Damiano Taricone - Relazione Progetto Machine Learning

Introduzione

Nel contesto dell'analisi dei media social e del marketing digitale, comprendere l'impatto dei contenuti pubblicati è fondamentale per ottimizzare l'engagement e la portata dell'influencer. In particolare, l'analisi dei video pubblicati da influencer su piattaforme come YouTube può fornire insight preziosi sul comportamento degli utenti e sull'efficacia del contenuto.

Questo progetto mira a costruire e analizzare un dataset composto da 500 video di un influencer selezionato, per esplorare la relazione tra vari fattori, come la data di pubblicazione, il numero di visualizzazioni, la durata, e il numero di commenti, e l'impatto che hanno sul rapporto tra like e visualizzazioni. Questo rapporto è un indicatore chiave dell'engagement del pubblico e della popolarità del contenuto.

L'obiettivo principale è sviluppare un modello di machine learning che possa predire il rapporto like/visualizzazioni, utilizzando come feature l'età del video (calcolata dalla data di pubblicazione), il numero di visualizzazioni, la durata del video e il rapporto commenti/visualizzazioni.

Raccolta del Dataset

Per la costruzione del dataset, ho utilizzato un approccio programmatico per estrarre le informazioni dai video di YouTube. Il processo di raccolta dei dati è stato eseguito utilizzando un linguaggio di scripting e l'API di YouTube per accedere ai dettagli specifici dei video. Di seguito sono riportati i passaggi chiave del processo:

Creazione del File CSV
Inizialmente, ho creato un file CSV denominato `youtube_videos_info.csv`, che funge da contenitore per i dati raccolti. Il file è stato preparato per accogliere le seguenti colonne: `Title`, `Publish_Date`, `Likes`, `Views`, `Duration`, `Comment_Count`. Queste colonne rappresentano rispettivamente il titolo del video, la data di pubblicazione, il numero di like, il numero di visualizzazioni, la durata del video e il numero di commenti.

Estrazione dei Dati tramite API di YouTube
Per ogni video, identificato da un `video_id` unico, ho effettuato una richiesta all'API di YouTube per estrarre le informazioni pertinenti. Il processo è stato iterato per un elenco predefinito di `video_ids` che rappresentano i video pubblicati dall'influencer selezionato.

Scrittura nel File CSV
Per ogni video, i dettagli estratti sono stati scritti nel file CSV. Questo include i dati come il titolo del video, la data di pubblicazione, il numero di like e di visualizzazioni, la durata e il numero di commenti. In caso di successo nell'estrazione delle informazioni, i dati sono stati aggiunti come una nuova riga nel file CSV. Nel caso in cui le informazioni di un video non fossero disponibili o l'ID del video non fosse valido, è stato generato un messaggio di errore.

Gestione delle Eccezioni

Durante il processo di estrazione dei dati, ho implementato controlli per gestire situazioni in cui un video potrebbe non essere stato trovato o le informazioni non fossero disponibili. In tali casi, il codice gestisce queste eccezioni senza interrompere l'intero processo di raccolta dei dati, assicurando così la continuità dell'operazione e la massimizzazione della raccolta dei dati disponibili.

Risultato Finale
Al termine del processo, il file CSV risultante contiene una raccolta completa dei dati richiesti per i 500 video, pronti per essere analizzati ulteriormente.

```python
def get_video_ids(youtube, channel_id):
    video_ids = []
    next_page_token = None

    while True:
        request = youtube.search().list(
            part='id',
            channelId=channel_id,
            maxResults=50,
            pageToken=next_page_token,
            type='video'
        )
        response = request.execute()

        video_ids += [item['id']['videoId'] for item in response['items']]

        next_page_token = response.get('nextPageToken')
        if next_page_token is None:
            break

    return video_ids
```

```python
def get_youtube_video_info(api_key, video_id):
    # Configurazione del client API di YouTube
    youtube = build( serviceName: 'youtube',  version: 'v3', developerKey=api_key)

    # Richiesta all'API di YouTube
    request = youtube.videos().list(
        part="snippet,statistics,contentDetails",
        id=video_id
    )
    response = request.execute()

    # Controllo se il video è stato trovato e estrazione dei dati
    if response['items']:
        video_data = response['items'][0]
        info = {
            'title': video_data['snippet']['title'],
            'publish_date': video_data['snippet']['publishedAt'],
            'likes': video_data['statistics'].get('likeCount', 'N/A'),
            'views': video_data['statistics'].get('viewCount', 'N/A'),
            'duration': video_data['contentDetails']['duration'],
            'comment_count': video_data['statistics'].get('commentCount', 'N/A')
        }
        return info
    else:
        return "Video non trovato."
```

