UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea in Informatica

TESI DI LAUREA

Classificazione delle domande sulla sicurezza dei post di Stack Overflow

RELATORE

Prof. Fabio Palomba

Dott. Emanuele Iannone

Università degli Studi di Salerno

CANDIDATO

Giuseppe Grano

Matricola: 0512110454

Anno Accademico 2022-2023

Questa tesi è stata realizzata nel

"Roads? Where we're going, we don't need roads."

Back to the Future Part II

Abstract

La crescente problematica delle minacce informatiche, richiede nuove metodologie di prevenzione degli attacchi, ma anche una maggiore analisi del tema sicurezza dal punto di vista dei programmatori. Il numero consistente di dati, su cui svolgere le analisi, è stato un punto a favore per l'introduzione dell'intelligenza artificiale in questo contesto. Molti dei dati per svolgere le analisi si possono trovare sulla piattaforma Stack Overflow o su piattaforme simili. L'obiettivo di questa ricerca è stato, partendo proprio dai dati presenti su Stack Overflow, quello di realizzare un classificatore di testi multi classe che opera su domande relative all'ambito della sicurezza informatica. Il metodo che è stato seguito consiste nel pulire i dati di input, convertirli in dati numerici e poi darli in pasto ad un modello di classificazione. A questo punto è stato necessario valutare le prestazioni dei modelli considerati per scegliere il migliore. In questa ricerca sono stati, infatti, affrontati il problema relativo alla scelta del modello più adatto e il problema del bilanciamento dei dati. I risultati presi in analisi sono stati quelli relativi al micro F1 score. Il migliore valore ottenuto è pari a 0.78. Questo risultato è stato ottenuto con il modello Multinominal Naive Bayes, che si adatta molto bene ai task di classificazione del testo. A seguito dell'analisi sui modelli, è stato applicato il bilanciamento dei dati tramite la tecnica SMOTE, facendo aumentare lo score da 0.78 a 0.80. Questa ricerca ha permesso di aggiungere alcune considerazioni sui modelli Multinominal Naive Bayes, LinearSVC e Decision Tree, utilizzati nella classificazione dei testi. Inoltre, è stato mostrato che il modello Multinominal Naive Bayes risulta essere il più adatto in questi contesti e che la tecnica di oversampling, per il bilanciamento dei dati, è da preferire rispetto l'undersampling.

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

1.1 Contesto, motivazioni e obiettivi

Ogni giorno vengono scoperte vulnerabilità nei sistemi che utilizziamo, ed in base alla loro gravità queste rappresentano più o meno un rischio per gli utenti. Di conseguenza, la sicurezza informatica è diventata una delle preoccupazioni principali di organizzazioni, di grandi e piccole imprese e anche dei singoli utenti. Il mondo della sicurezza informatica è vasto e complesso. Si divide in diverse branche che vanno dalla crittografia, alla sicurezza web ed ancora alla sicurezza in dispositivi IoT (Internet of Things). Data la complessità di questo tema, molti programmatori utilizzano piattaforme come Stack Overflow[1] per porre domande sia teoriche che di implementazione e domande sulla risoluzione di problemi comuni.

I dati presenti nei post di Stack Overflow sono un punto di partenza per effettuare delle analisi sul tema sicurezza. Le analisi di questi dati permettono di conoscere il rapporto che c'è tra gli sviluppatori e la sicurezza dei sistemi informatici. Queste, però, richiedono molto lavoro e pertanto si stanno diffondendo sempre di più soluzioni che fanno uso di Intelligenza Artificiale, automatizzando l'analisi dei dati. L'Intelligenza Artificiale è un tema che negli ultimi anni sta suscitando molto interesse, sia da

[1] `https://stackoverflow.com/`

parte degli utenti sia da parte degli sviluppatori che sempre più spesso propongono soluzioni basate su algoritmi capaci di apprendere. Uno dei principali campi di applicazione dell'Intelligenza Artificiale, per l'analisi teorica del tema sicurezza, è la classificazione dei testi.

In letteratura esistono già lavori di ricerca che analizzano i post presenti su Stack Overflow e su piattaforme simili. Tra questi troviamo *A large-scale study of security vulnerability support on developer q&a websites* [1] e *Secure coding practices in java: Challenges and vulnerabilities* [2].

Il primo lavoro di ricerca citato fornisce una analisi sui principali topic dei post di Stack Overflow relativi alla sicurezza. Inoltre vengono individuati i topic più complessi e quelli che attraggono maggiore attenzione. Il secondo lavoro citato, invece, restringe il focus sulla sicurezza nel mondo Java[2], fornendo una analisi che mostra quali sono le vulnerabilità principali e le maggiori preoccupazioni dei programmatori.

Questi lavori analizzano i dati, cercando di estrarre informazioni utili sul tema sicurezza. Nella nostra ricerca, invece, l'attenzione si sposta più sugli strumenti utilizzati per condurre queste analisi. Si cerca di capire quali sono gli approcci migliori di classificazione, viene data importanza alla pulizia del testo grezzo e si considera il problema dei dati sbilanciati. Tutti i risultati analizzati fanno riferimento alle performance di diversi classificatori, per capire quali sono gli approcci più adatti in questo contesto.

L'obiettivo di questo studio è, quindi, quello di implementare un classificatore di testi, capace di identificare la classe di appartenenza delle domande di sicurezza presenti su Stack Overflow. Le principali problematiche considerate durante la ricerca sono state la scelta del modello e l'influenza che il bilanciamento dei dati ha sulle prestazioni.

[2] `https://www.java.com/it/`

1.2 Struttura della tesi

La tesi è strutturata in 5 capitoli:

- **Capitolo 2 Background e Stato dell'arte**: In questo capitolo è presentata una panoramica generale sulla classificazione dei testi e vengono citati alcuni lavori di ricerca già presenti in letteratura.
- **Capitolo 3 Sperimentazione dei Classificatori di Domande**: In questo capitolo vengono presentate le domande di ricerca ed il metodo che è stato seguito in questo studio.
- **Capitolo 4 Analisi dei Risultati**: In questo capitolo sono presentati i risultati ottenuti a seguito dell'applicazione della metodologia scelta.
- **Capitolo 5 Conclusioni**: In questo capitolo vengono descritti i risultati.

CAPITOLO 2

Background e Stato dell'arte

2.1 Background

L'aumento della consapevolezza e dell'importanza del tema vulnerabilità del software, ha spinto i ricercatori a condurre studi per analizzare i dati presenti in rete e comprendere il livello attuale di sicurezza nei sistemi. Le informazioni sulla sicurezza possono essere raccolte da fonti controllate da esperti come OWASP (Open Web Application Security Project) [1] e CWE (Common Weakness Enumeration) [2].

OWASP e CWE sono dei progetti che forniscono un elenco di vulnerabilità comuni, un catalogo di errori, difetti e debolezze conosciute che possono essere sfruttate da malintenzionati. Entrambi gli strumenti vengono utilizzati, da parte di esperti della sicurezza, come punto di riferimento per identificare e correggere possibili vulnerabilità durante lo sviluppo del software. Allo stesso tempo, questi strumenti non forniscono un metodo per far comunicare gli sviluppatori, i quali potrebbero avere dei problemi nella comprensione delle soluzioni proposte o problemi che possono riscontrare nei loro sistemi.

[1] https://owasp.org/
[2] https://cwe.mitre.org/

Per ottenere informazioni relative al punto di vista degli sviluppatori possono essere utilizzate piattaforme di question and answer (Q&A). Le piattaforme di Q&A principali per la tecnologia e la sicurezza sono rispettivamente Stack Overflow e Security Stack Exchange[3]. Stack Overlow è una delle piattaforme più popolari, che offre uno spazio per scrivere domande e relative risposte. Si focalizza principalmente sul tema della programmazione. Fornisce informazioni sotto forma di documentazione, esempi di codice e discussioni sulla programmazione. Security Stack Exchange è una piattaforma simile a Stack Overlow ma orientata al tema della sicurezza. Nella Figura 2.1 è presente uno screenshot dell'interfaccia di Stack Overflow relativa ad una domanda, mentre nella Figura 2.2 è mostrato lo screenshot della relativa risposta.

Figura 2.1: Interfaccia Stack Overflow relativa ad una domanda sulla piattaforma.

[3] https://security.stackexchange.com/

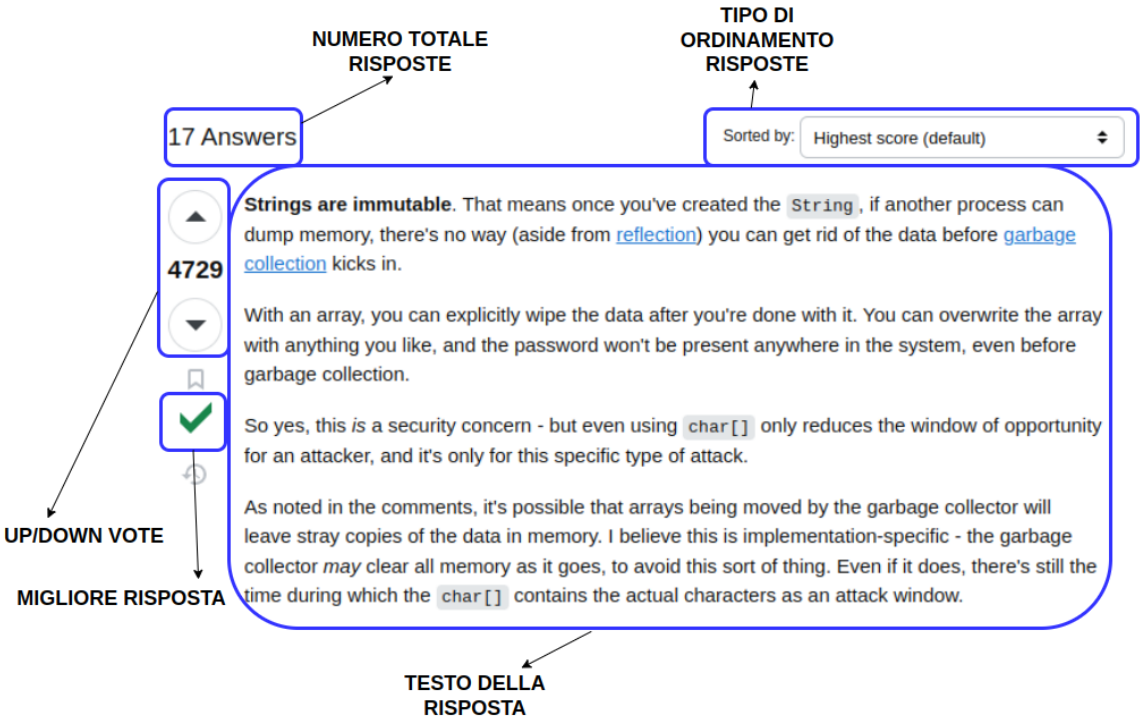

Figura 2.2: Interfaccia Stack Overflow relativa ad una risposta sulla piattaforma.

2.2 NLP e Classificazione dei testi

Il termine NLP (Natural Language Processing) fa riferimento ad una branca dell'informatica, che si occupa di dare ad una macchina la capacità di interpretare i testi scritti in un linguaggio comprensibile dall'uomo. Essa è strettamente collegato al mondo dell'intelligenza artificiale. In generale l'NLP combina il linguaggio naturale a dei modelli matematici e statistici, con lo scopo di rendere il testo scritto interpretabile dagli algoritmi. Diversi sistemi che ne fanno uso sono ad esempio i software di traduzione e assistenti vocali. Tra le principali attività nell'ambito dell'NLP troviamo la generazione di testi, il riconoscimento vocale e la classificazione dei testi.

L'attività di classificazione consiste, date una serie di classi e degli elementi in input, di identificare la classe di appartenenza degli elementi. Con il Machine Learning possiamo effettuare classificazione di immagini, video, audio e anche di testo. L'operazione della classificazione dei testi prende il nome di text classification ed è interessante in quanto ci permette di classificare non solo semplici frasi ma anche testi relativi a conversazioni, interi documenti, recensioni e commenti [3]. Più in generale si usa la parola "documento" per far riferimento ai testi in input.

Alcuni esempi di utilizzo di classificatori di testo sono la classificazione delle notizie e il sentiment analysis. Anche nell'ambito della sicurezza si sono diffusi alcuni classificatori come spam classifier e classificatori di link, i primi per classificare i messaggi o le mail di spam, i secondi per classificare un link come insicuro se si tratta di un attacco di pishing o sicuro altrimenti.

2.3 Approcci di classificazione

Principalmente esistono due approcci per la classificazione dei testi ovvero l'approccio basato sul contenuto e quello basto sulla richiesta. Nella classificazione basata sul contenuto si vanno a considerare gli argomenti principali presenti in un documento, mentre nella classificazione basata su richiesta si va a dare più importanza agli utenti target di quei documenti. Indipendentemente dall'approccio usato per classificare, un modello ha bisogno di apprendere prima alcune informazioni sui dati in input.

Il modo in cui i modelli apprendono differiscono a seconda se si sceglie un approccio supervisionato o non supervisionato. Nel Machine Learning esistono anche altri due metodi di apprendimento che sono quello per rinforzo e quello semi supervisionato, ma in genere nell'ambito della classificazione non vengono usati.

