuracy = $(TP+TN)/(P+N)$
- Error = $(FP+FN)/(P+N)$
- Precision = $TP/(TP+FP)$
- Recall/TP rate (Sensitivity) = TP/P
- Specificity = TN/N
- FP Rate = FP/N

precision can be understood as the probability that a randomly chosen predicted positive instance would be relevant
recall is how close we are to a specific target on average.

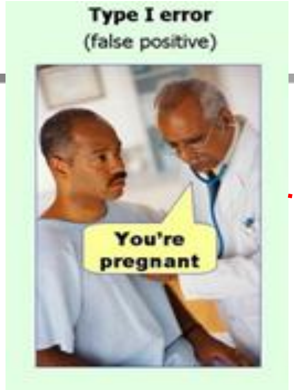
		Predicted class		
		Pos	Neg	**Recall**
Actual class	Pos	*TP*	*FN*	*P*
	Neg	*FP*	*TN*	*N*
	Precision			

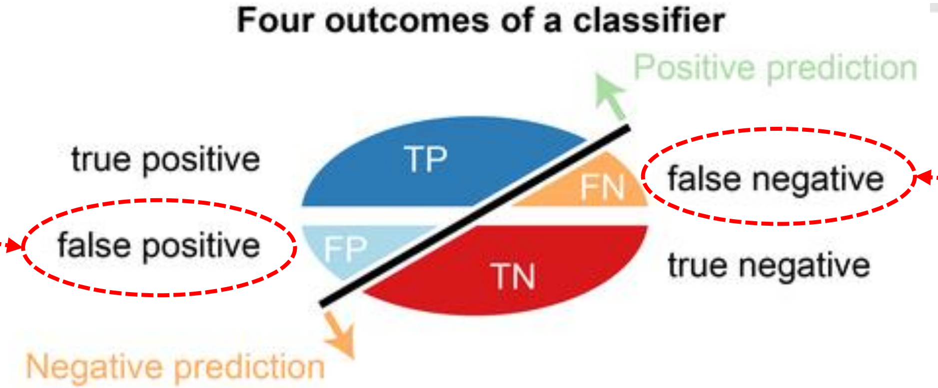
Example:

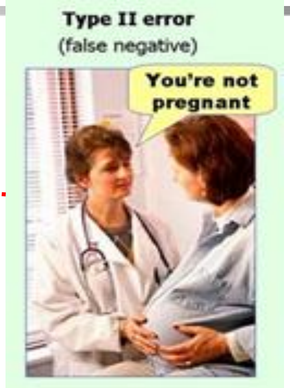
		Predicted class		
		Pos	Neg	
Actual class	Pos	*100*	*20*	*120*
	Neg	*30*	*50*	*80*
	Precision	**130**		

- Accuracy = (100 + 50)/(120 + 80) = 150/200 = 75
- Error = (30 +20)/(120 + 80) = 50/200 = 25%
- Precision = TP/(TP+FP) = 100 / 130 = 76.9%
- Recall/TP rate = TP/P = 100/120 = 83.3%
- Specificity = TN/N = 50/80 = 62.5%
- FP Rate = FP/N = 30/80 = 37.5%

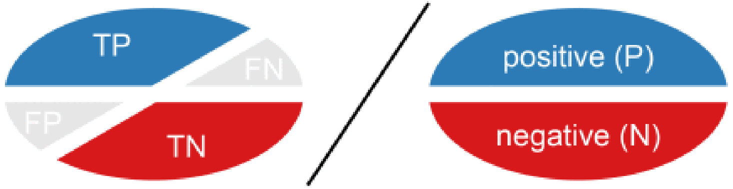
Classification Measures





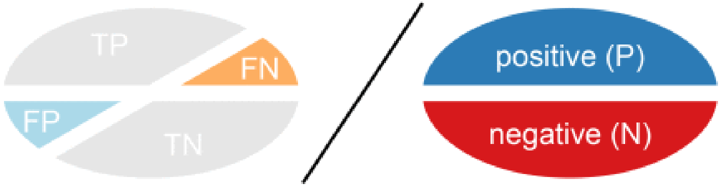


Accuracy: (TP + TN) / (P + N)



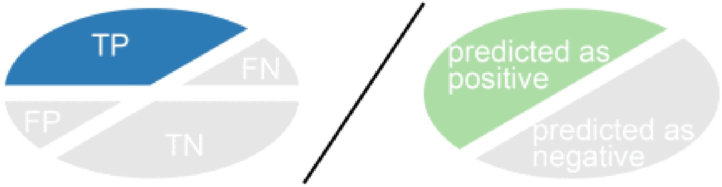
Accuracy is calculated as the total number of two correct predictions (TP + TN) divided by the total number of a dataset (P + N).