Pre-elaborazione dei Dati

Dopo aver raccolto i dati necessari, il prossimo passo è stato preparare e trasformare questi dati in un formato adatto per l'analisi e la modellazione. La pre-elaborazione dei dati è un passaggio cruciale in qualsiasi progetto di data science, poiché garantisce che l'input per il modello di machine learning sia pulito, corretto e in un formato utilizzabile. Di seguito sono riportate le principali fasi di pre-elaborazione che ho implementato:

Conversione della Durata del Video
I dati originali includevano la durata dei video in formato ISO 8601. Ho utilizzato una funzione personalizzata, `convert_duration_to_seconds_iso`, per convertire questo formato in secondi totali, rendendo così la durata dei video in un formato numerico e standardizzato per l'analisi.

Trattamento della Data di Pubblicazione
Le date di pubblicazione dei video sono state convertite in oggetti `datetime` per facilitarne la manipolazione. Inoltre, ho impostato la data attuale come `datetime` consapevole del fuso orario (tz-aware) in UTC, per mantenere la coerenza nel calcolo dell'età del video.

Calcolo dell'Età del Video
L'età del video è stata calcolata come la differenza in giorni tra la data attuale e la data di pubblicazione del video. Questa trasformazione è cruciale per analizzare l'impatto dell'età del video sul suo engagement.

Calcolo dei Rapporti
Ho poi calcolato due rapporti chiave:
- **Rapporto Commenti/Visualizzazioni**: Questo è stato calcolato dividendo il numero di commenti per il numero di visualizzazioni, fornendo un'indicazione del livello di interazione degli utenti con il video.
- **Rapporto Like/Visualizzazioni:** Analogamente, il rapporto tra like e visualizzazioni è stato calcolato per valutare la popolarità e l'accettazione del contenuto del video.

Categorizzazione del Rapporto Like/Visualizzazioni
Per facilitare l'analisi, ho categorizzato il rapporto like/visualizzazioni in 8 categorie distinte utilizzando la tecnica di binning quantile. Questo aiuta a semplificare l'analisi e fornisce una base per una classificazione più dettagliata nell'analisi successiva.

Salvataggio del Dataset Trasformato
Infine, il dataset trasformato e pronto per l'analisi è stato salvato in un nuovo file CSV, `youtube_updated.csv`, per un facile accesso e utilizzo nelle fasi successive dell'analisi.

Guardando i boxplot caricati, possiamo commentare l'effetto della standardizzazione e della rimozione degli outlier sui dati:

Data Pertubation

Ho iniziato selezionando alcune colonne numeriche dal mio dataset di YouTube, come "Likes", "Views", e altre metriche simili. Il mio obiettivo era introdurre delle variazioni in questi dati per testare come queste piccole alterazioni potrebbero influenzare eventuali analisi o modelli di machine learning che avrei potuto creare in seguito.

Per fare questo, ho scelto di applicare un tipo di perturbazione chiamata rumore gaussiano. Il rumore gaussiano è interessante perché segue una distribuzione normale, molto comune in molti fenomeni naturali e sociali. Ho scritto una funzione, `add_gaussian_noise`, per aggiungere questo rumore ai miei dati in modo controllato. Questa funzione prende una serie di dati e aggiunge una variazione casuale a ogni punto dati, basata sul livello di rumore che ho deciso di impostare.

Ho fatto attenzione a controllare il livello di questo rumore, utilizzando il parametro `noise_level`. Era importante per me che la perturbazione fosse realistica e non troppo eccessiva, per mantenere l'integrità dei dati ma allo stesso tempo introdurre variazioni utili.

