Progetto di Architetture Dati

Pulerà Francesca mat. 870005

Zarantonello Massimo mat. 866457

Indice

1. Introduzione

L'obiettivo del seguente progetto è costruire un modello di **classificazione di alcuni Punti di Interesse (POI)**, al fine di assegnare a ciascun POI una categoria corretta (ad esempio, museo, ristorante, parco, negozio). Partendo da un dataset estratto da OpenStreetMap, il nostro scopo è analizzare come le **problematiche di qualità dei dati**, come dati mancanti, errori ortografici, e incoerenze, possano influenzare l'efficacia dei modelli di machine learning (ML).

In particolare, vogliamo esplorare il principio del "**garbage in, garbage out**": ovvero, come la qualità dei dati in ingresso influisca direttamente sulle prestazioni del modello. A tal fine, confrontiamo i risultati ottenuti utilizzando versioni del dataset con vari livelli di qualità dei dati (completi, incompleti, errati e inconsistenti), per osservare l'impatto su metriche di performance come **accuracy**, **precision**, **recall**, e **F1-score**.

I passaggi principali di questo progetto comprendono:

1. **Estrazione dei dati** da OpenStreetMap tramite Overpass API.
2. **Pulizia dei dati**: gestione dei valori mancanti, correzione degli errori ortografici e gestione delle incoerenze nei dati.
3. **Creazione di versioni del dataset** per testare l'impatto della qualità dei dati: versioni "pulite", "sporche" (con errori e dati mancanti), e "inconsistenti".
4. **Costruzione e validazione dei modelli** di classificazione sui diversi dataset, comparando le performance.
5. **Conclusioni** e raccomandazioni per il miglioramento continuo della qualità dei dati.

Il nostro approccio mira a mostrare chiaramente come un processo di data cleaning possa impattare sull'apprendimento dei modelli di machine learning e come la qualità dei dati dovrebbe essere un aspetto fondamentale nella costruzione di qualsiasi sistema predittivo.

1.2. Scelta di OpenStreetMap come fonte di dati

La scelta di **OpenStreetMap (OSM)** come fonte principale per l'estrazione dei dati sui Punti di Interesse (POI) è stata motivata da diverse considerazioni pratiche e metodologiche:

1. **Accesso aperto e gratuito ai dati**: OpenStreetMap è un progetto open-source che fornisce dati geospaziali altamente dettagliati e aggiornati, utili per una vasta gamma di applicazioni. Non richiede costosi abbonamenti né licenze commerciali, offrendo una risorsa gratuita e facilmente accessibile.

2. **Diversità di dati**: OSM include una grande varietà di informazioni sui POI, suddivisi in diverse categorie come **amenities**, **tourism**, e **shop**, che sono essenziali per la nostra attività di classificazione. La possibilità di estrarre informazioni dettagliate su luoghi e attività ci ha permesso di progettare un modello in grado di discriminare tra vari tipi di POI in base alle loro caratteristiche geografiche e tematiche.
3. **Comunità e aggiornamenti continui**: la comunità di contributori di OSM è vasta e attiva, il che garantisce aggiornamenti continui dei dati. Questo è un aspetto cruciale per il nostro progetto, in quanto ci permette di lavorare con dati che riflettono costantemente il cambiamento del panorama urbano e del turismo a livello globale.
4. **Flessibilità e disponibilità tramite API**: OpenStreetMap offre interfacce di accesso facili da utilizzare come **Overpass API**, che ci ha consentito di estrarre dati in modo mirato e flessibile, filtrando i POI in base alle nostre necessità (es. POI turistici, ristoranti, ecc.) e velocizzando così la raccolta dei dati necessari per il nostro progetto.

1.3. Applicazioni pratiche nel mondo reale

Le applicazioni di un modello come il nostro sono numerose e variano in diversi settori. Ecco quelle che secondo noi sono alcune possibili applicazioni nel mondo reale:

1. **Turismo e navigazione intelligente**: con una classificazione accurata dei POI, le applicazioni turistiche e di navigazione potrebbero suggerire in modo intelligente luoghi di interesse personalizzati in base alla geolocalizzazione e alle preferenze dell'utente.
2. **Pianificazione urbana nelle smart cities**: il modello potrebbe essere utilizzato dalle amministrazioni locali per pianificare meglio l'infrastruttura urbana, comprendere la distribuzione dei servizi (ospedali, scuole, trasporti) e ottimizzare la mobilità.
3. **Marketing e targeting pubblicitario**: le aziende potrebbero utilizzare questo modello per segmentare il mercato e indirizzare le campagne pubblicitarie in base alla tipologia di POI (ad esempio, pubblicità per i negozi vicino a musei o parchi).
4. **Settore della salute**: una corretta classificazione dei POI potrebbe essere utilizzata per mappare punti di interesse legati alla salute, come ospedali, cliniche, farmacie, utili per migliorare la pianificazione dei servizi sanitari.
5. **Applicazioni sociali e per il sociale**: la categorizzazione dei POI in base alla loro accessibilità può risultare utile per garantire che spazi pubblici, trasporti e strutture siano accessibili a tutte le persone, inclusi anziani e persone con disabilità.

In sostanza, il nostro progetto non si limita a essere un'esercitazione accademica, ma potrebbe avere applicazioni pratiche utili, come supportare la pianificazione urbana, migliorare la gestione delle risorse pubbliche, o ottimizzare strategie di marketing basate sulla geolocalizzazione.

1.4. Qual è il problema che vogliamo risolvere?

Classificazione: l'obiettivo è identificare il tipo di punto di interesse (POI) partendo da un insieme di caratteristiche. Per raggiungere questo scopo, abbiamo utilizzato **Overpass Turbo** per estrarre tutti i POI italiani associati ai seguenti tag di OpenStreetMap:

- **amenity**: classifica i POI in base alla loro funzione, ad esempio *restaurant*, *hospital*, *school*.
- **shop**: utile per distinguere varie tipologie di negozi, come *supermarket* e *bakery*.
- **tourism**: include attrazioni turistiche come *museum*, *hotel* e *viewpoint*.

In questo modo abbiamo ottenuto un dataset di **2677 istanze**.

2. Esplorazione, Preparazione e Pulizia dei dati

Per **migliorare la qualità** del nostro dataset e garantire analisi accurate e significative, è stato essenziale adottare un approccio sistematico alla pulizia dei dati. Questo processo ci ha permesso di eliminare errori, anomalie e incongruenze che avrebbero potuto compromettere la validità dei risultati.

La **pulizia dei dati** non è solo un passaggio preparatorio, ma una fase fondamentale per valorizzare e sfruttare appieno le potenzialità dei nostri dati.

Durante la prima fase di esplorazione, ci siamo concentrati sui seguenti punti principali:

- **Distribuzione delle feature principali**: abbiamo esaminato le categorie principali (amenity, tourism, ecc.) per comprendere meglio la composizione del dataset.
- **Definizione delle etichette**: abbiamo analizzato i dati per stabilire criteri chiari e coerenti per l'etichettatura del dataset.
- **Valutazione della qualità dei dati**: abbiamo identificato eventuali problemi, come dati mancanti, duplicati o incoerenti, che avrebbero potuto compromettere le prestazioni dei modelli.
- **Analisi della struttura del dataset**: abbiamo esaminato le caratteristiche (properties) dei POI per valutarne la rilevanza e l'utilità nel processo di classificazione.

Questa analisi ci ha aiutato ad identificare **possibili criticità** e ad implementare interventi di **data cleaning**, permettendoci di preparare un dataset etichettato che utilizzeremo efficacemente nella pipeline di classificazione.

2.1. Distribuzione delle feature principali

Per determinare il criterio con cui etichettare i POI, abbiamo analizzato la distribuzione delle categorie estratte. In particolare, abbiamo calcolato il numero totale di POI per ciascun tag e il dettaglio delle sotto-categorie.

I risultati mostrano la seguente distribuzione:

- **Amenity: 781 POI** in totale, suddivisi in:
  - *pharmacy*: 212
  - *drinking_water*: 100
  - *restaurant*: 85
  - *social_facility*: 53
  - *fountain*: 38
  - *cafe*: 32

    - *shelter*: 27
    - *fast_food*: 25
    - *ticket_validator*: 22
    - *vending_machine*: 21
    - *... (fino ai meno rappresentati come place_of_mourning, water_point e crematorium con 1 ciascuno).*
- **Shop: 442 POI** in totale, suddivisi in:
    - *deli*: 44
    - *farm*: 36
    - *convenience*: 33
    - *pastry*: 19
    - *wine*: 17
    - *butcher*: 16
    - *bakery*: 14
    - *greengrocer*: 13
    - *stationery*: 13
    - *... (fino a categorie minori come orthopedics, tea e tool_hire con 1 ciascuno).*
- **Tourism: 1543 POI** in totale, suddivisi in:
    - *information*: 1326
    - *attraction*: 72
    - *artwork*: 40
    - *guest_house*: 35
    - *viewpoint*: 23
    - *museum*: 14
    - *apartment*: 13
    - *... (fino a categorie meno frequenti come chalet, picnic_site, hostel e camp_pitch con 1 ciascuno).*

Alcune considerazioni emerse dall'esplorazione iniziale:

- Per quanto riguarda i dati di **amenity**, si evidenzia una netta prevalenza delle categorie *pharmacy*, *drinking_water* e *restaurant*, che costituiscono una parte consistente di questa classe.
- Analizzando i dati della categoria **shop**, si osserva una significativa concentrazione di POI legati al settore alimentare e gastronomico, come *deli*, *farm*, *convenience*, *pastry*, *wine*, *butcher*, *bakery* e *greengrocer*. Queste sotto-categorie sono state quindi successivamente raggruppate in una macro-categoria denominata **"food_retail"** per uniformare il dataset e semplificare il processo di classificazione (riportiamo al paragrafo subito successivo per maggiori informazioni).
- I dati relativi a **tourism** mostrano un'evidente predominanza della categoria *information*, con 1326 occorrenze su un totale di 1543 POI. Tuttavia, la categoria *attraction* emerge come rilevante nel contesto turistico, con 72 occorrenze che ne sottolineano l'importanza all'interno del dataset.

