UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Laurea in Informatica

Natural Language Processing e classificazione: modelli predittivi per l'analisi di una comunicazione efficace in ambito Cultural Heritage.

Candidato:
Enrico Fiorini

Relatori:
Prof. Riccardo Martoglia
Prof.ssa Manuela Montangero

Anno Accademico 2020/2021

RINGRAZIAMENTI

I miei ringraziamenti vanno in primo luogo al Professor Riccardo Martoglia e alla Professoressa Manuela Montangero per avermi guidato ed aiutato nella fase finale di conseguimento del titolo, ed in ultima istanza durante la scrittura della tesi.

Ringrazio il Professor Mauro Andreolini, per avermi pazientemente formato in ambito Cybersecurity e guidato in ambito professionale e lavorativo.

Ringrazio mia mamma, mio fratello Francesco e le mie sorelle Azzurra e Vittoria, per avermi sempre sostenuto, ed avermi supportato durante questo percorso che mi ha regalato tante emozioni. Un ringraziamento speciale va a mio fratello, che è stata la figura più presente, e sostenendo scherzosamente che “se ti laurei tu, possono farlo proprio tutti”, mi ha sempre strappato un sorriso.

Ringrazio Carmine, per essere stato presente in ogni momento di questo percorso, rivelandosi in primo luogo un collega valido, e un ottimo amico. Lo ringrazio soprattutto per avermi erudito con la sua gestione maniacale delle repositories git.

Concludo ringraziando Dario e Francesco, e chiunque in un modo o nell’altro abbia incontrato in questo percorso.

PAROLE CHIAVE

Modello
Analisi
Cultural
Heritage
Python

INDICE

Introduzione

Lo studio è suddiviso in due parti, nella prima parte verrà presentato il caso di studio, nella seconda parte, verranno invece dettagliate le scelte implementative del modello predittivo, ed i risultati ottenuti.

Nella parte I, vengono introdotti gli strumenti utilizzati, la situazione di partenza, ed i dati sui quali le analisi verranno svolte. In particolare, si definiranno gli obiettivi del presente lavoro, la natura del problema, e risultati ottenuti antecedentemente, in modo da gettare le basi di partenza per quanto svolto per migliorare l'efficacia predittiva del modello preesistente.

Nella parte II, vengono dettagliate le scelte implementative, in particolare questa parte vedrà in primis il focus sul processo di automazione di creazione del dataset, elemento fondamentale per poter svolgere analisi utilizzando Machine Learning. Verranno definite le classi di tweet individuate, e come queste trasformate in feature possano migliorare l'accuratezza del modello predittivo preesistente; saranno discusse le regole deterministiche di automazione del processo precedentemente discusso, al fine di esplicitare come sia valutato un tweet, e come esso venga inserito nel dataset in corso di costruzione.

Nel capitolo finale, verranno discussi i risultati ottenuti, in particolare si analizzeranno le tendenze per determinate classi di tweet in correlazione al numero di followers che il museo possiede; proseguendo verranno mostrati i risultati ottenuti utilizzando come feature le classi di tweet determinate, comparando l'accuratezza del modello esistente prima dell'utilizzo delle classi come feature, e l'accuratezza ottenuta utilizzando le medesime come feature del modello predittivo.

Parte I

Il Caso di studio

Introduzione alla Parte I

Lo scopo della parte I è l'introdurre al lettore dell'ambiente rispetto al quale tutto il lavoro di analisi e predizione dati è stato svolto; in particolare il focus del precedentemente citato capitolo, è integrare la comprensione dell'ambiente di provenienza dei dati, gli strumenti utilizzati per l'analisi degli stessi, e le principali caratteristiche del linguaggio e delle librerie utilizzate.

Verranno dunque approfonditi gli aspetti intrinsechi alla natura del problema e degli approcci utilizzati per la creazione di un modello predittivo basato sui dati a disposizione, nonché librerie per manipolazione di big data, come *pandas*, in collaborazione con il linguaggio di scripting Python.

Nel capitolo in questione, vengono inoltre gettate le basi degli approcci definiti per la creazione del dataset di dimensioni cospicue alla base di tutto il lavoro, nonché dell'automatizzazione del processo di creazione del dataset stesso, strumento fondamentale e non derogabile per l'implementazione di un modello predittivo basato su tecniche di Machine Learning.

Capitolo 1

Presentazione del contesto e degli strumenti utilizzati

1.1 Contesto di analisi

Il contesto rispetto al quale lo studio si è svolto, riguarda l'attività su piattaforme social media di musei operanti in ambito Cultural Heritage. In particolare, il social media di riferimento per il quale il modello è stato sviluppato, è Twitter.

L'interesse e lo scopo principale del presente lavoro è il miglioramento della comunicazione sulla già menzionata piattaforma, attraverso l'analisi di dati estratti da svariati musei con una base di followers significativa, al fine di poter ottenere un campione di dati sufficientemente eterogeneo.

Un tweet, oltre all'esplicito messaggio che vuole trasmettere, contiene una quantità di dati e metadati utili ai fini di comprendere meglio gli interessi dei propri followers; per citare qualche parametro, possono essere analizzati il numero di retweet, la fascia oraria di pubblicazione oppure il numero di like ottenuti.

La natura del contesto e dei dati a disposizione, suggerisce che un approccio sensato possa essere dato dall'utilizzo del Machine Learning, volto all'addestramento e al riconoscimento delle caratteristiche intrinseche di un tweet che possa avere un potenziale successo ed essere apprezzato, l'analisi dei dati si rivela essere dunque il punto chiave per poter fare una predizione con un buon grado di accuratezza, rispetto a quali saranno i tweet che avranno successo, e invece quali saranno meno apprezzati.
I dati, inizialmente ottenuti dal social mediante API e richiesta formale, vengono

successivamente organizzati su file testuali, in modo da poter essere organizzati e utilizzati da script e tool per la costruzione di un dataset consistente, strumento fondamentale ed indispensabile, per gettare le basi di un modello predittivo efficace.

Il punto di partenza di questo lavoro è proprio costituito da una grande quantità di dati grezzi, dalla quale si è rivelato necessario poter affinare, in primo luogo manualmente, ed in secondo luogo in modo automatizzato, i dati non strettamente necessari ai fini della predizione, ed invece quei dati fondamentali per poterlo fare.

Il primo passo compiuto per l'implementazione di un modello predittivo, è stata l'analisi manuale dei dati, volta ad individuare le *feature* che potessero potenzialmente portare un tweet al successo, cercando di comprendere quali fossero le sostanziali differenze nei dati provenienti dai diversi musei. Questa analisi manuale dei dati, in prima istanza ha gettato le basi per la successiva automatizzazione della creazione del *dataset* che è stato utilizzato per il presente lavoro.

1.2 Dati Iniziali

Per poter introdurre efficacemente ed in modo completo gli strumenti utilizzati in questo lavoro, si rende necessario introdurre i dati sui quali questi strumenti siano stati utilizzati.

I dati sono fisicamente organizzati in file testuali, a loro volta suddivisi per colone, le quali rappresentano metadati relativi ai tweet analizzati.

Al fine di poter individuare i tweet che meglio si accordano con l'utenza che frequenta i profili Twitter delle istituzioni in ambito cultural heritage, si è reso necessario in prima istanza individuare quali fossero i metadati più significativi collegati ai tweet presenti nei dati iniziali, per citare qualche esempio: numero di retweet, numero di like ricevuti, momento della giornata in cui il tweet è stato postato, il numero di immagini presenti nel tweet, il numero di URL presenti nel tweet e la lunghezza del testo.

Oltre la presenza di metadati puramente numerici, un ulteriore contenuto informativo fondamentale del tweet si rivela essere il testo: esso infatti costituisce il mezzo per eccellenza nella comunicazione, unitamente ad immagini e link di approfondimento, costituisce una sorgente di informazioni non indifferente per l'individuazione delle preferenze dell'utenza, con conseguente potenzialità di individuazione di pattern e feature che possano permettere la formalizzazione e l'implementazione di feature che possano *automaticamente* prevedere con un buon grado di precisione, se il tweet incontrerà o meno i favori del generico follower interessato al profilo.

Non tutti i dati sono però strettamente necessari allo stato attuale del lavoro per il corretto funzionamento del modello; infatti, diversi dati contenuti nei file testuali non vengono al momento utilizzati in modo attivo come input del problema.

In seconda istanza, dati testuali come il testo del tweet, contengono come verrà specificato nella sezione [inserire sezione lemmatizzaione/parsing testo], devono in primo luogo essere preprocessati, al fine di poter ottenere un raffinamento da tutti i caratteri o le parole che potrebbero incidere negativamente sulla qualità predittiva del modello implementato; per citare qualche esempio, verrà rimossa la punteggiatura, verrà effettuata una lemmatizzazione dei verbi, tutte le parole saranno trasformate se necessario in minuscolo, e verranno rimossi i link esterni ed eventuali riferimenti ad immagini se presenti.

1.3 Strumenti utilizzati

1.3.1 Python

Python viene considerato come un linguaggio interpretato di programmazione di alto livello, con una sintassi semplificata, ma non per questo meno efficace, rispetto alla maggior parte degli altri linguaggi.
La sintassi del linguaggio Python viene spesso definita informalmente come *zucchero sintattico,* in quanto rende il codice sorgente estremamente leggibile e compatto.

Tale caratteristica, permette la codifica di istruzioni complesse, che in altri linguaggi risulterebbero ostiche alla comprensione del leggente, in poche righe di codice ben leggibile.

Il linguaggio Python è un linguaggio orientato agli oggetti, il che lo rende idoneo a molteplici utilizzi, tra i principali si possono elencare:

- Web application
- System programming e System scripting
- Mobile application

- Embedded systems
- Test automation
- Machine learming
- Deep learning
- Data scraping

Tra le *feature* di Python rientra il fatto di essere un linguaggio multi-paradigma: ovvero il supportare il paradigma *Object-oriented*, il paradigma *strutturale* e il paradigma *funzionale*. Questa molteplicità di paradigmi utilizzabili permette di sfruttare a pieno l'idea di un paradigma ibrido, con conseguente codifica delle aree dell'applicativo nel paradigma più efficiente, o per rendere più efficiente il processo di maintenance.

