DATA MINING 2

# LEZIONE 1 (20/02)

* 4 moduli
* (lui può spacciare i libri)
* esame nelle stesse modalità della prima parte
* Stesso dataset della prima parte ma non esattamente uguale: dobbiamo rifare il data understanding ma non ci sono nè missing values nè errori, le features sono 434 ed estratte in modo diverso. Già diviso in training e test, possiamo mantenere la loro divisione.
* Francesco ci spiega come gestire i raw audio files

**Imbalanced learning**

Si parla di classificazione, abbiamo due predictive attribute e le due classi sono molto sbilanciate (tipo 98% e 2%). Quando c’è una grande discrepanza tra il numero di esempi presenti per ciascuna classe è difficile usare le stesse tecniche viste nella prima parte, sia per la classificazione che per la valutazione.

Questo viene trattato dal learning sbilanciato.

Loro chiedono di simulare questa situazione rimuovendo alcuni esempi.

Quando le classi sono sbilanciate di poco non serve bilanciarle, ma anche un dataset con 75% casi positivi, con un trivial classifier raggiungerebbe il 75% di accuratezza.

Ci sono 4 opzioni per gestire i dati sbilanciati:

1. Balance training set (NOT the test set) by oversampling the minority class or undersampling the majority class
2. Balance at the algorithm level: data remains unchanged but we adjust the class weight, the decision threshold or design a new algorithm
3. Switch to Anomaly Detection task: la classe minore è così piccola che viene considerata un’anomalia della classe maggiore
4. Do nothing and hope to be lucky, non sempre è la soluzione peggiore

Undersampling → si decide una strategia per selezionare una parte dei dati della classe maggiore per farla diventare della dimensione di quella minore. Può essere random (e in quel caso dovremmo fare il sampling più di una volta e fare la media) o usare altre tecniche. Condensed nearest neighbor (CNN): rivedi algoritmo nelle slide, si seleziona random un elemento e se è classificato bene si mette in grab, se è classificato male si mette in store. Alternative sono Tomek Links e Edited Nearest Neighbor or Cluster Centroids, che ha come debolezze il fatto che i centroidi non sono i punti reali.

With undersampling we are throwing away information 🙁

Oversampling → oltre a random (randomly replicating records, quindi abbiamo redundancy) ci sono SMOTE (randomly generates points between the points of the minority class, il problema è sempre che i punti non sono reali ma nemmeno plausibili perchè sono solo collegamenti tra punti già esistenti) e molte sue variazioni, tra cui ADASYN.

Adjust class weight → si crea una weight matrix che ci dice il costo di classificare male una delle due classi. Si può creare un modello che cambia i dati di training basandosi su una matrice di questo tipo

Adjust the decision threshold → possiamo personalizzarlo e ottenere performance diverse.

(Parte di laboratorio)

→ RIVEDI TUTTI GLI ALGORITMI

# LEZIONE 2 (21/02)

**Dimensionality reduction**: processo di ridurre il numero di variabili del dataset. Si considera divisa in feature selection (si selezionano alcune delle features originali) e feature projection (le variabili finali non sono necessariamente quelle originali). Tutto questo riguarda solo le colonne del dataset.

Feature selection può essere fatta con 3 strategie:

* filter strategy: si definisce un threshold e si tolgono quelle sopra o sotto. Questo può essere il variance threshold, univariate feature selection
* wrapper strategy: si decide quali sono le variabili più importanti dopo aver applicato un algoritmo (RFE\*)
* embedded strategy: features rimosse automaticamente durante la costruzione del modello.

Ci serve perchè classificazione e regressione possono essere fatte in modo migliore in uno spazio ridotto

\*Recursive feature elimination: dato un estimator esterno che assegna pesi alle features, questo seleziona le features ricorsivamente considerando set sempre più piccoli. L’estimator è inizialmente trained su tutte le features.

