University of Udine

J48SS: a Decision Tree Approach for the Handling of Sequential and Time Series Data

Andrea Brunello
andrea.brunello@uniud.it
www.andreabrunello.com

28th November 2023

Temporal aspects play a major role in the extraction of information from data:

- **sequences**:
  - track the evolution of customer behaviour over time
  - can be used in text classification
- **time series**:
  - may be useful for stock market analysis, or
  - weather forecasts

Temporal/sequential information may be complemented by other kinds of data, for instance in the medical domain:

- **numerical**: age and weight of the patient
- **categorical**: smoker or not, gender, blood group
- **sequential**: history of symphoms and medical prescriptions, or even free text
- **time series**: pulse and oxygen saturation values over time

Name	Age	Gender	Medication	Fever
Ann	42	Female	Aspirin>(Paracetamol,Caffeine)>Paracetamol	37.5,38,38.5,37.5
Mark	27	Male	Paracetamol>Paracetamol	39,38,38,38.5
Jane	59	Female	Ibuprofen>Ibuprofen>(Ibuprofen,Valium)	38,37,37.5

Also, in the contact center domain:

- **numerical**: age of the called person
- **categorical**: gender and place of residence of the called person
- **sequential**: textual data obtained from the call recordings
- **time series**: sound pressure over time

Name	Age	Number	Transcription
Ann	42	+39 339 5179842	hi>i>would>like>to>reserve>a>table
Mark	27	+39 337 5563221	hello>may>i>ask>for>some>information
Jane	59	+39 348 9784121	good>morning>are>you>open>this>evening

An all-in-one, interpretable model

In some domains, **understanding the classification process** is as important as the accuracy of the classification itself.

Decision trees are a natural model to rely on:

- can be understood by (non I.T.) domain experts
- easy to train and apply to new data

J48SS decision tree learner `https://github.com/dslab-uniud/J48SS`:

- based on Quinlan's C4.5, VGEN frequent pattern extraction algorithm and NSGA-II multi-objective EA
- can handle a **mixture** of numeric, categorical, sequential and time series data during the same execution cycle

Why J48SS

Several advantages:

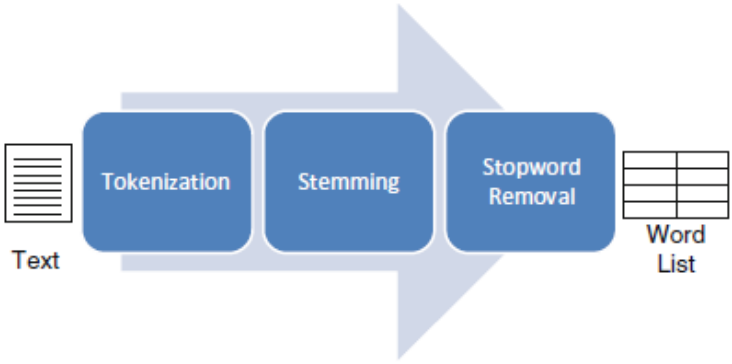
- it is **interpretable** for the domain experts
- it can make the **data preparation step easier**, for example it is capable of directly handling text (no *ngram* preprocessing is needed) and time series
- it can effectively **combine and make use of different kinds of data**, which would require the use of separate algorithms

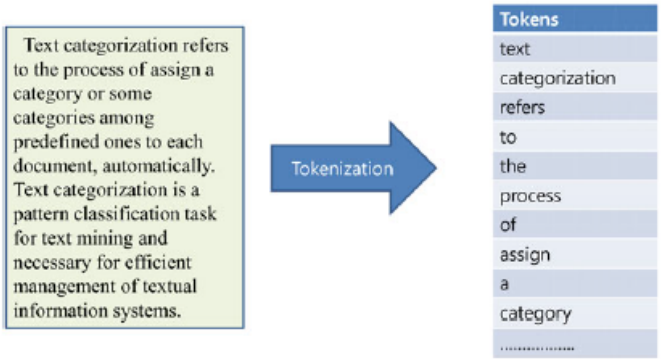
```
sequence_attribute !contains (A,B)>D
|   attribute_numeric <= 20: class_1
|   attribute_ numeric > 20
|   |   attribute_nominal = value_1: class_0
|   |   attribute_nominal = value_2: class_1
sequence_ attribute contains (A,B)>D: class_0
```

Text is a kind of unstructured data, it is difficult to analyze it in its original form.

To apply classical machine learning approaches, we have to describe it by means of a **fixed number of attributes**.