Error rate: (FP + FN) / (P + N)



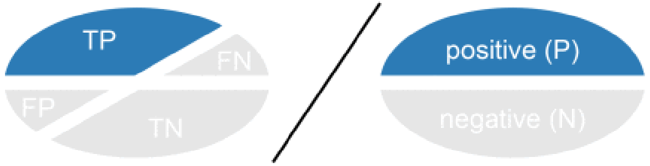
Error rate is calculated as the total number of two incorrect predictions (FN + FP) divided by the total number of a dataset (P + N).

Precision: TP / (TP + FP)



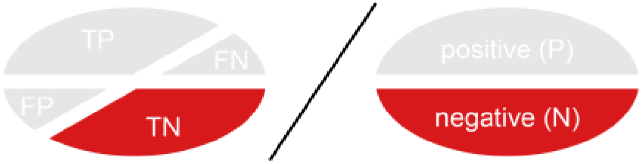
Precision is calculated as the number of correct positive predictions (TP) divided by the total number of positive predictions (TP + FP).

Sensitivity: TP / P



Sensitivity is calculated as the number of correct positive predictions (TP) divided by the total number of positives (P).

Specificity: TN / N



Specificity is calculated as the number of correct negative predictions (TN) divided by the total number of negatives (N).

False positive rate: FP / N

False positive rate is calculated as the number of incorrect positive predictions (FP) divided by the total number of negatives (N).

F_1-value (also called F_1-score)

- It is hard to compare two classifiers using two measures. F_1 score combines precision and recall into one measure

Precision = *TP*/(*TP*+*FP*)
Recall/TP rate (Sensitivity) = TP/P

$$F_1 = \frac{2pr}{p+r}$$

F_1-score is the harmonic mean of precision and recall.

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

- The harmonic mean of two numbers tends to be closer to the smaller of the two.
- For F_1-value to be large, both *p* and *r* much be large.

Cost Matrices & Classification

Goal of machine learning is based around making a computer **generalize** its **observation**.

The measure of performance of a machine learning algorithm is based on its accuracy of classifying a data set.

In practice, different types of classification errors often incur different costs

Examples:

- Terrorist **profiling**
- “Not a terrorist” correct 99.99% of the time
- Loan **decisions**
- Fault **diagnosis**
- **Promotional** mailing

Cost Matrices

True class	Predicted class: Positive	Predicted class: Negative
Positive	TP Cost	FN Cost
Negative	FP Cost	TN Cost

***Usually, TP Cost and TN Cost are set equal to 0!**

If the classifier outputs class probability, adjust it to minimize the expected prediction cost.

Expected cost is computed as dot product of class probabilities vector with appropriate column in cost matrix.

Cost Matrices & Classification

Example

Assume that a classifier returns for an instance probs $p_{pos} = 0.6$ and $p_{neg} = 0.4$.

The expected cost if the instance is classified as positive:
$0.6 * 0 + 0.4 * 10 = 4$

The expected cost if the instance is classified as negative:
$0.6 * 5 + 0.4 * 0 = 3$

Simple methods for cost sensitive learning
- Resampling of instances according to costs;
- Weighting of instances according to costs.

True class \ Predicted class	Positive	Negative
Positive	0	5
Negative	10	0

True class \ Predicted class	Positive	Negative
Positive	0.6 * 0	0.6 * 5
Negative	0.4 * 10	0.4 * 0

ROC Curve

How do we pick the probability threshold that gives us the best performance for the situation that we want?

ROC Curve: Receiver operating characteristic curve

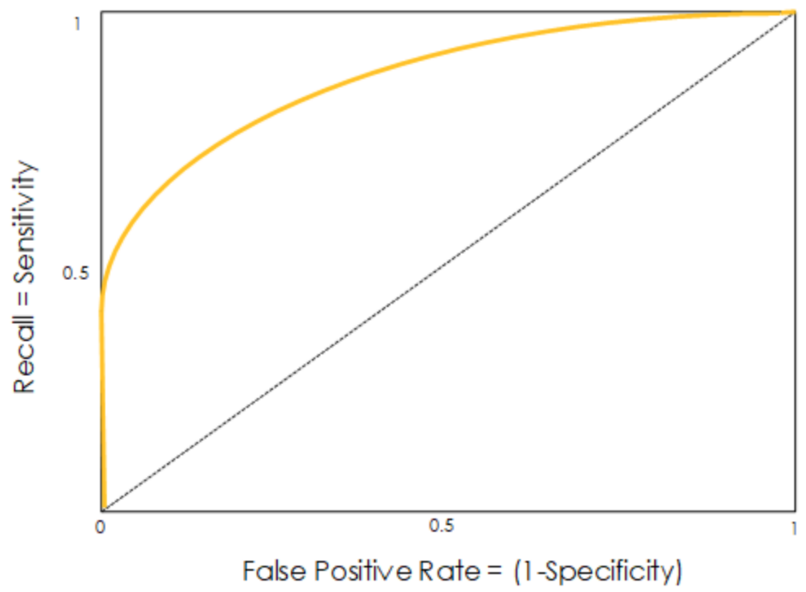
It is a graphical representation of how two of these metrics (Sensitivity or Recall and the Specificity) vary as we change this probability threshold.