Feature projection/extraction può essere:

* lineare: PCA
* non lineare

Principal Component Analysis (PCA) → si trova un nuovo set di dimensioni che cattura meglio la variabilità dei dati. La prima dimensione cattura più variazione possibile. La seconda è ortogonale alla prima e cattura la maggiore variazione rimanente.

Trovare una nuova componente si può rappresentare con una linea. Dobbiamo trovare una linea che, quando i dati sono proiettati su di essa, ha il massimo della varianza, quindi minimizza il sum of squares del projection error.

Non si può fare con valori categorici.

Per la seconda componente dobbiamo trovare la retta ortogonale, facile in due dimensioni ma più difficile in un iperpiano.

* covariance
* eigenvector
* eigenvalue

Dobbiamo estrarre la PCA. O decidiamo il numero di dimensioni o definiamo un threshold per la variazione.

PCA è lineare e reversibile con qualche possibile errore.

Limiti: solo linear projections, che non è sempre il metodo migliore.

Partial Least Squares (PLS) → It is a supervised alternative to PCA. Cerca delle dimensioni ortogonali che spieghino sia outcome che features.

Random Subspace Projection → si scelgono dimensioni casuali e si fanno gli stessi calcoli. Molto veloce ma non si sa che cosa si massimizza o si trova. Otteniamo ogni volta risultati diversi.

Multi-dimensional Scaling (MDS) → mappa i dati in uno spazio di dimensionalità inferiore preservando le distanze relative. E’ una famiglia di metodi, e solitamente si cercano dimensioni come 2 o 3. Ci sono 3 gruppi: metric (usa metriche come distanze), non-metric (usa ranks invece di distanze), classic (usa euclidean distance). Non può essere invertita. Si calcola prima la matrice di dissimilarità e la dimensionalità. Si usa poi il gradient descent per risolvere un problema di ottimizzazione per minimizzare la funzione (dij = distance nello spazio originale | d2(xi, xj) = distance in the new space).

Sammon mapping is a generalization of the usual metric MDS, ma inserisce un sistema di pesatura che normalizza gli errori. Mantiene le distanze tra dissimilarities piccole e avvicina le distanze larghe.

Prima di fare molte di queste trasformazioni bisogna normalizzare (solitamente standard scaler).

Isometric Feature Mapping (IsoMap) → preserva la geometria dei dati. Usa le geodetic manifold distances between all pairs. Calcola la distanza euclidea tra le coppie di punti e poi usa un raggio per calcolare i vicini di un punto. Con queste informazioni crea un weighted graph tra i punti. A quel punto si calcola la geodesic distance tra tutte le coppie di punti (cioè il numero di hop che devo percorrere per raggiungere un altro punto, the shortest path on the graph). Si usa poi questa distanza nella formula J.

The most widely used approach for visualization is t-Distributed Stochastic Neighbor Embedding (t-SNE). While PCA tries to find global structure in the data, t-SNE tries to preserve the local structure. Preserve the notion of neighborhood. It is extremely slow.

SNE usa le distribuzioni invece delle distanze. La distanza tra due distribuzioni si calcola di solito con KL divergence. La distanza del vicinato si calcola con una misura di probabilità che dipende da sigma.

# LEZIONE 3 ()

outlier detection

# LEZIONE 4 ()

outlier detection

# 

# SUPPORT VECTOR MACHINE

Maximum margin hyperplane divide i dati sul piano in due parti. Per decidere se la retta migliore è B1 o B2, si considera quanto è vicina ai punti. Questo è importante perché le caratteristiche che cerchiamo sono una buona generalizzazione ed evitare l’overfitting. L’hyperplane migliore è quello che massimizza i margini, cioè le distanze tra i punti che deve separare la retta. I support vector sono i punti dei due gruppi più vicini alla retta.

SVM consiste nel trovare l’iperpiano che massimizza i margini tra i support vector points e trovare i punti che sono più utili per separare i dati. Gli altri punti non sono veramente presi in considerazione.

