MACHINE LEARNING FOR SOFTWARE ENGINEERING

Fardella Roberto – 0334186

A.A. 2022/2023

AGENDA

- **Introduzione**
- **Obiettivo**
- **Metodologie**
- **Risultati e conclusioni**

1 - CONTESTO

Definizione degli obiettivi dell'ingegneria del software:

- Sviluppo di software di alta qualità.
- Garanzia di affidabilità e sicurezza.
- Come possiamo soddisfare tali requisiti?
  - Mirare a effettuare attività di testing che ci permettano di far emergere bug nel software.

1 - CONTESTO (2)

Ostacoli nella definizione degli obiettivi dell'ingegneria del software:

- **Quali problemi possono sorgere dura?**
- Testare l'intero software può essere un compito impegnativo e complesso, e spesso possono emergere diversi problemi durante questo processo. Ecco alcuni dei problemi comuni che possono sorgere nel testare tutto il software:
  - **Copertura incompleta:** è difficile testare tutte le possibili combinazioni di input, scenari e situazioni che potrebbero verificarsi nel software.
  - **Tempo e risorse limitate:** il test completo di un software richiede tempo e risorse significative. Le scadenze strette o le limitazioni di budget possono rendere difficile eseguire tutti i test necessari o dedicare abbastanza tempo per individuare e risolvere i difetti.

2 - OBIETTIVO

Scopo e Obiettivi dello Studio Empirico

- Per risolvere tale problema, con queste informazioni, è possibile sfruttare gli strumenti del Machine Learning per predire quali sono le classi che attualmente possono contenere dei difetti, poiché le attività di testing sull'intero software è molto oneroso.

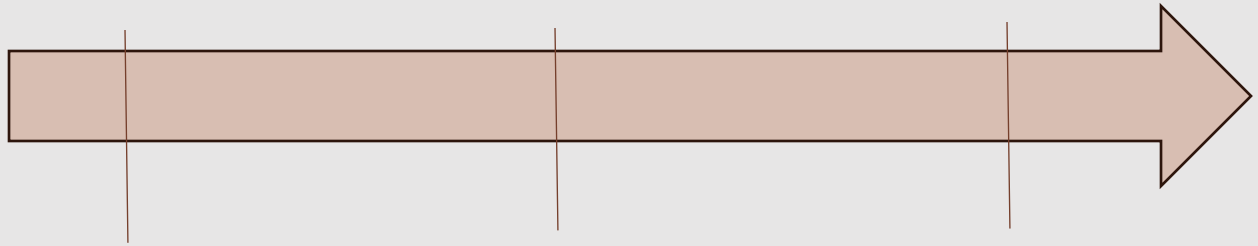
- Dopo aver preso due progetti Apache (BookKeeper e Avro), aver individuato quali loro classi sono state buggy e in quali release, e aver considerato tre classificatori (Random Forest, Naive Bayes e IBk), il nostro scopo è stabilire quale classificatore effettua le predizioni migliori e con quali tecniche di utilizzo.

- È stata utilizzata come tecnica di labeling dei bug Proportion Incremental e, come tecnica di valutazione del modello predittivo, Walk Forward.

- Le tecniche di utilizzo di un classificatore che andremo a considerare sono la feature selection, il sampling, la cost sensitive classifier e alcune loro combinazioni.

La nostra intuizione: è possibile supporre che esista una proporzionalità tra l'arco di tempo che trascorre da quando un bug viene rilevato a quando viene risolto e l'arco di tempo che trascorre da quando un bug viene introdotto a quando viene risolto.

METODOLOGIE



Sfruttare il ciclo di vita dei difetti per etichettare versioni e classi difettose interessate.

Injected version: release in cui il bug è stato introdotto.

Opening version: release in cui il bug è stato rilevato a seguito di una failure del sistema.

Fixed version: release in cui il bug è stato eliminato.

- Le informazioni sulla versione di apertura (opening version) e sulla versione di correzione (fixed version) sono sempre disponibili su Jira, poiché i tempi in cui il bug viene rilevato e risolto sono noti. Tuttavia, non tutti gli issue aperti su Jira sono contrassegnati da una versione di inserimento (injected version) e determinarla non è un compito semplice.

La metodologia utilizzata per stimare l'injected version dei bug è chiamata Proportion, in cui è possibile definire un fattore di proporzione p come costante di scala secondo la seguente modalità:

METODOLOGIE

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

Sfruttare il ciclo di vita dei difetti per etichettare versioni e classi difettose interessate.