Python è un linguaggio a tipizzazione dinamico, ciò consente di lasciare all'interprete il compito di definire in automatico a quale tipo di dato appartiene il contenuto informativo di un idenfiticatore.

Il linguaggio abbraccia inoltre la filosofia *beautiful is better than ugly*, rendendo parte integrante della sintassi, l'indentazione del codice sorgente.

Python nasce inizialmente come un linguaggio di scripting, ma l'enorme quantità di librerie e a sua disposizione, ne consente un utilizzo molto più strutturato e adatto a sviluppare applicativi complessi.

Il linguaggio è stato progettato per essere altamente estensibile tramite moduli. Questa modularità compatta lo ha reso particolarmente popolare come mezzo per aggiungere interfacce programmabili ad applicazioni preesistenti.

Python cerca una sintassi e una grammatica più semplici e meno disordinate, offrendo agli sviluppatori una scelta nella loro metodologia di codifica. In contrasto con la filosofia di altri linguaggi, ovvero "c'è più di un modo per farlo", Python sposa una filosofia "dovrebbe esserci un modo ovvio per farlo, e preferibilmente solo uno".

1.3.2 Pandas

Pandas è una libreria Python che fornisce strutture dati performanti, flessibili e largamente espressive progettate per rendere facile e intuitivo lavorare con dati logicamente relazionali o etichettabili.

La libreria mira a essere il pilastro fondamentale per eseguire analisi dati di applicativi reali utilizzando il linguaggio Python. Inoltre, essa ha l'obiettivo più ampio di diventare lo strumento di analisi/manipolazione dei dati open source più potente e flessibile disponibile in qualsiasi lingua.

Un DataFrame è una struttura dati, che organizza i dati in una matrice di righe e colonne, ricalcando lo stile di un foglio di calcolo. I DataFrame sono una delle strutture dati più comunemente utilizzate nella moderna analisi dei dati, in quanto rappresentano un modo flessibile e intuitivo di archiviare e lavorare con i dati.
Ogni DataFrame contiene un progetto, noto come schema, che definisce il nome e il tipo di dati di ciascuna colonna. Un DataFrame può contenere tipi di dati universali come stringhe o interi, nonché tipi di dati specifici, come StructType.

I valori mancanti o incompleti vengono archiviati come valori *null* in un DataFrame, permettendo di mantenere una solida consistenza nella struttura dati.
Una semplice analogia è che un DataFrame può essere visto come un foglio di calcolo con colonne etichettate.
Le principali caratteristiche della menzionata struttura dati, sono il resizing *inplace* della struttura dati, l'estrazione di valori *unique* da ogni colonna di interesse, oppure la manipolazione selettiva dei dati colonna per colonna.

Per esempio, è possibile manipolare la struttura in questo modo: se si volesse lavorare la colonna della struttura nella quale vengono mantenuti i tweet, per filtrare le stringhe da eventuali impurità, risulta possibile farlo in una sola riga di codice:

```
df["F_CONTENT"] = df["F_CONTENT"].str.replace("'s", "")
```

Figura 0 – Esempio di zucchero sintattico

Un accesso così semplice, diretto e compatto ai dati da analizzare, permette una manipolazione dati estremamente efficiente per gli scopi del presente lavoro.
La libreria *pandas* consente l'utilizzo dei DataFrame in ambiente Python.

1.3.3 Machine Learning

L'apprendimento automatico, o *Machine Learning* è lo studio di algoritmi che possono migliorare automaticamente il proprio apprendimento attraverso l'utilizzo dei dati; il machine learning è considerato parte integrante come una parte integrante della disciplina *Artificial Intelligence*.

L'approccio degli algoritmi di apprendimento automatico, prevede la costruzione di un modello basato su un set di dati campione, detti dati di *training*, tale approccio permette al modello di effettuare previsioni o decisioni senza essere che esso sia esplicitamente programmato per farlo.

L'apprendimento automatico viene utilizzato in un'ampia varietà di applicazioni, come ad esempio:

- Indagini sanitarie e medicina
- Filtraggio di e-mail per identificare potenziale spam
- Riconoscimento vocale e Speech To Text (STT)
- Artificial Vision
- Big Data Analytics

Un sottoinsieme del machine learning è strettamente correlato alla statistica computazionale, esso fornisce una predizione mediante l'utilizzo della computazione; ma

tale sottoinsieme rappresenta soltanto una parte delle più ampie tecniche utilizzate in ambito di apprendimento automatico. Attraverso lo studio dell'ottimizzazione matematica, si ricavano metodi, teoria e domini applicativi nel campo dell'apprendimento automatico. L'analisi esplorativa dei dati, realizzata mediante l'apprendimento automatico non supervisionato, è un campo di studio correlato, chiamato *data mining.*

Alcune implementazioni dell'apprendimento automatico utilizzano dati e reti neurali in un modo che imita il funzionamento di un cervello umano. Nella sua applicazione a problemi reali, ci si riferisce al machine learning con il termine di analisi predittiva.

La libreria utilizzata per l'implementazione del modello del presente lavoro è *XGBoost.*

1.3.4 Classificatore XGBoost

"XGBoost, acronimo di "e**X**treme **G**radient **B**oosting", è una libreria di apprendimento automatico scalabile e distribuita, con albero decisionale a gradiente potenziato (GBDT)." [1]

XGBoost implementa il potenziamento dell'albero parallelo, ed è la principale libreria di *machine learning* per problemi di regressione e classificazione.

Per comprendere XGBoost è fondamentale comprendere prima i concetti e gli algoritmi di apprendimento automatico su cui si essa si basa: supervised machine learning, alberi decisionali, apprendimento d'insieme e potenziamento del gradiente.
Avendo già discusso nella sezione 1.2.4 del machine learning, passeremo ora alla discussione degli alberi decisionali.

Gli alberi decisionali creano un modello che prevede l'etichetta valutando un albero di domande caratteristiche se-allora-altro vero/falso e stimando il numero minimo di domande necessarie per valutare la probabilità di prendere una decisione corretta.

Gli alberi decisionali possono essere utilizzati per la classificazione per prevedere una categoria o per la regressione per prevedere un valore numerico continuo.
Il Gradient Boosting Decision Trees (GBDT) è un algoritmo di apprendimento dell'insieme di alberi decisionali simile alla foresta casuale, per la classificazione e la regressione.
Gli algoritmi di apprendimento dell'insieme combinano più algoritmi di apprendimento automatico per ottenere un modello migliore.
Sia la random forest che il GBDT costruiscono un modello costituito da più alberi decisionali. La differenza sta nel modo in cui gli alberi sono costruiti e combinati; inoltre, costruiscono un modello costituito da più alberi decisionali.

Il termine "aumento del gradiente" deriva dall'idea di "potenziare" o migliorare un singolo modello debole combinandolo con una serie di altri modelli deboli al fine di generare un modello collettivamente più forte, il potenziamento del gradiente è un'estensione del potenziamento in cui il processo di generazione additiva di modelli deboli è formalizzato come un algoritmo di discesa del gradiente su una funzione obiettivo.
Il potenziamento del gradiente imposta risultati mirati per il modello successivo nel tentativo di ridurre al minimo gli errori. I risultati mirati per ciascun caso si basano sul gradiente dell'errore (da cui il nome gradient boosting) rispetto alla previsione.

I GBDT addestrano in modo iterativo un insieme di alberi decisionali poco profondi, con ogni iterazione che utilizza i residui di errore del modello precedente per adattarsi al modello successivo. La previsione finale è una somma ponderata di tutte le previsioni dell'albero. Il "bagging" casuale della foresta riduce al minimo la varianza e l'overfitting, mentre il "boosting" GBDT riduce al minimo la distorsione e l'underfitting.

XGBoost è un'implementazione scalabile e altamente accurata dell'aumento del gradiente che spinge i limiti della potenza di calcolo per algoritmi ad albero potenziati, essendo costruito principalmente per potenziare le prestazioni del modello di apprendimento automatico e la velocità di calcolo.

Con XGBoost, gli alberi vengono costruiti in parallelo, invece che in sequenza come GBDT. Segue una strategia a livello di livello, scansionando i valori del gradiente e utilizzando queste somme parziali per valutare la qualità delle divisioni ad ogni possibile divisione nel set di allenamento.

1.3.5 Named Entity Recognition

“L’acronimo **NER**, noto anche come identificazione di entità con nome, suddivisione in parti di entità ed estrazione di entità, indica un compito secondario dell'estrazione di informazioni che cerca di individuare e classificare le entità nominative menzionate in un testo non strutturato in categorie predefinite” [2], come per esempio:

- Persone
- Nomi
- Organizzazioni
- Località
- Codici sanitari
- Espressioni temporali
- Quantità
- Valori monetari
- Entità governative riconosciute

Il riconoscimento completo delle entità nominative è spesso scomposto, concettualmente e possibilmente anche nelle implementazioni, come due problemi distinti: rilevamento dei nomi e classificazione dei nomi in base al tipo di entità a cui si riferiscono.

La prima fase è tipicamente semplificata a un problema di segmentazione: i nomi sono definiti come intervalli contigui di token, senza annidamento, in modo che "Banca

d’Italia" sia un unico nome, a prescindere dal fatto che all'interno di questo nome, la sottostringa "Italia" è esso stesso un nome, questo problema di sotto segmentazione ed è formalmente simile al macro-problema, secondo l’approccio *divide et impera*.

Anche le espressioni temporali e alcune espressioni numeriche possono essere considerate entità con nome in ambito NER; alcuni esempi di queste espressioni sono buoni esempi di entità rigide, come per esempio anno “2001”, mentre ce ne sono altri ambigui, come ad esempio: vado in vacanza a "giugno".

Nel primo caso, l'anno 2001 si riferisce al 2001° anno del calendario gregoriano, nel secondo caso, il mese di giugno può riferirsi al mese di un anno indefinito, come per esempio lo scorso giugno, oppure il prossimo giugno, o ancora ogni giugno.
La definizione del termine entità denominata non è quindi rigorosa e spesso deve essere spiegata nel contesto in cui viene utilizzata, quindi in sostanza, il riconoscimento dell’entità non è sempre svincolabile dal contesto in cui essa è denominata.

1.3.6 Libreria Spacy

SpaCy è una libreria per il linguaggio Python open source per l’utilizzo di Natural Language Processing, ovvero NLP, in cui molteplici funzionalità sono integrate.