I nostri dati sono linearly separable se possiamo trovare una retta o un iperpiano che separi le nostre due classi di dati perfettamente.

Nelle slide: la linea centrale si trova ( vectorW dot product vectorX + bias = 0 )

Le altre linee parallele a quella centrale, che toccano i support vector si trovano ( vectorW dot product vectorX + bias = 1 ). Abbiamo bisogno di trovare vectorW e b (bias) e si fa attraverso un sistema.

DIstance tra uno qualsiasi dei punti e decision boundary → si trova calcolando la distanza tra punto e retta.

Distanza tra B1 e una delle due linee che tocca il support vector ci permette di trovare la grandezza del margine. Il margine (2 / ||vectorW||) deve essere massimizzato quindi dobbiamo minimizzare ||vectorW||.

Come troviamo w e b? Massimizziamo il margine minimizzando L(W)= ||W\*\*2||/2

E’ una funzione di loss. CI sono dei constraints da rispettare (sistema nelle slide)

w e b si ottengono con un optimization algorithm ottenuto con i precedenti constraints. Si possono usare diversi approcci matematici, in particolare Lagrange multiplier method, che sfrutta lambda o alpha. (formalizzazione nelle slide).

**Lineare SVM in non separable case**

Se usiamo la stessa idea iniziale, dobbiamo accettare degli errori nella nostra formulazione. In real data lenarly separable cases do not happen.

Inequality constraints devono essere smorzati. Si usa lo xi nell’equazione.

rette che toccano i support vector si trovano:

* vectorW dot product vectorX + bias = - 1 + xi
* vectorW dot product vectorX + bias = 1 - xi

xi è un errore del decision boundary rispetto agli esempi misclassificati.

slack variables (xi) = 0 per tutti i punti che sono nella parte giusta della separazione. Vogliamo che le slack variables siano il più vicine possibile allo zero. Nell’equazione il parametro più importante è C, the overall impact of the sum of the slack variables. The closest is C and the smaller is this impact.

Lagrangian multipliers (lambda or alfa) sono compresi tra 0 e C.

Quando abbiamo qualcosa che funziona cerchiamo di farlo funzionare anche per altro applicando delle piccole modifiche.

**Non linear SVM**

Abbiamo un problema non risolvibile in modo lineare in una dimensione, quindi facendo il quadrato proviamo a rappresentarlo su due dimensioni. A questo punto riusciamo ad effettuare una separazione lineare corretta. Lo stessi si può fare con un cerchio in due dimensioni che può diventare una parte di piano in tre dimensioni.

The trick is to transform the data from its original space x into a new space Φ(x) (phi) so that a linear decision boundary can be used. Decision boundary si trova: w•F(x) + b = 0

! la formulazione del nostro problema non cambia, tutti i calcoli rimangono uguali ma con la funzione phi aggiunta. Il numero di dimensioni aumentate però porta alla curse of dimensionality. Nella pratica molti calcoli nel problema di ottimizzazione hanno un dot product tra due record nel nuovo multidimensional space.

The Kernel trick → usa una funzione K che viene applicata a due istanze e ci restituisce il loro prodotto nello spazio a più dimensioni. Φ(x1) • Φ(x2) = K(x1, x2)

La funzione kernel viene espressa in termini di coordinate nello spazio originale.

Ci sono tre Kernel function principali: polinomial kernel, radial bases e sigmoid. Discusse nelle slide.

The linear SVC relies on inner product between the vector.

# NEURAL NETWORK

Neural network is an old model (‘70) e viene dalla metafora del neurone biologico.

ci sono molti input (le nostre features) con diversi pesi combinati insieme nel nucleo, che ritorna l’output dopo essere passato in n funzioni di attivazione.

# ENSEMBLE METHODS

si creano più dataset per il train, chiamati weak learners, che sono piccoli. quando facciamo i classificatori facciamo le query su tutti i weak learners e si sceglie la risposta più comune combinandoli.

