Classification and Regression

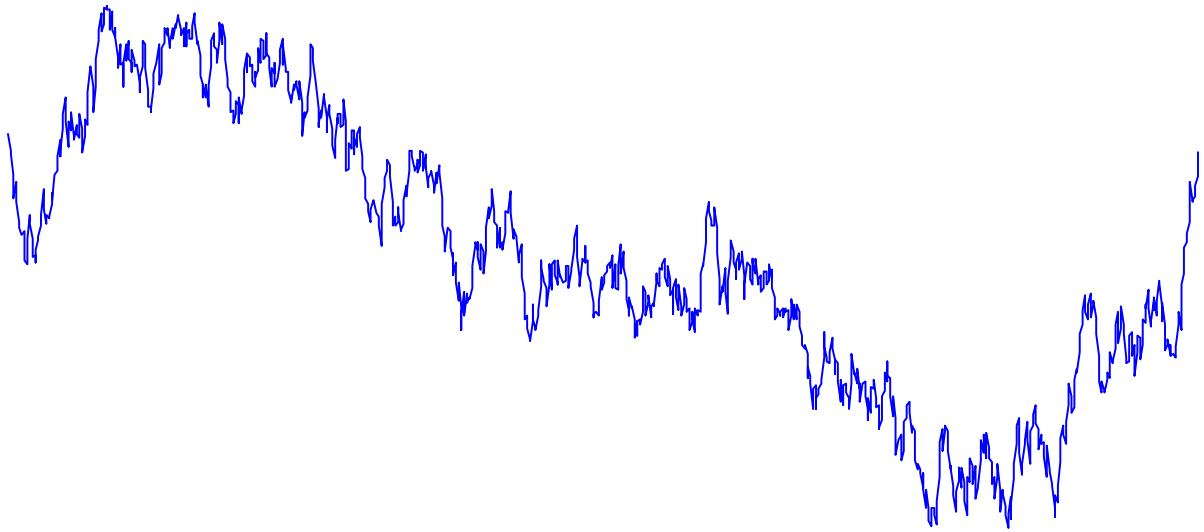
Part 1

Riccardo Guidotti

UNIVERSITÀ DI PISA

Syllabus

- Problems Setting
- Models Evaluation
- Instance-based Models
- Linear Models
- Tree-based Models
- Interval-based Model
- Shapelet-based Models
- Dictionary-based Models



- Time Series Transformations

Problems Setting

Time Series Classification - Examples

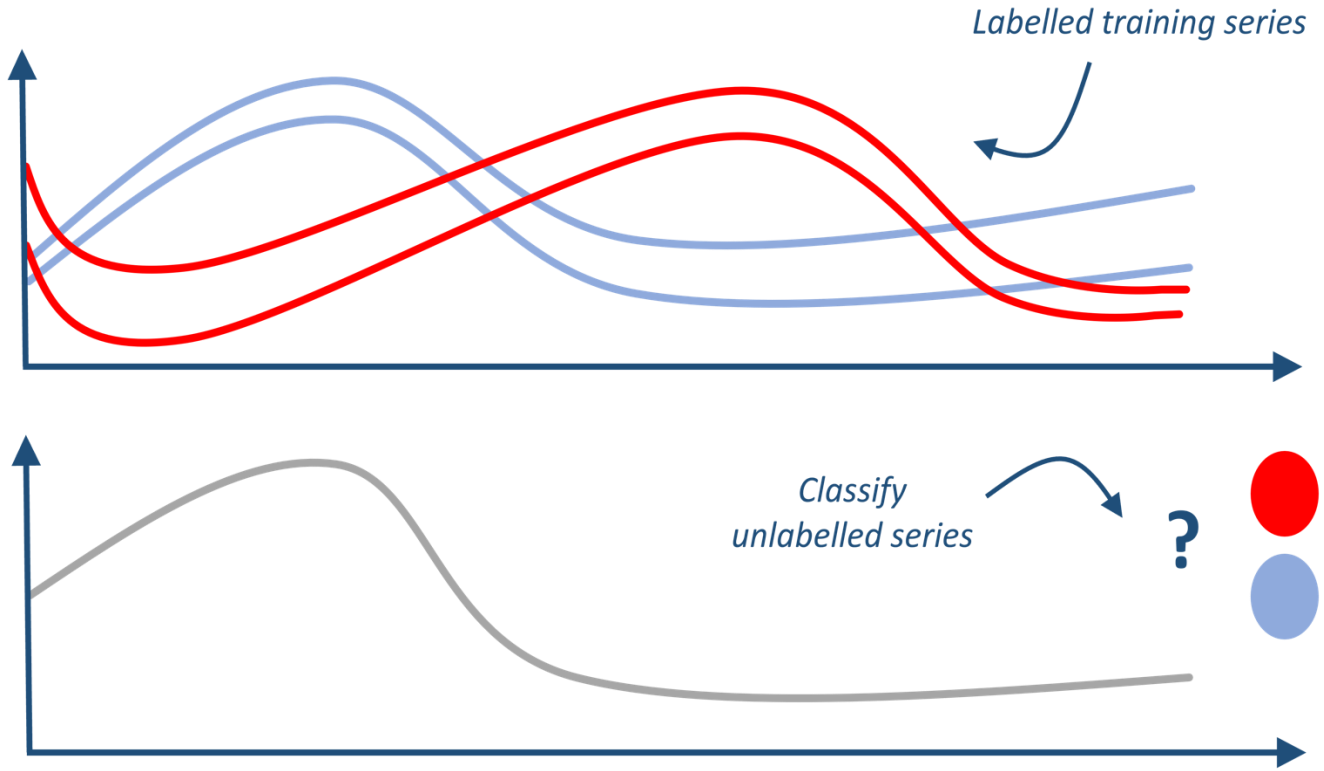
- ***Predictive variables***: multivariate time series as accelerometers on the x, y, z axes and speed coming from cars black boxes. ***Target variable***: means of transport, crash vs noCrash, car model, etc.
- ***Predictive variables***: multivariate time series as accelerometers on the x, y, z axes, heart rate and number of steps per sec coming from personal smart devices. ***Target variable***: type of movement, type of device, next location.
- ***Predictive variables***: multivariate time series as machine sensors such has temperature, humidity, number of rotations, accelerations, etc. ***Target variable***: failure vs nofailure, product in production, type of machine, etc.
- ***Predictive variables***: multivariate time series as EEG or ECG. ***Target variable***: symptoms, disease, treatment, etc.
- ***Predictive variables***: univariate or multivariate time series as students' marks. ***Target variable***: student background, university, etc.

Time Series Regression - Examples

- ***Predictive variables***: multivariate time series as accelerometers on the x, y, z axes and speed coming from cars black boxes. **Target variable**: engine temperature, number of car repairs, etc.
- ***Predictive variables***: multivariate time series as accelerometers on the x, y, z axes, heart rate and number of steps per sec coming from personal smart devices. ***Target variable***: age, number of times gym is made.
- ***Predictive variables***: multivariate time series as machine sensors such has temperature, humidity, number of rotations, accelerations, etc. ***Target variable***: number of products realized, number of failures.
- ***Predictive variables***: multivariate time series as EEG or ECG. ***Target variable***: patient age, patient weight, etc.
- ***Predictive variables***: univariate or multivariate time series as students' marks. ***Target variable***: student age, family income.

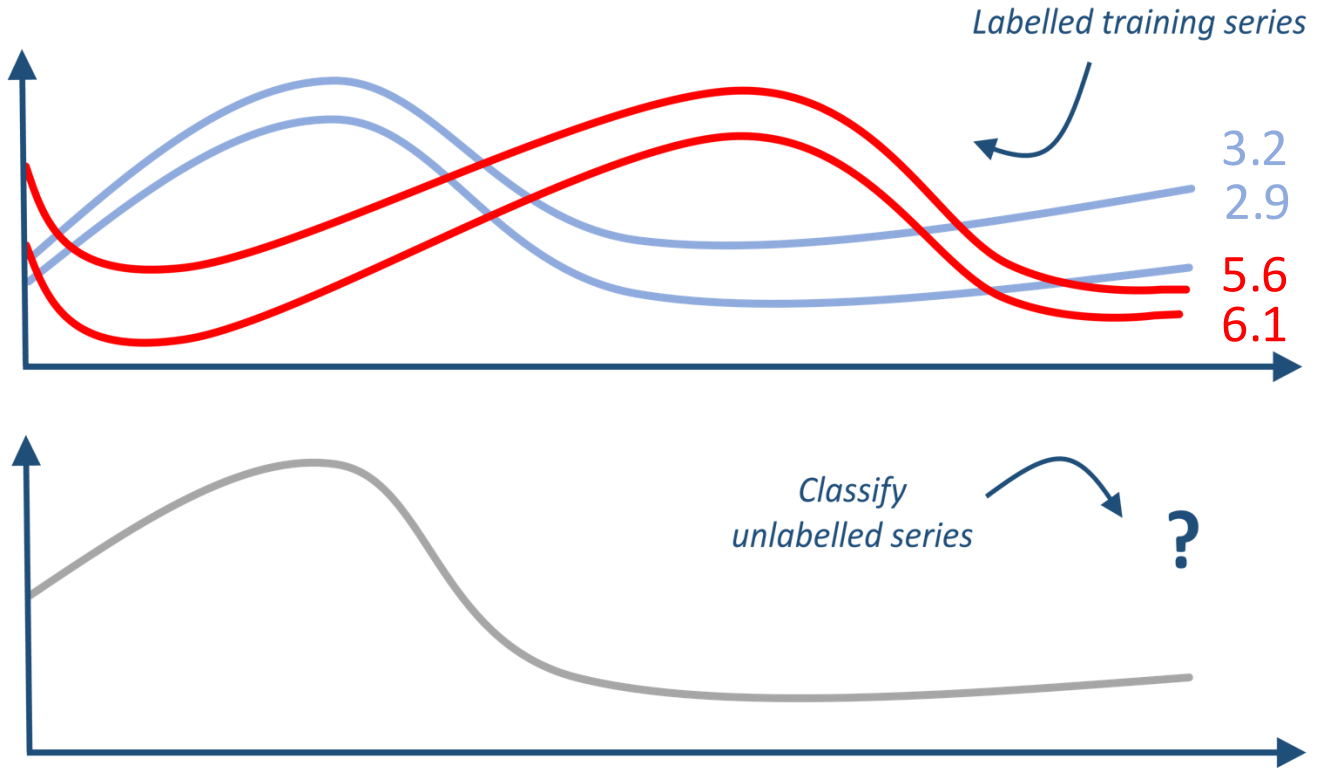
Time Series Classification - TSC

- Given a dataset $X = \{T_1, \ldots T_n\}$, TSC is the task of training a model f to predict an exogenous categorical output y for each time series T, i.e., $f(T) = y$.



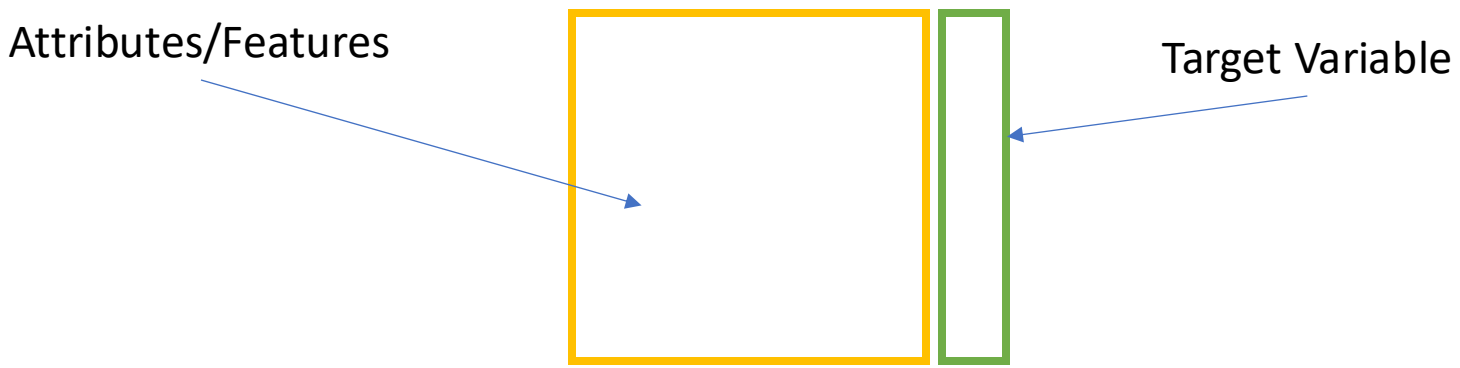
Time Series Extrinsic Regression - TSER

- Given a dataset $X = \{T_1, \ldots T_n\}$, TSER is the task of training a model f to predict an exogenous continuous output y for each time series T, i.e., $f(T) = y$.



Supervised Learning

- Supervised learning refers to problems where the value of a target attribute should be predicted based on the values of other attributes.
- Problems with a ***categorical target*** attribute are called **classification**, problems with a ***numerical target*** attribute are called **regression**.



What is Machine Learning?

- Machine Learning (ML) is the science (and art) of programming computers that can learn from data.

"ML is the field of study that gives computers the ability to learn without being explicitly programmed"
(Arthur Samuel, 1959)

What is Machine Learning?

- Machine Learning (ML) is the science (and art) of programming computers that can learn from data.

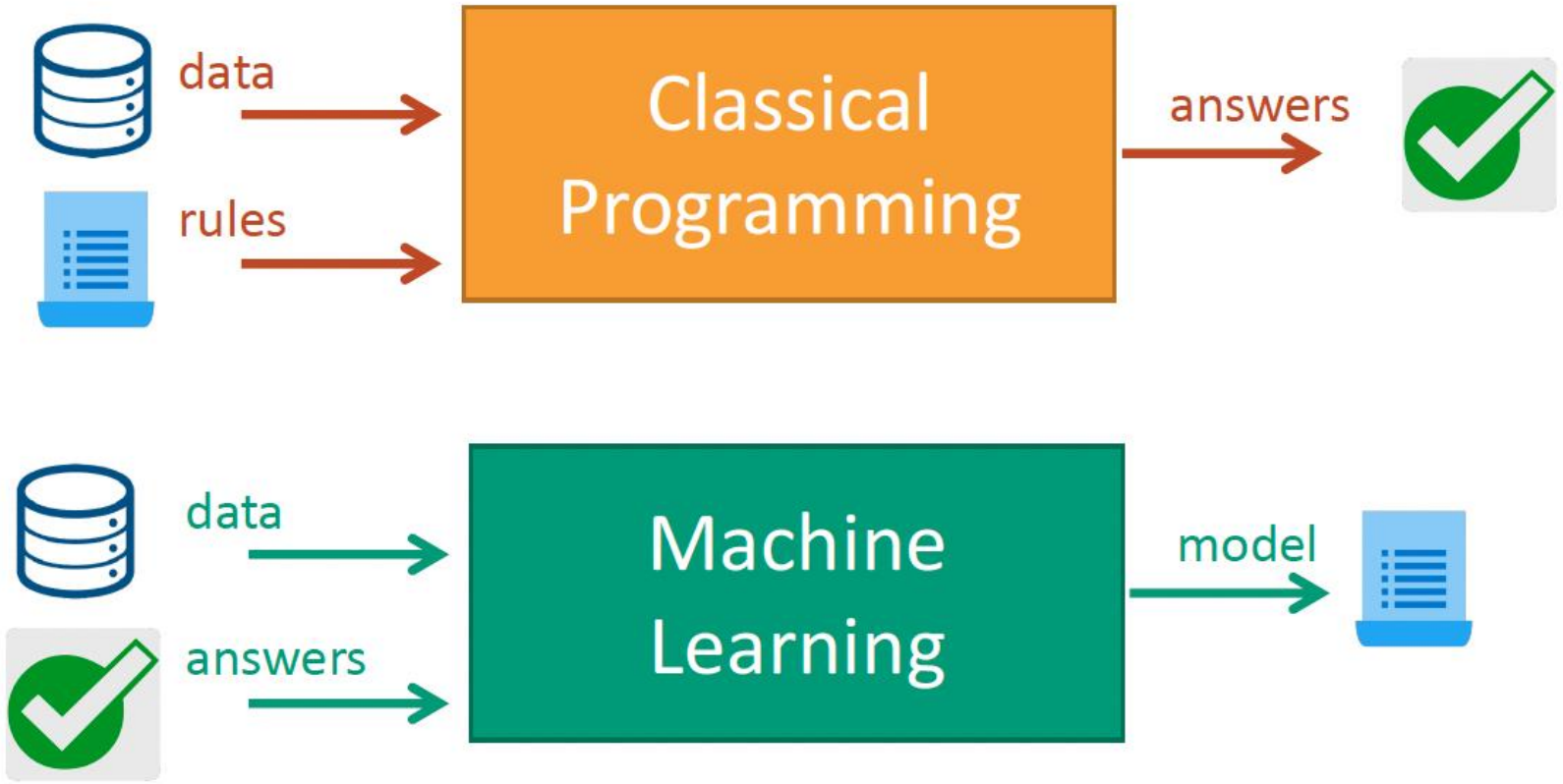
"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."
(Tom Mitchell, 1997)

Classical Programming vs Machine Learning

- A ML system is trained rather than programmed

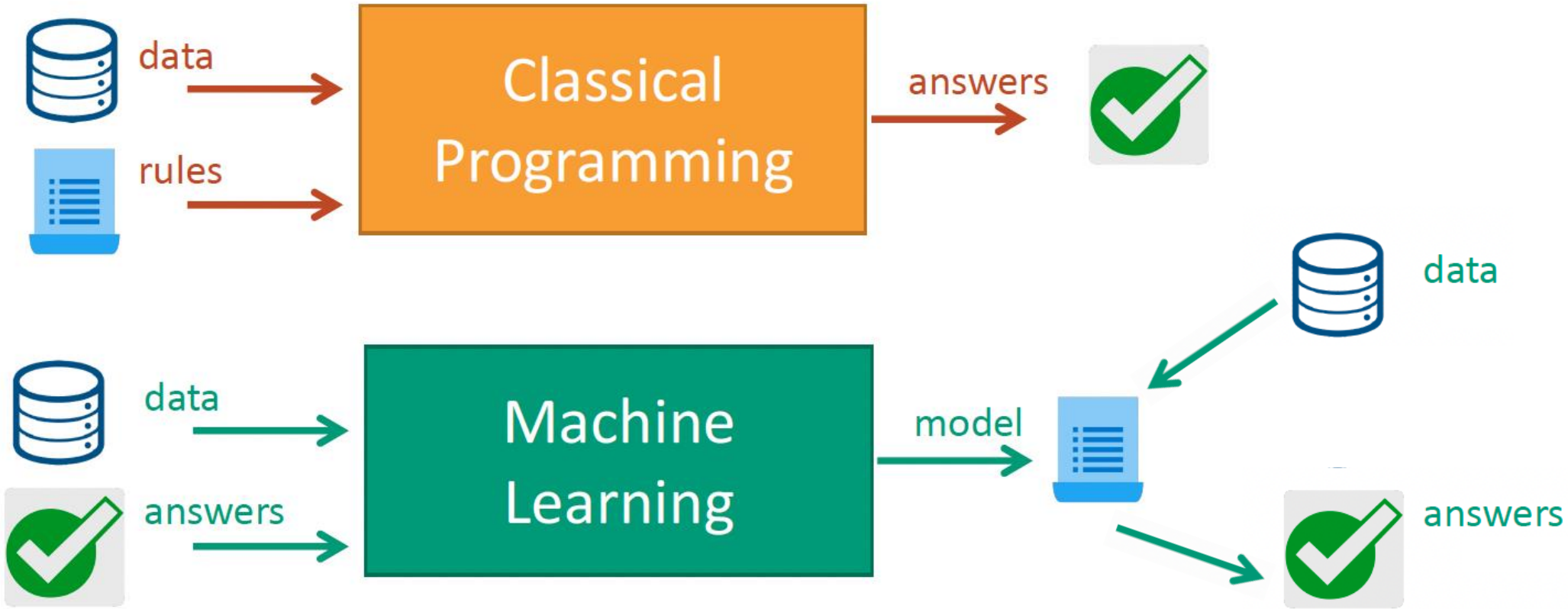
Classical Programming vs Machine Learning

- A ML system is trained rather than programmed

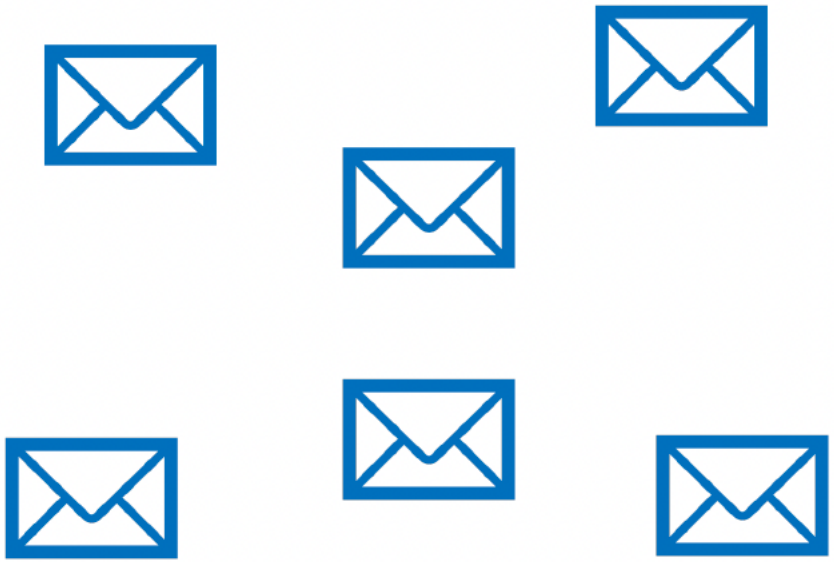


Classical Programming vs Machine Learning

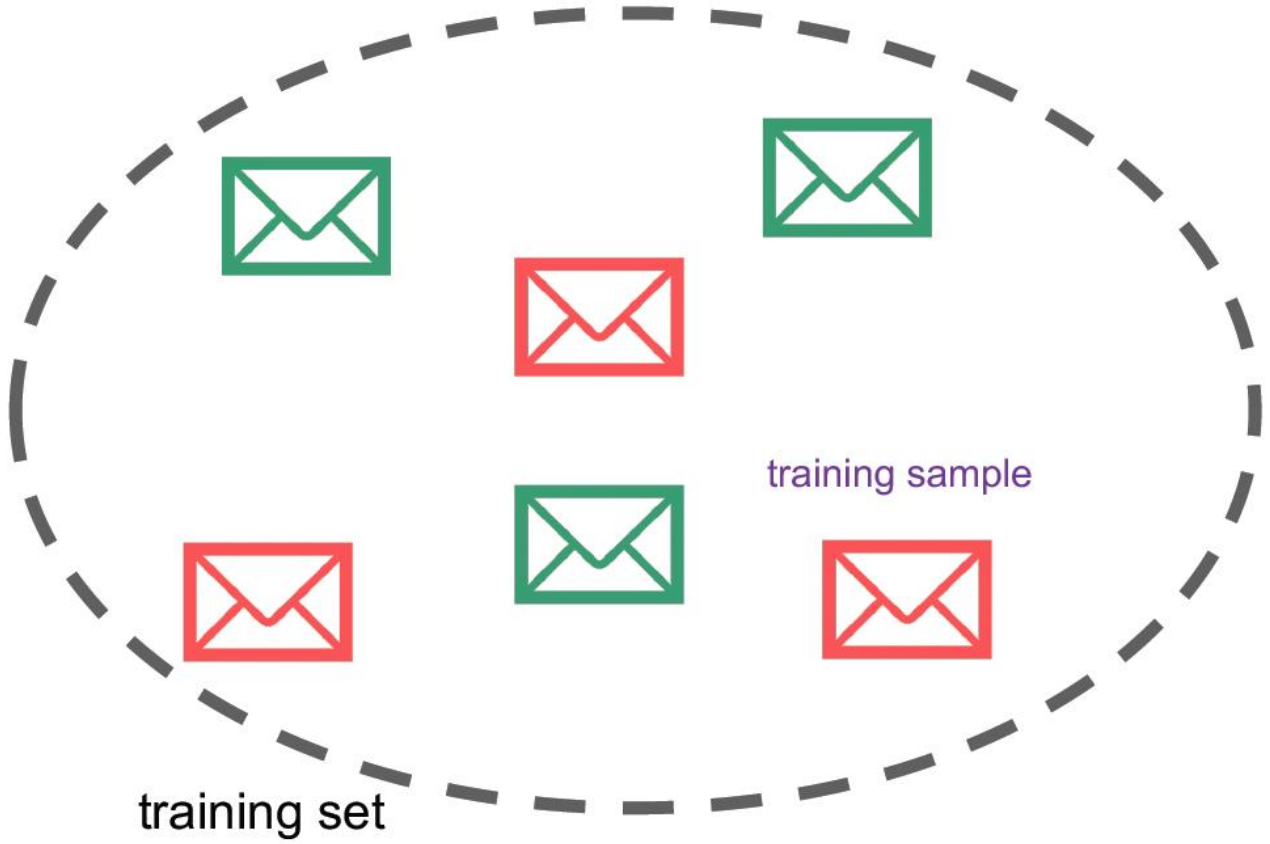
- A ML system is trained rather than programmed