I dati testuali non strutturati, ovvero i testi, vengono utilizzati su larga scala ed è importante elaborarli e ricavarli efficacemente, per fare ciò, è necessario rappresentare tali dati in un contenuto informativo elaborabile da un calcolatore.
Le feature disponibili ed utilizzabili mediante libreria spaCy sono:

- Suddivisione del corpo del testo in token
- Segmentazione del testo in parole e segni di punteggiatura
- Categorizzazione POS, ovvero Part-of-speech

- Assegnazione di tipi di parole a token, come verbo o sostantivo
- Lemmatizzazione
- Sentence Boundary Detection, ovvero individuare e segmentare singole frasi
- Named Entity Recognition
- Entity Linking, ovvero rendere le entità testuali in identificatori univoci in base al contesto di appartenenza.
- Categorizzazione del testo a granularità variabile

Tramite la nominata libreria, è stato implementato lo script di automatizzazione per la costruzione del dataset, basandosi su regole di aderenza rispetto alle entità, che verranno definite nella seconda parte del presente lavoro, pilastro fondamentale per la costruzione dello strumento principe per l'utilizzo di uno strumento come il supervised machine learning.

1.3.7 Scikit-learn

"Scikit-learn, è una libreria open source di apprendimento automatico per il linguaggio di programmazione Python. Essa contiene algoritmi di classificazione, di regressione e clustering, ovvero raggruppamento, e *support-vector machines*, regressione logistica, classificatori bayesiano, k-mean e DBSCAN. Essa è stata progettata per integrarsi con le librerie NumPy e SciPy. " [3]

La libreria è un componente fondamentale del presente lavoro, avendo fornito le primitive necessarie all'addestramento del modello, nonché il supporto di encoding dei dati provenienti dal DataFrame contenente i dati da analizzare.
La libreria inoltre permette l'utilizzo di svariati classificatori, ognuno dei quali espone diverse caratteristiche, permettendo di pre-allenare un modello su scikit, e poi utilizzare

l'oggetto creato su più classificatori, potendo così controllare le diverse performance degli stessi, senza variare i dati di partenza, né forzando l'utilizzatore al paradigma N classificatori corrispondenti a N oggetti sottoposti ad addestramento.

Assumendo **X** e **y** come dati generici dipendenti dal contesto, X come generico dato da classificare, e y come risposta del classificatore possiamo formalizzare il lavoro compiuto da scikit-learn come una funzione così definita:

Definendo $\mathbb{D}$ come il dominio della funzione di traning (i dati di partenza), e $\mathbb{H}$ come il codominio (la classificazione), allora la funzione di addestramento della libreria scikit si occuperà di svolgere il seguente compito, definito matematicamente come
$f: \mathbb{D} \rightarrow \mathbb{H}$**, ovvero** il classificatore categorizza un dato secondo una computazione statistica, nel codominio $\mathbb{H}$**.**

Scendendo nel dettaglio, un classico esempio di training può essere il seguente:

```
        X_train, X_test, y_train, y_test =
              train_test_split(
...                             X, y, test_size=0.33, random_state=42
                               )
```

Figura 1 – Train test split

Dove X e y sono i dati di (rispettivamente) dominio e codominio, *test_size* rappresenta la dimensione del dataset utilizzato per l'addestramento, e *random_state* rappresenta il seme di inizializzazione pseudo-randomica del numero di sotto dataset in cui i dati di partenza verranno divisi.

1.4 Modello di partenza e situazione iniziale

1.4.1 Suddivisione di musei in gruppi

I dati ricevuti sono stati da terze parti selezionati da 23 famosi musei d'arte diffusi nel mondo, con il seguente raggruppamento, suddiviso in cinque gruppi:

- G1: musei con almeno tre milioni di followers
- G2: musei con più di un milione di followers
- G3: musei con più di 400.000 followers
- G4: musei con più di 200.000 followers
- G5: musei italiani

Seguendo tale raggruppamento, l'analisi non verrà influenzata dalle caratteristiche di un singolo ente.

Il numero di tweet per ogni gruppo è rappresentato di sotto nella tabella in Figura 2:

	G1	G2	G3	G4	G5	Total
# Tweets	6097	11017	4936	7808	8233	38091

Figura 2 – Numero di tweet per gruppi

Per il presente lavoro sono stati utilizzati tweet provenienti dai gruppi G1, G2, G3, filtrando tutti i tweet che non fossero in lingua inglese tramite *NLTK*. I gruppi G4 e G5 sono stati scartati in quanto la percentuale di tweet in lingua inglese era di molto inferiore rispetto ai gruppi G1, G2, G3, e la quantità di tweet per alimentare una run di machine learning rispetto alle categorie definite in sezione 2.1, era troppo esigua.

1.4.2 Obiettivo di analisi iniziale e definizione di categorizzazione positiva o negativa di un tweet

Il compito del lavoro inizialmente impostato e svolto da Furini et al. [4] è prevedere se una bozza di tweet scritta da un media manager di un museo avrà successo (quindi una categorizzazione positiva) oppure no (quindi una categorizzazione negativa), qualora venisse pubblicata sul profilo Twitter del museo in questione.

Il progetto originale, in configurazione standard, e con in dati provenienti dai gruppi citati in sezione 1.4.1, produce la seguente tabella di accuratezza per il classificatore di riferimento per il lavoro svolto, ovvero XGBClassifier:

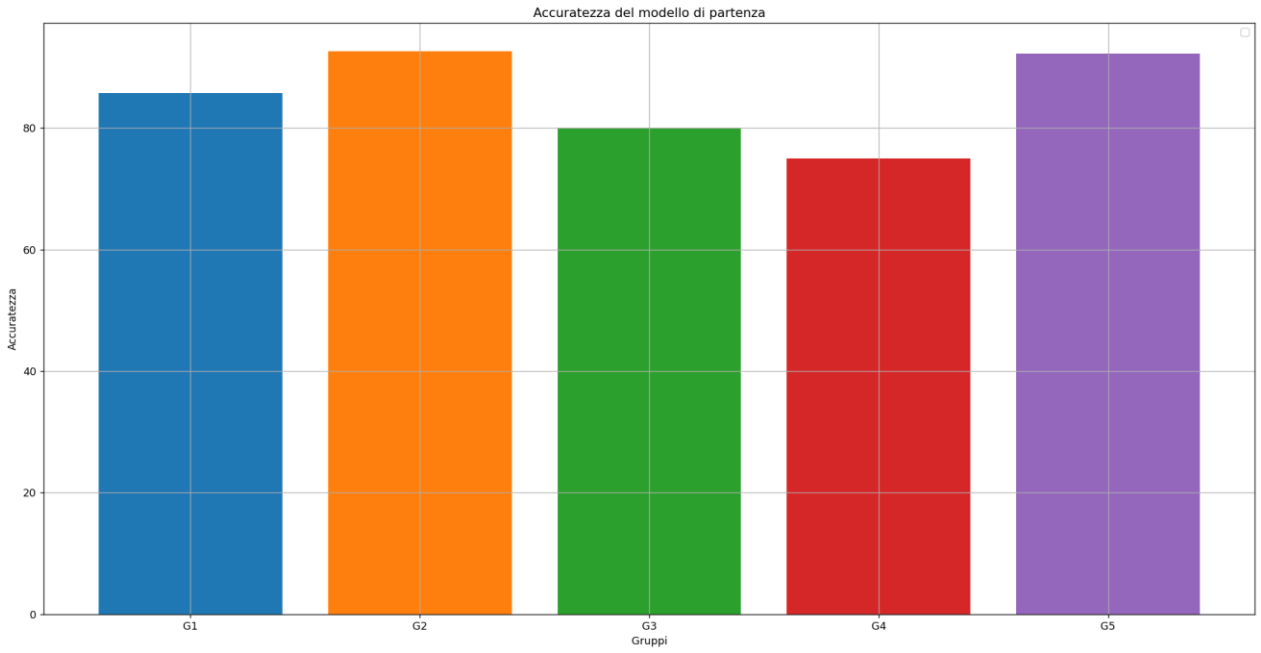

Figura 3 – Accuracy del progetto in versione standard per il classificatore XGBClassifier

L'obiettivo del presente lavoro è quindi il miglioramento per il classificatore XGBClassifier dell'accuratezza predittiva, mediante l'utilizzo delle classi definite in sezione 2.1 come feature di input di XGBoost.

Parte II

Implementazione di un modello predittivo

Introduzione alla Parte II

Lo scopo della parte II è l'immersione del lettore nell'effettiva implementazione del modello predittivo e di tutto l'ecosistema che lo circonda ai fini dell'analisi e predizione dati svolta; in particolare il focus del precedentemente citato capitolo, è la definizione dei dettagli implementativi rispetto ai quali ruota il presente lavoro, nonché l'esposizione dei risultati ottenuti e le considerazioni per le possibili ed eventuali svolte future del modello predittivo.

Verranno dunque approfonditi i metodi di implementazione del modello, l'analisi manuale dei dati che ha portato all'individuazione di *feature* per la creazione di un primo dataset, che ha successivamente permesso l'automazione del processo di creazione di una base di tweet basandosi su un punteggio ottenuto mediante regole di aderenza terminologica e rispetto a entità generiche.

Al termine del secondo capitolo verranno inoltre esposte considerazioni rispetto alla composizione dei dati, alle possibili motivazioni della distribuzione degli stessi in modo eterogeneo rispetto ai diversi gruppi di musei, nonché delle implicazioni derivanti.

Il capitolo finale, si occuperà di dimostrare come le classi di tweet individuate nel secondo capitolo, siano una feature migliorativa in termini di miglioramento dell'accuratezza del modello predittivo esistente.

Capitolo 2

Creazione manuale ed automatizzata del dataset

2.1 Definizione di categorie di tweet

Il primo tassello per la costruzione di un dataset volto all'analisi dati, e la conseguente implementazione di un modello predittivo, è stata l'analisi manuale dei dati contenuti sui file testuali strutturati come descritto nella sezione 1.2.
In prima battuta, la lettura di un sottoinsieme dei tweet, si è rivelata fondamentale per stendere una prima struttura manuale delle potenziali classi di appartenenza dei tweet in esame.

