PROGETTO PROVA FINALE -- PROGRAMMING WITH PYTHON

Studenti: Laura Proto, Giacomo de Gioia

§ 1 - OBIETTIVO

Il progetto, a partire dall'analisi del **database** zoo.csv[1], composto da 101 righe (animali) per 18 colonne (caratteristiche o *feature*), ha l'obiettivo di raggruppare gli animali in base alle feature in 7 cluster da confrontare con le classi fornite nel file class.csv[2], che chiameremo d'ora in avanti classi "reali". È richiesto un **approccio non supervisionato**, verranno pertanto utilizzate tecniche di clusterizzazione senza l'impiego negli algoritmi delle classi reali associate agli animali, con lo scopo di rivelare *pattern* nascosti nei dati **sulla sola base della similarità** definita tra i dati stessi. A conclusione del progetto procederemo alla **valutazione della bontà dei risultati ottenuti**, mediante l'uso dei dati contenuti nel file class.csv a valle dell'applicazione degli algoritmi per misurare e confrontare le performance[3].

§ 2 - ANALISI E SCELTE PROGETTUALI

§ 2.1 - Analisi preliminare dei dati

Il database è di dimensioni contenute: non sono presenti missing values e non ci sono incongruenze nei tipi di dato. Tutte le feature sono di tipo binario a parte la caratteristica *legs* che presenta valori da 0 a 8 (e class_type, cioè le classi reali che come suddetto utilizzeremo solo alla fine).

§ 2.1.1 - Analisi qualitativa dei dati

Un approfondito studio iniziale del database ha suggerito i seguenti interventi:

- traduzione automatica dall'inglese dei nomi di colonna e di riga con conseguenti correzioni o chiarificazioni ove necessario:
  – es. *hair*, traduzione automatica "capelli" → corretto in "pelliccia";
  – es. *bass*, traduzione automatica "basso" → corretto in "persico";
- correzione di errate valorizzazioni delle feature:
  – es. *seasnake* (= serpente di mare), non depone uova (*eggs*= 0) → corretto in depone uova (*eggs* = 1);
  – es. *clam* (= mollusco), non acquatico (*aquatic* = 0) → corretto in acquatico (*aquatic* = 1);
- disambiguazione di doppioni:
  – *frog* (= rana), presente due volte tra i nomi degli animali con differenza nella caratteristica *venomous* 0 e 1 → sdoppiata in *frog* con *venomous* = 0 e *frog2* con *venomous* = 1.

[1] Fonte dati: https://www.kaggle.com/uciml/zoo-animal-classification
[2] Stessa fonte dati della nota 1
[3] Non useremo dunque la SSE (*sum of the squared errors*), tipicamente utilizzata nei problemi di clusterizzazione per la valutazione delle performance, per due motivi: *i*) la SSE si usa necessariamente quando la classificazione finale non è predefinita o non è nota all'analista; *ii*) se la metrica utilizzata nell'algoritmo non è euclidea, la SSE, che di quella metrica è espressione, è indicatore inappropriato.

§ 2.1.2 - Analisi quantitativa dei dati

Trattandosi di variabili binarie, eccezion fatta per *legs*, le statistiche descrittive di base (media, deviazione std, min, max, percentili) hanno contenuto informativo solo limitatamente alla media – e, di conseguenza alla deviazione standard, nel senso qui specificato in calce[4] –, in quanto minimi, massimi e percentili sono evidentemente e necessariamente ridotti essi stessi a 0-1.

§ 2.2 - Analisi e selezione dei modelli

I principali algoritmi di clusterizzazione non supervisionata sono:

- **K-Means**
- **Clusterizzazione gerarchica,** considereremo il tipo **agglomerativo**
- **DBScan**

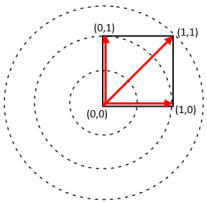
Tutti questi modelli partono dalla specificazione di una particolare **nozione di similarità** tra i punti nello spazio delle feature:

Algoritmi:	*Similarità intesa come:*
K-Means	vicinanza a punti prototipo (cd centroidi)
Agglormerative clustering	vicinanza tra i punti senza uso di prototipi
DBScan	vicinanza tra i punti entro un determinato raggio, senza uso di prototipi

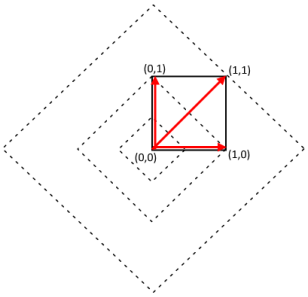
similarità che, come si coglie dalla tabella, richiede a sua volta l'impiego di una metrica.