L'apprendimento supervisionato è una tecnica di apprendimento che consiste nel dare al modello di Machine Learning dei dati etichettati. Una etichetta non è altro che un valore che permette di identificare una classe di appartenenza per un elemento. Il modello lavorerà su un dataset in cui i dati sono affiancati da delle etichette, questo facilita l'apprendimento da parte del modello. Durante l'estrazione delle feature dai dati, il modello affiancherà le feature alle rispettive classi grazie all'uso delle etichette.

L'apprendimento non supervisionato è una tecnica che invece non ha bisogno di dati etichettati. Il modello in questo caso andrà ad estrarre le feature dai dati senza sapere la classe di appartenenza. Una volta ottenute le feature si tenta di avvicinare i documenti con feature simili supponendo che questi appartengano alla stessa classe. Questo apprendimento è più complesso e viene utilizzato soprattutto per il document clustering.

Gli algoritmi di classificazione si possono inoltre dividere in base alle classi restituite in output. Esistono classificatori binari, multi classe e multi label. I classificatori binari sono dei classificatori che lavorano solo con due classi. I classificatori multi classe sono una estensione di quelli binari infatti lavorano con un numero di classi maggiori di due. Mentre i classificatori multi label lavorano con due o più classi ma la differenza dagli altri algoritmi sta nel fatto che ad ogni elemento possono essere assegnate più classi di appartenenza.

2.4 Progettazione di un classificatore di testi

Per la realizzazione di un classificatore di testi bisogna eseguire diverse fasi ovvero la preparazione dei dati, la pre elaborazione dei dati, il Feature Engineering, l'addestramento, la valutazione e la distribuzione del modello.

2.4.1 Preparazione dei dati

La fase di preparazione dei dati consiste nell'andare ad organizzare quelli che saranno i dati su cui il modello verrà addestrato. In altre parole in questa fase si andrà a creare il datatset, che in base al tipo di apprendimento conterrà o meno le etichette. In generale il dataset deve contenere abbastanza elementi poiché in seguito, questo, sarà diviso in un training set e test set. Il training set conterrà tutti gli elementi per l'allenamento del modello, mentre il test set conterrà dei dati che il modello non ho mai visto ed è utile per valutare la correttezza dei risultati. Il dataset spesso presenta dati inconsistenti o dati mancanti, pertanto è utile una fase di pulizia. In questa fase vengono usate delle tecniche (Data Imputation) che permettono di pulire il dataset e aggiungere dati mancanti. Tra queste tecniche troviamo alcune molto semplici come l'eliminazione di righe o colonne, ed altre più complesse come l'imputazione statistica o quella deduttiva.

- **Rimozione di righe** - questa tecnica è usata quando nel dataset esistono delle righe che presentano parecchi dati mancanti e consiste nel rimuovere le intere righe. Lo svantaggio di usare questa tecnica riguarda la perdita di dati. Il vantaggio è la facilità di implementazione.

- **Rimozione di colonne** - questa tecnica è la stessa della rimozione delle righe ma applicata alle colonne e presenta lo stesso svantaggio e vantaggio.

- **Imputazione statistica** - permette di inserire dei valori dove mancano. Viene usata a seguito di una indagine statistica sui dati che permette di individuare alcuni valori mancanti. Questa tecnica è semplice da implementare ma lo svantaggio è che può essere applicata solo a dati di tipo numerico.

- **Most Frequent Imputation** - questo metodo inserisce al posto dei dati mancanti i valori più ricorrenti in una colonna. Lo svantaggio è che se ci sono molti dati mancanti, l'uso di questa tecnica, potrebbe influenzare la distribuzione dei dati.

- **Imputazione deduttiva** - questa è la tecnica più efficace ma allo stesso tempo più difficile perché consiste nel trovare una regola di imputazione sulla base di una deduzione logica sui dati.

2.4.2 Pre elaborazione dei dati

La prima fase che permette di lavorare sui dati testuali è quella della pre elaborazione. La maggior parte dei dati che vengono analizzati e utilizzati per l'apprendimento dei modelli nell'ambito dell'NLP, sono dati testuali presi dalla rete. I dati testuali spesso contengono errori, caratteri speciali, tag HTML o parole di uso comune ripetute molte volte. I tipi di parole appena elencate ostacolano l'apprendimento del modello, infatti vengono considerare come rumore. Successivamente alla pre elaborazione i dati vengono convertiti in numeri per poter essere interpretati da una macchina, quindi è importante che parole con lo stesso significato vengano convertite nello stesso dato numerico. Questo non potrebbe avvenire senza una corretta fase di pre elaborazione.

Questa fase è molto importante [4], il suo obiettivo è quello di eliminare il rumore dai dati testuali grezzi attraverso le seguenti tecniche:

- **Lowercasing** - è un'operazione semplice che consiste nell'andare a trasformare ogni carattere del testo grezzo in minuscolo. Questo fa si che ad esempio la parola "NLP" e "nlp" vengano trattate allo stesso modo.

- **Puntuaction Removal** - è un'altra tecnica che consiste nell'andare a rimuovere i segni di punteggiatura dal testo. Questa tecnica è applicata siccome la punteggiatura non aggiunge significato al testo ed inoltre permette di ridurre la dimensione dei dati.

- **Stopword Removal** - le stopwords sono quelle parole utilizzate tanto, ma che non fanno riferimento allo specifico argomento del testo. Un esempio di stopword possono essere gli articoli o le preposizioni. Togliendo le stopwords da una frase fa si che l'attenzione si focalizza sulle altre parole. Un modo per eliminare le stepword è quello di andare a costruire un vocabolario che contiene le principali stopwors, oppure possiamo andare ad utilizzare alcune librerie con metodi già pronti per ripulire il testo grezzo dalle stopwords. Un esempio di libreria per lo stopword removal in Python[4] è `TextBlob`[5].

- **Text Standardization** - nel testo grezzo, soprattutto quello preso da forum, sono presenti molte abbreviazioni o acronimi. Il text standardization permette di trasformare queste abbreviazioni nella forma standard, in maniera da uniformare quella parola con il resto del testo. Anche in questo caso l'implementazione di questa funzione richiede la costruzione di un vocabolario per individuare le abbreviazioni e acronimi utilizzati frequentemente in quella lingua.

- **Spelling Correction** - durante la scrittura attraverso la tastiera, si possono digitare delle parole contenenti errori di spelling come un carattere mancante. L'implementazione di tecniche di spelling correction permette di individuare questi tipi di errori e correggerli. Esistono librerie con metodi capaci di fare spelling correction.

- **Tokenization** - questa tecnica permette di andare a dividere una frase in più blocchi, ogni blocco è detto token. Il token può essere una singola parola, un insieme di più parole o anche una sillaba, questo dipende da tipo di tokenizzazione che viene effettuata. Quando il token è rappresentato da una singola parola, allora stiamo utilizzando un word tokenizer, mentre se un singolo token

[4] `https://www.python.org/`
[5] `https://textblob.readthedocs.io/en/dev/`

è costituito da più parole stiamo utilizzando un sentence tokenizer. La tokenizzazione è utilizzata in quanto facilita il calcolo della frequenza delle parole in un determinato testo.

- **Stemming e Lemmatization** - queste due tecniche sono simili in quanto permettono, data una parola, di estrarre la sua radice. Consideriamo le parole "pescare" e "pescato", la radice è "pesca". Senza ridurre le parole alla loro radice, avremmo tante parole che verranno codificate con diversi numeri ma che si riferiscono allo stesso concetto. Lo stemming e Lemmatization facilitano l'apprendimento del modello. La differenza principale tra le due tecniche è che il Lemmatization estrae la radice utilizzando un vocabolario e questo la rende più efficace. Consideriamo le parole "macchina" e "auto", queste due parole fanno riferimento allo stesso concetto, utilizzando lo Stemming otterremo due radici diverse mentre utilizzando il Lemmatization otterremo una sola radice.

Oltre a queste tecniche, si possono utilizzare funzioni ad hoc in base al testo a nostra disposizione. Ad esempio immaginiamo che il testo grezzo è stato ottenuto da pagine HTML. Questo testo conterrà tag HTML che non influenzano il significato, perciò si può procedere con la creazione di una funzione che individua e cancella i tag HTML dal testo grezzo.

L'uso di una tecnica non esclude le altre infatti nella fase di pre elaborazione si può combinare l'uso di più tecniche. Si può implementare, ad esempio, una pipeline di funzioni di pulizia, ovvero si sviluppa una funzione che preso il testo grezzo in input andrà ad applicargli a cascata più funzioni. In output la pipeline ci restituirà il testo pulito.

2.4.3 Feature Enginering

Una volta terminata la fase di pre elaborazione, il testo ottenuto deve essere trasformato in un formato comprensibile da una macchina quindi in valori numerici. Esistono diversi metodi e la scelta di questi influenza l'efficienza del modello da realizzare. Qui di seguito vengono descritti i principali:

- **One Hot Encoding** - questo modello è abbastanza semplice, esso considera tutte le parole (prendendo una sola istanza per quelle ripetute) e ad ognuna associa un array a valori binari. Se nel testo sono state individuate n parole allora l'array usato per rappresentare le parole sarà di lunghezza n. I vettori saranno unici in modo tale che ogni vettore corrisponde ad una sola parola. Questo modello è chiamato One Hot Encoding perché utilizza dei vettori costituiti da un gruppo di bit tutti posti a 0 tranne che un solo bit posto invece a 1. Le frasi vengono considerate come un'array bidimensionale.

- **N-Gram** - il modello N-Gram non considera le singole parole come feature, ma una feature può essere vista anche come una combinazione di n parole adiacenti. Consideriamo una frase che contiene le due parole adiacenti "non male", il significato di queste parole considerate insieme è "bene", ma se considerate separate il significato potrebbe diventare diverso. L'uso della tecnica N-Gram risolve il problema dell'esempio precedente perché permette di considerare più parole infatti un Token può essere costituito da un unigramma (una sola parola), un digramma (due parole), trigramma (tre parole) e più in generale da un n-gramma (n parole).

- **Co-occurence Matrix** - questo modello genera come output una matrice n*n dove n è il numero di parole che sono state individuate nel testo. La matrice rappresenta il legame che c'è tra le parole. Ogni cella [x,y] della matrice, rappresenta il numero di volte che la parola nella riga x è apparsa insieme alla parola nella colonna y.

- **Hash Vectorizing** - il modello Hash Vectorizing risolve il problema della memoria relativo al modello Co-Occurence Matrix, infatti, questo modello quando usato con grandi quantità di testo occupa molta memoria. L'Hash Vectorizer permette appunto di risparmiare memoria utilizzando una tabella hash. Il costo da pagare, però, è una più complessa interpretazione dei dati e la possibilità di ottenere delle collisioni.

- **Bag of Word** - in questo modello i documenti di testo in input vengono convertiti in vettori e ogni vettore rappresenta un documento. I vettori hanno una

lunghezza pari ad n, ovvero il numero di parole individuate in tutti i documenti. Il vettore contiene dei valori che indicano il numero di occorrenze per ogni parola nel documento. Una parola può avere anche frequenza 0 se questa non appare in un determinato documento. Il modello Bag of Word va a contare il numero di occorrenze delle singole parole ma esiste anche una variante detta N-Gram Bag o Word che va a contare le occorrenze degli n-grammi in un documento. Nella Figura 2.3 è mostrato come vengono viste le frasi sotto forma di vettori.

	and	beautiful	blue	cheese	is	love	sky	so	the
0	0	0	1	0	1	0	1	0	1
1	1	1	1	0	2	0	2	0	0
2	0	1	1	0	1	0	1	1	1
3	0	0	1	1	0	1	0	0	0

Figura 2.3: Esempio di vettorizzazione tramite Bag of Word di 4 documenti.

- **Tf-Idf** - nel modello Bag of Words può verificarsi un errore durante l'apprendimento. I termini molto frequenti otterranno un valore di frequenza alto, andando a mettere in secondo piano alcune parole che appaiono poche volte ma che potrebbero essere più significative per l'apprendimento del modello. Questo problema viene risolto nel modello Tf-Idf, infatti, in questo modello vengono calcolate, per ogni Token in un documento, due metriche. Queste due metriche sono il Term Frequency e l'Inverse Document Frequency. Il valore finale Ff-Idf sarà dato dal prodotto delle due metriche. Il term frequency è quello che viene calcolato nel modello Bag of Word, ovvero è il numero di volte che una parola appare all'interno di un documento. L'Inverse Document Frequency invece si calcola con la seguente formula:

$$idf(t) = 1 + log(\frac{C}{1 + df(t)})$$

Il valore di C corrisponde al numero totale di documenti, mentre df(t) rappresenta il Term Frequency della parola t. Al denominatore, nell'argomento del logaritmo, df(t) è sommato con uno per evitare divisioni per zero, e tutto il

logaritmo è sommato ad uno in maniera tale da evitare i termini con valore Itf pari a zero.

2.4.4 Addestramento del modello

In questa fase viene usato un algoritmo di classificazione, questo lavorerà sulle feature estratte nella fase precedente ed in particolare cerca di associare le feature alle classi. Gli algoritmi di classificazione, in generale, sono di apprendimento supervisionato e tra questi troviamo:

- **Naive Bayes** - l'algoritmo naive bayes è un algoritmo di classificazione che si basa sulla probabilità. In particolare calcola la probabilità che una istanza facciano parte di una classe tramite l'applicazione del teorema di Bayes.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

  P(A) e P(B) rappresentano la probabilità di osservare le caratteristiche A e B indipendentemente l'una dall'altra, P(B | A) rappresenta la probabilità di osservare B dato che A si è già verificato, mentre P(A | B) rappresenta la probabilità di osservare A dato che si è già verificato B. Il naive bayes è semplice poiché assume che le feature siano indipendenti tra di loro. Durante la valutazione, l'algoritmo non va a considerare l'utilità data dalla combinazione di più caratteristiche.

