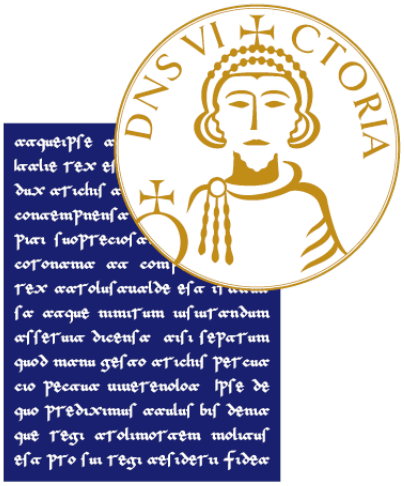
UNIVERSITÀ DEGLI STUDI DEL SANNIO
DIPARTIMENTO DI INGEGNERIA



CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

UNO STUDIO SULLE FASI DI UN MACHINE LEARNING WORKFLOW E UTILIZZO DI LIBRERIE

RELATORE:
CHIAR.MO PROF.
MASSIMILIANO DI PENTA

CORELATORE:
DOTT.SSA
VITTORIA NARDONE

CANDIDATO:
RAFFAELE MIGNONE
399000298

ANNO ACCADEMICO 2020-2021

[5 giugno 2021 at 18:25 –]

INDICE

LISTA DELLE FIGURE

ACRONIMI

API Application Program Interface
ML Machine Learning
VPS Virtual Private Server

1 COLLEZIONE DEI DATI E ANALISI PRELIMINARE

1.1 SELEZIONE DEI PROGETTI

L'individuazione dei progetti da analizzare è avvenuta mediate l'ausilio dell'Application Program Interface (API) messa a disposizione da GitHub. In particolare è stata eseguita una query per ottenere una lista di repository che fanno uso di librerie e framework di Machine Learning (ML) come `TensorFlow`, `Pytorch` e `scikit-learn`. In questo modo è stato possibile ottenere una lista di 26758 repository che è stata successivamente filtrata per individuare solo i progetti d'interesse per la seguente analisi.

L'operazione di filtraggio è avvenuta attraverso due fasi; una prima automatica e una seconda manuale. La prima fase ha avuto l'obiettivo di selezionare unicamente i repository *popolari*. Nella maggior parte dei casi viene utilizzato il numero di stelle come indice della popolarità di un progetto [1], ma per questo lavoro si è preferito dare maggiore rilevanza ad altri aspetti, come il numero di fork, il numero di *contributors* e il numero di issues chiuse. Questa scelta è stata dettata dall'esigenza di selezionare non solo repository popolari, ma anche caratterizzati da una forte partecipazione della community.

I progetti che hanno superato questa prima selezione dovevano:

- essere lavori originali, per cui sono stati esclusi tutti i fork.
- avere almeno cento issues chiuse.
- avere almeno dieci contributors.
- avere almeno venticinque fork.

Alla fine di questa prima selezione il numero di repository si è ridotto a sessantasei e sono stati analizzati manualmente per rimuovere listati associati a libri e/o tutorial, progetti non in lingua inglese e librerie. Alla fine di questa seconda fase il numero di progetti è sceso a trenta.

1.2 FETCH DI ISSUES E COMMIT

Una volta individuati i progetti da analizzare si è reso necessario recuperare l'intera storia dei progetti e le issues ad essi associate. Per entrambe le operazioni è stato utilizzato il tool *perceval*[2]. Nel caso delle issues, essendo queste informazioni non direttamente contenute

all'interno del repository `git`, è stato necessario utilizzare nuovamente l'API di GitHub. Poiché le chiamate associate ad un singolo *token* sono limitate nel tempo si è scelto di configurare *perseval* in modo tale da introdurre in automatico uno ritardo ogni qualvolta veniva raggiunto il limite. Inoltre il codice è stato dispiegato su un Virtual Private Server (VPS) in modo da poter eseguire il fetch senza che fosse necessario mantenere attiva una macchina fisica.

Con il processo precedentemente illustrato è stato possibile recuperare:

- 34180 commit.
- 15267 tra issues e pull request.

1.3 CLASSIFICAZIONE DEI DATI

1.3.1 *Classificazione delle issues*

Al fine di poter eseguire un confronto tra i *fix* di ML e quelli *generici* è stato necessario classificare sia le issues che i commit. Per quanto riguarda i primi si è scelto di attuare una classificazione basata sul testo, in particolare considerando il titolo e il corpo della issue, ma escludendo i commenti di risposta in modo da non rendere i dati troppo rumorosi. Il numero elevato di elementi non rende praticabile una classificazione manuale per cui si è optato per una classificazione automatica. A tal fine sono stati implementati ed analizzati due classificatori, uno supervisionato e uno non supervisionato.

