AstroVerseIA Report

ADRIANO DE VITA and PELLEGRINO PICCOLO, Università degli studi di Salerno, Italy

CCS Concepts: • **Computing methodologies** → **Machine learning**.

Additional Key Words and Phrases: Random Forests, Recommendation Systems, Social Web Platforms, Machine Learning

ACM Reference Format:
Adriano De Vita and Pellegrino Piccolo. 2025. AstroVerseIA Report. 1, 1 (January 2025), 16 pages.

https://github.com/Adry04/AstroVerseIA

Authors' address: Adriano De Vita, adrianodevita8@gmail.com; Pellegrino Piccolo, piccolopellegrino22@gmail.com, Università degli studi di Salerno, Salerno, Italy.

ACM XXXX-XXXX/2025/1-ART
https://doi.org/

Contents

1 INTRODUZIONE

1.1 Scopo del sistema

Due studenti del dipartimento di Informatica dell'Università degli studi di Salerno vogliono ideare un progetto riguardante un modulo di Intelligenza Artificiale, chiamato AstroVerseIA. Il modulo verrà implementato per una nuova applicazione web, un Social chiamato AstroVerse. Lo scopo del progetto è quello di facilitare la connessione di gruppi di persone all'interno di spazi virtuali, dove gli utenti possono condividere post e idee su un argomento scelto dal creatore del canale.

1.2 Specifiche P.E.A.S.

1.2.1 ***P. Performance Measures****.* Le prestazioni del modello saranno misurate attraverso l'utilizzo di metriche di valutazione comuni per i sistemi basati su Machine Learning, quali:

- Precision;
- Recall;
- F1 Score;
- Accuracy.

1.2.2 ***E. Environment****.* L'ambiente in cui il modello opera e gli elementi che lo costituiscono sono:

- Dataset di suggerimenti generato attraverso data syntesis e prompt engineering sul modello GPT 4o di OpenAI;
- Sezione di registrazione dove vengono salvate le preferenze dell'utente;
- Sezione gestione spazi del social AstroVerse.

1.2.3 ***A. Actuators****.* Gli attuatori che prenderanno le decisioni all'interno del modello sono:

- Un ensemble di classificatori che andranno a stabilire se uno spazio può essere suggerito o meno ad un utente basandosi sulle sue preferenze.

1.2.4 ***S. Sensors****.* I sensori attraverso cui il modello riceve gli input percettivi sono:

- Le preferenze compilate durante la fase di registrazione;
- Le conoscenze apprese dal dataset riguardanti gli spazi da suggerire relativamente ai macro argomenti selezionati dall'utente.

1.3 Caratteristiche dell'ambiente

L'ambiente che andremo ad utilizzare presenta le seguenti caratteristiche:

- Completamente osservabile: il modello ha visione dell'intero dataset da cui apprendere le dipendenze del problema in esame;

- Deterministico: il modello stabilisce con certezza come la sua percezione dovrà influire sull'ambiente;
- Episodico: ogni scelta dell'agente è un episodio atomico e non correlato in alcun modo alle precedenti percezioni;
- Statico: l'ambiente utilizzato non viene aggiornato dopo l'output percettivo dell'agente
- Discreto: il numero di percezioni dell'agente è finito e chiaramente definibile;
- Multi-agente Cooperativo: Per risolvere il problema in esame sono state utilizzate più istanze di agente che operano su porzioni diverse dell'ambiente adempiendo allo stesso fine.

2 OBIETTIVI DI BUSINESS

2.1 Definizione degli obiettivi di business

Per suggerire in maniera efficace gli spazi, il modello deve riuscire ad effettuare delle **classificazioni** all'interno dell'ambiente social. Per stabilire se un certo spazio verrà o meno suggerito, esso si baserà principalmente sulle preferenze compilate dall'utente in fase di registrazione. Il modello, per funzionare, deve essere addestrato su un'adeguata quantità di dati, in modo da non ricadere in problematiche comuni di questo tipo di sistemi, quali:

- **Varianza**, il modello non deve cioè essere troppo rigido nel seguire le preferenze espresse dall'utente, ma deve invece bilanciare precisione e scoperta. Questo implica suggerire non solo gli spazi strettamente legati alle preferenze dichiarate, ma anche spazi correlati o potenzialmente interessanti, anche se non perfettamente coerenti con le preferenze iniziali. In questo modo, si garantisce un'esperienza di scoperta personalizzata, ma non limitata, promuovendo l'esplorazione di nuovi contenuti.
- **Bias**, quindi suggerire all'utente solo gli spazi suggeriti più frequentemente senza considerare in modo adeguato le preferenze specifiche dell'utente.