- **Decision Tree** - il Decision Tree è un algoritmo che mira ad organizzare le istanze del dataset all'interno di una struttura ad albero. I nodi dell'albero rappresentano dei sottoinsiemi di caratteristiche, mentre gli archi rappresentano le decisioni. Questa struttura fa si che ogni percorso dell'albero corrisponde ad una regola, perciò data una nuova istanza è possibile capire la classe di appartenenza in base alle decisioni (quindi gli archi) percorsi. Per creare un Decision Tree si seguono tre passi principali. Il primo passo consiste nell'individuare una buona feature e la si posiziona al nodo radice. Una buona feature divide in maniera adeguata il training set. Il secondo passo consiste invece nel dividere il training set in sottoinsiemi. Se un sottoinsieme contiene solo elementi della stessa classe allora viene detto puro, altrimenti il sottoinsieme non è puro. In

questo caso c'è ancora incertezza nei dati e si procede con l'individuazione di un'altra feature, che divida ulteriormente il sottoinsieme. Il terzo passo consiste nel ripetere i primi due passi per ogni sottoinsieme fino a quando non si ottengono solo insiemi puri. L'operazione fondamentale nel Decision Tree è quella di scegliere le migliori feature per dividere i dati. Una feature viene valutata, ad esempio, attraverso l'information Gain. Questa metrica misura quanto una feature divide adeguatamente il dataset. Il calcolo dell'information Gain si basa sul concetto di entropia. L'entropia misura il grado di complessità di un messaggio, maggiore è l'entropia di un messaggio minore sarà l'informazione che esso contiene. Nel Decision Tree l'entropia è usata per dividere i dati in sottoinsiemi. Per ogni attributo verrà calcolato l'information Gain tramite la formula:

$$Gain(D, A) = H(D) - \sum_{v \in values(A)} \frac{|Dv|}{|D|} * H(Dv)$$

D è l'entropia del dataset, Dv è il sottoinsieme di D per cui l'attributo A ha valore v, |Dv| è il numero di elementi di Dv e |D| è il numero di elementi del dataset. La Figura 2.4 mostra un esempio ad alto livello di un Decision Tree.

Alcuni algoritmi sono più adatti per il task della Text Classification come Multinomial Naïve Bayes e Support Vector Machines in quanto sono meno tendenti all'Overfitting. L'Overfitting insieme all'Underfitting sono due problemi che si verificano quando un modello apprende male dai dati. L'Overfitting si ha quando un modello apprende troppe informazioni dai dati di addestramento. Di conseguenza il modello è incapace di classificare correttamente nuove istanze anche se di poco diverse da quelle viste nel training set, in quanto incapace di generalizzare. L'Underfitting si ha quando, invece, il modello apprende troppo poco quindi questo non ha abbastanza conoscenza per fare una predizione corretta.

- **Multinomial Naive Bayes** - Il Multinomial Naive Bayes è un'estensione dell'algoritmo Naive Bayes. Infatti questo modello lavora con un numero di classi superiore a due. In questo modello, le feature analizzate sono i vettori ottenuti dalla vettorizzazione del testo tramite il modello Bag of Words o TF-IDF. Sia F una frase e V il vettore relativo alla frase calcolato con il modello Bag o Word, sia inoltre D un insieme di documenti appartenenti alla classe C. La classificazione

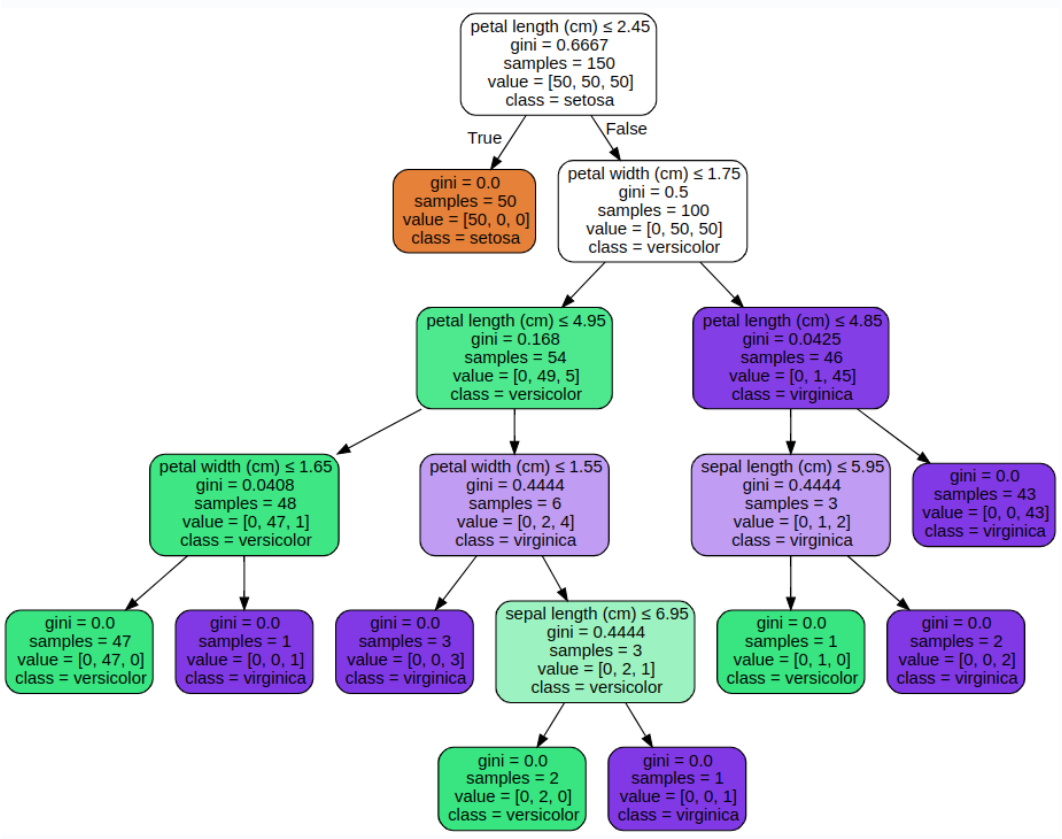

Figura 2.4: Albero decisionale su tre classi relativo al dataset Iris della libreria `sklearn`.

avviene calcolando la probabilità che i termini in F siano presenti nei documenti appartenenti a D.

- **SVM - SVM** sta per Support Vector Machine ed è un set di algoritmi per l'apprendimento supervisionato, che vengono utilizzati per diversi tasks come la classificazione, la regressione e l'individuazione di outlier. Tra i vari algoritmi di questo set troviamo il SVC (Support Vector Classification) ed il NuSVC. Questi due algoritmi sono molto simili tra di loro ma differiscono nella funzione matematica che implementano. Ogni algoritmo di questo set differisce dagli altri a seconda della funzione che implementa, questa funzione è detta kernel function. Esistono diverse kernel function come la polinomiale, la lineare e rbf. Inoltre c'è la possibilità di creare una funzione kernel custom adatta ai dati del problema. Oltre l'algoritmo SVC e NuSVC troviamo il LinearSVC che risulta più veloce del SVC classico. Questi algoritmi lavorano sui dati organizzandoli come punti in uno spazio. Dato un problema di classificazione binaria l'algoritmo individua un iperpiano che divide nettamente i punti nello spazio. I punti da un

lato dell'iperpiano appartengono ad una classe, mentre i punti al lato opposto apparterranno all'altra classe. Una buona separazione dei dati si ha quando la distanza tra i punti e l'iperpiano è ampia. Gli algoritmi SVM si possono

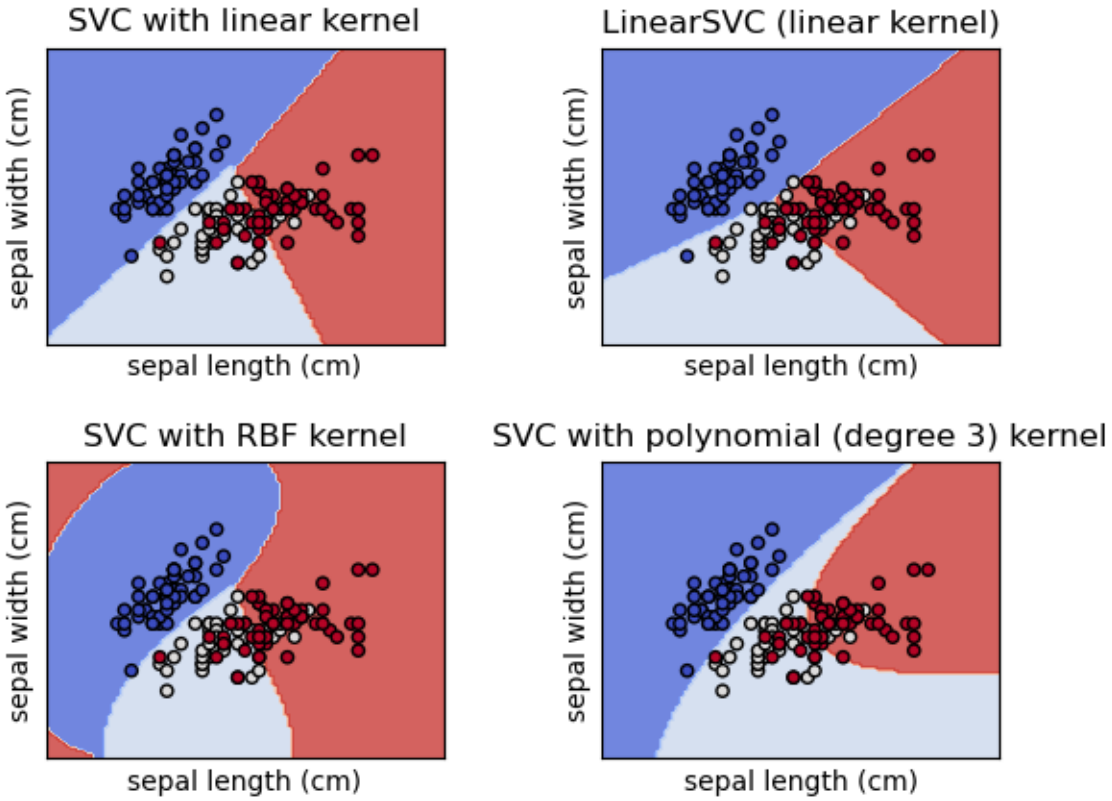

Figura 2.5: Esempio di utilizzo degli algoritmi SVC, LinearSVC, SVC con funzione kernel RBF e SVC con funzione polinomiale sugli stessi dati.

utilizzare anche per classificazione multi classe ma a seconda dell'algoritmo si procede in modi diversi. Con SVC e NuSVC si segue un approccio del tipo "one-versus-one". L'approccio "one-versus-one" consiste nel creare (num_classi * (num_classi - 1) / 2) classificatori, dove num_classi rappresenta il numero delle classi. Ogni classificatore lavora su due classi. Per unire i risultati ottenuti, si passano ad una funzione di decisione che usa un approccio del tipo "one-versus-rest". Mentre l'algoritmo LinearSVC utilizza direttamente l'approccio "one-versus-rest" dove si creano num_classi classificatori. Ogni classificatore è un classificatore binario tra la classe considerata e le num_classi -1 classi restanti. Nella Figura 2.5 sono mostrati degli esempi di rappresentazione delle istanze nello spazio in base alla unzione kernel utilizzata.

Una volta scelto l'algoritmo da utilizzare per la classificazione, gli verrà passato il training set per l'addestramento ed il test set per predire le classi delle nuove istanze. I dati nel test set sono dei valori nuovi che il modello non ha mai visto. I risultati predetti dal modello verranno salvati e saranno usati nella fase di valutazione.

In alcuni casi il train set non è bilanciato e questo può ostacolare l'addestramento del modello. Il bilanciamento dei dati permette di risolvere questo problema. Esso consiste nell'aggiungere o eliminare documenti dal train set affinché il numero di questi sia simile per ogni classe. Esistono due approcci per bilanciare il dataset ovvero l'Undersampling e l'Oversampling. L'Undersampling consiste nell'eliminare delle istanze della classe di maggioranza. Questa tecnica presenta dei problemi, infatti, l'eliminazione dei dati potrebbe rendere impossibile l'apprendimento da parte del modello e la perdita di alcune istanze particolarmente rilevanti. Dall'altra parte troviamo invece la tecnica dell'Oversampling che consiste nel generare nuove istanze per la classe di minoranza, partendo da quelle esistenti.

2.4.5 Valutazione e distribuzione del modello

Nella fase di valutazione i risultati previsti dal modello vengono confrontati con i dati attesi. Esistono diverse metriche per valutare i risultati di un modello:

- **Matrice di confusione** - è una matrice nxn dove n indica il numero di classi su cui si lavora. Nel caso di classificazione binaria avremo una matrice 2x2, dove sulle colonne abbiamo i valori predetti e sulle righe i valori attesi. Il contenuto delle quattro celle indicano: i veri positivi, i veri negativi, i falsi positivi e i falsi negativi. Sulla diagonale principale della matrice si trovano i valori predetti in maniera corretta e che quindi corrispondono con il risultato atteso, mentre nelle altre celle avremo i valori delle istanze che sono state predette in maniera errata. Nella Figura 2.6 è mostrata una matrice di confusione dei un problema di classificazione binaria.