- Recall/TP rate (Sensitivity) = TP/P
- Specificity = TN/N

		Predicted class	
		Pos	Neg
Actual class	Pos	*TP*	*FN*
	Neg	*FP*	*TN*

- It is a plot of the true positive rate (TPR) against the false positive rate (FPR).
- True positive rate: $$TPR = \frac{TP}{TP + FN}$$
- False positive rate: $$FPR = \frac{FP}{TN + FP}$$

The ROC Curve



ROC Curves and Analysis

A ROC (Receiver Operating Characteristics) curve :

Classifier 1

True	Predicted pos	Predicted neg
pos	40	60
neg	30	70

Classifier 2

True	Predicted pos	Predicted neg
pos	70	30
neg	50	50

Classifier 3

True	Predicted pos	Predicted neg
pos	60	40
neg	20	80

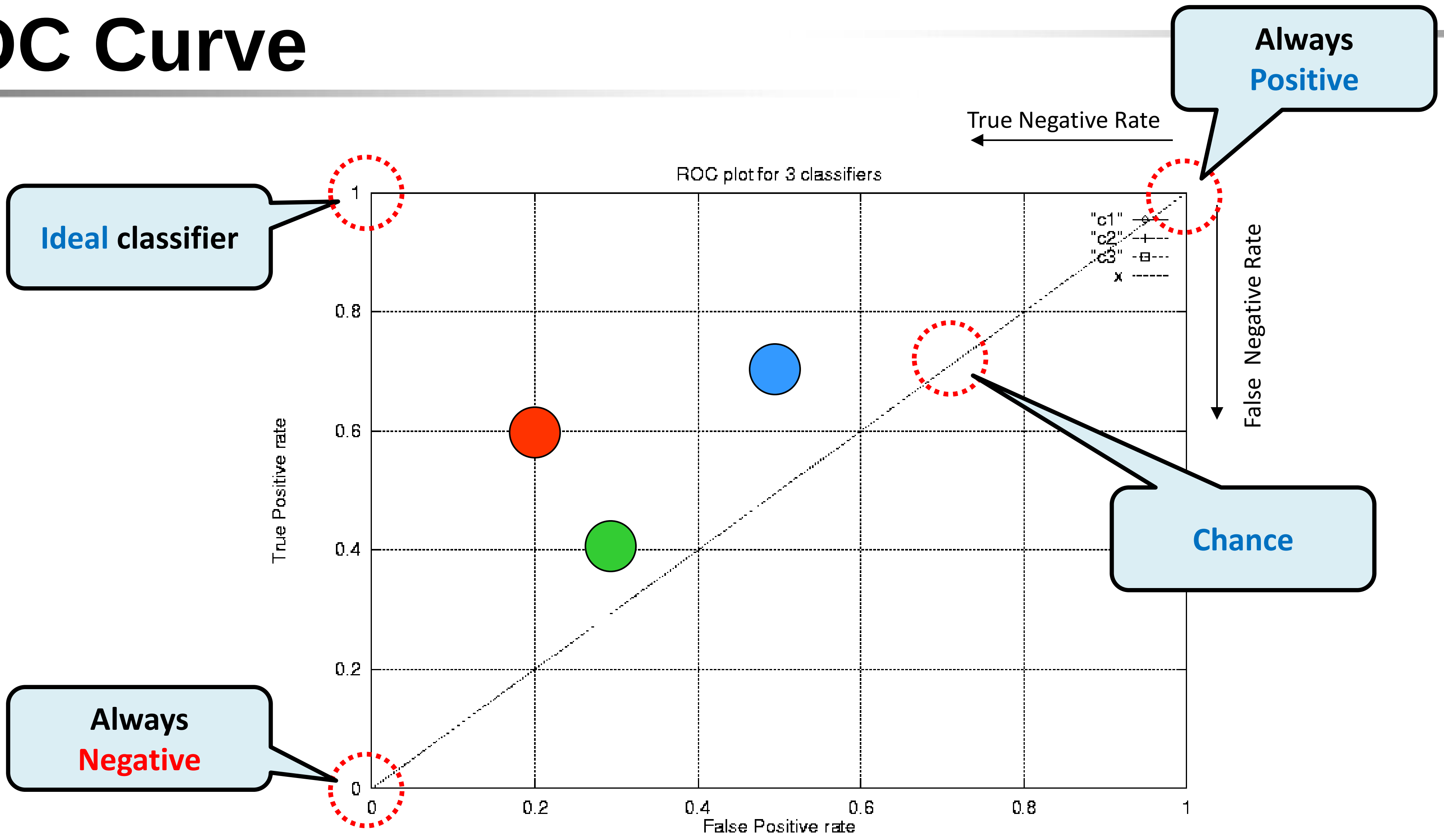
<u>Classifier 1</u>
TPr = 0.4