I due modelli considerati sono:

- un classificatore statico basato su una lista di vocaboli tipici del ML.
- un modello *naïve Bayes* [3], [4].

La classificazione mediate il classificatore statico non necessita di un *labeling* manuale dei dati, ma richiede la definizione dei vocaboli tipici del ML. Questa lista non è stata costruita da zero, ma è basata su lavori precedenti[5]. In questo modo tutte le issues che utilizzavano almeno un vocabolo tipico del Machine Learning sono state classificate come issues di ML.

Nel caso del modello *naïve Bayes*, essendo questo un algoritmo di apprendimento supervisionato, si è resa necessaria una classificazione manuale delle issues. A tal scopo è stato eseguito un campionamento stratificato in base al progetto di provenienza di 376 issues che sono state divise tra due lettori e labellate. Durante il labeling si scelto di classificare ulteriormente le issue di ML al fine di individuare anche

la fase in cui il problema si è palesato. La definizioni delle varie fasi è avvenuta partendo da un lavoro di *Microsoft*[6].

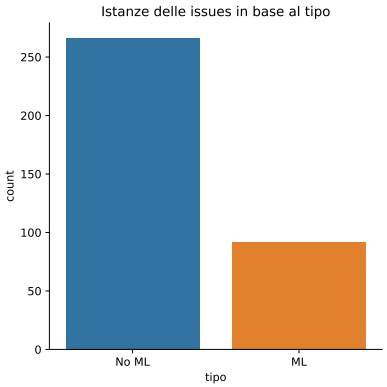
Le fasi considerate sono:

- *Model Requirements*: questa fase comprende tutte le discussioni rispetto all'individuazione del modello più adatto, le funzionalità che questo deve esporre e come adattare un modello esistente per eseguire una diversa funzionalità.
- *Data Collection*: comprende le operazioni volte alla definizione di un dataset. Rientrano in questa fase sia la ricerca di dataset già esistenti che la costruzione di nuovi dataset.
- *Data Labeling*: questa fase si rende necessaria ogni qual volta si opera con modelli basati su apprendimento supervisionato.
- *Data cleaning*: in questa fase non rientrano soltanto le operazioni strettamente di pulizia dei dati come ad esempio rimozione di record rumorosi o incompleti, ma tutte le trasformazioni eseguite sui dati, quindi anche operazioni di standardizzazione, flip di immagini ecc.
- *Feature Engineering*: questa fase serve per identificare le trasformazioni da attuare sui dati e le migliori configurazioni degli *hyperparametri* al fine di migliorare il modello.
- *Model Training*: questa fase racchiude il training vero e proprio del modello.
- *Model Evaluation*: in questa fase vengono valutate le performance del modello utilizzando metriche standard come *precision* e *recall*, ma anche andando a confrontare i risultati ottenuti rispetto a quelli generati da altri modelli o rispetto all'esperienza[1].
- *Model Deployment*: questa fase riguarda il dispiegamento del modello sul dispositivo target.
- *Model Monitoring*: una volta dispiegato il modello deve essere continuamente monitora al fini di assicurasi un corretto comportamento anche sui dati reali.

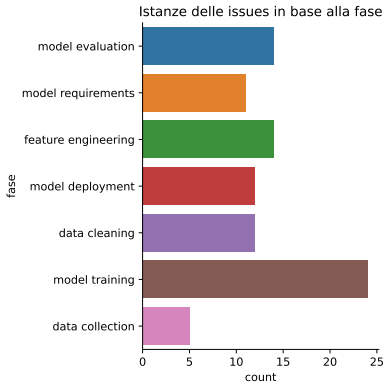
A partire dal dataset *labellato* è stato possibile costruire un training e un test set, mediante i quali è stato possibile allenare e valutare le performance del modello bayesiano. Mentre le performance del primo modello sono state valutate sull'intero dataset.

Al fine di poter confrontare i due modelli sono state utilizzate le metriche di precision e recall. Com'è possibile notare dai valori riportati in tbl. 1.1, il modello basato sulla lista di vocaboli è leggermente più preciso del modello bayesiano, ma presenta una recall decisamente più bassa. Dalla fig. 1.1a si evince la natura minoritaria delle issues di

1 Non sempre è possibile valutare un modello in modo oggettivo, ci sono determinati contesti, come ad esempio la generazione di *deep fakes*, in cui è comunque necessaria una valutazione umana per determinare la qualità del risultato.



(a) Numero di issues rispetto al tipo



(b) Numero di issues rispetto alla fase

Figura 1.1: Risultati della classificazione manuale delle issues

ML rispetto alle issues generiche, per questo motivo si è scelto di preferire il modello naïve Bayes in modo da perdere quante meno istanze possibili anche a costo di sacrificare leggermente la precisione.