2.2 Criteri di successo

AstroVerseIA punta quindi a connettere in maniera efficace e originale i suoi utenti, consentendo loro di conoscere nuovi argomenti attraverso l'utilizzo di un modulo d'Intelligenza Artificiale. Il progetto ivi citato dovrà quindi:

- Permettere agli utenti di esprimere delle preferenze durante la fase di registrazione;
- Suggerire spazi agli utenti registrati al social basandosi sulle preferenze espresse dall'utente, favorendo la scoperta di contenuti di qualità.

2.3 Risorse utilizzate

In quanto la piattaforma social AstroVerse è ancora in fase di sviluppo locale e non è ancora stata pubblicata su un dominio web, ci aiuteremo nella generazione del dataset su cui l'algoritmo verrà addestrato, di un **LLM** ad oggi molto conosciuto, *ChatGPT4o* di OpenAI. Logicamente (e verrà esplicitato nelle prossime sezioni del documento) questi dati verranno accuratamente analizzati e successivamente modellati in modo tale da ridurre, quanto più possibile, la presenza di potenziali problemi di **bias** e **varianza**. Detto ciò, il modello verrà progettato interamente utilizzando:

- Il linguaggio di programmazione **Python v3+**;
- L'IDE **PyCharm** per il supporto alla programmazione in Python;
- Il framework **Django** che si occuperà della gestione delle richieste al server;
- Le librerie:
  - **scikitlearn** per l'implementazione dell'algoritmo utilizzato
  - **pandas** per la lettura del dataset e il calcolo della deviazione standard
  - **joblib** per l'esportazione e l'importazione del modello
  - **matplotlib** per la generazione di plot grafici
- L'ambiente **DataSpell** per effettuare modifiche rapide al dataset.

3 ACQUISIZIONE DEI DATI

Come descritto in precedenza, i nostri dati saranno generati sinteticamente. Il prompt consiste nell'istruire il LLM riguardo la comprensione del contesto di utilizzo del dataset. Una volta fatto ciò, richiedere la generazione del dataset strutturato in formato *.CSV* con 500 istanze.

3.1 Comprensione dei dati

Il dataset generato presenta le seguenti features:

- **id_utente**, che rappresenta il numero identificativo dell'istanza utente, questo attributo non verrà utilizzato per l'addestramento del modello;
- **macro_argomento**, che rappresenta i macro-argomenti su cui l'utente andrà ad esprimere le sue preferenze in fase di registrazione;
- **argomento_spazio**, che rappresenta i vari argomenti presenti sulla piattaforma AstroVerse;
- **suggerito**, che rappresenta il valore dato dalla classificazione (outcome), consisterà in un attributo binario true o false, a seconda di se lo spazio viene suggerito o meno.

Il codice per la generazione del dataset è:

```
1    import pandas as pd
2    import random
3
```

```
    macro_topics = [
        "Tecnologia e Ingegneria", "Finanza e Investimenti", "
            Salute e Benessere",
        "Cultura e Societa'", "Relazioni e Famiglie", "Hobby e
            Interessi Personali",
        "Scienze e Natura", "Arte e Creativita'"
    ]

    recommended_topics = [
        "Economia", "Matematica", "Fisica", "Scienze della terra"
            , "Biologia",
        "Geografia", "Storia", "Giornalismo", "Politica", "Sport"
            , "Programmazione",
        "Intelligenza Artificiale", "Web Design", "Cybersecurity"
            , "Cloud Computing",
        "Data Science", "Crypto", "Videogiochi", "Teatro", "
            Chitarra", "Musica",
        "Montagna", "Fotografia", "Cucina", "Letteratura", "
            Cinema", "Anime", "Filosofia"
    ]

    num_instances = 500
    data = {
        "id_utente": [f"{i}" for i in range(1, num_instances + 1)
            ],
        "macro_argomento": [random.choice(macro_topics) for _ in
            range(num_instances)],
        "argomento_spazio": [random.choice(recommended_topics)
            for _ in range(num_instances)],
        "suggerito": [random.choice([True, False]) for _ in range
            (num_instances)]
    }

    df = pd.DataFrame(data)

    file_path = "social_dataset.csv"
    df.to_csv(file_path, index=False)

    print(f"Dataset creato e salvato in: {file_path}")
```