In this sense, the first step is that of performing *text indexing*, that is, converting texts into a list of words. Then, we have to devise a way to assign an importance score to each word, a task that is named *term weighting*.


Text
Tokenization
Stemming
Stopword Removal
Word List


Text categorization refers to the process of assign a category or some categories among predefined ones to each document, automatically. Text categorization is a pattern classification task for text mining and necessary for efficient management of textual information systems.
Tokenization
Tokens
text
categorization
refers
to
the
process
of
assign
a
category
.....

Varied Form	Root Form
better	good
best	good
simpler	simple
simplest	simple
assigning	assign
assigned	assign
assignment	assign
complexity	complex
analysis	analyze
categorization	categorize
categorizing	categorize
categorizes	categorize

Stop word removal is the process of removing stop words from the list of tokens.

Stop words are the most common words in a language, though there is no single universal consensus on them.

Some examples are prepositions, such as ‘in’, ‘on’, ‘to’, and so on. Conjunctions such as ‘and’, ‘or’, ‘but’, and ‘however’.

Intuitively, one may want to remove stop words since they do not bring any information, being far too common.

Caveat: in some cases, stop words may actually be important. For instance, consider the English rock band *The Who*. Or, the phrase *I told you that she was not happy*, which may be left with just [‘told’, ‘happy ’].

N-grams help to preseve sequential information in the text. However, the longer the n-gram, the lower the number of its occurrences (frequency, *support*).

Full sentence	It does not, however, control whether an exaction is within Congress's power to tax.
Unigrams	"It"; "does"; "not,"; "however,"; "control"; "whether"; "an"; "exaction"; "is"; "within"; "Congress's"; "power"; "to"; "tax."
Bigrams	"It does"; "does not,"; "not, however,"; "however, control"; "control whether"; "whether an"; "an exaction"; "exaction is"; "is within"; "within Congress's"; "Congress's power"; "power to"; "to tax."
Trigrams	"It does not"; "does not, however"; "not, however, control"; "however, control whether"; "control whether an"; "whether an exaction"; "an exaction is"; "exaction is within"; "is within Congress's"; "within Congress's power"; "Congress's power to"; "power to tax."

	about	bird	heard	is	the	word	you
About the bird, the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

Splitting procedure in J48SS

Algorithm 1 Node splitting procedure

```
1:  procedure NODE_SPLIT(NODE)
2:      if NODE is "pure" or other stopping criteria met then
3:          make NODE a leaf node
4:      else
5:          best_attr ← null
6:          best_ng ← 0
7:          for each numeric or categorical attribute a do
8:              a_ng ←get normalized information gain of a
9:              if a_ng > best_ng then
10:                 best_ng ← a_ng
11:                 best_attr ← a
12:         for each sequential string attribute s do
13:             pat, pat_ng ←get best frequent pattern in s
14:             if pat_ng > best_ng then
15:                 best_ng ← pat_ng
16:                 best_attr ← pat
17:         for each time series string attribute t do
18:             shap, shap_ng ←get shapelet in t using NSGA-II
19:             if shap_ng > best_ng then
20:                 best_ng ← shap_ng
21:                 best_attr ← shap
22:         children_nodes ← split instances in NODE on best_attr
23:         for each child_node in children_nodes do
24:             call NODE_SPLIT(child_node)
25:             attach child_node to NODE
26:     return NODE
```

Sequential pattern mining

Some terminology:

- let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of **items**
- **sequence**: ordered list of **itemsets** $\{i_1, i_2\}, \{i_3\}, \{i_1, i_3, i_4\}$
- intuitively, items that belong to the same itemset are considered to occur at the same time
- we consider **frequent patterns**, i.e., concatenations of itemsets that frequently appear in a set of sequences
- to reduce the number of generated patterns, and to avoid overfitting, we focus on **sequential generator patterns**
- a frequent pattern is considered to be a **generator** if it does not exist any pattern which is contained in it and has the same frequency

VGEN pattern extraction algorithm

VGEN algorithm **recursively** extracts all generator patterns that are more frequent than a predefined threshold:

- it starts by listing all single-itemset, single-item frequent patterns
- then, it grows them by means of:
  - **s-extension**: a new itemset is enqueued to the pattern
  - **i-extension**: a new item is added to the last itemset
- the growing phase continues until the minimum support threshold is reached
- the algorithm is also capable of extracting **non strictly contiguous patterns** (gap tolerance between the itemsets, useful, e.g., for textual data)

VGEN, example, min. support 50%

Dataset of Sequences:

- A>(B,D)>C
- A>C
- A>(B,D)>C
- (B,D)>C
- A>B>C

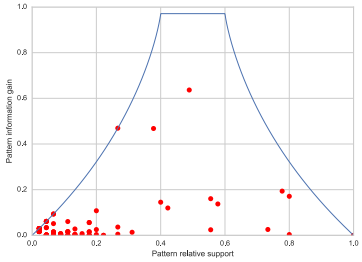
Generation example:

- **A** [4] $\rightsquigarrow_{s\_ext}$ A>C (1 < 50% if contiguous) .
  $\rightsquigarrow_{s\_ext}$ **A>B** [3] $\rightsquigarrow_{s\_ext}$ A > B > C [3] $\rightsquigarrow$ ...
- **D** [3] $\rightsquigarrow_{i\_ext}$ (B,D) [3] $\rightsquigarrow_{s\_ext}$ (B,D) > C [3] $\rightsquigarrow$ ...
- **C** [5] $\rightsquigarrow_{s\_ext}$ B>C [4] $\rightsquigarrow$ ...

Changes made to VGEN

We are not interested in the set of generator patterns, but rather in obtaining the **most discriminative one**:

- we relied on the **information gain** criterion: for a pattern, it may be calculated by partitioning the instances in the dataset into those that satisfy it (in which the pattern appears), and those which do not (*binary/boolean test*)
- we can prune the search space based on an upper bound on the information gain that a pattern can have



Patterns and information gain

Istance	Transcription	Label
1	hello>i>would>like>to>reserve>a>table	booking
2	hello>may>i>ask>for>a>table	booking
3	hello>morning>i>would>like>to>cancel>a>reservation	cancellation
4	hello>may>i>cancel>the>table>i>booked	cancellation

Here, the pattern “hello” has a lower information gain than the pattern “a>table”, which instead perfectly classifies the instances.

Changes made to VGEN: / 2

Sometimes, the most informative pattern might be too complex, and thus **overfit** training data:

- we store the best generator pattern (in terms of information gain) for each encountered pattern size
- at the end of the algorithm, we extract the pattern that **minimizes**:

$$W * (1 - pattern_{rIG}) + (1 - W) * pattern_{rlen}$$

- intuitively, the larger the user sets W, the longer and more accurate (on the training set) the extracted pattern will be, and vice-versa

Experiments: task description

We applied J48SS in the context of Contact Centers:

- we focused on the analysis of agent-side calls originated in the context of an **outbound survey service**
- in such situation, agents should complete a **predefined script** in order for a call to be considered successful
- a series of models are built, each capable of recognizing a different part of the script in a call conversation transcript, by attaching **tags** to each phrase
- this is extremely important to signal to the supervising staff **problematic calls**, i.e., the ones missing specific tags

Experiments: considered tags

12 tags have been considered, each capturing a specific step of the script:

age	*greeting_initial*	*profession*
call_permission	*greeting_final*	*question_1*
duration_info	*person_identity*	*question_2*
family_unit	*privacy*	*question_3*

Phrase (English)	**Tags**
Hello, my name is X and I am calling from X of X, am I talking with Mrs X?	greeting_initial, person_identity
You are retired. Last question... your birth date?	age, profession
Understood. May I ask you for your first name?	person_identity
May I? Thirty seconds, three quick questions...	duration_info

Experiments: data preparation

Experimental data:

- **4884 text chunks** (phrases delimited by silence pauses) transcribed from 482 distinct outbound calls
- split into training (75%) and test (25%) set (stratified per call)
- lowercase, stemming, stopwords and numbers removal
- each instance is characterized by the **chunk transcription**, and by a **list of boolean attributes** that track the presence or absence of each specific tag
- a dataset has been created for each tag:

Phrase (English)	**person_identity**
Understood. May I ask you for your first name?	true

Experiments: J48S results

Tag name	Accuracy	Precision	Recall
age	0.9655	0.9241	0.8171
call_permission	0.9167	0.6500	0.6454
duration_info	0.9739	0.9008	0.8516
family_unit	0.9722	0.8779	0.8712
greeting_initial	0.9739	0.9397	0.8195
greeting_final	0.9916	0.9823	0.9328
person_identity	0.9428	0.7851	0.6934
privacy	0.9916	0.9550	0.9550
profession	0.9840	0.8969	0.9063
question_1	0.9857	0.9688	0.9051
question_2	0.9815	0.9153	0.9000
question_3	0.9798	0.9237	0.8790
Macro average	**0.9716**	**0.8933**	**0.8480**