- **Accuracy** - indica il numero di predizioni corrette per tutte le classi.

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

	p' (Predicted)	n' (Predicted)
p (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

Figura 2.6: Matrice di confusione di un problema di classificazione binario

- **Precision** - la precision è il rapporto tra il numero delle previsioni corrette di una classe sul totale delle volte che il modello lo prevede.

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

- **Recall** - la recall misura la sensibilità del modello. E' il rapporto tra le previsioni corrette per una classe sul totale dei casi in cui si verifica effettivamente.

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

- **F1 Score** - è la media armonica della Precision e Recall.

$$F1score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Queste sono le metriche principali. Esistono, infatti, altre metriche utilizzabili anche nel contesto della classificazione multi classe [5]. Dopo la valutazione, con la fase della distribuzione del modello, il prodotto ottenuto verrà reso disponibile ed utilizzabile. Da questo momento in poi il modello sarà monitorato costantemente per verificare la necessità di manutenzione.

2.5 Stato dell'arte

Attualmente esistono già diversi lavori di ricerca, condotti sui dati presi da Stack Overflow e piattaforme simili, con obiettivi diversi. In questo capitolo, verrà fatta una panoramica dei lavori precedenti, correlati alla classificazione di post su Stack Overflow, e alla classificazione dei testi applicata alle tematiche di sicurezza.

Nel lavoro di ricerca di Triet Huynh Minh Le, Croft Roland, David Hin, M. Ali Babar [1], viene analizzato un insieme di post relativi a vulnerabilità di sicurezza, presi da Stack Overflow e Security Stack Exchange. La ricerca ha permesso di individuare ben 13 topic di appartenenza dei post analizzati ed ogni post è stato assegnato ad una classe. Per arrivare a questo risultato è stato utilizzato il modello LDA (Latent Dirichlet Allocation).

Inoltre un altro risultato importante della ricerca è stato l'individuazione dei topic più popolari e più complessi nell'ambito della sicurezza. I topic più popolari sono risultati Brute-force/Timing attacks e Vulnerability Theory, mentre i post più complessi sono Vulnerability Scanning Tools e Network Attacks.

I risultati relativi ai topic più popolari sono stati ottenuti tenendo in considerazione quattro parametri collegati ad ogni post ovvero il numero di visualizzazioni di un post, il punteggio, il numero di preferiti e numero di commenti. Se il valore di questi parametri sono alti significa che il post ha attirato molto interesse e pertanto può essere valutato come appartenente ad un topic popolare.

Per quanto riguarda, invece, i topic più complessi sono stati tenuti in considerazione altri tre parametro ovvero la percentuale di risposte accettate, il tempo medio di risposta ed il rapporto medio tra risposte e visualizzazioni. I topic più complessi hanno ottenuto un tempo medio di risposta alto mentre la percentuale di risposte e il rapporto tra risposte e visualizzazioni hanno ottenuto un valore basso.

Nel lavoro di N. Meng, S. Nagy, D. Yao, W. Zhuang e G. A. Argoty [2], si focalizza l'attenzione sulla sicurezza in Java, analizzando sempre i post di StackOverflow. La ricerca ha come obiettivo quello di individuare quali sono le maggiori preoccupazioni e le vulnerabilità più comuni nell'ambiti Java.

Dalla ricerca sono emersi alcuni risultati come il fatto che la maggior parte dei post che contengono codice sono relativi a Java Spring e Java EE. Le principali preoccupazioni riguardano il tema dell'autenticazione e la configurazione delle applicazioni enterprise. Una osservazione importante emersa da questo studio riguarda il fatto che spesso nei post relativi all'implementazione sono contenuti snippet di codice obsoleto, che potrebbe essere potenzialmente vulnerabile.

Nella ricerca di F. Fischer, K. Bottinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, e S. Fahl [6], si cerca di valutare l'impatto che ha il copia ed incolla, di snippet di codice da StackOverflow, sulla sicurezza delle applicazioni Android[6]. In questa ricerca[6] si prova a rispondere a questa domanda.

Nonostante la comodità di copiare il codice, quando si tratta di sicurezza ci sono alcune considerazioni da tenere a mente, infatti è molto difficile fornire soluzioni pronte all'uso e sicure per ogni problema. L'analisi è stata condotta sulle applicazioni Android disponibili sul Google Play Store. Sono stati individuati frammenti di codice non sicuri copiati in applicazioni Android, che milioni di utenti installano ogni giorno.

Per quantificare l'entità del problema, sono stati scansionati i frammenti di codice presi da StackOverflow ed è stato valutato se essi sono sicuri o meno tramite un classificatore. I risultati sono che il 15.4% degli 1.3 milioni di applicazioni Android che sono state analizzate conteneva frammenti di codice relativi alla sicurezza copiati da Stack Overflow e di questi, il 97.9% contiene almeno uno snippet di codice non sicuro.

Nel lavoro di ricerca di Seyed Mohammad Hossein Hasheminejad e Saeed Jalili [7], viene presa in considerazione la grande quantità di Design Pattern di sicurezza. Negli ultimi anni, questi si sono diffusi notevolmente per la loro comodità e per il fatto che il loro uso, nello sviluppo del software, rappresenta una garanzia aggiuntiva.

Il grande numero di Design Pattern di sicurezza ha permesso di condurre la ricerca appena citata, che ha come obiettivo quello di valutare diversi modelli di classificazione. I modelli presi in esame sono stati il Naive Bayes, Decision Tree, Support Vector Machines e il k-nearest neighbor. Il metodo seguito permette di

[6] `https://www.android.com/intl/it_it/`

classificare i problemi di sicurezza nella classe di Design Pattern che si adatta meglio a quel problema.

I risultati hanno mostrato che il modello migliore per questo task è il Naive Bayes, in quanto esso permette di raggiungere i risultati migliori in termini di precision, recall ed F1 score.

Il lavoro di ricerca di Hanmin Qin e Xin Sun [8], considera il problema dei Bug Report che vengono considerati erroneamente come tali. La classificazione manuale dei report permette di ridurre gli errori, ma allo stesso tempo richiede molto lavoro.

Esistono già alcuni metodi per classificare i Bug Report, tra questi troviamo quelli che si basano sull'individuare i topic nei documenti. Nella ricerca [8] viene proposta un'altra soluzione per classificarli, dove ogni Bug Report viene classificato come Bug o non-Bug Report. Il metodo proposto si basa sulla classificazione dei testi attraverso LSTM (Long Short-Term Memory), ovvero una rete neurale.

La classificazione è avvenuta su dati ottenuti da quattro database di progetti Java open-source, ed i risultati mostrano che il metodo proposto permette di ottenere classificazioni più accurate rispetto i metodi esistenti. Inoltre, i risultati di questa ricerca hanno mostrato come l'errata classificazione dei Bug Report è un errore frequente.

CAPITOLO 3

Sperimentazione dei Classificatori di Domande

Nel capitolo vengono descritti gli elementi principali che sono stati utilizzati ed il flusso di lavoro che è stato seguito per poter poi rispondere alle domande di ricerca. I passi fondamentali del metodo seguito sono mostrati nella Figura 3.1.

3.1 Domande di ricerca

L'obiettivo di questa ricerca è quello di realizzare un classificatore di testi, in particolare il classificatore deve identificare la classe di appartenenza di domande relative alla sicurezza informatica prese da Stack Overflow e da Security Stack Exchange. Le domande di ricerca che sono state poste sono due. La prima riguarda la scelta del modello di classificazione più adatto al contesto di applicazione di questa ricerca. In questo capitolo verranno descritti i modelli presi in considerazione e il metodo per valutarli. La seconda domanda è relativa al problema dello sbilanciamento dei dati nel dataset. Verranno valutate due tecniche di bilanciamento dei dati per stabilire quella più efficace.

Di seguito sono presentate le domande di ricerca:

RQ$_1$. *Quale è il modello di classificazione più adatto per determinare la tipologia di domanda relativa alla sicurezza?*

RQ$_2$. *Qual è l'algoritmo di bilanciamento più adatto per determinare la tipologia di domanda relativa alla sicurezza?*

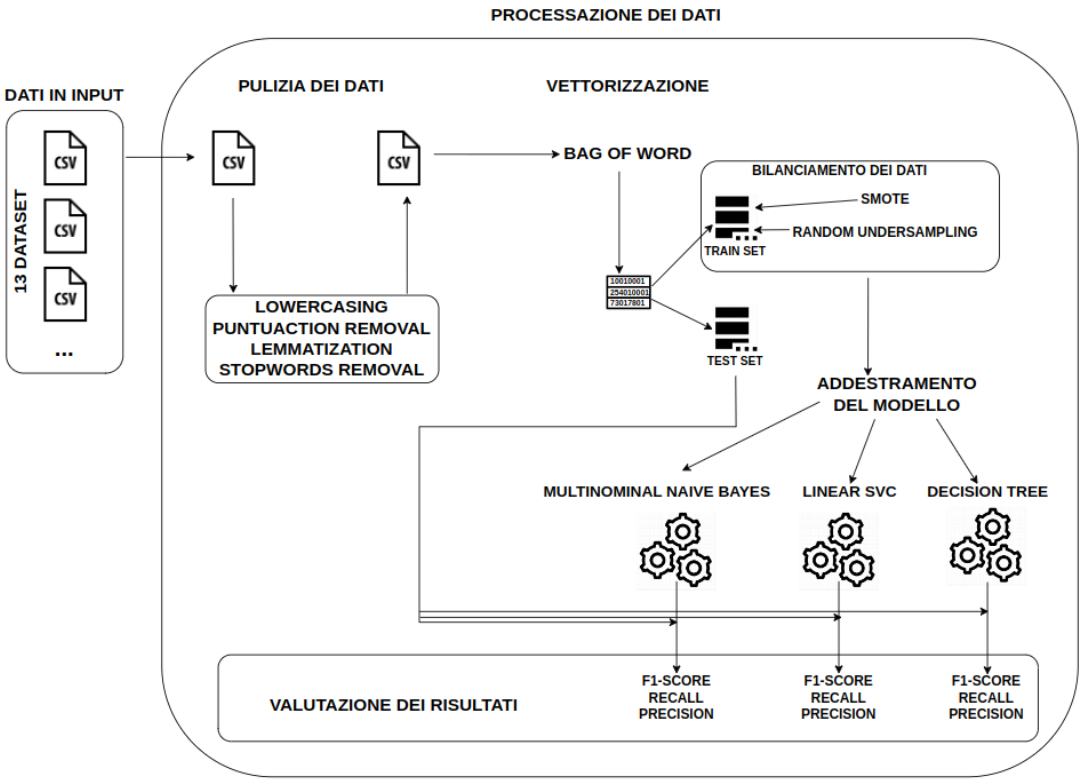

Figura 3.1: Flowchart relativo all'approccio utilizzato.

3.2 Costruzione del Dataset

La fase immediatamente successiva a quella della raccolta delle informazioni è quella che consiste nel cercare i dati su cui lavorare. In questa fase è stato utile la piattaforma GitHub[1]. GitHub è una piattaforma multo utilizzata nell'ambito dello sviluppo software in quanto fornisce servizi di version control del codice, per conservare e condividere il codice con altri sviluppatori. Il dataset che è stato utilizzato è di Triet Huynh Minh Le, Roland Croft, David Hin e Muhammad Ali Babar, gli autori del lavoro di ricerca citato nel capitolo precedente[1]. I risultati ottenuto da

[1] `https://github.com/`

questo lavoro e tutta la parte implementativa sono presenti sulla repository pubblica [2] del progetto.

Tra le domande di ricerca, risposte nella ricerca citata, troviamo "*What are SV discussion topics on Q&A sites?*". La soluzione proposta a questa domanda consiste nell'andare, dato un dataset iniziale di post relativi al tema della sicurezza informatica, a dividere i post in base al loro topic di appartenenza. Il risultato ottenuto è stato l'individuazione di 13 topic di appartenenza tramite l'algoritmo LDA (Latent Dirichlet Allocation).

L'algoritmo LDA è un algoritmo non supervisionato che permette di identificare argomenti all'interno di una raccolta di documenti. L'algoritmo utilizza il parametro k che indica il numero di topic in cui dividere i documenti. Il valore del parametro k utilizzato per rispondere alla domanda è pari a 13 ed è stato ottenuto facendo variare il parametro in un range che va da 2 ad 80, scegliendo il migliore. All'interno della reposotory del progetto i post sono stati divisi in 13 dataset.

Ogni dataset contiene le informazioni relative ai post come il postId che permette di identificare il post all'interno della piattaforma di Stack Exchange [3], l'answer ovvero le risposte date ad un post di domanda, il creation_date che indica la data di creazione del post, len che indica la lunghezza del post calcolata contando le parole, question che indica la domanda del post, il ratio ovvero il rapporto tra le risposte date e le visualizzazioni, la source che indica la sorgente di provenienza del post quindi SO per indicare StackOverflow o SSE per indicare Security Stack Exchange, il tags che indica un elenco di tags assegnati al post ed il titolo del post.

Una volta ottenuti i dati divisi in 13 file si è proceduto unendo tutti i post in un unico dataset, che è stato utilizzato per le successive fasi. Il dataset risultante è costituito da due colonne una colonna "text" che contiene il testo delle domande ed una colonna "label" che contiene un numero che va da 0 a 12 ed indica il topic di appartenenza. La Tabella 3.1 illustra i 13 topic e le label ad essi associati.