Tabella 1.1: Confronto dei due modelli per la classificazione delle issues.

	Classificatore statico	naïve Bayes
precision	0.46	0.41
recall	0.74	0.94

1.3.2 *Classificazione dei commit*

Prima di poter classificare i commit si è reso necessaria un'ulteriore fase di filtraggio in modo da poter separare i commit di *issue fixing* da quelli generici. Sono stati considerati come commit di *fix* tutti quei commit al cui interno veniva fatto riferimento a delle issues attraverso la notazione "#". Questa operazione ha ridotto il dataset dei commit a 3321 unità la cui distribuzione in base al tipo è riportata in fig. 1.2.

A questo punto è stato possibile separare i *fix* di Machine Learning da quelli generici. La classificazione è avvenuta attraverso la lista delle issues citate all'interno del *commit message* e sono stati considerati come commit di ML tutti quei commit che facevano riferimento ad almeno una issue di ML.

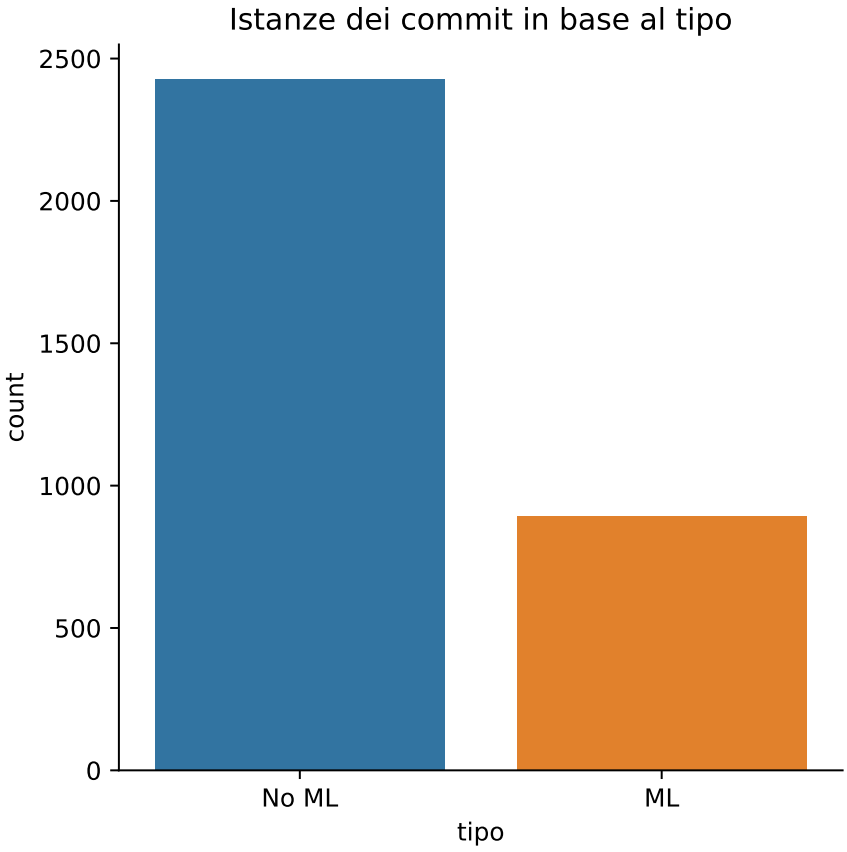


Figura 1.2: Risultato della classificazione dei commit

BIBLIOGRAFIA

[1] H. Borges, A. Hora, e M. T. Valente, «Understanding the Factors That Impact the Popularity of GitHub Repositories», *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pagg. 334–344, ott. 2016, doi: 10.1109/ICSME.2016.31.

[2] S. Dueñas, V. Cosentino, G. Robles, e J. M. Gonzalez-Barahona, «Perceval: Software Project Data at Your Will», in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, mag. 2018, pagg. 1–4, doi: 10.1145/3183440.3183475.

[3] «Naive Bayes Classifier», *Wikipedia*. mag. 21, 2021, Consultato: giu. 03, 2021. [Online]. Disponibile su: https://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&oldid=1024247473.

[4] P. Harrington, *Machine Learning in Action*. Manning Publications, 2012.

[5] N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, e P. Tonella, «Taxonomy of Real Faults in Deep Learning Systems», nov. 07, 2019. http://arxiv.org/abs/1910.11015 (consultato mar. 17, 2021).

[6] S. Amershi *et al.*, «Software Engineering for Machine Learning: A Case Study», in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, mag. 2019, pagg. 291–300, doi: 10.1109/ICSE-SEIP.2019.00042.