Experiments: model for tag *greeting_final*

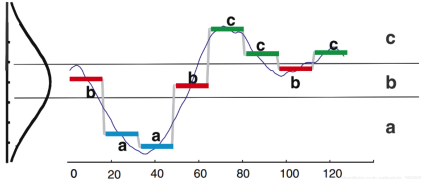
"Ok, we have finished. Thank you for the cooperation and good day, also on behalf of [company_name]. If you allow me, I inform you that [company_name] may recontact you."

```
transcription !contains recontact
|  transcription !contains thank>cooperation
|  |  transcription !contains inform: 0
|  |  transcription contains inform
|  |  |  transcription !contains inform>company_name: 0
|  |  |  transcription contains inform>company_name
|  |  |  |  transcription !contains allow: 0
|  |  |  |  transcription contains allow: 1
|  transcription contains thank>cooperation
|  |  transcription !contains ok>finish>thank>behalf
|  |  |  transcription !contains allow>inform
|  |  |  |  transcription !contains cooperation>day: 0
|  |  |  |  transcription contains cooperation >day: 1
|  |  |  transcription contains allow>inform: 1
|  |  transcription contains ok>finish>thank>behalf: 1
transcription contains recontact: 1
```

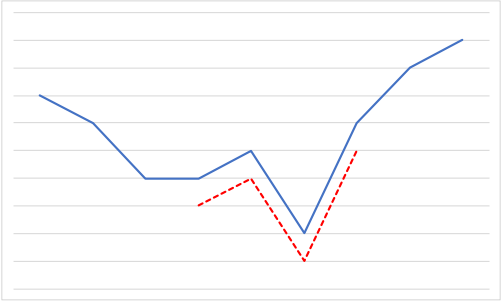
Time series classification

Time series classification is recognized as a difficult problem in the literature:

- commonly adopted solutions include data discretization (e.g., SAX) + sequential pattern extraction. Otherwise, extract general time series characteristics (e.g., mean, stddev):
  - information loss
  - less interpretable results
- alternative approaches rely on patterns that can be generated directly from the time series numerical data, such as **time series shapelets**



Time series shapelets

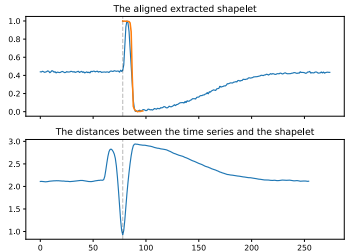


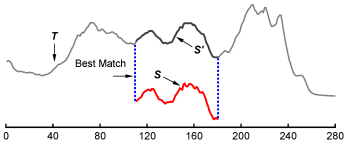
- Contiguous **time series subsequences** which are in some sense maximally representative of a class
- Meant to capture local properties of the TS

Why evolutionary algorithms

- Given n time series of length l, the possible contiguous subsequences that can be extracted are: $(\mathbf{n} * \mathbf{l}!)$
- Such a large space may be explored by means of **evolutionary algorithms**, i.e., adaptive meta-heuristic search algorithms inspired by the natural selection process
- **Multi-objective** EAs are particularly interesting, as they are capable of searching for multiple optimal solutions in parallel, wrt different desired properties

Time series shapelets for classification

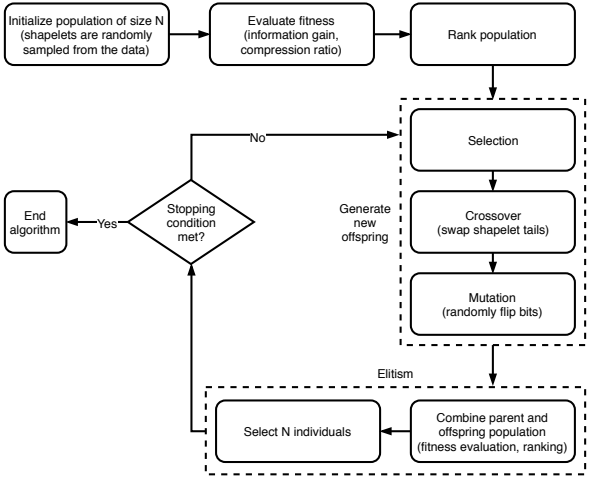




Shapelets can classify time series instances in a binary manner:

- given a shapelet, calculate its (e.g., Euclidean) distance w.r.t. all time series
- calculate a threshold to perform a binary split as you would do with a numerical attribute
- evaluate the information gain w.r.t. the classes

Overview of the EA



Algorithm implementation

We propose a solution based on the well-known **NSGA-II** multi-objective evolutionary algorithm.

