Classificatore di corpi celesti

Celestial-Project

Componenti del gruppo:

Caterina Lafirenze

Matricola: 783966

e-mail: c.lafirenze@studenti.uniba.it

Link GitHub:

https://github.com/CaterinaLafirenze/ICON-Progetto.git

Anno Accademico 2024-2025

Docente: Prof. Nicola Fanizzi

Indice

1. Introduzione

Il progetto si propone come obiettivo quello di classificare i corpi celesti tramite un dataset primario (SDSS DR17) apprendimento supervisionato per i corpi che possiedono già un etichetta ben definita e il ragionamento logico in Prolog per tutti i casi in cui si riscontra ambiguità.

La seconda fase del progetto prevede l'utilizzo di un dataset secondario (Gaia Data Release 3) che è stato classificato tramite apprendimento non supervisionato.

Lo scopo principale nell'utilizzo delle due metodologie è quella di classificazione in primo luogo e secondariamente poterne confrontare l'accuratezza tramite media e distribuzione standard.

La realizzazione del progetto è stata effettuata con Python tramite l'IDE PyCharm.

Le librerie importate e utilizzate sono:

- pandas: per caricare e manipolare i dati dai file .csv;
- numpy: fornisce funzioni matematiche complesse;
- matplotlib e seaborn: per la visualizzazione tramite grafici;
- pyswip: interfaccia che connette Pyhton con il motore di ragionamento Prolog;
- scikit-learn nello specifico:
    - sklearn.model_selection: per l'utilizzo di k-fold utile a dividere i set di addestramento e test;

- sklearn.ensemble: contiene algoritmo di apprendimento Random Forest utilizzato nella sezione supervisionata;
- sklearn.preprocessing: per l'elaborazione dei dati utilizzando MinMaxScaler;
- sklearn.cluster: per gli algoritmi di clustering della parte non supervisionata;
- sklearn.metrics: per la valutazione dei modelli.

2. Creazione del Dataset principale

Il dataset per l'apprendimento supervisionato è stato acquisito tramite i dati registrati da Sloan Digital Sky Survey , il link di riferimento per reperire il dataset è il seguente: https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17 (link incluso a causa delle grandi dimensioni, dunque non visualizzabile in GitHub); e si presenta in questo modo:

obj_ID	alpha	delta	u	g	r	i	z	run_ID
1237660961327743232	135.6891066036	32.4946318397087	23.87882	22.2753	20.39501	19.16573	18.79371	3606
1237664879951151360	144.826100550256	31.2741848944939	24.77759	22.83188	22.58444	21.16812	21.61427	4518
1237660961330430208	142.188789562506	35.5824441819976	25.26307	22.66389	20.60976	19.34857	18.94827	3606
1237663478724297984	338.741037753146	-0.402827574587482	22.13682	23.77656	21.61162	20.50454	19.2501	4192
1237680272041378048	345.282593210935	21.1838656010284	19.43718	17.58028	16.49747	15.97711	15.54461	8102
1237680272039609088	340.995120509191	20.5894762801019	23.48827	23.33776	21.32195	20.25615	19.54544	8102
1237678858481565952	23.2349264301638	11.4181876197835	21.46973	21.17624	20.92829	20.60826	20.42573	7773

Nello specifico

- u: ultravioletto,
- g: verde,
- r: rosso,
- i: infrarosso,
- z: infrarosso,

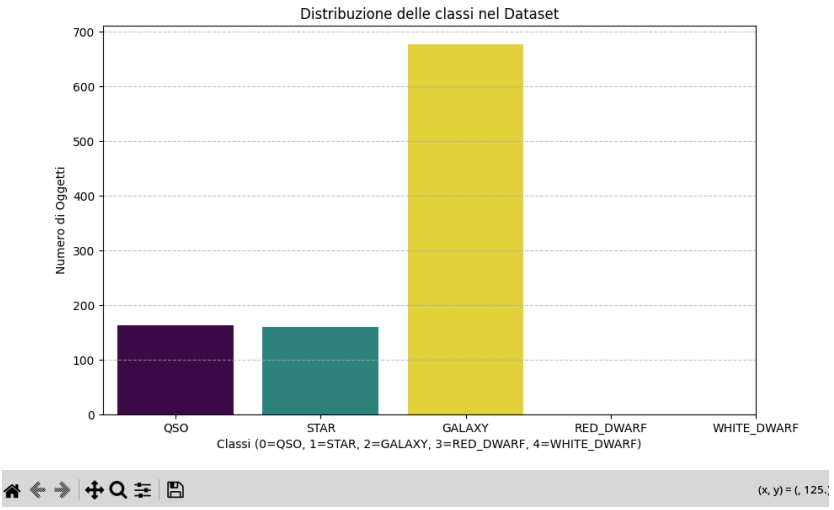
ognuna di queste sezioni indica delle bande di colore (alcune essendo molto ampie sono suddivise es.infrarosso) utilizzate per la classificazione delle stelle.

