PROGETTO ISW2

Machine Learning for S.E.

TIZIANO TAGLIENTI
MATRICOLA 304926

INDICE

Introduzione.....1

Progettazione.....3

Risultati.....10

Conclusioni.....17

Threats.....19

Links.....20

Introduzione

Scopo

Il progetto punta a utilizzare strumenti di machine learning per predire se una classe è buggy.
Vengono usati e confrontati più modelli sulla base di metriche di qualità, per poi scegliere il migliore da applicare nel contesto del progetto scelto.

Contesto

L'applicativo è stato sviluppato in linguaggio Java (IntelliJ), interagendo con *Weka* e le API di *Jira* e *Git*.

Introduzione

Scopo

Si esegue uno studio finalizzato a misurare l'effetto di tecniche di:

- Sampling (oversampling, undersampling, *SMOTE*);
- Feature selection (no selection, best first);
- Cost sensitive classification (no cost sensitive, sensitive threshold, sensitive learning).

Tutto ciò viene fatto utilizzando la tecnica di valutazione *walk forward* e i classificatori *RandomForest*, *NaiveBayes* e *IBk*.

Progettazione

Il lavoro è stato diviso in due. Il primo mira a costruire un file CSV che, per ogni file e ogni versione, mostra alcune metriche scelte e la bugginess di quel file. Con il secondo si costruisce un altro csv che riporta metriche riguardo la parte di machine learning.

I progetti analizzati sono *Apache Bookkeeper* e *Apache Storm*.
Il primo è un sistema di storage distribuito per la gestione affidabile dei dati di log, mentre il secondo è un framework di elaborazione distribuita di dati in modo scalabile.

Versioni

Per risalire alla lista delle versioni dei progetti considerati è stata utilizzata l'API di *Jira*, attraverso la richiesta al seguente url:

```
https://issues.apache.org/jira/rest/api/2/project/"+this.projectName+"/versions
```

Da qui sono stati estratti il nome e la data di rilascio per ogni versione, creando una lista di versioni ordinata temporalmente.

Successivamente sono state dimezzate le release, perché si ritiene che gli ultimi dati siano i meno puri (per effetto dello **snoring**).

RELEASE USATE	RELEASE SCARTATE

Il prossimo step è quello di ottenere i bug relativi ai progetti, per ognuno dei quali si considerano:

- **IV**: versione in cui il bug è stato inserito
- **OV**: versione che ha aperto il ticket per la failure
- **FV**: prima versione esente dal bug

Bug

Anche la lista di bug dei progetti è stata ottenuta tramite l'API di *Jira*, andando poi a scartare i bug che non hanno FV o OV, quelli che non rispettano FV > OV e OV > IV.
Si eliminano anche quelli con IV = OV = FV perché rendono impossibile l'applicazione della tecnica **proportion**.

Proportion

Non tutti i bug possiedono la lista delle cosiddette *affected versions*.
Ciò rende impossibile etichettare alcune classi come buggy.

Questa tecnica, utilizzata nella sua variante incrementale, aiuta a ovviare al problema, trovando una correlazione tra IV, OV e FV.
Si calcola il valore di P e si estrae IV.

$$P = \frac{FV - IV}{FV - OV}$$

$$IV = FV - (FV - OV) * P$$

Commit

Per l'integrazione con *Git* è stato importato *JGit* nel progetto.
Durante la creazione dei dati, si eseguono iterazioni sui commit del repository *Git* per analizzare le modifiche associate, per poi iterare su queste stesse modifiche e calcolare le metriche associate ai file Java modificati.

Metriche

Per la costruzione del dataset, viene popolato un file csv estraendo le seguenti metriche:

Version Name	File Name	LOC Touched	#Revs	#Bug Fix	LOC Added	Chg Set Size	Max Chg Set	Avg ChgSet	Buggy

Header
del dataset

Version Name	
File Name	
LOC Touched	Linee di codice aggiunte, cancellate o modificate in tutti i commit di una release.
#Revs	Numero di commit in una release.
#Bug Fix	Numero di bug ticket committati.