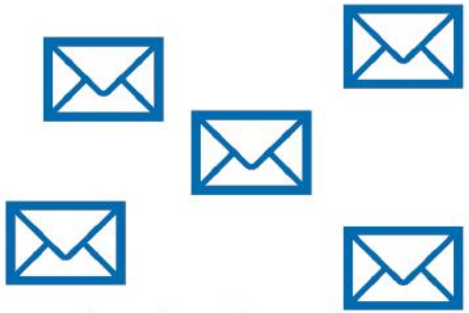


Example Spam Filter



Example Spam Filter



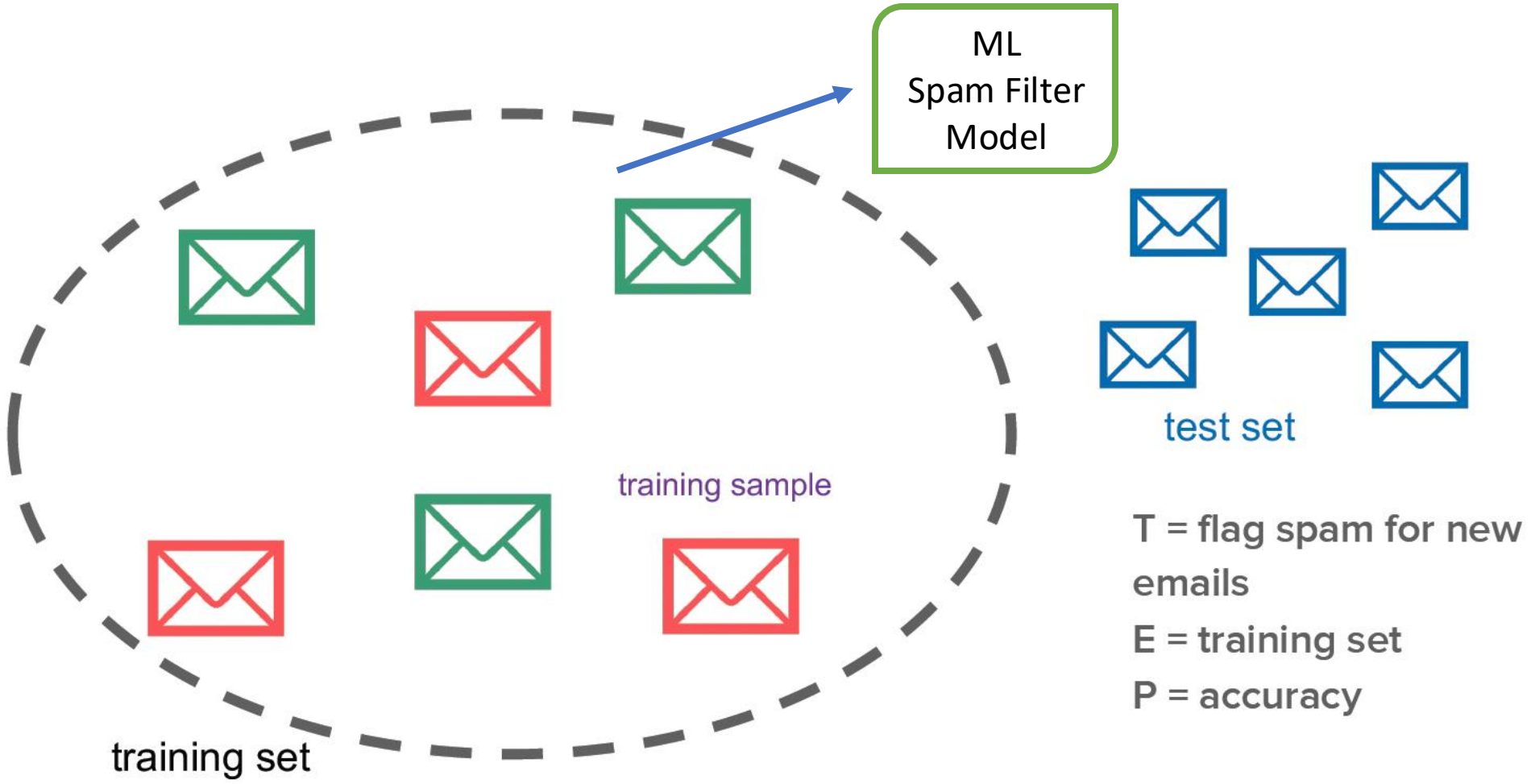


T = flag spam for new emails

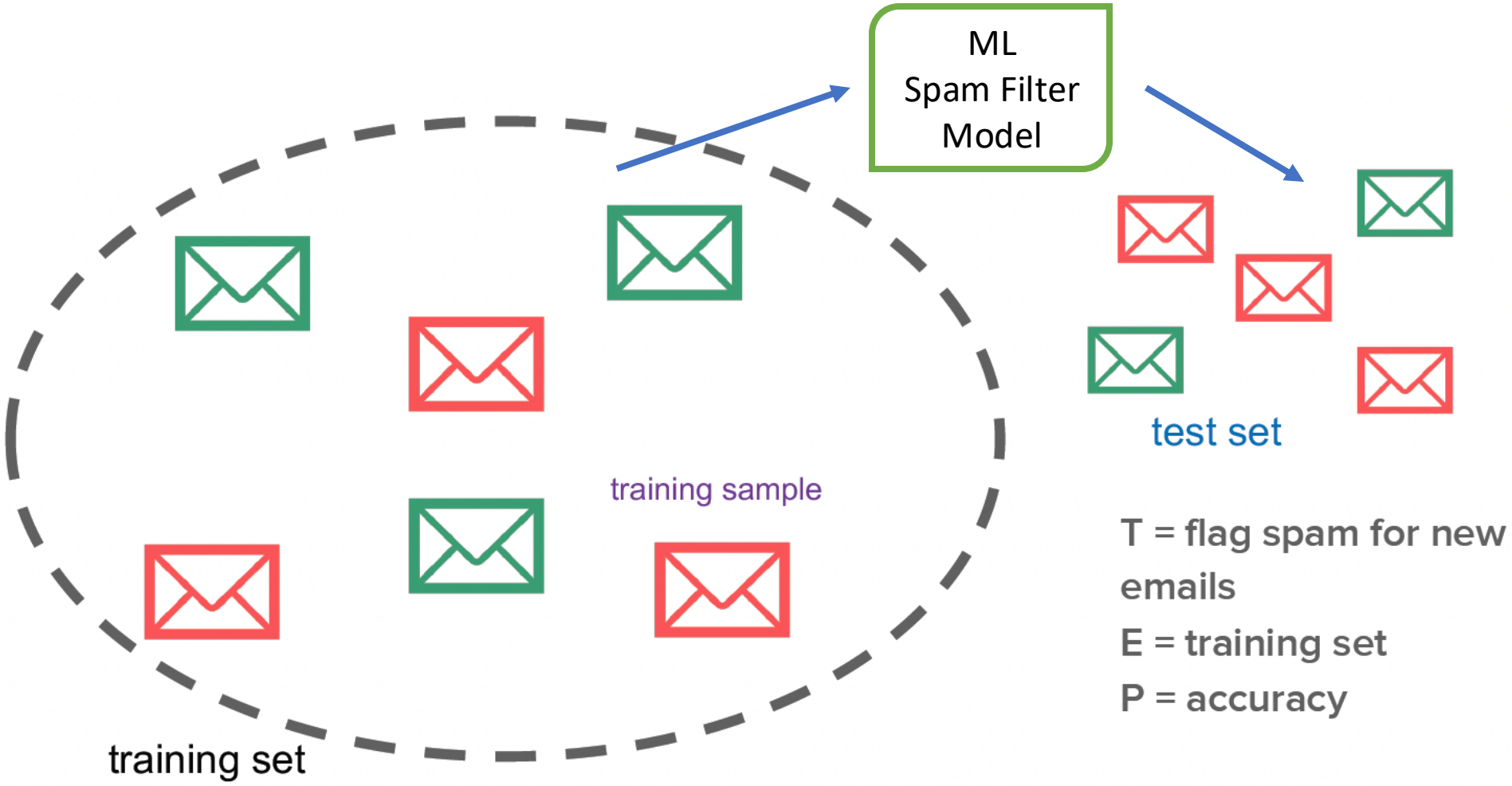
E = training set

P = accuracy

Example Spam Filter

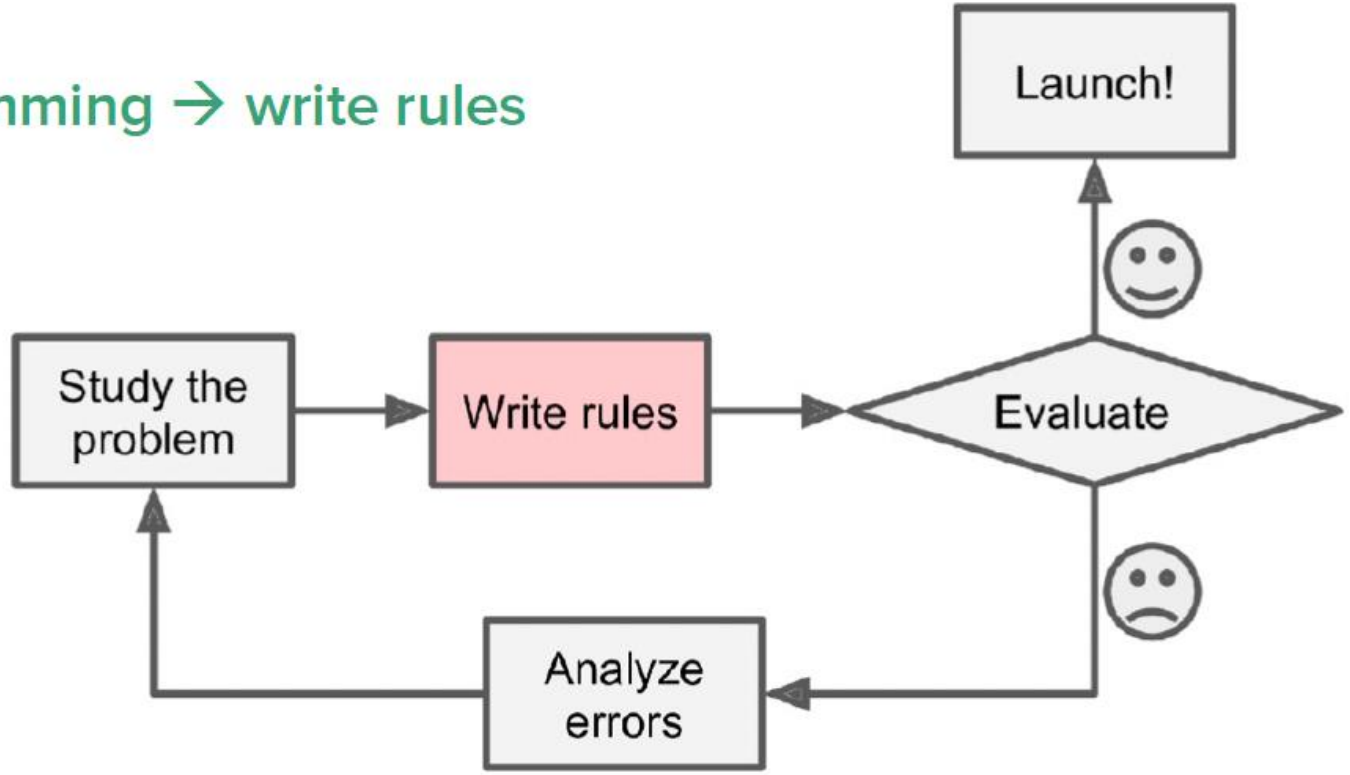


Example Spam Filter



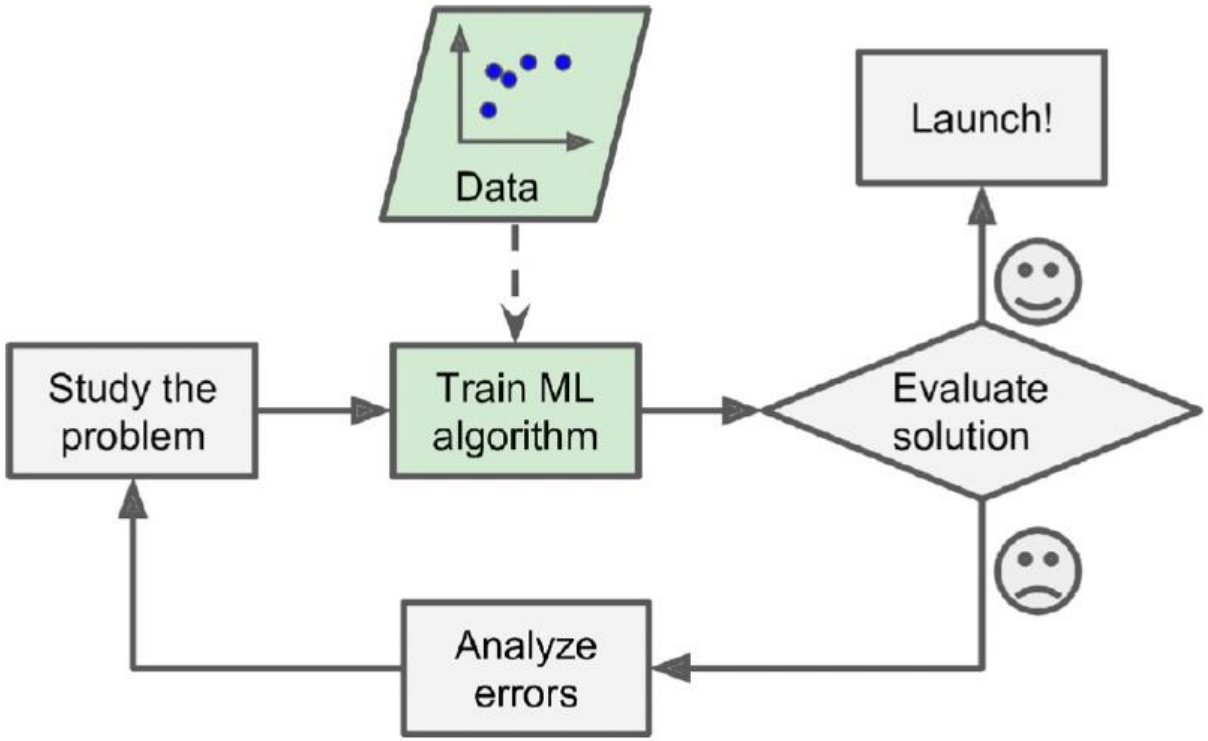
Why do we want to use Machine Learning?

Traditional Programming → write rules



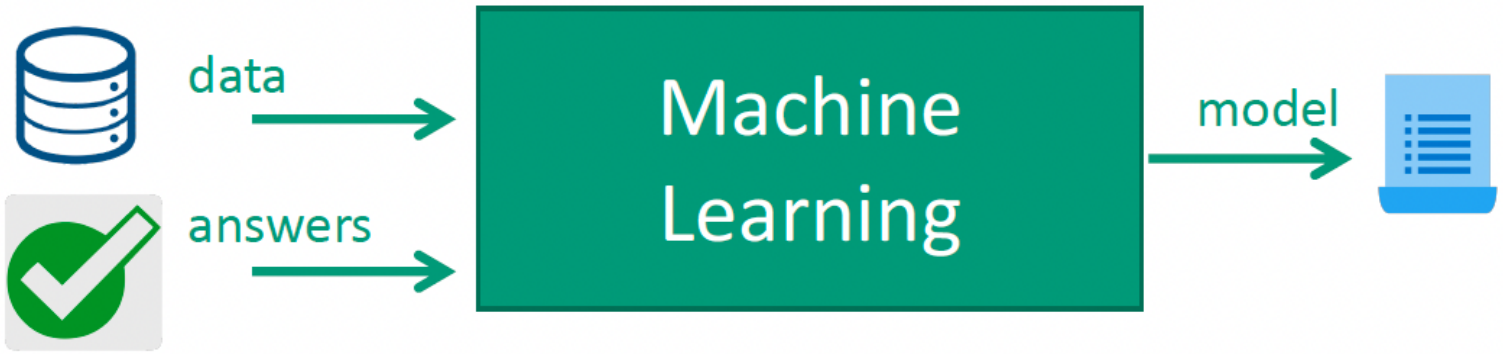
Why do we want to use Machine Learning?

Machine Learning: train based on data (examples)



Why do we want to use Machine Learning?

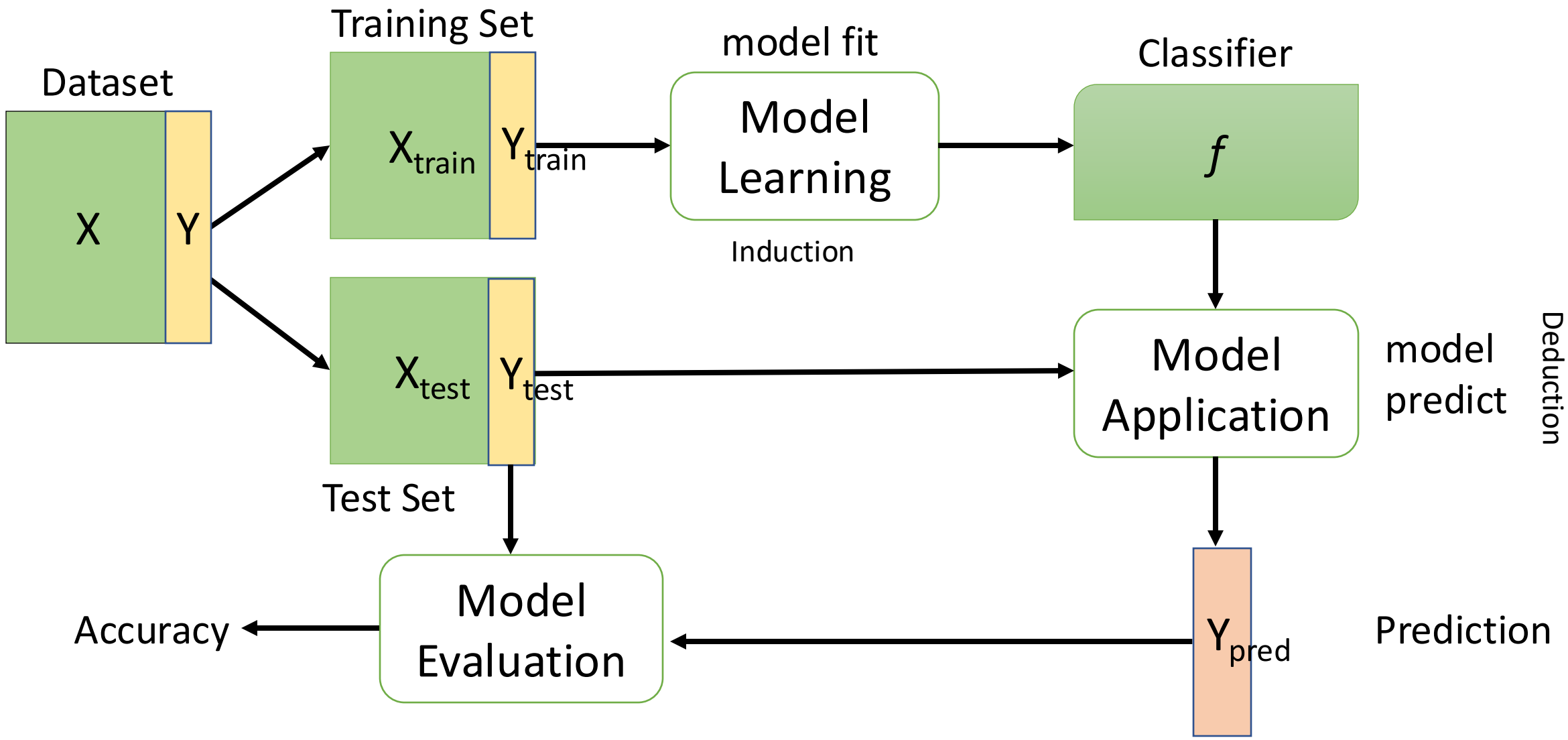
- Problems for which existing solutions require a lot of finetuning or a long list of rules
- Complex problems for which a traditional approach yields no good solution
- Changing environments
- Getting insights about complex problems and large amount of data



What is a Supervised Predictive Task?

- Classification/Regression consists in learning a **model/function *f*** that maps each attribute set ***x*** into one a class label/target value ***y***: *f(x) = y*
- The ***learning*** is performed on a given a collection of records named ***training set*** where each record is by characterized by a tuple ***(x, y)***, where ***x*** is the attribute set and ***y*** is the target variable
  - x: attribute, predictor, independent variable, input
  - y: class/target value, response, dependent variable, output
- **Goal**: previously unseen records should be predicted as accurately as possible. A ***test set*** is used to determine the effectiveness of the model ***f***.
- Usually, the available dataset is divided into training and test sets, with training set used to build the model and test set used to evaluate it.

What is a Supervised Predictive Task?



Models Evaluation

Evaluation Classification

- Let suppose we have a vector y of actual/real class labels:
- y = [0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 0]

- Let name y' the vector returned by a kNN:
- y' = [0 0 1 1 1 0 0 1 0 1 1 1 0 0 0 0]

- **Accuracy** = # record correctly classified / # records classified
- Accuracy = 11 / 16 = 0.69

Confusion Matrix

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- **Confusion Matrix:**

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

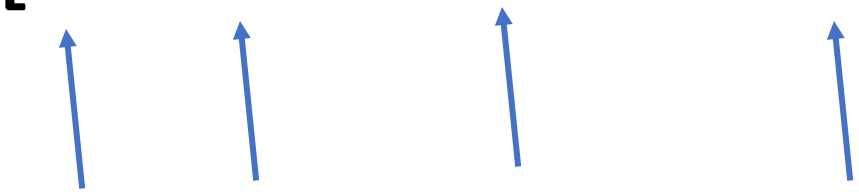
b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Confusion Matrix

- y = [0 0 0 1 1 1 0 1 0 1 0 1 1 1 0 0]
- y' = [0 0 1 1 1 0 0 1 0 1 1 1 0 0 0 0]



TN FP FN TP

Confusion Matrix

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

- Most widely-used metric: $$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

- Error Rate = 1- Accuracy

Limitation of Accuracy

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

- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
- Accuracy is misleading because model does not detect any class 1 example

Cost-Sensitive Measures

$$\text{Precision (p)} = \frac{TP}{TP + FP}$$

$$\text{Recall (r)} = \frac{TP}{TP + FN}$$

$$\text{F-measure (F)} = \frac{2rp}{r + p} = \frac{2TP}{2TP + FN + FP}$$

- Precision is biased towards C(Yes|Yes) & C(Yes|No)
- Recall is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is biased towards all except C(No|No)

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

Binary vs Multiclass Evaluation

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	TP	FN
	Class=No	FP	TN

		PREDICTED CLASS		
		Class=A	Class=B	Class=C
ACTUAL CLASS	Class=A	TP-A		
	Class=B		TP-B	
	Class=C			TP-C

Accuracy = TP+TN / (TP+TN+FN+FP) = # correct / N

Accuracy = # correct / N = (TP-A + TP-B + TP-C) / N

Multiclass Evaluation

		PREDICTED CLASS		
ACTUAL CLASS		Class=A	Class=B	Class=C
	Class=A	TP-A	a	b
	Class=B	c	TP-B	d
	Class=C	e	f	TP-C

$$\text{Precision (p)} = \frac{TP}{TP + FP}$$

$$\text{Recall (r)} = \frac{TP}{TP + FN}$$

$$\text{F-measure (F)} = \frac{2rp}{r + p} = \frac{2TP}{2TP + FN + FP}$$

A	PREDICTED CLASS		
ACTUAL CLASS		Class=A	Class=Not A
	Class=A	TP-A	a + b
	Class=Not A	c + e	TP-B + TP-C + d + f

B	PREDICTED CLASS		
ACTUAL CLASS		Class=B	Class=Not B
	Class=B	TP-B	c + d
	Class=Not B	a + f	TP-A + TP-C + b + e

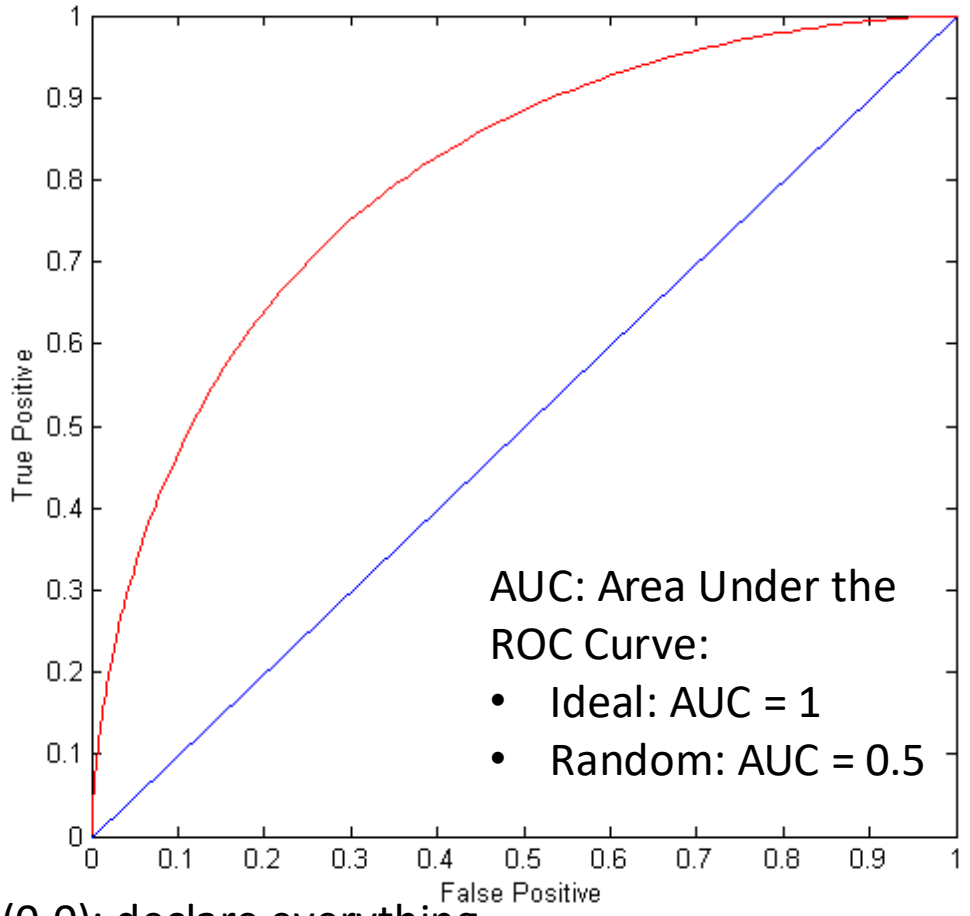
C	PREDICTED CLASS		
ACTUAL CLASS		Class=C	Class=Not C
	Class=C	TP-C	e + f
	Class=Not C	b + d	TP-A + TP-B + a + c

Receiver Operating Characteristic Curve

- It illustrates the ability of a binary classifier as its discrimination threshold THR is varied.
- ***ROC*** curve is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at various THR.
- TPR = TP / (TP + FN) also known as ***sensitivity***, ***recall*** or probability of detection.
- FPR = FP / (TN + FP) also known as probability of ***false alarm.***
- Diagonal line: random guessing
- Below diagonal line: prediction is opposite of the true class

(0,1): ideal

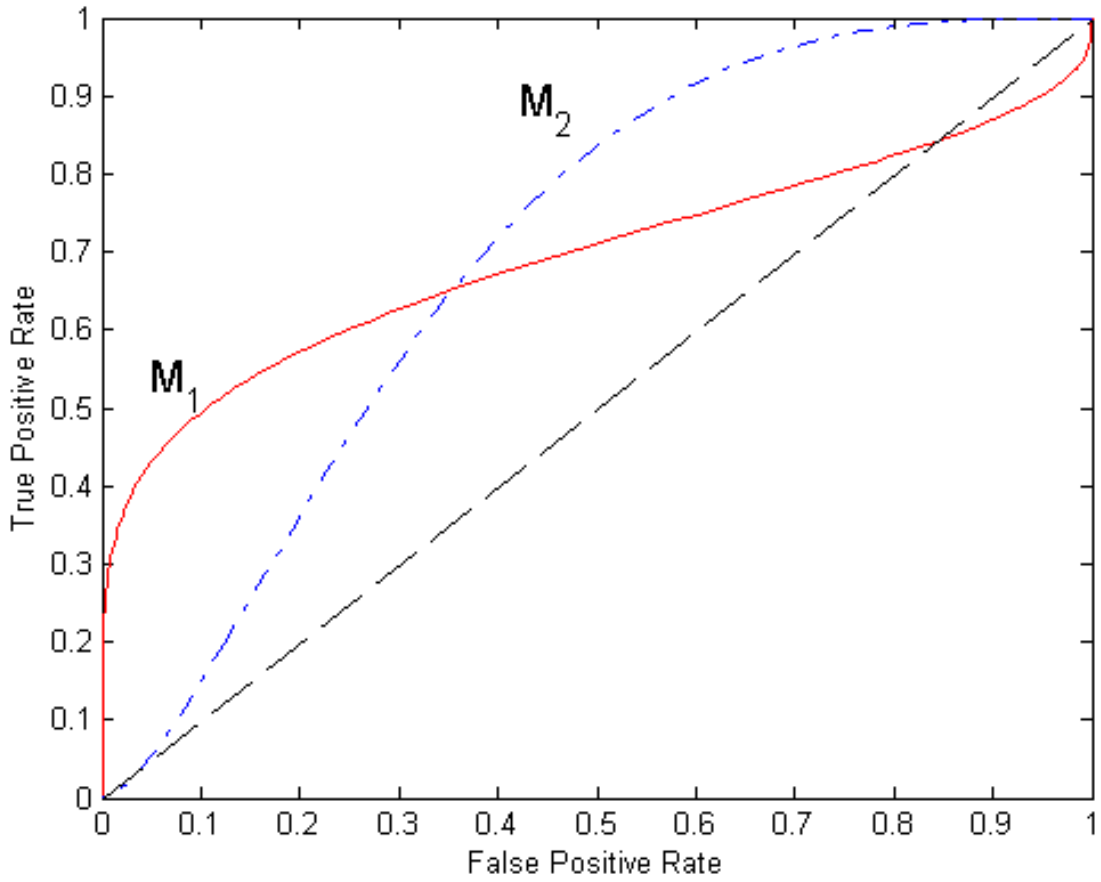
(1,1): declare everything to be positive class



(0,0): declare everything to be negative class

Using ROC for Model Comparison

- No model consistently outperform the other
- M1 is better for small FPR
- M2 is better for large FPR



Evaluating Regression

- **Coefficient of determination** R^2
  - is the proportion of the variance in the dependent variable that is predictable from the independent variable(s)

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

hat means predicted

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \text{ and } \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \epsilon_i^2$$

- **Mean/Media Squared/Absolute Error** MSE/MAE
  - a risk metric corresponding to the average/median value of the squared (quadratic)/absolute error

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2 \qquad \text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i|$$

Evaluating Regression

- **Mean Absolute Percentage Error** MAPE
  - average absolute error normalized between 0 and 100 (the lower the better)

$$MAPE(y, \hat{y}) = 100 \ * \ \frac{1}{n} \sum_{i=0}^{n-1} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

hat means predicted

Methods of Estimation

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
- Random Subsampling
  - Repeated holdout
- Cross Validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n

Holdout

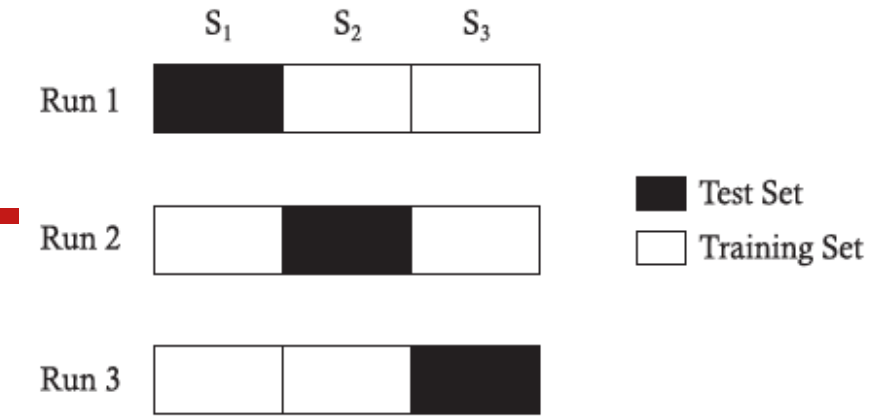
- The holdout method reserves a certain amount for **testing** and uses the remainder for **training**
- Usually, **one third for testing**, the rest for training.
- Typical quantities are 60%-40%, 66%-34%, 70%-30%.
- For small or “unbalanced” datasets, **samples might not be representative**
  - For instance, few or none instances of some classes
- Stratified sample
  - **Balancing the data**
  - Make sure that each class is represented with approximately equal proportions in both subsets

Repeated Holdout

- Holdout estimate can be made more reliable by **repeating the process with different subsamples**
- **Repeated holdout method**
  - In each iteration, a certain proportion is **randomly selected for training** (possibly with stratification)
  - The accuracy scores on the different iterations are **averaged** to yield an overall accuracy

- Still not optimum: the different test sets overlap

Cross Validation



- Avoids overlapping test sets
  - **First step:** data is split into k subsets of equal size
  - **Second step:** each subset in turn is used for testing and the remainder for training
- This is called **k-fold cross-validation**
- Often the subsets are stratified before cross-validation is performed
- The **accuracy scores** are **averaged** to yield an overall accuracy estimate.
- **Even better:** repeated stratified cross-validation E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

Data Partitioning

Holdout (e.g.70/30)

Train the model for final testing

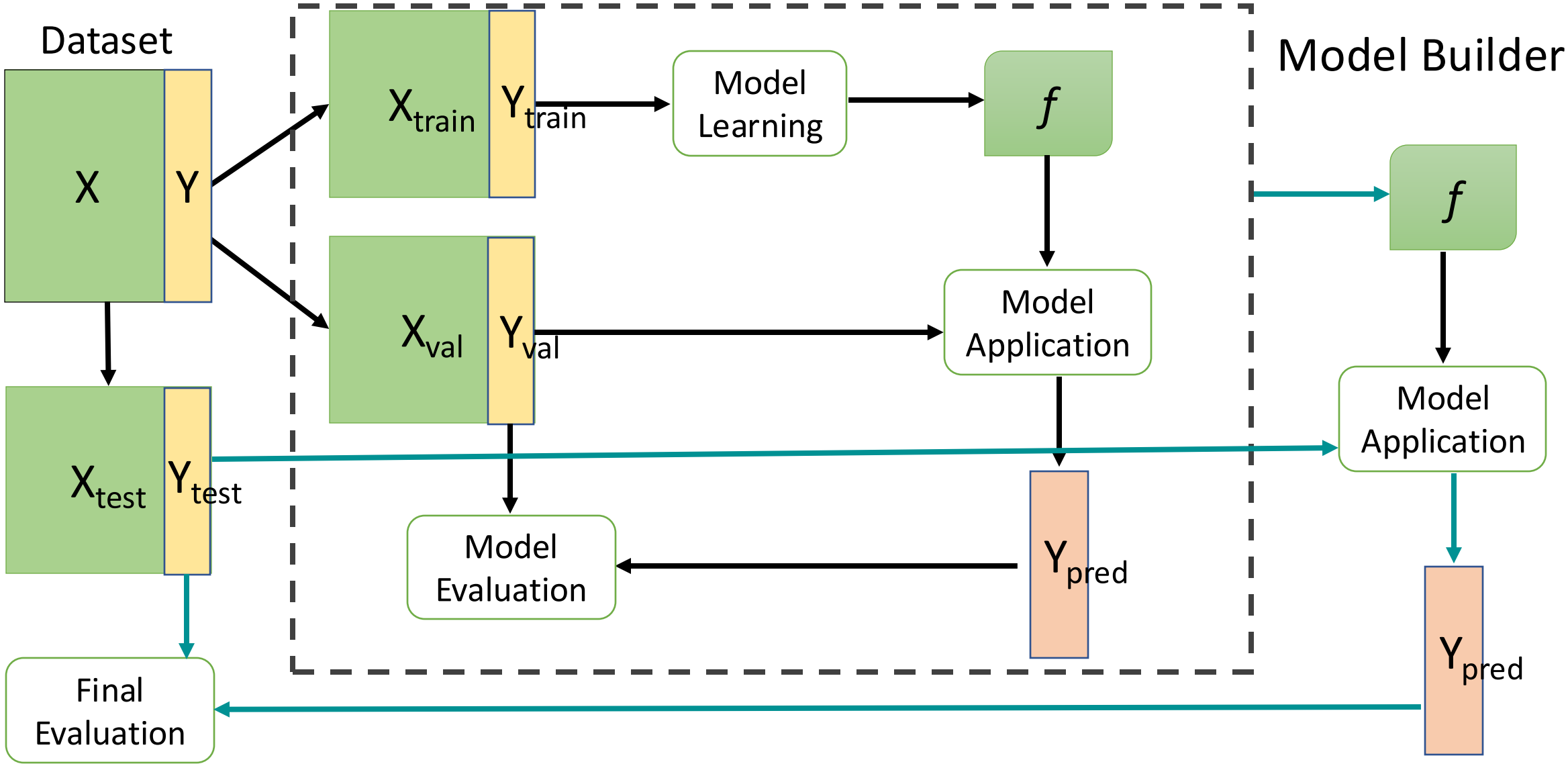
Train the model for parameter selection

Validate the model (early stopping, parameter selection, etc.)