§ 2.2.1 - Nozioni di distanza. Pregi e difetti.

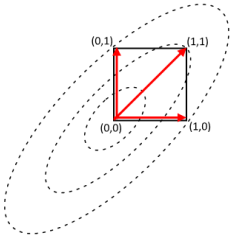
Abbiamo preso in considerazione le seguenti distanze, qui esemplificate graficamente in una regione di punti $[0,1]^2$ che richiama il database in esame (nei primi tre grafici le orbite tratteggiate rappresentano punti di equidistanza dall'origine $(0,0)$):



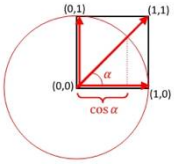
distanza euclidea



distanza Manhattan



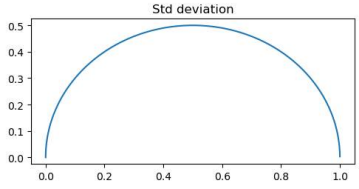
distanza di Mahalanobis



distanza coseno

- la **distanza euclidea** $\left(\sqrt{\sum(x_i - y_i)^2}\right)$ si caratterizza per l'isotropia, l'indipendenza della misura dalla direzione nello spazio, che può essere un pregio o un difetto a seconda dei

[4] È immediato verificare che, essendo X variabile a valori 0-1, risulta $\sigma_X = \sqrt{\mu_X - {\mu_X}^2} = \sqrt{\mu_X(1-\mu_X)}$, cioè σ_X completamente determinata dalla media. In particolare, vista come funzione di μ, σ descrive una semicirconferenza di centro $(0.5, 0)$ e raggio 0.5.

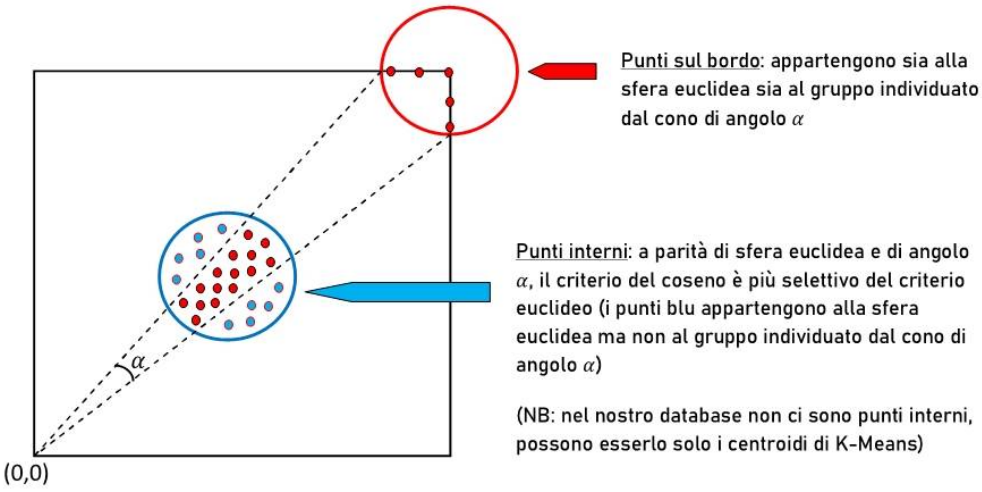


casi: un difetto, quando – come nel nostro caso – tende a raggruppare punti nello spazio delle feature che in realtà, per semantica, dovrebbero appartenere a gruppi distinti;

- tale problema non trova soluzione se si utilizza la **distanza Manhattan**, che, nel caso del nostro database, risulta essere banalmente una trasformazione strettamente crescente della stessa distanza euclidea[5]:

$$\sqrt{\Sigma(x_i - y_i)^2} \quad \underset{\text{=}}{\overset{x_i, y_i \in \{0,1\}}{}} \quad \sqrt{\Sigma|x_i - y_i|} \quad \Leftrightarrow \quad d_{eucl} = \sqrt{d_{Manh}}$$