Progettazione

LOC Added	Linee di codice aggiunte in un commit.
Chg Set Size	Numero di file committati nel commit corrente.
Max Chg Set	Massimo numero di file committati in un commit.
Avg ChgSet	Numero medio di file committati in un commit.
Buggy	

Weka

Una volta costruito, il dataset viene diviso sulla base del numero di versione, per poi salvare tutti i file in formato CSV e ARFF; i dati in questo formato vengono poi passati al tool di *Weka* per la valutazione.

Al variare della configurazione di {balancing, feature selection, cost sensitive selection}, si valuta l'accuratezza dei classificatori esaminati. Per ogni configurazione vengono eseguite N-1 run di walk forward, dove N è il numero delle release, con lo scopo di creare un file CSV finale, contenente le seguenti metriche di machine learning:

TP	FP	TN	FN	Precision	Recall	AUC	Kappa

Parte dell'header del dataset

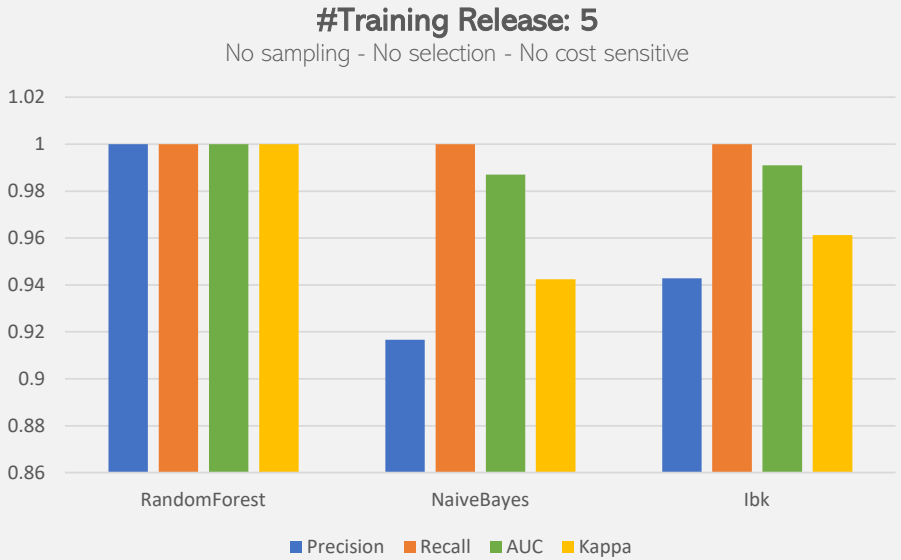
Risultati

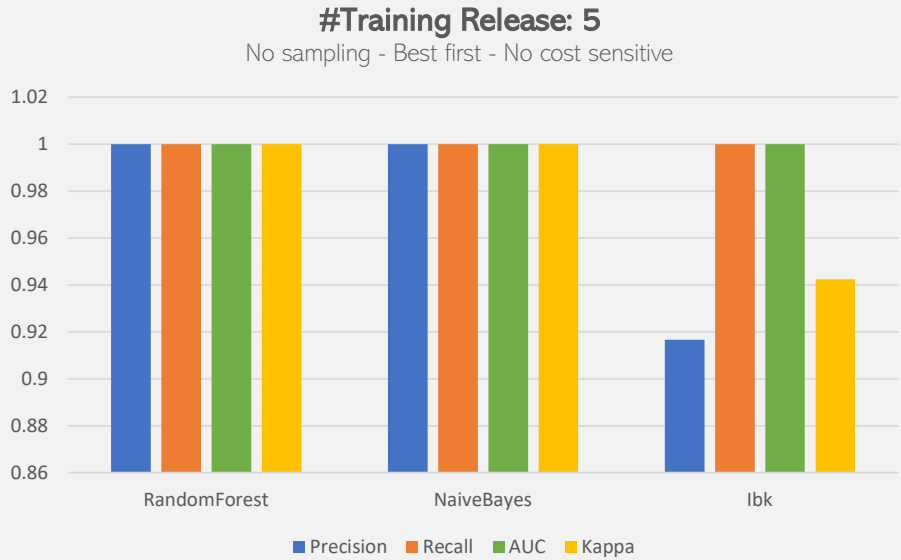
I classificatori che verranno comparati sono RandomForest, NaiveBayes e Ibk.