[2] `https://github.com/lhmtriet/SV_Empirical_Study`
[3] `https://stackexchange.com/`

Topic Name	Label	Example
Malwares	0	Can Windows 10 bootable USB drive get infected while trying to reinstall Windows?
SQL Injection	1	How to parameterize complex oledb queries?
Vulnerability Scanning Tools	2	How do I turn off automated testing in OSWAP ZAP.
Cross-site Request Forgery	3	What is double submit cookie? And how it is used in the prevention of CSRF attack?
File-related Vulnerabilities	4	Is SAXParserFactory susceptible to XXE attacks?
Synchronization Errors	5	How to avoid dead lock due to multiple oledb command for same table in ssis.
Encryption Errors	6	How is it that SSL/TLS is so secure against password stealing?
Resourse Leaks	7	I m using mapview ver 10.0.1 i am getting a memory leak mapview is holds activity context leakcanary trace.
Network Attaks	8	How to prevent ARP spoofing attack in college?
Memory Allocation Errors	9	Segmentation fault removal duplicate elements in unsorted linked list.
Cross-site Scripting	10	How does the anchor tag (<a>) let you do an Reflected XSS?
Vulnerability Theory	11	How to properly disclose a security vulnerability anonymously?
Brute-force/Timing Attaks	12	What's the big deal with brute force on hashes like MD5.

Tabella 3.1: Tabella rappresentante le 13 classi del dataset.

Pulizia del Dataset

Il dataset è stato pulito tramite una pipeline che comprende funzioni per il Lowercasing, Punctuation Removal, Lemmatization e Stopwords Removal. Il Lowercasing è stata implementata tramite la funzione Python `lower()` applicata ad ogni documento del dataset. Il Punctuation Removal è stata implementata tramite il metodo `replace(str, str)` che prende in input il testo ed un carattere di punteggiatura. Se individuato un carattere di punteggiatura allora viene sostituito da uno spazio. Per quanto riguarda il Lemmatization è stato implementata grazie all'uso della libreria `Textblob` [4] che fornisce delle funzioni per individuare le radici delle parole. Invece

[4] `https://textblob.readthedocs.io/en/dev/`

lo Stopwords Removal è stato implementato tramite la libreria `nltk` [5]che fornisce una funzione per trovare le stopword.

Divisione del dataset

Una volta che i documenti sono stati puliti, il dataset viene diviso in due parti. Il training set che contiene i dati sui quali il modello può lavorare per apprendere le caratteristiche e il test set che contiene i dati che verranno utilizzati per testare il modello. La divisione del dataset scelta è 80-20. Questo significa che l'80% dei dati sarà usato per l'addestramento, mentre il restante 20% è utilizzato per il test. Nella Figura 3.2 è rappresentata la divisione delle istanze nel train set e test set.

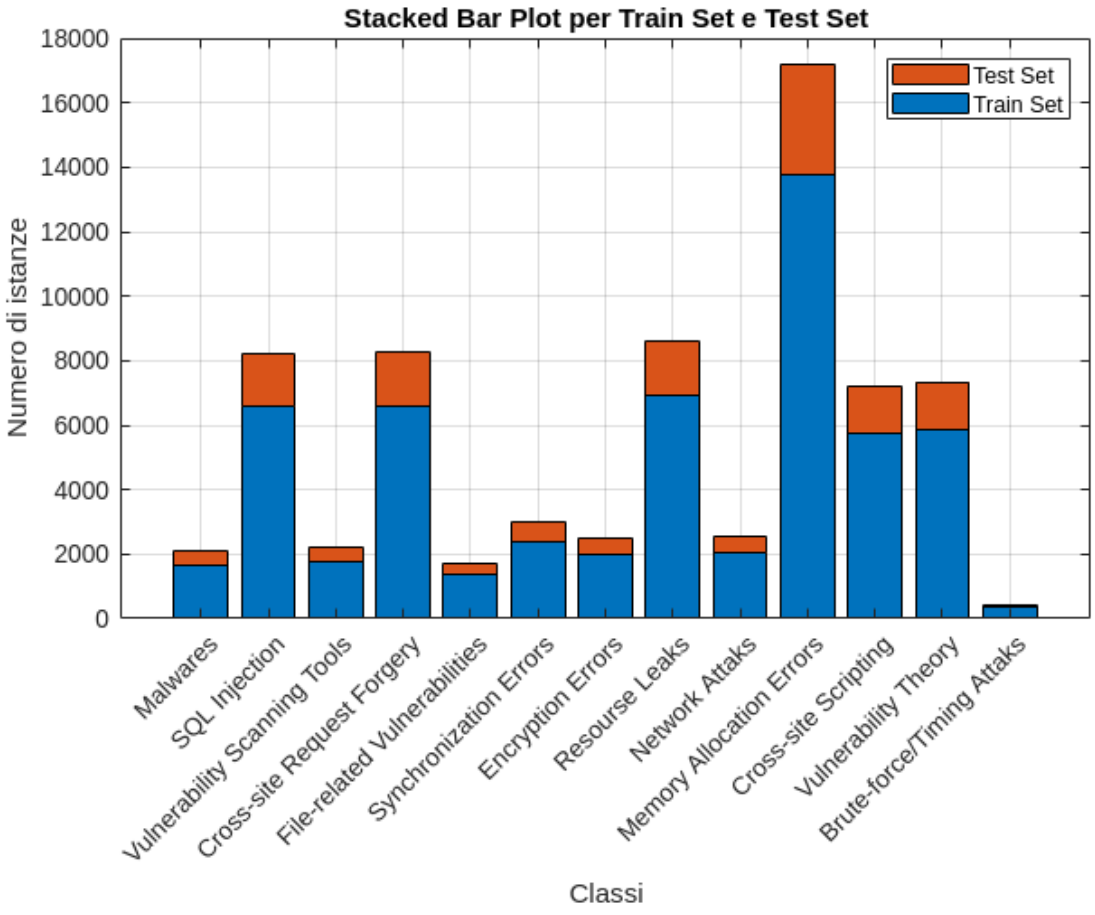

Figura 3.2: Distribuzione istanze nel train set e test set.

[5]`https://www.nltk.org/`

Vettorizzazione dei dati

Successivamente alla divisione del dataset i dati devono essere convertiti in un formato comprensibile da una macchina quindi in numeri. Il testo verrà convertito in vettori attraverso un algoritmo. Questo viene scelto in base ai dati e all'obiettivo. Gli algoritmi presi in considerazione sono il Co-occurrence matrix, il Bag of Word e TF-IDF.

La tecnica che è stata scelta è Bag of word. Questo tecnica mantiene come informazione il numero di occorrenze di ogni parola all'interno di un documento. Questa informazione risulta essere necessaria e sufficiente per quanto riguarda il task della classificazione. Inoltre il modello Bag of Word si adatta bene ai task di classificazione dei testi [9]. Gli altri modelli sono stati invece scartati a seguito di alcune considerazioni.

La prima considerazione da fare è che il modello da realizzare è un classificatore, che deve riconoscere la classe di appartenenza di un post di StackOverlow. Per fare questo non ha bisogno di comprendere per intero il significato dei testi che gli vengono passati, bensì deve trovare le parole (feature) caratterizzanti di una frase e associarla alla classe.

Data questa considerazione ora bisogna concentrarsi sul modo in cui questi algoritmi lavorano. Per farlo prendiamo in esempio due frasi con diverso significato ma che fanno riferimento allo stesso argomento. Le frasi *"Le password sono migliori delle passkey per autenticarsi ai servizi online."* e *"Le passkey sono migliori delle password per autenticarsi ai servizi online."* contengono al loro interno le stesso parole, la differenza sta nel loro ordine. In particolare le parole "password" e "passkey" si invertono facendo si che le due frasi esprimano due significati diversi.

In un contesto dove il modello deve comprendere il significato, queste due frasi sono diverse tra di loro. Pertanto abbiamo bisogno di algoritmi che vadano a considerare le relazioni che ogni parola ha con le altre. In un contesto di classificazione degli argomenti di una frase, il modello può considerare queste due frasi allo stesso modo. Infatti possono essere ricondotte entrambe, ad esempio, ad una classe "autenticazione". Quindi anche se le frasi esprimono due cose differenti parlano dello stesso argomento.

Dato questo esempio ora consideriamo gli algoritmi disponibili. L'algoritmo Co-occurrence matrix crea una tabella nxn, dove n è il numero di parole individuate nei documenti. Sulle righe della tabella troviamo le n parole e lo stesso sulle colonne. Ogni cella rappresenta l'affinità che c'è tra due parole espressa come il numero di volte in cui le due parole si sono trovate adiacenti nei documenti. Pertanto la Co-occurrence matrix si può utilizzare quando c'è bisogno che il modello comprende il significato della frase.

La seconda considerazione da fare riguarda invece le parole presenti nei documenti a seguito della pulizia. La pipeline di pulizia implementata, tra le diverse funzione presenta quella che permette di eliminare le stopword. Le stopword ostacolano l'apprendimento del modello, infatti, se non vengono tolte un modello potrebbe considerare alcune parole di uso comune, che non fanno riferimento all'argomento del documento, come caratterizzanti per quella classe.

L'algoritmo Tf-Idf tiene in considerazione ciò, infatti, per ogni parola calcola questo valore Tf-Idf. Tf è la frequenza della parola e sarà più alta nei termini più comuni. Idf è la frequenza inversa di una parola in tutti i documenti e sarà bassa nelle parole comuni e alta per i termini più specifici di un documento. Attraverso questo valore è quindi possibile dare più importanza ai termini caratterizzanti di un documento.

Nonostante l'utilità di questo modello, la pipeline di pulizia comprende, come già descritto, una funzione della libreria `nltk` per rimuovere le stopword nei testi in lingua inglese. Questo comporta che il testo quando arriva in input al modello di vettorizzazione è già ripulito dalle stopword, rendendo meno utile il modello Tf-Idf.

3.3 Scelta dei modelli

Dopo che i dati sono stati processati si trovano nella forma di vettori numerici e saranno dati in input ad un algoritmo di classificazione. Gli algoritmi che sono stati testati sono il Multinominal Naive Bayes, il Linear SVC e il Decision Tree. Ogni algoritmo è stato valutato in base alle metriche ed è stato scelto quello con i valori migliori. Prima di valutare gli algoritmi bisogna individuare i migliori parametri di

configurazione (iperparametri) per i singoli algoritmi. I parametri di configurazione influenzano il modo di apprendere del modello.

Multinominal Naive Bayes

Questo algoritmo [10] organizza la distribuzione delle feature in un vettore $v_y = (v_{y1}, v_{y2}, ..., v_{yn})$ che è calcolato per ogni classe y con n che indica il numero di feature. v_{yi} indica la probabilità di trovare la feature i in un esempio che appartiene alla classe y. Il valore v_{yi} è calcolato come $\frac{n_{yi}+a}{n_y+an}$, dove n_{yi} è il numero di volte che la caratteristica i appare in un documento della classe y del training set, n_y è invece il numero di caratteristiche totali della classe y, infine il parametro a detto alpha è il parametro di smoothing che è >= 0 e previene che le probabilità calcolate abbiano valore pari a 0.

Sarà proprio il parametro alpha quello ad essere configurato, per farlo verrà utilizzata una lista di valori per quel parametro. La lista verrà iterata ed ad ogni iterazione l'elemento i-esimo della lista sarà il parametro alpha. Alla fine di tutte le iterazioni sarà restituito il parametro che ha dato i migliori risultati.

Decision Tree

Questo algoritmo [11] organizza le istanze del dataset all'interno di una struttura ad albero, distribuendole in base alle feature individuate. Anche in questo algoritmo c'è bisogno di individuare la migliore configurazione degli iperparametri. Il modello DecisionTreeClassifier della libreria `scikit-learn` presenta diversi iperparametri. Tra i vari iperparametri sono stati considerati solo alcuni che influenzano maggiormente l'apprendimento del modello. Gli iperparametri presi in considerazione sono il criterior, lo splitter ed il max_depth.

L'iperparametro criterior permette di definire la qualità di una feature quindi definisce il modo in cui una feature divide bene l'insieme dei dati. Tra i possibili valori per questo parametro troviamo "gini" ed "entropy". Di default è impostato "gini". Gini e entropy permettono entrambe di calcolare quanto una caratteristica divide adeguatamente il dataset ma lo fanno in modi differenti. Gini calcola, per ogni sottoinsieme, la probabilità che un campione del dataset possa essere etichettato

in modo errato se gli viene assegnata come etichetta una classe del sottoinsieme in modo casuale. L'entropia calcola il grado di purezza di una caratteristica basandosi sull'entropia dei dati presenti in un sottoinsieme.

Il parametro splitter invece indica la strategia da adottare per la divisione dei nodi. I possibili valori da assegnare a questo parametro sono due ovvero "best" e "random". Se il parametro è impostato a best allora verrà scelto il miglior split, mentre se impostato a random verrà scelto in maniera casuale lo split dei nodi. Infine il parametro max_depth indica la profondità massima che può raggiungere l'albero decisionale. Se questo valore non è impostato allora il limite massimo di profondità non esiste, pertanto l'albero si espanderà fino a quando non raggiunge la situazione in cui tutti i sottoinsiemi sono puri.