Dalla precedetemene menzionata lettura, sono emerse le seguenti potenziali classi manualmente individuate:

- Promozione di servizi/punti ristoro interni ai musei
- Descrizione di opere d'arte/oggetti esposti/e dal museo
- Offerte commerciali per i biglietti d'ingresso
- Biografia breve di personaggi storici di rilievo
- Promozione di mostre ed eventi interni al museo
- "Guess the artist", contest in cui vengono dati indizi sulle opere d'arte
- relative ad un artista e si deve indovinare chi esso sia
- "Work of The Week", ovvero l'opera d'arte proposta settimanalmente
- Promozioni di eventi live su Twitch/Facebook/Youtube
- Messaggi di auguri per feste/ricorrenze

- Commemorazione di eventi storici passati
- #OnThisDay, commemorazione delle ricorrenze storiche nel giorno del tweet
- Celebrazione ricorrenze di invenzioni scientifiche
- Citazioni personaggi Famosi

La suddivisione nelle presenti categorie, è stato il primo passo fondamentale per il concetto di categorizzazione di un tweet, ma a seguito di una seconda ulteriore analisi, alcune categorie si sono rivelate logicamente e semanticamente sovrapposte, portando pertanto all'accorpamento di più sottocategorie in un'unica entità.

A seguito di questo processo, le categorie di tweet rimanenti sono le seguenti:

- Artworks
- Festivities (inteso come messaggi di auguri per feste e ricorrenze)
- History (celebrazione di eventi storici, o spiegazione di eventi storici correlati ad opere d'arte esposte)
- #OnThisDay
- Promotions
- VIP & CIT (Celebrazione di personaggi storici, frasi a loro riconducibili, motti storici)

Il motivo dell'accorpamento di alcune categorie è stata mancanza di tweet estendibili a tutti i musei, ovvero erano presenti solo per certi gruppi di musei, senza una controparte, rischiando un potenziale sbilanciamento del bilanciamento del dataset, rendendo la distribuzione dei tweet per categoria non omogenea.
Come citato nel precedente paragrafo, è emersa inoltre una sovrapposizione logica tra diverse categorie, come per esempio: promozione di mostre, promozione di punti ristoro, che sono state accorpate nella generica categoria "promozioni generiche".

Al termine dell'individuazione delle categorie di tweet menzionate nella sezione 2.1, si è proceduto con il campionamento di un primo dataset di dimensioni ridotte, 900 tweet totali, inserendoli manualmente in un file in formato .csv, e procedendo con la classificazione manuale, tweet per tweet, di ognuno di essi all'interno del file .csv.
Per istanza, un record del dataset in formato .csv può considerarsi così costruito:

0, OnThisDay-1.txt ,"Tweet text", OnThisDay

Dove il primo intero rappresenta l'indice di riga al quale il tweet è locato all'interno del file. csv, la dicitura "categoria-[index_integer].txt indica quanti tweet di quella categoria sono presenti all'interno del dataset, seguito dal contenuto del tweet, e dalla categoria d'appartenenza.

2.2 Preparazione dei dati per l'analisi e primi test

Al termine della costruzione manuale del dataset articolata nella sezione 2.1, si ottiene un dataset di dimensioni ridotte, 900 tweet, suddivisi secondo l'istogramma rappresentato in Figura 4, con una considerabile preponderanza di composizione per le classi *#OnThisDay* e *Promotions*:

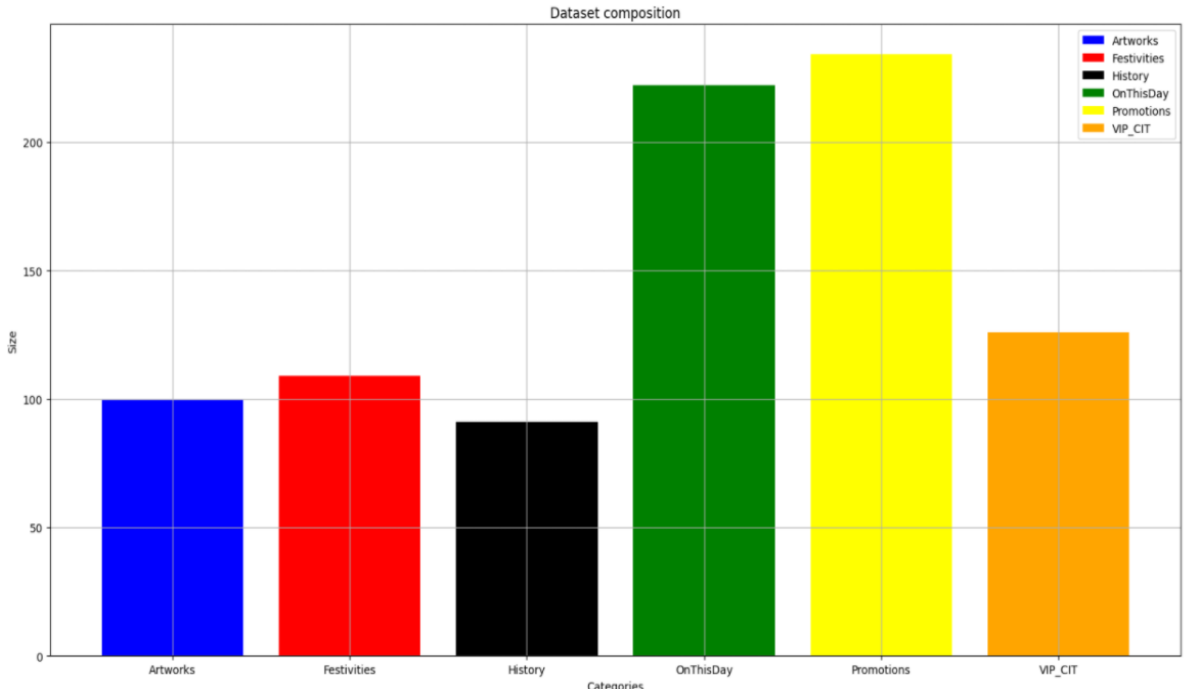

Figura 4 – Composizione del dataset iniziale

Questo primo dataset, contenuto su un file .csv, è stato il primo gradino per l'automazione della creazione di dataset più ampi, in quanto mediante un primo approccio al machine learning, si sono estratte le prime parole ed i primi pattern utili per la scrittura di un tool di costruzione automatica, che verrà dettagliato in seguito.

L’utilizzo di questi dati manualmente ricavati, ha richiesto in prima istanza un processo di raffinamento dei tweet, che ha previsto:

- *Rimozione caratteri speciali*: caratteri come "\n”, doppi apici, e più in generale qualsiasi carattere speciale, deve essere rimosso dal testo poiché non ci si aspetta alcun valore predittivo aggiunto mantenendoli, rischiando inoltre una distorsione del contenuto informativo in fase predittiva.
- *Upcase/downcase*: è necessario garantire che, ad esempio, “Book” e “book” siano la stessa parola e abbiano lo stesso potere predittivo.
- *Segni di punteggiatura*: caratteri come “?”, “!”, “;”, e qualsivoglia segno di punteggiatura, è stato rimosso.
- *Genitivo sassone*: è necessario garantire che “Marco” e “Marco’s” abbiano lo stesso potere predittivo.
- *Lemmatizzazione*: lo stemming è il processo di riduzione delle parole derivate alla loro radice. La lemmatizzazione è il processo di riduzione di una parola al suo lemma. La principale differenza tra entrambi i metodi è che la lemmatizzazione fornisce parole esistenti, mentre lo stemming fornisce la radice, che potrebbe non essere una parola esistente.
- *Stop word*: parole come “cosa” o “il” non avranno alcun potere predittivo poiché presumibilmente saranno comuni a tutti i documenti. Per questo motivo possono rappresentare un rumore eliminabile.

Al termine della procedura di raffinamento del contenuto informativo dei tweet, si può quindi procedere all’utilizzo del dataset ottenuto per un primo test di machine learning, volto ad individuare i pattern e le parole comuni che possono emergere, elementi necessari per automatizzare il processo di creazione di un dataset più ampio.

Mediante il primo training sul dataset di partenza:

```
#Training
X_train, X_test, y_train, y_test = train_test_split(df["F_CONTENT"],

                                                    df["F_CAT_CODE"],
                                                    test_size=0.15,
                                                    random_state=8
                                                   )
```

Figura 5 – Training dataset iniziale

Si ottengono le prime feature interessanti, salvate su quattro file pickle mediante:

```
#X_train
with open('expanded_ds_pickles/X_train.pickle', 'wb') as output:
    pickle.dump(X_train, output)

# X_test
with open('expanded_ds_pickles/X_test.pickle', 'wb') as output:
    pickle.dump(X_test, output)

# y_train
with open('expanded_ds_pickles/y_train.pickle', 'wb') as output:
    pickle.dump(y_train, output)

# y_test
with open('expanded_ds_pickles/y_test.pickle', 'wb') as output:
    pickle.dump(y_test, output)
```

Figura 6 – Salvataggio delle feature estratte dal primo dataset manualmente costruito.

Terminata la fase di estrazione delle prime feature, è possibile riutilizzare i dati ottenuti per l'automazione del processo di creazione del dataset, utilizzando una combinazione di

parole ottenute come appena descritto, e regole di aderenza rispetto alle entità, che verranno definite nella sezione 2.3.

Il secondo pilastro fondamentale per il processo di automazione è rendere più scalabili e generiche le parole ottenute mediante il primo run di machine learning, tale obiettivo è stato raggiunto mediante l'utilizzo della libreria spaCy, che permette l'utilizzo di tecniche di Natural Language Processing, ottenendo così un secondo pickle contenente le entità più frequenti per ogni categoria.

Al termine del processo di raccolta delle parole necessarie per automatizzare il processo di creazione del dataset, è stato quindi implementato il tool discusso nella sezione 2.3.

2.3 Automazione della creazione di un dataset

Al fine di snellire la procedura di raccolta dati, e al contempo aumentare la dimensione del dataset dal quale attingere per l'implementazione del modello predittivo, è stato implementato un tool che automatizzasse tale procedura. Al fine di poter categorizzare in modo deterministico i tweet proveniente dai file testuali, sono stati utilizzati due punteggi:

- Punteggio di aderenza terminologica: mediante le feature ottenute come descritto nella sezione 2.3.1, viene assegnato ad ogni word estratta un punteggio unitario di peso uno.
- Punteggio di aderenza rispetto alle entità: mediante le entità ottenute come descritto nella sezione 2.3.2, viene assegnato ad ogni entità estratta un punteggio unitario di peso uno.