Pre-processing dataset

Questa fase avviene nel modulo [sdss_supervised_classifier.py] si occupa di trasformare il dataset grezzo SDSS in un formato contenete feature adatte all'apprendimento supervisionato.

Questa operazione è svolta dalla funzione load_and_preprocess_data che si occupa di svolgere le seguenti operazioni:

- carica e riduce le dimensioni del dataset, essendo questo composto da circa 100000 elementi lo riduce con un valore più semplice da gestire ad esempio 1000;
- rimuove le colonne superflue che potrebbero compromettere l'analisi dei dati ad esempio obj_ID, alpha, delta, ecc dato che non hanno nessun ruolo nella fase di classificazione;
- visualizzazione della distribuzione delle classi nel dataset tramite istogramma:



- mappa le etichette STAR, GALAXY, QSO... con valori numerici 0, 1, 2 dato che l'algoritmo Random Forest utilizza dati numerici e non stringhe;
- normalizza le feature con MinMaxScaler per scalarle in un intervallo tra 0 e 1, evita che il modello sia dominato da feature con valori troppo grandi.

3. Apprendimento supervisionato

L'apprendimento supervisionato utilizzato in questo caso rientra nella tipologia di Classificazione in quanto in output si avrà una associazione dato-classe.

In base alla scelta del modello di classificazione ho deciso di utilizzare un algoritmo di Ensemble Learning che combina le predizioni di un dato numero di semplici modelli, ognuno appreso da un learner di base e nello specifico utilizza la metodologia Bagging: Random Forest che crea molti alberi di decisione da cui il valore di output è ottenuto tramite una media su ogni predizione.

Gli iperparametri utilizzati sono ricavati tramite K-fold Cross Validation.

Tramite la funzione train_and_evaluate_model infatti le operazioni svolte sono:

- Utilizzo di cross-validation a 10 fold, questo evita le operazioni in un solo test che potrebbe essere solo parzialmente corrette.

  Quindi suddivide il dataset in 10 parti, su 9 si addestra mentre sulla 10 effettua il test ripetendo questo processo;

```
cv_scores = cross_val_score(model, X, y, cv=10, scoring='accuracy')

print("\n----------------------------------------------------")
print("Valutazione del Modello di Base (Cross-Validation):")
print(f"Accuratezza Media: {np.mean(cv_scores):.4f}")
print(f"Deviazione Standard: {np.std(cv_scores):.4f}")
print("----------------------------------------------------")
```

- Viene quindi riaddestrato il modello per l'analisi succesiva creando un dataframe apposito per i casi ambigui

```
X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.2, random_state=33)
model.fit(X_train, y_train)
```

```
ambiguous_indices = np.where(max_probabilities < confidence_threshold)[0]
ambiguous_cases = X_test.iloc[ambiguous_indices]
```

- ottiene i valori per calcolare accuratezza media e deviazione standard del modello per dimostrare che non si tratta solo di una suddivisione fortuita dei dati:

```
------------------------------------------------------
Valutazione del Modello di Base (Cross-Validation):
Accuratezza Media: 0.9630
Deviazione Standard: 0.0168
------------------------------------------------------

Identificati 55 casi ambigui con confidenza < 90.0%
```

Come si può notare dal risultato l'accuratezza è dello 0.96 e una deviazione standard dello 0.016.

Nonostante questi valori positivi vengono identificati "casi ambigui", si tratta di casi in cui la Random Forest da sola non riesce a classificare correttamente i dati che hanno confidenza inferiore al 90%, dunque tutto ciò che viene etichettato come ambiguo viene passato al ragionamento Prolog per correggere queste incertezze.