2.2. Definizione delle etichette

In questa fase della pipeline, l'obiettivo è creare un dataset etichettato basato sulle label di OSM, come *amenity*, *tourism*, *leisure* e *shop*, fornendo una base strutturata per il training dei modelli di machine learning.

2.2.1. Categorizzazione per *amenity*

Per la categoria *amenity*, sono state mantenute come etichette principali *pharmacy*, *drinking_water* e *restaurant*, che risultano le più frequenti. Tutte le altre categorie, con un numero di occorrenze significativamente inferiore, sono state raggruppate sotto l'etichetta **"other"**. Questa strategia permette di semplificare il modello, migliorandone la generalizzazione e riducendo i rischi legati a squilibri tra classi.

La distribuzione risultante delle etichette per *amenity* è la seguente:

```
amenity_label
other          2280
pharmacy        212
drinking_water  100
restaurant       85
```

2.2.2. Categorizzazione per *shop*

Le categorie legate ai negozi alimentari, come *deli*, *farm*, *bakery*, e simili, sono state unificate sotto una macro-categoria **"food"**, riducendo il numero complessivo di etichette e migliorando la gestibilità del modello. Le rimanenti categorie, meno rappresentate, sono state raggruppate sotto **"other"**, mantenendo un equilibrio tra semplicità e interpretabilità.

La distribuzione risultante per *shop* è:

```
shop_label
other   2462
food     215
```

2.2.3. Categorizzazione per *tourism*

Nel caso della categoria *tourism*, si è osservata una predominanza della classe *information* (1326 occorrenze), seguita da *attraction* (72 occorrenze), che rappresenta una componente significativa. Tutte le altre categorie sono state raggruppate sotto l'etichetta **"other"** per evitare frammentazioni eccessive del dataset e incrementare la robustezza del modello.

La distribuzione risultante è la seguente:

```
tourism_label
information   1326
other         1279
attraction      72
```

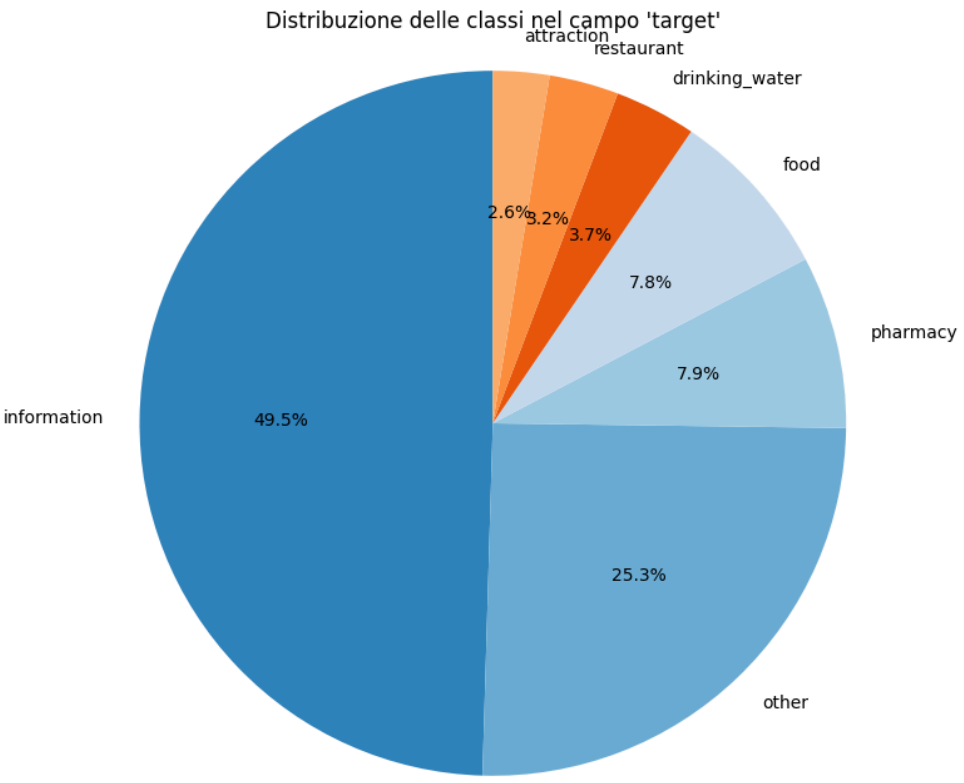
2.2.4. Creazione della colonna finale *target*

Per ottenere una classificazione univoca, le tre colonne relative alle categorie (*amenity_label*, *shop_label*, *tourism_label*) sono state combinate in un'unica colonna denominata **"target"**. Se in tutte e tre le colonne è presente il valore **"other"**, anche *target* assume il valore **"other"**.

Questo approccio garantisce una rappresentazione chiara e compatta per ciascun record.

La distribuzione finale delle etichette nella colonna *target* è:

```
target
information     1326
other            676
pharmacy         212
food             209
drinking_water   100
restaurant        85
attraction        69
```



Questa distribuzione evidenzia un certo sbilanciamento, con un alto numero di occorrenze per alcune categorie principali e un numero minore per altre.

2.3. Valutazione della qualità dei dati

Questa fase ha l'obiettivo di eseguire controlli preliminari sul dataset per identificare rapidamente anomalie, inconsistenze e problemi comuni che potrebbero influenzare negativamente le analisi successive. Durante i **sanity checks**, si esaminano aspetti come:

- **Valori Mancanti**: identificazione delle colonne o righe con una percentuale significativa di dati nulli.
- **Valori Duplicati o Incoerenti**: verifica che mira ad individuare righe ripetute all'interno del dataset, che potrebbero rappresentare informazioni ridondanti, oppure dati che non rispettano il formato atteso o contraddicono le regole del dominio.
- **Cardinalità**: verifica del numero di categorie uniche nelle variabili categoriche, utile per rilevare livelli inaspettati o rumore.

Questi controlli iniziali forniscono una panoramica dello stato del dataset e permettono di identificare priorità per interventi di pulizia.

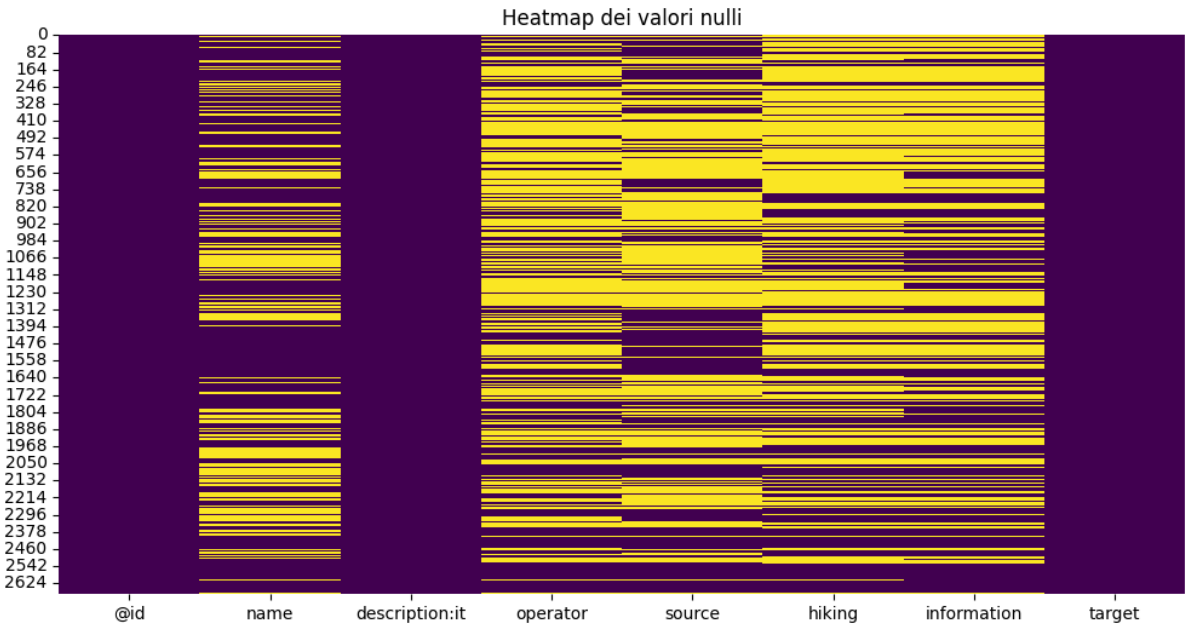
2.3.1. Valori Mancanti

Nell'analisi preliminare del dataset, sono state individuate diverse colonne con valori mancanti. Per assicurare l'integrità dei dati, è stato applicato un filtro per mantenere esclusivamente le colonne con meno del 60% di valori mancanti, preservando così una quantità sufficiente di informazioni per il processo di classificazione.