- la **distanza di Mahalanobis** $\left(\sqrt{(x-y)^T V^{-1}(x-y)}\right)$ consente invece di tener conto di direzioni preferenziali nello spazio a seconda della correlazione tra le feature, facendo uso nella misura della matrice di varianza-covarianza (V), e precisamente della sua inversa (V^{-1}): quest'ultima si può interpretare come una sorta di reciproco dell'affinità tra i punti;
- molto interessante, in generale, è l'utilizzo della **distanza coseno** $(1 - \cos(\widehat{xy}))$, che privilegia nel raggruppamento i punti che nello spazio delle feature hanno un minore angolo compreso tra i rispettivi raggi vettori $(\widehat{xy})$: al ridursi di tale angolo, infatti, i punti tendono ad allinearsi geometricamente nello spazio rispetto all'origine, vale a dire che tendono a una condizione di lineare dipendenza e che dunque sono "simili" a meno di un fattore di scala. Tuttavia, sorge il sospetto che nel caso particolare del database in esame il vantaggio di questa misura, rispetto per es. alla distanza euclidea, potrebbe essere attenuato dalla circostanza che nel nostro spazio delle feature i punti sono dislocati sul bordo, e dunque "lontano" dall'origine degli assi coordinati:



Tale sospetto è stato poi confermato da alcune prove condotte in Python sul modello di clustering agglomerativo.

In definitiva, alla luce di queste riflessioni e di varie preliminari sperimentazioni in Python, abbiamo deciso di portare alla valutazione finale tre diversi modelli con due diverse distanze:

1. **K-Means** con **distanza euclidea**;
2. **Agglomerative clustering** con **distanza euclidea**;
3. **DBScan** con **distanza di Mahalanobis**.

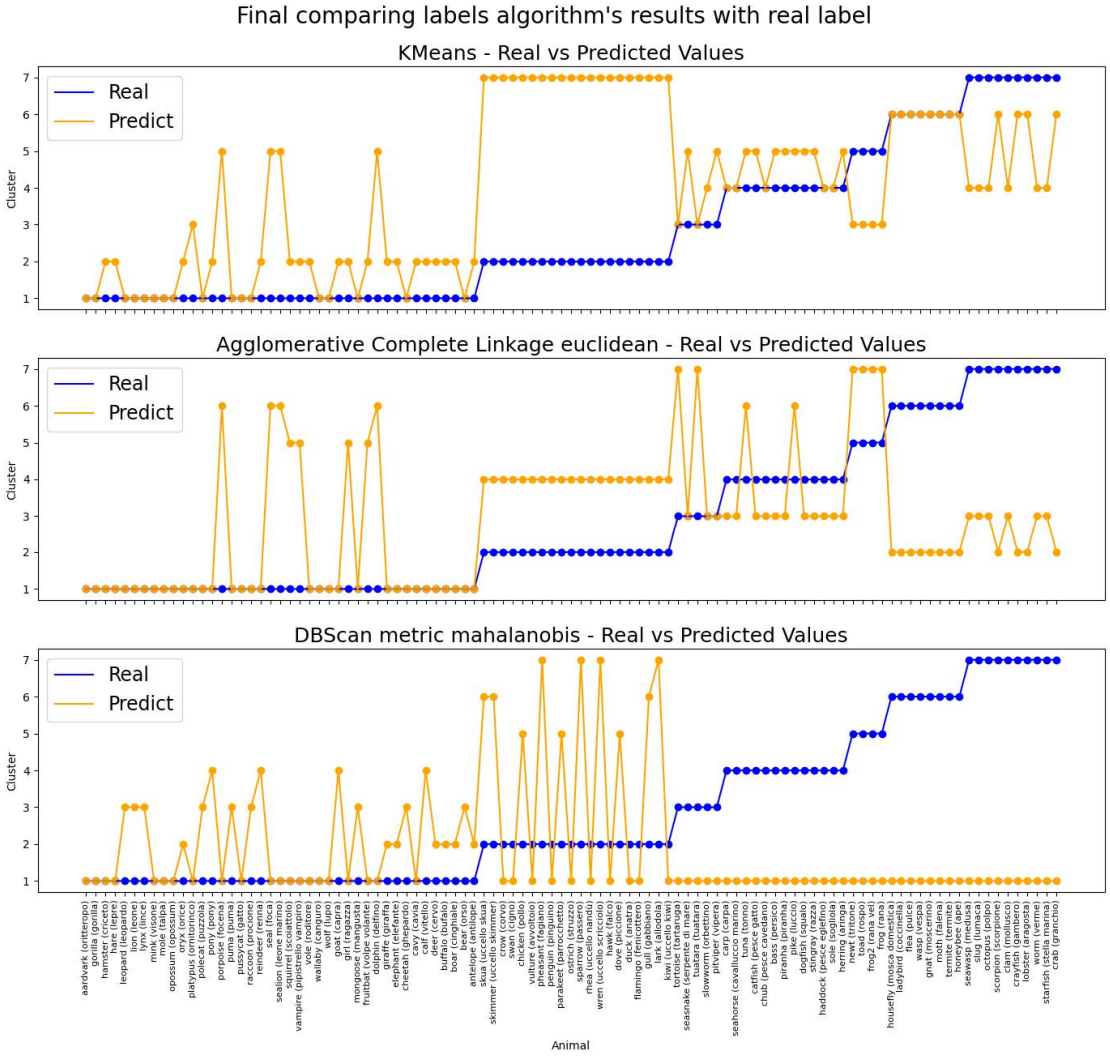
[5] Stiamo qui trascurando, per semplicità, la feature *legs*.