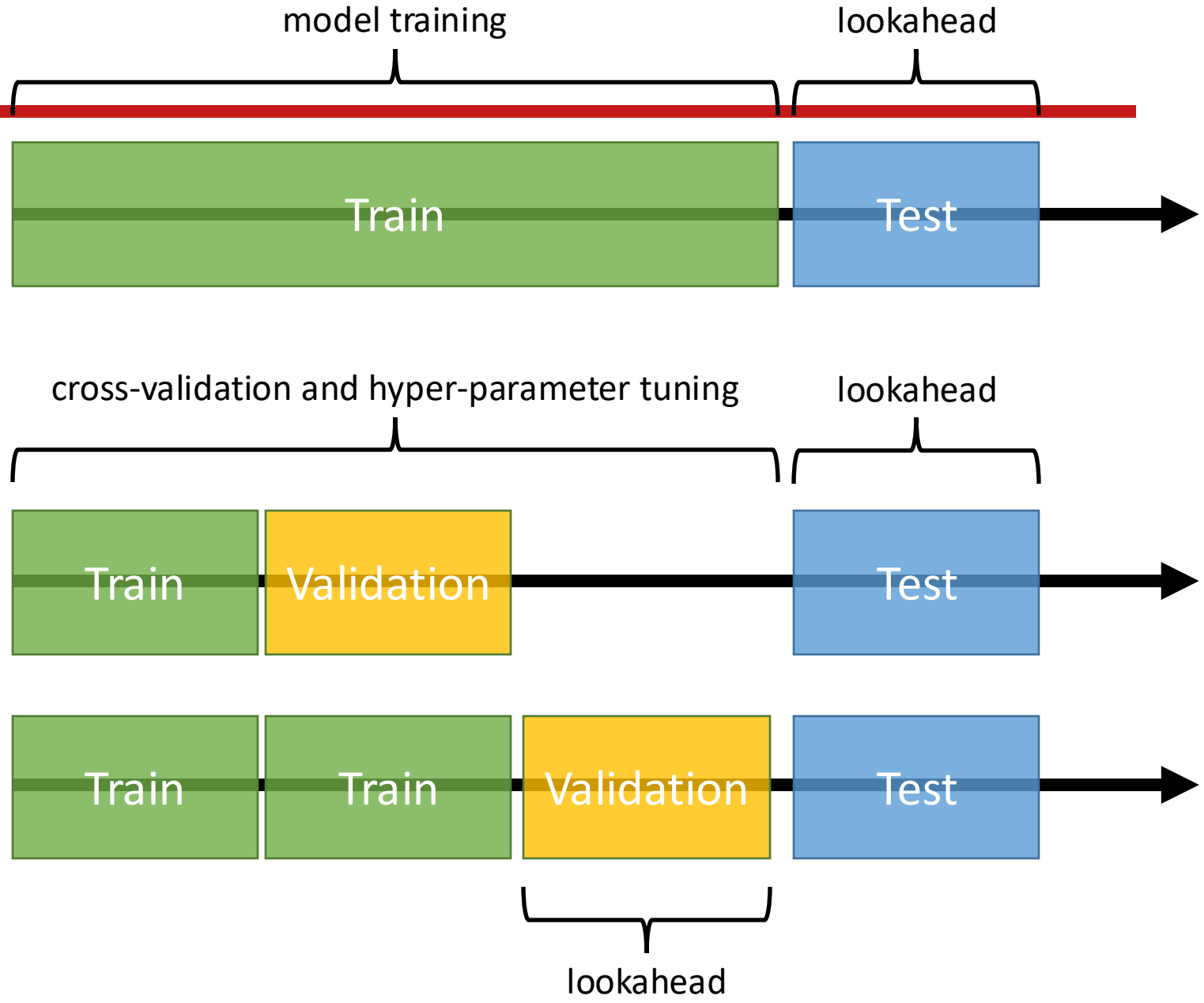
- Test the model
- Compare different models once parameters have been selected

Cross Validation (check potential dataset bias)

Evaluation: Training, Validation, Tests



Cross Validation with Time



Cross Validation with Time

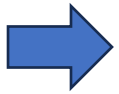
Overfitting and Underfitting

- Underfitting: a model cannot adequately capture the underlying structure of a dataset.
  - Low performance on training set

- Overfitting: a model that corresponds too closely or exactly to a particular dataset of data and may therefore fail to predict unseen data or future observations in a reliable way.
  - High performance on training set
  - Low performance on test set

From Static to Time Series Datasets

id	nbr calls	min calls	call center	churn
a123	40	160	0	No
b456	80	156	3	Yes
c789	10	200	8	No
d321	34	190	2	No
e654	56	1007	1	Yes
f213	13	9	0	No
g435	67	88	0	Yes
h768	186	1234	2	No
i098	49	194	12	No
l012	9	78	11	Yes



id	date	nbr calls	min calls	call center	churn
a123	01-2024	10	40	0	No
a123	02-2024	9	50	0	No
a123	03-2024	11	30	0	No
a123	04-2024	10	40	0	No
…	…	…	…	…	…
b456	01-2024	60	111	1	Yes
b456	02-2024	11	40	0	Yes
b456	03-2024	4	3	0	Yes
b456	04-2024	5	2	2	Yes
…	…	…	…	…	…

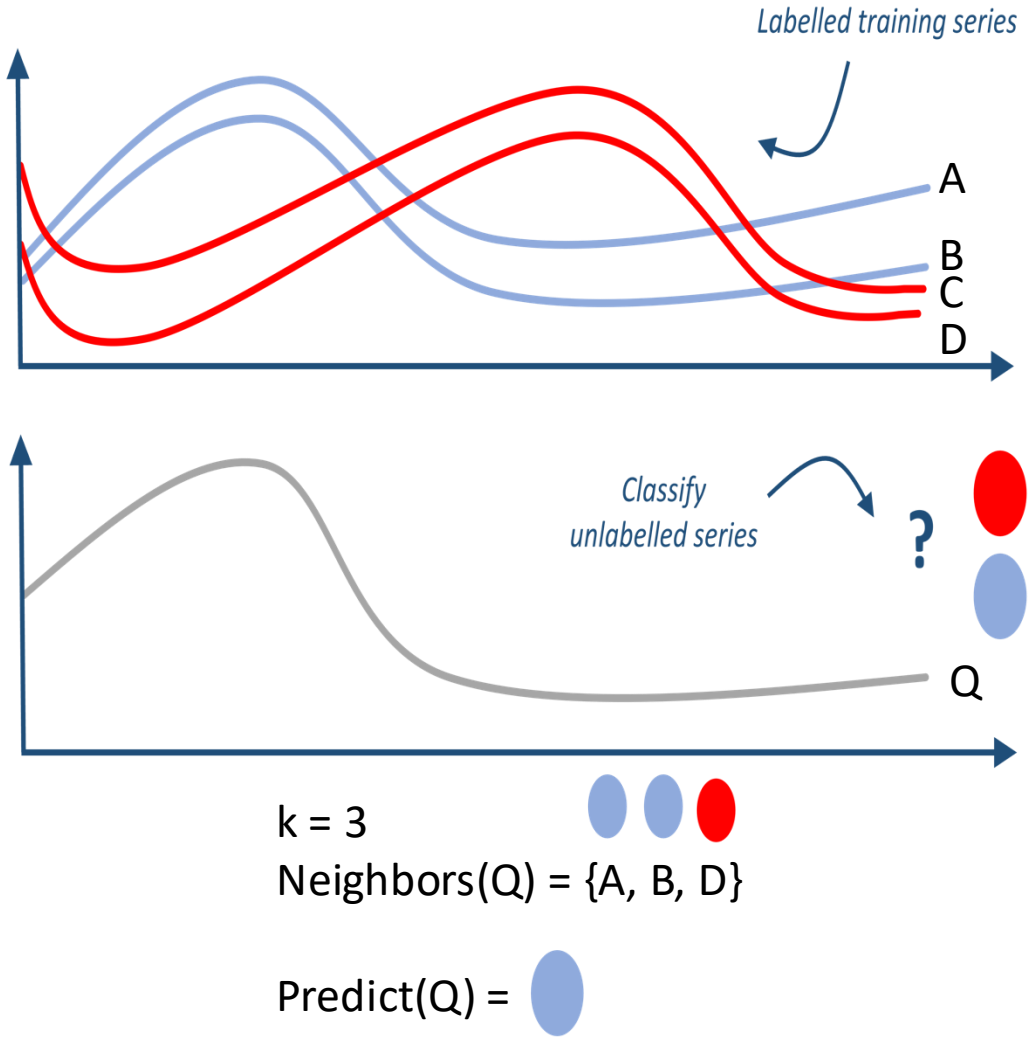
From Static to Time-Aware Datasets

id	time	temp	speed	product	failure	sys temp
a123	t_0	20.1	111	A	0	32.5
a123	t_1	18.6	110	A	0	32.5
a123	t_2	19.4	111	A	1	32.5
a123	t_3	20.4	112	A	0	32.5
a123	t_4	21.5	120	A	0	32.5
b456	t_0	12.7	89	B	0	34.6
b456	t_1	19.8	76	B	0	34.6
b456	t_2	17.4	69	B	0	34.6
b456	t_3	8.4	85	B	1	34.6
b456	t_4	7.9	65	B	1	34.6

Instance-based Models

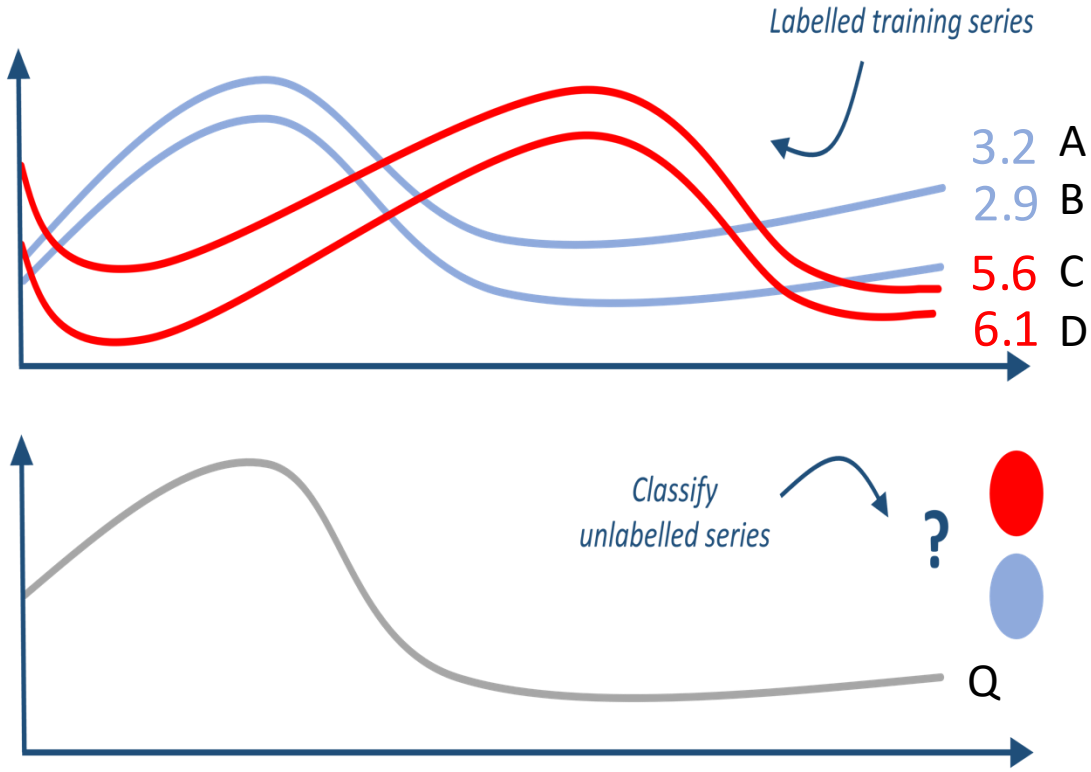
Nearest-Neighbor Classifier (K-NN)

- Basic idea: If it walks like a duck, quacks like a duck, then it is probably a duck.
- Given a set of training records, and a test record:
  1. Compute the distances from the test to the training records.
  2. Identify the k "nearest" records.
  3. Use class labels of nearest neighbors to determine the class label of test record (e.g., by taking majority vote).



Nearest-Neighbor Classifier (K-NN)

- Basic idea: If it walks like a duck, quacks like a duck, then it is probably a duck.
- Given a set of training records, and a test record:
  1. Compute the distances from the test to the training records.
  2. Identify the k "nearest" records.
  3. Use target values of nearest neighbors to determine the target value of test record (e.g., by making the average).



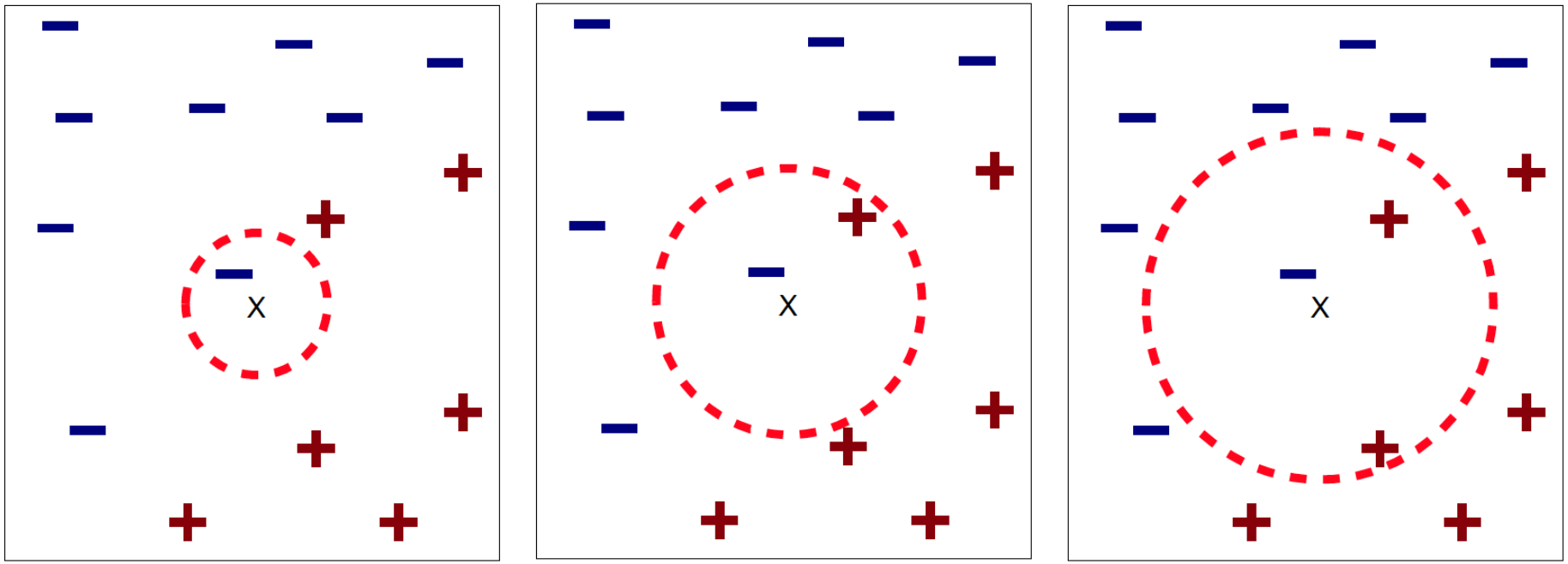
k = 3 3.2 2.9 6.1

Neighbors(Q) = {A, B, D}

Predict(Q) = 3.8

Definition of Nearest Neighbor

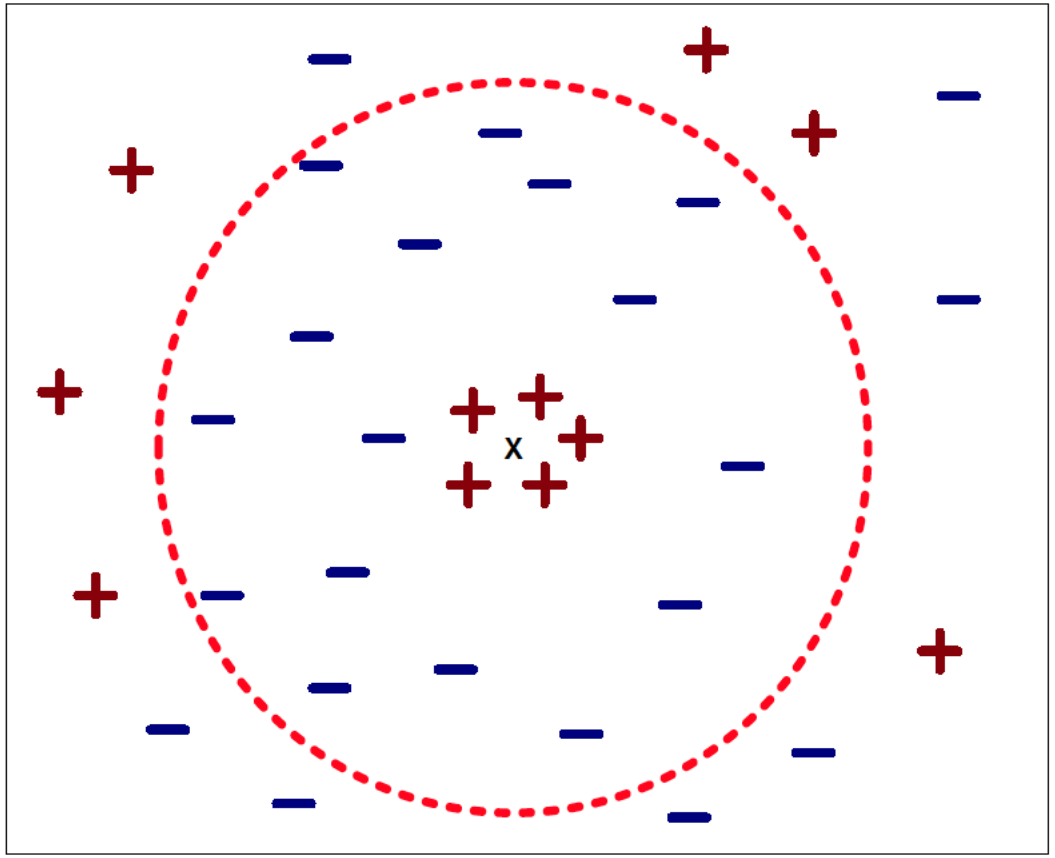
- *K*-nearest neighbors of a record *x* are data points that have the *k* smallest distance to *x*.



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Choosing the Value of K

- If k is too small, it is sensitive to noise points and it can lead to overfitting to the noise in the training set.
- If k is too large, the neighborhood may include points from other classes.
- General practice k = sqrt(N) where N is the number of samples in the training dataset.



Nearest Neighbor Prediction

Compute distance between two points:

- Euclidean distance
- DTW

Determine the prediction from nearest neighbors

- Classification: take the majority of class labels among the k nearest neighbors
- Regression: make the mean of the target values among the k nearest neighbors

Weight the vote/mean according to distance (e.g. weight factor, $w = 1/d^2$) such that closest instance have a higher impact.

Dimensionality and Scaling Issues

- Problem with distance measure calculus.
- For each test instance q we need to calculate the distances between the time series q and all the instances in the training set X.
- If the dimensionality of the TS considered is large this affects the number of calculus required.
- This effect is amplified by the usage of distances considering warpings like DTW.
- The usage of global constraints and/or time-dependent approximations is recommended

Instance-based Models

- Instead of performing explicit generalization, compare new instances with instances seen in training, which have been stored in memory.
- Sometimes called ***memory-based*** learning.
- ***Advantages***
  - Adapt its model to previously unseen data by storing a new instance or throwing an old instance away.
- ***Disadvantages***
  - Lazy learner: it does not build a model explicitly.
  - Classifying unknown records is relatively expensive: in the worst case, given *n* training items, the complexity of classifying a single instance is *O(n)*.

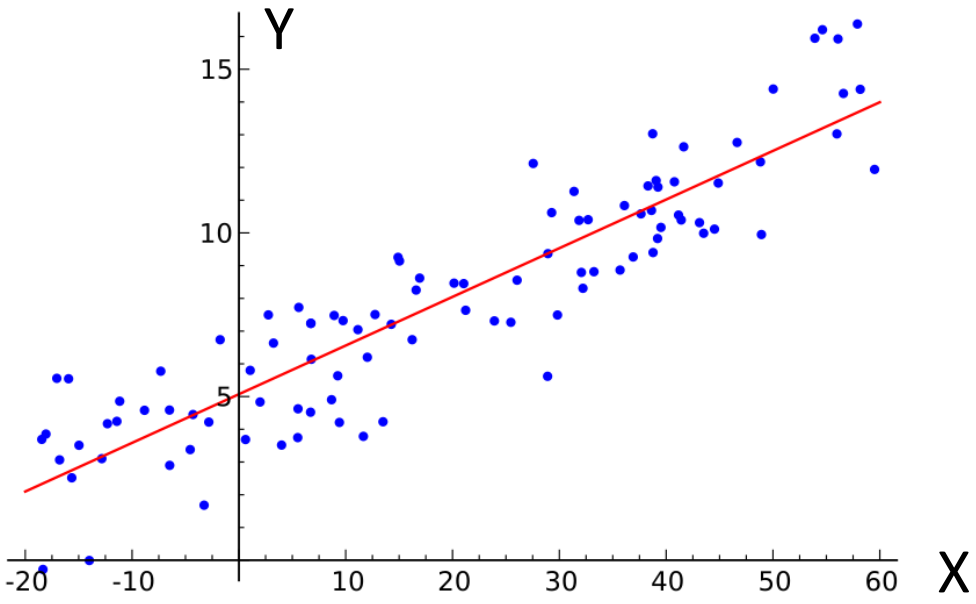
Time Series Representation and kNN

- If raw data of the TS is used or the TS is approximated with PAA or SAX then kNN can be used with both Euclidean distance and DTW
- If the other approximations are used or global structural features are extracted, then kNN can only be used with distances that do not make usage of time or warpings.

Linear Models

Linear Regression

- **Linear regression** is a linear approach to modeling the relationship between a *dependent variable Y* and one or more *independent* (explanatory) variables *X*.
- The case of *one* explanatory variable is called **simple linear regression**.
- For *more than one explanatory variable*, the process is called **multiple linear regression**.
- For *multiple correlated dependent variables*, the process is called **multivariate linear regression**.



Simple Linear Regression

Linear Model: $Y = mX + b \qquad Y = \beta_1 X + \beta_0$

Dependent Variable — Independent Variable

Slope — Intercept (bias)

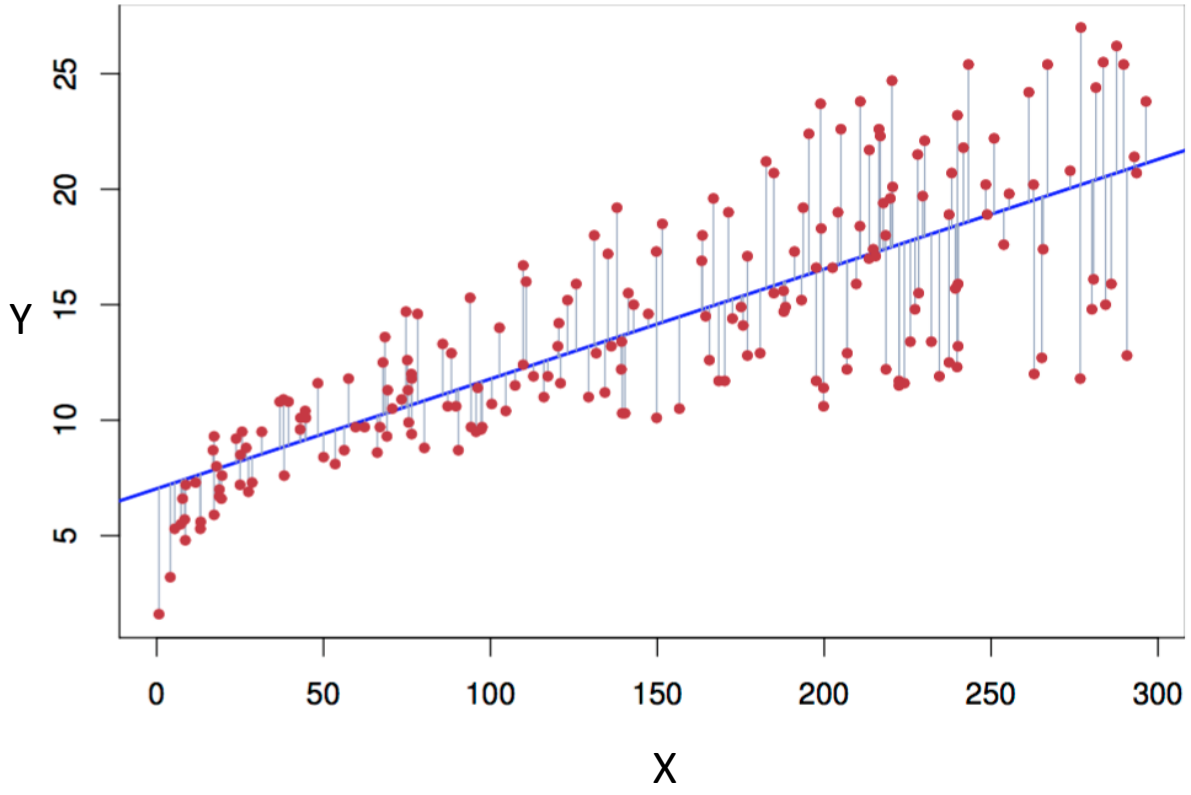
- Such linear relationship may not hold exactly for all the population.
- We call the deviations from *Y* errors or residuals, i.e., $y_i - f(x_i)$
- The objective of linear regression is to find values for the parameters *m* and *b* which would provide the "best fit" for the observed points.

Least Square Method

- A standard approach for doing this is to apply the **method of least squares** which attempts to find the parameters *m*, *b* that minimizes the sum of squared error.
- SSE = $\sum_i (y_i - f(x_i))^2 = \sum_i (y_i - mx_i - b)^2$
- also known as the **residual sum of squares**.
- The LSM finds *m, b* by setting to zero the first partial derivative of the above function w.r.t. *m* and *b* which are therefore calculated as follows:
- $m = (n \sum(xy) - \sum x \sum y) / (n \sum(x^2) - (\sum x)^2)$
- $b = (\sum y - m \sum x) / n$
- LSM can be extended to multiple linear regression.
- An alternative to find *m, b*, typically adopted in case of multivariate regression is the Gradient Descent method.

Least Square Method

- Blue line shows the least square fit. Lines from red points to the regression line illustrate the residuals.
- For any other choice of slope *m* or intercept *b* the SSE between that line and the observed data would be larger than the SSE of the blue line.



Linear Regression and Regularization

- Simple $\beta_0 + \beta_1 x \ - y$
- Multiple $\beta_0 + \sum_i (y_i - \beta_i x_i)^2$
- Ridge $\beta_0 + \sum_i (y_i - \beta_i x_i)^2 \ + \lambda \sum_j {\beta_j}^2$
- Lasso $\beta_0 + \sum_i (y_i - \beta_i x_i)^2 \ + \lambda \sum_j |\beta_j|$

regularization

- Lasso regularization: performs both variable selection, i.e., minimizes the number of coefficient different from zeros, and regularization to enhance the prediction accuracy and interpretability.
- Ridge regularization: mitigates the problem of multicollinearity which commonly occurs in models with large numbers of parameters.

Logistic Regression

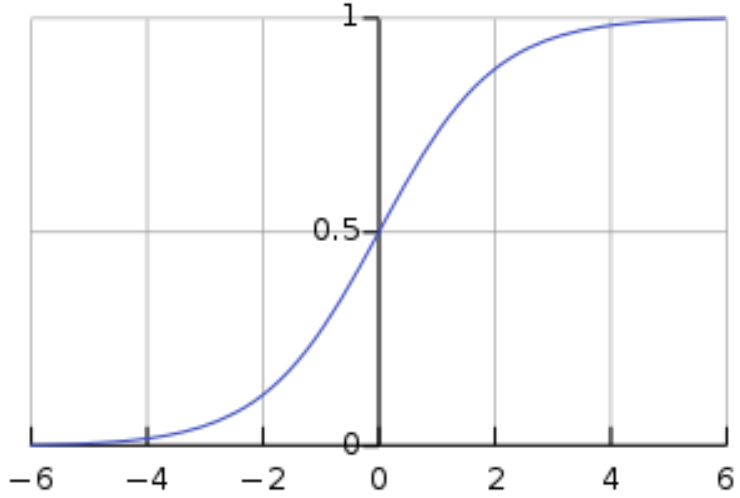
Y = logit(p) = log(p/(1-p))

- If instead of predicting a continuous value, we want the predict a categorical one using a linear model we can consider to predict the probability with which that categorical value will be verified.
- Let *p* be the probability that a certain event will happen.
- *p/(1-p)* are the **odds** of this event and *log(p/(1-1p))* are the **log-odds**.
- If we estimate the log-odds using a linear model, we have.

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X$$

$$\Leftrightarrow \frac{p}{1-p} = e^{\beta_0 + \beta_1 X}$$

$$\Leftrightarrow p = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$



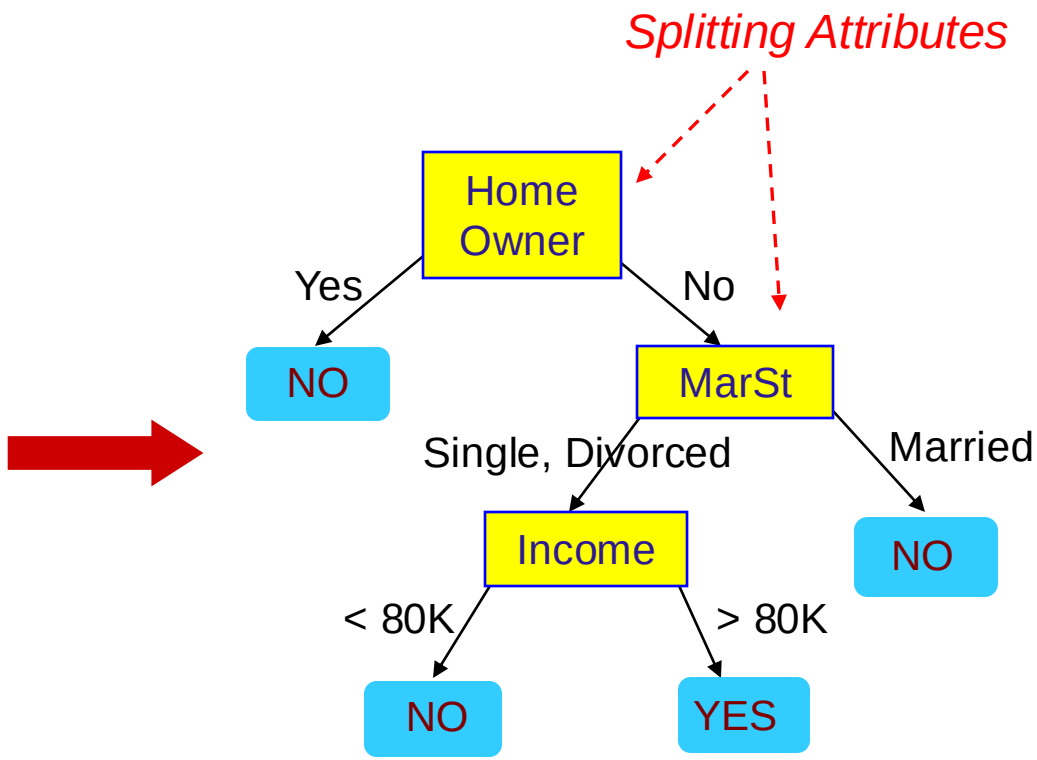
Tree-based Models

Example of a Decision Tree

Consider the problem of predicting whether a loan borrower will repay the loan or default on the loan payments.

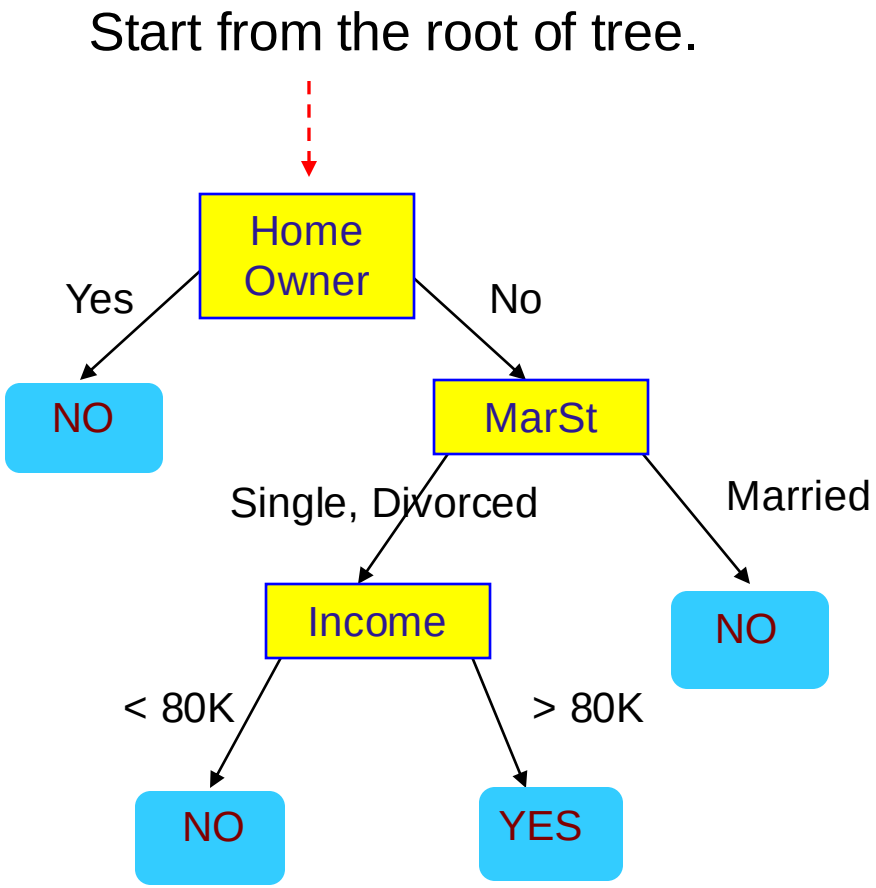
	categorical	categorical	continuous	class
ID	**Home Owner**	**Marital Status**	**Annual Income**	**Defaulted Borrower**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

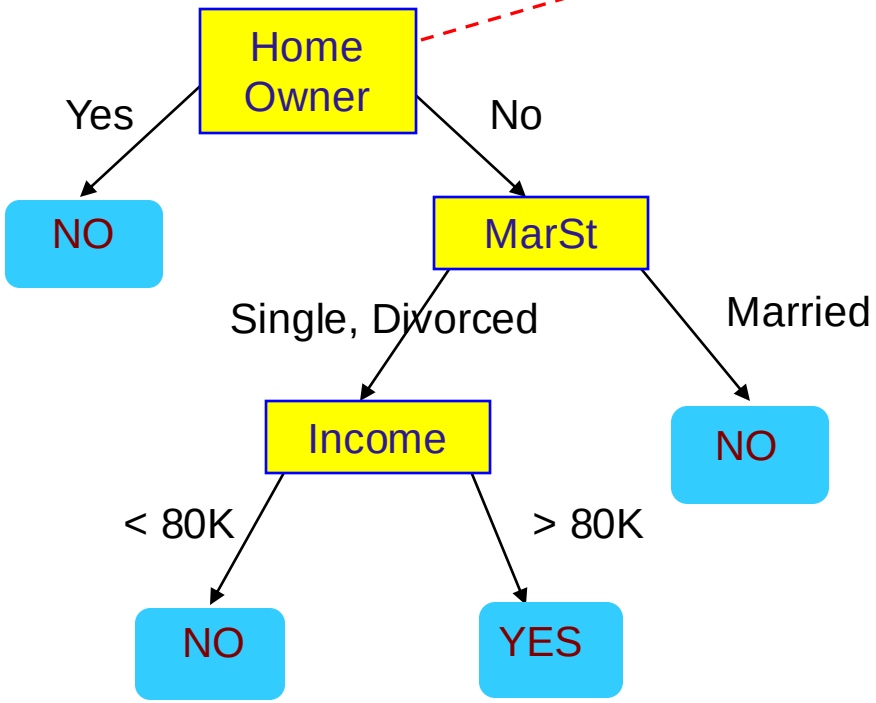
Apply Model to Test Data



Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	**?**

Apply Model to Test Data



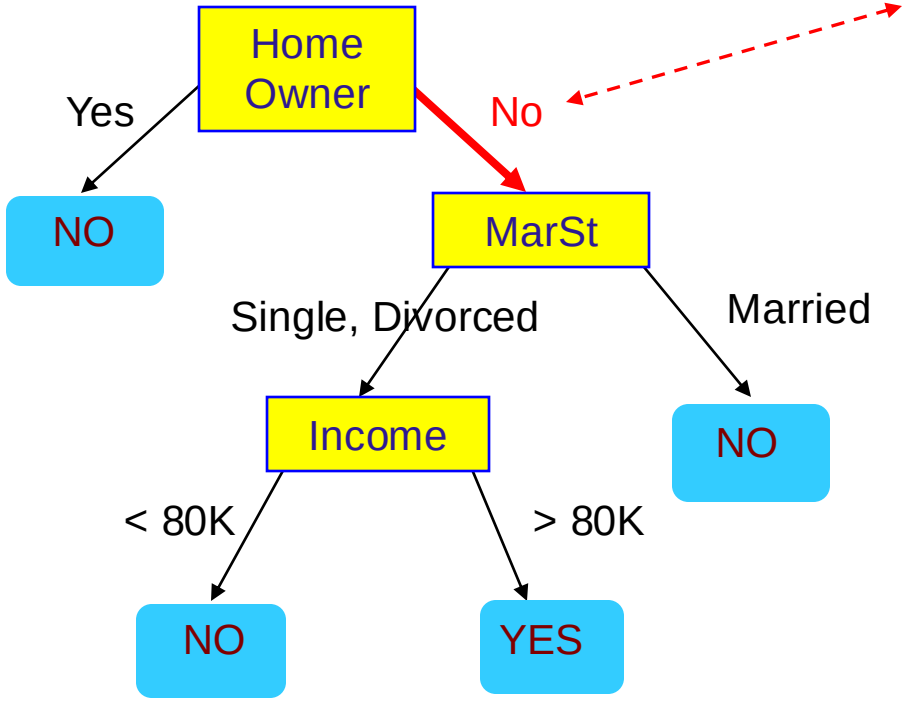
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