Essi sono stati valutati variando le configurazioni di:
Balancing = {no sampling, smote, oversampling, undersampling};
Feature selection = {no selection, best first};
Cost sensitivity = {no cost sensitive, sensitive learning, sensitive threshold}.

Come già detto, il metodo walk forward è stato applicato come tecnica di validazione.

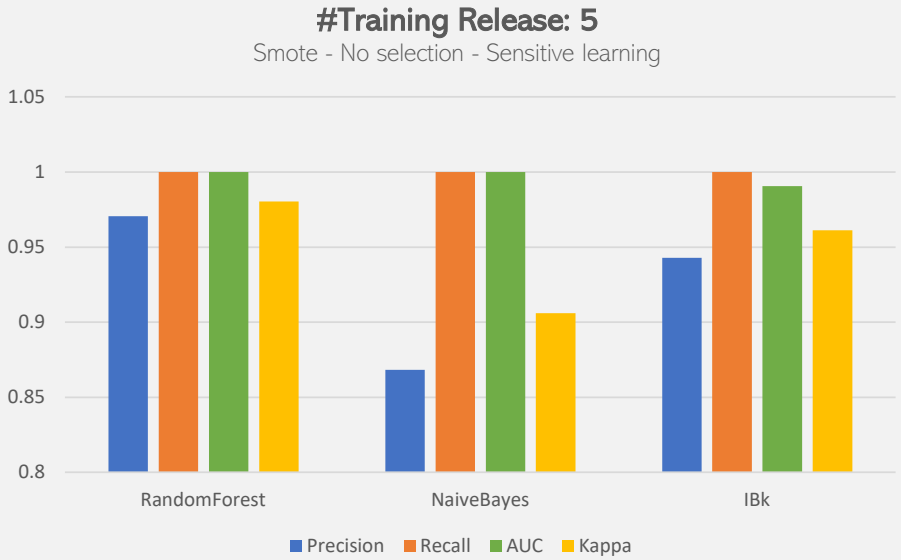
Risultati - Bookkeeper

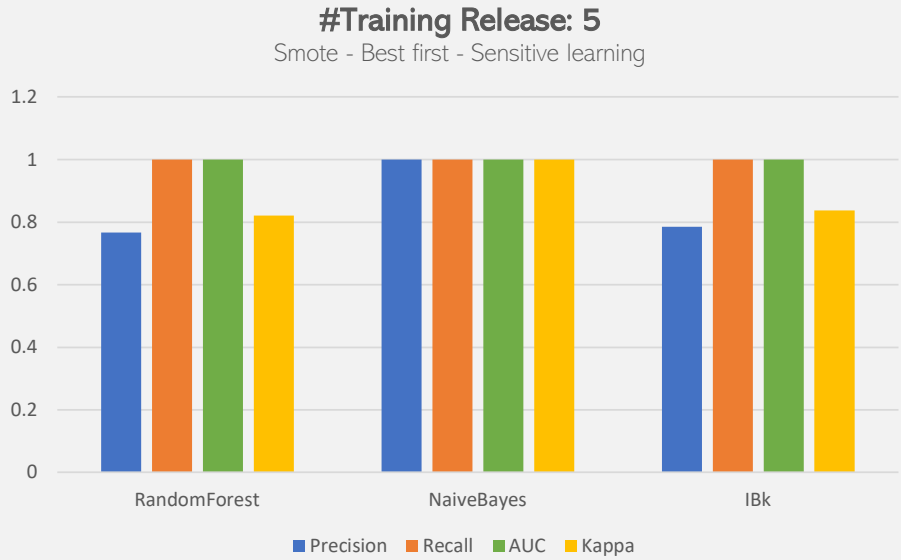




Facendo feature selection con Best first migliorano tutte le metriche per Naive Bayes e peggiorano quasi tutte tranne la AUC per Ibk.
La predizione non cambia utilizzando il classificatore RandomForest.