4. Ragionamento Logico e Prolog

Il modello ha identificato 55 casi ambigui con una 10-fold, si può supporre che sia dovuto al fatto che alcune caratteristiche siano a confine tra più classi dunque è necessario risolvere l'ambiguità.

La scelta dell'utilizzo del Prolog per risolvere tali ambiguità risiede nel fatto che la Random Forest è un metodo di apprendimento che effettua calcoli statistici e cerca degli schemi nei dati, i quali come possiamo notare non sempre sono chiari e definiti in un contorno, dunque la KB del Prolog contenendo regole permette di codificare la conoscenza in modo certo, quindi oltre ad effettuare una classificazione è in grado anche di giustificare la scelta proprio tramite le regole applicate.

Nello specifico ho optato dell'utilizzo del Prolog per i casi di ambiguità in modo che non fosse sovraccaricato con la quantità di dati del csv ma solo con il dataframe ricostruito dopo la prima classificazione, facendo in modo quindi che la Random Forest si occupi dei grandi dati e il Prolog effettui un perfezionamento.

Il file Prolog dunque è composto da un insieme di regole e fatti basati su indici di colore.

Gli indici di colore (come: u-g, g-r, r-i, i-z) sono differenze tra le magnitudini di due filtri diversi, nello specifico le regole riguardano:

- quasar(QSO): hanno indici bassi u-g
- stelle(STAR): hanno indici brillanti g e r e indici di colore più basso;
- galassie (GALAXY): indici più frequenti in r è i;
- nane rosse(RED_DWARF): indici nell'infrarosso r-i dunque molto alto;
- nane bianche(WHITE_DWARF): indici g-r indicati come negativi;

Dunque tramite la funzione resolve_ambiguities_whit_prolog si effettuano le seguenti operazioni:

- viene caricato il modulo [prolog_reasoner.pl] che contiene la KB;
- viene creato un ID univoco per l'oggetto e asserite le caratteristiche (ossia le magnitudini) dello stesso oggetto alimentando il ragionamento del prolog:

```python
for index, (case_id, row) in enumerate(ambiguous_cases.iterrows()):
    star_id = f"ambiguous_star_{index}"

    prolog.assertz(f"magnitude('{star_id}', u, {row['u']})")
    prolog.assertz(f"magnitude('{star_id}', g, {row['g']})")
    prolog.assertz(f"magnitude('{star_id}', r, {row['r']})")
    prolog.assertz(f"magnitude('{star_id}', i, {row['i']})")
    prolog.assertz(f"magnitude('{star_id}', z, {row['z']})")

    query = f"risolvi_ambiguita('{star_id}', FinalClass, Explanation)"
    solution = list(prolog.query(query))

    prolog_class = solution[0]['FinalClass'] if solution and 'FinalClass' in solution[0] else 'Indefinito'
    prolog_explanation = solution[0]['Explanation'] if solution and 'Explanation' in solution[
        0] else 'Nessuna spiegazione.'
```

- dopo aver effettuato la query e estratta la classe prevista viene effettua una valutazione e un conteggio sull'oggetto classificato correttamente confrontandolo con la precedente classificazione effettuata da Random Forest.

```
Esempio #3:
  - Caratteristiche: u=0.8040, g=0.6797, r=0.7381, i=0.8560, z=0.7634
  - Etichetta Reale: STAR
  - Previsione ML: STAR (Confidenza: 0.61)
  - Classificazione Prolog: QSO
  - Ragionamento di Prolog: Il suo indice u-g e' basso, tipico di un Quasar.
```

- alla fine restituisce il numero totale di casi che Prolog ha risolto;
- numero di casi risolti correttamente;
- lista della classificazione Prolog con confronto per ogni caso ambiguo.

```
Fold 3:
  - Casi ambigui passati a Prolog: 25
  - Accuratezza di Prolog sui casi ambigui: 0.2800
  - Accuratezza totale: 0.8100
```

5. Creazione del Dataset secondario

Per quanto riguarda il dataset secondario in questo caso è stato ottenuto tramite una query su Gaia Archive.

La scelta di utilizzare un dataset differente è dipesa dal fatto che per l'apprendimento non supervisionato ho deciso di utilizzare dati che non possiedono etichette esplicite come per SDSS.