Le colonne selezionate sono:

- **@id**: non presenta valori mancanti (0.0%).
- **description:it**: non presenta valori mancanti (0.0%).
- **name**: il 33.99% dei valori è mancante, ma rimane un contributo significativo ai fini della classificazione.
- **operator**: il 52.86% dei valori è mancante, ma è stato incluso in quanto rappresenta una potenziale informazione utile.
- **source**: il 52.11% dei valori è mancante, ma rientra tra le colonne considerate rilevanti.
- **hiking**: registra il 58.95% di valori mancanti, risultando appena entro la soglia stabilita.
- **information**: presenta il 50.47% di valori mancanti, rientrando comunque nei criteri di selezione.
- **target**: non presenta valori mancanti (0.0%).

Questa strategia di filtraggio ha permesso di ridurre la complessità del dataset, limitando l'impatto dei valori mancanti senza sacrificare informazioni critiche. Eventuali ulteriori azioni, come l'imputazione dei valori mancanti o l'esclusione di righe specifiche, saranno valutate nel contesto del modello di machine learning adottato.



La heatmap presentata mostra la distribuzione dei valori mancanti (nulli) all'interno del dataset. Ogni riga rappresenta un'osservazione (o elemento) del dataset, mentre ogni colonna rappresenta una variabile (o feature). I colori della heatmap indicano la presenza o l'assenza di dati:

- Viola: valori presenti.
- Giallo: valori mancanti (nulli).

La maggior parte delle righe presenta valori mancanti almeno in alcune colonne.

Alcune colonne hanno una densità particolarmente elevata di valori nulli, suggerendo la possibilità che queste variabili contengano informazioni scarsamente rilevanti o incomplete per l'analisi.

L'analisi dei valori nulli è un passo fondamentale per garantire la qualità del dataset e, di conseguenza, la robustezza dei risultati ottenuti durante il progetto. Interventi mirati per trattare i valori mancanti contribuiranno a migliorare sia la precisione del modello che la validità delle conclusioni.

2.3.2. Valori Duplicati o Incoerenti

Nell'ambito dei dati estratti da OpenStreetMap (OSM), ogni punto di interesse (POI) viene identificato da un ID univoco che è automaticamente assegnato dal sistema, garantendo che ogni istanza abbia un riferimento distinto. Questo sistema di identificazione univoca riduce significativamente il rischio di duplicati effettivi a livello di dati grezzi.

Nonostante questo, possono esistere situazioni in cui due POI condividano lo stesso nome e la stessa descrizione, ma rappresentino entità distinte (ad esempio, lo stesso nome per farmacie appartenenti a una catena o descrizioni simili per punti di interesse in località diverse).

Per verificare se i POI apparentemente duplicati (con nome e descrizione identici) rappresentano effettivamente delle duplicazioni errate, si potrebbe analizzare ulteriormente le informazioni disponibili, ad esempio verificando la loro posizione geografica tramite la latitudine e la longitudine.

Nel nostro dataset, si osserva che su un totale di 2677 istanze, solo 382 POI sono risultati avere lo stesso nome e la stessa descrizione. Questo valore rappresenta una percentuale trascurabile (~14%) del dataset complessivo. Considerando che i duplicati effettivi potrebbero essere minimi e che l'impatto sulla qualità del dataset complessivo è marginale, abbiamo deciso di mantenere tutte le istanze per semplicità.

Questa scelta consente di preservare la completezza dei dati senza introdurre potenziali errori legati all'eliminazione non corretta di entità valide.

Per individuare e gestire eventuali **incoerenze** all'interno del dataset, abbiamo avviato un processo di pulizia e standardizzazione, focalizzandoci inizialmente sulla

colonna **"description:it"**. Questa colonna contiene descrizioni testuali che potrebbero presentare discrepanze dovute a typo o formattazioni non uniformi.

Pre-elaborazione del testo

La pipeline di pulizia applicata alla colonna "description:it" comprende i seguenti passaggi:

1. **Rimozione di spazi extra** con `.str.strip()`
2. **Conversione a minuscolo** per uniformare il testo con `.str.lower()`
3. **Normalizzazione Unicode** per eliminare caratteri ambigui tramite la funzione `normalize_text`
4. **Uniformazione degli spazi** con una regex (`\s+`) per rimuovere spazi multipli.

Identificazione delle incoerenze

Abbiamo utilizzato un approccio basato sul calcolo della somiglianza tra le descrizioni tramite il **fuzzy string matching** (libreria `fuzz`). Abbiamo confrontato ogni coppia di descrizioni nel dataset e individuato quelle con una somiglianza superiore alla soglia del 90% ma inferiore al 100%. Questo ci ha permesso di evidenziare potenziali discrepanze, come:

- **Typo minori**: ad esempio, "orarfi attuale e turni sul sito web" vs. "orari attuale e turno sul sito web".
- **Differenze marginali nella punteggiatura**: come "orari attuale e turno sul sito web" vs. "orari attuale e turno sul sito web."
- **Distinzione semantica poco rilevante**: ad esempio, "percorsi mtb alta val cedra n.2" vs. "percorsi mtb alta val cedra n.4".

Le descrizioni identificate saranno analizzate per determinare se è possibile raggrupparle o rettificarle per migliorare la qualità del dataset.

Gestione delle altre colonne

- "**source**": sono stati rilevati typo che possono essere corretti per trasformare la colonna in una variabile categoriale uniforme, migliorando la classificazione.
- **"information"**: questa colonna è stata esclusa da questo processo, poiché la sua struttura non richiede interventi di pulizia o normalizzazione.

2.4. Analisi della Struttura del Dataset

2.4.1. Rappresentazione delle feature

In questa sezione ci siamo concentrati sulla trasformazione delle feature, attualmente tutte di tipo "object", in valori numerici adatti all'utilizzo con un classificatore. Di seguito, il tipo di dati originario per ciascuna colonna:

```
@id              object
name             object
description:it   object
operator         object
source           object
hiking           object
information      object
target           object
```

Inizialmente, abbiamo trasformato `target` in una variabile categorica, mentre la colonna `@id` è stata rimossa poiché non rilevante per l'analisi. Successivamente, sono state intraprese diverse operazioni per costruire tre dataset distinti, ognuno con scopi specifici.

```
Mapping delle categorie:
{'attraction': 0, 'drinking_water': 1, 'food': 2, 'information': 3, 'other': 4, 'pharmacy': 5, 'restaurant': 6}
```

Dataset df_1

Per il dataset *df_1*, abbiamo incluso tutte le feature selezionate, mantenendo però i dati grezzi (privi di pulizia approfondita). In questo caso, l'obiettivo è costruire una baseline per le prestazioni dei modelli. Le operazioni principali eseguite sono le seguenti:

1. **Preprocessing delle feature testuali**:
    - Conversione del testo in minuscolo (*lowercase*).
    - Tokenizzazione.
2. **Trattamento delle feature categoriche**:

    La colonna `source` è stata mantenuta come feature testuale, poiché una codifica categoriale avrebbe generato un numero elevato di categorie non significative.

    La colonna `hiking` è stata trattata come categorica, avendo solo tre valori distinti: `yes`, `no` e *null*. I valori mancanti (*null*) sono stati considerati come una categoria a sé stante, unificandoli con le tre istanze erroneamente popolate come "*yes'information'='guidepost*". La distribuzione risultante è:

    ```
    hiking
    null   1581
    yes     909
    ```

```
no     187
```

3. **Trattamento di "information"**:

- I valori mancanti sono stati assegnati alla categoria "missing".
- Le classi con frequenza minore di 10 sono state consolidate nella categoria "other". La distribuzione finale è:

```
information
missing       1351
guidepost     1103
board          120
route_marker    55
map             39
other            9
```

Sebbene *df_1* contenga dati preparati per un classificatore, alcune limitazioni permangono:

- Le feature testuali sono ancora sotto forma di testo tokenizzato e non vettorizzate, rendendo il dataset inadatto a modelli che richiedono input numerici.
- I classificatori, come Decision Tree e SVM, necessitano di variabili numeriche normalizzate o codificate.

Per la colonna `target`, è stato utilizzato il **Label Encoder** di scikit-learn, che mappa ogni categoria a un valore numerico. La vettorizzazione delle feature testuali è stata rimandata alla sezione dedicata alla "*4. Rappresentazione Vettoriale delle Feature Testuali*".

Con queste operazioni, *df_1* rappresenta un primo dataset etichettato e codificato, ma ancora non completamente pulito.

Dataset df_2

Il secondo dataset, *df_2*, include esclusivamente le caratteristiche testuali `name` e `description:it`. Esso viene utilizzato come termine di paragone per dimostrare che l'inclusione delle altre feature nel dataset contribuisce a migliorare le prestazioni dei modelli di machine learning.

In questa fase, abbiamo:

- Eliminato incoerenze da `description:it`, basandosi sull'analisi preliminare delle somiglianze.
- Applicato una mappa di sostituzione per unificare descrizioni simili, ad esempio:
    - "orarfi" → "orari"
    - "turno" → "turni"

    - "turnosul" → "turni sul"
- Effettuato il seguente preprocessing sulle feature testuali:
    - Rimozione di punteggiatura
    - Rimozione di stopwords
    - Rimozione di numeri
    - Lemmatizzazione

Nonostante il dataset *df_2* contenga solo due feature pulite, dimostreremo che l'inclusione delle altre feature, sebbene non perfettamente pulite, contribuisce a migliorare le prestazioni dei modelli.