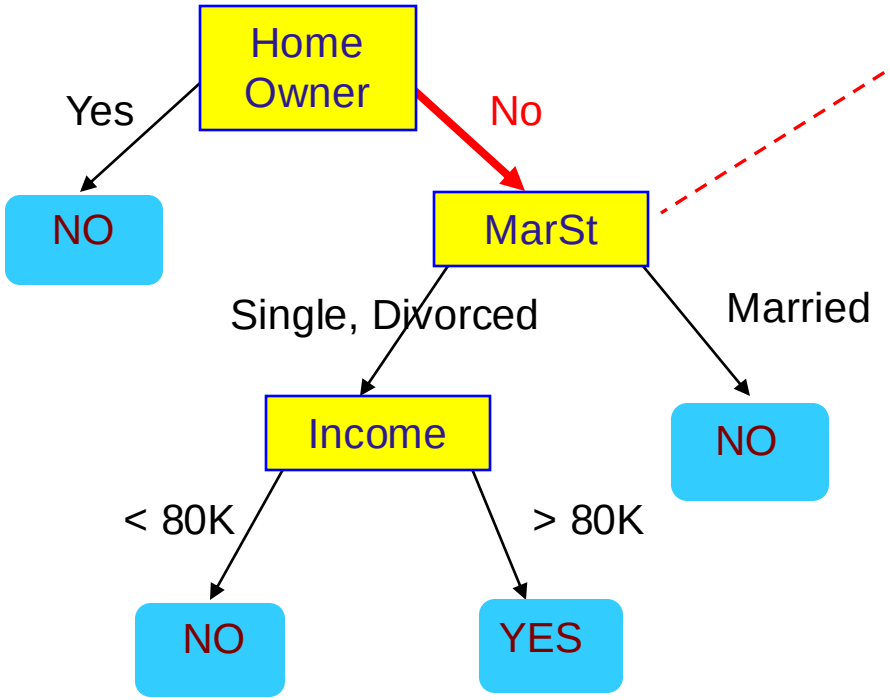
Apply Model to Test Data

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data



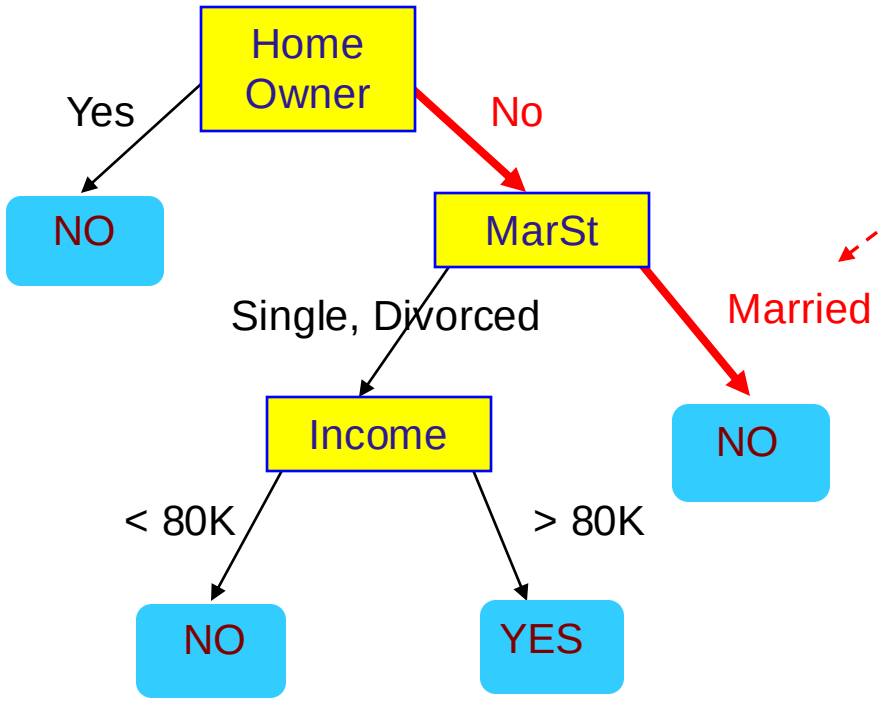
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

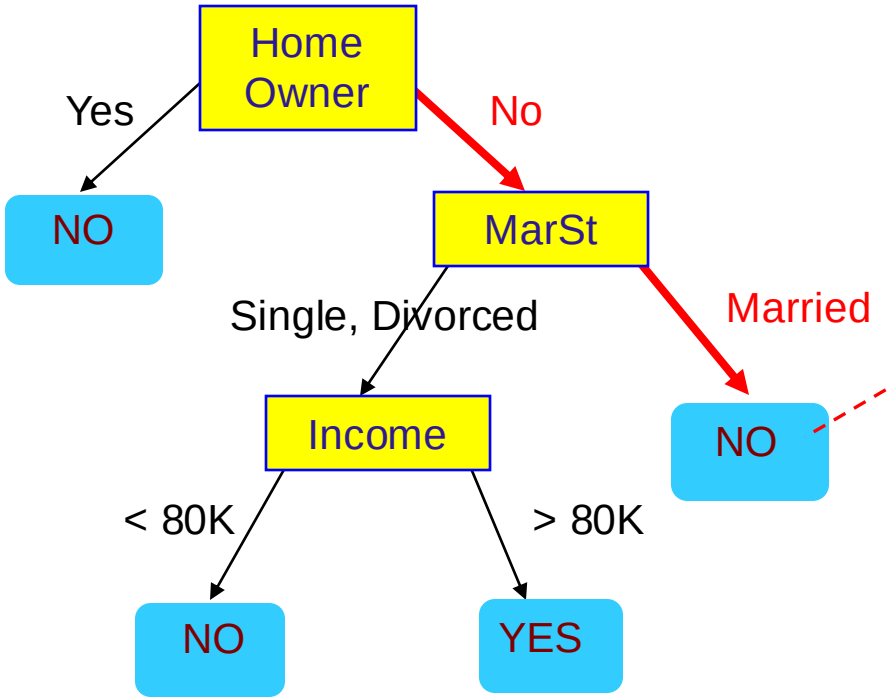
Apply Model to Test Data

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data



Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Assign Defaulted to "No"

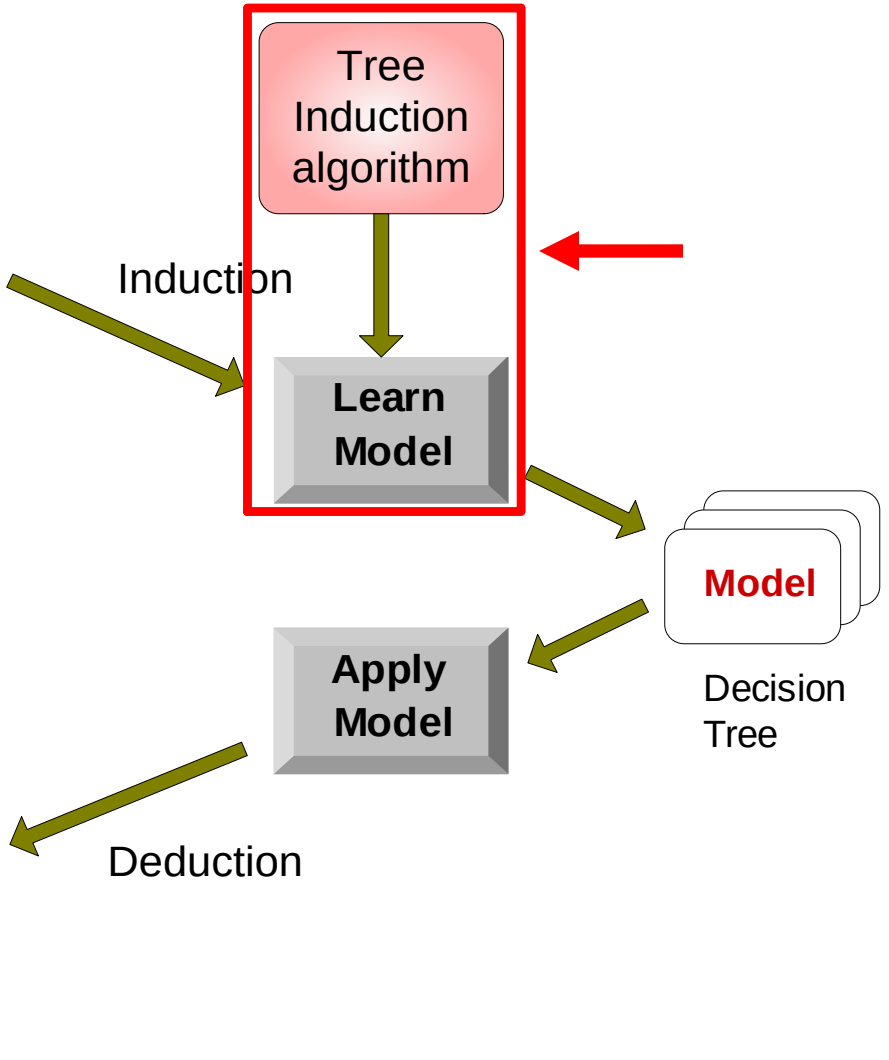
Decision Tree Classification Task

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 Set

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	?

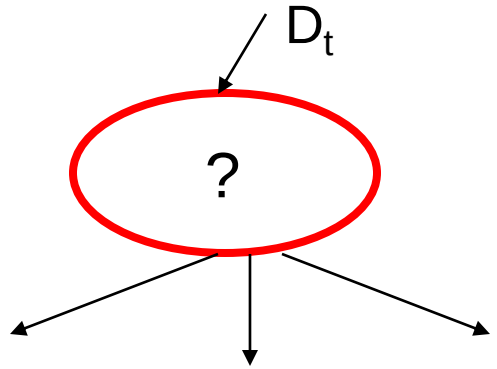
Test Set



Decision Tree Induction

- Let D_t be the set of training records that reach a node t
- General Procedure:
  - If D_t contains records that belong the same class or less than n records, then *t* is a **leaf node** predicting as value those equal to the majority of the labels in D_t or equal to the mean value of the values in D_t
  - Otherwise, use an attribute test to **split** the data into smaller subsets such that the children nodes contains records which are more homogeneous/pure w.r.t. those in D_t and recursively apply the procedure to each subset.
- Existing algorithms: Hunt's, CART, ID3, C4.5

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Finding the Best Split

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - M is the weighted impurity of children
3. Choose the attribute test condition that produces the highest Information Gain as G = P - M

Measures of Node Impurity

- Gini Index

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

- Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

- R2

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

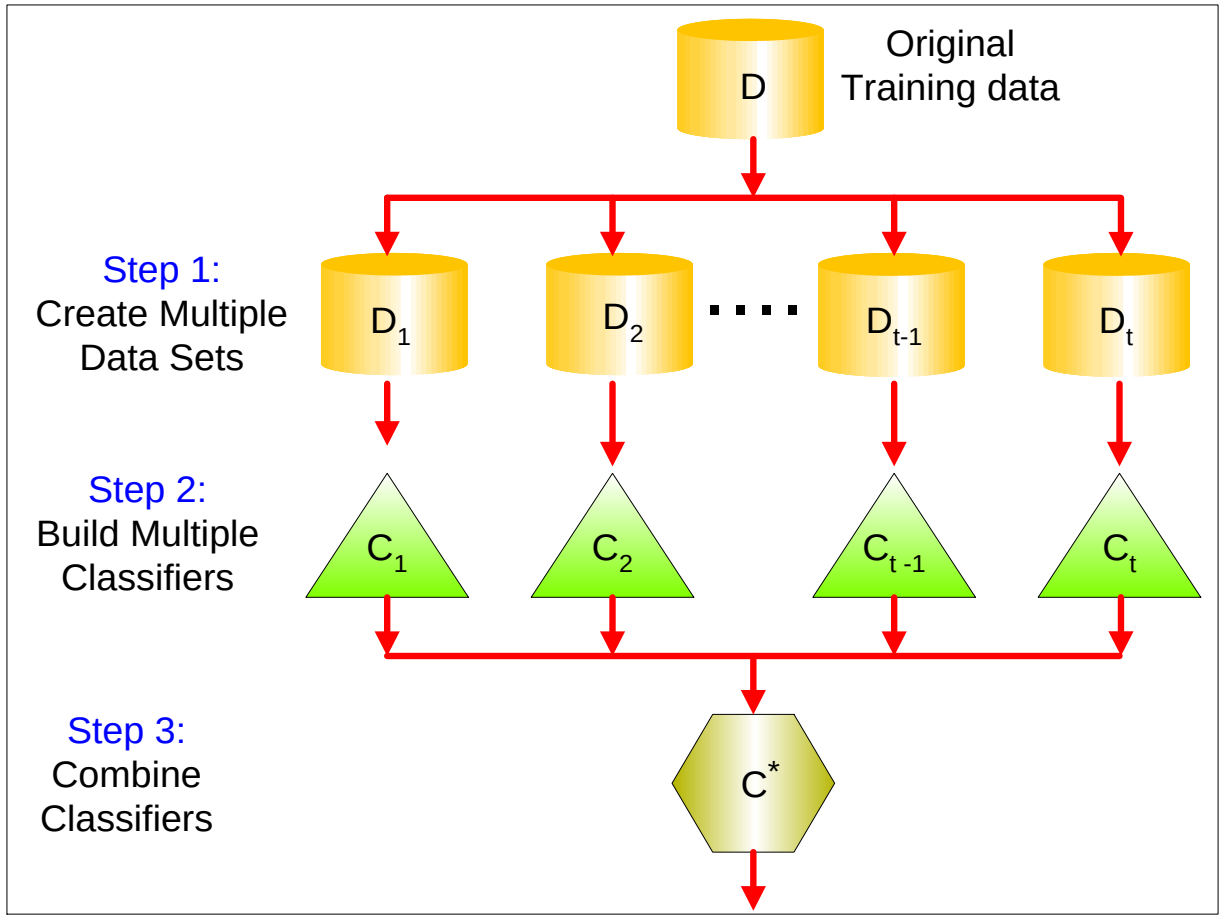
- MSE

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n_{\mathrm{samples}}} \sum_{i=0}^{n_{\mathrm{samples}}-1} (y_i - \hat{y}_i)^2$$

Trees Ensemble Models

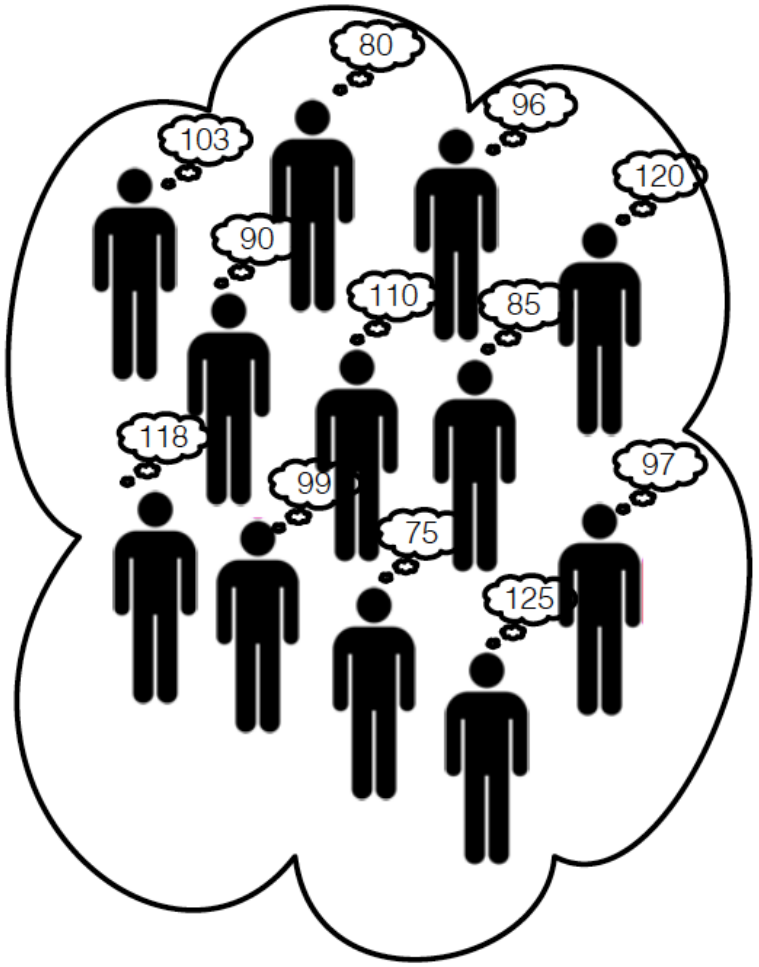
Ensemble Methods

- Improves the accuracy by **aggregating the predictions** of multiple classifiers.
- Construct a set of **base models** from the training data.
- Make the prediction of test records by combining the predictions made by multiple base models.
  - Majority for classification
  - Average for regression



Wisdom of Crowd

- The collective knowledge of a diverse and independent set of individuals harnessed by voting typically exceeds the knowledge of any single individual and can be.
- Ensemble Models exploit *Wisdom of crowd* ideas for specific tasks
- By combining models predictions, the aim is to combine independent and diverse classifiers.



Types of Ensemble Methods

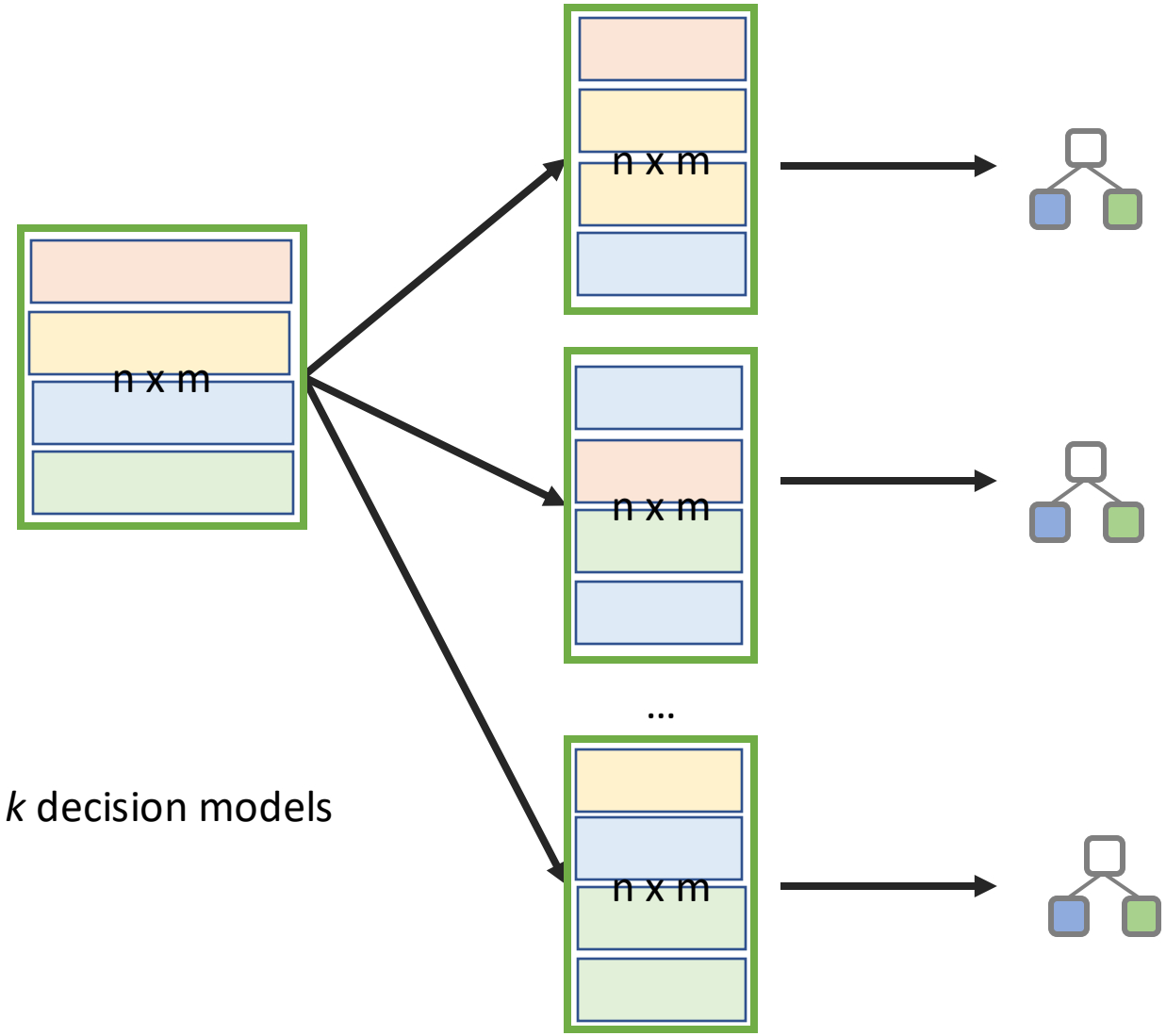
- Manipulate data distribution
  - Bagging
  - Boosting

- Manipulate input features
  - Random Forests

Bootstrap

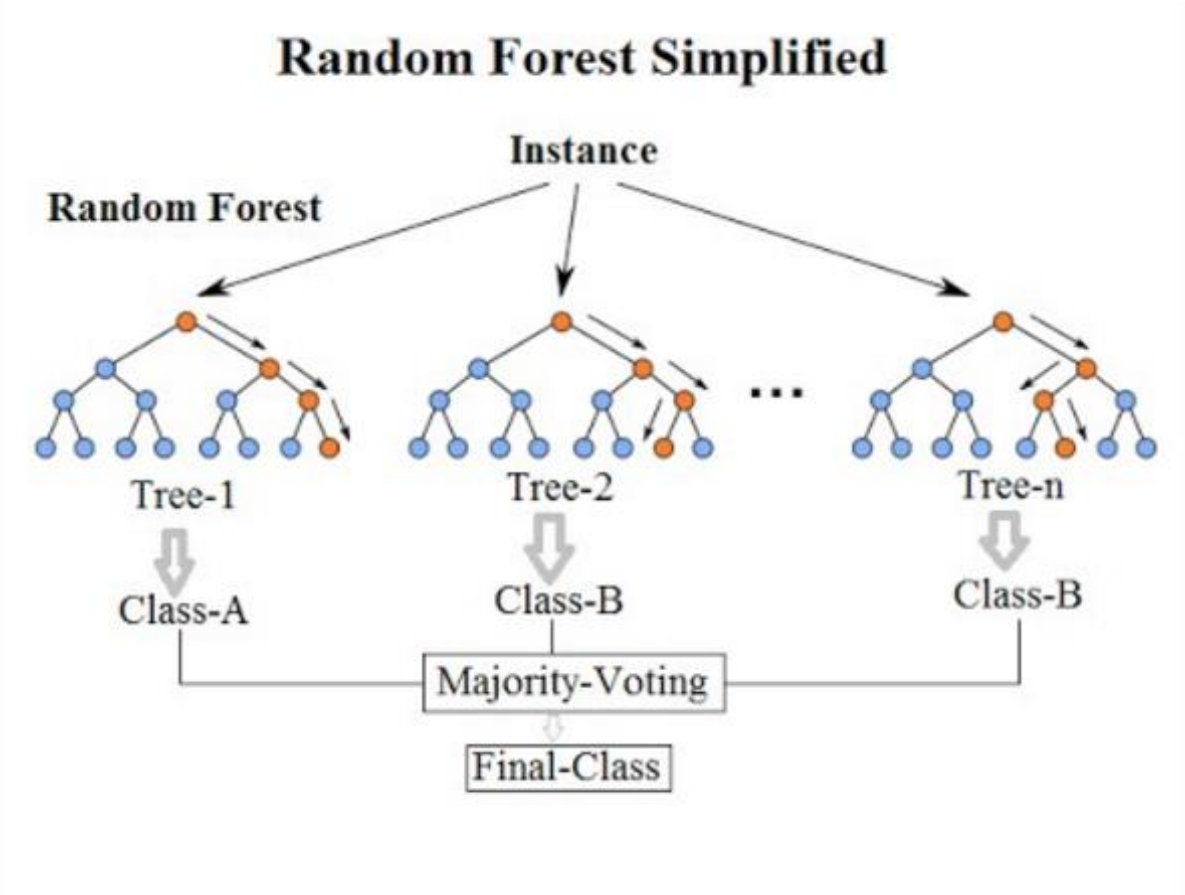
- It is a re-sampling statistical technique with re-entry to approximate the sample distribution of a statistic.
- We consider a dataset with *n* records $X = \{x_1, \ldots, x_n\}$.
- From *X* we sample *m* datasets of constant size *n*, say $\{X_1^*, \ldots, X_m^*\}$.
- In each bootstrap extraction X_i^*, each record in the original dataset has *1/n* probability of being extracted and can be extracted more than once or zero.

Bagging (a.k.a. Bootstrap AGGregatING)



Random Forest

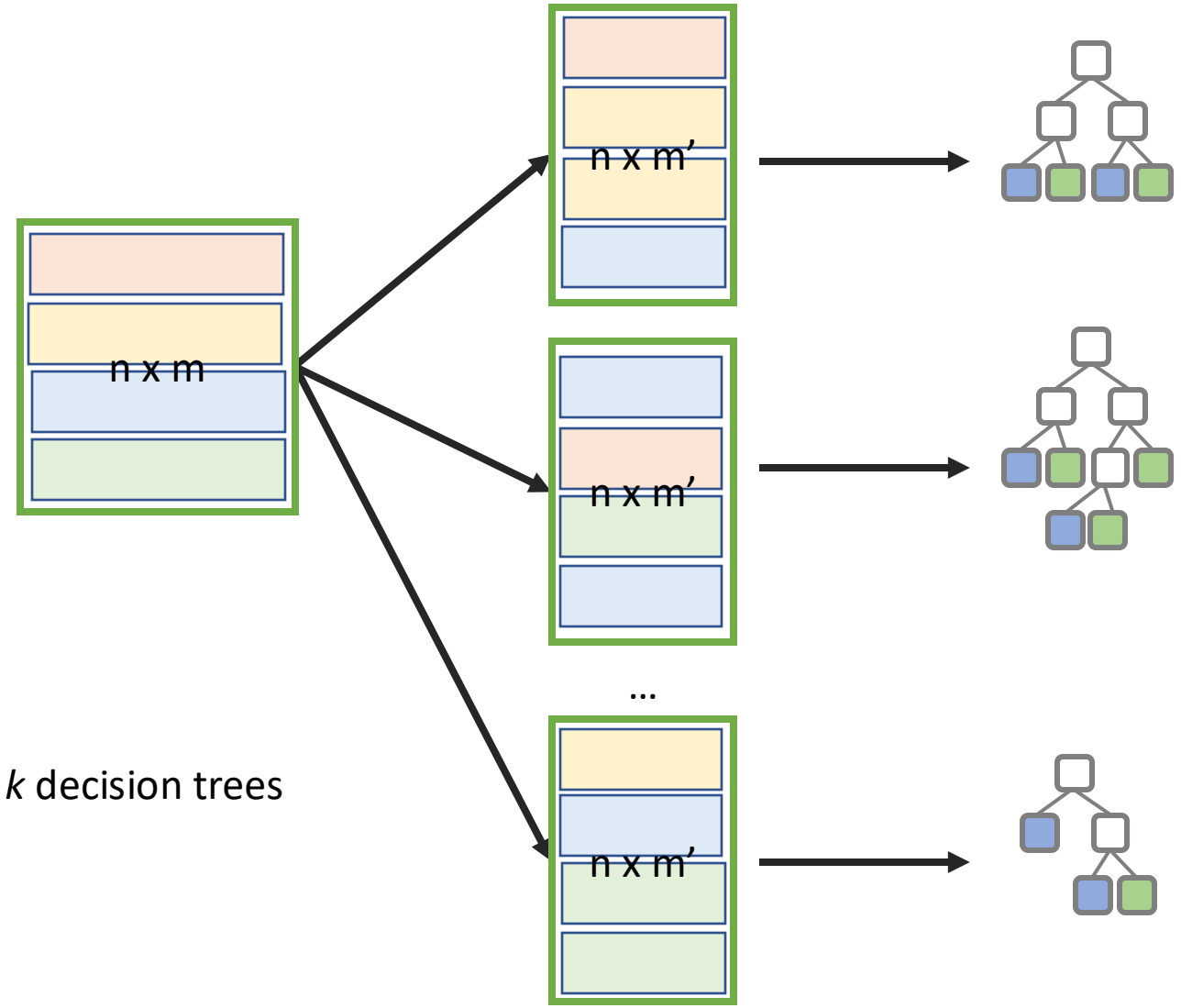
- Is a class of ensemble methods specifically designed for **decision trees**.
- It combines the predictions made by multiple decision trees and outputs the target that is the mode/mean of the output of individual trees.



Random Forest

- Each decision tree is built on a **bootstrap sample** based on the values of an **independent** set of random vectors.
  - Unlike AdaBost (see next slides), the random vector are generated from a fixed probability distribution.
  - Bagging using decision trees is a special case of random forests where randomness is injected into the model-building process.
- Each decision tree is built on ***m'* randomly chosen attributes** from the *m* available attributes
  - $m' \sim \sqrt{m}$, or
  - $m' \sim \log(m)$

Random Forest



Random Forest - Advantages

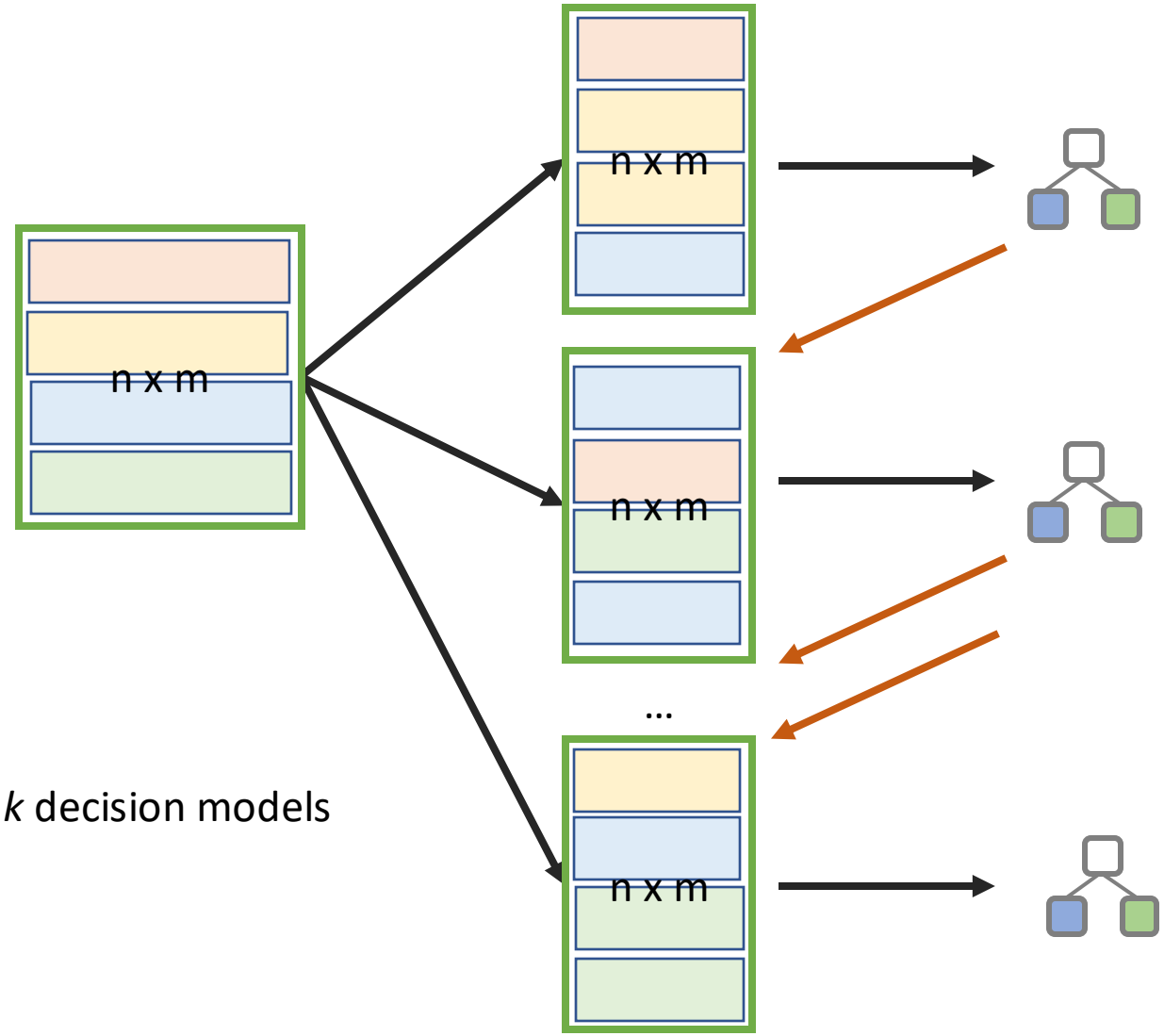
- It is one of the most accurate learning algorithms available. For many data sets, it produces a high accurate classifier.
- It runs efficiently on large databases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records.

- Initially, all the records are assigned equal weights.
- Unlike bagging, weights may change at the end of each boosting round.