LinearSVC

Per quanto riguarda l'algoritmo LinearSVC [12] l'unico parametro considerato, in questo caso, è il parametro C. L'iperparametro C indica all'algoritmo di quanto si vuole evitare di classificare in modo errato ciascun esempio di training. Per valori elevati di C, l'algoritmo sceglierà un iperpiano che classifica correttamente tutte le istanze anche se il margine dell'iperpiano è piccolo. Al contrario, un valore molto piccolo di C farà sì che l'algoritmo cerchi un iperpiano di separazione con margine più ampio, anche se quell'iperpiano classifica erroneamente alcune istanze.

Scelta degli iperparametri

Per la scelta degli iperparametri, si è proceduto con l'implementazione dello script in Python, che permette di scegliere gli iperparametri. Lo script utilizza tre array, uno per ogni classificatore preso in considerazione. Per il classificatore Multinomial Naive Bayes e il LinearSVC l'array contiene dei valori numerici che vanno da 0.1 ad 1 con intervalli di 0.1, quindi un totale di 10 parametri.

Con il Multinominal Naive Bayes è stato valutato l'iperparametro alpha. Per farlo è stato iterato l'array appena descritto. Con il LinearSVC è stato seguito lo stesso approccio ma sull'iperparametro C. Infine per l'albero decisionale è stato scelto di non impostare un valore per il parametro max_depth in modo tale che l'algoritmo

continua ad espandere l'albero fino a che non si ottengono tutti i sottoinsiemi puri, in modo da avere meno incertezza possibile durante la classificazione. I parametri considerati sono criterior e splitter. Entrambi i parametri possono assumere due valori. Per la scelta dei parametri migliori è stata considerata la combinazione di essi, ottenendo quattro test. Per effettuare i quattro test è stato utilizzato un array che contiene tutte le combinazioni, perciò la combinazione "gini" con "best" e con "random" e la combinazione "entropy" con "best" e con "random".

3.4 Balancing dei dati

Dopo la scelta del modello è necessaria una fase di balancing dei dati. Come possiamo vedere nella figura 3.3 i dati sono distribuiti in modo non uniforme nelle classi. La distribuzione del dataset la andremo a ritrovare nel train set, pertanto è utile effettuare il bilanciamento di quest'ultimo. Gli algoritmi presi in considerazione sono stati l'algoritmo SMOTE per aumentare il numero di istanze delle classi di minoranza e il Random Undersampling per eliminare alcune istanze delle classi di maggioranza. La tecnica utilizzata consiste nel valutare separatamente l'algoritmo SMOTE e l'algoritmo Random Undersampling confrontando i risultati ottenuti.

L'algoritmo Random Undersampling utilizza diverse strategie per l'eliminazione delle istanze. Le strategie sono majority, not majority, not minority e all. La strategia majority consiste nell'applicare l'undersampling solo alla classe di maggioranza, pertando il numero delle sue istanze diventa pari a quello della classe di minoranza. La strategia not majority applica l'undersampling a tutte le classi tranne quella di maggioranza. La strategia not minority applica l'undersampling a tutte le classi tranne quella di minoranza. Infine la strategia all applica l'undersampling a tutte le classi.

Per la valutazione di questo algoritmo sono state applicate tutte le strategie. Una volta ottenuti i risultati per ogni strategia, è stata scelta quella che ha restituito il risultato migliore ed è stata confrontata con i risultati dell'algoritmo SMOTE.

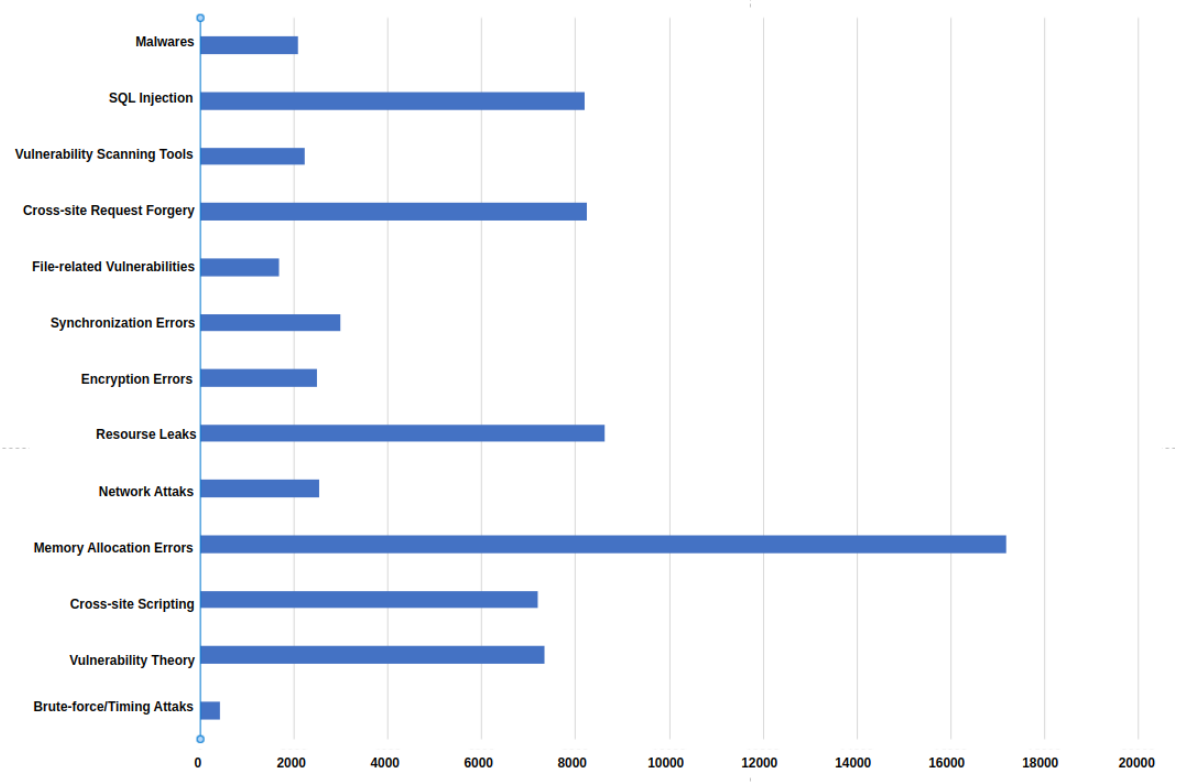

Figura 3.3: Distribuzione delle istanze nelle classi.

La tecnica SMOTE, invece, consiste nell'andare ad inserire nuove istanze che non sono delle copie di quelle esistenti, bensì delle istanze dette sintetiche. Il funzionamento consiste nel selezionare una istanza della classe da ampliare in modo casuale, una volta trovata questa istanza verrà collegata, tramite una linea, alle k istanze più vicine. In seguito verrà scelto un valore z casualmente, che va da 0 ad 1, e per ogni collegamento verrà individuato un punto che si trova ad una distanza del (z*100)% dall'istanza selezionata. Il punto individuato sarà una istanza sintetica.

Per quanto riguarda la valutazione dell'algoritmo SMOTE è stato tenuto in considerazione il parametro k. Per scegliere il migliore valore di questo, è stato applicato l'algoritmo SMOTE con k che varia da 2 a 338. Il valore 338 è il numero di istanze della classe più piccola, presenti nel train set. Per questo motivo è stato scelto come valore massimo da poter assegnare a k. Il valore di k che permette di raggiungere il risultato più alto, sarà quello tenuto in considerazione. La Figura 3.4 mostra un esempio di applicazione dell'algoritmo SMOTE.

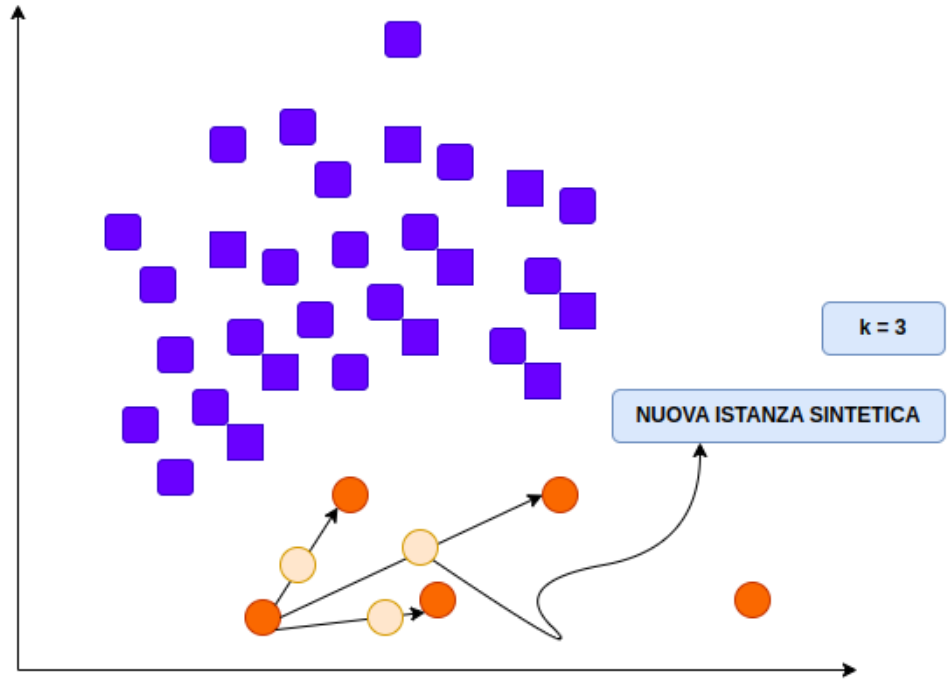

Figura 3.4: Rappresentazione del funzionamento base della tecnica SMOTE.

3.5 Metriche di Performance

Per quanto riguarda la valutazione delle prestazioni del modello, tra le metriche a disposizione è stata scelta la micro F1 score. Questa metrica è stata scelta in quanto tiene in considerazione sia la precision del modello che la recall. La precision misura la capacità del modello di identificare istanze di una determinata classe correttamente. La recall invece permette di misurare la capacità del modello di identificare tutte le istanze di una determinata classe.

Nel caso della classificazione multi classe, per l'F1 score ed in generale anche con le metriche precision e recall, si fa una media che indica il punteggio generale del modello. Queste metriche infatti vengono calcolate per singole classi, pertanto nel contesto multi classe bisogna trovare un modo per unire i risultati.

Esistono due approcci che permettono di calcolare la media del punteggio F1 score. I due approcci per il calcolo del punteggio sono macro F1 score e micro F1 score. Il macro F1 score è ottenuto calcolando l'F1 score per ogni classe e poi viene fatta la media algebrica di tutti i parametri ottenuti. Qui di seguito è mostrata la

formula per il calcolo della macro F1 score:

$$MacroF1score = \frac{\sum_{c \in C} F1score_c}{n}$$

dove C è l'insieme di tutte le classi, $F1score_c$ è il valore F1 score relativo alla classe c ed n indica il numero di classi totali. Questo metodo è semplice e non da peso alle classi, quindi una classe di minoranza può influire significativamente sulla media finale. Per calcolare i valori delle singole classi si utilizza principalmente la modalità one-vs-all.

La modalità one-vs-all consiste nel calcolare i parametri per ogni classe, dove la classe considerata in quel momento è la classe positiva e tutte le altre classi vengono raggruppate in un unica classe negativa, come se fosse una classificazione binaria. Invece il micro F1 score è calcolato con la formula per i classificatori binari solo che viene utilizzato il numero totale di True Positives (TP), False Positives (FP) e False Negatives (FN), anziché calcolare questi valori individualmente per ciascuna classe. Qui di seguito è mostrata la formula per calcolare il micro F1 score:

$$MicroF1score = \frac{2 * \frac{TP}{TP+FP} * \frac{TP}{TP+FN}}{\frac{TP}{TP+FP} + \frac{TP}{TP+FN}}$$

3.6 Strumenti utilizzati

Pandas

Pandas [6] è una libreria open-source che permette di manipolare i dati in modo semplice. Essa viene utilizzata molto in quanto incorpora diverse funzionalità ed è diventata quasi indispensabile per l'analisi di grandi quantità di dati [13].I dati possono provenire, come in questo caso da file csv, ma anche da file Exel o database.

Più in generale Pandas permette di lavorare con dati strutturati organizzandoli in un DataFrame. Un Dataframe è una struttura simile ad una tabella con righe e colonne. Una caratteristica di Pandas è che fornisce metodi utili per la pulizia dei dati mancanti, rimozione dei duplicati ed eliminazione dei dati nel formato errato.

[6] `https://pandas.pydata.org/`

TextBlob

TextBlob [7] è invece un'altra libreria utile per lavorare con il testo. Essa infatti mette a disposizione diverse funzioni di pulizia e per lavorare nell'ambito dell'NLP in generale. Tra le principali funzionalità disponibili troviamo la tokenization del testo, la divisione del testo in n-grammi, lo spelling correction, lo stemming e lemmatization.

Nltk

Infine l'altra libreria utilizzata sempre per la pulizia del testo è nltk (Natural Language Toolkit) [8]. Nel nostro caso è stata utilizzata per rimuovere le stopword dal testo. Essa offre anche altre funzioni ed inoltre permette di scaricare dei vocabolari in diverse lingue utili per identificare le stopword più frequenti.

Scikit-learn

Scikit-learn [9] è una libreria open-source per Python molto utilizzata per il machine learning e il data mining. Questa libreria fornisce degli strumenti ed algoritmi che facilitano lo sviluppo di modelli di machine learning. Tra gli strumenti messi a disposizione troviamo algoritmi per apprendimento supervisionato e non, strumenti per la vettorizzazione dei dati ed inoltre fornisce anche le funzioni per valutare un modello.