Dataset df_3

Il dataset *df_3* rappresenta la versione finale, completa, pulita e priva di inconsistenze.

Operazioni effettuate:

- Riutilizzo delle feature `name` e `description:it` preprocessate in *df_2*.
- Preprocessing di `operator`.
- Inclusione delle altre feature categoriche preprocessate da *df_1*.
- Pulizia e categorizzazione di `source`:
    - Uniformazione delle categorie simili con una mappa di sostituzione.
    - Sostituzione delle categorie con frequenza inferiore a 30 con "other".
    - Trasformazione in variabile categorica.

Esempi di mappatura per `source`:

- "survey : cai pallanza" → "survey : cai"
- "ministerio" → "ministero"
- "mise ministero dello sviluppo economico" → "ministero dello sviluppo economico"
- Vettorizzazione della feature `source` tramite **OneHotEncoder**:
    - Conversione della feature categorica in un array numerico.
    - Aggiunta delle nuove feature al dataset.
    - Eliminazione della colonna originale `source`.

Anche in questo caso, la vettorizzazione del testo verrà trattata nella sezione *"4.Rappresentazione Vettoriale delle Feature Testuali"*.

2.4.2. Valutazione dell'Utilità delle Feature

Per verificare l'utilità delle feature selezionate per la classificazione, abbiamo applicato tecniche statistiche e di machine learning. Tra queste, il **test del**

Chi-Quadrato si è rivelato utile per misurare la dipendenza tra le variabili categoriche e il target.

Analisi della feature hiking

Nella tabella di contingenza seguente, vengono riportate le distribuzioni delle categorie di *hiking* rispetto al target.

target hiking	attraction	drinking_water	food	information	other	pharmacy	restaurant
no	0	0	0	187	0	0	0
null	69	100	209	230	676	212	85
yes	0	0	0	909	0	0	0

Il test del Chi-Quadrato per la feature *hiking* fornisce un valore **p-value di 0.0**, suggerendo che la relazione tra *hiking* e il target è altamente significativa. Ciò implica che mantenere la feature *hiking* è giustificato per la classificazione.

Analisi della feature information

Un'analisi simile è stata condotta sulla feature *information*. La seguente tabella di contingenza mostra la distribuzione delle sue categorie rispetto al target.

target information	attraction	drinking_water	food	information	other	pharmacy	restaurant
board	0	0	0	120	0	0	0
guidepost	0	0	0	1103	0	0	0
map	0	0	0	39	0	0	0
missing	69	100	209	0	676	212	85
other	0	0	0	9	0	0	0
route_marker	0	0	0	55	0	0	0

Anche in questo caso, il test Chi-Quadrato ha fornito un **p-value di 0.0**, indicando una relazione significativa tra la feature *information* e il target. Di conseguenza, anche questa variabile è stata considerata rilevante per la classificazione.

Entrambe le feature verranno mantenute nel modello per contribuire alla sua efficacia.

3. Dataset Generati

Riassumendo, abbiamo creato **3 diversi dataset**, ognuno con caratteristiche specifiche, per confrontare le loro prestazioni. Di seguito una descrizione dei dataset:

- **df_1**:
  Include tutte le feature selezionate, con dati etichettati e codificati, ma **non ancora puliti**. La feature *source* è ancora in formato testuale. Questo dataset viene utilizzato come baseline.
- **df_2**:
  Contiene esclusivamente le feature *name* e *description:it*. È utilizzato per evidenziare l'impatto delle altre feature sul miglioramento delle prestazioni dei modelli di machine learning.
- **df_3**:
  È il dataset finale, **completo, pulito e privo di incoerenze**. La feature *source* è stata trasformata in una categoria e le feature testuali sono state preprocessate.

Questi dataset rappresentano i punti di riferimento per analizzare come il trattamento dei dati e la scelta delle feature influenzano l'efficacia dei modelli.

4. Rappresentazione Vettoriale delle Feature Testuali

Nel processo di trasformazione dei dati di testo in rappresentazioni numeriche, come i conteggi delle parole o i pesi **TF-IDF (Term Frequency-Inverse Document Frequency)**, vengono applicati metodi standard per assicurare la coerenza tra i dati di addestramento e quelli di test.

Per rappresentare i documenti testuali come matrici numeriche, abbiamo scelto di utilizzare **TF-IDF**, una tecnica che pesa l'importanza di una parola considerando:

- **TF (Term Frequency)**: frequenza della parola in un documento specifico.
- **IDF (Inverse Document Frequency)**: frequenza inversa della parola nel corpus complessivo, riducendo il peso dei termini troppo comuni.

Il TF-IDF consente di evidenziare termini rilevanti per un documento specifico riducendo l'impatto delle parole generiche, migliorando così le prestazioni dei modelli di machine learning.

5. Classificazione

5.1. Support Vector Machines (SVM)

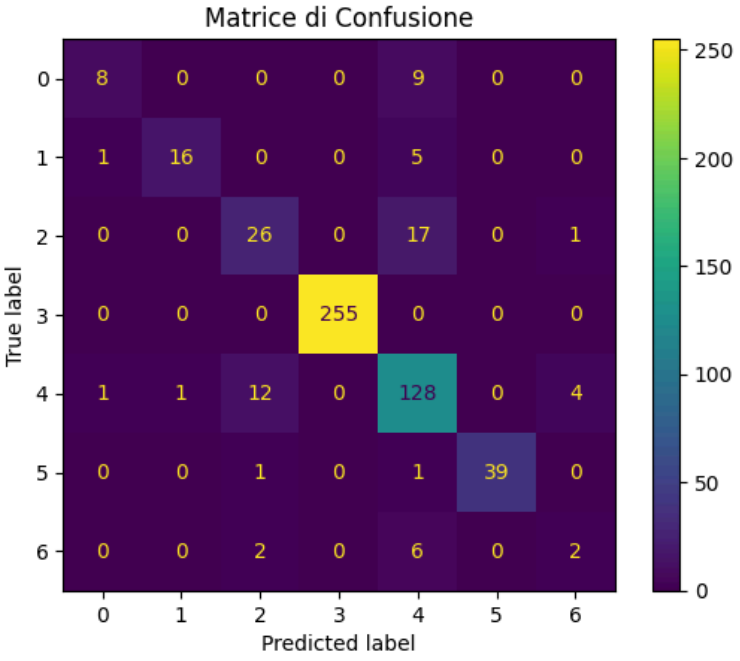
Le **SVM** (Support Vector Machines) sono un tipo di algoritmo di apprendimento supervisionato utilizzato sia per problemi di classificazione che di regressione. Due caratteristiche principali delle SVM sono:

1. **Margine massimale**: le SVM cercano di trovare l'iperpiano ottimale che massimizza il margine tra le classi nel dataset di addestramento. L'iperpiano ottimale è quello che separa le classi in modo tale da massimizzare la distanza tra l'iperpiano e i punti più vicini (support vector) appartenenti a ciascuna classe.
2. **Kernel trick**: le SVM possono gestire efficacemente dati non linearmente separabili tramite il "kernel trick". Questo consente di mappare i dati in uno spazio dimensionale superiore in cui diventano linearmente separabili. Ciò consente alle SVM di affrontare problemi complessi e di adattarsi a una vasta gamma di dataset.

In sintesi, le SVM sono potenti strumenti di classificazione che si distinguono per la loro capacità di trovare iperpiani di decisione ottimali che massimizzano il margine tra le classi e per la loro flessibilità nell'affrontare problemi complessi tramite il kernel trick.