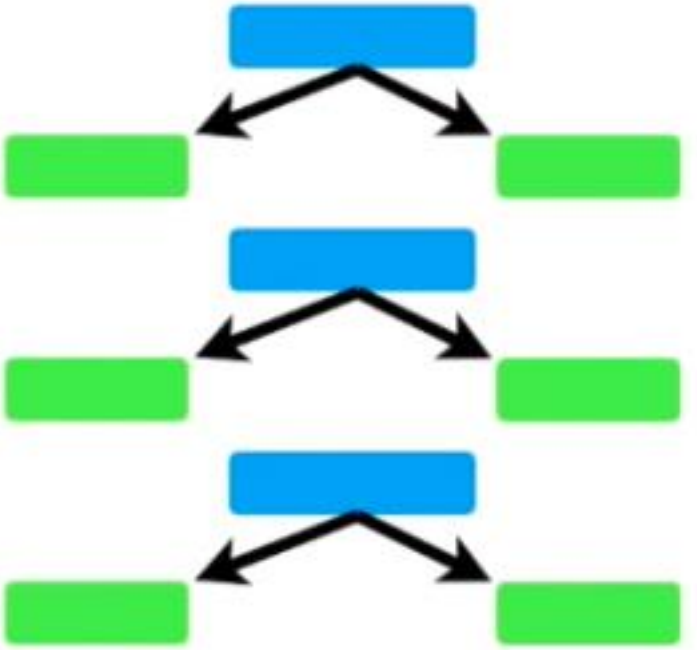
Boosting



k decision models

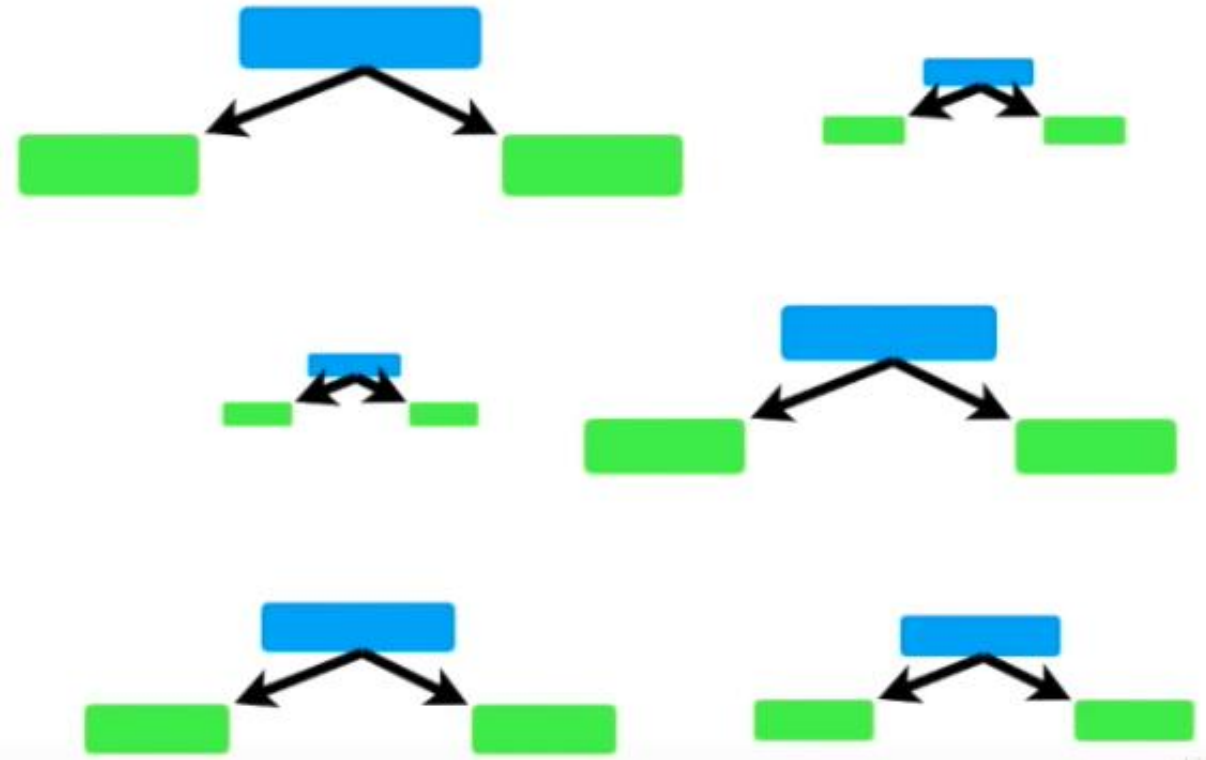
AdaBoost

- AdaBoost uses **stumps**, i.e., decision tree with only one node and two leaves.
- Thus, it uses a forest of stumps.
- Stumps are not good at making accurate predictions.
- Stumps are *weak* learners.



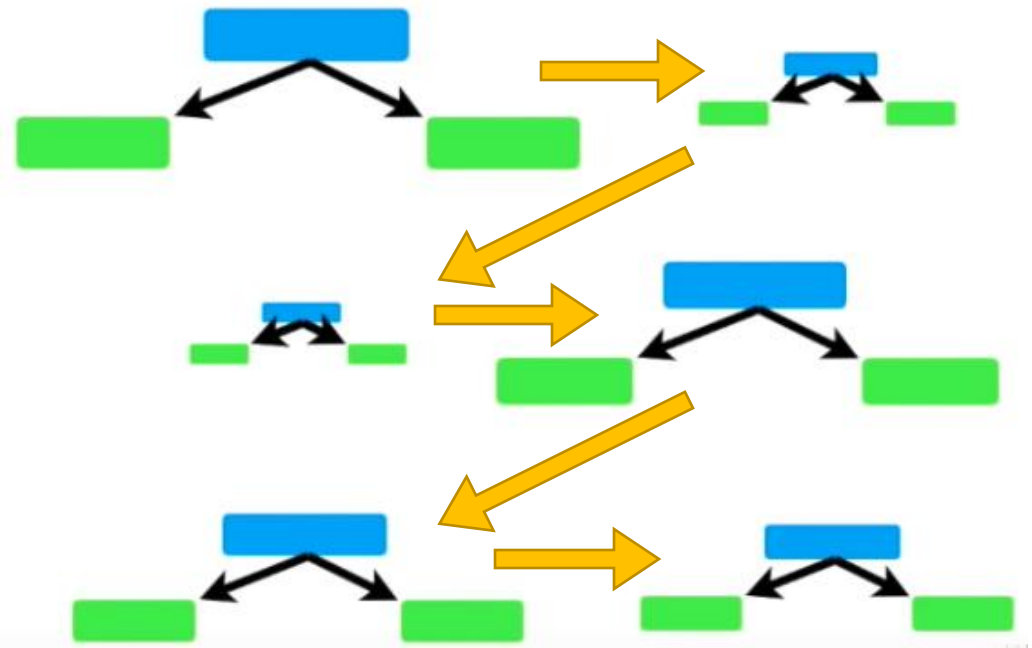
AdaBoost

- In forest of stumps some stumps have more importance in the final prediction.



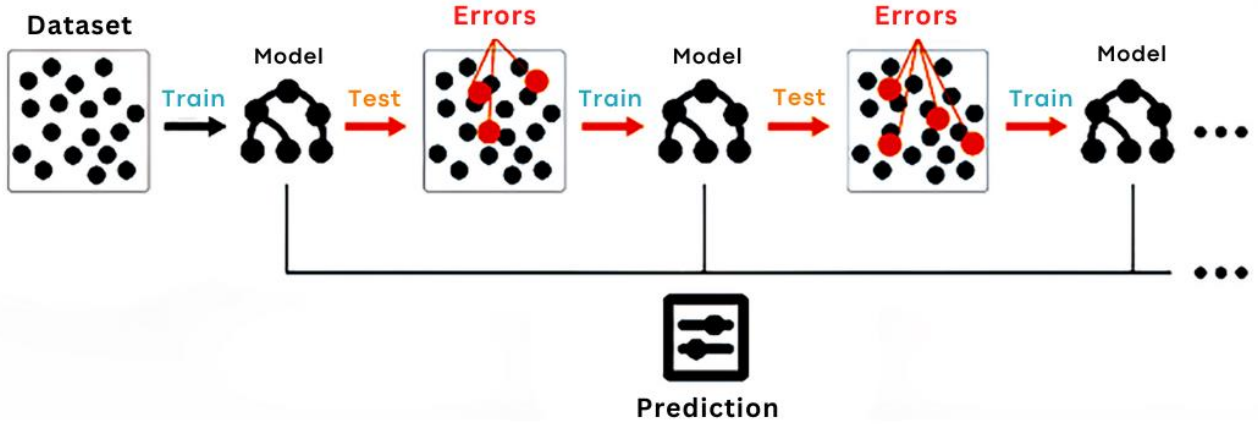
AdaBoost

- Each stump **is not** made independently from the others.
- The error that the first stump makes influences the second stump and so on and so forth.



Gradient Boosting Machines

- First builds a naïve model F_0 considering the entire training set as mode for classification, mean for regression
- Iteration *i*:
  - Applies F_i on the training set and calculate the prediction (pseudo)-residual as the difference between the real value and the predicted one.
  - Then train a decision tree regressor DTR_i to predict the (pseudo)-residual
  - Creates a model as $F_{i+1} = F_i + \text{LearningRate} * DTR_i$
  - Repeats iterations a predefined number of times or until the sum of the (pseudo)-residual is smaller than a certain threshold.



Gradient Boosting Machines Improved

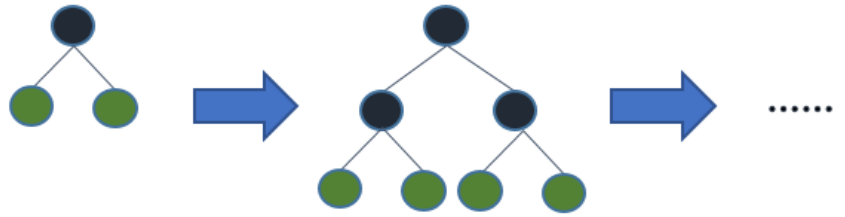
- XGBoost (Extreme Gradient Boost)
- LightGBM (Light Gradient Boosting Machine)
- CatBoost (Categorical Boosting)
- All designed for large complex datasets

XGBoost Characteristics

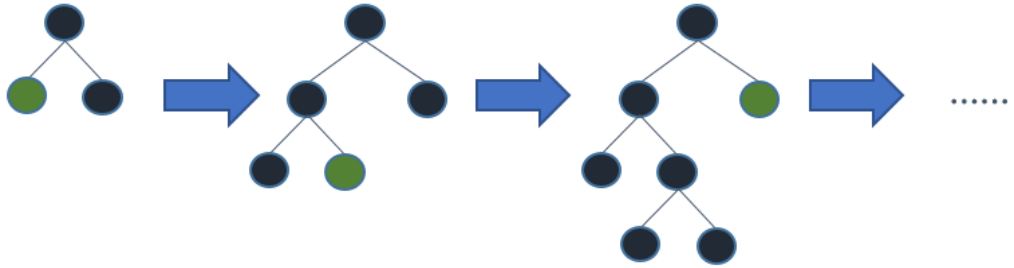
- Gradient Boosting Trees
- Regularization
- Approximate Greedy Algorithm
- Weighted Quantile Sketch
- Sparsity-Aware Split Finding
- Parallel Learning
- Cache-Aware Access
- Blocks for Out-of-Core Computation

XGBoost vs LightGBM vs CatBoost

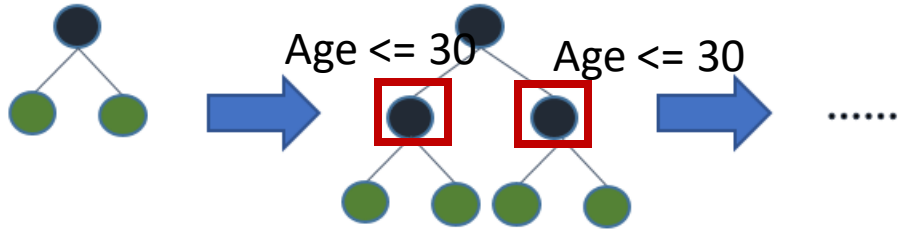
- XGBoost: level-wise (horizontal) growth
- XGBoost: novel splitting function



- LightGBM: out leaf-wise (vertical) growth
- LightGBM significantly faster than XGBoost with almost equivalent performance



- CatBoost: symmetric decision trees
- CatBoost: designed for categorical attributes

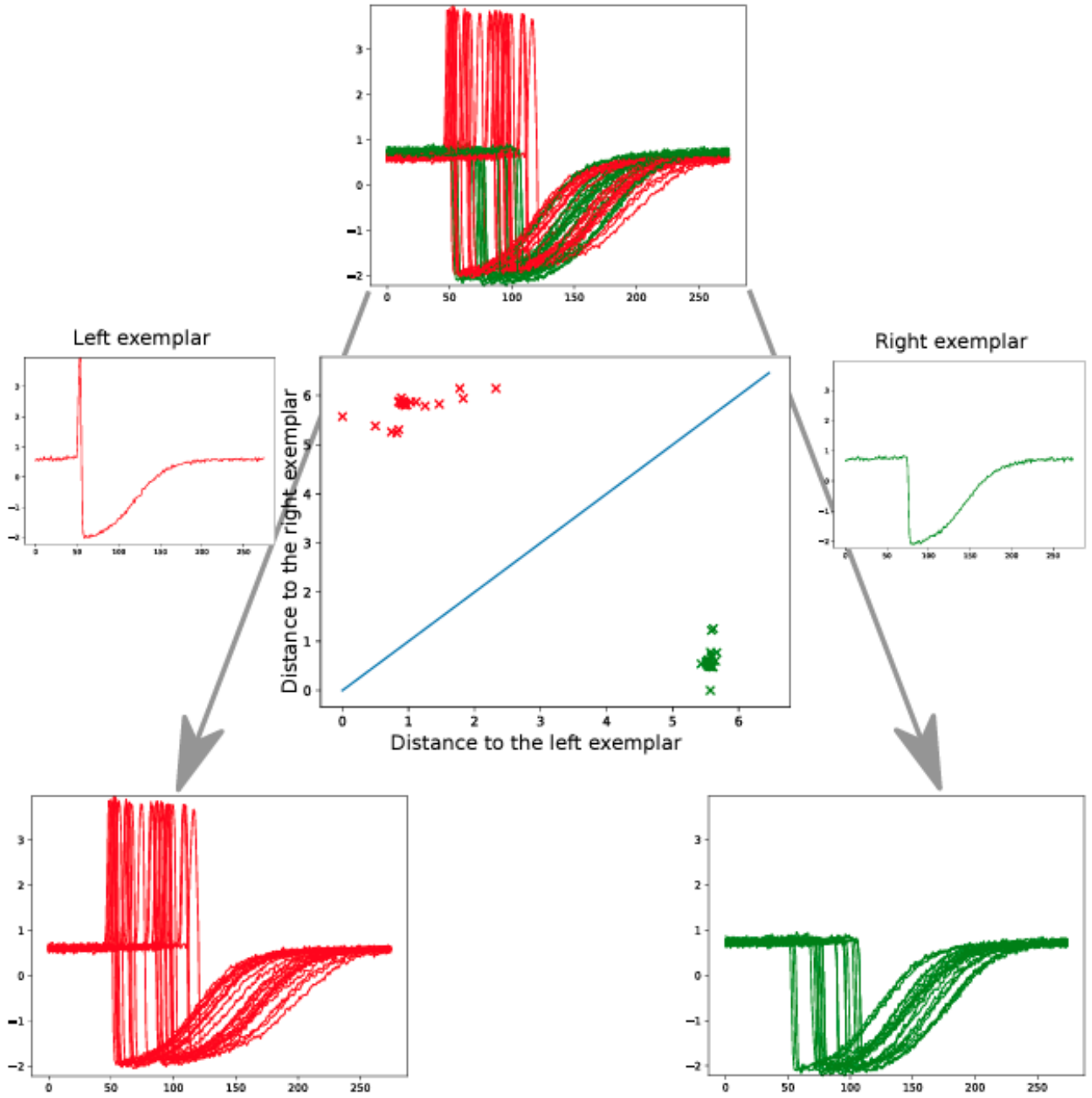


Problems with Tree-based Models in TS

- Decision trees make a split on the value of an attribute.
- Treating the values of a raw TS at each time stamp as belonging to a single attribute independent from the others does not work well on TS as the values in the time stamps are not independent as analyzed by the trees.
- Furthermore, the resulting tree is only limitedly interpretable vanishing the structure of the model.
- Thus, it is advisable to use tree-based models on TS only after having represented them in forms of independent features such as using global structural features or time-independent approximations.

Proximity Forest

- A Proximity Forest is an ensemble of *k* Proximity Trees.
- Each branch of an internal node has an associated exemplar TS.
- A test TS follows the branch corresponding to the exemplar to which it is closest according to a parameterized similarity measure.
- *R* sets of candidate exemplars are randomly selected for each split.
- Among the *R* candidate exemplars are selected those with the highest Information Gain (if *R=1* the choice is completely random).

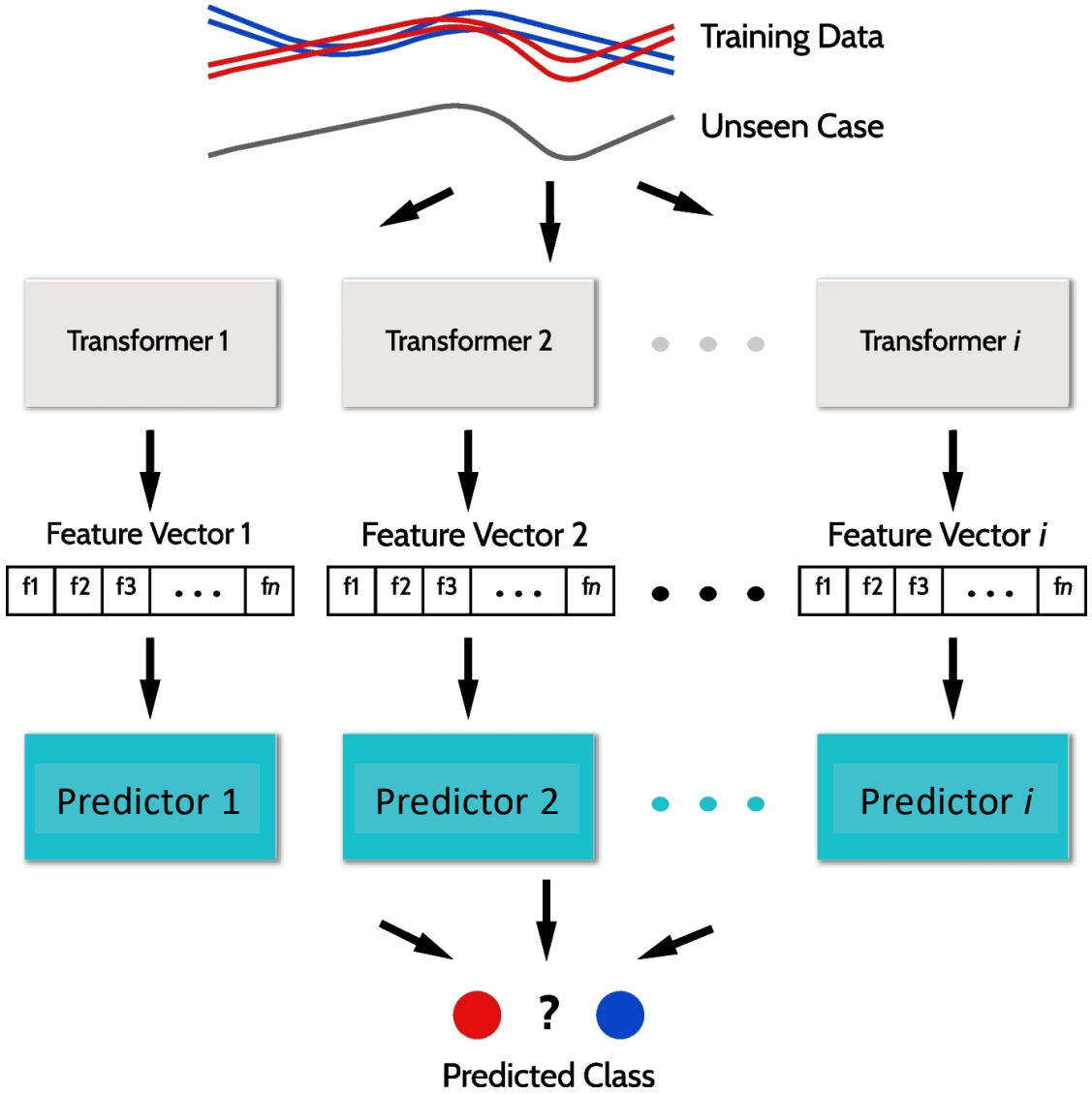


Proximity Forest Distances

1. Euclidean Distance (ED);
2. Dynamic Time Warping using the full window (DTW);
3. Dynamic Time Warping with a restricted warping window (DTW-R);
4. Weighted Dynamic Time Warping (WDTW);
5. Derivative Dynamic Time Warping using the full window (DDTW);
6. Derivative Dynamic Time Warping with a restricted warping window (DDTW-R);
7. Weighted Derivative Dynamic Time Warping (WDDTW);
8. Longest Common Subsequence (LCSS);
9. Edit Distance with Real Penalty (ERP);
10. Time Warp Edit Distance (TWE)

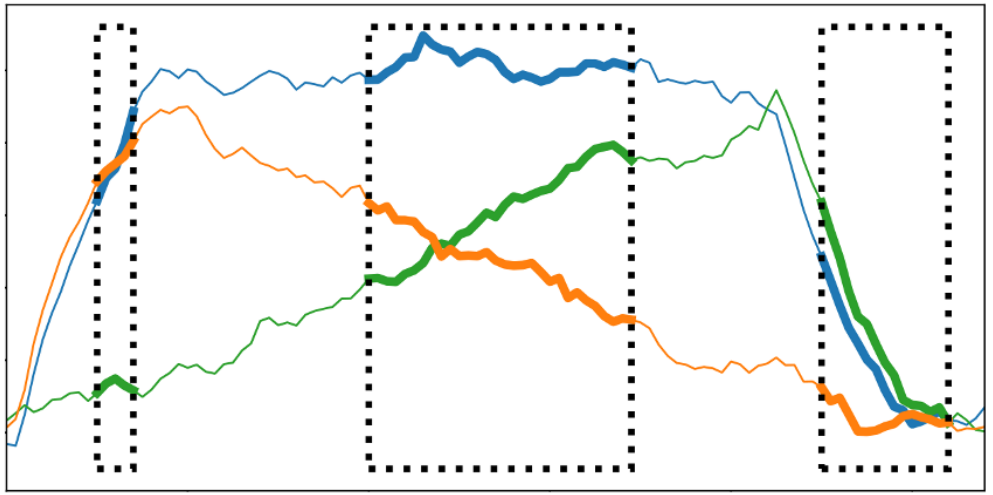
Interval-based Models

Ensemble of Interval-based Models



Interval-based Approaches

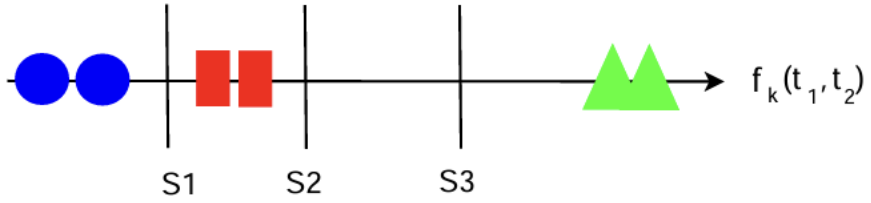
- Interval-based approaches look at phase dependent intervals of the entire TS, calculating summary statistics from selected subsequences to be used in prediction.
- Existing interval-based approaches
  - Time Series Forest (TSF)
  - Random Interval Spectral Ensemble (RISE)
  - Supervised Time Series Forest (STSF)
  - Canonical Interval Forest (CIF)
  - Diverse Representation CIF (DrCIF)

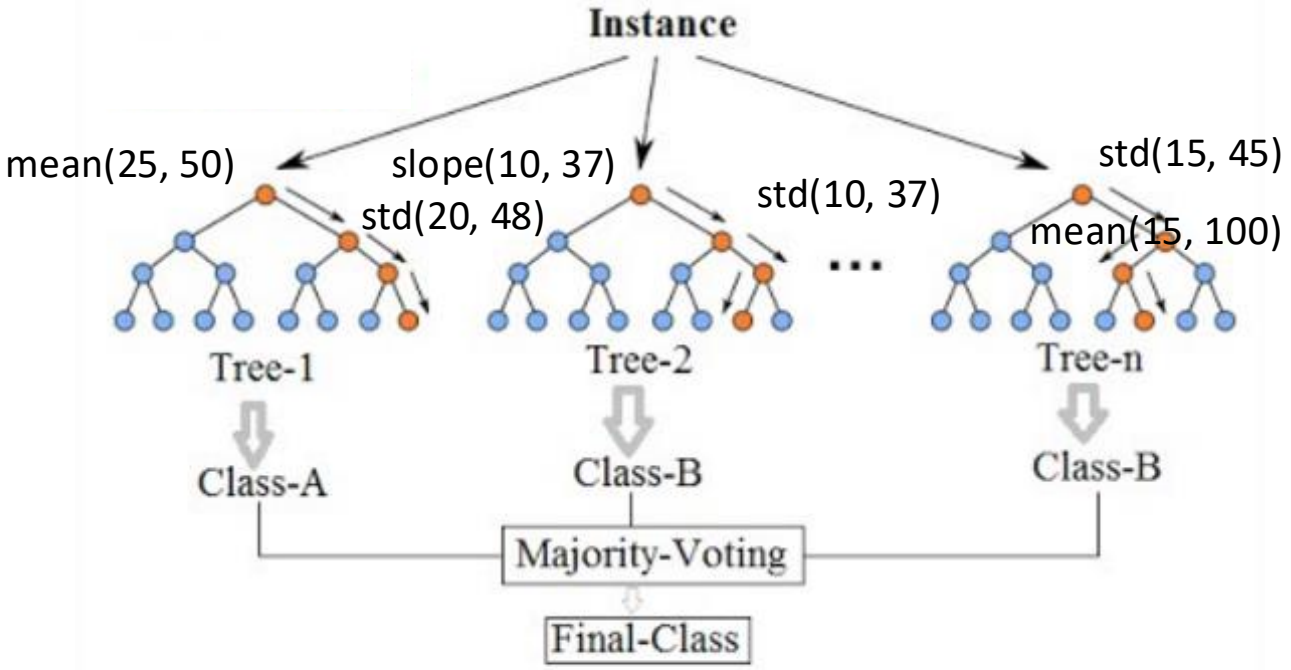


	Interval #1	Interval #2	Interval #3
	mean, std, …, cov	mean, std, …, cov	mean, std, …, cov
TS_1			
TS_2			
TS_3			

Time Series Forest (TSF)

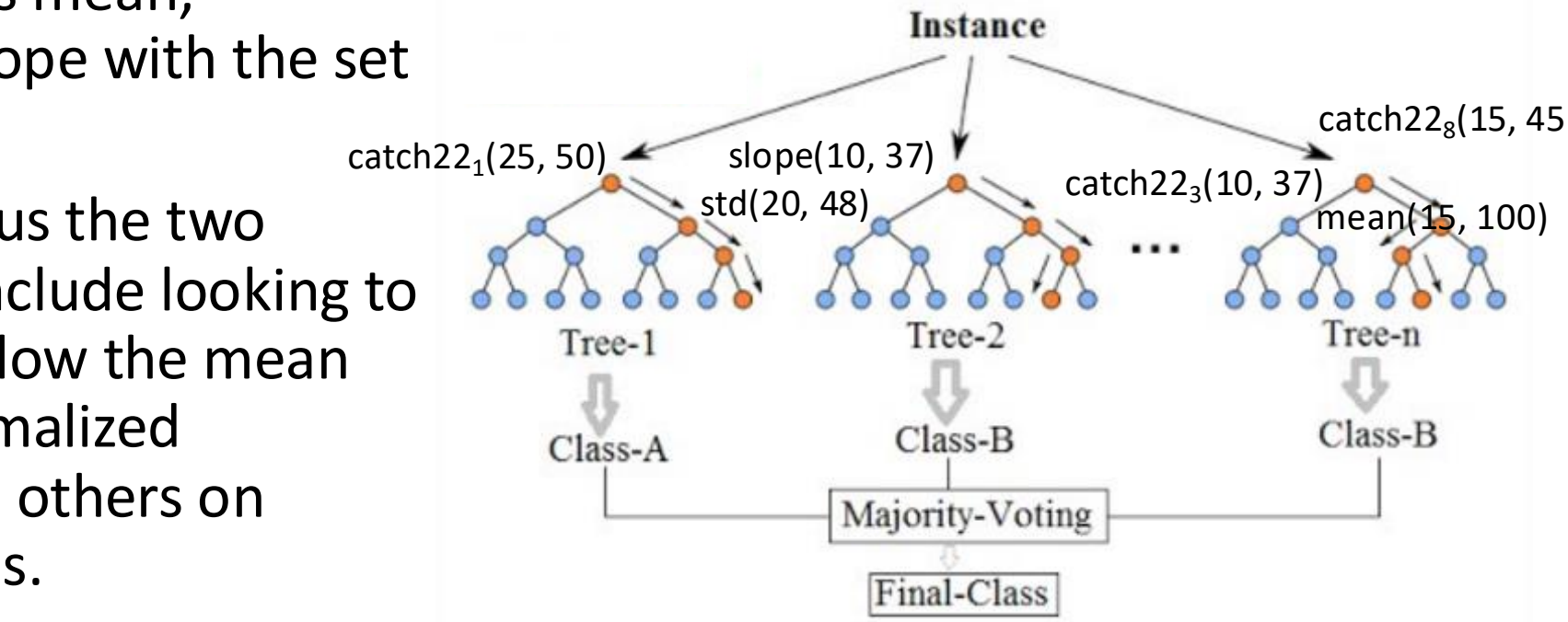
- A TSF is an ensemble TS trees.
- TS trees select the best split by employing Entrance (Entropy and margin distance) gain to identify high-quality splits, i.e.,
- *Entrance = ⊿Entropy + α·Margin*
- Interval features f_k: mean, standard deviation, slope.
- Randomly selected intervals for each TS tree.





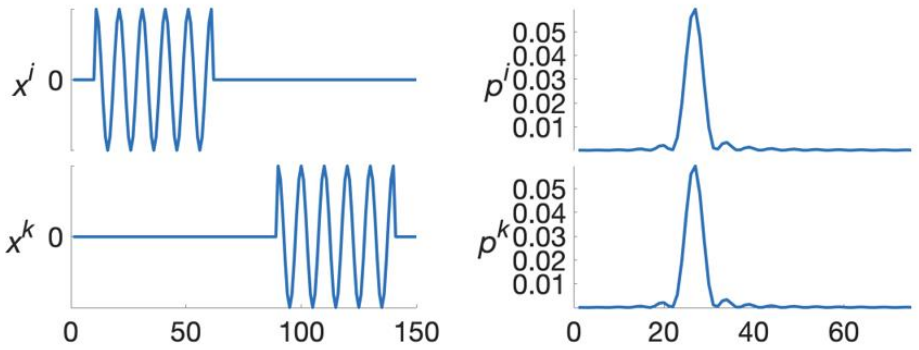
Canonical Interval Forest (CIF)

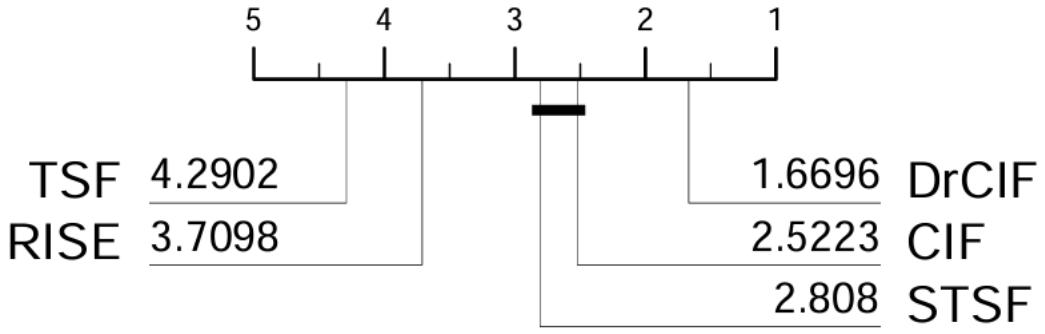
- CIF is an ensemble of TS trees.
- CIF extends TSF by augmenting the set of interval features mean, standard deviation, slope with the set of features catch22.
- To speedup the calculus the two features DN_OutlierInclude looking to outliers above and below the mean are calculated on normalized intervals, while all the others on unnormalized intervals.