FPr = 0.3

<u>Classifier 2</u>
TPr = 0.7
FPr = 0.5

<u>Classifier 3</u>
TPr = 0.6
FPr = 0.2

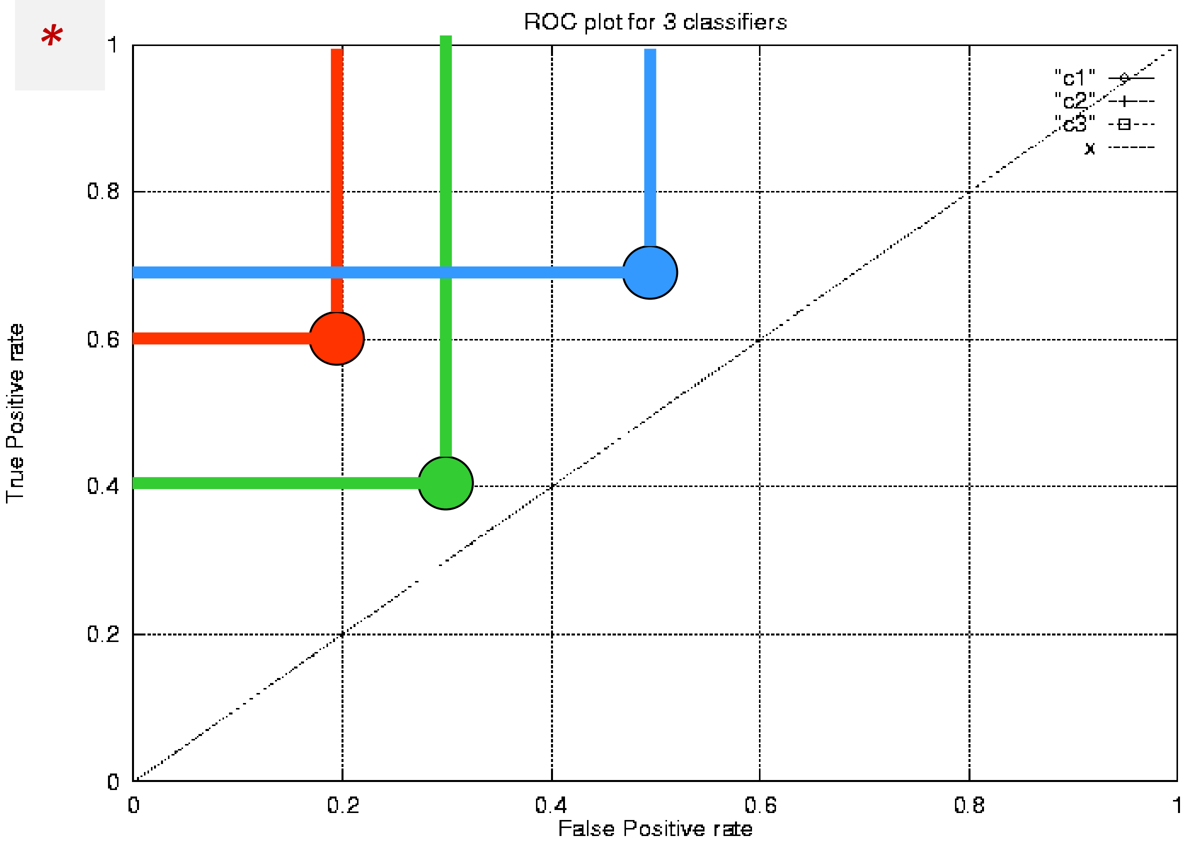
TPr – True Positive rate
FPr – False Positive rate

ROC Curve


Ideal classifier
Always Positive
True Negative Rate
False Negative Rate
Always Negative
Chance
ROC plot for 3 classifiers
"c1"
"c2"
"c3"
x
True Positive rate
False Positive rate
0
0.2
0.4
0.6
0.8
1

Dominance in the ROC Curve

Which one is the best option?



Can also use Euclidean distance formula to calculate how far from (0,1)

Classifier A dominates classifier B if and only if $TPr_A > TPr_B$ and $FPr_A < FPr_B$.

*How lose each classifier to point *?*

How to calculate?

Thank you