Ci sono diverse strategie per aggregare le predizioni che determinano la differenza tra i metodi.

→ the wisdom of the crowds

la conoscenza collettiva di un gruppo di persone di solito è maggiore della conoscenza di uno dei singoli individui, e può essere raggiunta facendo la media tra le diverse opinioni. Possiamo pesare la rilevanza dell’opinione di alcuni soggetti (esperti e non). Solitamente la risposta collettiva è meglio di quella del singolo esperto.

4 bullets of a wise crowd.

ogni modello farà esperienza di aspetti diversi. alcuni weak model avranno risultati migliori in base alla qualità delle partizioni o della parte di dati che hanno visto.

base models/base estimator = weak models/weak estimators

BAGGING = Bootstrap AGGregatING (appunti nelle slide)

RANDOM FOREST is a particular case of bagging

BOOSTING

# 

Time Series

# Time Series Similarity

Una time serie è una collezione di osservazioni fatte in modo sequenziale nel tempo, solitamente a intervalli costanti. Non c’è uno standard nelle rappresentazioni in un dataset: possono essere sia nelle righe che nelle colonne. La singola osservazione è detta time stamp. Possiamo considerare serie temporali sia le misurazioni di qualcosa che cambia nel tempo, sia alcune osservazioni su video/immagini/testi.

I maggiori problemi con le time series:

* Large amount of data.
* Similarity is not easy to estimate.
* Different data formats.
* Different sampling rates.
* Noise, missing values, etc.

Tutti questi problemi riguardano un confronto di similarità. Ne esistono due tipi:

1. Similarità a livello di forma (shape) → per molto tempo ha dato risultati scarsi. Ad esempio, dati due oggetti la loro distanza (dissimilarità) è indicata con D(A,B). Per definire le misure di distanza, esse devono soddisfare alcune proprietà: simmetria, costanza, positività, trangular inequality.

Un esempio è la distanza Euclidea, che per ogni istante calcola la differenza tra una time series e l’altra e poi fa la radice della sommatoria delle differenze alla seconda. Il problema è che distorce i dati, ma si può risolvere con delle trasformazioni:

* Offset Translation → Entrambe le time series si trasformano in Q = Q - mean(Q). In un dataset normale si fa per ogni riga, in una dataset tabulare si fa per ogni colonna.
* Amplitude Scaling → Entrambe le time series si trasformano in Q = (Q - mean(Q)) / std(Q)
* Linear Trend → Consiste nel fittare la migliore riga dritta nella time series e poi sottrarla dalla time series stessa.
* Noise → Si fa Q = smooth(Q). Si fa la media di ogni punto con i suoi vicini.

Si può usare una ***Moving Average (MA)***: data una finestra di lunghezza w e una serie temporale t, otteniamo per i = 1, …,n. Quindi se la finestra w=3 noi prendiamo le prime tre misurazioni e facciamo la media, poi spostiamo la finestra e continuiamo. Da ricordare che più la finestra è grande e più informazione perdiamo all’inizio e alla fine.

1. Similarità a livello strutturale (structural level) → basata su strutture ad alto livello. Consiste nell’estrarre le features globali dalla serie temporale, creare un vettore di features e usarlo per misurare la similarità o classificare. Le features globali sono quelle classiche, es. min, max, mean, autocorrelation, ecc.

Un metodo per calcolare la similarità è la ***Compression Based Dissimilarity***, che consiste nell’usare come features tutte le strutture che un algoritmo di compressione riesce a trovare, con la formula d(x,y) = CDM(x,y) = C(x,y) / c(x) + c(y). L’obiettivo è salvare spazio mettendo insieme cose che si assomigliano.

Possiamo usare il ***Dynamic Time Warping*** quando due serie temporali sono concettualmente equivalenti ma evolvono a velocità diverse. Se l’asse del tempo rimane fisso, le sequenze si allineano una a una e la similarità può essere calcolata solo con la distanza Euclidea. E’ possibile qui fare un allineamento non lineare in modo da correggere alcuni errori nei dati.