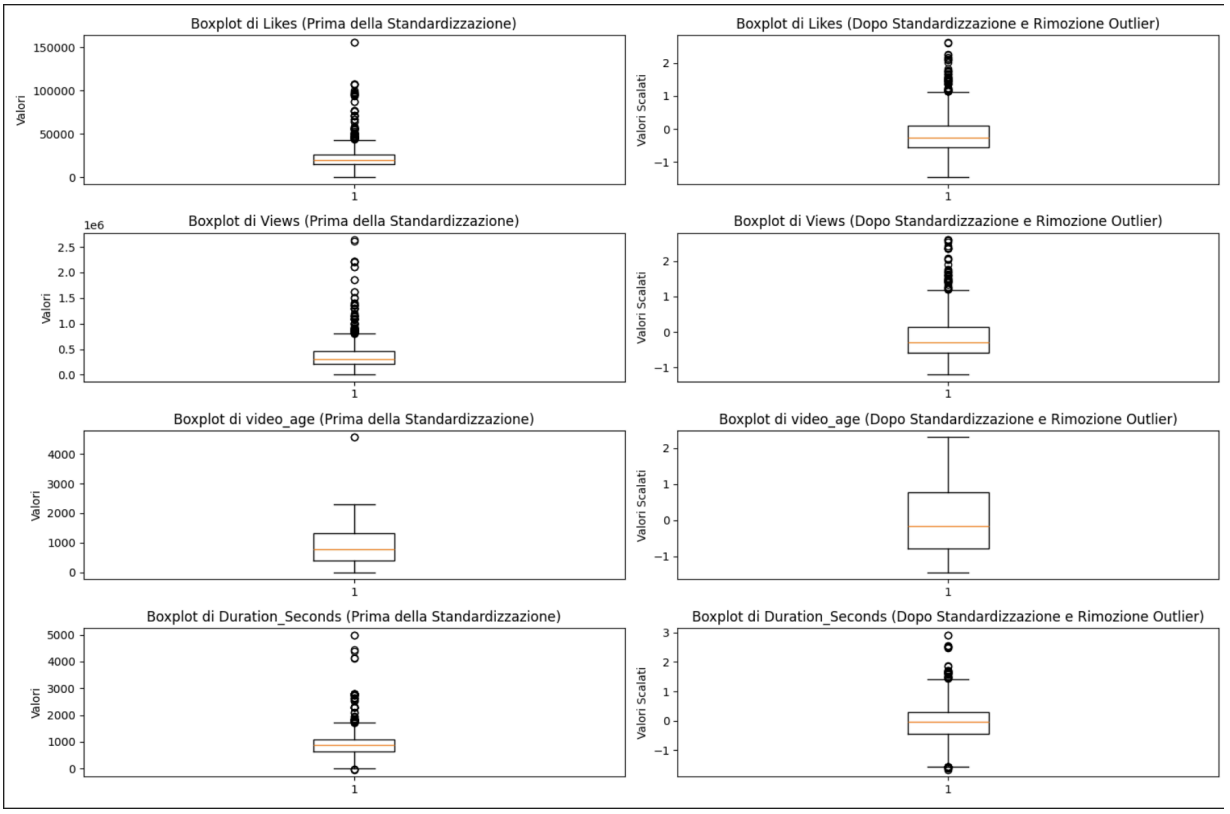
Con questa metodologia, ho potuto creare una versione del mio dataset che non solo manteneva le tendenze e le caratteristiche originali dei dati di YouTube, ma includeva anche una varietà di piccole alterazioni. Questo approccio è particolarmente utile nel campo del machine learning: aiuta a costruire modelli che sono più robusti e in grado di generalizzare meglio, evitando il rischio di sovra-adattamento ai dati originali.

In breve, ho utilizzato la perturbazione dei dati per arricchire e migliorare il mio dataset di concatenando il dataset con rumore con quello iniziale, rendendolo un candidato più forte per analisi e modellazione predittiva.