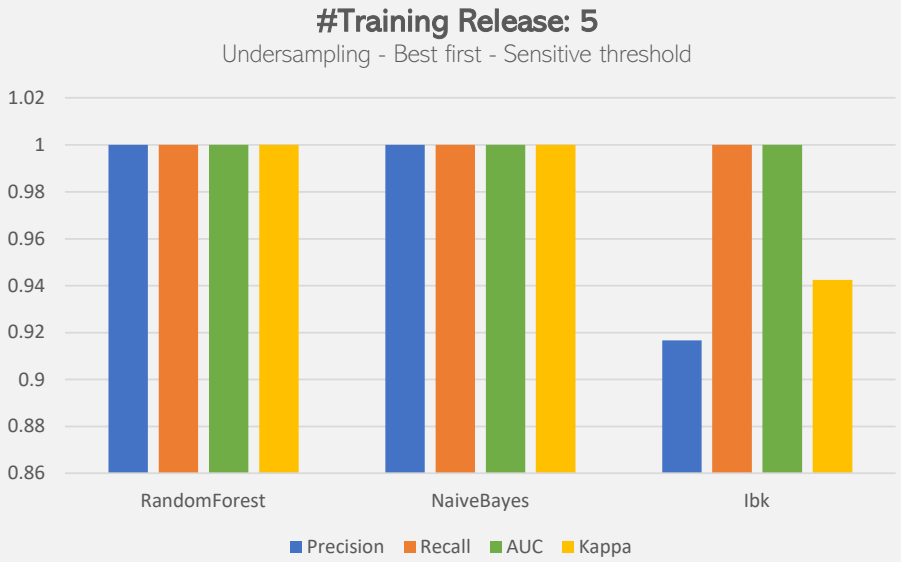
Risultati - Bookkeeper

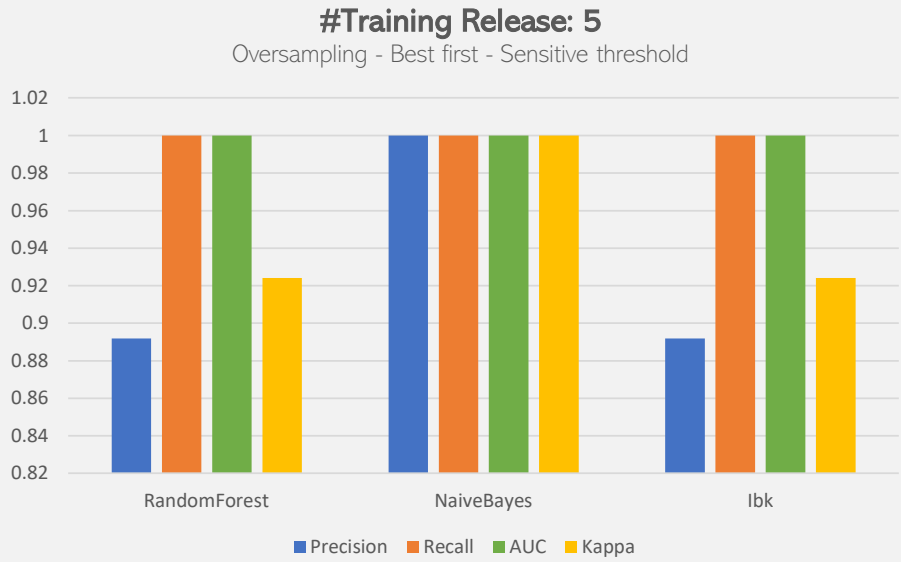




Anche qui varia solo la tipologia di feature selection tra i due grafici, e si nota che l'unico miglioramento avviene per il classificatore NaiveBayes quando si fa sensitive learning e sampling con Smote.