Creiamo una matrice |C|x|Q| e la riempiamo con le distanze tra ogni coppia di punti delle due serie temporali. Mentre nella distanza Euclidea si possono usare solo le misurazioni sulla diagonale, nel DTW ci si può allontanare da essa nei punti che risulteranno spostati nelle due serie temporali. Ogni possibile warping è un percorso attraverso la matrice: si parte da (0,0) e ci si può spostare sia in (i+1, j+1) sia in (i+1, j) e (i, j+1) in base al costo minore. La fine è sempre nell’angolo in alto a destra.

Il Dynamic Programming Approach consiste nel calcolare la matrice di tutte le distanze, poi calcolare la cumulative cost matrix di tutti i costi dei percorsi e trovare il path con il valore minore (best alignment). Per la cumulative cost matrix si parte da (1,1) e si procede lungo la colonna verso l’alto secondo la formula g(i,j) = d(qi,cj) + min{ g(i-1,j-1), g(i-1,j ), g(i,j-1) }. Si procede per tutte le colonne.

Il problema di DTW è che è molto lento da calcolare, anche se otteniamo risultati migliori rispetto all’Euclidean. Esiste un'approssimazione più veloce di DTW che sfrutta alcune rappresentazioni compresse o un minor numero di esempi, poi effettuano la DTW su questa nuova rappresentazione. Si inseriscono dei limiti globali (Global Costraints), che appunto limitano gli indici del percorso wk = (i,j)k in modo che j-r <= i <= j+r , dove r definisce un range di valori permessi nel warping. Fa in modo che il path non si allontani troppo dalla diagonale. Si riduce in questo modo il numero di calcoli.

In conclusione, se abbiamo TS corte è meglio usare DTW dopo aver stabilito la migliore misura della finestra di warping. Se abbiamo TS lunghe e sappiamo qualcosa dei nostri dati, estraiamo le features. Se non sappiamo niente sui dati dobbiamo provare una compression/approximation based similarity.

# Time series Approximation

L’approssimazione delle serie temporali consiste nel rappresentarle in un nuovo spazio più piccolo e semplice e usare questa rappresentazione per fare i calcoli. E’ una speciale forma di dimensionality reduction apposta per le time series. La differenza con la compressione è che lo spazio approssimato è sempre comprensibile mentre quello compresso no.

Esempio: abbiamo una serie temporale di 128 punti, a cui corrispondono 128 numeri delle misurazioni. Con La Discrete Fourier Transform scomponiamo i dati in 64 onde pure, e si rappresentano i Coefficienti di Fourier come una colonna di numeri. In questo modo noi abbiamo solo cambiato rappresentazione, non diminuito le dimensioni.. Si nota però tra le nuove onde, le prime tendono ad essere le più larghe, e andando giù per la colonna i coefficienti tendono a diminuire. Possiamo quindi troncare i coefficienti più piccoli senza causare ingenti perdite. Possiamo anche scegliere di tenere solo le onde che hanno i coefficienti migliori, cosa che ha molti vantaggi in termini di qualità di approssimazione ma che rende impossibile mettere gli indici. Si ottengono i Truncated Fourier Coefficients.

La ***Discrete Fourier Transform (DFT)*** rappresenta la serie temporale come una combinazione lineare di seni e coseni che tiene conto solo dei primi n/2 coefficienti, perchè ogni onda sinusoidale ha 2 numeri, uno per la phase (w) e uno per l’ampiezza (A,B). Molti coefficienti di Fourier hanno un’ampiezza molto bassa quindi non sono molto utili a ricostruire il segnale. Per questo possono essere eliminati salvando spazio di memoria e causando poca perdita di informazioni. Pro: buono per segnali naturali, veloce in O(nlog(n)). Contro: ha difficoltà con sequenze di lunghezze diverse, non supporta misure di distanza pesate.