- Poiché le AV (Associated Versions) non sono disponibili o complete nei ticket estratti da JIRA, si utilizza il calcolo delle AV tramite la tecnica "proportion incremental" basata sulla media dei valori di "proportion" dei ticket consistenti fino a quel momento.
- Nel caso in cui il numero dei ticket consistenti sia inferiore a una soglia specifica (impostata a 5 nel nostro caso), viene adottato un approccio "cold-start" in cui si prende la mediana dei valori medi di "proportion" calcolati sui ticket consistenti di altri progetti Apache con la più alta percentuale di ticket consistenti.
- Per evitare il fenomeno dello "snoring", che si manifesta quando ci sono coppie (classe, release) che risultano non affette da bug, ma in realtà presentano difetti non rilevati, è consigliabile escludere tutti i dati relativi alla seconda metà delle release e lavorare solo con i dati rimanenti al fine di eliminare tale problema.

METODOLOGIE

Costruzione base dei dataset

- Nella prima fase del progetto sono state raccolte delle metriche nel contesto di 2 progetti open-source:
    1. Apache/Bookkeeper;
    2. Apache/Avro.
- Il processo di misurazione è stato effettuato tramite l'utilizzo di due piattaforme:
    1. JIRA: issue tracking system per la raccolta dei ticket e delle versioni;
    2. Git: Version Control System per la raccolta dei commit.

- Nella seconda fase del progetto vengono costruiti dei dataset di training e di testing, per poi applicare algoritmi di ML per eseguire predizioni.

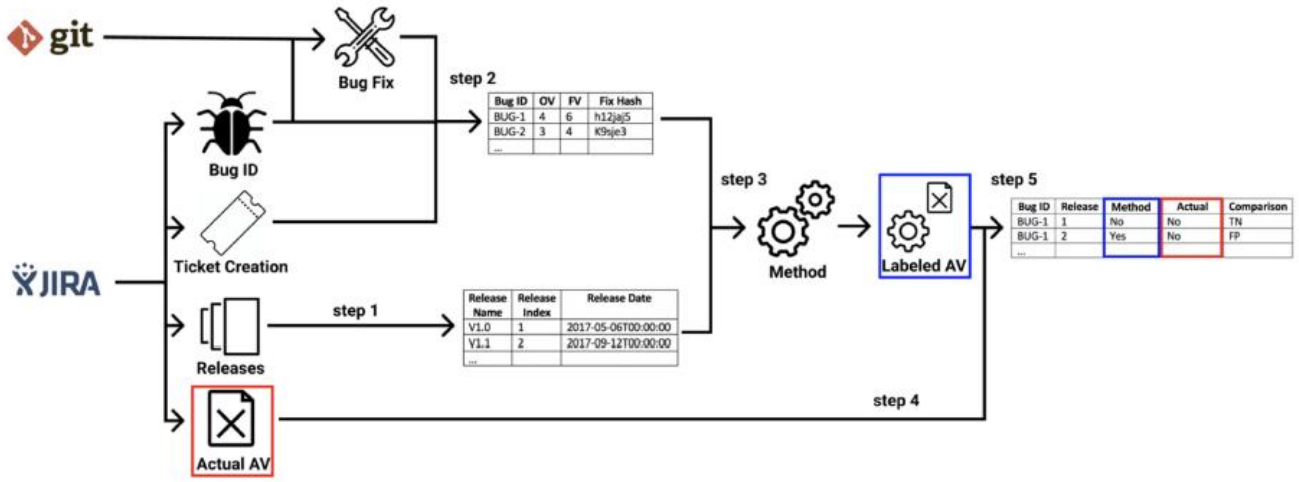


Fig. 4. The process to measure the accuracy of methods in labeling affected versions.

METODOLOGIE

11 Metriche considerate

nome	Descrizione
Loc	Numero di linee di codice.
NR	Numero di revisioni all'interno della singola release.
Nauth	Numero di autori.
CC	Complessità ciclomatica
NComm	Numero di linee di codice commentate.
LocAdded	Somma delle linee di codice aggiunte sulle revisioni relative alla release.
MaxLocAdded	Numero massimo delle linee di codice aggiunte in una singola revisione.
AvgLocAdded	Media delle linee di codice aggiunte sulle revisioni relative alla release.
Churn	Somma di |linee di codice aggiunte linee di codice rimosse| sulle revisioni relative alla release.
MaxChurn	Valore massimo del churn in una singola revisione.
AvgChurn	Media dei churn sulle revisioni relative alla release.

