n = | Ts |

k = # Ts

c = # candidati shapelet ( #motifs + #discords )

cs = # shapelet ( candidati usati effettivamente dall’albero. **NB cs << c** )

m = |sliding\_window|

**Costo computazionale funzioni**

1. DistanceProfile “MASS algorithm” (Ts) = **O**(n\*logn)
2. MatrixProfile “STOMP algorithm” (Ts) = **O**(n^2)
3. retrieveAll (Ts) = **O**(n^2)
4. candidateFilter (candidatesList) = **O**(c)
5. getDataStructures (datasetTrain) = **O**(k \* **O**(n^2) \* **O**(c) ) -> invoca retrieveAll e candidateFilter
6. computeSubSeqDistance (dataset, candidatesList) = **O**(k \* c \* **O**(n\*logn)) -> invoca distanceProfile
7. split (dataset) = **O**(k)
8. entropy (dataset) = **O**(k)
9. checkIfIsLeaf (dataset) = **O**(k)
10. getBestIndexAttribute (vecMutualInfo, candidatesUsedList) = **O**(c^2) -> per ogni attributo scandisco tutta candidatesUsedList
11. computeMutualInfo (dataset) = **O**(c \* k \* k \* k \* costoSort) = **O**(c \* k^3 \* costoSort) -> per ogni attributo, scandisco df intero, sort, split, calcolo entropia
12. findBestAttributeValue (dataset, candidatesUsedList) = **O**( **O**(c \* k^3 \* costoSort) + **O**(c^2) ) -> invoca getBestIndexAttribute e computeMutualInfo
13. fit & buildTree (dataset, candidatesUsedList) = **O**( **O**(c \* k^3 \* costoSort) + **O**(c^2) + **O**(k) ) -> invoca findBestAttributeValue e checkIfIsLeaf
14. computeSubSeqDistanceTest (datasetTrain, datasetTest, candidatesList, attributeList ) = **O**(k \* cs \* **O**(n\*logn)) -> invoca distanceProfile
15. predict (datasetTest) = **O**(k \* depthTree)