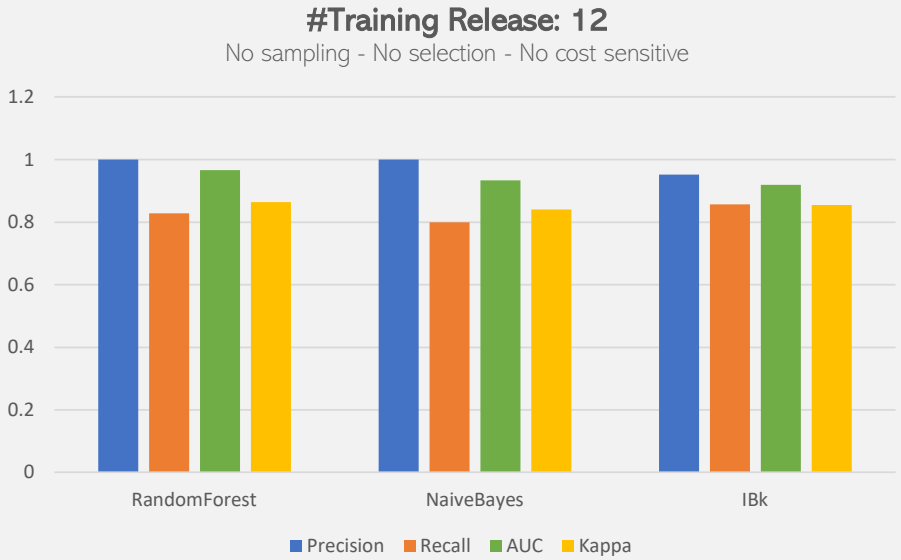
Risultati - Bookkeeper

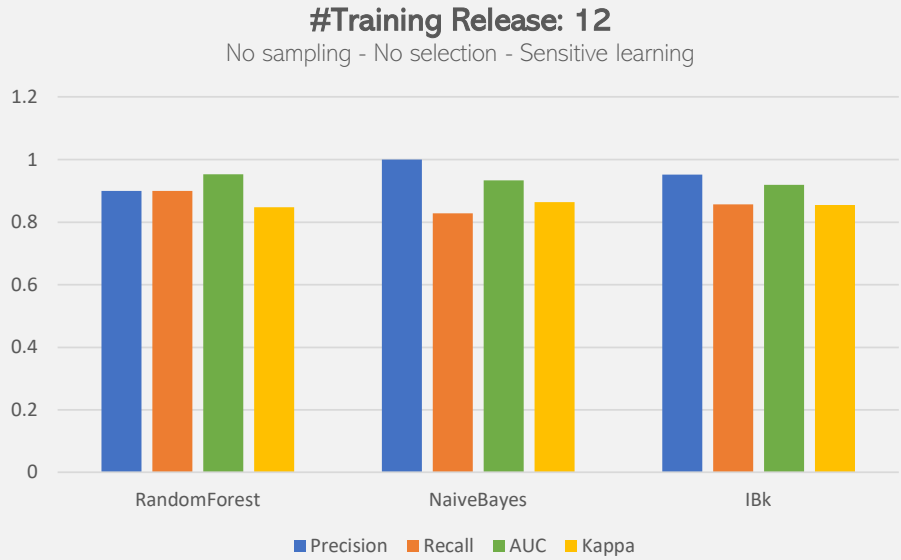




Utilizzando sensitive threshold e fissando la feature selection Best first, è evidente che è meglio scartare alcune istanze dalla classe maggioritaria piuttosto che aggiungere classi buggy con oversampling.