La ***Discrete Wavelet Transform (DWT)*** rappresenta la time series come una combinazione lineare di funzioni base Wavelet, ma tiene solo i primi N coefficienti. Wavelet rappresenta i dati in termini di somma e differenza di una funzione prototipica detta analyzing/mother wavelet. I coefficienti in questo caso sono localizzati nel tempo quindi alcuni rappresentano sottosezione dei dati piccole e locali, al contrario di Fourier che rappresenta sempre contributi globali. Pro: buono per segnali statici, tempo lineare. Contro: solo per sequenze con lunghezza che è un integrale di due, altrimenti l’approssimazione del lato sinistro è fatta a spese del lato destro.

La ***Singular Value Decomposition (SVD)*** rappresenta le serie temporali come combinazioni lineari di eigenwaves ma tiene solo i primi N coefficienti. I dati sono sempre rappresentati come combinazioni lineari di forme, ma le eigenwaves sono data dependent. Ruotiamo gli assi in modo che l’asse 1 sia allineato con la direzione della massima varianza e l’asse 2 con la direzione della massima varianza ortogonale all’asse 1. Le prime eigenwaves contengono la maggioranza della varianza del segnale quindi il resto può essere rimosso.

La ***Piecewise Linear Approximation (PLA)*** rappresenta le serie temporali come sequenze di linee dritte, o connesse (quindi sono ammesse K/2 linee) o disconnesse (quindi sono ammesse K/3 linee). Questo è uno dei tanti algoritmi che segmentano le serie temporali, ma la questione qui è come scegliere il K giusto, quindi il numero ottimale di segmenti. Questo coinvolge sia l’accuratezza che la compattezza, e non esiste una soluzione generica. I pro sono: compressione dei dati, filtro al rumore, supporta alcune misure di similarità non lineari.

La ***Piecewise Aggregate Approximation (PAA)*** rappresenta le serie temporali come una sequenza di funzioni base box con ogni box della stessa misura. La serie temporale è approssimata dividendola in segmenti della stessa lunghezza e registrando il valore medio dei punti di ogni segmento. Il vettore contenente tutte le medie diventa la nuova rappresentazione ridotta dei dati. Pro: veloce da calcolare, supporta misure non euclidee e misure euclidee pesate.

La ***Adaptive Piecewise Constant Approximation (APCA)*** permette ai segmenti di avere lunghezze arbitrarie, quindi diventa necessario avere due numeri per ogni segmento: il primo è la media dei punti del segmento, il secondo è la lunghezza. Bisogna considerare la struttura dei dati, comunque è in grado di mettere un solo segmento in un’area di bassa attività e molti segmenti in un’area di alta attività. Pro: veloce in O(n), supporta misure non euclidee e misure euclidee pesate.

Ci sono alcuni metodi per segmentare una serie temporale detti Change point detection, ad esempio il ***Symbolic Aggregate Approximation (SAX)***, che converte i dati in formato discreto. Ogni parte della rappresentazione contribuisce con lo stesso numero di informazioni alla forma della serie temporale. Come nella PAA si divide la serie in n segmenti della stessa misura e si calcola la media. La nuova rappresentazione è il vettore delle medie. Poi si determinano i breakpoints che dividono lo spazio in tot regioni equiprobabili. Ad ogni regione è assegnato un simbolo (lettera dell’alfabeto) e il numero di simboli è scelto a priori dall’utente. I breakpoint sono scelti in modo che la probabilità che un segmento ricada in una delle regioni sia uguale. I coefficienti PAA sono mappati nei simboli corrispondenti alle regioni. I simboli sono assegnati alle regioni dal basso verso l’alto: nella regione più bassa c’è la a.

