Relazione Progetto Ingegneria del Software 2

Machine Learning for Software Engineering

Christian Santapaola 0294464

Università di Roma Tor Vergata

- github.com/christiansantapaola/ISW2_DELIVERABLE_2
- sonarcloud.io/project/overview?id=qrowsxi_ISW2_DELIVERABLE_2

Indice

Descrizione del problema

L'obiettivo di questa Deliverable consiste nell'estrazione di vari dati riguardanti l'evoluzione e la buggyness di progetti software, al fine di usare questi ultimi per costruire un modello di machine learning il cui obbiettivo è di predire se una classe di codice sia buggata o meno.
I progetti presi in considerazione per l'estrazioni di questi dati sono stati Apache Bookkeeper e Apache Openjpa, due progetti open source. Questo Progetto si divide naturalmente in due fasi:

- L'estrazione dei dati.
- La creazione e l'analisi dei modelli di machine learning.

Milestone 1: Estrazione dei dati - Dati riguardanti l'evoluzione della classe

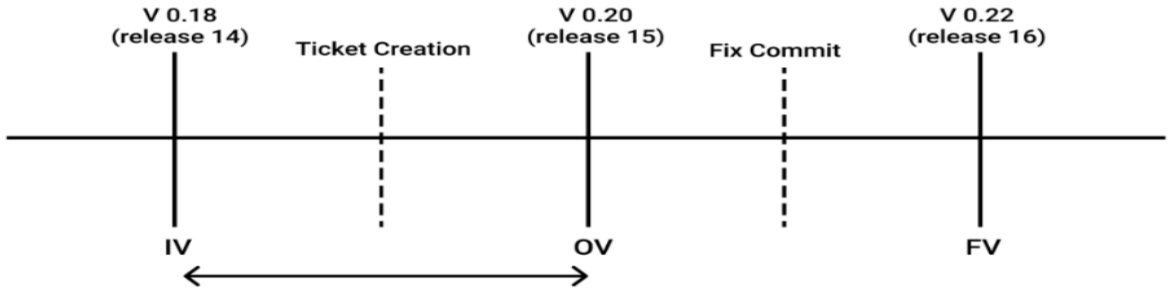
Le informazioni che vogliamo estrarre per ogni classe presente ad una particolare versione del software sono:

- Informazioni sul suo LOC.
- Informazioni sul Churn.
- Il numero di autori che hanno modificato la classe.
- L' eta della classe.

Questi dati vengono estratti tramite l'ausilio del software GIT, analizzando lo stato del software commit dopo commit.

Milestone 1: Bug LifeCicle

Gli altri dati che abbiamo necessita di estrarre riguardano la buggyness della classe.



Da jira vogliamo sapere la data di creazione di un ticket riguardante un bug, dalla quale possiamo ricavare la prima versione in cui il bug è stato scoperto. Dall'analisi dei commit git, possiamo ricavare la data del commit di fix, da cui ricaviamo la Prima versione corretta del softaware.

Milestone 1: proportion

Il problema da porsi adesso è come ricavare dalla conoscenza dell OV e della FV la IV del bug. Per fare ciò utilizziamo una tecnica chiamata proportion, ovvero assumiamo che il numero di versioni per scoprire un bug sia proporzionale al numero di versioni necessarie per fixarlo.

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

Il calcolo della costante di proporzionalità P, nel codice sono state valutate due tecniche:

1. Simple Proportion: si assume IV=OV, ovvero che $P = 1$
2. Increment Proportion: La proporzionalità è da una media sul numero di bug risolti prima della versione presa in esame.

Per il calcolo finale si è scelto il simple proportion, per i seguenti motivi:

- I valori tra il simple proportion e l'increment proportion sono risultati molto simili tra loro, non sono stati apprezzati differenze particolari.
- Il simple proportion è il più semplice da calcolare e da implementare.

Milestone 1: Snoring e taglio del dataset

Nella nostra analisi consideriamo solo i bug risolti, non possiamo considerare bug che esistono ma non sono stati risolti, questo crea un Bias nel dataset, questo bias causa una distorsione nella predizione di classe buggate.
Chiamiamo la causa di questo bias snoring.
Per cercare di arginare gli effetti dello snoring, prendiamo il dataset nel seguente modo:

- Dividiamo a meta le versioni è prendiamo la prima meta.
- Delle versioni successive alla prima meta prendiamo solo le classi che sono affetti da bug che sono stati risolti.