Analisi Esplorativa dei Dati

La standardizzazione e la rimozione degli outlier hanno reso le distribuzioni dei dati più uniformi e facilmente confrontabili. Senza gli outlier, i pattern nei dati possono essere osservati più chiaramente, e qualsiasi modello di machine learning applicato in seguito avrà meno probabilità di essere influenzato da valori anomali che potrebbero alterare il processo di apprendimento. Questo passaggio di pre-elaborazione è cruciale per preparare il dataset a un'analisi più approfondita e per costruire un modello predittivo efficace.

Il confronto tra i boxplot prima e dopo la pre-elaborazione mostra l'impatto significativo che tali tecniche hanno sulla forma e sulla distribuzione dei dati, enfatizzando l'importanza di una corretta pre-elaborazione prima di qualsiasi analisi successiva.



Scelta Modello ML

Dopo aver diviso il dataset ho scelto le feature richieste dal problema per la predizione : `'video_age', 'Duration_Seconds', 'Views','Comment_View_Ratio'`

inizialmente mi sono concentrato sull'addestramento e sulla valutazione di diversi modelli di classificazione: KNN, Random Forest, SVM, Logistic Regression, Naive Bayes e Gradient Boosting. Ogni modello ha le sue peculiarità, quindi ero curioso di vedere come avrebbero performato sui miei dati.

K-Nearest Neighbors Accuracy: 0.4368421052631579
Random Forest Accuracy: 0.8368421052631579
SVM Accuracy: 0.4473684210526316
Logistic Regression Accuracy: 0.33157894736842103
Naive Bayes Accuracy: 0.32105263157894737
Gradient Boosting Accuracy: 0.7210526315789474

Dopo aver valutato i modelli base, ho esplorato ulteriormente il modello Random Forest poichè sembrava il più performante, attraverso tecniche di ottimizzazione dei parametri come Grid Search e Random Search. Questo mi ha permesso di trovare la migliore combinazione di parametri per migliorare le prestazioni del modello.

Ho calcolato diverse metriche di prestazione come accuratezza, precisione, recall e F1 score, sia per i modelli base che per quelli con parametri ottimizzati. Ho anche utilizzato la cross-validation per ottenere una valutazione più affidabile.

Performance del modello Random Forest (Grid Search):
Accuracy: 0.8368421052631579
Precision: 0.8465172112370388
Recall: 0.8413551454584063
F1 Score: 0.8384580630393041
Cross-Validation Scores: [0.39473684 0.42105263 0.55263158 0.31578947 0.47368421]
Mean CV Score: 0.43157894736842106

Performance del modello Random Forest (Random Search):
Accuracy: 0.8315789473684211
Precision: 0.8395668356120394
Recall: 0.835558044009131
F1 Score: 0.8321585522912249

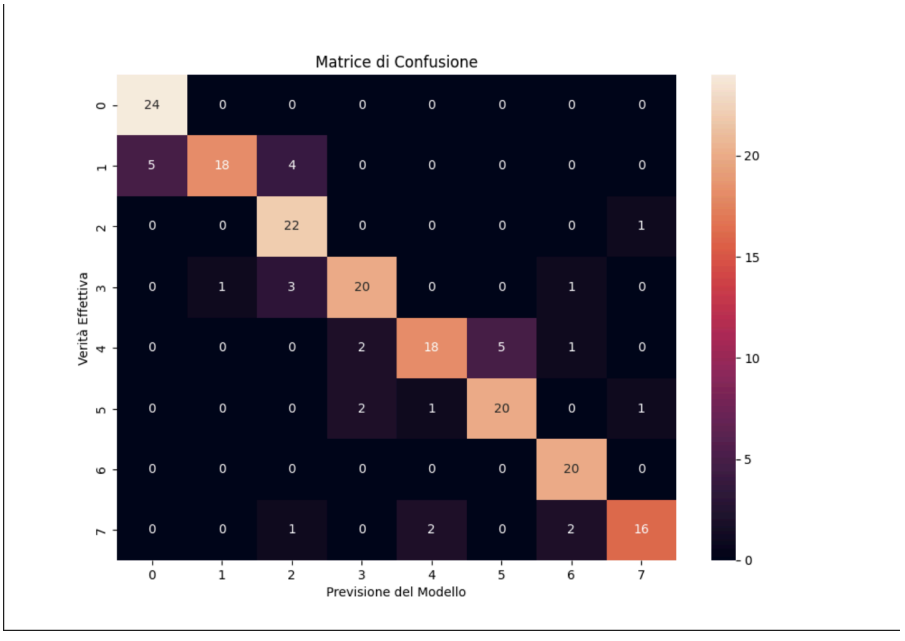
Cross-Validation Scores: [0.39473684 0.44736842 0.55263158 0.31578947 0.47368421]
Mean CV Score: 0.436842105263158