2.3.1 Regole di aderenza terminologica

Al termine dell'estrazione delle parole come descritto nella sezione 2.2, si ottiene una lista di *keyword* per ogni categoria, contenente una entry per ogni categoria definita nella sezione 2.1, e ad ognuna di esse corrisponde la lista di parole chiave estratte per la medesima.

Mediante tale struttura dati, è possibile definire una funzione che riceva in input un generico tweet da inserire nel dataset, e si occupi controllare quante occorrenze di quella parola sono presenti all'interno del tweet.

Il punteggio ottenuto, può essere quindi "massimizzato", ovvero per ogni categoria viene calcolato il punteggio con la medesima metodologia, e viene mantenuto traccia del punteggio massimo ottenuto e della relativa categoria, ottenendo così un nuovo record per il dataset.

La struttura dati ottenuta mediante le regole grammaticali, sono salvate in una struttura dati del tutto simile a quella riportata di sotto in figura 7:

```
categories = {
                        "Artworks" :
["cup", "photo", "painter", "opera", "ceramic", "portrait", "paint"
,"japan", "august", "vase", "photo"],

                        "Festivities" :
["staff", "celebrate", "birthday","anniversary", "happy", "happy
birthday", "celebrating", "wish", "fun"],

                        "History" :
["history", "war", "past", "years", "century", "antinous", "food",
"artistinterventions", "historian tashamarks", "curator blog",
"historian"],

                        "OnThisDay" :
```

```
["today", "onthisday", "tonight", "born", "die" , "happen", "happens
today", "happened today"],

                        "Promotions" :
["tickets", "check", "check out", "week", "exhibition", "explore", "2021",
"2020", "open", "opens", "don't miss"],

                        "VIP_CIT" :
["artist", "said", "phrase", "citation", "famous", "series", "artquote",
"architect", "photographer", "find inspire"]
}
```

Figura 7 – Keywords regole di aderenza terminologica

La struttura dati in questione, può essere utilizzata rispetto alle regole di aderenza terminologica per ottenere un primo punteggio parziale, calcolando quante occorrenze di ogni parola contenuta nelle liste di parole di una categoria, compaiano nel tweet che è stato sottoposto a segmentazione in token. Al termine della valutazione del numero di occorrenze, si ottiene il punteggio parziale rispetto alle regole terminologiche definite nella presente sezione.

```
def hits(s="", lst=[]) -> int:
        total_hits = 0
        for token in lst:
                if token.lower() in s:
                        total_hits += 1

        return total_hits
```

Figura 8 – Funzione di scoring

Tale metodo, è più ampiamente integrato all'interno di un chiamante che si occupa di svolgere la funzione di dispatcher, previa tokenizzazione del tweet estratto dal file testuale.In questo modo, è possibile svolgere una "precategorizzazione" di ogni tweet e mantenere traccia del massimo punteggio ottenuto e della relativa categoria, così da poter inserire una nuova entry nel dataset. Il punteggio ottenuto per le regole di aderenza

terminologica è normalizzato rispetto alla lunghezza delle *keyword*, per evitare una sovrastima dei punteggi per una disparità di lunghezza delle liste.

```
other = 0
NLP = spacy.load("en_core_web_sm")
df = pd.DataFrame(columns = ["F_NAME", "F_CONTENT", "CATEGORY"])
with open("raw_ds_path.txt", "r", encoding="utf8") as f:
        lines = f.readlines()
        for i, line in enumerate(lines):
                content = line.strip()
                max_cat_score = 0.0
                max_score_cat_label = ""
                for category, tokens in categories.items():
                        curr_score = hits(content, tokens) / len(categories[category]) #normalizzazione
                        content_ent = NLP(content)
                        entities_tweet = [entity.label_ for entity in content_ent.ents]
                        curr_score_ent = hits_ent(entities_tweet, categories_ents[category]) / len(categories_ents[category])
                        curr_score += curr_score_ent
                        if curr_score > max_cat_score:
                                max_score_cat_label = category
                                max_cat_score = curr_score
                if max_cat_score > 0.0:
                        cat_hits[max_score_cat_label] += 1
                        df.loc[len(df)] = [f"{max_score_cat_label}-{cat_hits[max_score_cat_label]}.txt", content, max_score_cat_label]
                        max_cat_score = 0
                        content_ent = NLP(content)
                        for entity in content_ent.ents:
                                categories_ents[max_score_cat_label].add(entity.label_)
                else:
                        df.loc[len(df)] = [f"Other-{other}.txt", content, "Other"]
                        other += 1
```

Figura 9 – Integrazione della funzione di scoring

Sia tweet il contenuto informativo da inserire come nuovo record del dataset, e sia categories[i] l'i-esima categoria appartenente al set delle categorie, sia tokenize una funzione di che spezzi un tweet in token univoci, e sia hits la funzione descritta in figura 8, sia D l'insieme delle parole chiave estratte per ogni categoria come in Figura 7, allora il punteggio ottenuto per il tweet verrà ottenuto mediante:

$$hits(token) = \begin{cases} 1 \ se\ token \ \in D \\ 0\ se\ token \ \notin D \end{cases}$$

$$term\_\text{score} = \frac{MAX(\sum_{i=1}^{n} hits(tokenize(tweet), categories[i]))}{\text{n}}$$

2.3.2 Regole di aderenza rispetto alle entità

Dualmente a quanto definito per le regole di aderenza terminologica, mediante la libreria spaCy è stato possibile ottenere per ogni categoria, una lista di *entità*, che rendesse più scalabile e flessibile le regole di aderenza terminologica, in modo da astrarre dalla specifica parola, ma collocare un eventuale scoring non nullo, rispetto a generiche entità.

```
categories_ents = {
                'Artworks':     ['WORK_OF_ART', 'TIME', 'QUANTITY',
'CARDINAL', 'PRODUCT','PERCENT', 'EVENT', 'LOC', 'GPE'],

                'Festivities':  ['TIME', 'QUANTITY', 'PRODUCT', 'MONEY',
'PERCENT', 'DATE', 'EVENT', 'GPE'],

                'History':      ['WORK_OF_ART', 'TIME', 'CARDINAL',
'PRODUCT', 'LAW', 'DATE', 'EVENT', 'LOC', 'GPE'],

                'OnThisDay':    ['TIME', 'LOC', 'PRODUCT', 'MONEY',
'PERCENT', 'DATE', 'EVENT', 'QUANTITY', 'GPE'],

                'Promotions':
['TIME', 'PRODUCT', 'LOC', 'MONEY', 'DATE', 'EVENT', 'QUANTITY', 'GPE',
'WORK_OF_ART'],

                'VIP_CIT':      ['TIME', 'QUANTITY', 'LAW', 'CARDINAL',
'PRODUCT', 'PERCENT', 'DATE', 'LOC', 'GPE']
 }
```

Figura 10 – Entità per le regole di aderenza terminologica

La funzione di calcolo di occorrenze di entità all'interno di un tweet ricalca la funzione rappresentata in Figura 5, in quanto sostanzialmente il problema algoritmico alla base è il medesimo: ottenuta un tweet tokenizzato in sottostringhe, si rende necessario contare quante di esse siano appartenenti alla lista di entità della categoria in corso di valutazione.

```
def hits_ent(ent=[], lst=[]) -> int:
        total_ent_hits = 0
        for e in ent:
                if e in lst:
                        total_ent_hits += 1
        return total_ent_hits
```

Figura 11 – Funzione di scoring rispetto alle entità

Dualmente a quanto definito nella sezione 2.3.1, all'interno del codice rappresentato in Figura 6, viene integrato un meccanismo di calcolo del punteggio rispetto alle entità, elemento complementare al punteggio rispetto all'aderenza terminologica.

La funzione di riferimento è hits_ent, come descritta in Figura 8, che analogamente alla funzione hints rappresentata in Figura 5, riceve in input una lista di entità riferite ad una categoria, ed un tweet sottoposto a processo di segmentazione in token.

Formalmente, si può descrivere il punteggio ottenuto rispetto alle entità come segue nel prossimo paragrafo.

Sia tweet il contenuto informativo da inserire come nuovo record del dataset, e sia categories[i] l'i-esima categoria appartenente al set delle categorie, sia tokenize una funzione di che spezzi un tweet in token univoci, e sia hits_ent la funzione descritta in figura 11, sia D l'insieme delle entità estratte per ogni categoria come in Figura 10, allora il punteggio ottenuto per il tweet verrà ottenuto mediante:

$$hits\_ent(token) = \begin{cases} 1 \ se\ token\ \in D \\ 0\ se\ token\ \notin D \end{cases}$$

$$ent\_\text{score} = \frac{MAX(\sum_{i=1}^{n} hits\_ent(tokenize(tweet), categories[i]))}{\text{n}}$$

2.3.3 Calcolo del punteggio finale

Il punteggio finale viene calcolato nel seguente modo

$$PUNTEGGIO\ TOTALE = PUNTEGGIO\ ENTITÀ + PUNTEGGIO\ TERMINOLOGICO$$

Per esemplificare e fornire un'istanza concreta di calcolo di un punteggio, si supponga di voler inserire automaticamente nel dataset il tweet: "*Check our tickets promotion. Exhibition starts soon!*"

Tale tweet verrà segmentato in questo modo (tutti i token sono in lowercase):

1. check
2. our
3. tickets
4. promotions
5. exhibition
6. starts
7. soon

A seguito della suddivisione in token, e supponendo come categoria di riferimento *Promotions*, i punteggi ottenuti, terminologici e rispetto alle entità, sarebbero i seguenti:

Punteggio terminologico: 5 (le parole: check, tickets, exhibition, starts, soon),

Punteggio entità: 3 (le entità: TIME, PRODUCT, EVENT)

Normalizzazione del punteggio terminologico rispetto al totale di parole della categoria di riferimento: 5 / 11 = 0.45

Normalizzazione del punteggio entità rispetto al totale di entità della categoria di riferimento: 3 / 9 = 0.34.

Il punteggio totale (per la categoria *Promotions*) = 0.45 + 0.34 = 0.79.

Iterando per ogni categoria, e tenendo traccia del massimo e della relativa categoria, si otterrà una nuova entry per il dataset.