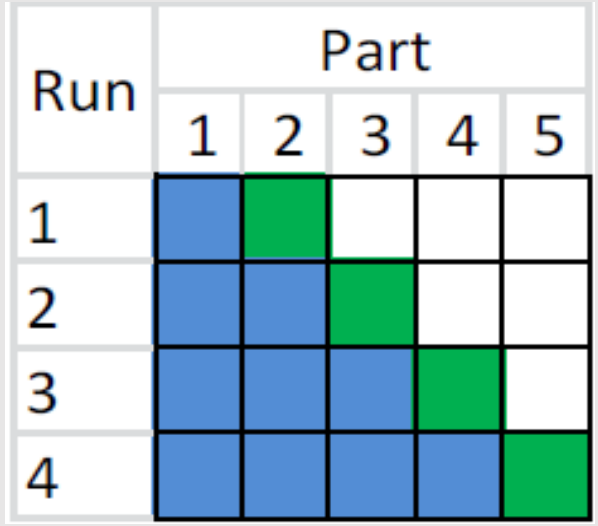
La tecnica di validazione Walk Forward è un metodo time series che considera l'ordine temporale dei dati. È fondamentale che il set di addestramento non contenga informazioni future rispetto ai dati di test, per garantire una valutazione accurata.

METODOLOGIE

- Suddividendo il dataset in periodi temporali consecutivi, si addestra il modello sui dati precedenti e si valuta la sua capacità di generalizzazione testandolo sui dati successivi.

Walk-Forward come Tecnica di valutazione

Training set	Testing set
Il training set deve essere rappresentativo dell'utilizzo effettivo del classificatore, cercando di riflettere fedelmente le condizioni reali e non contenendo informazioni del futuro.	Il testing set deve essere rappresentativo della realtà, sfruttare tutti i dati disponibili e non essere affetto da fenomeni di snoring.



METODOLOGIE

Metriche utilizzate per valutare l'efficacia dei modelli di classificazione

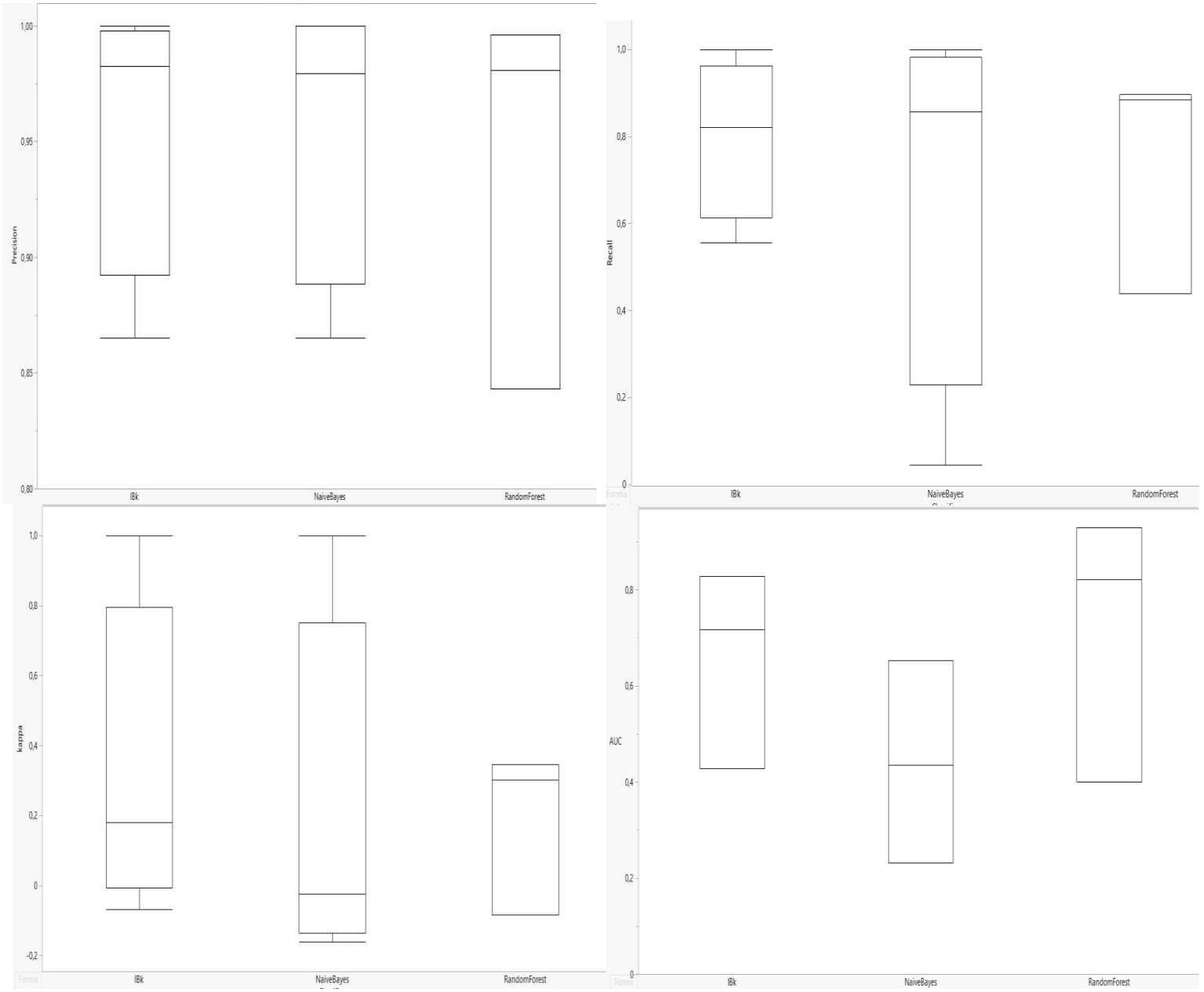
Le metriche considerate, per valutare le prestazione di un dato classificatore, sono le seguenti:

- Precision.
- Recall.
- Kappa.
- Area Under Roc (AUC).

- La tecnica di ricampionamento utilizzata per affrontare il problema delle classi di risposta sbilanciate in un dataset è Synthetic Minority Over-sampling Technique (SMOTE).

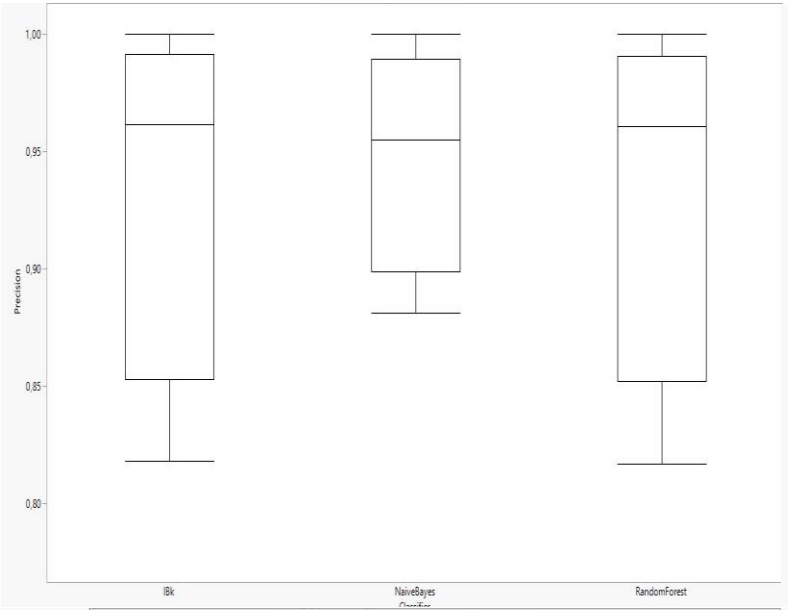
- In alternativa, per gestire lo sbilanciamento dei dati, è stata utilizzata la tecnica di Cost-Sensitive Learning.