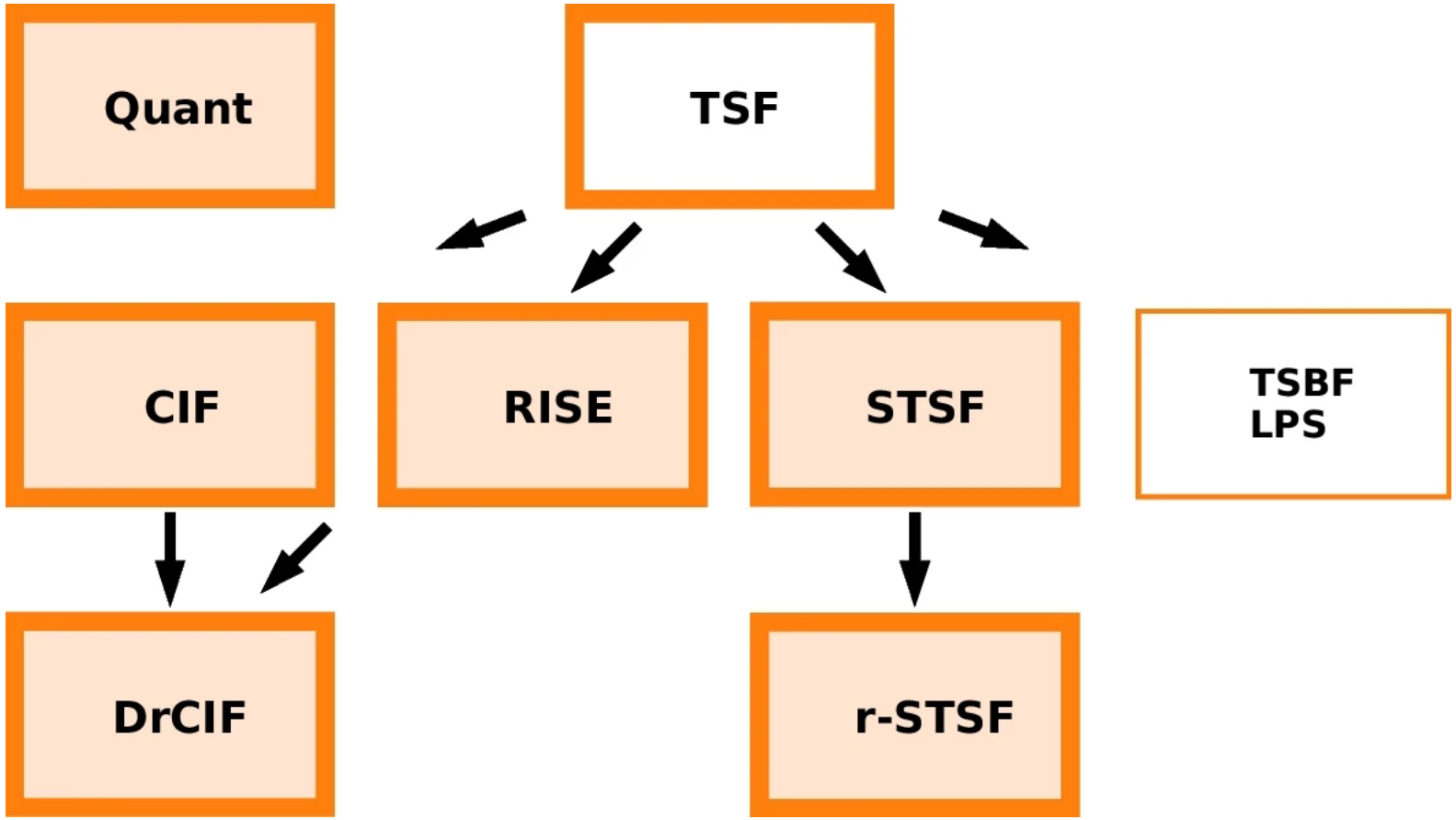
Diverse Representation CIF (DrCIF)

- DrCIF extends CIF using alternative data representations.
- DrCIF extends RISE and STSF using catch22 features.
- DrCIF selects multiple intervals taken from
  - the raw TS
  - the differencing of the TS
  - the periodograms of the TS, i.e., its DFT





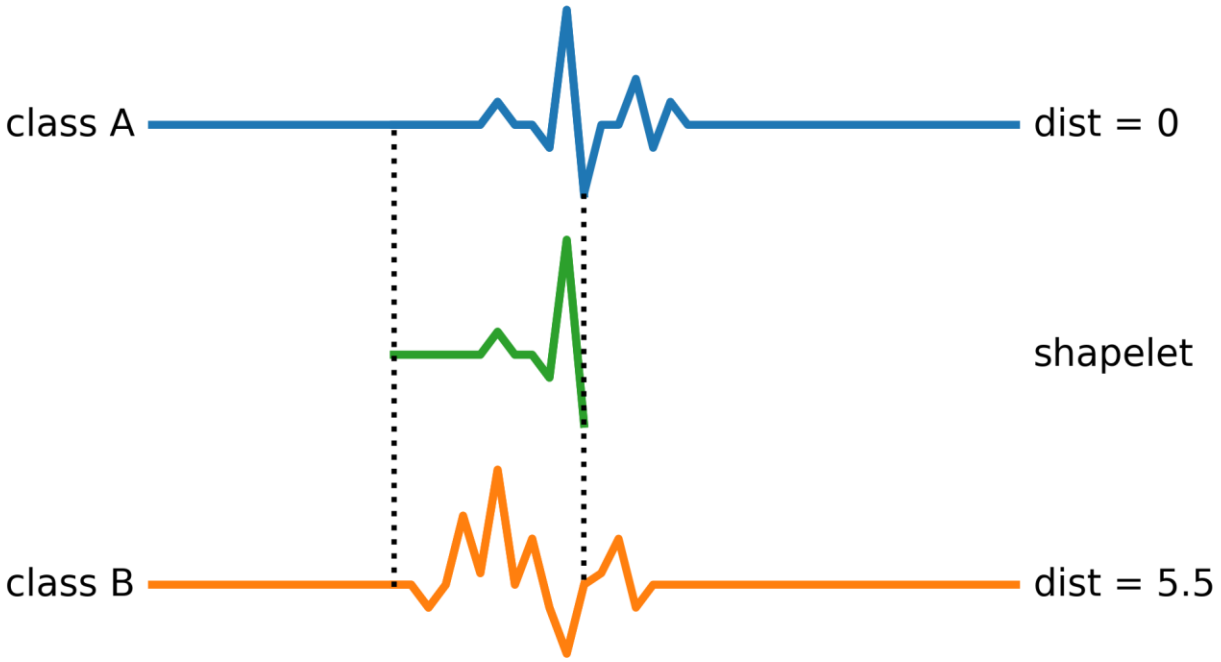
Overview of Interval-based Models and Relationships

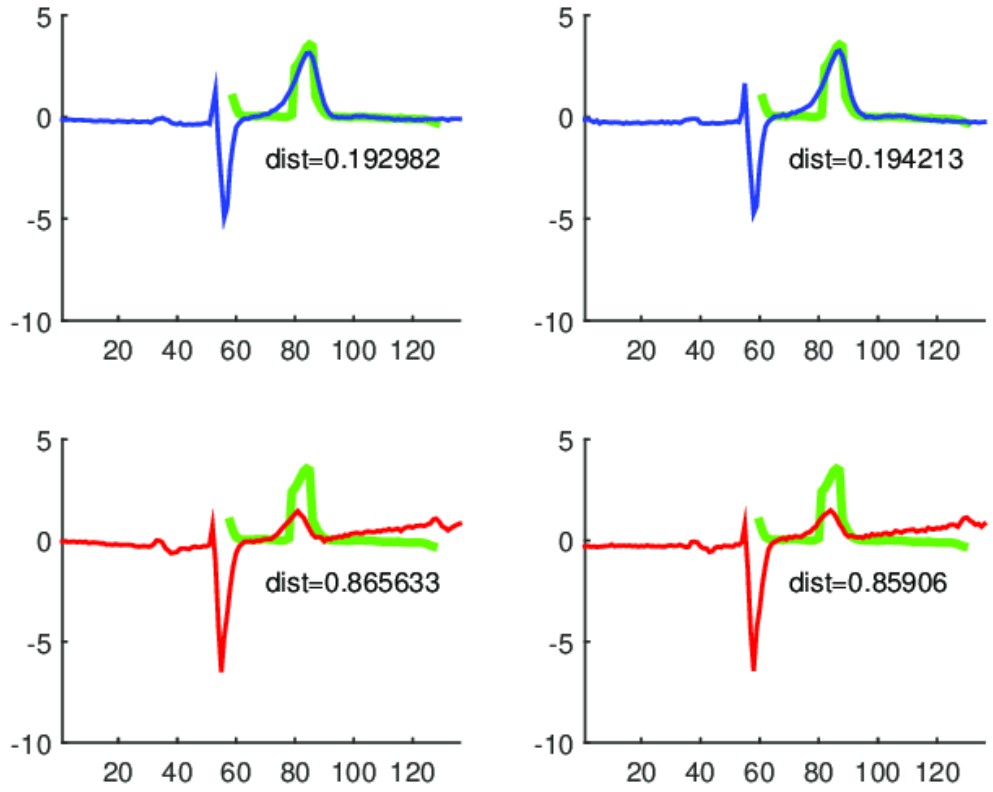


Shapelet-based Models

Shapelets

- Shapelets are TS subsequences which are maximally representative of a class or maximally discriminative between a class and another





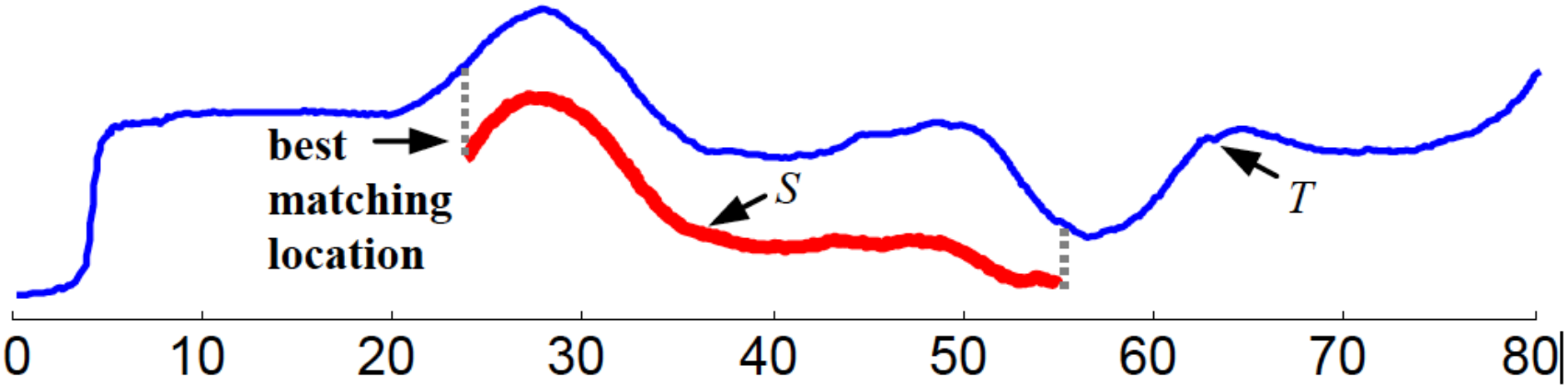
Distance with a Subsequence

- The distance between a TS and a subsequence *subsequenceDist(T, S)* is a function that takes *T* and *S* as inputs and returns a nonnegative value *d*, which is the distance from *T* to *S* as

$$subsequenceDist(T, S) = min(Dist(S, S')), for\ S' \in S_T^{|S|}$$

where $S_T^{|S|}$ is the set of all possible subsequences of *T*

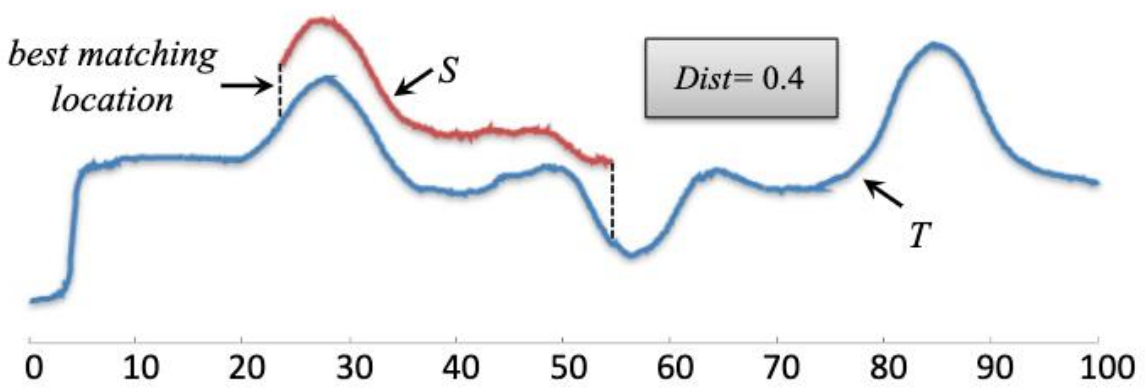
- Intuitively, *subsequenceDist(T, S)* it is the distance between *S* and its best matching location in *T*.

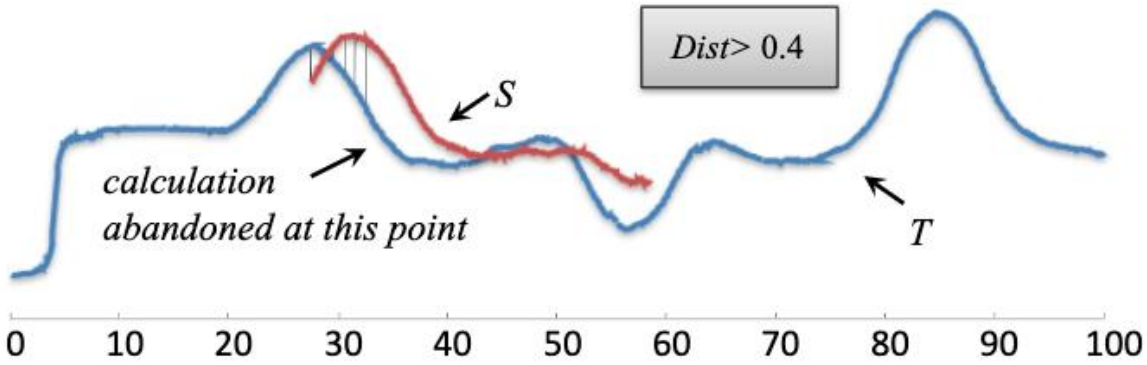


Distance Early Abandon

- We only need the minimum distance.

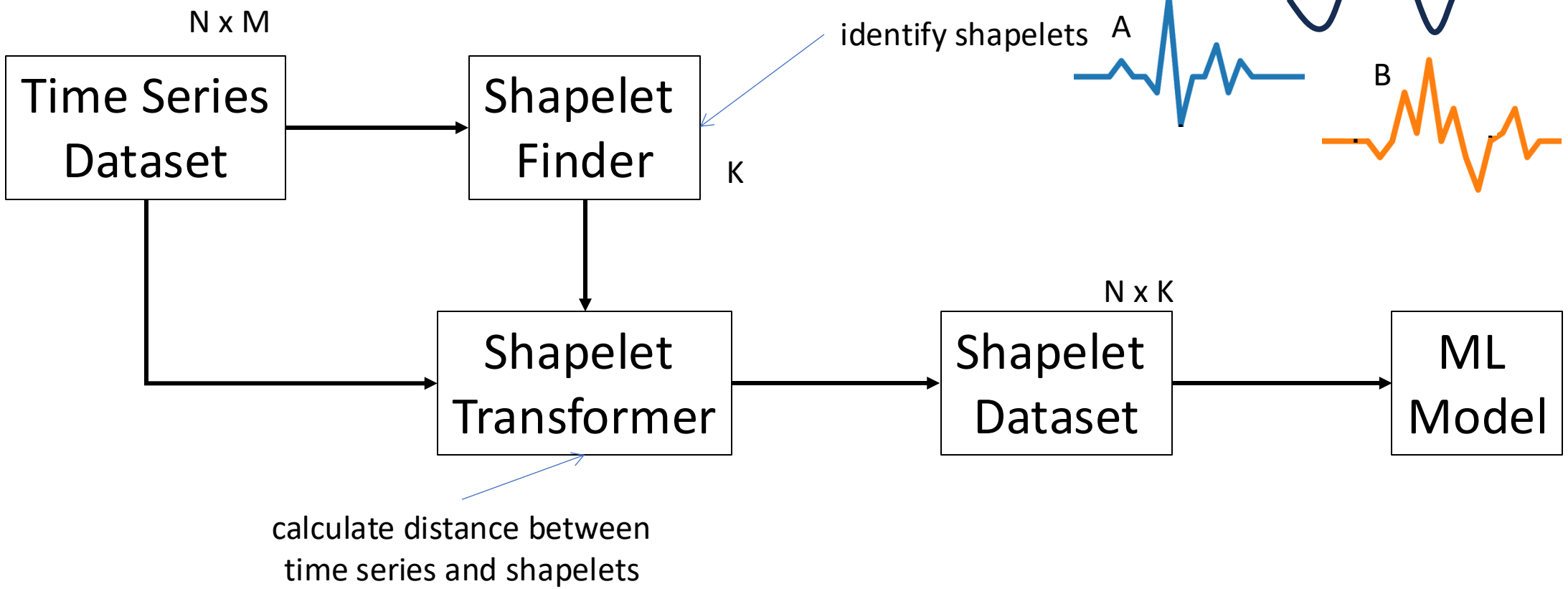
- Method
  - Keep the best-so-far distance
  - Abandon the calculation if the current distance is larger than best-so-far.



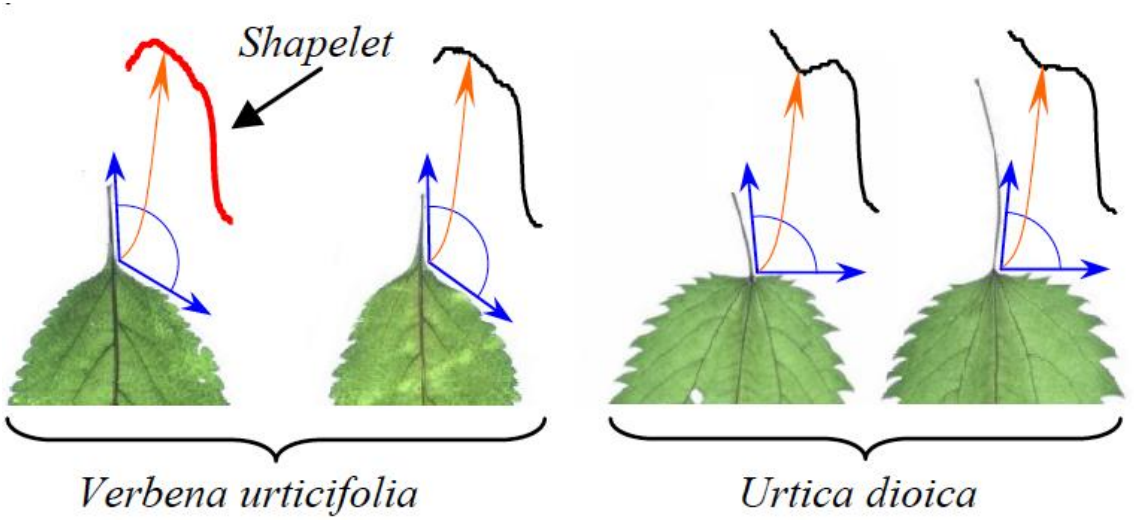


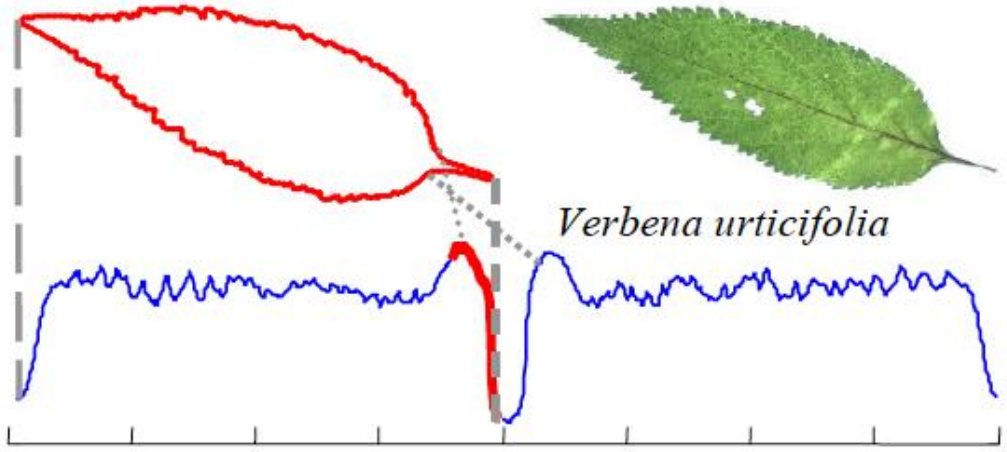
Shapelet-based Model

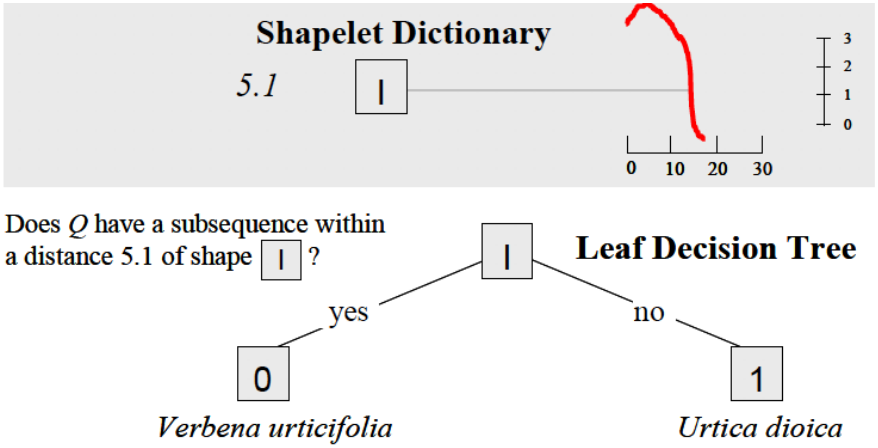
1. Given a TS dataset for classification, extract a set of K highly discriminative shapelets.
2. Transform each TS as a vector of distances with the K shapelets.
3. Train any ML model.



Shapelets



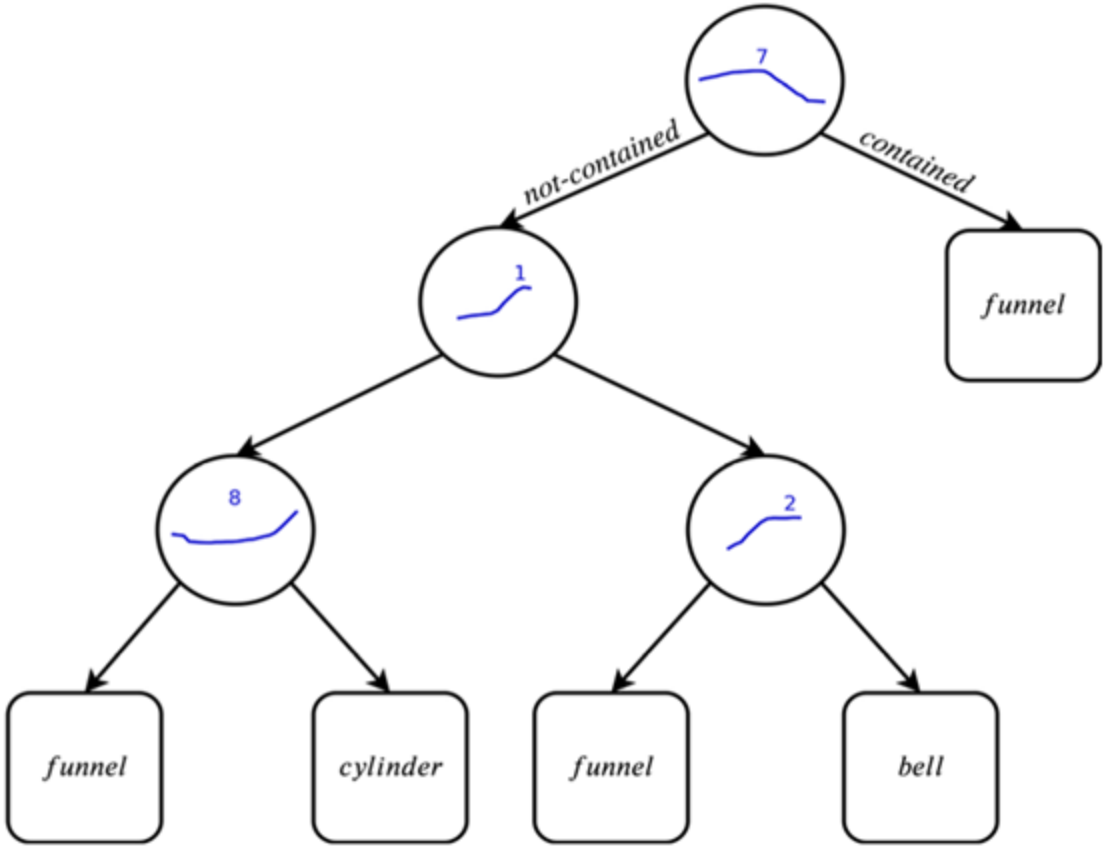


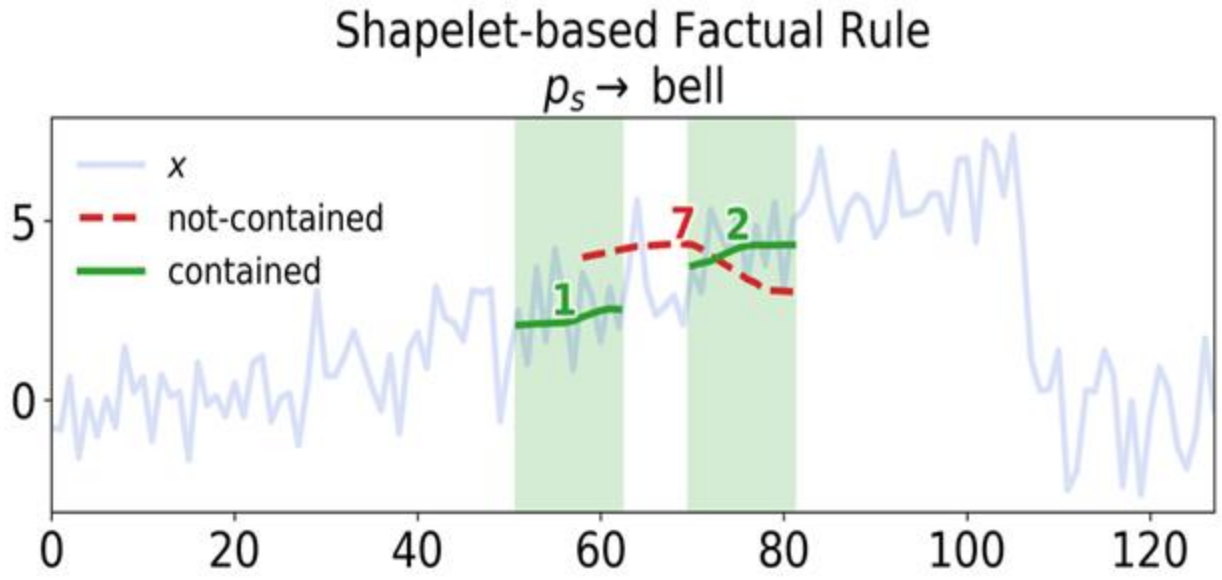


3.2	8.7
1.4	7.9
6.7	4.2
9.2	3.4

Shapelet-based Classifier

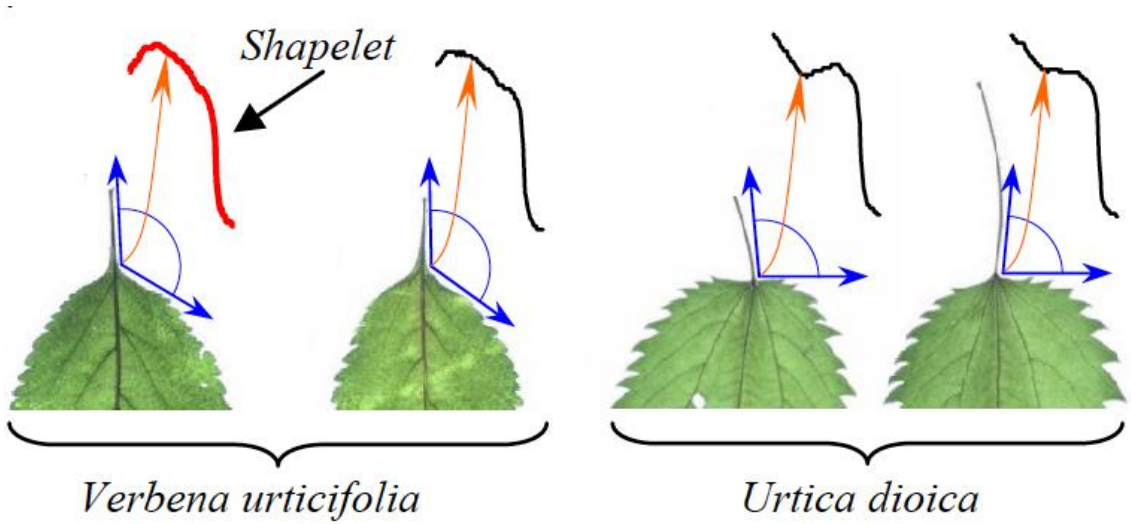
- The Shapelet-transformed TS dataset can be paired with any ML model like Decision Tree or kNN.





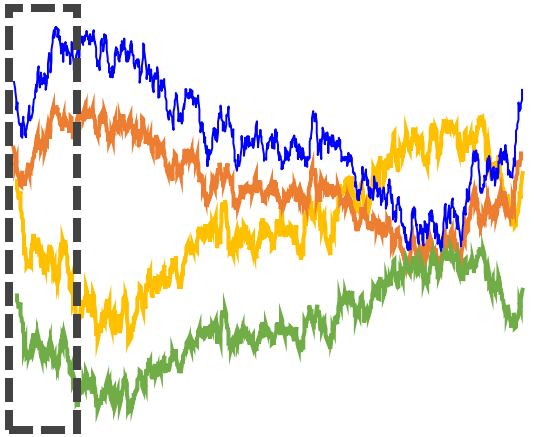
How to Extract Shapelets?

- Brute Force
- Random
- Gradient-based
- Genetic-based



Brute Force Shapelet Extraction

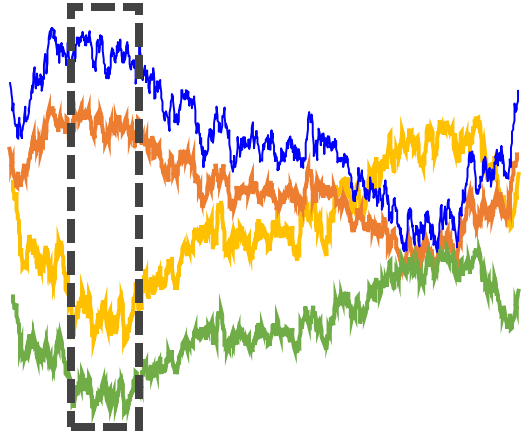
- Given a set of time windows w_1, w_2, …, w_l with different lengths and slices s_1, s_2, …, s_l
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.

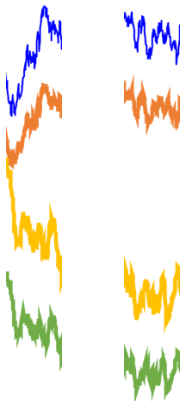


candidate shapelets

Brute Force Shapelet Extraction

- Given a set of time windows w_1, w_2, …, w_l with different lengths and slices s_1, s_2, …, s_l
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.

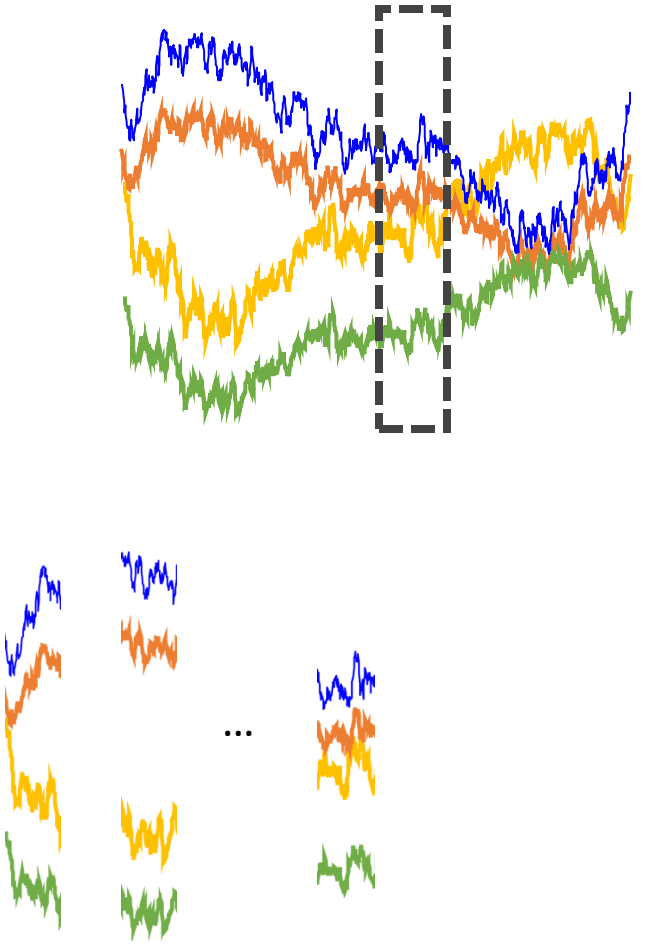




candidate shapelets

Brute Force Shapelet Extraction

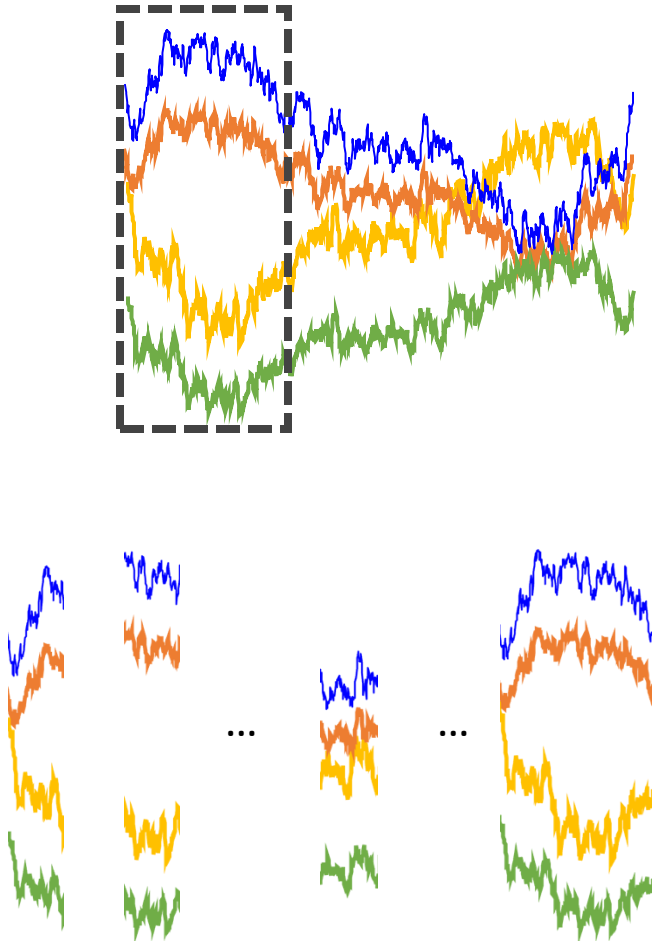
- Given a set of time windows w_1, w_2, …, w_l with different lengths and slices s_1, s_2, …, s_l
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.



candidate shapelets

Brute Force Shapelet Extraction

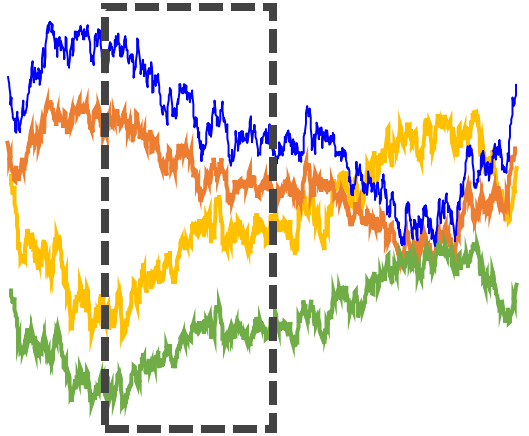
- Given a set of time windows $w_1, w_2, \ldots, w_l$ with different lengths and slices $s_1, s_2, \ldots, s_l$
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.