2.3.4 Composizione del Dataset al termine del processo di automazione

Il processo di automazione di creazione del dataset è stato effettuato usando come dati iniziali, i tweet provenienti dai gruppi G1, G2 e G3, e filtrando tutti i tweet che non fossero in lingua inglese.

Al termine del processo di automazione la percentuale di tweet correttamente inseriti ed etichettati con una categoria è riportata di sotto in tabella in Figura 9, come si può apprezzare, la componente di tweet senza alcuna classificazione si attesta al 17%. La dimensione totale del dataset è di 5637 tweet totali, con un picco di composizione nelle categorie *Promotions* e *Artworks*.

Tabella riepilogativa al termine del processo di automazione:

Dimensione iniziale del dataset contenente i soli tweet	6791
Totale tweet categorizzati secondo le regole determinate	5637
Totale tweet non categorizzati secondo le regole determinate	1154

Percentuale tweet categorizzati	83%
Percentuale tweet non categorizzati	17%

Figura 12 – Tweet correttamente categorizzati

Composizione del dataset, suddiviso per classi, al termine del processo di automazione:

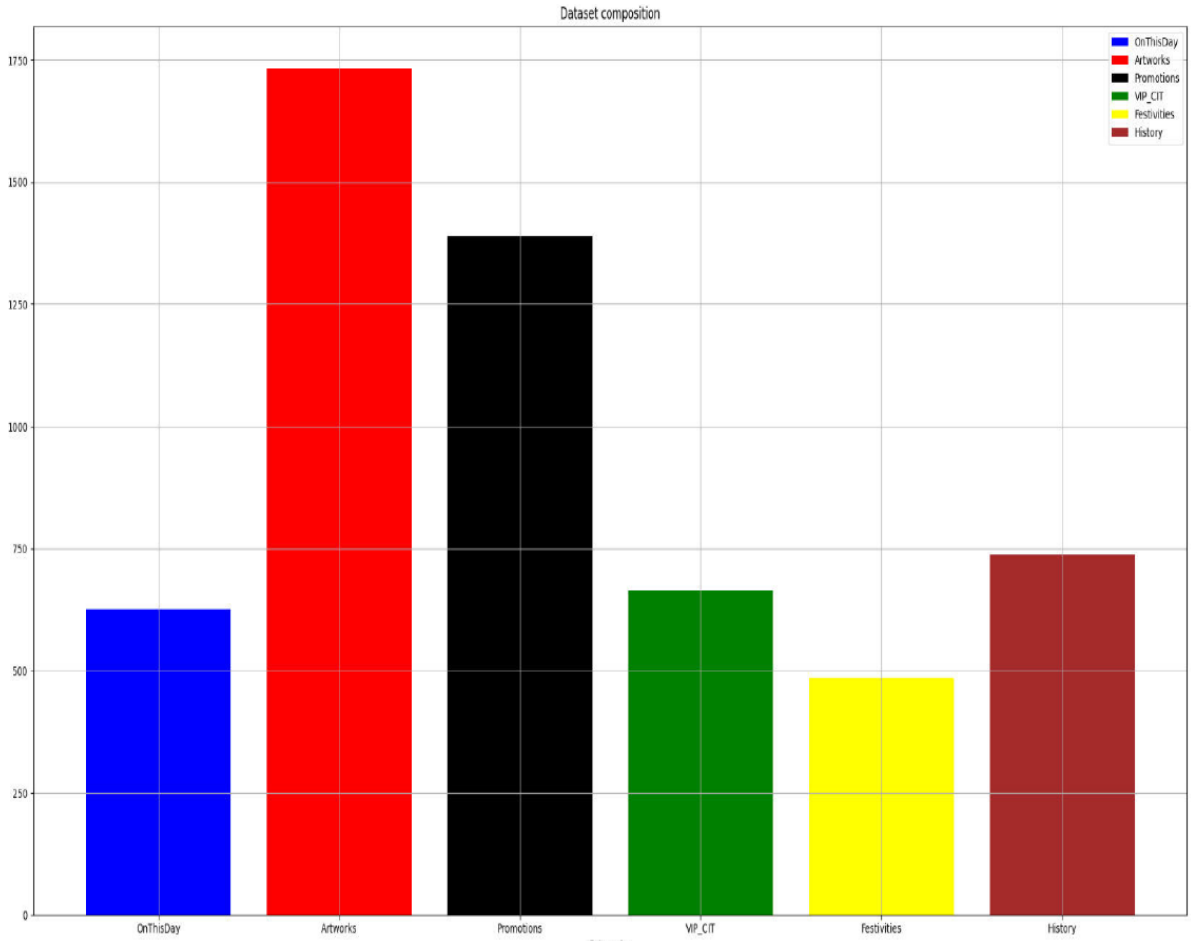

Figura 13 – Composizione del dataset al termine dell'automazione del processo

Capitolo 3

Risultati ottenuti

3.1 Test e fine tuning dei classificatori

Al termine del processo di automazione di creazione di un dataset, si è reso necessario poter verificare come i dati selezionati rispondessero a diversi modelli, al fine di valutarne l'effettiva accuratezza, passaggio fondamentale prima di procedere alle successive analisi sui dati.

Sono stati testati i seguenti i seguenti classificatori:

- RandomForestClassifier
- KNN Classifier
- MultinomialNaiveBayes Classifier
- LogisticRegression Classifier
- XGBClassifier (XGBoost)

Per i modelli sopra elencati, è stata effettuata un'indagine rispetto all'accuratezza ottenuta, con i risultati ottenuti nella tabella sottostante.

Classifier	Accuracy
RandomForestClassifier	83%
KNN Classifier	58%
LogisticRegression Classifier	88%
XGBoost	86%
MultinomialNaiveBayes Classifier	68%

Come è possibile apprezzare dai risultati appena riportati, i classificatori *LogisticRegression* e *XGBClassifier* sono i meglio performanti, con una percentuale di accuratezza maggiore.

I classificatori *KNN* e *MultinomialNaiveBayes* non hanno riportato un'accuratezza particolarmente alta, e per questo motivo non sono stati presi in considerazione come candidati all'utilizzo nelle analisi successive.

Nonostante l'accuratezza di *LogisticRegression* risultasse di poco superiore all'accuratezza riportata per *XGBClassifier*, è stato scelto questo ultimo classificatore come riferimento del lavoro da questo punto in avanti, in quanto esso mette a disposizione svariati metodi per la personalizzazione delle feature, elemento che sarà necessario come descritto in sezione 3.3, in quanto l'intero obiettivo del presente lavoro è l'utilizzo delle feature estratte come elemento migliorativo in termini di accuratezza predittiva.

3.2 Analisi della composizione per classi dei dati riferiti ai gruppi G1, G2, G3

Al termine del processo di automazione di creazione di un Dataset come descritto in sezione 2.3, si ottiene una struttura contenente 5637 tweet provenienti dai musei dei gruppi G1, G2 e G3. Come riportato nella tabella rappresentata in figura 2, il totale di tweet per i gruppi sopra citati è di 22.050 tweet. A fronte di un dataset rappresentante il 25.5% dei tweet totali, risulta interessante analizzare di quali classi di tweet definite in sezione 2.1, siano composti i dati raccolti dai gruppi da G1 a G3.

Per questa ragione, è stato effettuato un training con le feature precedentemente testate ed estratte come da figura 6, in tale modo:

```
 cv = CountVectorizer(max_features=5000, encoding="utf-8",
      ngram_range = (1,3),
      token_pattern = "[A-Za-z_][A-Za-z\d_]*")

    X_train, X_test, y_train, y_test =
train_test_split(cv.fit_transform(df["F_CONTENT"]).toarray(),
df['F_CAT_CODE'],
      test_size=0.33,
      random_state=15)

    model =xgb.XGBClassifier()
    model.fit(X_train, y_train)
```

Figura 14 – Fitting del modello XGBClassifier

La dimensione del test set è del 33% rispetto al dataset creato, il seme pseudo-randomico di inizializzazione, è 15.

Una volta terminato il training del test set, viene effettuato il *fitting* per le label di training, ottenendo così quindi un classificatore pronto all'utilizzo e la predizione delle classi di appartenenza dei tweet dei gruppi precedentemente elencati.

```
 for group in groups:
        df1 = pd.read_csv(base_path_raw+'threads-'+group+'-
ng.txt',sep='\t')
        df2 = pd.read_csv(base_path_raw+'threads-'+group+'-
2020ng.txt',sep='\t')
        df = pd.concat([df1,df2])

        #Special characted cleaning
        df["FULL_TEXT"] = df["FULL_TEXT"].str.replace("'s", "")
        df["FULL_TEXT"] = df["FULL_TEXT"].str.replace("’s", "")
        df["FULL_TEXT"] = df["FULL_TEXT"].apply(lambda x:
re.sub(r'http\S+',"", str(x)))
```

```
        df["FULL_TEXT"] =
df["FULL_TEXT"].str.strip().str.lower().str.replace('"','')
```
Figura 15 – Rimozione del genitivo sassone e link esterni

I dati vengono sottoposti, come specificato in sezione 2.2 alle seguenti procedure di filtraggio e rimozione delle caratteristiche non rilevati ai fini della predizione, in particolare in figura 15, possiamo apprezzare i seguenti accorgimenti:

- Rimozione del genitivo sassone
- Rimozione dei link esterni
- Rimozione di apici

Continuando, si rende necessario proseguire con la lemmatizzazione di tutti i verbi contenuti nei tweet che verranno sottoposti a predizione, trasformando così quindi un verbo nella sua forma base.

Tale obiettivo, è raggiunto nel codice mediante quanto rappresentato in figura 16:

```
 nltk.download('punkt')
        nltk.download('wordnet')

        #Lemmatization
        wordnet_lemmatizer = WordNetLemmatizer()
        lemmatized_text_list = []

        for i in range(len(df)):
            lemmatized_list = []
            text = str(df.iloc[i]["FULL_TEXT"])[1:].strip()
            text_words = text.split(" ")

            for word in text_words:
                lemmatized_list.append(wordnet_lemmatizer.lemmatize(word,
pos="v"))

            lemmatized_text_list.append(" ".join(lemmatized_list))

        df["FULL_TEXT"] = lemmatized_text_list
```
Figura 16 – Lemmatizzazione del contenuto del tweet.