[7] https://textblob.readthedocs.io/en/dev/
[8] https://www.nltk.org/
[9] https://scikit-learn.org/stable/

CAPITOLO 4

Analisi dei Risultati

In questo capitolo vengono presentati i risultati ottenuti dall'applicazione della metodologia descritta precedentemente. I risultati da analizzare sono relativi ai modelli di classificazione, alla scelta degli iperparametri migliori e al bilanciamento dei dati.

4.1 Prestazioni dei Modelli

Per ogni modello preso in considerazione, è stato necessario individuare quali sono i valori degli iperparametri che permettono di raggiungere i migliori risultati. Il metodo che è stato seguito è quello illustrato nel capitolo precedente. La valutazione è stata fatta in base al micro F1 score ma non solo, infatti, è stata tenuta in considerazione anche la matrice di confusione, per avere un quadro generale sulle predizioni.

Per il modello Multinominal Naive Bayes è stato ottenuto il valore di micro F1-score pari a 0.78 con il parametro alpha settato a 0.2. Per quanto riguarda il modello LinearSVC, il risultato è leggermente inferiore rispetto il precedente ed è pari a 0.76 ottenuto settando il parametro C al valore 0.1. Infine l'ultimo modello valutato è il DecisionTree. Con questo modello si sono ottenuti dei risultati nettamente inferiori

rispetto ai primi due. I test effettuati utilizzando questo modello sono stati quattro, uno per ogni combinazione dei parametri 'criterior' e 'splitter'. I quattro valori ottenuti sono stati 0.57, 0.56, 0.55 e 0.55. Il valore migliore è pari a 0.57 ed è stato ottenuto dalla combinazione del parametro 'criterior' settato a 'gini' e il parametro 'splitter' settato a 'best'.

Una volta individuati i valori degli iperparametri migliori per ogni modello, si può procedere con il confronto. Per il confronto sono stati considerati i valori di precision, recall, F1-score ottenuti per ogni classe in modalità one-vs-all e la matrice di confusione. La matrice di confusione multi classe è organizzata in una tabella, in cui le righe rappresentano le classi reali, e le colonne rappresentano le classi predette dal modello. Ogni cella della matrice contiene il numero di campioni che sono stati classificati in quella determinata combinazione di classe reale e predetta. Le celle sulla diagonale principale rappresentano le previsioni corrette del modello, mentre le celle fuori dalla diagonale rappresentano gli errori di classificazione.

MULTINOMINAL NAIVE BAYES

PRECISION	0.665	0.839	0.540	0.874	0.572	0.835	0.753	0.765	0.790	0.899	0.781	0.653	0.589
RECALL	0.695	0.804	0.443	0.865	0.443	0.842	0.665	0.905	0.725	0.871	0.780	0.719	0.333
F1-SCORE	0.680	0.821	0.487	0.869	0.500	0.839	0.707	0.829	0.756	0.885	0.781	0.684	0.425

LINEAR SVC

PRECISION	0.623	0.808	0.516	0.858	0.460	0.822	0.639	0.817	0.708	0.880	0.764	0.642	0.510
RECALL	0.658	0.804	0.505	0.850	0.504	0.861	0.631	0.852	0.723	0.862	0.762	0.618	0.494
F1-SCORE	0.640	0.806	0.510	0.854	0.481	0.841	0.635	0.834	0.716	0.871	0.763	0.630	0.502

DECISION TREE

PRECISION	0.340	0.607	0.175	0.719	0.154	0.654	0.320	0.690	0.438	0.793	0.513	0.352	0.149
RECALL	0.402	0.612	0.273	0.657	0.292	0.767	0.432	0.648	0.468	0.687	0.479	0.307	0.222
F1-SCORE	0.369	0.609	0.214	0.687	0.202	0.706	0.368	0.668	0.453	0.736	0.496	0.328	0.178

Figura 4.1: Risultati di precision, recall ed F1 score, ottenuti in modalità one-vs-all, di tutti e tre i modelli di classificazione.

Nella Figura 4.1 sono mostrate le metriche per ogni classe di precision, recall e F1

score ottenute in modalità one-vs-all con tutti e tre i modelli. Analizzando prima i valori ottenuti con il modello Multinominal Naive Bayes, si può notare che in generale la precision è buona. Ciò significa che se ad una istanza viene assegnata una classe di appartenenza, con buona probabilità, quella sarà la sua classe effettiva. Gli unici valori di precision poco al di sotto della sufficienza sono quello relativo alla classe Vulnerabilities Scanning Tools che è pari a 0.54, alla classe File-related Vulnerabilities che è pari a 0.57 e alla classe Brute-orce/Timing Attacks con valore di 0.58. Per quanto riguarda la recall, anche qui, i risultati più bassi sono in corrispondenza delle classi Vulnerability Scanning Tools, File-related Vulnerabilities e Brute-force/Timing Attacks dove i valori di recall sono rispettivamente 0.443, 0.443 e 0.333.

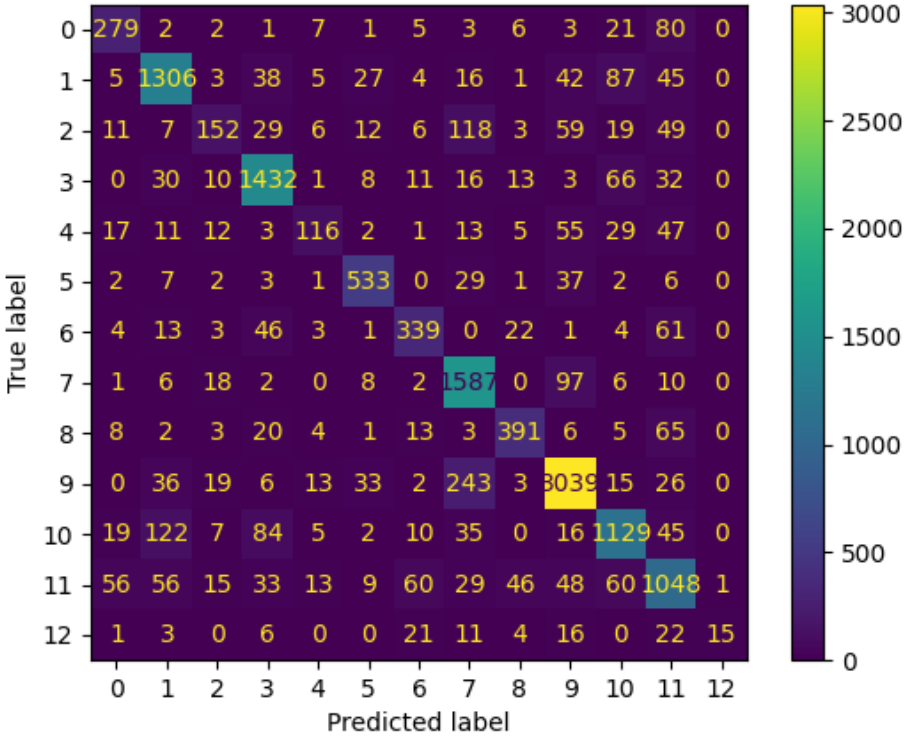

Figura 4.2: Matrice di confusione relativa al modello Multinominal Naive Bayes.

Ciò significa che di tutte le istanze appartenenti effettivamente alla classe Vulnerability Scanning Tools e File-related Vulnerabilities circa il 44% delle istanze vengono individuate come appartenenti alla loro classe, mentre per quanto riguarda la classe Brute-force/Timing Attacks meno del 35% delle istanze vengono individuate. La

classe Brute-force/Timing Attacks presenta molti errori, per capire il motivo di questi errori è utile analizzare la matrice di confusione.

Nella matrice di confusione in Figura 4.2 si nota che più del 20% delle istanze appartenenti alla classe Brute-force/Timing Attacks viene classificato come appartenente alla classe Encryption Errors. Le frasi *'i heard that we shouldn t rely on adler32 and i want to ask why why shouldn t we trust adler32 to hash ? is it reversible ? or can we just see the real text with ease'* e *'i am familiar with how offline brute-force attacks work but for online accounts assuming no social engineering how feasible is it to brute-force attack a password? for example is this dependent upon password complexity or possibly some other vulnerability like eavesdropping on ssl/tls handshakes'* sono due esempi di istanze appartenenti alla classe Brute-force/Timing Attacks ma vengono classificate come appartenenti alla classe Encryption Errors, invece la frase *'since gmail uses ssl the entire request password and all is encrypted by the browser that being said the question is then how am i able to see the plaintext when using zap? zaproxy intercepts the ssl handshake and establishes it s own connection with gmail it then uses a different certificate to communicate with your browser this allows you to view the request/response in plaintext because proxy has access to it'* è un esempio di frase appartenente alla classe Encryption Errors. Considerando queste tre frasi possiamo notare che contengono alcune parole ricorrenti come 'password' e 'ssl'. Questo le rende molto simili tra di loro e quindi è facile che una istanza di una classe venga scambiata con l'altra. Il motivo di questa somiglianza è dovuto al fatto che entrambe le classi appartengono allo stesso macro argomento ovvero la Crittografia.

Per quanto riguarda invece il modello LinearSVC, i risultati sono molto simili a quelli del modello Multinominal Naive Bayes per quasi tutte le classi. Considerando i valori della precision solo la classe File-related Vulnerabilities presenta un valore molto basso, inferiore a 0.5 pertanto possiamo affermare che con questo modello se una istanza è classificata come appartenente a questa classe solo nel 46% dei casi sarà una predizione corretta.

Considerando i valori di recall sono sempre le stesse tre classi che presentano degli score non sufficienti, come accade nel modello Multinominal Naive Bayes. Nella Figura 4.3 è rappresentata la matrice di confusione relativa all'algoritmo LinearSVC, che confrontata con quella del modello Multinominal Naive Bayes si può notare un leggero aumento degli errori in quasi tutte le classi.

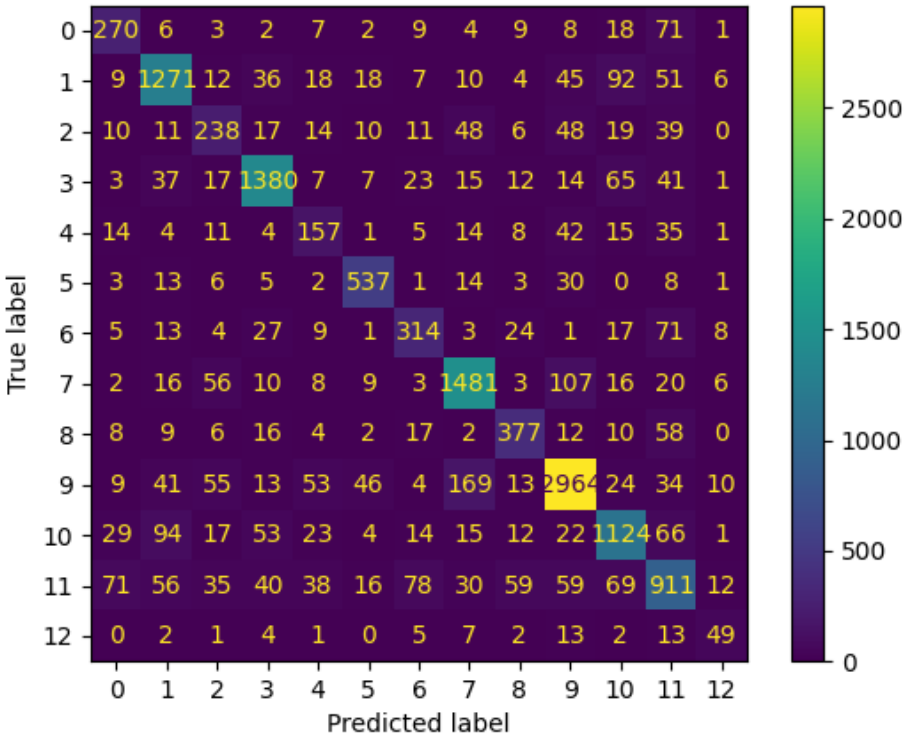

Figura 4.3: Matrice di confusione relativa al modello LinearSVC.

Infine l'ultimo modello da valutare è il Decision Tree. Questo modello, come si vede nella Figura 4.1, ha raggiunto dei risultati decisamente più bassi rispetti gli altri due. Per quanto riguarda la precision sette classi su tredici hanno un valore minore o uguale a 0.438. I valori della recall sono altrettanto bassi, infatti, anche qui le stesse sette classi presentano un valore inferiore o uguale a 0.468. Questi risultati indicano che il modello trova fatica nell'individuare le classi di appartenenza delle istanze ed inoltre le predizioni non sono affidabili. Nella Figura 4.4 è rappresentata la matrice di confusione ottenuta utilizzando il modello Decision Tree.

👉 **Answer to RQ$_1$.** Il modello Multinominal Naive Bayes è quello che ha ottenuto i risultati migliori in termini di micro F1-score, di precision e recall. Il valore del micro F1 score è pari a 0.78. Il numero di errori per ogni classe è minore rispetto quelli degli altri modelli come si è visto anche dalla matrice di confusione.