Listing 1. Ci teniamo a precisare che, come si può notare dal codice, attraverso l'utilizzo di un'algoritmo randomico il comportamento nella fase di generazione del dataset è del tutto stocastico

3.2 Data Analysis

Il dataset è poi stato analizzato e successivamente imputato manualmente per bilanciare quanto più possibile la distribuzione dei dati sintetici attraverso l'utilizzo di DataSpell. La distribuzione pre-imputazione e bilanciamento dei macro-argomenti relativi agli spazi è:

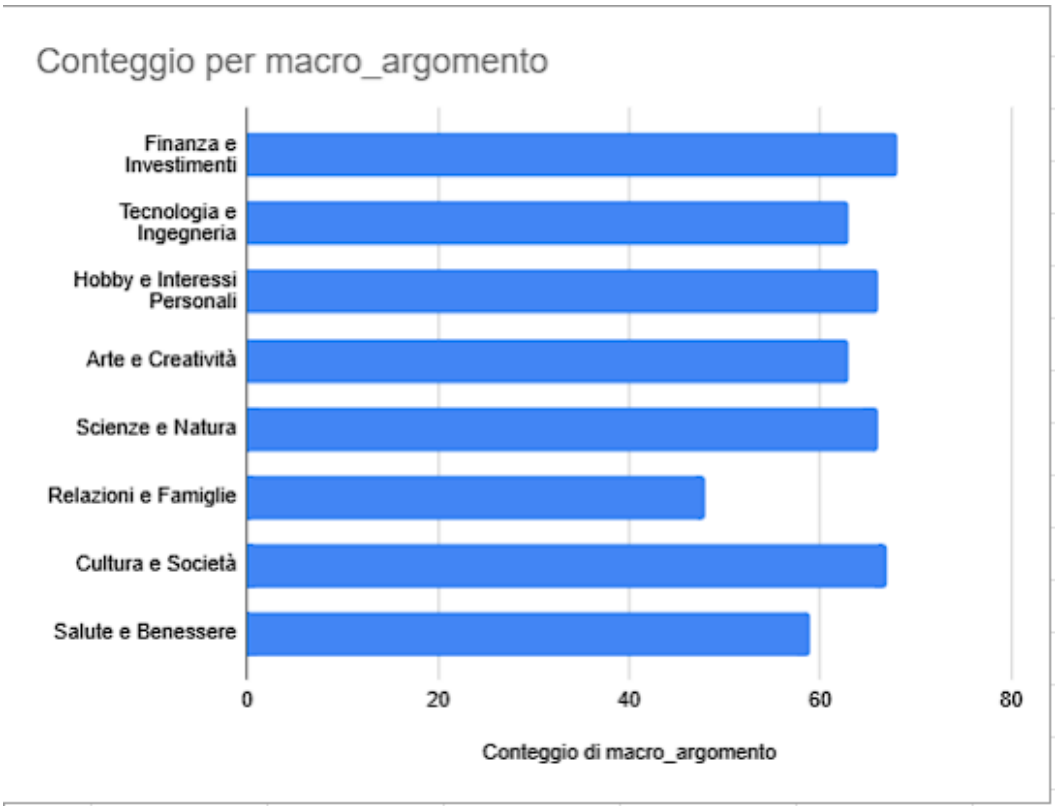

Fig. 1

Mentre la distribuzione degli argomenti degli spazi all'interno del dataset è:

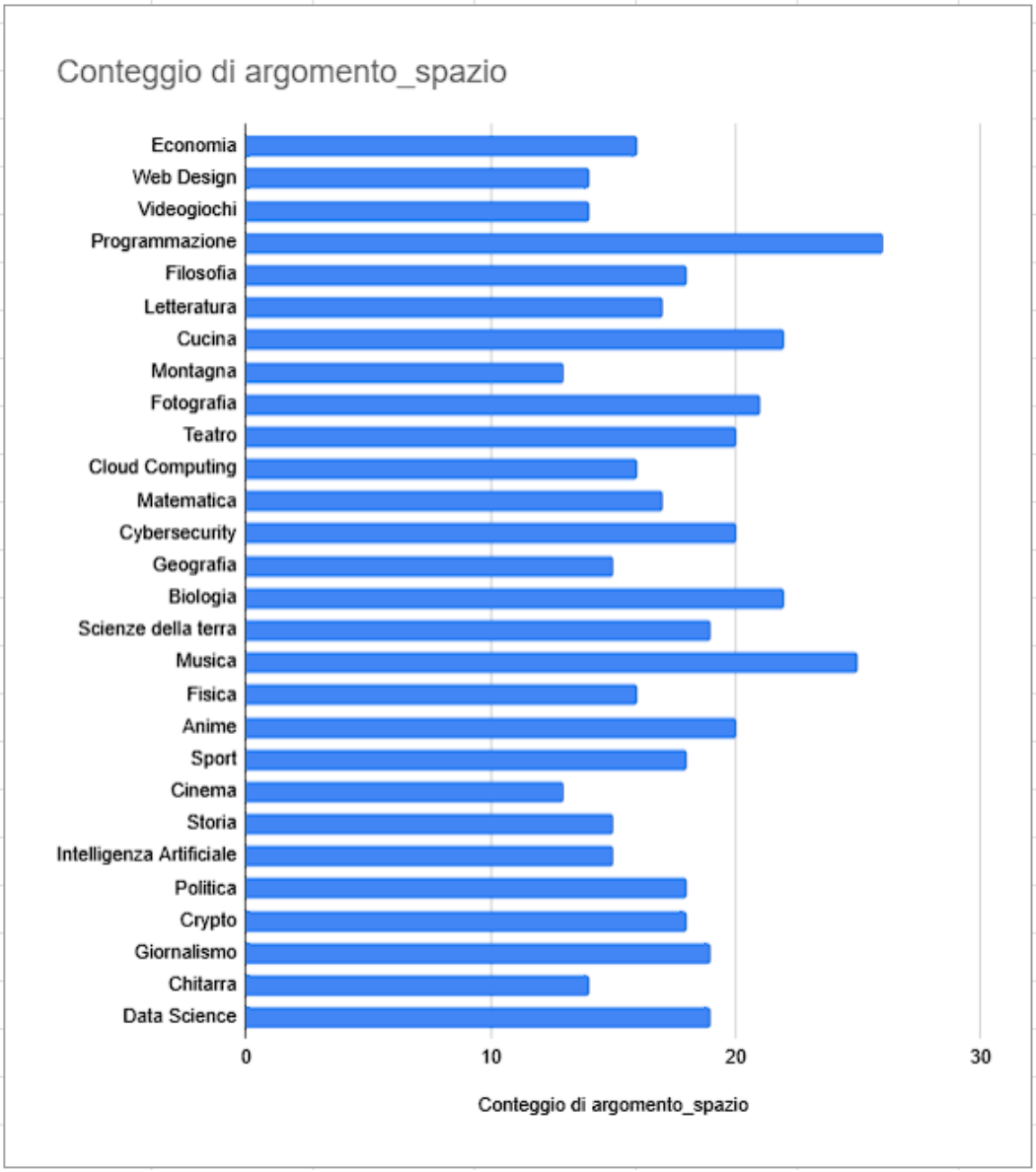

Fig. 2

4 PULIZIA E PREPARAZIONE DEI DATI

In questa fase andiamo prima a effettuare operazioni di pulizia dei dati e poi andremo a identificare e definire su quali **features** il modello verrà addestrato e quale sarà la feature che il modello dovrà predire.

4.1 Data Imputation

In questa fase siamo andati a modificare alcuni dei valori della feature 'suggerito' in modo da far sembrare i suggerimenti il più verosimili possibile, la tecnica di imputazione utilizzata è stata *l'imputazione deduttiva*. Un esempio di riga del dataset prima e dopo l'imputazione:

ID	macro_argomento	argomento_spazio	suggerito
5	Arte e Creatività	Filosofia	FALSE
5	Arte e Creatività	Filosofia	TRUE

Table 1. Esempio di riga prima e dopo l'imputazione

4.2 Data Balancing

In questa fase abbiamo bilanciato ulteriormente il dataset attraverso l'ambiente DataSpell, trasformando le classi più rappresentate in quelle meno rappresentate. I risultati post imputazione e bilanciamento sono:

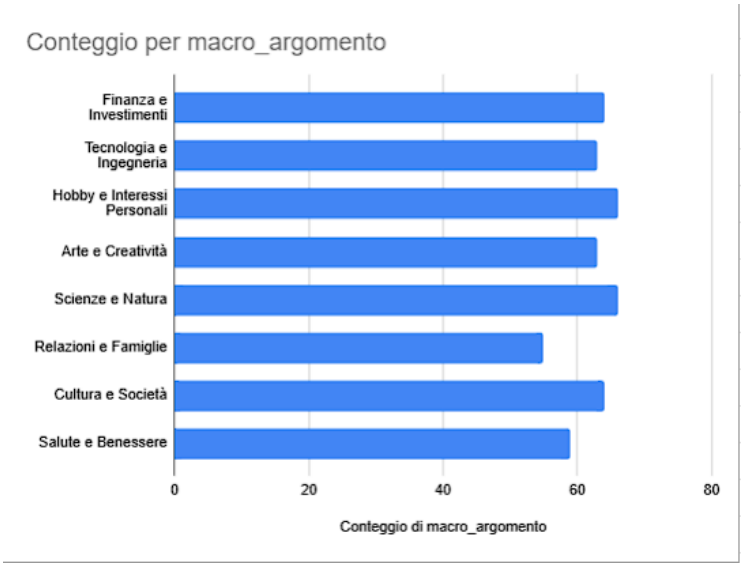

Fig. 3. Distribuzione dei macroargomenti dopo l'imputazione e il bilanciamento del dataset

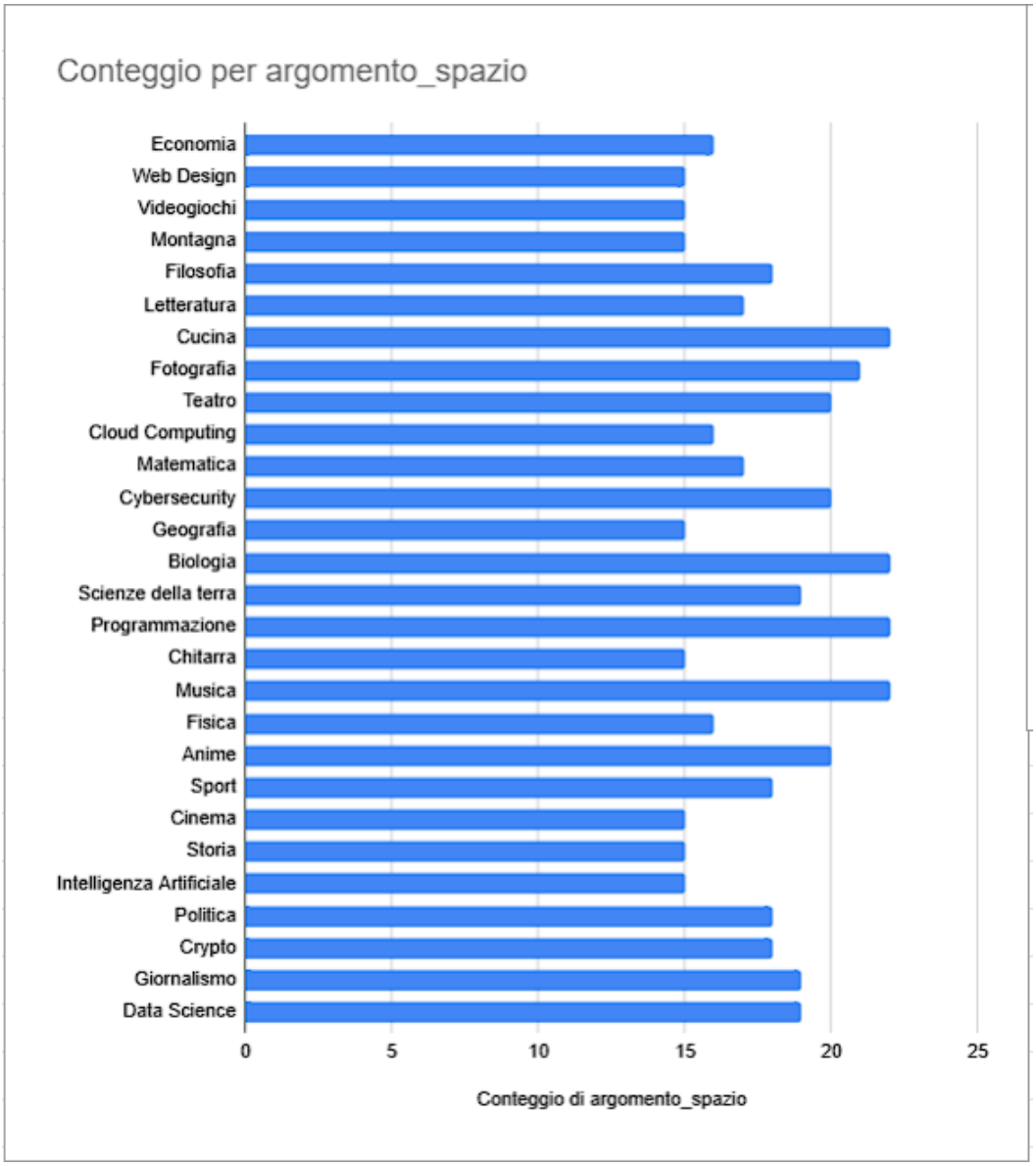

Fig. 4. Distribuzione degli argomenti degli spazi dopo l'imputazione e il bilanciamento del dataset