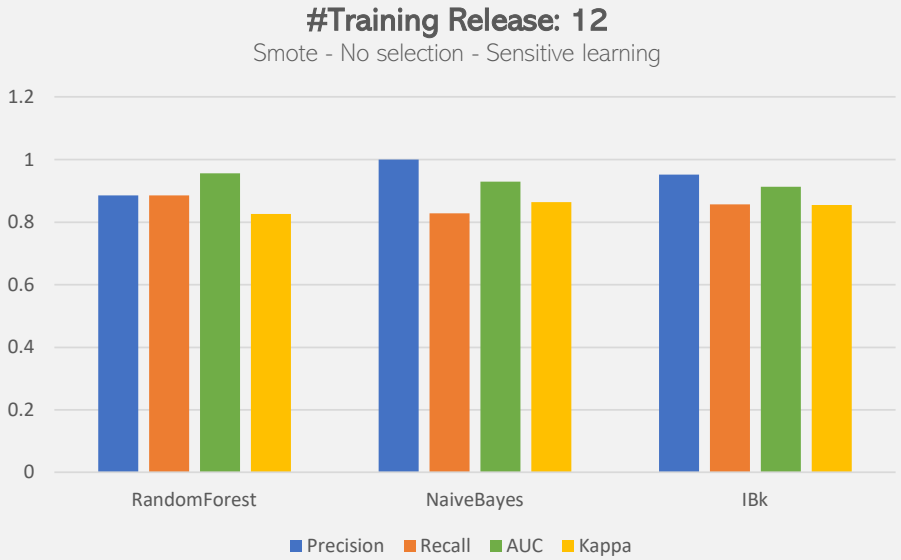
Risultati - Storm

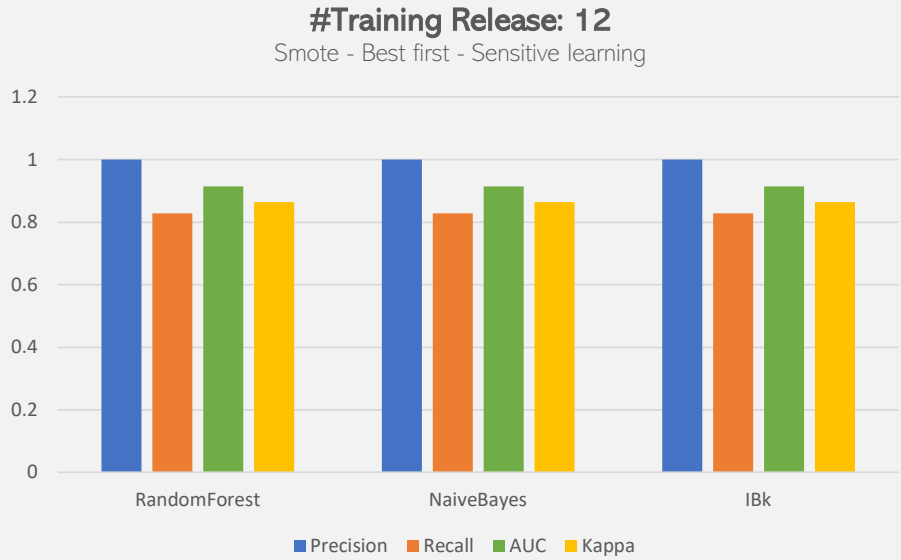




L'utilizzo del sensitive learning, nel caso in cui non si applica sampling né feature selection, non porta a miglioramenti significativi per le metriche considerate.