Le motivazioni sono le seguenti, lo snoring ha effetti disastrosi sulla predizione di classi buggate, quindi prendere la prima meta delle versioni ci permette di ridurre drasticamente questo bias, per la seconda meta, lo snoring è un fenomeno che avviene solo nelle classi non difettose, quindi le classi difettose non presentano il problema, quindi le prendiamo nella nostra analisi.

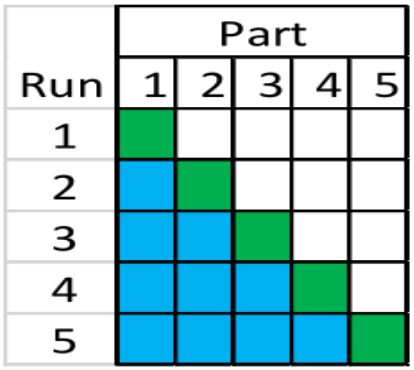
Milestone 2: Introduzione al problema

Creati i dataset, quello che ci rimane da fare è provare i modelli di machine learning. Cosa possiamo dire sul dataset e che è molto sbilanciato, le classi buggy sono molto di meno rispetto alle classi non buggy. I modelli che vogliamo provare sono tutte le combinazioni tra:

- Classificatori:
  - Naive Bayes
  - Random Forest
  - IBK
- Tecniche di Feature Selection:
  - Nessuna tecnica
  - Best First
- Tecniche di Balancing:
  - UnderSampling
  - OverSampling
  - SMOTE
- Tecniche di Cost Sensitive:
  - Nessuna Tecnica
  - Threshold
  - Sensitive Learning