Notiamo un incremento nel macroargomento relazioni e famiglia, adesso tutti i macroargomenti hanno un numero di spazi da classificare >= 55, notiamo inoltre un bilanciamento nel conteggio di tutti gli argomenti degli spazi che ora hanno un numero di istanze >= 15.

4.3 Feature Engineering

Prima di dare i dati in pasto al modello per iniziare la fase di addestramento, effettuiamo uno step di normalizzazione del dataset, nello specifico delle colonne con etichette testuali utilizzando un Label Encoder, in questo modo tutte le colonne del dataset verranno convertite in valori numerici e saranno meglio interpretabili dal modello.

4.4 Selezione delle Features

La selezione delle features avverrà secondo i seguenti criteri di split calcolati attraverso l'utilizzo della metrica *'cross-entropy'* per l'information gain:

(1) **Argomento spazio**
(2) **Macro argomento**

Questo è l'Information Gain di ciascuna feature utilizzata per la classificazione con il relativo codice python per visualizzarla:

```
Importanza macro_argomento : 0.3455804561587005
Importanza argomento_spazio : 0.6544195438412996
Feature selezionate: ['macro_argomento', 'argomento_spazio']
```

Fig. 5. Information Gain (%) per macro_argomento e argomento_spazio.

```
    #Importanza delle feature selezionate secondo il criterio di
       Information Gain
    importances = model.named_steps['model'].feature_importances_
    for feature, importance in zip(x.columns, importances):
        print(f'{feature}: {importance}')
    selected_features = [feature for feature, importance in zip(x
       .columns, importances)]
    print(f'Feature selezionate: {selected_features}')
```

Listing 2. Codice per la visulizzazione dell'importanza delle feature secondo il criterio di Information Gain

5 MODELLAZIONE DEI DATI

5.1 Scelta del modello e motivazioni

Studiando il nostro caso di utilizzo dove abbiamo la necessità di un modello che suggerisce spazi secondo delle preferenze, parliamo sicuramente di un task di machine learning supervisionato, nello specifico di classificazione binaria. Visto e considerato che ricerche popolari nel campo della **data science** stabiliscono che statisticamente un ensemble di classificatori performa meglio di un singolo classificatore addestrato su un dataset (Fonti: Single classifier vs. ensemble, Are

ensemble classifiers always better than single classifiers?), abbiamo deciso di utilizzare un ensemble, ovvero un raggruppamento, di alberi di decisione. La tipologia di ensemble consisterà nel bagging, ovvero più classificatori verranno addestrati su porzioni diverse del dataset e poi i risultati combinati di tutte le classificazioni verranno a convergere su se: data una determinata preferenza di un utente riguardo a un certo macroargomento e dati gli spazi a cui l'utente può far parte, essi verranno suggeriti. Il nostro modello stabilirà quindi se uno spazio verrà suggerito attraverso la variabile *target*: Suggerito = TRUE oppure Suggerito = FALSE

5.2 Scelta del Classificatore e Addestramento

Nello specifico il classificatore che abbiamo utilizzato è un **Random Forest Classifier** costituito da 40 alberi di decisione (valore scelto empiricamente sulla base delle metriche restituite in varie iterazioni). La suddivisione tra dataset di training e di testing è stata effettuata attraverso il principio Paretiano 80|20 (l'80% del dataset è destinato al training del modello ed il restante 20% al testing del modello)

```
#Selezione delle feature che il modello deve predire e su cui
    deve essere addestrato
x = ds.drop(['id_utente', 'suggerito'], axis='columns')
y = ds.suggerito

#Suddivisione in dati di train e test
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
   test_size=0.2)

model = RandomForestClassifier(n_estimators=40, criterion='
   entropy', max_features=None, random_state=42)

model.fit(xtrain, ytrain)
```

Listing 3. Codice utilizzato per l'addestramento del modello

Il modello di Machine Learning eseguirà quindi in ordine:

(1) Lo step di pre-processing dove si andrà a trasformare le features in colonna in vettori numerici meglio comprensibili al modello;
(2) Lo step di training dove eseguiamo il classificatore con 40 alberi differenti, criterio di scelta delle feature di classificazione basato su cross-entropy per il calcolo dell'Information Gain e un seed randomico per replicare l'esperimento = 42.