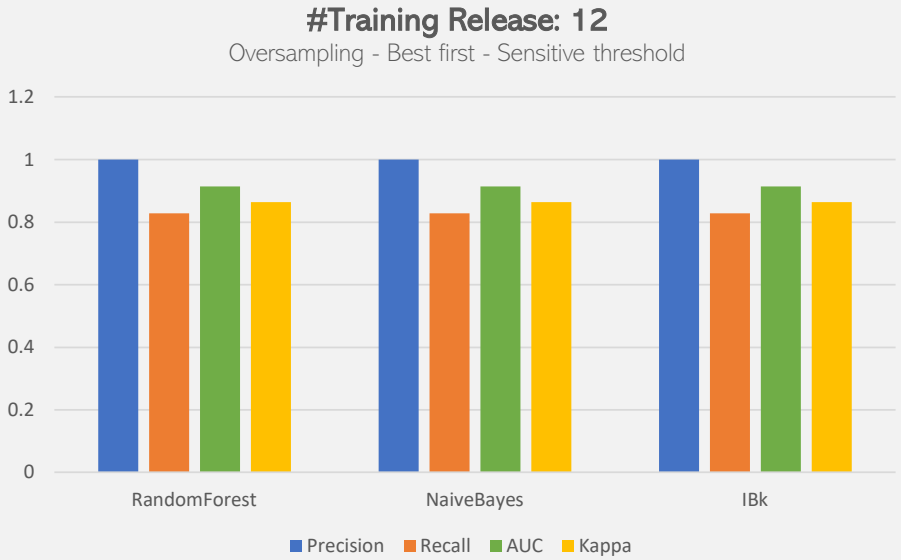
Risultati - Storm

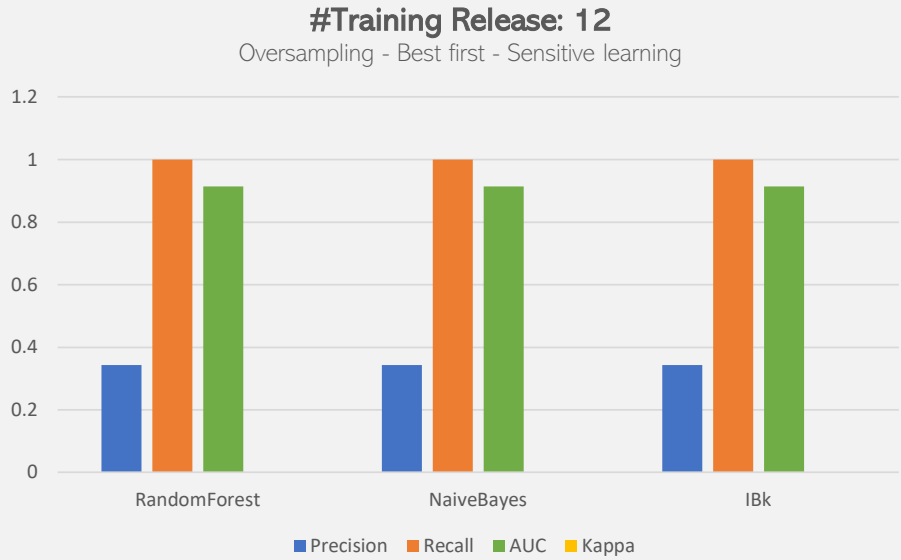




Utilizzando sensitive threshold e fissando la feature selection Best first, è evidente che è meglio scartare alcune istanze dalla classe maggioritaria piuttosto che aggiungere classi buggy con oversampling.

Risultati - Storm





Riferendosi a ciò che è stato detto nella slide precedente, in questo caso l'oversampling porta alla predizione di soli elementi positivi.

Conclusioni

Dall'analisi effettuata, ovviamente non è possibile eleggere una configurazione che sia la migliore in assoluto rispetto alle altre (no silver bullet).

Conclusioni

Dall'analisi effettuata, ovviamente non è possibile eleggere una configurazione che sia la migliore in assoluto rispetto alle altre (no silver bullet).

È opportuno notare che alcuni risultati ottenuti rispecchiano ciò che è stato studiato nella teoria.
Per esempio, introducendo una classificazione cost sensitive, la recall aumenta nella maggior parte dei casi, come ci si aspetta.
Inoltre, i valori alti di Kappa e AUC nella maggior parte dei casi sono indici del fatto che l'insieme delle feature scelte sia correlato alla bugginess.

Conclusioni

Per Bookkeeper, anche se di poco, i risultati migliori sono stati ottenuti con l'undersampling.
Inoltre, la tecnica di feature selection Best first tende a peggiorare i valori delle metriche per Ibk e a migliorare quelli per NaiveBayes.
Anche per Storm si notano gli stessi miglioramenti a seguito dell'utilizzo della tecnica di undersampling, con risultati che possono essere critici per alcune configurazioni con oversampling.

La compatibilità delle considerazioni effettuate per i due progetti suggerisce che queste conclusioni potrebbero essere estese, con un grado ragionevole di sicurezza, ad altri progetti open source simili.

Threats

La principale minaccia alla validità di questo studio riguarda la qualità del dataset.

Garbage In Garbage Out: questo principio sottolinea che i risultati di un'analisi saranno solo buoni quanto i dati inseriti nel processo.

Inoltre, nella generazione del dataset si possono riscontrare molte inconsistenze nei dati, particolarmente evidenti durante il processo di merge tra le informazioni provenienti da Jira e da GitHub, quindi causate dalle transizioni tra le piattaforme.

Link

Github

https://github.com/tizianotaglienti/proj-isw2

SonarCloud

https://sonarcloud.io/project/overview?id=tizianotaglienti_proj-isw2

GRAZIE

PER L'ATTENZIONE