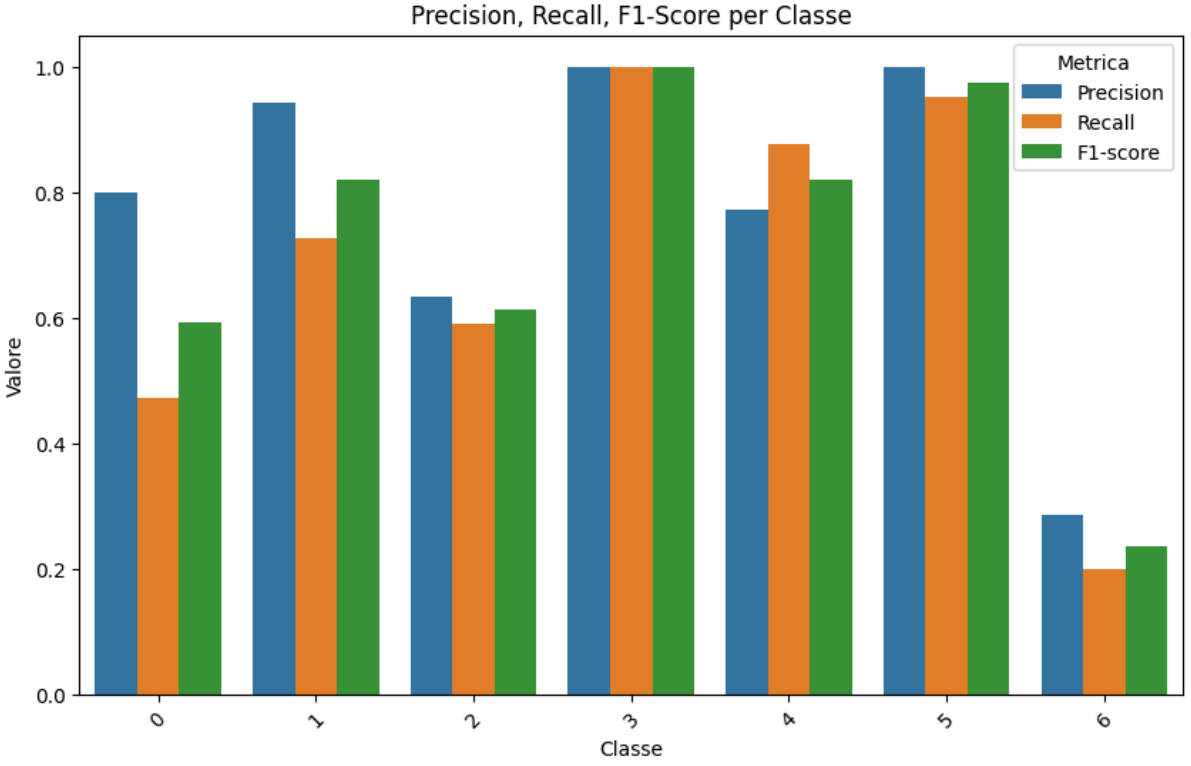
5.1.1. df_1

- La matrice di confusione mostra come le predizioni del modello si distribuiscono tra le classi. Si nota che alcune classi meno rappresentate (es., classe 6) vengono spesso confuse con altre classi, mentre le classi più popolose (es., classe 3) ottengono prestazioni migliori.



Metriche Aggregate:
Precisione macro: 0.764
Recall macro: 0.717
F1-score macro: 0.731
Precisione weighted: 0.880
Recall weighted: 0.886
F1-score weighted: 0.880

- Le metriche macro evidenziano come il modello si comporta mediamente su tutte le classi, penalizzando maggiormente le classi con meno dati. Nel nostro caso, l'F1-score macro è inferiore (0.73) rispetto all'F1-score weighted (0.88), mostrando che il modello è meno efficace sulle classi meno rappresentate.
- Le metriche per la classe 6 mostrano un recall molto basso (20%), suggerendo difficoltà a identificare correttamente istanze di questa classe. Questo potrebbe essere affrontato migliorando il bilanciamento del dataset.

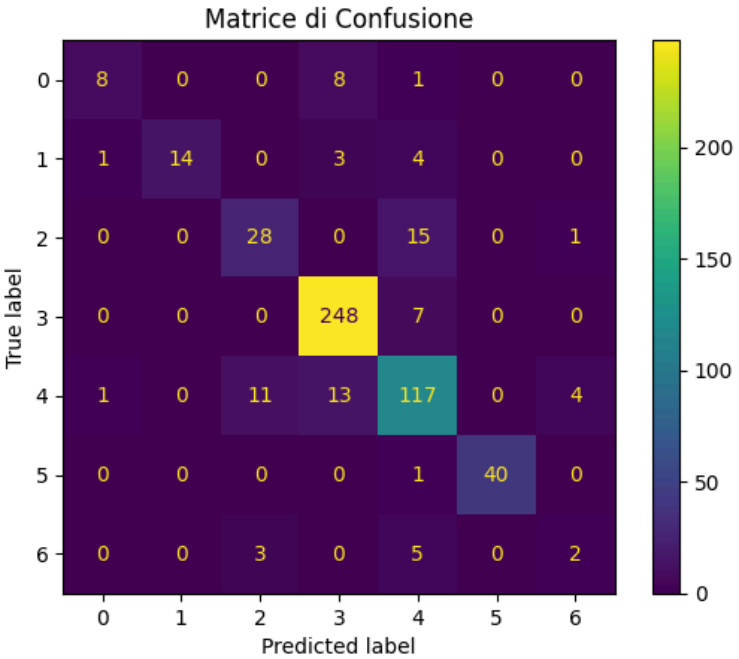


- **Accuratezza sul set di test: 0.886**:

5.1.2. df_2

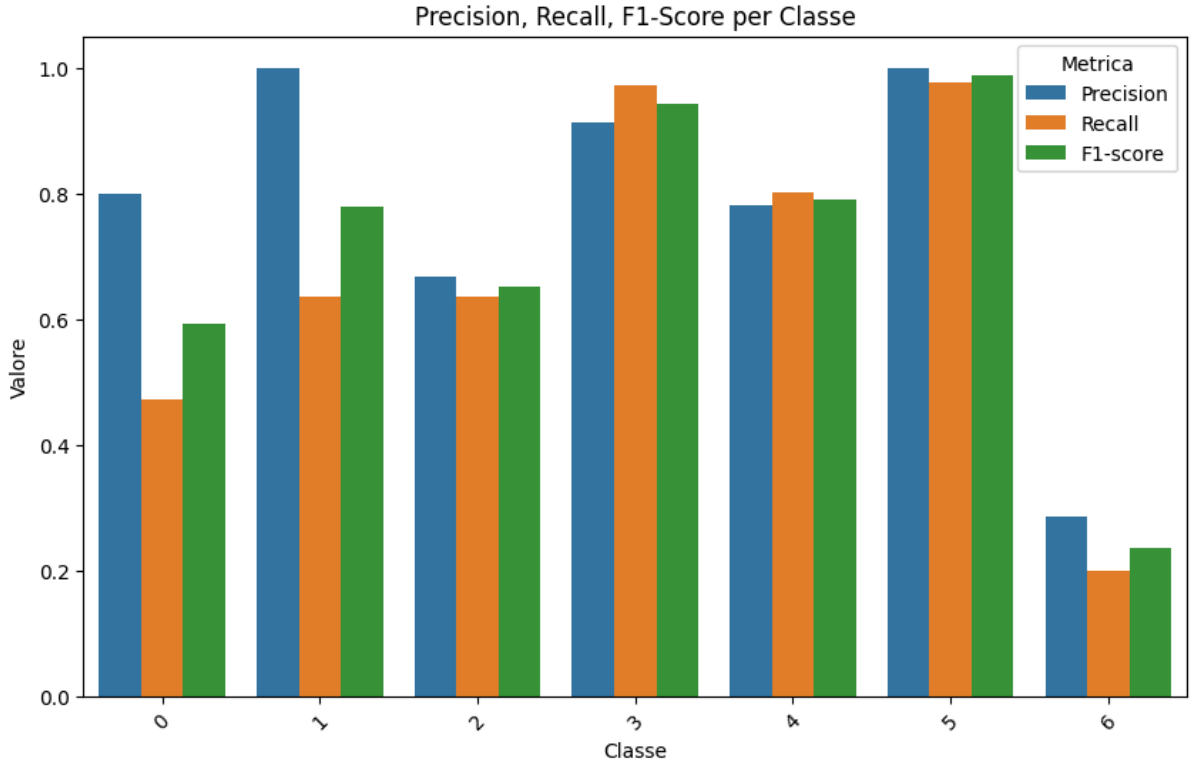
- La matrice di confusione è molto simile a quella di df_1

Metriche Aggregate:
Precisione macro: 0.758
Recall macro: 0.701
F1-score macro: 0.715
Precisione weighted: 0.848
Recall weighted: 0.858
F1-score weighted: 0.849



- Classe 6 ha una recall molto bassa (0.20), indicando problemi nel riconoscere questa classe.

- Le classi 5 e 3 hanno prestazioni eccellenti (alta precisione e recall), mentre classe 0 ha un recall basso (0.47), evidenziando false negatività.
- Le metriche macro indicano buone prestazioni generali, ma penalizzano le classi meno rappresentate.
- Le metriche weighted evidenziano prestazioni solide nelle classi più grandi.