§ 3 - IMPLEMENTAZIONI E RISULTATI

§ 3.1 - Confronto risultati finali. Criteri di valutazione.

Il risultato finale dei vari algoritmi consiste nel raggruppamento dei campioni (gli animali, nel nostro specifico caso) in cluster che soddisfano i criteri di similarità fissati dagli algoritmi stessi. Si tratta in definitiva di ottenere partizioni dell'insieme degli animali – siamo in ambito *hard clustering* –, che confronteremo con la partizione data nel file class.csv. Qui si pone preliminarmente il problema di stabilire cosa significhi esattamente confrontare due partizioni, definire quando e in che termini – a parte il caso banale della coincidenza – due partizioni si possano dire più o meno "simili".

Per iniziare, abbiamo fatto un **confronto visivo** tra le classi reali e quelle ottenute dai tre algoritmi oggetto d'esame:

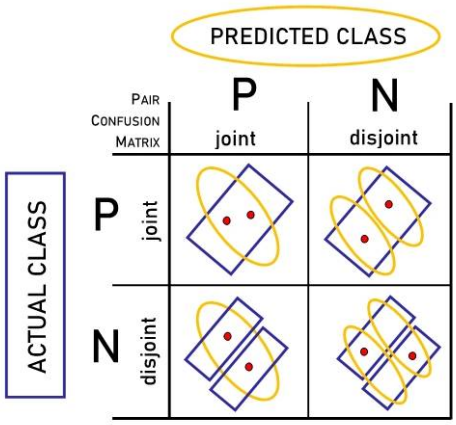


Questi grafici (x = animali, y = label dei cluster, nella scala puramente convenzionale 1, 2, ..., 7) mostrano la riallocazione dei campioni per effetto della nuova clusterizzazione derivante da ciascuno dei modelli rispetto alle classi effettive.

Si notano alcuni aspetti interessanti:

- la scarsa capacità di discriminare di DBScan, se si guarda alla parte destra dell'ultimo grafico, nel quale si nota che ben cinque classi reali risultano accorpate in un unico cluster risultato dell'algoritmo;
- nei primi due grafici emerge su ogni altra cosa la conferma della classe reale n. 2 come cluster risultato in entrambi gli algoritmi, K-Means e Agglomerative.

Anche se nessuno dei modelli è riuscito a replicare le classi date, già da questa prima ispezione visiva si coglie che la scelta del modello previsivo migliore porterebbe senz'altro ad **escludere innanzitutto DBScan**. Occorre tuttavia il conforto di una valutazione quantitativa oggettiva, la quale – tornando al tema del confronto di partizioni di un insieme – è fornita, tra le metodologie più usate in questo ambito, dall'analisi della cd **pair confusion matrix**.



La pair confusion matrix, qui di fianco schematicamente raffigurata, è una matrice $2x2$ contente il conteggio delle coppie di campioni che risultano:

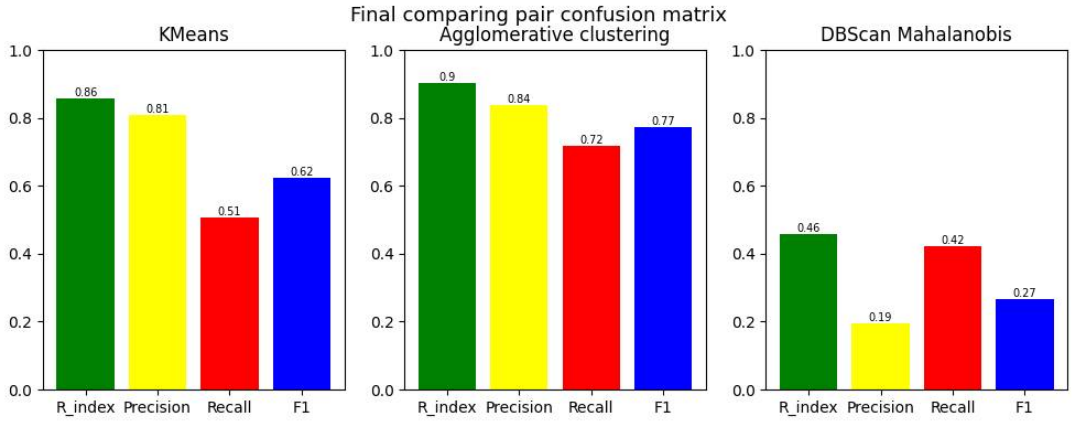
- nella stessa classe *actual* e nella stessa classe *predicted* (caso TP, *true positive*);
- nella stessa classe *actual*, ma in classi *predicted* distinte (caso FN, *false negative*);
- in classi *actual* distinte, ma nella stessa classe predicted (caso FP, *false positive*);
- in classi *actual* distinte e in classi *predicted distinte* (caso TN, *true negative*).