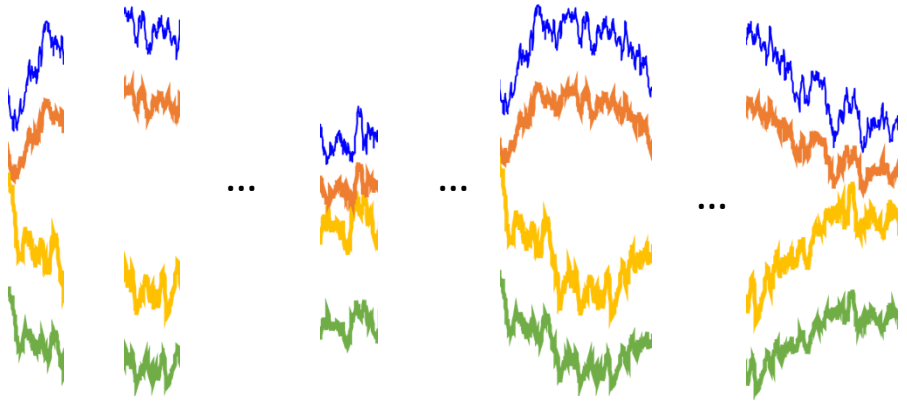


candidate shapelets

Brute Force Shapelet Extraction

- Given a set of time windows w_1, w_2, …, w_l with different lengths and slices s_1, s_2, …, s_l
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.

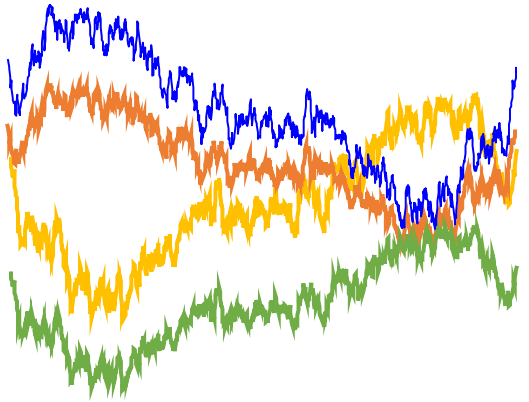


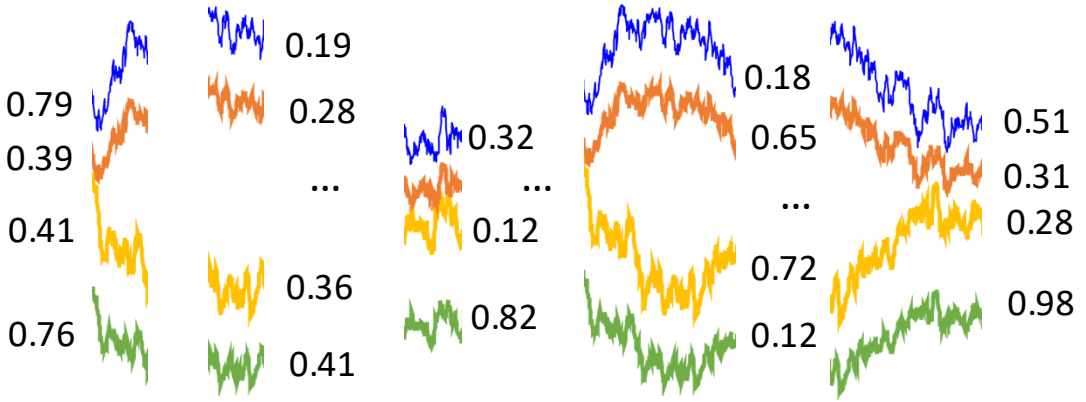


candidate shapelets

Brute Force Shapelet Extraction

- Given a set of time windows $w_1, w_2, \ldots, w_l$ with different lengths and slices $s_1, s_2, \ldots, s_l$
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.

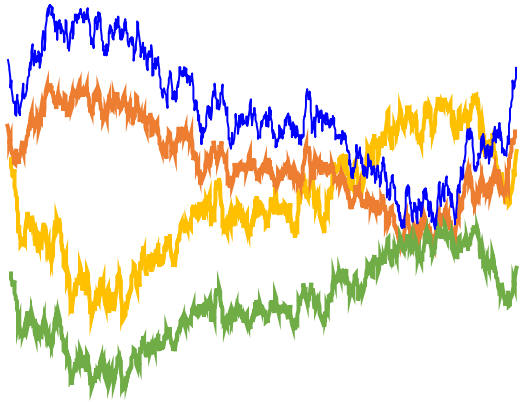


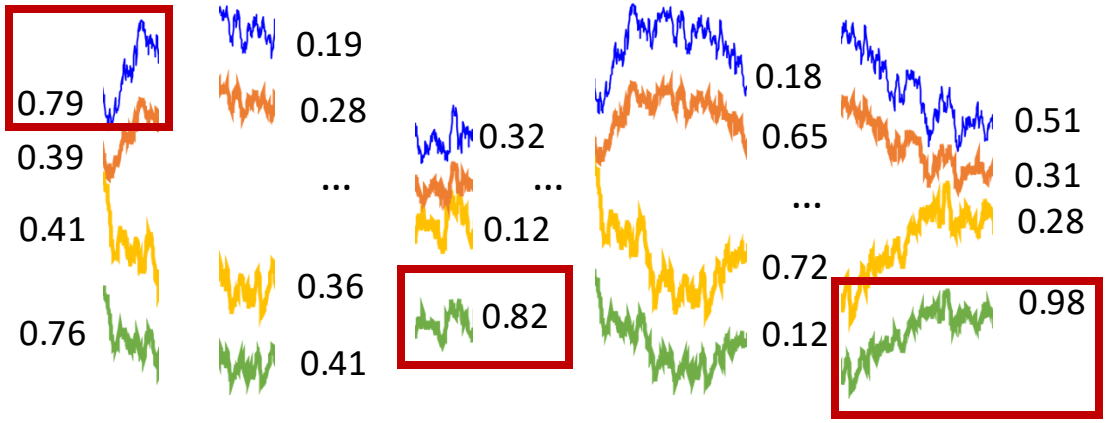


candidate shapelets

Brute Force Shapelet Extraction

- Given a set of time windows w_1, w_2, …, w_l with different lengths and slices s_1, s_2, …, s_l
- For each time window w_i and slice s_j
- For each time series T in the dataset X
- Move the time window w_i along T and store all the subsequencs S with length w_i as candidate shapelets.
- Calculate the distance between each candidate and the time series in X.
- Evaluate the Information Gain of each candidate shapelet and select the K shapelets with the highest score.





candidate shapelets

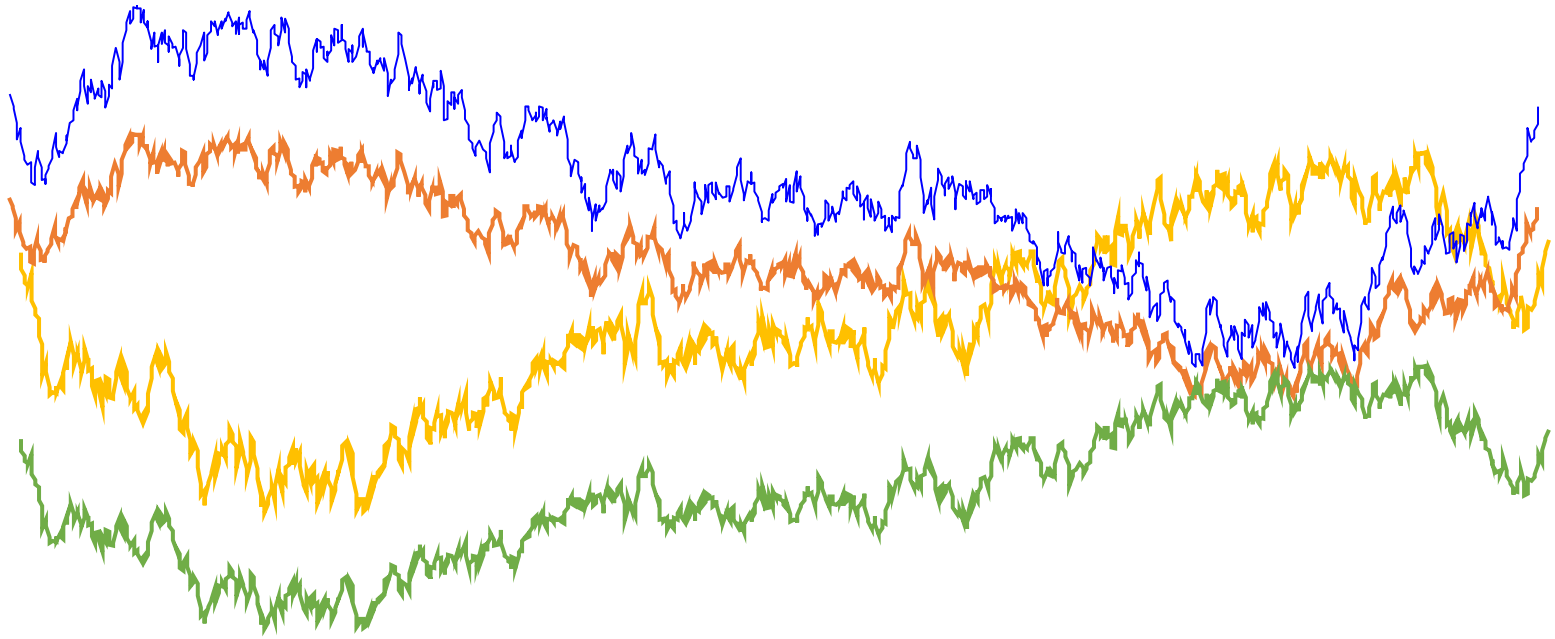
Problem with Brute Force Shapelet

- The total number of candidate is $\sum_{l=MINLEN}^{MAXLEN} \sum_{T_i \in D} (|T_i| - l + 1)$

- For each candidate you have to compute the distance between this candidate and each training sample
- For instance
  - 200 instances with length 275
  - 7,480,200 shapelet candidates

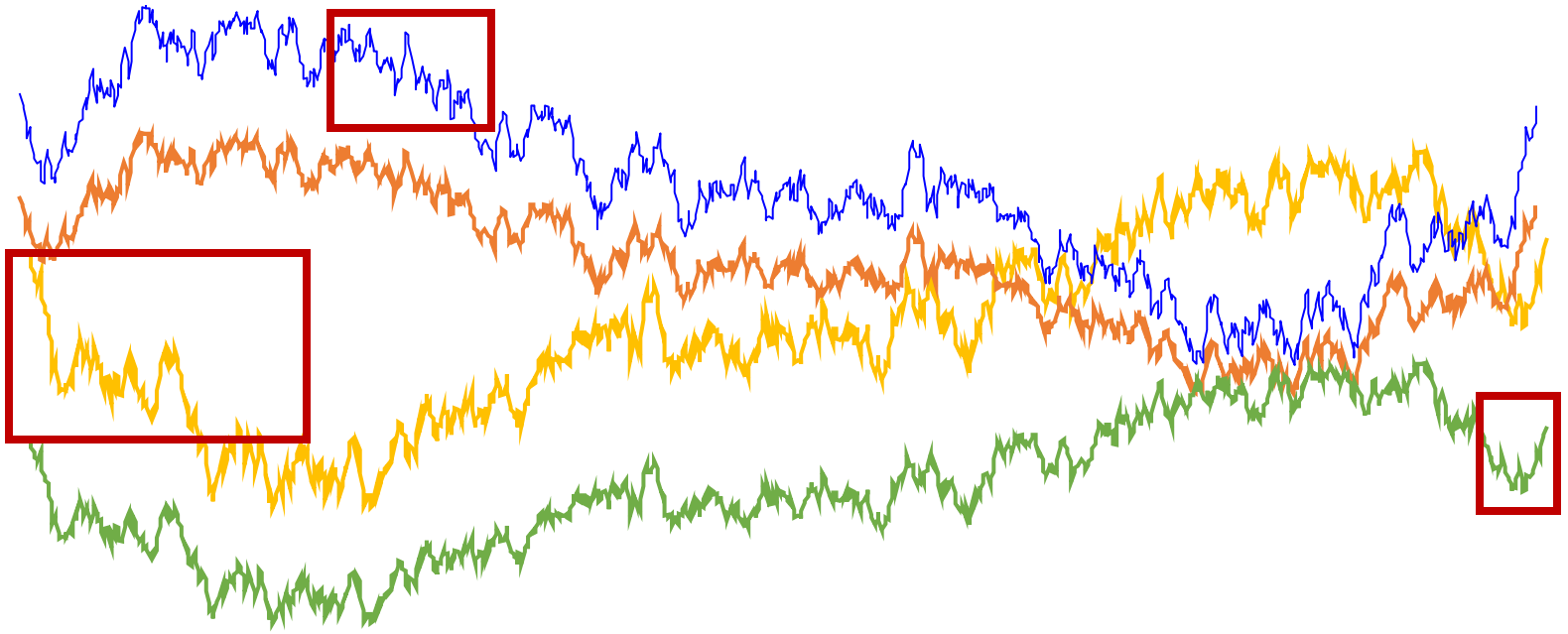
Random Shapelet Extraction

- Given a set of time windows w_1, w_2, …, w_l and a dataset X of time series randomly select K subsequences from the time series in X to be used as shapelets.



Random Shapelet Extraction

- Given a set of time windows w_1, w_2, …, w_l and a dataset X of time series randomly select K subsequences from the time series in X to be used as shapelets.



Gradient-based Shapelet

$$\mathcal{F}_i = \mathcal{L}(Y_i, \hat{Y}_i) + \frac{\lambda_W}{I} \sum_{k=1}^{K} {W_k}^2$$

$$\hat{Y}_i = W_0 + \sum_{k=1}^{K} M_{i,k} W_k, \quad \forall i \in \{1, \ldots, I\}$$

$$\mathcal{L}(Y, \hat{Y}) = -Y \ln \sigma(\hat{Y}) - (1 - Y) \ln \left(1 - \sigma(\hat{Y})\right)$$

- Learn optimal shapelets without exploring all possible candidates.
- Step 1: start with rough initial guesses for the shapelet
- Step 2: iteratively learn/optimize the shapelets by minimizing a loss function by using a predictive model that is differentiable with respect to shapelets.
- Shapelets can be updated in a stochastic gradient descent optimization fashion by taking steps towards the minimum of the classification loss function

Algorithm 1 Learning Time-Series Shapelets

Require: $T \in \mathbb{R}^{I \times Q}$, Number of Shapelets K, Length of a shapelet L, Regularization λ_W, Learning Rate η, Number of iterations: maxIter

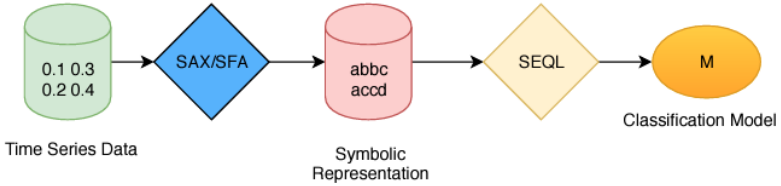
Ensure: Shapelets $S \in \mathbb{R}^{K \times L}$, Classification weights $W \in \mathbb{R}^{K}$, Bias $W_0 \in \mathbb{R}$

for iteration=$\mathbb{N}_1^{\text{maxIter}}$ **do**
  for $i = 1, \ldots, I$ **do**
    for $k = 1, \ldots, K$ **do**
      $W_k \leftarrow W_k - \eta \frac{\partial \mathcal{F}_i}{\partial W_k}$
      for $L = 1, \ldots, L$ **do**
        $S_{k,l} \leftarrow S_{k,l} - \eta \frac{\partial \mathcal{F}_i}{\partial S_{k,l}}$
      end for
    end for
    $W_0 \leftarrow W_0 - \eta \frac{\partial \mathcal{F}_i}{\partial W_0}$
  end for
end for
return S, W, W_0

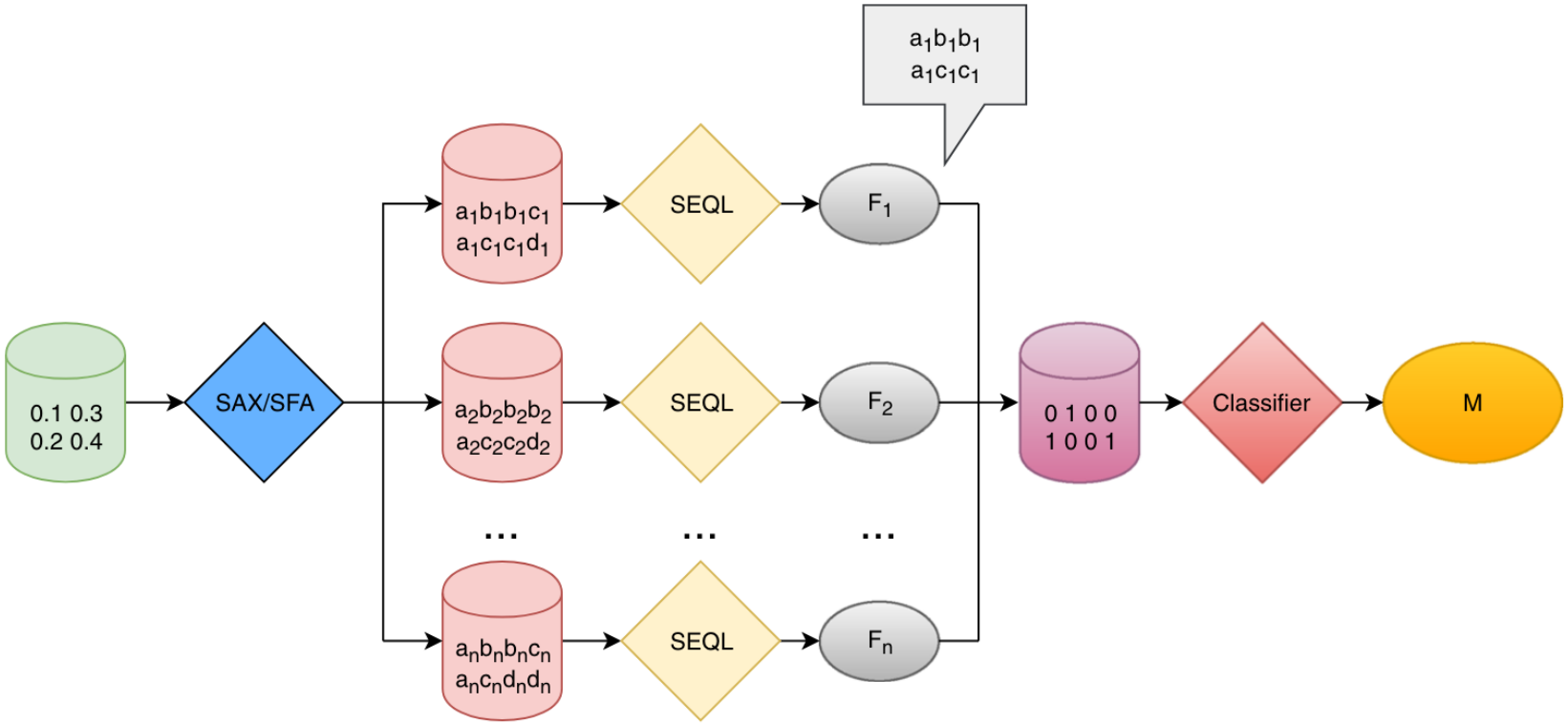
Shapelets Remarks

- A shapelet is a pattern/subsequence which is maximally representative of a class/maximally discriminative between a class and the rest with respect to a given dataset of TS.
- Shapelets can be significantly more accurate/robust then global structural features because as they are *local features* based on distances

MrSEQL – Multi Resolution SQL

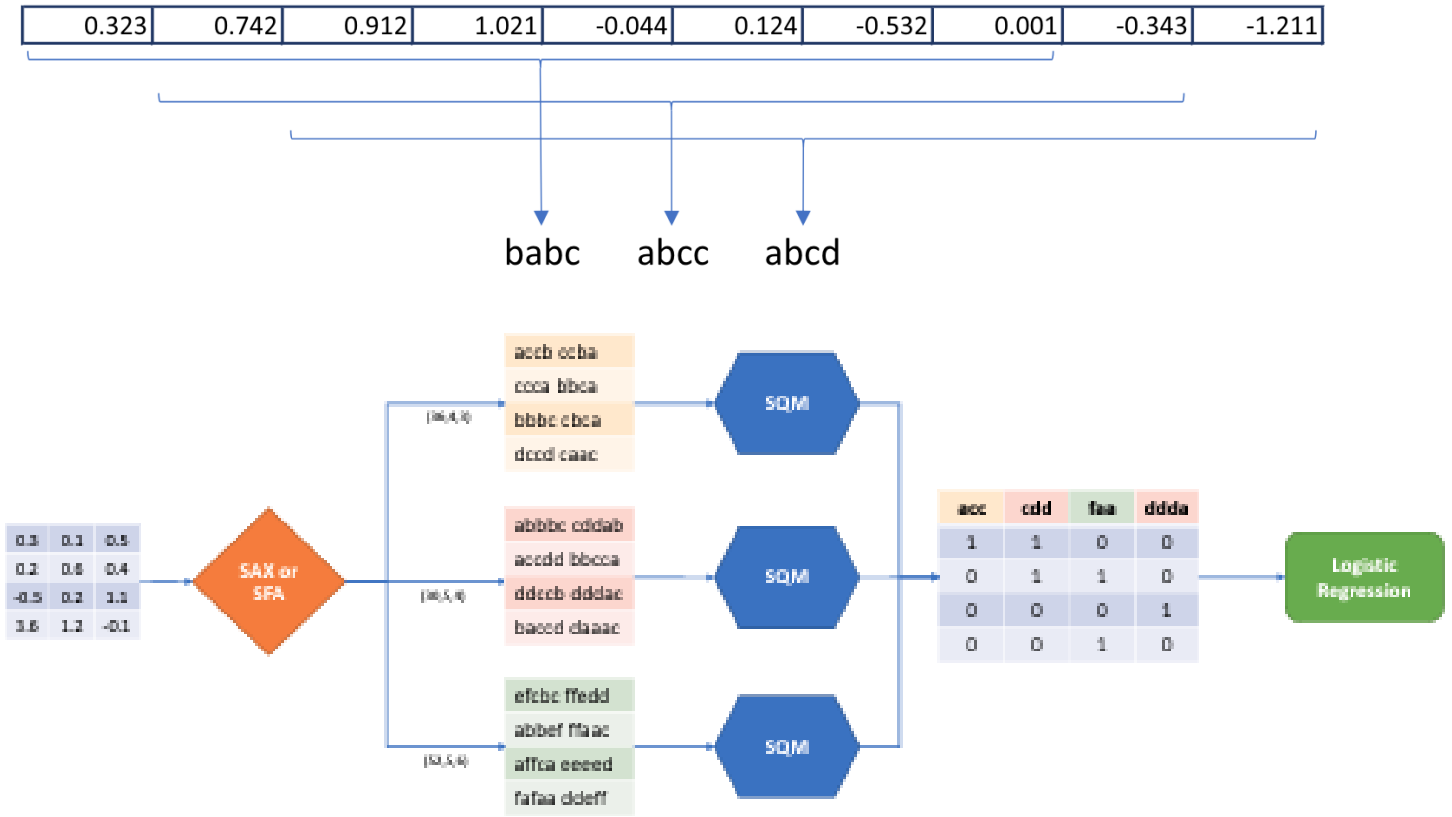


- MrSEQL is an ensemble of SEQL algorithms.
- SEQL (SEQuence Learner) selects a set of discriminative subsequences.
- MrSEQL produce *k* SEQL models from different SAX or SFA representations.

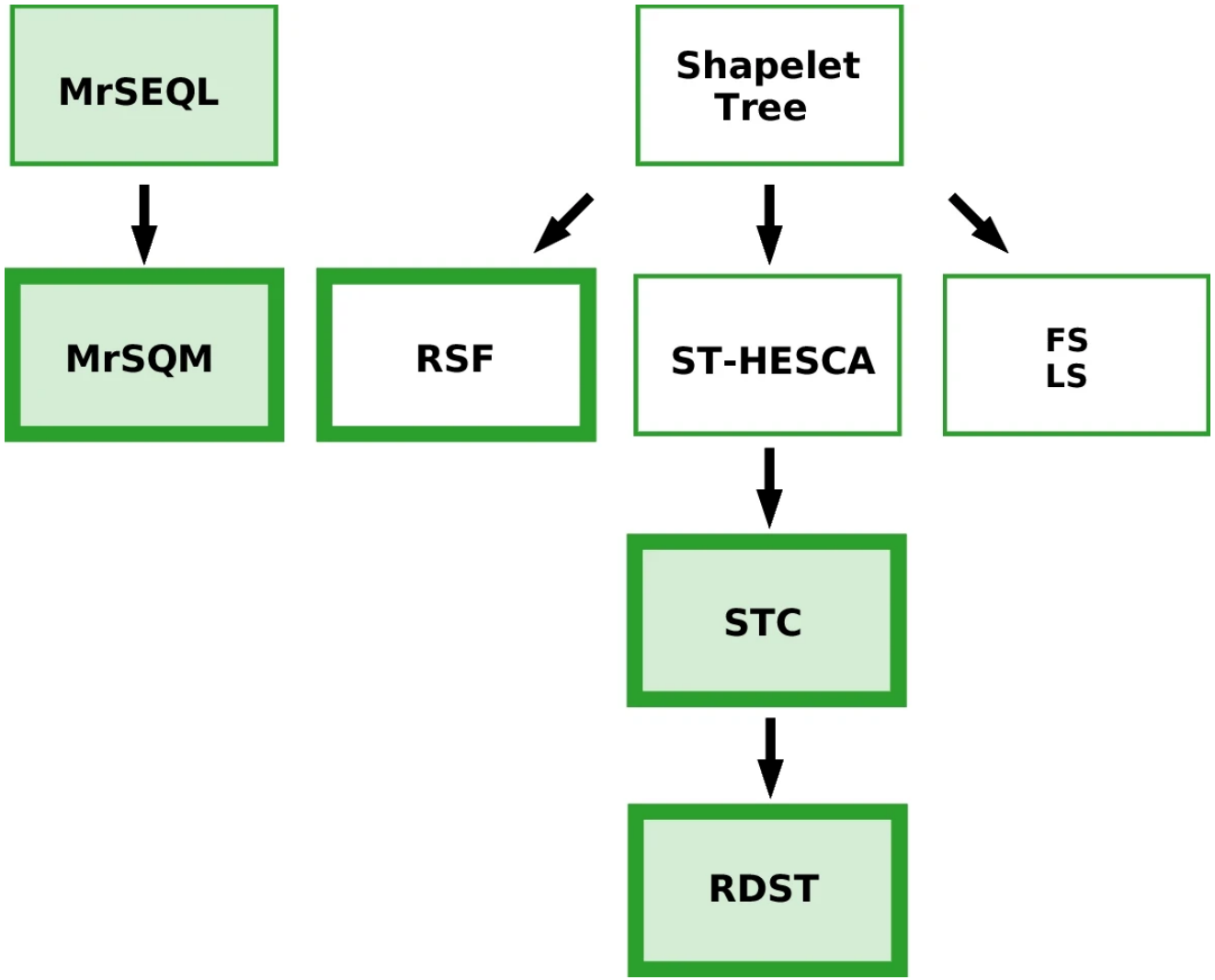


MrSQM - Multiple Representations Sequence Miner

- Extends MrSEQL with a sampling strategy for reducing the number of features generated in the BOP.



Overview of Shapelet-based Models and Relationships



Dictionary-based Models

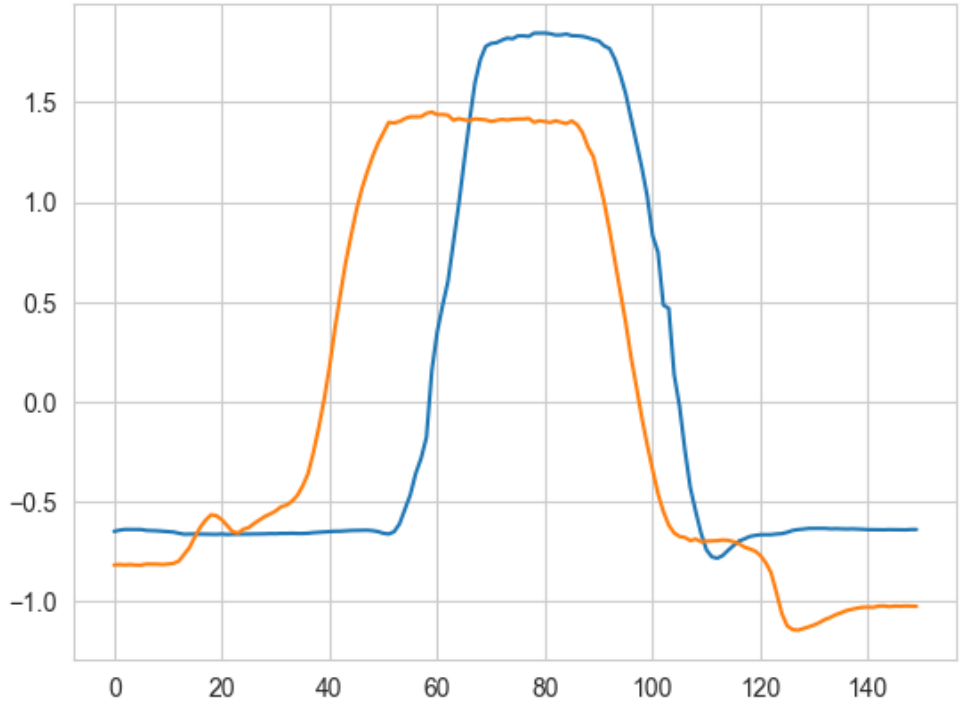
Bag of Words

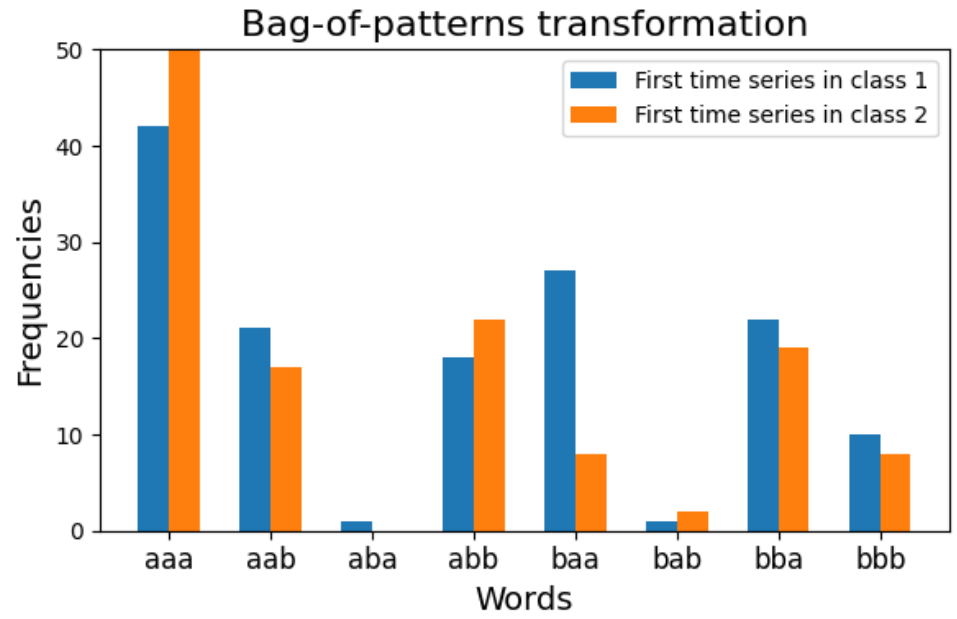
- Typically used in Natural Language Processing (NLP) and Information Retrieval (IR) but common also in TSA.
- A bag-of-words transformation turns a text into an unordered collection (or “bag”) of words typically accounting for multiplicity.