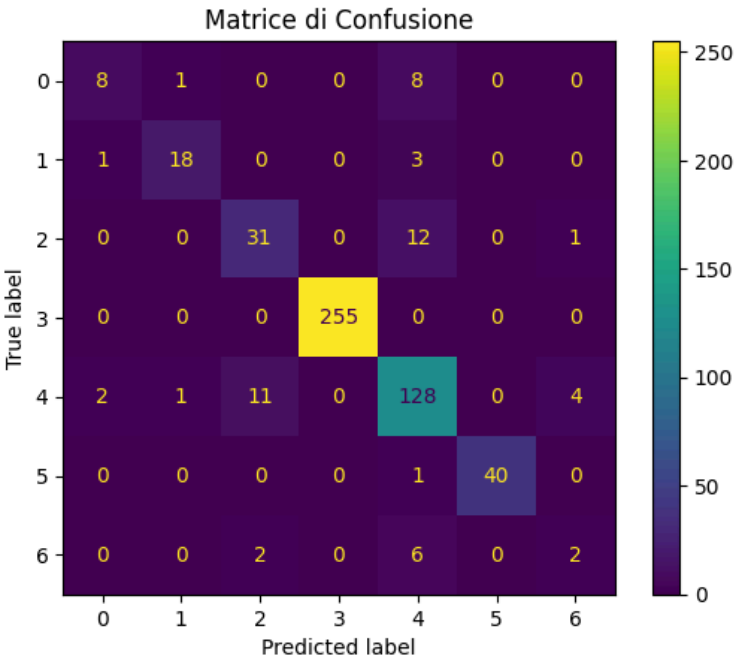


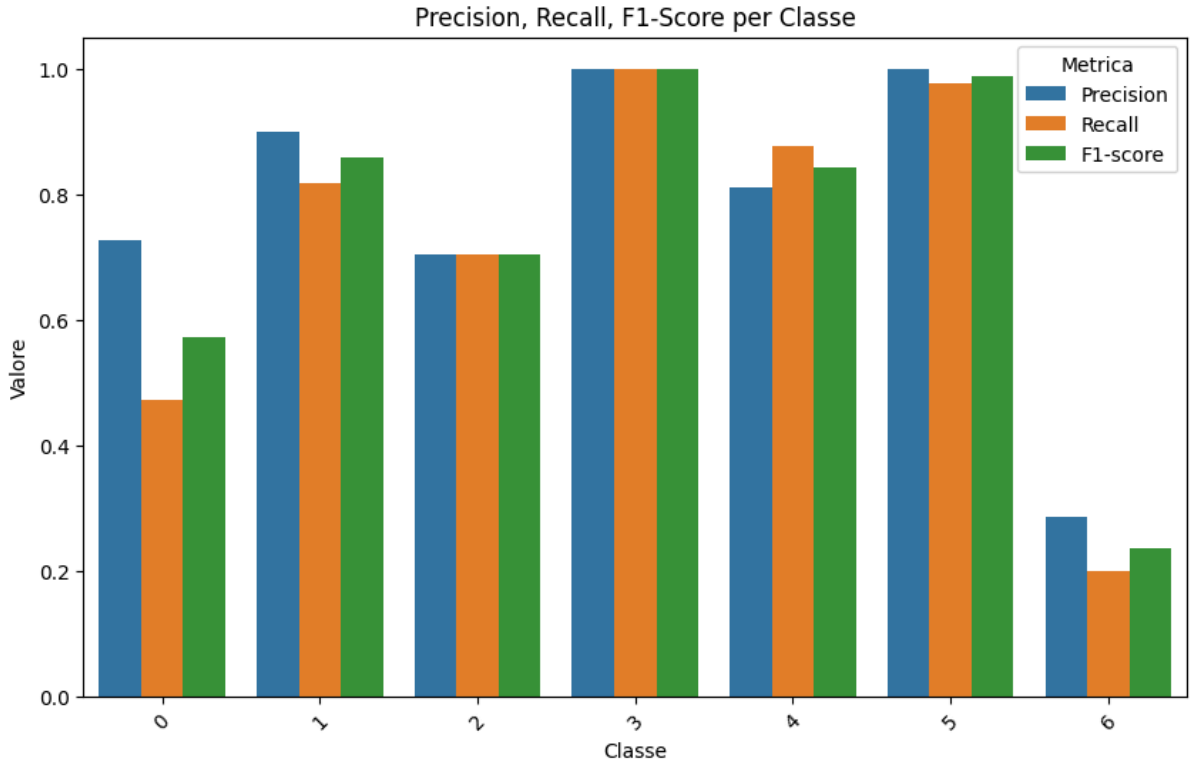
- **Accuratezza sul set di test: 0.858**

5.1.3. df_3

- La matrice di confusione è molto simile a quella di df_1 e a quella di fd_2

Metriche Aggregate:
Precisione macro: 0.753
Recall macro: 0.745
F1-score macro: 0.741
Precisione weighted: 0.879
Recall weighted: 0.888
F1-score weighted: 0.882





- **Accuratezza sul set di test: 0.888**

5.1.4. Considerazioni e Analisi

- In generale, le metriche sul test set sono sempre simili a quelle del validation set -> il modello ha generalizzato bene.
- **df_1** (dataset sporco): **0.886** di accuratezza
- **df_2** (dataset pulito, solo feature testuali): **0.858** di accuratezza
- **df_3** (dataset pulito, tutte le feature): **0.888** di accuratezza

ANALISI:

- **df_1 (baseline):**
  Il dataset di baseline (prima della pulizia) è già molto buono, con un livello di accuratezza elevato. Le performance sono stabili grazie alla varietà di feature presenti. Il modello dà buoni risultati nelle classi maggiori, ma le metriche macro indicano un lieve squilibrio nelle prestazioni tra le classi. Questo suggerisce che i dati non puliti mantengono una ricchezza di informazioni che favorisce il modello, soprattutto per le classi più rappresentate.
- **df_2 (dati parzialmente puliti):**
  L'accuratezza inferiore su questo dataset indica un calo delle prestazioni dovuto alla rimozione di molte feature durante il processo di pulizia. Le metriche macro e ponderate sono peggiorate, suggerendo una perdita di informazioni importanti per la generalizzazione. Il modello sembra risentire

della ridotta varietà di caratteristiche, dimostrando la necessità di preservare le informazioni più rilevanti durante il preprocessing.

- **df_3 (dati completamente puliti):**
  Il miglioramento dell'accuratezza rispetto a df_2 positivo dimostra che la pulizia e il preprocessing possono avere un impatto quando eseguiti correttamente. Le metriche weighted evidenziano prestazioni superiori per le classi più piccole, suggerendo una maggiore generalizzazione del modello. Questo dataset rappresenta una versione più robusta e bilanciata, capace di fornire risultati migliori nelle classi meno rappresentate senza sacrificare la precisione nelle classi maggiori.

5.2. Decision Tree Classifier (DT)

Gli **alberi decisionali** sono modelli di apprendimento automatico utilizzati sia per problemi di classificazione che di regressione. Operano dividendo ripetutamente il dataset in sottoinsiemi omogenei in base alle caratteristiche dei dati, fino a quando non viene raggiunta una condizione di stop predefinita. Due caratteristiche principali degli alberi decisionali sono:

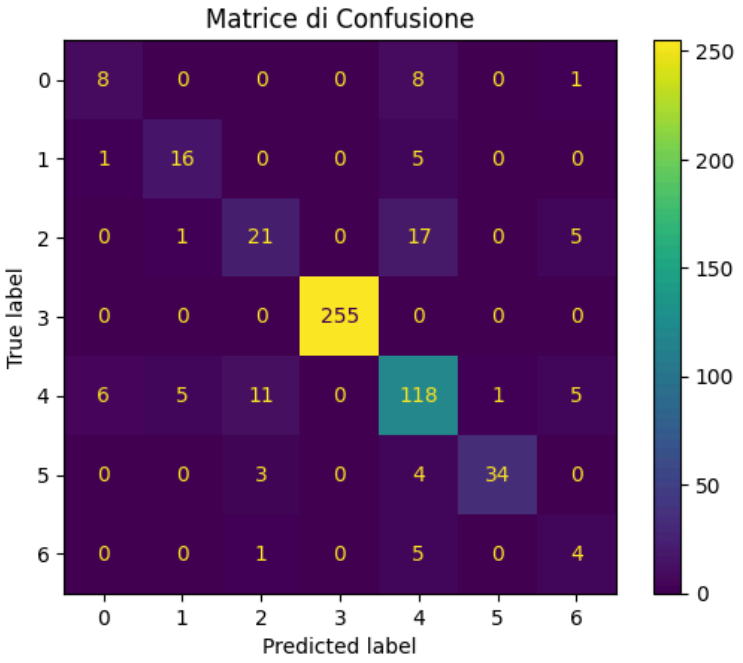
1. **Interpretabilità**: Gli alberi decisionali sono modelli facilmente interpretabili, poiché le regole di decisione sono rappresentate sotto forma di albero. Questo permette di comprendere facilmente il processo decisionale seguito dal modello.
2. **Versatilità**: Gli alberi decisionali possono gestire sia dati numerici che categorici e possono essere utilizzati per risolvere una vasta gamma di problemi di machine learning. Possono anche essere combinati con tecniche come il pruning per evitare l'overfitting e migliorare le prestazioni del modello.

Gli alberi di decisione possono gestire variabili categoriche, ma è preferibile convertire le variabili categoriche in variabili numeriche o booleane per garantire una migliore compatibilità con l'algoritmo e per ottenere risultati più significativi.

5.2.1. df_1

Metriche Aggregate:
Precisione macro: 0.700
Recall macro: 0.723
F1-score macro: 0.709
Precisione weighted: 0.866
Recall weighted: 0.869
F1-score weighted: 0.867

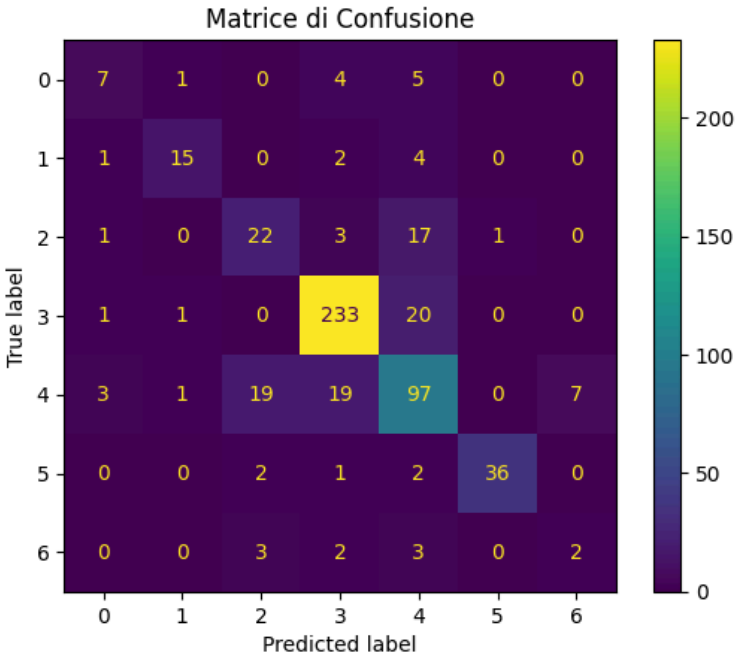
Accuratezza sul set di test: 0.869



5.2.2. df_2

Metriche Aggregate:
Precisione macro: 0.683
Recall macro: 0.627
F1-score macro: 0.641
Precisione weighted: 0.780
Recall weighted: 0.780
F1-score weighted: 0.777