Come ultimo passaggio di filtraggio rispetto al contenuto del tweet, abbiamo la rimozione delle stopwords, così definite:

"Le stopwords è una qualsiasi parola contenuta in un dizionario negativo che viene filtrata prima o dopo l'elaborazione dei dati del linguaggio naturale.

Non esiste un unico elenco universale di stopwords utilizzato da tutti gli strumenti di elaborazione del linguaggio naturale, né regole concordate per identificare le parole non significative, e in effetti non tutti gli strumenti utilizzano tale elenco.

Pertanto, qualsiasi gruppo di parole può essere scelto come stop word per un determinato scopo. La tendenza generale nel tempo è variata dall'uso standard di elenchi di fermate piuttosto grandi, circa 200–300 termini, all'utilizzo di elenchi di fermate molto piccoli, dai 7 ai 12 termini, fino a nessun elenco di stopwords." [5]

Tale obiettivo nel codice, è raggiunto mediante quanto rappresentato in figura 17, come sotto raffigurato:

```
#Stopwords
        nltk.download('stopwords')
        stop_words = list(stopwords.words('english'))
        for stop_word in stop_words:
            regex_stopword = r"\b" + stop_word + r"\b"
            df["FULL_TEXT"] = df["FULL_TEXT"].str.replace(regex_stopword,
'')
```

Figura 17 – Rimozione delle stopwords

Al termine della fase di preparazione dei dati, otteniamo l'effettiva predizione mediante il seguente codice:

```
preds = model.predict(cv.fit_transform(df["FULL_TEXT"]).toarray())
```

Figura 18 – Predizione dati per un generico gruppo Gn

Per il gruppi indicati in sezione 1.4, si riportano nelle figure 19-20-21 gli istogrammi per la relativa analisi di composizione rispetto alle classi:

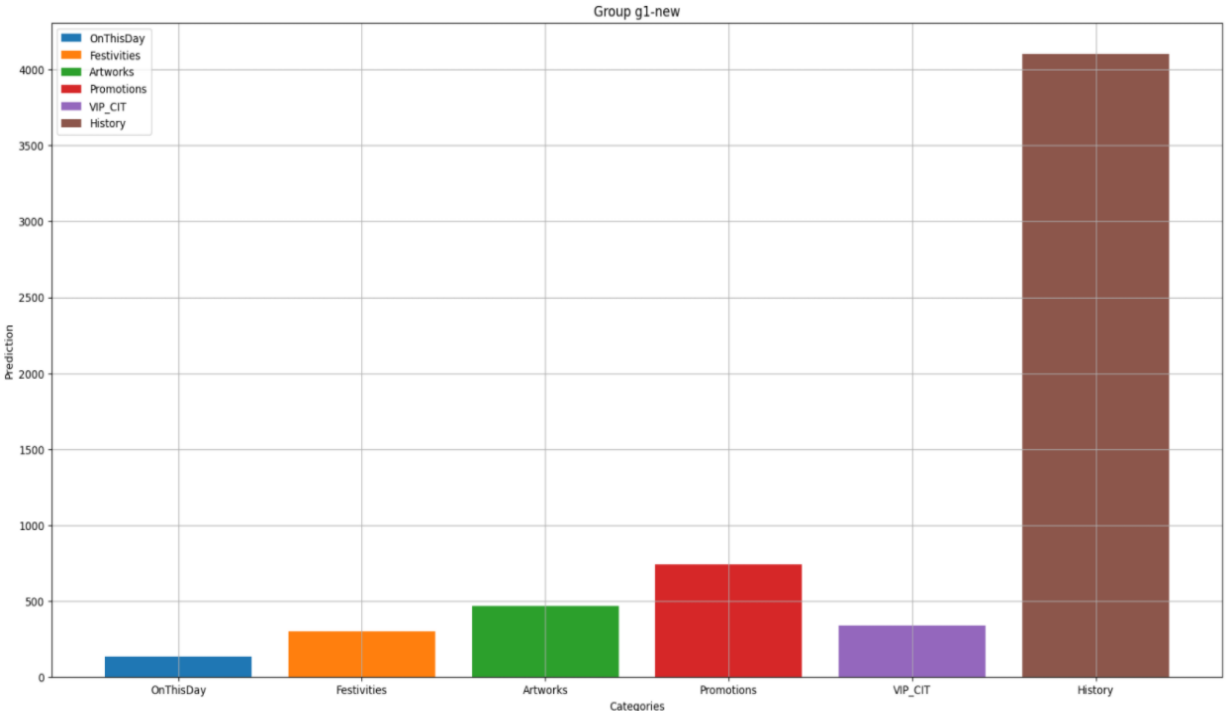

Figura 19 – Composizione dei tweet postati dal gruppo di musei G1 per ogni classe

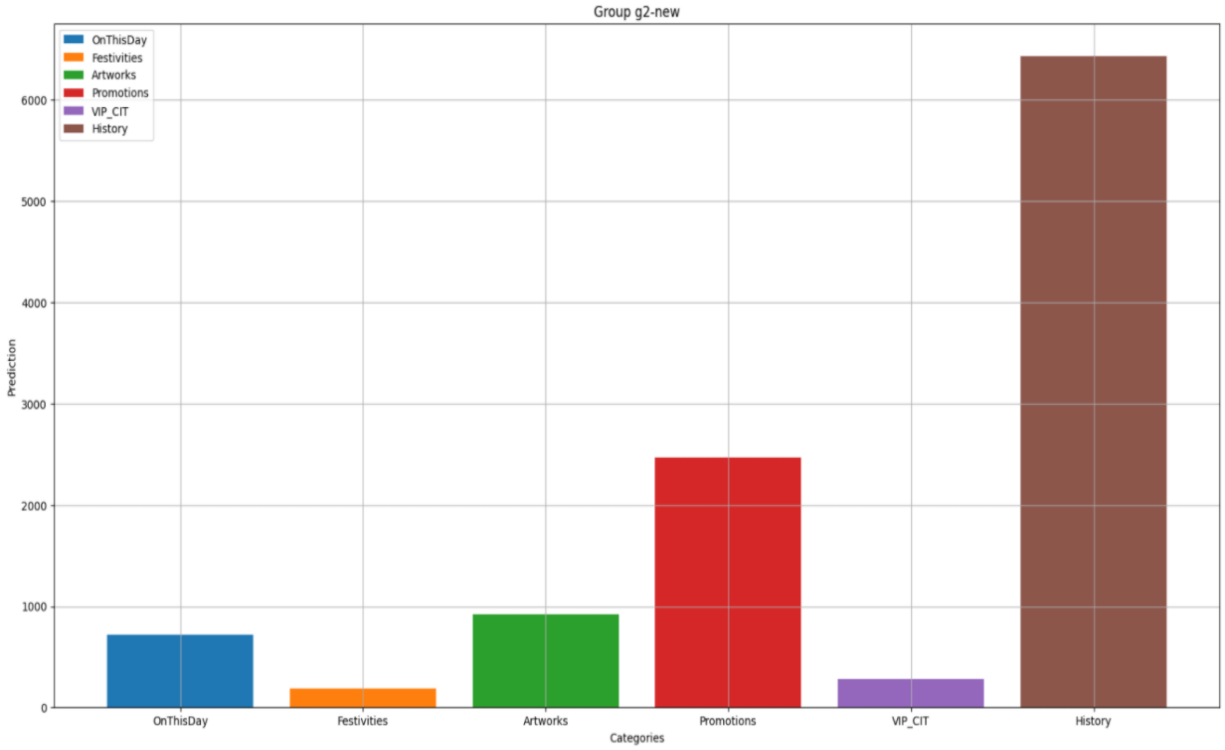

Figura 20 - Composizione dei tweet postati dal gruppo di musei G2 per ogni classe

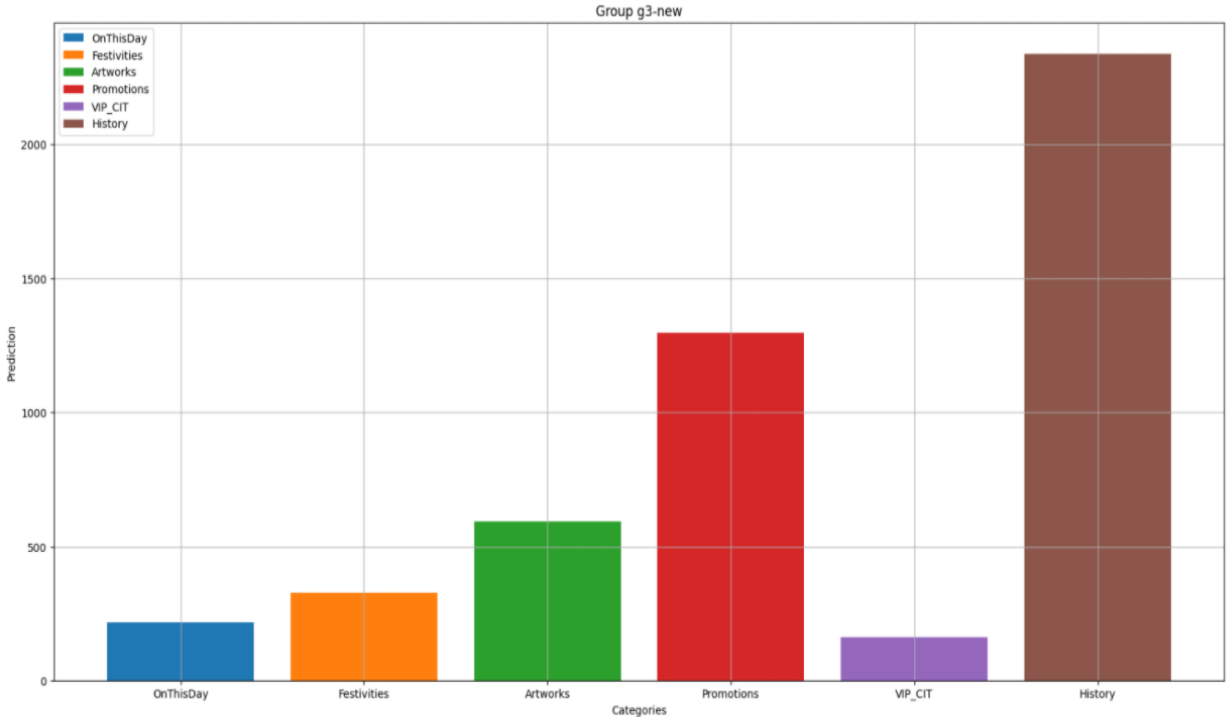

Figura 21 - Composizione dei tweet postati dal gruppo di musei G3 per ogni classe

3.2.1 Interpretazione dei dati ottenuti

Come è possibile apprezzare, per ogni gruppo vi è una netta preponderanza dei tweet categorizzati come appartenenti alla categoria *History*, come ampiamente prevedibile in quanto il topic più trattato è sicuramente quello di argomenti a tema storico.
Cionondimeno, risulta interessante notare come l'andamento dei tweet promozionali in correlazione al numero di followers che il museo posside sul proprio profilo social Twitter.