- Example: *John likes to watch movies. Mary likes movies too.*
- BoW: "John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1

Bag of Words in TSA

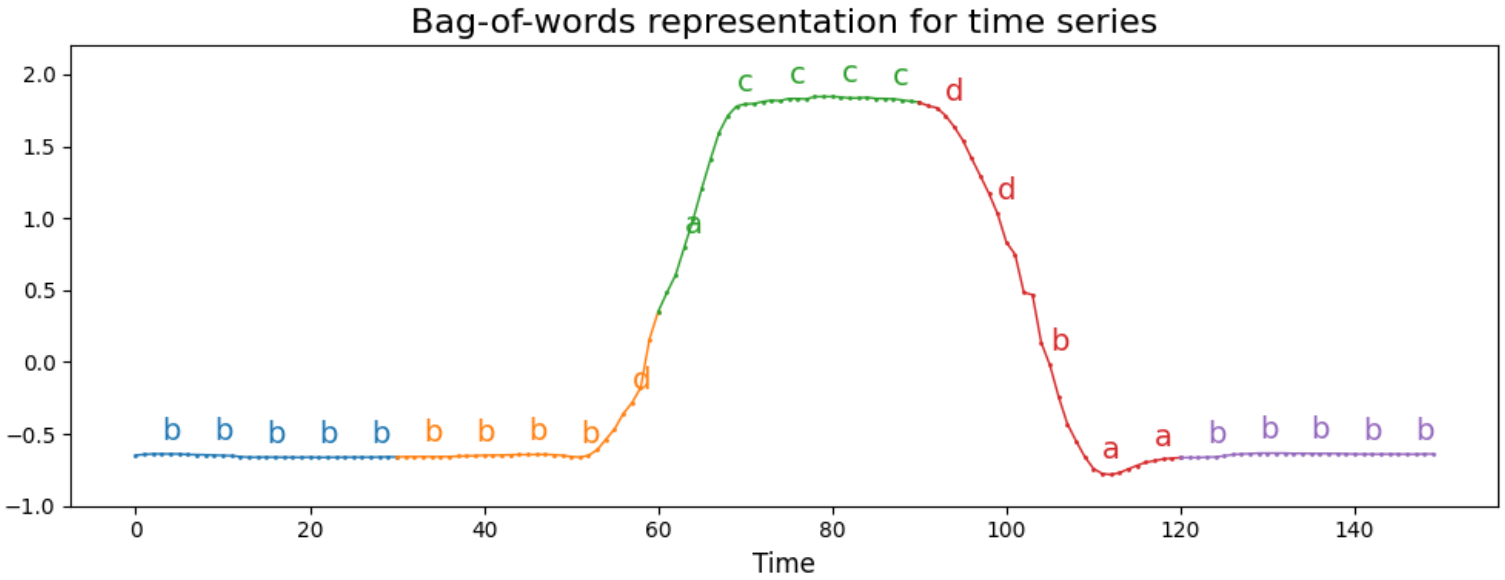
- Given a TS *T*, Bag of Words extracts subseries using a sliding window *w*, then normalize each subseries and transforms it into a **word** using an approximation approach such as:
- SAX - Symbolic Aggregate approXimation
- SFA - Symbolic Fourier Approximation





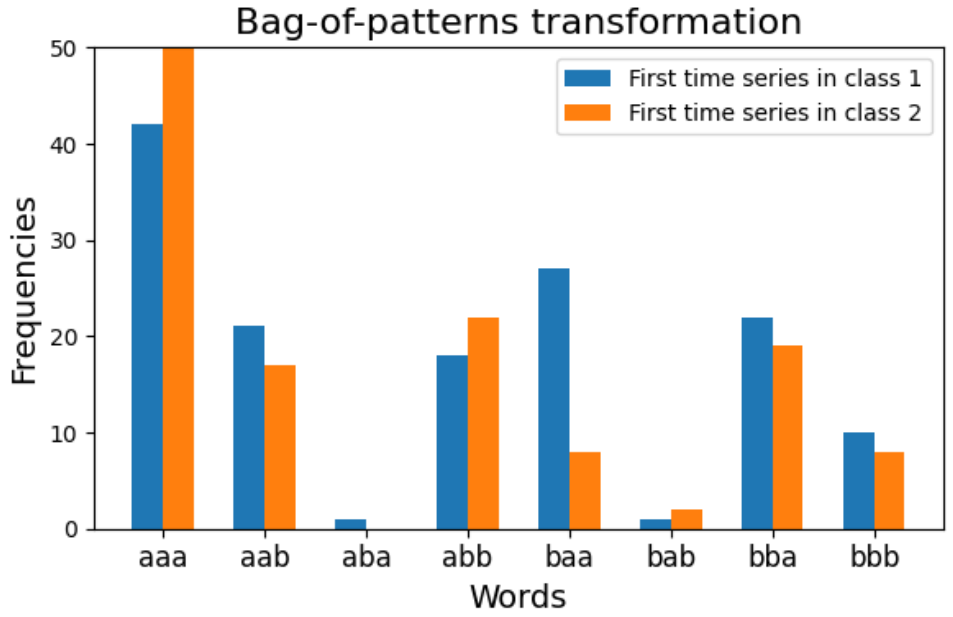
Bag of Words in TSA Hyperparameters

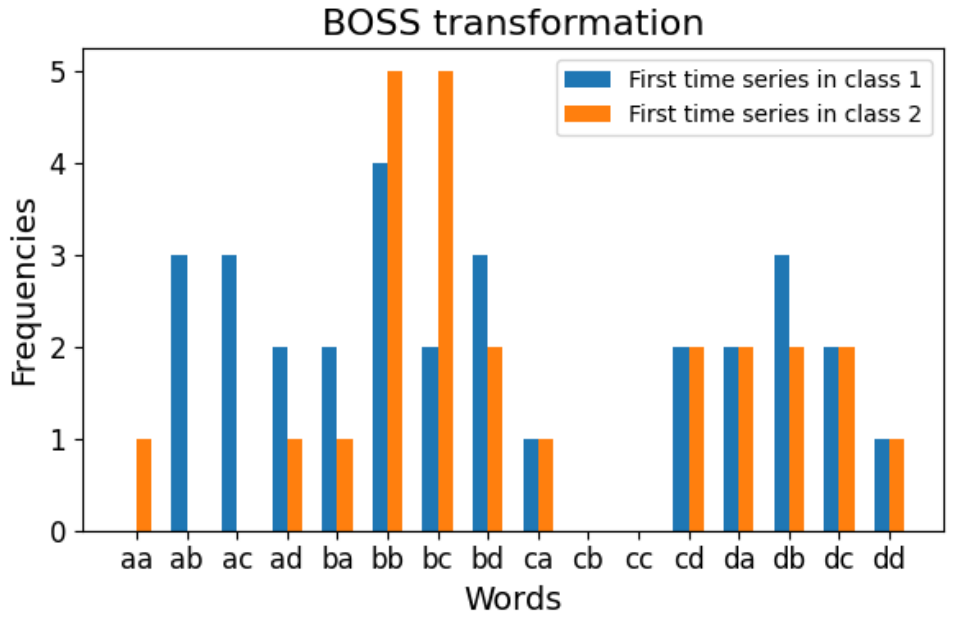
- window_size: length of the sliding window w
- window_step: step of the sliding window w (default 1)
- word_size: length of the words, i.e., number of characters
- n_bins: size of the alphabet, i.e., number of distinct characters



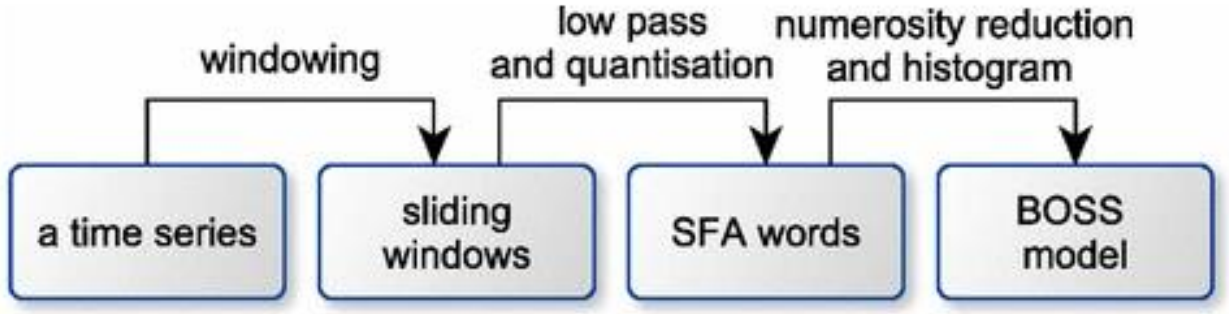
Bag of Words Algorithms

- Bag of Patterns (BOP) transforms each subsequence into a word using SAX*
- Bag-of-SFA Symbols (BOSS) transforms each subsequence into a word using SFA
- * Sometimes also BOSS transformations are named BOP.

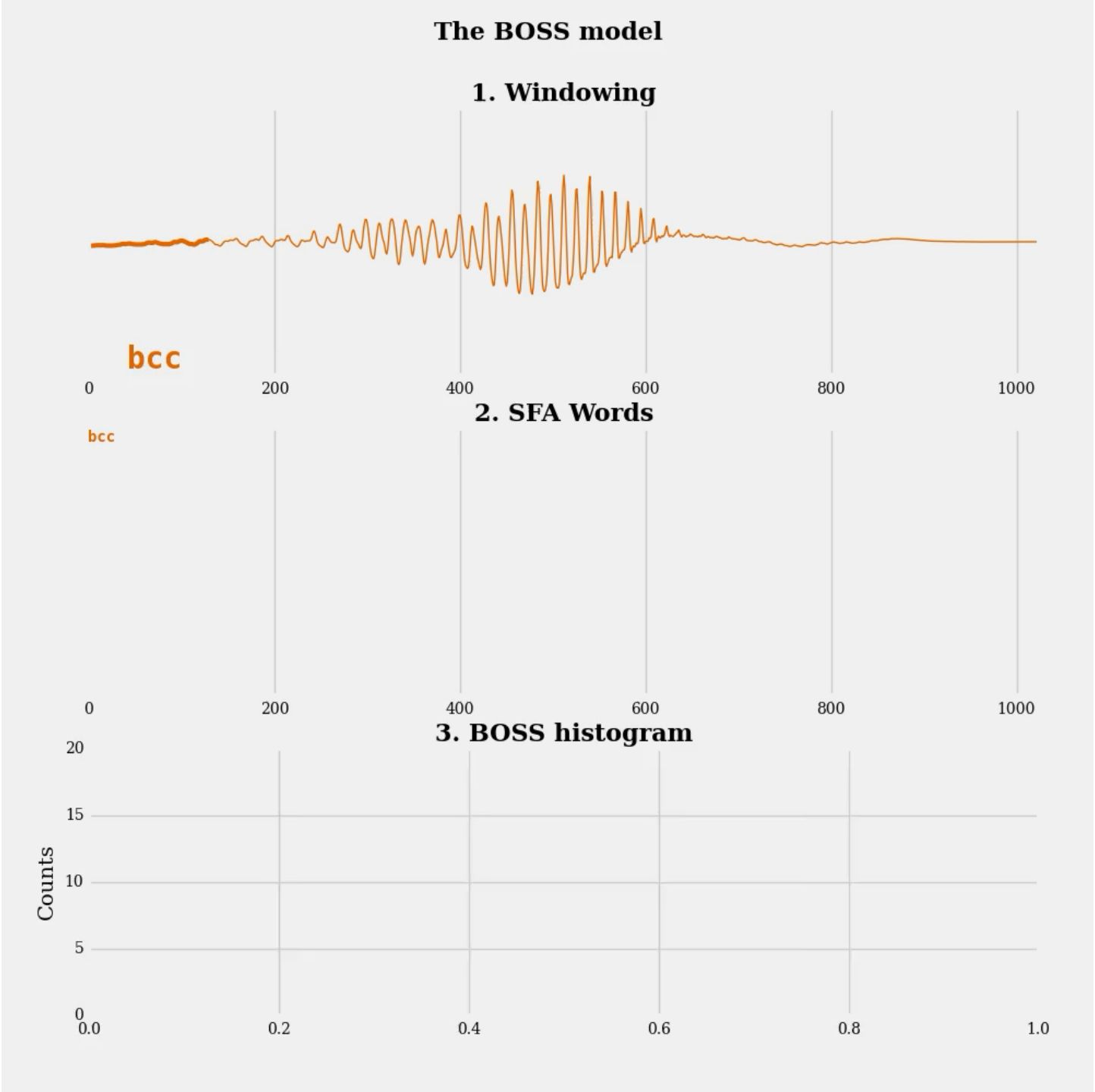


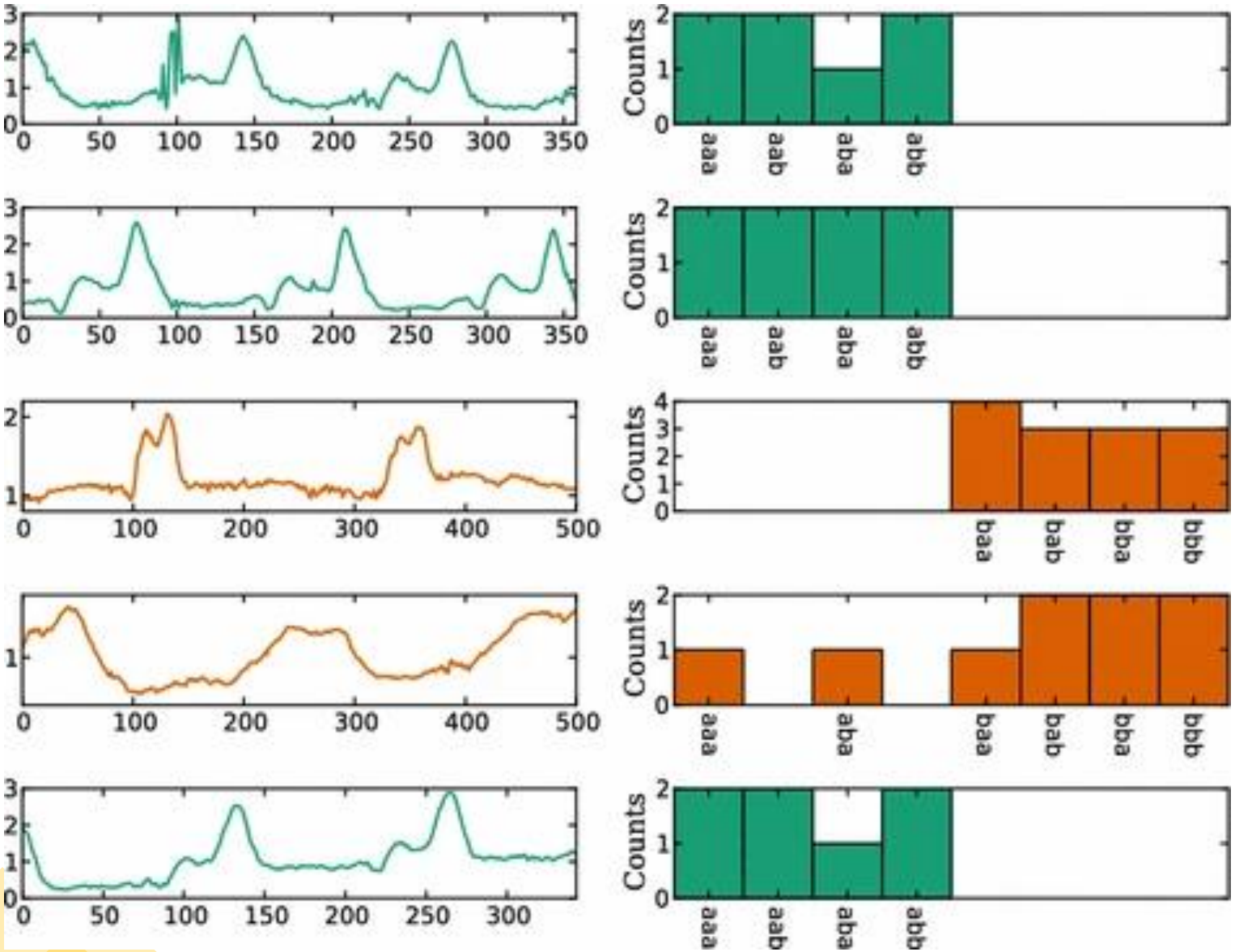


The BOSS Model



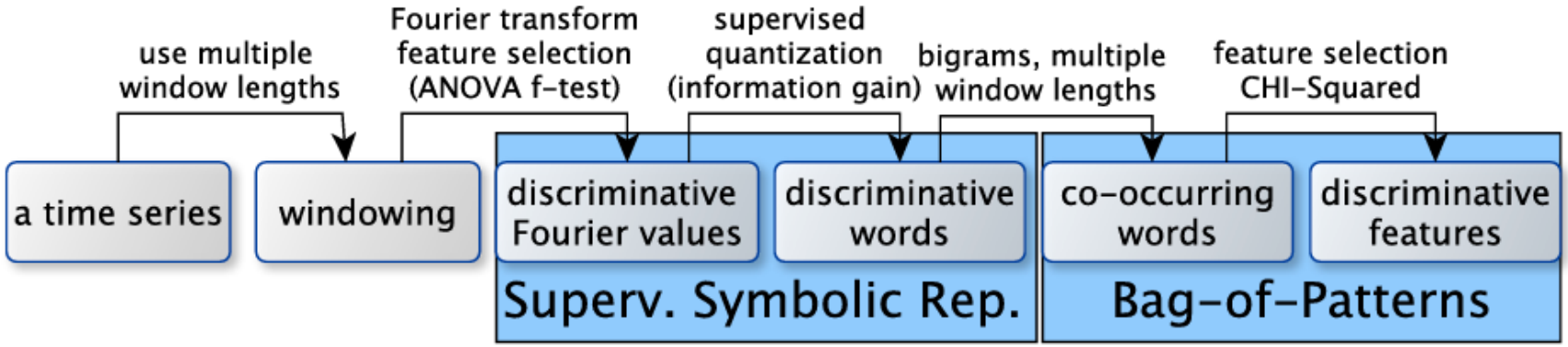
- First, sliding windows of length are extracted from a TS.
- Next, each sliding window is normalized to have a standard deviation of 1 to obtain amplitude invariance.
- SFA transformation is applied to each real valued sliding window transforming a time series into an *unordered* set of SFA words.
- Using an unordered set provides invariance to the horizontal alignment of each substructure within the TS (phase shift invariance).
- The first occurrence of an SFA word is registered and all duplicates are ignored.
- From these SFA words a histogram is constructed, which counts the occurrences of the SFA words


The BOSS model
1. Windowing
bcc
0
200
400
600
800
1000
2. SFA Words
bcc
0
200
400
600
800
1000
3. BOSS histogram
Counts
20
15
10
5
0
0.0
0.2
0.4
0.6
0.8
1.0


Counts
aaa
aab
aba
abb
baa
bab
bba
bbb

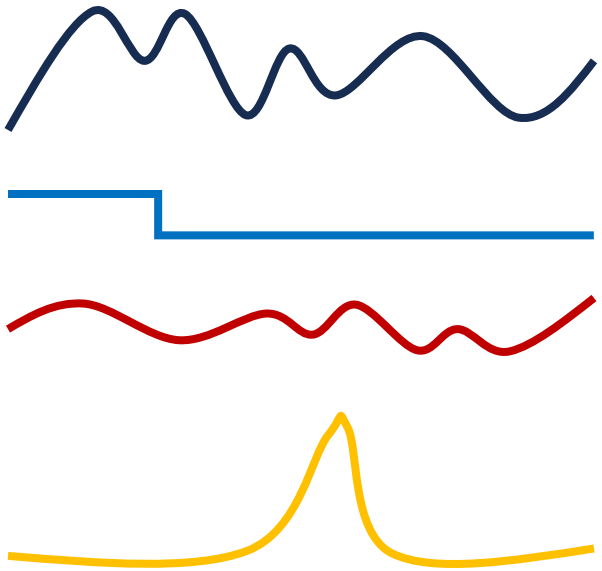
WEASEL - Word ExtrAction for time Series cLassification

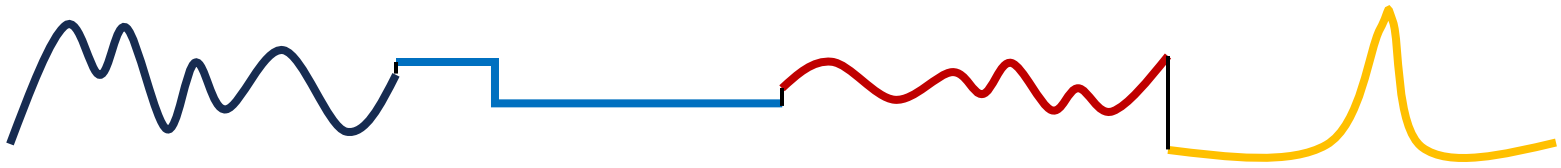
- WEASEL extracts normalized windows of different lengths from a time series.
- Each window is approximated using the SFA, and those Fourier coefficients are kept that best separate TS from different classes using the ANOVA F-test.
- The remaining coefficients are discretized into a word using information gain binning.
- A bag-of-patterns is built from the words (unigrams) and neighboring words (bigrams), also including windows of variable lengths.
- The ChiSquared test is applied to filter out irrelevant words.
- A logistic regression classifier is applied.



How to Deal with Multivariate TS

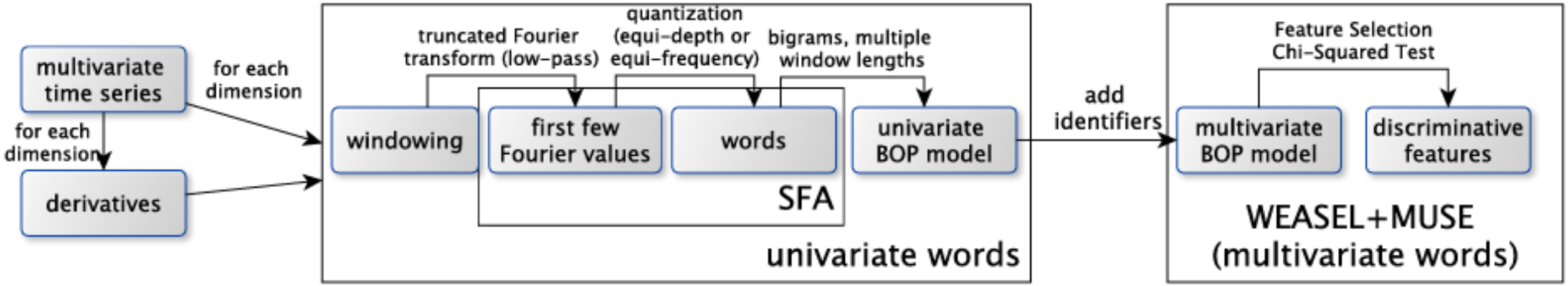
- We can assume that the various signals are independent
- A trivial way to adopt all the models analyzed is to concatenate the different signals to obtain a (long) univariate time series.

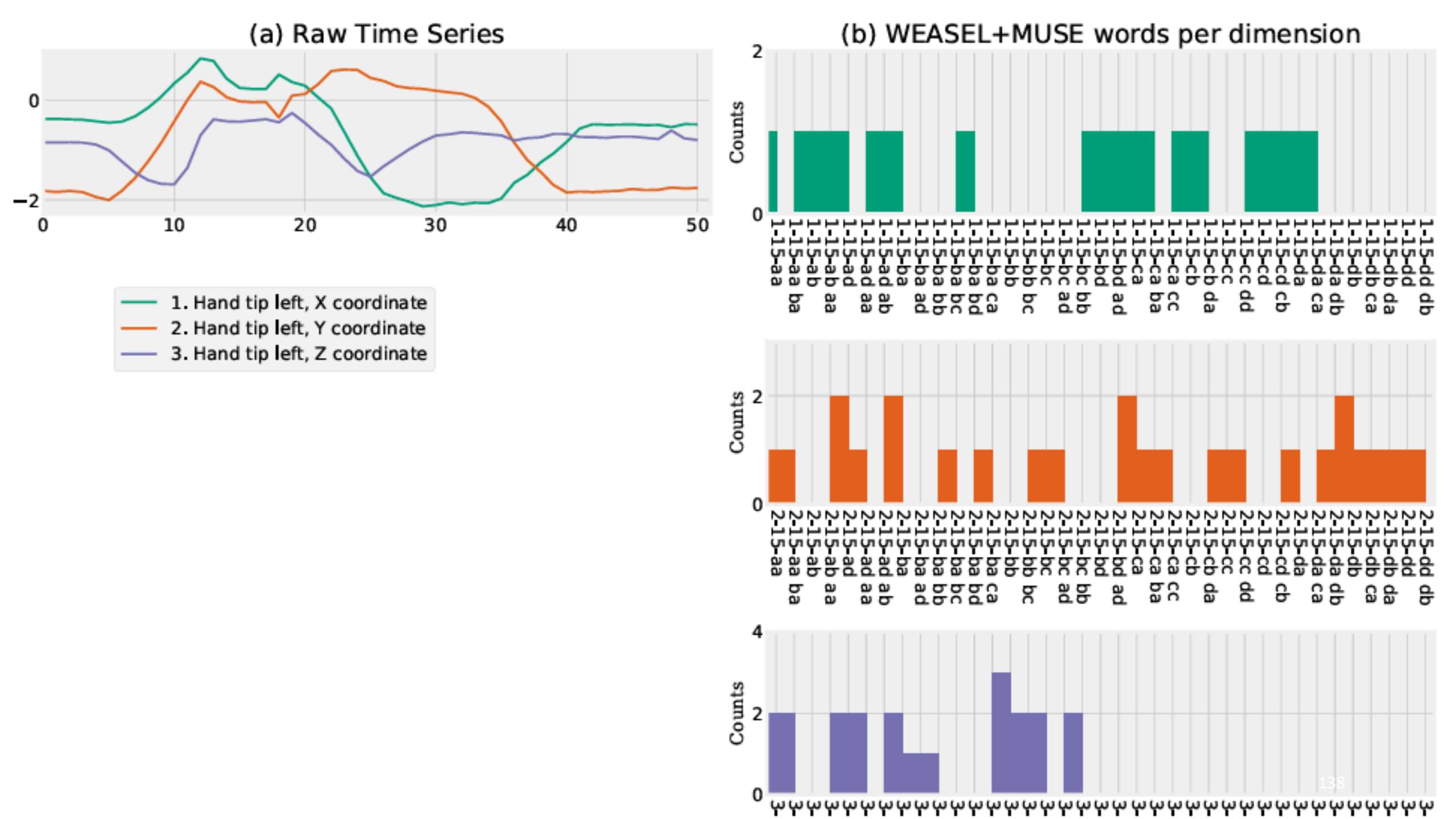




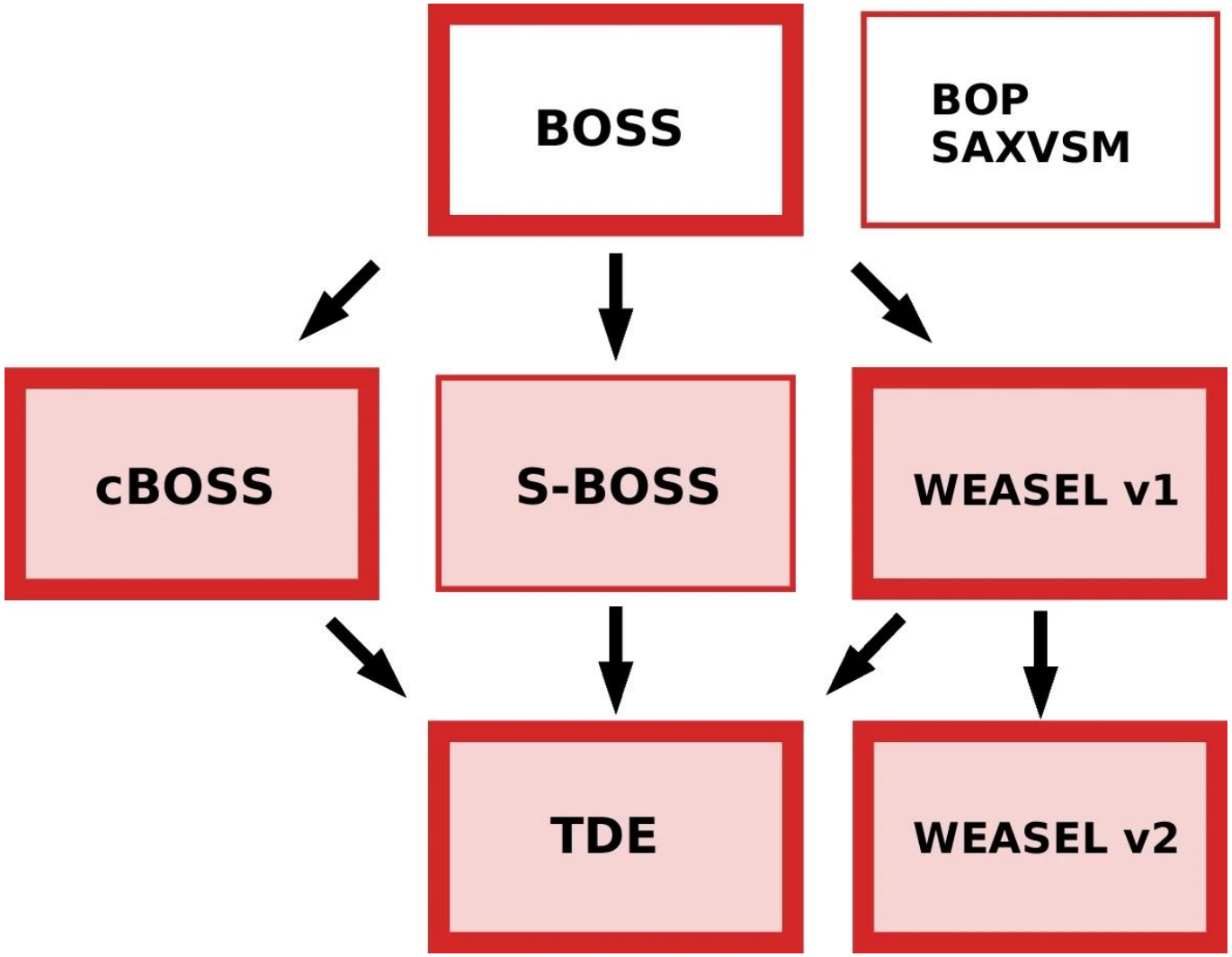
MUSE - Multivariate Unsupervised Symbols and dErivatives

- Also named WEASEL+MUSE, extends WEASEL by considering multivariate words.
- Multivariate words are obtained from the univariate words by concatenating each word with an identifier representing the sensor and the window size.




(a) Raw Time Series
0
−2
0
10
20
30
40
50
1. Hand tip left, X coordinate
2. Hand tip left, Y coordinate
3. Hand tip left, Z coordinate
(b) WEASEL+MUSE words per dimension
Counts
2
0
1-15-aa
1-15-aa ba
1-15-ab
1-15-ab aa
1-15-ad
1-15-ad aa
1-15-ad ab
1-15-ba
1-15-ba ad
1-15-ba bb
1-15-ba bc
1-15-ba bd
1-15-ba ca
1-15-bb
1-15-bb bc
1-15-bc
1-15-bc ad
1-15-bc bb
1-15-bd
1-15-bd ad
1-15-ca
1-15-ca ba
1-15-ca cc
1-15-cb
1-15-cb da
1-15-cc
1-15-cc dd
1-15-cd
1-15-cd cb
1-15-da
1-15-da ca
1-15-da db
1-15-db
1-15-db ca
1-15-db da
1-15-dd
1-15-dd db
Counts
2
0
2-15-aa
2-15-aa ba
2-15-ab
2-15-ab aa
2-15-ad
2-15-ad aa
2-15-ad ab
2-15-ba
2-15-ba ad
2-15-ba bb
2-15-ba bc
2-15-ba bd
2-15-ba ca
2-15-bb
2-15-bb bc
2-15-bc
2-15-bc ad
2-15-bc bb
2-15-bd
2-15-bd ad
2-15-ca
2-15-ca ba
2-15-ca cc
2-15-cb
2-15-cb da
2-15-cc
2-15-cc dd
2-15-cd
2-15-cd cb
2-15-da
2-15-da ca
2-15-da db
2-15-db
2-15-db ca
2-15-db da
2-15-dd
2-15-dd db
Counts
4
2
0
3-
138

Overview of Dictionary-based Models and Relationships



References

- catch22: CAnonical Time-series CHaracteristics selected through highly comparative time-series analysis Carl H Lubba et al. 2019.
- Proximity Forest An effective and scalable distance-based classifier for time series. Benjamin Lucas et al. 2018.
- A time series forest for classification and feature extraction. Houtao Deng. 2013.
- The Canonical Interval Forest (CIF) Classifier for Time Series Classification. Matthew Middlehurst et al. 2020.
- Diverse Representation Canonical Interval Forest. (HIVE-COTE 2.0: a new meta ensemble for time series classification). Matthew Middlehurst et al. 2021.
- Time series shapelets: a new primitive for data mining. L. Ye and E. Keogh. 2009.
- Classification of time series by shapelet transformation. Jon Hills et al. 2014.
- Learning time-series shapelets. Josif Grabocka et al. 2014.
- Binary Shapelet Transform for Multiclass Time Series Classification. A. Bostrom et al. 2017
- The BOSS is concerned with time series classification in the presence of noise. Patrick Schäfer 2014.
- Interpretable Time Series Classification using Linear Models and Multi-resolution Multi-domain Symbolic Representations. Thach Le Nguyen et al. 2020.
- MrSQM: Fast Time Series Classification with Symbolic Representations. Thach Le Nguyen et al. 2022.
- Fast and Accurate Time Series Classification with WEASEL. Patrick Schäfer et al. 2017.
- Bake off redux: a review and experimental evaluation of recent time series classification algorithm. Matthew Middlehurst et al. 2024