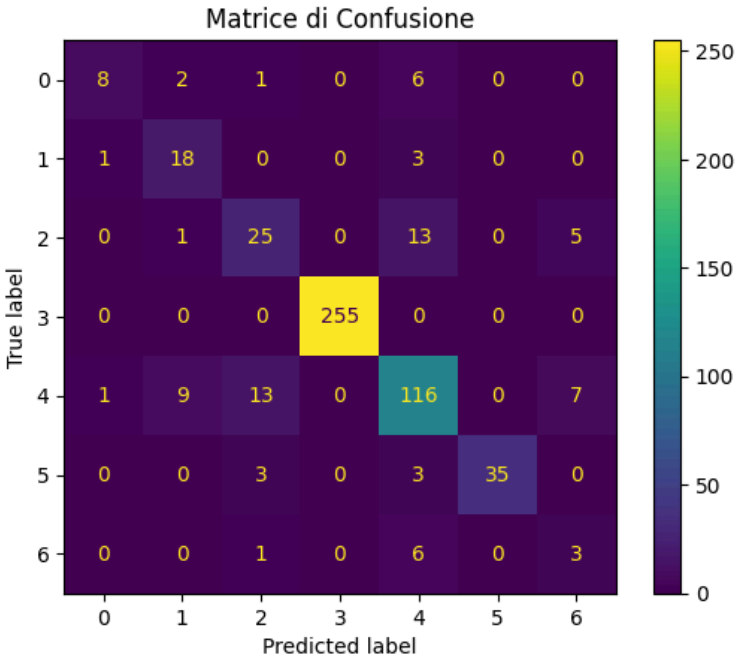
Accuratezza sul set di test (df2): 0.780



5.2.3. df_3

Metriche Aggregate:
Precisione macro: 0.700
Recall macro: 0.723
F1-score macro: 0.709
Precisione weighted: 0.866
Recall weighted: 0.869
F1-score weighted: 0.867

Accuratezza sul set di test (df_3): 0.882



5.2.4. Considerazioni e Analisi

- Anche qui, in generale, le metriche sul test set sono sempre simili a quelle del validation set -> il modello ha generalizzato bene.
- **df_1** (dataset sporco): **0.869** di accuratezza
- **df_2** (dataset pulito, solo feature testuali): **0.780** di accuratezza
- **df_3** (dataset pulito, tutte le feature): **0.882** di accuratezza

ANALISI:

- **df_1 (baseline):**
  Questo dataset offre già buone performance, con metriche macro e ponderate che indicano un buon bilanciamento tra le classi minoritarie e maggioritarie. Il Decision Tree, noto per la sua capacità di gestire dati eterogenei, si adatta bene a questo scenario. Le metriche weighted elevate mostrano un'ottima capacità del modello di gestire le classi più rappresentate, mentre le metriche macro solide indicano una buona predizione anche per le classi meno frequenti.
- **df_2 (dati parzialmente puliti):**
  Le metriche più basse su questo dataset, in particolare il recall macro e l'F1-score macro, segnalano una perdita significativa di informazioni chiave durante la pulizia. La riduzione delle feature sembra aver reso il modello meno efficace nel prevedere correttamente le classi minoritarie. Anche le metriche ponderate, più influenzate dalle classi maggioritarie, subiscono un calo marcato, suggerendo che il Decision Tree risente fortemente della perdita di complessità nei dati.
- **df_3 (dati completamente puliti):**
  Le metriche su df_3 sono molto simili a quelle di df_1, indicando che il preprocessamento ha preservato o migliorato la qualità dei dati rispetto alla baseline. L'accuratezza elevata e le metriche macro stabili mostrano che il modello è ben bilanciato e capace di generalizzare. Questo dataset rappresenta una versione più robusta e raffinata di df_1, con una minore variabilità interna, rendendolo un'opzione migliore per il training del modello.

5.3. Random Forest

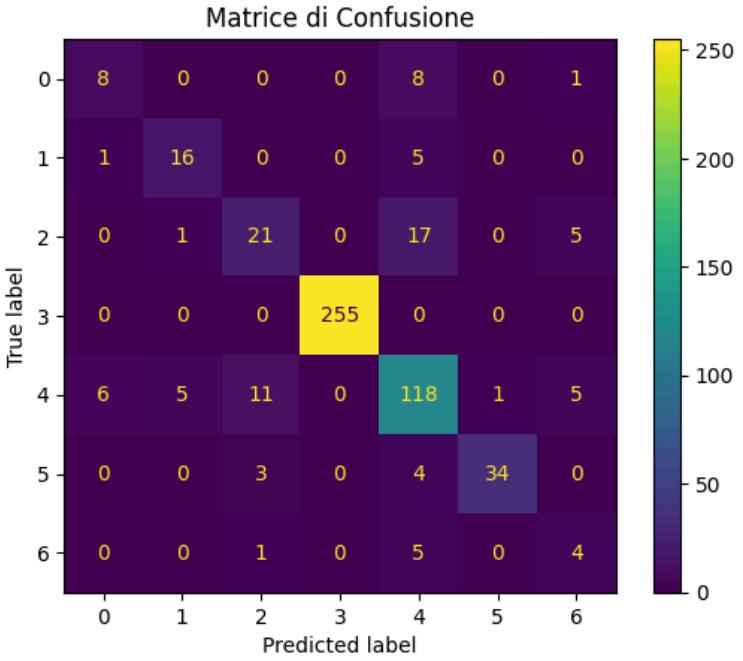
Abbiamo scelto di utilizzare il modello **Random Forest** per la classificazione poiché è un metodo di ensemble learning che combina più alberi decisionali per migliorare le previsioni e ridurre il rischio di overfitting. Questo modello è efficace nel gestire grandi quantità di dati con variabili numeriche e categoriche, ed è robusto nel trattare dati complessi e non lineari.

A differenza di singoli alberi decisionali, Random Forest gestisce meglio la varianza e gli errori dovuti a sovradattamento. Inoltre, non richiede un'accurata pre-elaborazione dei dati, come la normalizzazione. Per queste ragioni, Random Forest è una scelta adatta per problemi di classificazione con dati reali, come il nostro.

5.3.1. df_1

Metriche Aggregate:
Precisione macro: 0.700
Recall macro: 0.723
F1-score macro: 0.709
Precisione weighted: 0.866
Recall weighted: 0.869
F1-score weighted: 0.867

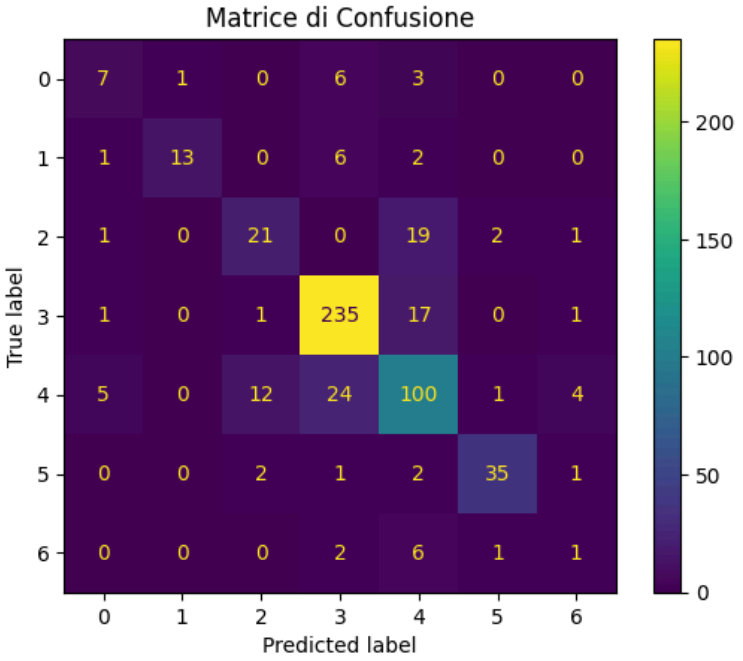
Accuratezza sul set di test (df1): 0.800



5.3.2. df_2

Metriche Aggregate:
Precisione macro: 0.626
Recall macro: 0.614
F1-score macro: 0.618
Precisione weighted: 0.772
Recall weighted: 0.780
F1-score weighted: 0.775