5.3 Validazione del Modello

Per la validazione del modello è stato utilizzato lo stratified k-fold cross-validation con il valore di k=10 (il valore più comunemente utilizzato in questo tipo di validazione), questo perché la stratificazione permette di mantere la distribuzione originale delle classi in ciascun fold simile a quella del dataset originale, inoltre abbiamo impostato il miscelamento e un seed randomico per ripetere l'esperimento = 42.

```
# 10-fold cross-validation
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state
    =42)
```

Listing 4. Implementazione del 10-fold cross-validation

5.4 Valutazione del Modello

Dopo la fase di training del modello, attraverso l'utilizzo della libreria *matplotlib* è stata generata la seguente **Confusion Matrix**:

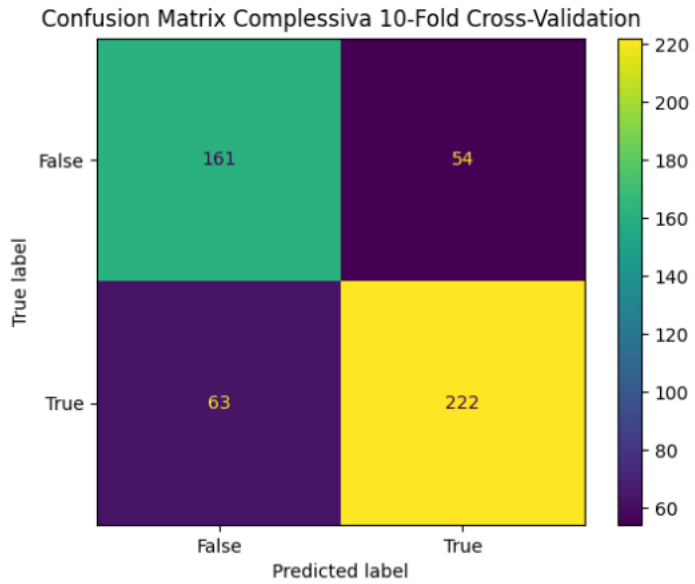

Fig. 6. Confusion Matrix complessiva.

In alto a sinistra osserviamo i veri negativi (TN) ovvero gli spazi da non suggerire che effettivamente non sono stati suggeriti, in alto a destra i falsi positivi (FP) ovvero gli spazi che sono stati suggeriti ma che **non** dovevano esserlo, in basso a sinistra i falsi negativi (FN) ovvero gli spazi che **non** sono stati suggeriti

ma che **dovevano** esserlo, ed infine in basso a destra i veri positivi (TP) ovvero gli spazi che dovevano essere suggeriti e lo sono stati. Attraverso questi valori è stato possibile calcolare le seguenti metriche di valutazione del modello (in ordine di apparizione in console python):
Accuratezza generale del modello nell'effettuare le predizioni

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Rapporto tra quanto il modello ha predetto bene rispetto al numero di falsi positivi

$$Precision = \frac{TP}{TP + FP}$$

Rapporto tra quanto il modello ha predetto bene rispetto al numero di falsi negativi