# Time series Clustering

Il clustering è basato sulla similarità tra serie temporali e segue le regole solite del clustering. Ci sono due metodi:

1. **Partitional Clustering** → tipicamente usa K.means per ottimizzare la funzione obiettivo minimizzando la somma quadratica degli errori intra-cluster. Deve essere specificato il numero di cluster. Inoltre, la funzione di distanza gioca un ruolo fondamentale sia per la qualità che per l’efficienza del risultato.
2. **Hierarchical Clustering** → calcola la distanza tra coppie di serie temporali e unisce le coppie più simili. Non bisogna specificare il numero di cluster a priori. Crea un dendrogramma. La sua applicazione è limitata a piccoli dataset perchè ha una complessità quadratica.

Ci sono diversi tipi di clustering:

* **Whole clustering**: dato un set di time series, l’obiettivo è di raggruppare le time series simili nello stesso cluster
* **Features-based clustering**: estrae le features o i motifs e li usa per fare clustering
* **Compression-based clustering**: comprime le time series e fa il clustering sulla versione compressa
* **Subsequence clustering**: data una sola time series, il clustering è fatto su tutte le serie estratte dalla prima con una sliding window.

# Time series Matrix Profile, Motifs & Discords

Trovare dei motivi nelle serie temporali equivale a trovare dei pattern, delle sequenze ripetute. Le regole di associazione nelle serie temporali richiedono i motivi (motifs, o primitive shapes, o frequent patterns). Molti classificatori lavorano costruendo prototipi di ogni classe che possono essere considerati motivi. Molti algoritmi di anomaly detection nelle time series consistono nel modellare il comportamento normale con un set di forme per individuare pattern che se ne discostano.

Data una lunghezza predefinita del motivo, il metodo a forza bruta cerca i motivi in tutte le possibili sottosequenze. E’ lungo e costoso quindi ci sono delle alternative.

***Random Projection*** → si sceglie a caso una maschera e si usano i suoi valori per proiettare la matrice negli slot. Le collisioni sono registrate incrementando la collocazione nella matrice di collisione. Allo stesso modo si sceglie random un'altra maschera e si fa lo stesso. Alla fine le perturbazioni mostrano i motivi osservando la matrice in ordine decrescente di occorrenze. DIpende molto dalla tecnica di approssimazione adottata.

La **Matrix Profile** è una struttura dati che annota una time series e può essere usata per il motif discovery. Serve una time series e una lunghezza m della sottosequenza. Si usa una finestra scorrevole di lunghezza m per estrarre tutte le sottosequenze di lunghezza m, poi si calcola la distanza tra le coppie di sottosequenze. Per ogni sottosequenza si tiene solo la distanza con il vicino più vicino. Questa viene registrata in un vettore detto matrix profile P. L’indice del vicino in questione è registrato in un vettore detto matrix profile index. I puntatori nella matrice non sono necessariamente simmetrici (non è detto che se il punto A ha come più vicino B, allora anche B ha come più vicino A). Per valori molto bassi, sappiamo che la sottosequenza avrà almeno una sottosequenza simile da qualche parte nei dati, e queste regioni sono dette motivi. Per valori molto alti, la sottosequenza dovrà essere unica nella sua forma, e queste aree sono dette anomalie.

Inizialmente si inizializza la matrix profile come ettore inf, poi si sceglie una sottosequenza random di T e si calcola la sua distanza da tutte le altre sottosequenze e si inseriscono in un altro vettore. La distanza tra la sottosequenza e se stessa è 0. Si fa l’update della matrix profile inserendo la distanza minima di una sequenza dalla prima, dalla seconda, ecc. Alla prima iterazione matrix profile = primo vettore ottenuto. Alla seconda iterazione si sceglie per ogni posizione il minimo tra il valore della matrix profile e il valore del nuovo vettore. Alla fine, confrontando la serie temporale e la matrix profile, i minimi locali corrispondono ai motivi.