La tendenza di tweet promozionali è in forte aumento tra i musei del gruppo G1, ed i musei del gruppo G2, è presente una decrescita della tendenza tra il gruppo G2 ed il gruppo G3.

Questa tendenza generale (comunque in crescita tra G1 e G3) potrebbe essere potenzialmente spiegabile dal fatto che musei di portata maggiore, e con una fama già nota, abbiano meno interesse nel proporre ai propri seguaci contenuti promozionali,

concentrandosi di più su contenuti di carattere storico e/o relativo ad opere d'arte reperibili all'interno dell'ente stesso.

Al contrario, musei di portata minore, potrebbero avere interesse nel promuovere maggiormente contenuti promozionali, al fine di stimolare maggiormente le visite all'ente e promuovere quindi i contenuti da esso offerti e/o durante gli eventi e le mostre.

3.3 Utilizzo delle classi come input del modello

Come definito in sezione 2.1, ricordiamo che le classi individuate come feature di input per il modello predittivo, sono le seguenti:

- Artworks
- Festivities (inteso come messaggi di auguri per feste e ricorrenze)
- History (celebrazione di eventi storici, o spiegazione di eventi storici correlati ad opere d'arte esposte)
- #OnThisDay
- Promotions
- VIP & CIT (Celebrazione di personaggi storici, frasi a loro riconducibili, motti storici)

Utilizzando il modello addestrato come menzionato in Figura 14, comprendente del testo dei tweet provenienti dai musei dei gruppi G1, G2 e G3, sui tweet categorizzati per le classi sopra menzionate, mediante una predizione l'inizializzazione di *CountVectorizer*, per ottenere una trasformazione di dati in formato testuale, in un vettore sulla base della frequenza di ogni parola che compare nell'intero testo.

```
cv = CountVectorizer(max_features=5000, encoding="utf-8",
     ngram_range = (1,3),
```

```
    token_pattern = "[A-Za-z_][A-Za-z\d_]*")
```

Figura 23 – Inizializzazione CountVectorizer

La predizione effettiva avviene mediante tale istruzione, avente come target il testo di ogni tweet:

```
preds = model.predict(cv.fit_transform(df["FULL_TEXT"]).toarray())
```

Figura 24 – Predizione

Avendo inserito un dato non considerato inizialmente, risulta interessante verificare quale sia l'impatto della feature introdotta rispetto alla situazione iniziale, ovvero verificare se l'accuratezza del modello sia influenzata positivamente da tale feature, ricorando che il dato di interesse è reperibile in sezione 1.4.2, in figura 3, osserviamo che:

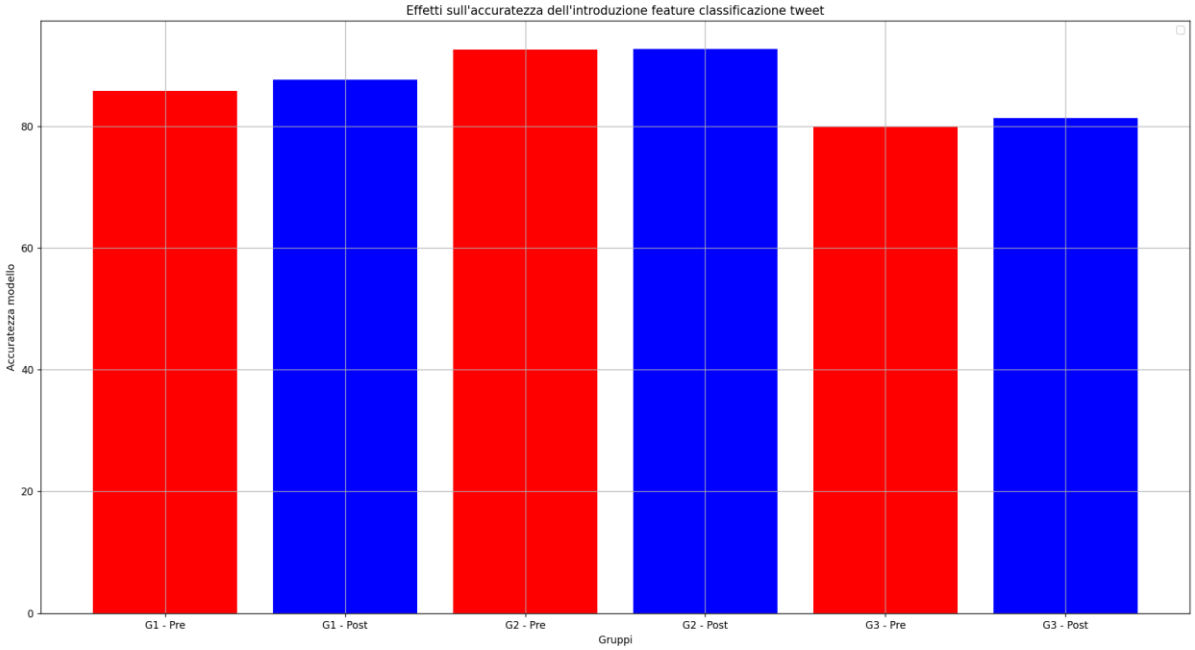

Figura 25 – Miglioramento dell'efficacia predittiva

Effettivamente, data l'introduzione delle classi di tweet e del testo come feature al classificatore, essa produca un piccolo miglioramento nell' efficacia predittiva dello stesso, come dimostrato in Figura 25.

Per ogni gruppo, si riporta nella sottostante tabella l'aumento di efficacia predittiva in termini percentuali:

Gruppo	**Accuratezza % Pre-Feature**	**Accuratezza % Post-Feature**	**Incremento efficacia predittiva in %**
G1	85.78	87.63	2.16
G2	92.60	92.65	0.05
G3	79.90	81.29	1.74

Come riportato nella tabella soprastante, è quindi possibile affermare che l'introduzione della nuova feature rispetto alla situazione illustrata in sezione 1.4.2, conduca ad un aumento dell'efficacia predittiva media (calcolata sui gruppi G1, G2 e G3), del 1.316%.

Capitolo 4

Conclusioni

4.1 Considerazioni finali

Attraverso questo studio, è stato possibile approfondire come e con quali mezzi sia possibile implementare un modello predittivo per il miglioramento dell'efficacia comunicativa su piattaforme social media, con focus in ambito cultural heritage.

La principale differenza rispetto al lavoro inziale, è rappresentato dall' introduzione e dell'utilizzo di classi di tweet come feature di classificazione. Esse rappresentano un punto di inizio per la potenziale stesura di un applicativo che possa aiutare un social media manager a scrivere un tweet che venga statisticamente apprezzato dai followers, migliorando così di fatto l'efficacia comunicativa.

4.2 Il processo di automazione è utile?

Il processo di automazione è fondamentale per l'eventuale implementazione di un applicativo che si basi sul presente modello predittivo per il miglioramento dell'efficacia comunicativa. Potendo contare su dati di facile reperibilità previa richiesta all'ente competente, avere a disposizione un tool che permetta di automatizzare lo raccolta dei dati stessi, e la successiva predisposizione degli stessi in un dataset pronto per essere importato ed utilizzato, facilita lo svolgimento delle analisi e l'interpretazione dei dati, in quanto permette la costruzione di un dataset incrementale, in cui le precedenti run di machine learning vengono riutilizzate per la costruzione cumulativa, svolgendo così una duplice funzione: una funzione di output, in quanto producente un'analisi dei dati, ed in secondo luogo una funzione di input, in quanto le feature estratte possono essere ricorsivamente utilizzare poter migliorare lo strumento di costruzione automatica del dataset.

4.3 Quanto sono incisive le feature introdotte?

Le feature introdotte apportano un piccolo miglioramento a livello di efficacia predittiva del modello preesistente. Mediante l'utilizzo delle classi definite, è possibile ottenere una nuova feature attivamente utilizzata dal classificatore XGBClassifier al fine di affinare la predizione di quale sia un tweet positivo, o quale sia un tweet negativo.

Tramite l'introduzione di queste feature, è lievemente accresciuto anche il potenziale di un eventuale applicativo basato sul modello predittivo, in quanto il rilevamento automatico della classe del tweet di bozza digitato dal social media manager, può essere un primo indicatore sulla categorizzazione positiva o negativa di un tweet.

L'utilizzo delle classi, oltre ad essere feature intese in senso tecnico, possono intendersi utili anche per verificare su un eventuale applicativo il tipo di tweet apprezzati dai followers con una semplice indagine statistica, meno complicata di un modello predittivo: con un semplice conteggio di quanti tweet categorizzati come positivi di una fissata categoria, e calcolando in termini percentuali a quanto questo conteggio corrisponda, si può quindi anche verificare quali *categorie* di tweet siano più apprezzate direttamente dai followers, senza utilizzarle come strumento indiretto di input del modello predittivo, ma come strumento esplicito.

BIBLIOGRAFIA E SITOGRAFIA

[1] Nvidia: https://www.nvidia.com/en-us/glossary/data-science/xgboost/

[2] Wikipedia: https://en.wikipedia.org/wiki/Named-entity_recognition

[3] Scikit-learn.org: https://scikit-learn.org/stable/about.html#history

[4] Marco Furini, Federica Mandreoli, Riccardo Martoglia, and Manuela Montangero. 2021. A Predictive Method to Improve the Effectiveness of Twitter Communication in a Cultural Heritage Scenario, ACM Journal on Computing and Cultural Heritage (JOCCH), 2022.

[5] Wikipedia: https://en.wikipedia.org/wiki/Stop_word