$$Recall = \frac{TP}{TP + FN}$$

La media armonica tra precision e recall:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Attraverso il seguente codice:

```
import pandas as pd

ytrue = []
ypred = []
accuracy_scores = []
precision_scores = []
recall_scores = []
f1_scores = []

modelCv = model

for train_idx, test_idx in cv.split(x, y):
    # Suddivisione dei dati
    xtrain_fold, xtest_fold = x.iloc[train_idx], x.iloc[test_idx]
    ytrain_fold, ytest_fold = y[train_idx], y[test_idx]

    # Addestramento del modello
    modelCv.fit(xtrain_fold, ytrain_fold)

    # Predizione
    y_pred_fold = modelCv.predict(xtest_fold)

    # Accumula le predizioni e le etichette vere
    ytrue.extend(ytest_fold)
```

```
    ypred.extend(y_pred_fold)

    # Calcolo delle metriche
    acc = accuracy_score(ytest_fold, y_pred_fold)
    prec = precision_score(ytest_fold, y_pred_fold, average='binary')
    rec = recall_score(ytest_fold, y_pred_fold, average='binary')
    f1 = f1_score(ytest_fold, y_pred_fold, average='binary')

    # Salvataggio dei risultati
    accuracy_scores.append(acc)
    precision_scores.append(prec)
    recall_scores.append(rec)
    f1_scores.append(f1)

# Visualizzazione dei risultati per ogni fold
for i in range(len(accuracy_scores)):
    print(f"Fold {i + 1}: Accuracy={accuracy_scores[i]:.4f}, Precision={precision_scores[i]:.4f}, Recall={recall_scores[i]:.4f}, F1 Score={f1_scores[i]:.4f}")

# Calcolo delle metriche medie e relative deviazioni standard
print("\nMetriche medie e deviazioni standard:")
print(f"Accuracy: {sum(accuracy_scores)/len(accuracy_scores):.4f} ± {pd.Series(accuracy_scores).std():.4f}")
print(f"Precision: {sum(precision_scores)/len(precision_scores):.4f} ± {pd.Series(precision_scores).std():.4f}")
print(f"Recall: {sum(recall_scores)/len(recall_scores):.4f} ± {pd.Series(recall_scores).std():.4f}")
print(f"F1 Score: {sum(f1_scores)/len(f1_scores):.4f} ± {pd.Series(f1_scores).std():.4f}")

#Calcolo della confusion matrix complessiva
cm = confusion_matrix(ytrue, ypred)

#Visualizzazione Confusion Matrix Complessiva
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
disp.plot()
plt.title('Confusion Matrix Complessiva 10-Fold Validation')
plt.show()
```

Listing 5. Calcolo delle metriche per la valutazione del modello

Il risultato dell'esecuzione di questo codice è:

```
Fold 1: Accuracy=0.7800, Precision=0.8750, Recall=0.7241, F1 Score=0.7925
Fold 2: Accuracy=0.7000, Precision=0.7333, Recall=0.7586, F1 Score=0.7458
Fold 3: Accuracy=0.7400, Precision=0.7353, Recall=0.8621, F1 Score=0.7937
Fold 4: Accuracy=0.7600, Precision=0.7742, Recall=0.8276, F1 Score=0.8000
Fold 5: Accuracy=0.7400, Precision=0.8077, Recall=0.7241, F1 Score=0.7636
Fold 6: Accuracy=0.7600, Precision=0.7857, Recall=0.7857, F1 Score=0.7857
Fold 7: Accuracy=0.7800, Precision=0.9048, Recall=0.6786, F1 Score=0.7755
Fold 8: Accuracy=0.7400, Precision=0.7778, Recall=0.7500, F1 Score=0.7636
Fold 9: Accuracy=0.8800, Precision=0.8929, Recall=0.8929, F1 Score=0.8929
Fold 10: Accuracy=0.7800, Precision=0.8148, Recall=0.7857, F1 Score=0.8000

Metriche medie e deviazioni standard:
Accuracy: 0.7660 ± 0.0472
Precision: 0.8101 ± 0.0619
Recall: 0.7789 ± 0.0664
F1 Score: 0.7913 ± 0.0399
```

Fig. 7. Risultati delle metriche calcolate in media e per ogni fold

Come possiamo notare la deviazione standard indica di quanto, al più, le metriche del modello possono variare.

6 PUBBLICAZIONE DEL MODELLO

6.1 Performance del modello

Come possiamo notare dalle metriche, dunque, il modello performa discretamente bene ed è quindi pronto per la fase successiva ovvero quella di *deployment* (pubblicazione) del modello sulla piattaforma *AstroVerse*.

6.2 Deployment e usabilità del modello

Il modello è stato esportato tramite la libreria *joblib* di python all'interno della piattaforma *AstroVerse* dove sono stati resi visualizzabili gli spazi suggeriti all'utente. Dunque l'utente durante la compilazione del form di registrazione potrà apporre le sue preferenze sui *macro argomenti* inseriti nel dataset in modo tale che dopo essersi registrato, egli possa subito visualizzare gli spazi più attinenti alle sue preferenze. Inoltre al di sotto della sezione **spazi suggeriti** si troveranno anche i restanti spazi che non sono stati suggeriti, in modo tale da dare all'utente l'opportunità di interfacciarsi anche con nuovi argomenti non per forza attinenti alle sue preferenze.