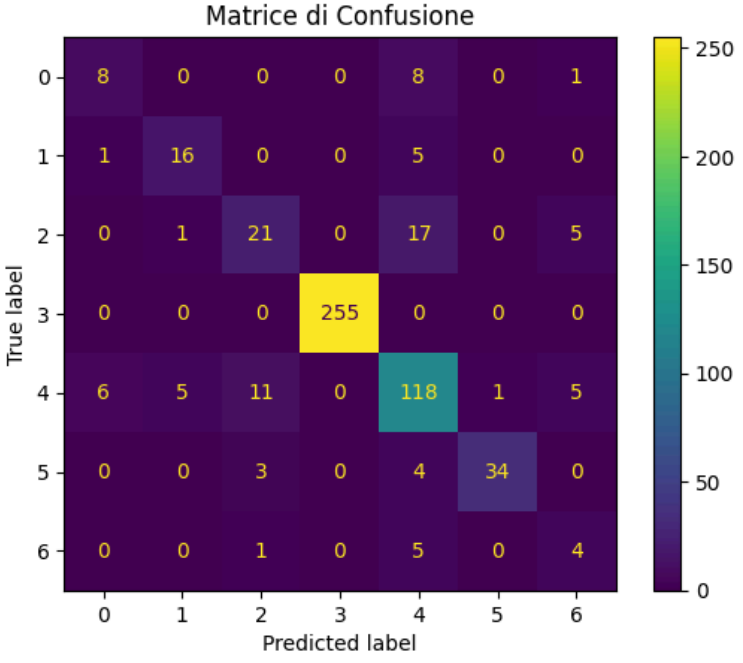
Accuratezza sul set di test (df2): 0.780



5.3.3. df_3

Metriche Aggregate:
Precisione macro: 0.700
Recall macro: 0.723
F1-score macro: 0.709
Precisione weighted: 0.866
Recall weighted: 0.869
F1-score weighted: 0.867

Accuratezza sul set di test (df3): 0.890



5.3.4. Considerazioni e Analisi

- Anche qui, in generale, le metriche sul test set sono sempre simili a quelle del validation set -> il modello ha generalizzato bene.
- **df_1** (dataset sporco): **0.800** di accuratezza
- **df_2** (dataset pulito, solo feature testuali): **0.780** di accuratezza
- **df_3** (dataset pulito, tutte le feature): **0.890** di accuratezza

ANALISI:

- **df_1 (baseline):**
  L'accuratezza e le metriche aggregate sono buone, ma inferiori rispetto agli altri modelli analizzati su questo dataset. Questo potrebbe indicare che il Random Forest fatica a sfruttare appieno le caratteristiche del dataset baseline. Nonostante la sua robustezza, il modello sembra meno efficace rispetto a SVM e Decision Tree, probabilmente a causa di una ridondanza o di una mancanza di profondità nelle feature disponibili.
- **df_2 (dati parzialmente puliti):**
  Il calo significativo delle metriche aggregate rispetto a df_1 indica che l'utilizzo di un numero ridotto di feature ha compromesso la capacità del modello di generalizzare. Il recall macro particolarmente basso riflette la difficoltà del Random Forest nel prevedere correttamente le classi minoritarie quando le informazioni disponibili sono limitate. Questo risultato sottolinea che il Random Forest necessita di dati più ricchi e diversificati per costruire alberi ben bilanciati.
- **df_3 (dati completamente puliti):**
  Le metriche tornano ai livelli di df_1, con prestazioni che suggeriscono un netto miglioramento dovuto alla qualità e completezza del dataset. Il preprocessamento e l'inclusione di feature aggiuntive hanno migliorato la capacità del modello di generalizzare e di bilanciare le predizioni tra classi maggioritarie e minoritarie. Questo risultato conferma che il Random Forest è altamente sensibile alla qualità del dataset e beneficia notevolmente di una pulizia e selezione delle feature adeguate.

Il Random Forest beneficia notevolmente dell'uso di un dataset completo e pulito, mostrando le migliori prestazioni con il dataset finale.

6. Considerazioni Finali

Nel corso di questo lavoro, abbiamo analizzato le prestazioni di tre modelli di machine learning — Support Vector Machine (SVM), Decision Tree e Random Forest — applicati a un dataset sbilanciato e valutati su tre diverse versioni dello stesso dataset (df_1, df_2 e df_3). I risultati hanno permesso di trarre alcune considerazioni chiave.

Effetto della qualità del dataset sulle prestazioni del modello
I risultati ottenuti mostrano che la qualità del dataset è un fattore cruciale per migliorare le performance dei modelli. In particolare:

- **df_3**, il dataset più completo e pulito, ha prodotto le metriche più alte per tutti i modelli analizzati, evidenziando l'importanza del preprocessing.
- Al contrario, **df_2**, che includeva solo due feature principali ("name" e "description"), ha portato a un calo significativo delle performance, specialmente nel caso del Random Forest e del Decision Tree. Questo conferma che la perdita di informazioni rilevanti nel dataset può penalizzare i modelli, rendendo difficile la generalizzazione.

Impatto delle feature sulla performance del modello
La riduzione del numero di feature disponibili ha influenzato significativamente i risultati:

- Su **df_2**, l'accuratezza è calata notevolmente per tutti i modelli, con un impatto maggiore sul Random Forest, che si basa su un ampio insieme di feature per costruire alberi ben bilanciati.
- Al contrario, un insieme completo di feature in **df_3** ha migliorato la capacità dei modelli di generalizzare, evidenziando come un approccio ricco di feature possa favorire una classificazione più precisa.

Performance dei modelli

- **SVM** si è dimostrato il modello più consistente, con buone prestazioni sia su **df_1** che su **df_3**. Su **df_2**, pur subendo un calo, ha mantenuto un equilibrio tra precisione e recall, gestendo bene le classi sbilanciate.
- **Decision Tree** ha mostrato risultati discreti su **df_1** e **df_3**, ma su **df_2** ha registrato un calo significativo, indicando una maggiore vulnerabilità alla riduzione delle feature rispetto agli altri modelli.
- **Random Forest** ha ottenuto performance simili al Decision Tree su **df_1** e **df_2**, ma ha beneficiato maggiormente del dataset completo e pulito di **df_3**, grazie alla sua capacità di combinare più alberi e ridurre l'overfitting. Tuttavia, non ha mai raggiunto i livelli di performance dell'SVM.

6.1. Sviluppi Futuri

Nonostante i buoni risultati ottenuti durante la fase di valutazione, ci sono alcuni aspetti che possono essere migliorati, tanto nella fase di preprocessamento dei dati quanto nell'esplorazione di nuove tecniche e metodi per migliorare ulteriormente le prestazioni del modello.

1. **Esplorazione di nuove architetture di machine learning**
    - **Reti Neurali Profonde (Deep Learning)**: la sperimentazione con modelli di deep learning come le **Convolutional Neural Networks** (CNN) o le **Recurrent Neural Networks** (RNN) potrebbe migliorare ulteriormente la capacità del modello di catturare informazioni complesse nei dati testuali o nelle sequenze, come nel caso delle recensioni.
2. **Ottimizzazione dei modelli**
    - L'integrazione di tecniche di **hyperparameter tuning** come la **grid search** o la **random search** potrebbe portare a un miglioramento significativo nelle performance, migliorando i risultati tramite la scelta ottimale dei parametri per ogni modello.
    - La **regolarizzazione** nei modelli SVM o Decision Tree potrebbe aiutare a migliorare la generalizzazione e a ridurre l'overfitting, particolarmente nei dataset piccoli o sbilanciati.
3. **Data Augmentation e Costruzione di nuovi dataset**
    - La **generazione artificiale di dati**, come nel caso di tecniche di **data augmentation**, potrebbe essere una strada interessante da esplorare. Metodi come la **morfologia dei dati testuali**, aggiungendo variabilità alle recensioni esistenti, potrebbero arricchire ulteriormente il dataset.
    - Un'altra opzione potrebbe essere la raccolta di un dataset con una maggiore diversità e un migliore bilanciamento fra le classi, in modo da permettere una più equa valutazione delle performance dei modelli.
4. **Miglioramento delle metriche in scenari sbilanciati**
    - Infine, durante il processo di addestramento, l'adozione di **techniques di bilanciamento delle classi**, come il **SMOTE** (Synthetic Minority Over-sampling Technique), potrebbe migliorare le performance in scenari sbilanciati, aumentando la robustezza e le capacità predittive del modello in relazione alle classi minoritarie.
5. **Automazione nel Preprocessamento dei Dati**
   Per migliorare ulteriormente l'efficienza e la qualità del dataset, un'area di sviluppo futuro potrebbe essere l'integrazione di sistemi automatizzati per la gestione dei dati, tra cui:
    - **Correzione automatica dei typo:** utilizzando modelli pre-addestrati di Natural Language Processing (NLP), come **spell**

checkers o algoritmi basati su dizionari, è possibile correggere errori ortografici comuni, migliorando la coerenza e l'accuratezza delle feature testuali.
    - **Riconciliazione delle entità:** identificare e normalizzare automaticamente i valori categoriali (ad esempio, raggruppare "Ministero della Salute" e "Miniistero della Salute") tramite algoritmi di fuzzy matching come `fuzzywuzzy` o similari.
    - **Rilevamento automatico delle inconsistenze:** implementare script per confrontare attributi chiave, come latitudine/longitudine rispetto alle categorie, per individuare anomalie logiche o duplicazioni.
6. **Automazione nella Pipeline del Modello**
   Un'altra area di sviluppo interessante sarebbe automatizzare il **monitoraggio delle performance del modello** e l'aggiornamento continuo:
    - **Pipeline MLOps:** integrare una pipeline che automatizzi la raccolta, il preprocessing e il retraining del modello, assicurando che rimanga aggiornato rispetto ai nuovi dati raccolti.
    - **Validazione automatica dei dati:** implementare controlli di qualità dinamici durante l'acquisizione dei dati da OpenStreetMap, rilevando e scartando i POI rumorosi o con informazioni incomplete.