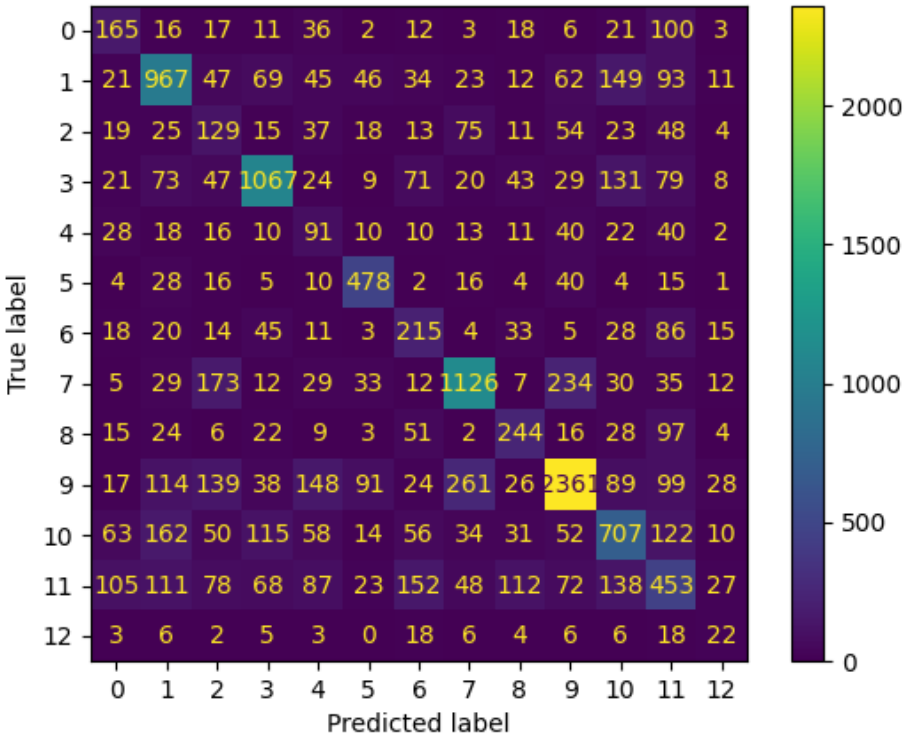

Figura 4.4: Matrice di confusione relativa al modello Decision Tree.

4.2 Prestazioni dei Balancer

A seguito della valutazione dei modelli, è stata effettuata la valutazione delle tecniche di bilanciamento del train set. Gli algoritmi utilizzati per il bilanciamento sono lo SMOTE e il Random Undersampling. Gli algoritmo sono stati applicati con il modello Multinominal Naive Bayes, in quanto considerato il più adatto.

Per quanto riguarda l'algoritmo SMOTE è stato necessario individuare il miglior valore del parametro k. I risultati sono stati valutati in base al micro F1 score. Si è notato che questo aumenta leggermente, con l'aumentare di k. Infatti il miglior valore è stato ottenuto con k settato a 338. In termini di micro F1 score non c'è stato un grande miglioramento, ma guardando le matrici di confusione, si possono notare alcune differenze.

Nella Figura 4.5 viene mostrata sulla sinistra la matrice di confusione relativa al modello Multinominal Naive Bayes, mentre sulla destra è mostrata la matrice di

confusione del modello Multinominal Naive Bayes addestrato sul dataset bilanciato attraverso l'algoritmo SMOTE. Le differenze principali si possono notare nelle classi di minoranza, dove il numero di predizioni corrette è aumentato.

Prendiamo in considerazione la classe Brute-force/Timing Attaks, senza il bilanciamento vengono predette in maniera corretta soltanto il 15% delle istanze, mentre con il train set bilanciato vengono predette in maniera corretta più del 30% delle istanze. Lo stesso vale per le altre classi di minoranza come Malwares, Vulnerability Scanning Tools e File-related Vulnerabilities, dove si può notare un aumento delle istanze predette correttamente, e la conseguente diminuzione delle istanze predette in modo errato.

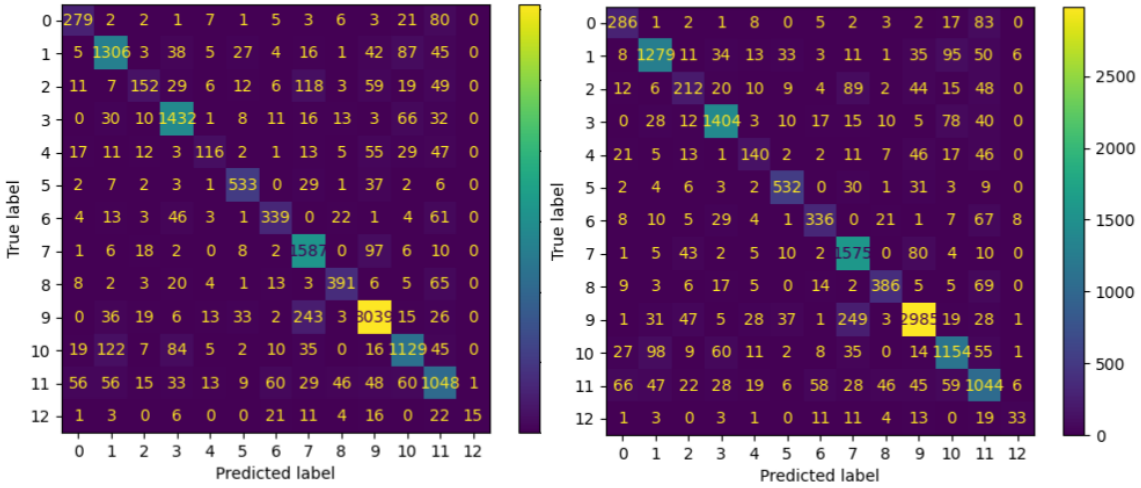

Figura 4.5: Matrice di confusione relativa al modello Multinominal Naive Bayes (a sinistra) e matrice di confusione relativa al modello Multinominal Naive Bayes addestrato sul train set bilanciato (a destra).

Per quanto riguarda l'altra tecnica per bilanciare il train set, ovvero il Random Undersampling, sono stati ottenuti dei valori nettamente inferiori. Con la strategia majority è stato ottenuto un valore di micro F1 score pari a 0.62, con la strategia not majority lo score ottenuto è pari a 0.57, mentre per la strategia not minority e all lo score ottenuto è stato di 0.71. Il migliore punteggio è stato ottenuto con le strategie all e not minority. Queste, a differenza delle altre due strategie, ovvero majority e not majority, permettono di ottenere un train set più equilibrato.

👉 **Answer to RQ_2.** Per rispondere alla seconda domanda è stato tenuto inconsiderazione il miglior risultato ottenuto dall'algoritmo SMOTE e il migliore risultato ottenuto dall'algoritmo Random Undersampling. Per l'algoritmo SMOTE, il valore più alto in termini di micro F1 scotre è di 0.80. Questo punteggio è stato raggiunto settando il parametro k a 338. Per il Random Undersampling invece il valore migliore è stato ottenuto con la strategia all e not minority con valore di 0.71. Da questi risultati possiamo dire che l'algoritmo migliore per il balancing dei dati è stato lo SMOTE. In generale è comunque da preferire l'oversampling rispetto l'undersampling per bilanciare il train set [14]. Il motivo di questa preferenza stà nel fatto che l'undersampling presenta il problema legato all'eliminazione delle istanze.

CAPITOLO 5

Conclusioni

Lo scopo di questa ricerca è stato quello di realizzare un classificatore di testi di domande relative all'ambito della sicurezza informatica. La realizzazione del classificatore è avvenuta in diverse fasi. Si è partiti dalla ricerca dei dati su cui lavorare e quindi del dataset. La fase successiva è stata quella della pulizia dei dati e della vettorizzazione, che ha permesso di convertire il testo in dati numerici. Poi è stato scelto il modello ed è stata effettuato il balancing dei dati di training. Infine ci sono state le fasi di valutazione del modello e l'analisi dei risultati ottenuti.

La fase più corposa è stata quella della pulizia dei dati, dove è stata implementata la pipeline di pulizia. Le fasi più importanti e che hanno richiesto più tempo, invece, sono state quelle della scelta del modello e del balancing dei dati, infatti le domande di ricerca che ci siamo posti riguardano proprio queste due fasi.

A seguito della realizzazione del classificatore è stato possibile rispondere alle domande che ci siamo posti. I risultati dimostrano che il modello che si adatta meglio alla classificazione delle domande , presenti nel dataset, è il Multinominal Naive Bayes, mentre il risultato sulla tecnica di bilanciamento mostra che è da preferire l'oversampling rispetto l'undersampling.

Attraverso questo lavoro è stato possibile aggiungere alcune considerazioni sui modelli di classificazione Multinominal Naive Bayes, LinearSVC e DecisionTree ed

in particolare quali tra questi si adatta meglio al task della classificazione dei testi. Questo lavoro ha messo inoltre evidenza il fatto che l'uso di tecniche di oversampling permette di raggiungere dei risultati migliori, in generale, rispetto l'uso di tecniche di undersampling.

Bibliografia

[1] T. H. M. Le, C. Roland, D. Hin, and M. A. Babar, "A large-scale study of security vulnerability support on developer q&a websites," 2021. (Citato alle pagine 2, 20 e 24)

[2] N. Meng, S. Nagy, D. Yao, W. Zhuang, and G. A. Argoty, "Secure coding practices in java: Challenges and vulnerabilities," 2018. (Citato alle pagine 2 e 20)

[3] M. K. Dalal and M. A. Zaveri, "Automatic text classification: A technical review." (Citato a pagina 6)

[4] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification." (Citato a pagina 9)

[5] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview." (Citato a pagina 19)

[6] F. Fischer, K. Bottinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack overflow considered harmful? the impact of copy & paste on android application sceurity," 2017. (Citato a pagina 21)

[7] S. M. H. Hasheminejad and S. Jalili, "Selecting proper security patterns using text classification." (Citato a pagina 21)

[8] H. Qin and S. Xin, "Classifying bug reports into bugs and non-bugs using lstm." (Citato a pagina 22)

[9] W. A. Qader, A. M. M., and B. I. Ahmed, "An overview of bag of words;importance, implementation, applications, and challenges." (Citato a pagina 28)

[10] A. V. Ratz, "Multinomial naive bayes for documents classification and natural language processing (nlp)," https://towardsdatascience.com/multinomial-na%C3%AFve-bayes-for-documents-classification-and-natural-language-processing-nlp-e08cc848ce6. (Citato a pagina 30)

[11] J. R. QUINLAN, "Learning decision tree classifiers." (Citato a pagina 30)

[12] C. CORTES and V. VAPNIK, "Support-vector networks." (Citato a pagina 31)

[13] W. McKinney, "pandas: a foundational python library for data analysis and statistics." (Citato a pagina 35)

[14] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," 2004. (Citato a pagina 44)

Ringraziamenti

Fin dal primo giorno di università ho aspettato con ansia che arrivasse questo momento. Durante i tre anni ed in particolare gli ultimi mesi ci sono state delle persone molto vicine e che vorrei ringraziare. Inizio ringraziando il Prof. Fabio Palomba e il Dott. Emanuele Iannone per avermi seguito in questo periodo e per essere stati sempre disponibili per eventuali chiarimenti. Ringrazio i miei compagni di classe, in particolare Salvatore e Christian. Con Salvatore ci conosciamo dai tempi del liceo, mentre con Christian ci siamo conosciuti qui all'università, ed è stato bello aver potuto condividere gli studi insieme a loro. Ringrazio gli amici di Brienza ovvero Gianmario, Gianluigi, Giulia, Giovanni, Teresa, Michele, Gianfranco, Maria Teresa e Teresa Palladino. Ringrazio gli amici Gioacchino, Grieco, di nuovo Salvatore, Manuel, Mara, Davide, Peppe Rocco e Vincenzo per i bei momenti e le risate che abbiamo condiviso assieme. In particolare Gioacchino e Salvatore per tutte le giornate passate a scherzare e per i pomeriggi di studio e Grieco per aver trovato sempre un modo di passare una giornata in maniera diversa. Poi ci sono Bonelli, Antonio ed Enrico che sono le persone con cui ho passato la maggior parte del tempo, vivendo nella stessa casa un pò come una famiglia. Con Bonelli ci siamo trovati d'accordo fin dall'inizio, quasi come se fossimo amici da una vita. Antonio è un amico di lunga data, con cui abbiamo condiviso molte esperienze e vivere nella stessa casa non ha fatto altro che consolidare il nostro rapporto. Enrico l'ho conosciuto solo nell'ultimo anno, ma non è mancato il tempo per instaurare una bella amicizia. Ringrazio poi tutti i miei

familiari per essermi stati sempre vicini. In particolare i nonni, per il loro affetto, Zio Rocco, un pò mi sono rivisto in lui quando tanti anni fa veniva proprio in questa università, e Zia Laura per la sua costante presenza. Infine concludo ringraziando le persone più importanti della mia vita ovvero Ismaele, Giovanni, mamma e papà. Non le ringrazierò mai abbastanza per quello che fanno ed il bene che mi vogliono. A Giovanni perché, anche se a modo suo, è un buon fratello, abbiamo condiviso tutto insieme. Ad Ismaele, il mio fratellino, per tutte le sere che mi ha tenuto compagnia con le sue chiamate e per rendere il mio rientro a casa sempre speciale. A mamma e papà, per avermi reso la persona che sono, per essere presenti in ogni momento, per il loro sostegno, per tutte le parole di conforto e per tutti i loro sacrifici. Un grazie di cuore.

Questa tesi ha contribuito a piantare un albero in Kenya tramite il progetto Treedom.

`https://www.treedom.net/it/user/sesalab/event/sesa-random-forest`