Vengono create delle regole che permettano in fase di pre-processing di generare etichette utili a misurare la purezza del clustering.

Inoltre questo dataset verrà utilizzato nel confronto finale tra apprendimento supervisionato e non supervisionato.

Questo permetterà in secondo luogo di valutare l'apprendimento supervisionato non più solo tramite dati ben testati ma mostrando anche che è possibile applicarlo in svariati contesti.

```
SELECT
   TOP 10000
   source_id,
   ra,
   dec,
   parallax,
   phot_g_mean_mag,
   phot_bp_mean_mag,
   phot_rp_mean_mag
FROM
   gaiadr3.gaia_source
WHERE
```

```
    ra BETWEEN 266.0 AND 267.0
    AND dec BETWEEN -29.4 AND -28.4
    AND phot_g_mean_mag IS NOT NULL
    AND phot_bp_mean_mag IS NOT NULL
    AND phot_rp_mean_mag IS NOT NULL
    AND parallax_over_error > 5
```

Nella query si chiede di recuperare un certo numero di valori come:

- source id: identificativo univoco per stella;
- parallasse: utile a calcolare la distanza di una stella (>5 per evitare che il valore sia più significativo dell'errore);
- phot_g_mean_mag, phot_bp_mean_mag e phot_rp_mean_mag per le varie magnitudini.

Questi valori sono utili per etichettare in 5 classi principali (identiche a quelle del supervisionato) che includono: STAR, QSO, GALAXY, RED_DWARF, WHITE_DWARF.

Non sono utilizzati per addestrare il modello ma per valutarne l'accuratezza.

6. Apprendimento Non Supervisionato

All' interno del modulo [gaia_unsupervised_classifier.py] e qui seguono le fasi di:

- creazione degli indici di colore bp_rp e g_bp utili a rappresentare la differenza di magnitudine tra diverse bande di colore, sfruttate nella classificazione in quanto rappresentano il colore degli oggetti;

```
df['bp_rp'] = df['phot_bp_mean_mag'] - df['phot_rp_mean_mag']
df['g_bp'] = df['phot_g_mean_mag'] - df['phot_bp_mean_mag']
```

- etichettatura che prima suddivide per distanza (tramite valore di parallasse) e in secondo luogo suddivide ulteriormente in base a indici di colore, processo utile alla misurazione della purezza del cluster;

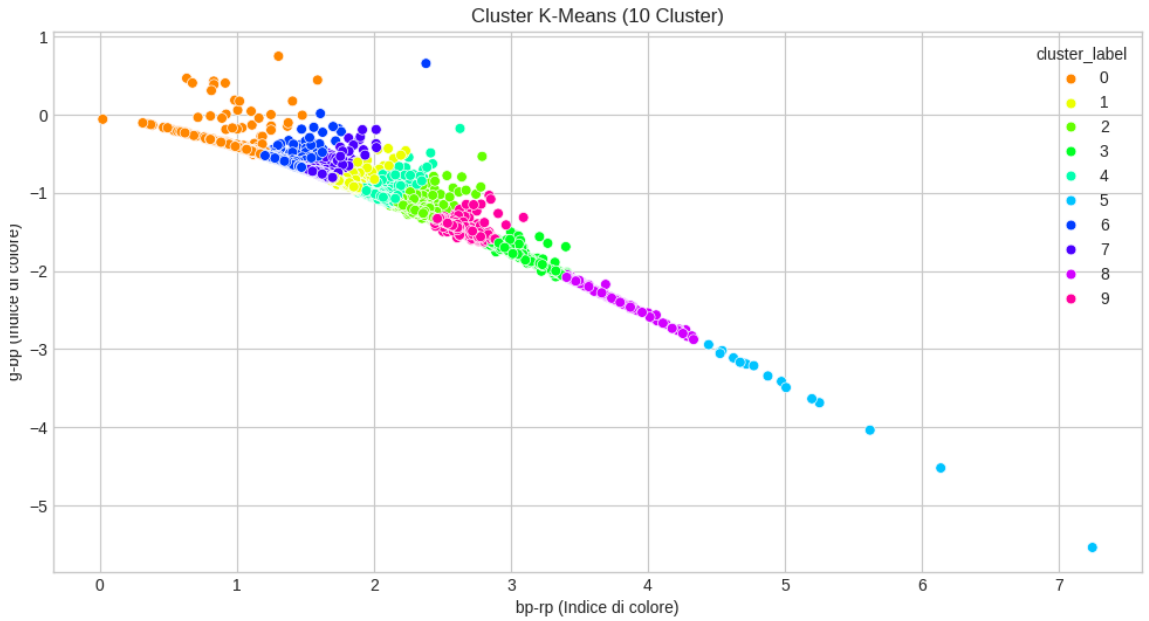
```
# Oggetti vicini (STAR, WHITE_DWARF, RED_DWARF)
df.loc[df['parallax'] > 0.1, 'class'] = 'STAR'
# Oggetti lontani (GALAXY, QSO)
df.loc[df['parallax'] <= 0.1, 'class'] = 'EXTRAGALACTIC'
```