i parametri calcolati sembrano riscuotere un buon risultato ma la cross validation in tutti e due i casi presenta una media notevolmente bassa rispetto agli altri valori quindi ho cercato di ottimizzare il mio modello.
Ho adottato un approccio più sofisticato con il modello di stacking. Ho combinato diversi modelli di base con un meta-classificatore, la regressione logistica, per vedere se potevo migliorare ulteriormente le prestazioni. Dopo aver addestrato e valutato il modello di stacking, ho anche eseguito una visualizzazione della matrice di confusione per avere un'idea migliore di come il modello stesse classificando le diverse categorie.

Infine, ho ottimizzato ulteriormente alcuni dei modelli base (RandomForest e SVC) utilizzati nel mio stacking attraverso Grid Search. Poi ho riaddestrato il modello di stacking con questi modelli base ottimizzati per vedere se potevo ottenere prestazioni ancora migliori.

In sintesi, ho eseguito un ciclo completo di machine learning, dalla preparazione dei dati fino all'addestramento e alla valutazione di diversi modelli, passando per l'ottimizzazione dei parametri e arrivando all'impiego di tecniche avanzate come lo stacking. Questo processo mi ha fornito un'ampia comprensione di come differenti approcci e modelli possono essere utilizzati e combinati per affrontare un problema di machine learning.

L'accuratezza del modello di Stacking è: 0.84
Precisione del modello di Stacking: 0.84
Ricordo del modello di Stacking: 0.84
F1 Score del modello di Stacking: 0.84
Cross-Validation Scores del modello di Stacking: [0.76315789 0.76973684 0.71710526 0.74342105 0.67763158]
Media Cross-Validation Score del modello di Stacking: 0.73



Analisi delle Prestazioni del Modello di Stacking

- *Accuracy*: L'accuratezza del modello di Stacking è del 84%. Questo è un risultato molto positivo, indicando che il modello è in grado di classificare correttamente le istanze nell'84% dei casi. È una buona performance, specialmente per un modello che combina le previsioni di più modelli base.

- *Precisione, Recall e F1 Score*: La precisione, il recall e l'F1 score sono tutti e tre pari a 0.84. Questo indica un eccellente equilibrio tra la capacità del modello di identificare correttamente le classi positive (precisione) e la sua abilità di catturare una grande proporzione delle classi positive effettive (recall). Un F1 score alto è particolarmente importante in situazioni dove è cruciale mantenere un buon equilibrio tra false positive e false negative.

- *Cross-Validation Scores*: I punteggi di cross-validation variano da un minimo di 0.6776 a un massimo di 0.7697, con una media di 0.73. Questa variazione è un po' più ampia di quanto mi aspetterei, suggerendo che il modello potrebbe non essere completamente stabile attraverso diversi subset di dati. La media di 0.73, sebbene sia inferiore all'accuratezza su un singolo set di test, è ancora una stima ragionevole delle prestazioni del modello.

Considerazioni Personali

Sono complessivamente soddisfatto delle prestazioni del mio modello di Stacking. Tuttavia, la discrepanza tra l'accuratezza sul set di test e la media dei punteggi di cross-validation mi suggerisce che il modello potrebbe beneficiare di ulteriori ottimizzazioni per migliorare la sua stabilità e affidabilità su diversi tipi di dati. La principale problematica è dovuta dalla scarsità dei dati che mi ha portato ad utilizzare tecniche di sovracampionamento che hanno sicuramente influenzato negativamente sulla cross validation portandomi ad un rischio di overfitting.