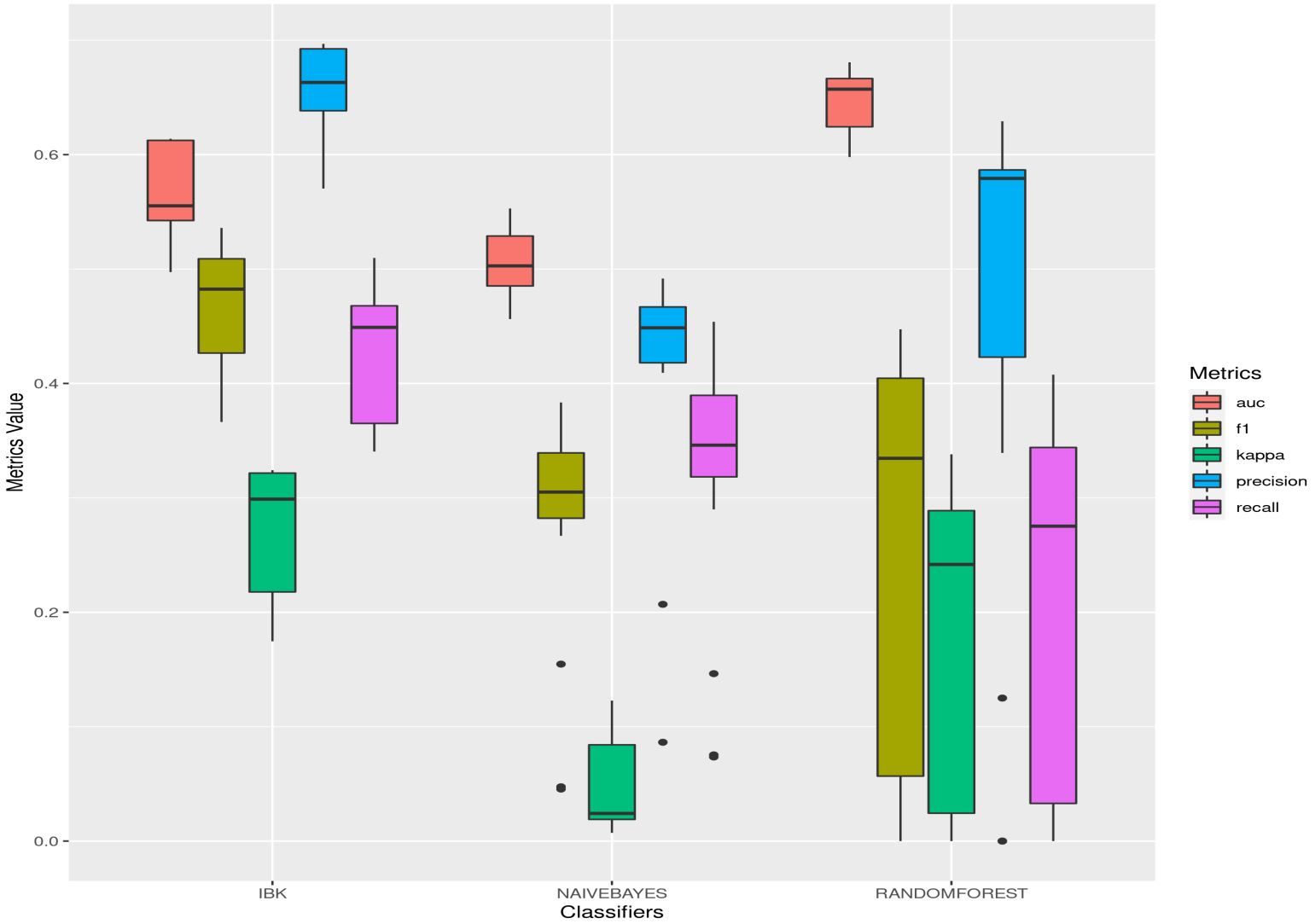
Milestone 2: Walking forward

Il dataset che stiamo valutando è una serie temporale discreta, il cui tempo è dato dalla versione del software. Per valutare l'efficacia del modello è stata utilizzata una tecnica di validazione chiamata walking forward.
Procediamo nel seguente modo:



1. Partizioniamo la nostra serie temporale in versioni (l'unita di tempo).
2. Usiamo come training set tutti i dati fino alla versione N.
3. Usiamo come testing set i dati della versione N+1.
4. Ripetiamo questo procedimento finche non arriviamo ad avere training set composto da tutte le versioni tranne l'ultima.
5. Infine effettuiamo una media dei risultati ottenuti.