Dagli elementi di questa matrice[6] si ottengono i principali indicatori di "distanza" tra partizioni di uno stesso insieme, che nel nostro contesto si prestano immediatamente ad essere interpretati come "distanza" tra clusterizzazioni di uno stesso insieme di campioni e dunque, in ultima analisi, come **misure di performance di un algoritmo di clusterizzazione rispetto a una classificazione obiettivo**. Questi indicatori sono:

- il **Rand index**: $\frac{TP+TN}{tot.\ matrice}$, un indicatore di accuracy dell'algoritmo;
- la **precision**: $\frac{TP}{TP+FP}$, un indicatore di affidabilità predittiva dell'algoritmo;
- la **recall**: $\frac{TP}{TP+FN}$, un indicatore di affidabilità predittiva condizionata dell'algoritmo;
- **F1**: $2 \times \frac{precision \times recall}{precision+recall}$, la media armonica dei precedenti due, è un indicatore di sintesi solitamente utilizzato nella scelta definitiva tra algoritmi.

[6] Occorre avvertire che nella libreria `sklearn` di Python la pair confusion matrix si trova rappresentata in forma $\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$, anziché, come sopra, $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$.

Calcolati tutti e quattro questi indici per ciascuno dei modelli in esame, presentiamo graficamente i risultati:



Come si vede, e come già anticipato, **l'algoritmo DBScan con distanza di Mahalanobis esibisce una performance scadente** per tutti e quattro gli indicatori rispetto agli altri due modelli, K-Means con distanza euclidea e Agglomerative clustering con distanza euclidea e linkage complete. Dal confronto di questi ultimi, emerge che l'**Agglomerative presenta una migliore performance per tutti e quattro gli indici rispetto al K-Means**: in particolare è da osservare che in termini di performance generale i due modelli risultano tutto sommato equiparabili (*Rand index* 0,9 vs 0,86; *precision* 0,84 vs 0,81), tuttavia ciò che maggiormente penalizza K-Means rispetto all'Agglomerative è la *recall*, molto più bassa per K-Means (0,51 contro 0,72 dell'Agglomerative, significa che tra le coppie di campioni che risultano nella stessa classe reale, K-Means ne rialloca in cluster distinti più di quanto non faccia l'Agglomerative), e questo naturalmente si riflette nella media F1 (0,62 per K-Means, 0,77 per l'Agglomerative).

<u>In conclusione</u>: dal nostro studio emerge che l'algoritmo che produce la clusterizzazione più vicina alla classificazione attesa è l'Agglomerative Clustering con distanza euclidea e linkage complete, con Rand index del 90% e F1 del 77%.

§ 3.2 - Semantica dei cluster

Per quanto riguarda le caratteristiche distintive dei cluster prodotti dall'Agglomerative clustering, osservando il grafico che pone a confronto le etichette delle classi reali, date, e quelle prodotte dall'algoritmo, etichette "predette", è stato possibile ricostruire un certo contenuto semantico dei nuovi cluster che qui proponiamo in ultima colonna della seguente tabella:

COMPOSIZIONE DEI CLUSTER PREDETTI				
Label	#	*composizione in termini delle classi reali*		*significato dei cluster predetti (una proposta)*
1	33	33	33/41 mammiferi	mammiferi terrestri quadrupedi
2	12	8 + 4	tot insetti + 4/10 invertebrati	invertebrati terrestri e acquatici
3	20	3 + 11 + 6	3/5 rettili + 11/13 pesci + 6/10 invertebrati	pesci (?)
4	20	20	tot uccelli	uccelli
5	4	4	4/41 mammiferi	mammiferi terrestri bipedi
6	6	4 + 2	4/41 mammiferi + 2/13 pesci	mammiferi acquatici (?)
7	6	4 + 2	tot anfibi + 2/5 rettili	ovipari acquatici
totale	**101**			(?) *qualche mismatching*

□