Possiamo visualizzare le serie temporali come punti in uno spazio m-dimensionale in cui le regioni dense corrispondono alle regioni in cui la matrix profile ha valori bassi. Scegliamo un parametro R con numero piccolo. Possiamo trovare le coppie più vicine di punti, dette motif pair, che corrispondono ai valori più bassi nella MP. Calcoliamo la distanza tra i nearest pair e disegniamo un cerchio di raggio R intorno ad entrambi. Tutti i punti che rientrano in quei cerchi sono aggiunti al motivo. Si va avanti per tutte le coppie di nearest pair. Il parametro K ci serve per fermarci nel cercare le coppie. Per trovare le anomalie, ci serve un parametro E di sottosequenze da escludere in prossimità dell’anomalia: si inizia a cercare la sottosequenza con la distanza maggiore nella MP e questa corrisponde alla più grande anomalia. Si cercano quindi le E più vicine sottosequenze all’anomalia e si rimuovono.

# Time series Classification

Dato un set di time series, ognuna ha m valori ordinati ed una classe. L’obiettivo è trovare una funzione che mappi dallo spazio di possibili time series allo spazio delle possibili classi. SI assume che tutte le time series abbiano la stessa lunghezza m.

Uno dei principali approcci è usare ***KNN sulle raw time series.*** Pro: semplice, DTW dà risultati migliori di Euclidean distance. COntro: alto costo computazionale e tempo, DTW è lento.

Altri classificatori si basano sulle shapelets, delle sottosequenze rappresentative. Per ottenerle si cambia la rappresentazione dei dati, non si ha più qualcosa di continuo nel tempo ma qualcosa che considera la shape e rappresenta una feature. Si considera ogni colonna come un pezzo della time series e si scrive il valore ottenuto confrontando quel pezzo con tutta la time series. La feature ottenuta è locale perché si confronta il singolo pezzo all’interno della propria ts. Gli ***Shapelet-based classifiers*** quindi partono da un dataset di serie temporali NxM, estraggono K shapelets, le trasformano e ottengono un dataset di shapelets NxK. Questo viene usato nel modello di ML.

Le shapelets possono dare dei risultati interpretabili che possono aiutare a comprendere meglio i dati. La distanza di una sottosequenza dalla serie temporale è calcolata come il minimo tra le distanze con tutte le sottosequenze della serie. Quindi sarà la posizione che più matcha la sottosequenza nella serie.

Di una shapelet può essere testata la sua utilità: si dispongono le sottosequenze nel dataset in base alla loro distanza dal candidato. Si deve trovare lo split point ideale che massimizza l’ottenimento dell’informazione, e si sceglie poi il miglior candidato che ottiene la maggiore utilità.

Date le proporzioni degli oggetti di una time series dispetto a due classi, si può calcolare l’entropia. L’informazione che rimane dopo aver diviso il dataset in due parti è definita dall’entropia media pesata di ogni sottoparte. Data una strategia di splitting e i valori dell’entropia prima e dopo lo split, si può calcolare l’information gain per quella regola di splitting.

Come splitting rule possiamo usare la distanza dalla serie temporale alla shapelet, ma calcolare questa distanza tra tutti i candidati è costoso, quindi si può ridurre il tempo in due modi:

1. **Distance Early Abandon** → riduce il tempo di calcolo alla sola distanza tra due time series. Usa solo la distanza minima tenendola a mente e sostituendola quando ne trova una migliore. Abbandona i calcoli se la distanza che sta calcolando è già maggiore del minimo.
2. **Admissible Entropy Pruning** → riduce il numero di distanze calcolate. Usa solo le migliori shapelets per ogni classe. Dopo aver calcolato qualche candidato, il limite superiore del guadagno di informazioni è minore del miglior shapelet candidato. Si fermano i calcoli e si prova con il prossimo

Esiste un metodo alternativo per estrarre le shapelets: la distanza minima tra una serie temporale e una shapelets può essere usata come predittore per approssimare la label della serie temporale usando un modello lineare. L’obiettivo è minimizzare una funzione di loss attraverso tutte le istanze.

A motif is a repeated pattern/subsequence in a given TS.

A shapelet is a pattern/subsequence which is maximally representative of a class with respect to a given dataset of TS.