- normalizzazione, anche in questo caso come nel caso dell'apprendimento supervisionato ho utilizzato la funzione MinMaxScaler. Questo processo è utile in quanto permette all'algoritmo K-Means di non essere dominato da feature specifiche;
- la funzione evaluate_clustering applica il K-Means per il raggruppamento dei dati, in questo caso è stato utilizzato un valore di 10 cluster scelto arbitrariamente.

  Potrebbe essere migliorato utilizzando la regola del gomito per visualizzare il punto in cui la curva non cambia più in modo visibile, andando a identificare il numero di cluster sufficiente a fare la valutazione;
- infine la purezza del clustering è utilizzata come metrica per valutare le prestazioni del modello di apprendimento non supervisionato.

  Per ogni cluster viene identificata la classe reale e tramite la media ponderata si ottiene la purezza del cluster.

Con questo grafico a dispersione è possibile comprendere come gli oggetti sono stati raggruppati nel modello K-Means in base alle feature (bp_rp e g_bp).



Non è stato inserito il grafo che visualizza le classi reali in quanto il risultato era pressoché identico e questo è dimostrato dal valore della purezza:

```
------------------------------------------------------
Analisi di Clustering Non Supervisionato (K-Means)
------------------------------------------------------

Purezza del Clustering: 0.8748

------------------------------------------------------
```

essendo dell’ 87% indica come mai il secondo grafico avrebbe riscoperto la classificazione a priori, dunque gli indici di colore per la distinzione dei corpi celesti risulta molto funzionale.

7. Confronto dei modelli e Conclusioni

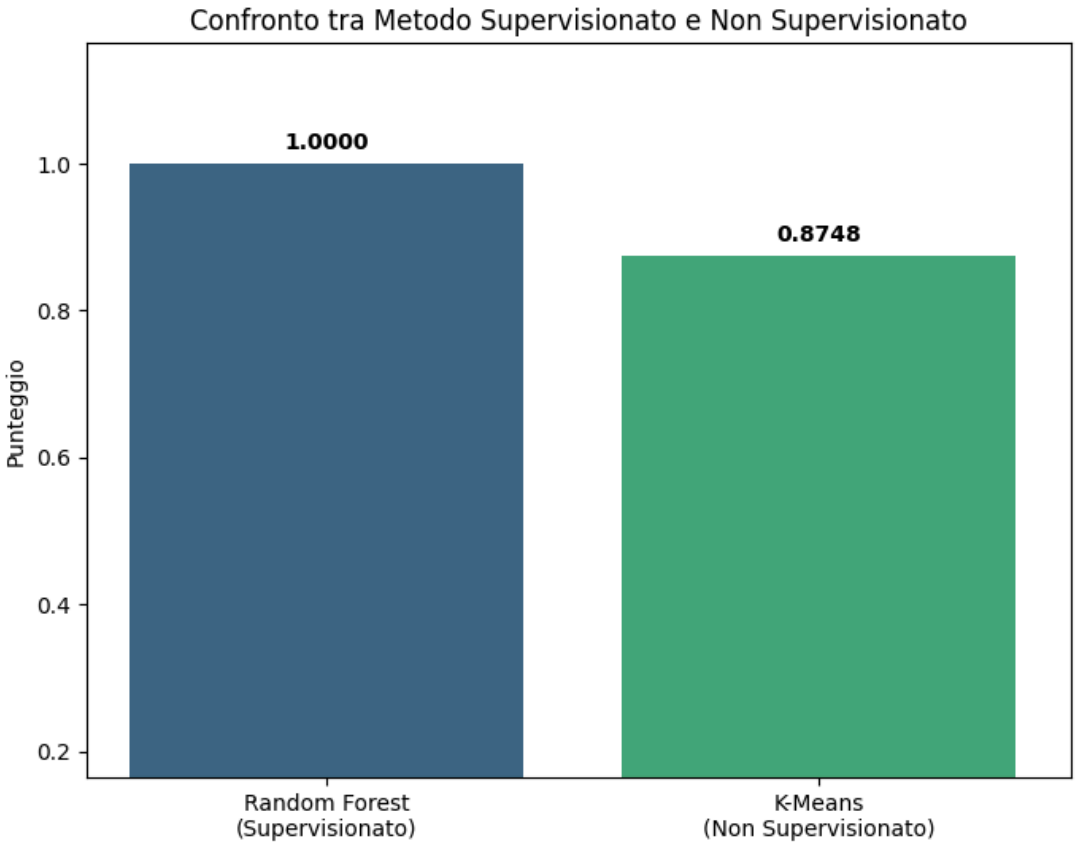
Nel modulo del confronto [supervised_unsupervised_comparison.py] si effettua il confronto diretto tra i due modelli sfruttando lo stesso dataset.