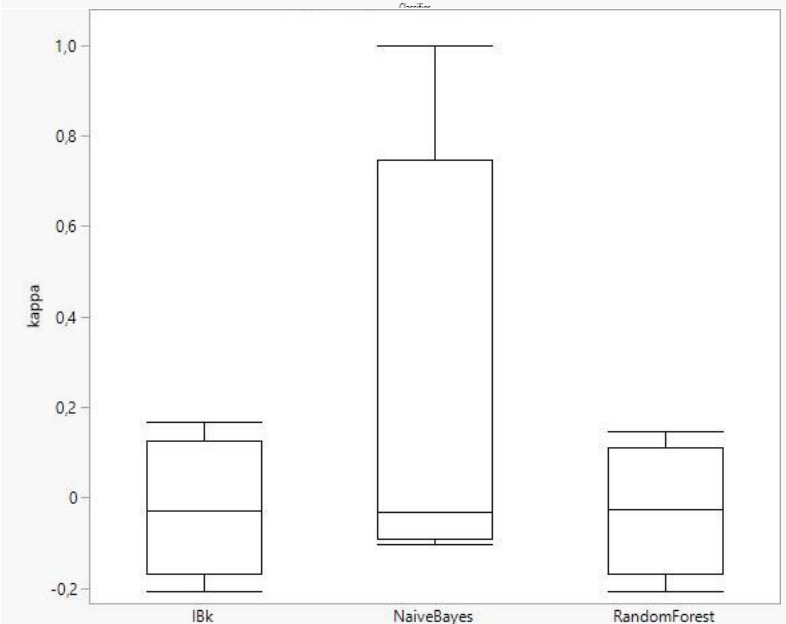
RISULTATI - Bookkeeper senza filtri

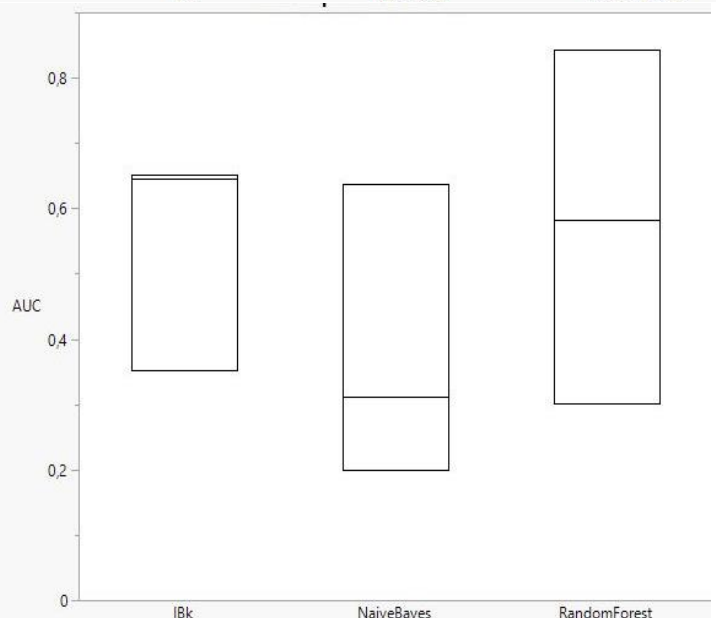


Il classificatore Random Forest sembra comportarsi meglio

Risultati – Bookeepper con feature Selection