Risultato dell'analisi: Bookkeeper

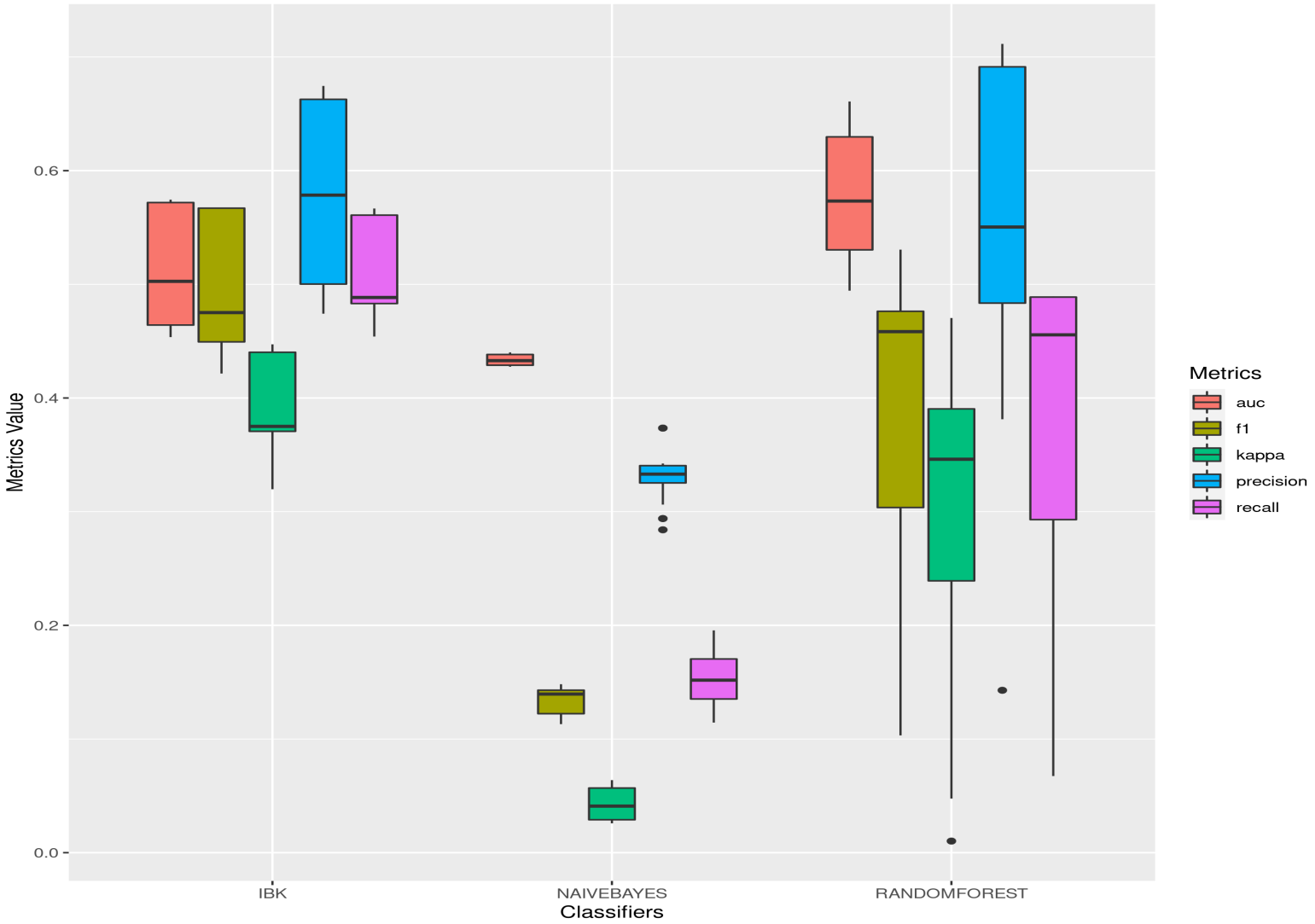


Risultato dell'analisi: Bookkeeper

Questo Box plot riassume per i risultati ottenuti mettendo l'enfasi sul classificatore scelto. Le prime osservazioni che possiamo fare sono:

- Il classificatore IBK tende ad ottenere risultato generalmente migliori rispetto agli altri due.
- Il classificatore NaiveBayes ottiene dei risultati meno variabili.
- Il classificatore RandomForest risulta il più variabile di tutti, passando da buoni risultati a risultati non ottimali, in particolare nonostate la sua precision rimanga relativamente alta, il recall cala drasticamente in base alle tecniche scelte.

Risultato dell'analisi: OpenJPA



Risultato dell'analisi: OpenJPA

Questo Box plot riassume per i risultati ottenuti mettendo l'enfasi sul classificatore scelto. Le prime osservazioni che possiamo fare sono:

- Il classificatore IBK tende ad ottenere risultato generalmente migliori rispetto agli altri due, come nell'analisi di bookkeeper.
- Il classificatore NaiveBayes ottiene dei risultati il meno variabili, in questo caso essendo il dataset molto sbilanciato, ottiene risultati molto peggiori rispetto a bookkeeper.
- Il classificatore RandomForest risulta il più variabile di tutti, passando da buoni risultati a risultati non ottimali, rispetto a bookkeeper tuttavia i risultati peggiori risultano migliori rispetto a naive bayes.

Risultato dell'analisi: Valutare i risultati ottenuti

Raccolti i dati, sorge il problema su come valutare qual'è il classificatore migliore. Sul nostro dataset possiamo affermare che:

- Il dataset è sbilanciato, le classi non difettose sono in numero molto superiore alle classi difettose.

Questo porta alle seguenti conclusioni:

- L'accuracy non è una metrica utile per un dataset sbilanciato.
- La metrica AUC non è la metrica migliore a causa dello sbilanciamento.
- La metrica F1 corrisponde alla media armonica tra la precision ed il recall, questo la rende molto utile nel valutare l'efficacia del classificatore in caso di sbilanciamento, una precision alta ma una recall bassa avra effetti nel valore dell'F1.

Risultato dell'analisi: Conclusioni

Quindi prendendo F1 come metrica di confronto otteniamo che sia per bookkeeper che per openjpa la seguente combinazione, Classificatore IBK, Tecnica di Balancing SMOTE, Nessuna tecnica di feature selection, tecnica di cost sensitivity indifferente.