L'obiettivo è:

1. di valutare quanto il K-means sia in grado di effettuare una classificazione adeguata rispetto alla Random Forest pur non avendo conoscenza a priori;
2. sottolineare come l'apprendimento supervisionato sia efficiente anche con i nuovi dati caratterizzati da una struttura completamente diversa.

Come nel primo caso supervisionato anche in questo modulo si addestra e testa il classificatore Random Forest con un K-fold Cross Validation su 10 fold, e anche qui si utilizzano accuratezza media e deviazione standard per valutare i risultati ottenuti dal modello.

Per quanto riguarda l'apprendimento non supervisionato i risultati vengono importati semplicemente dal modulo appena descritto [gaia_unsupervised_classifier.py] rieseguendo il calcolo di purezza per il confronto.



Il grafico specifica i valori di accuratezza media.

Si può riscontrare che per il caso supervisionato la classificazione dei corpi celesti è totale, nonostante possa sembrare una contraddizione in quanto abbiamo usato il dataset di Gaia che risulta più “rumoroso” in quanto solo in secondo luogo fornisce valori sul colore della stella concentrandosi maggiormente sul parallasse.

Il motivo di questa ottimizzazione estrema è dovuta al fatto che nel modulo [gaia_unsupervised_classifier.py], nella fase di pre-processing, sono state create delle etichette e questo ha permesso al Random Forest di classificare correttamente tutte le stelle dato che gli sono stati forniti tutti gli strumenti.

Per quanto riguarda il caso non supervisionato la classificazione è del 87%, nettamente inferiore rispetto a quella del caso supervisionato proprio a causa della natura dei cluster.

8. Sviluppi

In conclusione con questo confronto si possono trarre numerosi spunti:

- si è dimostrato come in presenza di etichette l'apprendimento supervisionato sia più efficiente rispetto a quello non supervisionato e dunque la struttura dei dati è un punto fondamentale per raggiungere l'efficienza per qualunque modello;
- un possibile miglioramento potrebbe essere evitare la creazione di etichette o utilizzare più feature del dataset di Gaia per comprendere come reagiscono a questi nuovi input i modelli;
- implementare il metodo del gomito come già citato o alternativamente si potrebbero testare algoritmi differenti anziché quelli di clustering;
- un altro punto fondamentale riguarda la prima sezione del progetto che tratta esclusivamente l'apprendimento supervisionato, qui si è potuto riscontrare come il Random Forest in realtà non si comporti sempre in modo ottimale, anzi mostra il suo grado di incertezze e combinarlo con metodologie di ragionamento permette un miglioramento del risultato;
- possibili migliorie potrebbero essere effettuate proprio sulle regole del Prolog di modo che si riescano a classificare tutte le tipologie di corpi celesti (infatti in questo caso non sono stati considerati pianeti o esopianeti che potrebbero avere un ruolo tra i corpi che a nessuno dei due metodi è stato richiesto a classificare) o testare una Leave One Out Cross Validation e quindi anche in questo caso testare altri algoritmi.