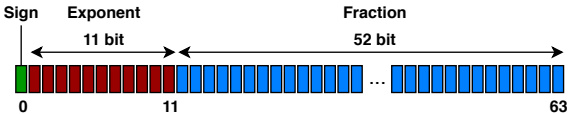
Need to define:

- representation of the individuals of the population, and their initialization strategy
- crossover, mutation, and selection operators
- fitness functions
- decision strategy, to select a single solution from the resulting Pareto front of optimal solutions

Population representation and initialization

Representation of an individual:

- each shapelet in the population is represented by an ordered list of binary arrays
- each binary array in the list corresponds to a floating point value, encoded in *IEEE 754 double-precision* 64 bit format



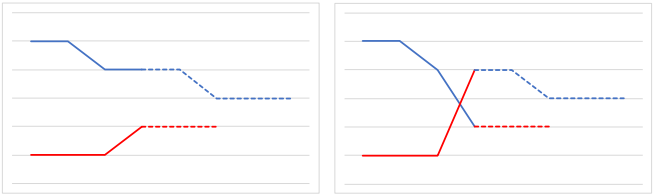
Initialization of an individual:

- a time series is randomly selected from the dataset, and then a begin and an end index are randomly generated
- the shapelet is then simply extracted from the portion of the time series that lies between the two indices

Design of the genetic operators

Crossover Given two shapelets:

- their two tail portions are randomly determined
- then, their tails are swapped



Mutation Given a shapelet, random flips are performed on the binary array representation of each of its elements, penalizing flips on the most significant (leftmost) bits

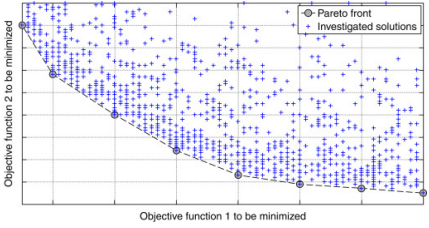
Selection The default NSGA-II binary tournament selection, based on the rank and crowding distance, is employed

Fitness functions

Two functions are to be optimized:

- **maximize** the *Information Gain* of the given shapelet:
  - calculate its Euclidean Distance wrt each of the time series
  - partition the time series with respect to a numeric threshold
- **minimize** the *functional complexity* of the shapelet (LZW compression rate), so as to reduce overfitting

Since there are 2 objectives, a Pareto front of optimal solutions will be generated.



Decision strategy

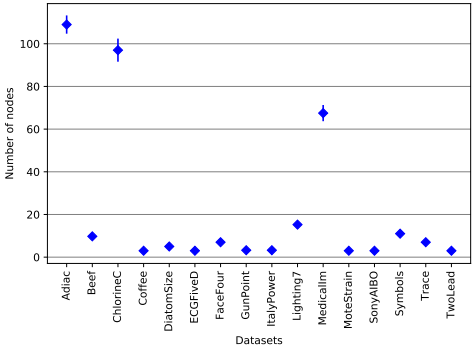
To select a single solution from the Pareto front, we minimize the formula:

$$(1 - W) * ComprRate_{rel} + W * (1 - InfoGain_{rel})$$

Intuitively,

- the larger the user sets W, the more accurate on the training set the extracted shapelet will be
- on the contrary, smaller values of W should result in less complex solutions being selected

Experiments (UCR): model interpretability




```
d(TS,[-0.21788,-0.21741,-0.21622,-0.21433,-0.20888,-0.20046,-0.19195,-0.18197,-0.15951,-0.14946,-0.13987,-0.036398,-0.04448]) <= 0.109855
|   d(TS_slope,[-0.080505,4.35E-4]) <= 0.01709: 2
|   d(TS_slope,[-0.080505,4.35E-4]) > 0.01709
|   |   d(TS_slope,[-0.0429]) <= 5.95E-4
|   |   |   kurtosis <= 0.197753: 5
|   |   |   kurtosis > 0.197753: 3
|   |   d(TS_slope,[-0.0429]) > 5.95E-4: 5
d(TS,[-0.21788,-0.21741,-0.21622,-0.21433,-0.20888,-0.20046,-0.19195,-0.18197,-0.15951,-0.14946,-0.13987,-0.036398,-0.04448]) > 0.109855
|   slope_avg <= -0.003305: 4
|   slope_avg > -0.003305: 1
```