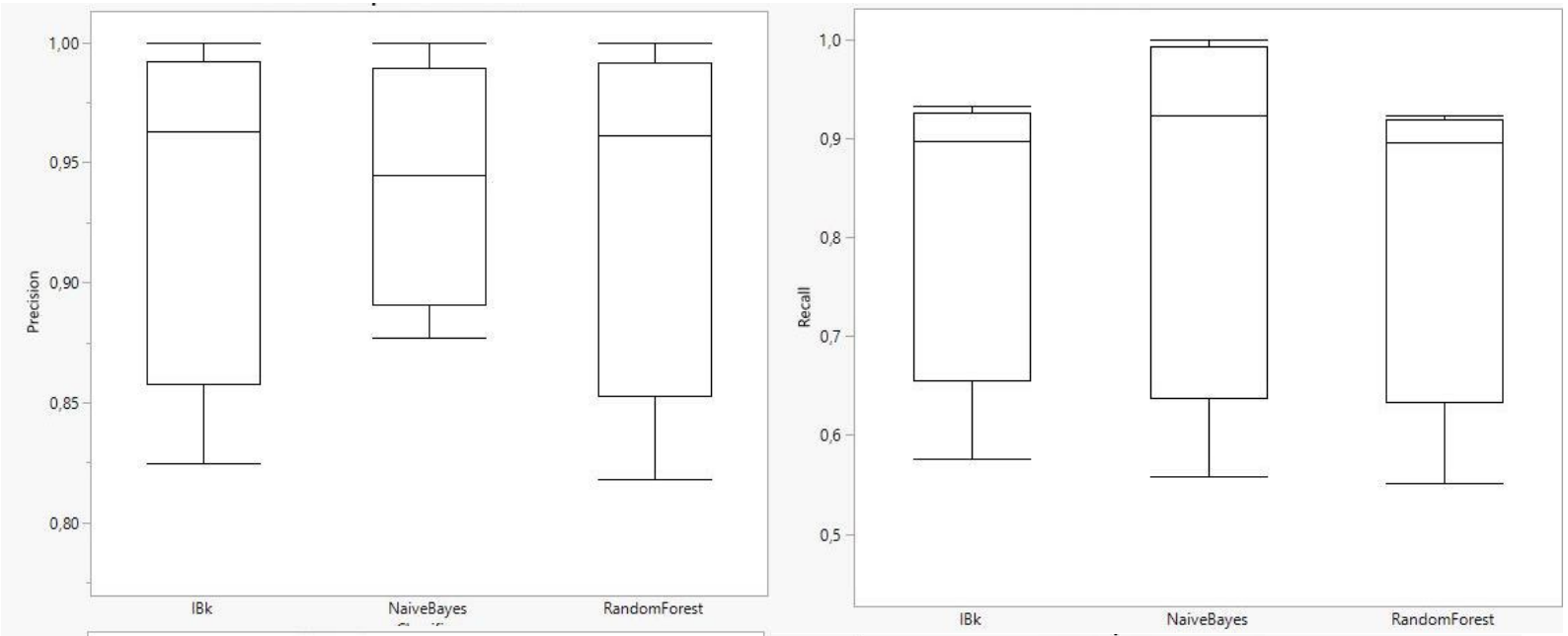


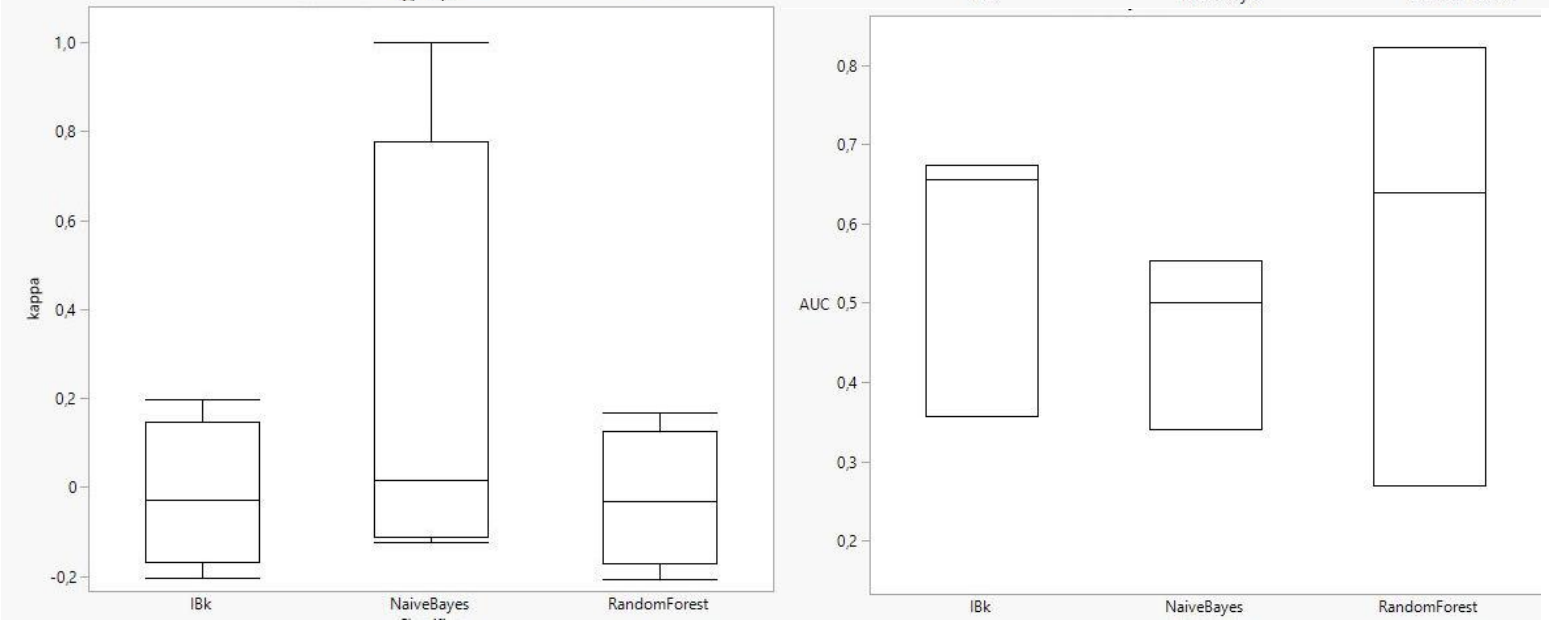




Non c'è un classificatore che si distingue particolarmente. NaiveBayes presenta meno variabilità rispetto al caso senza feature selection.

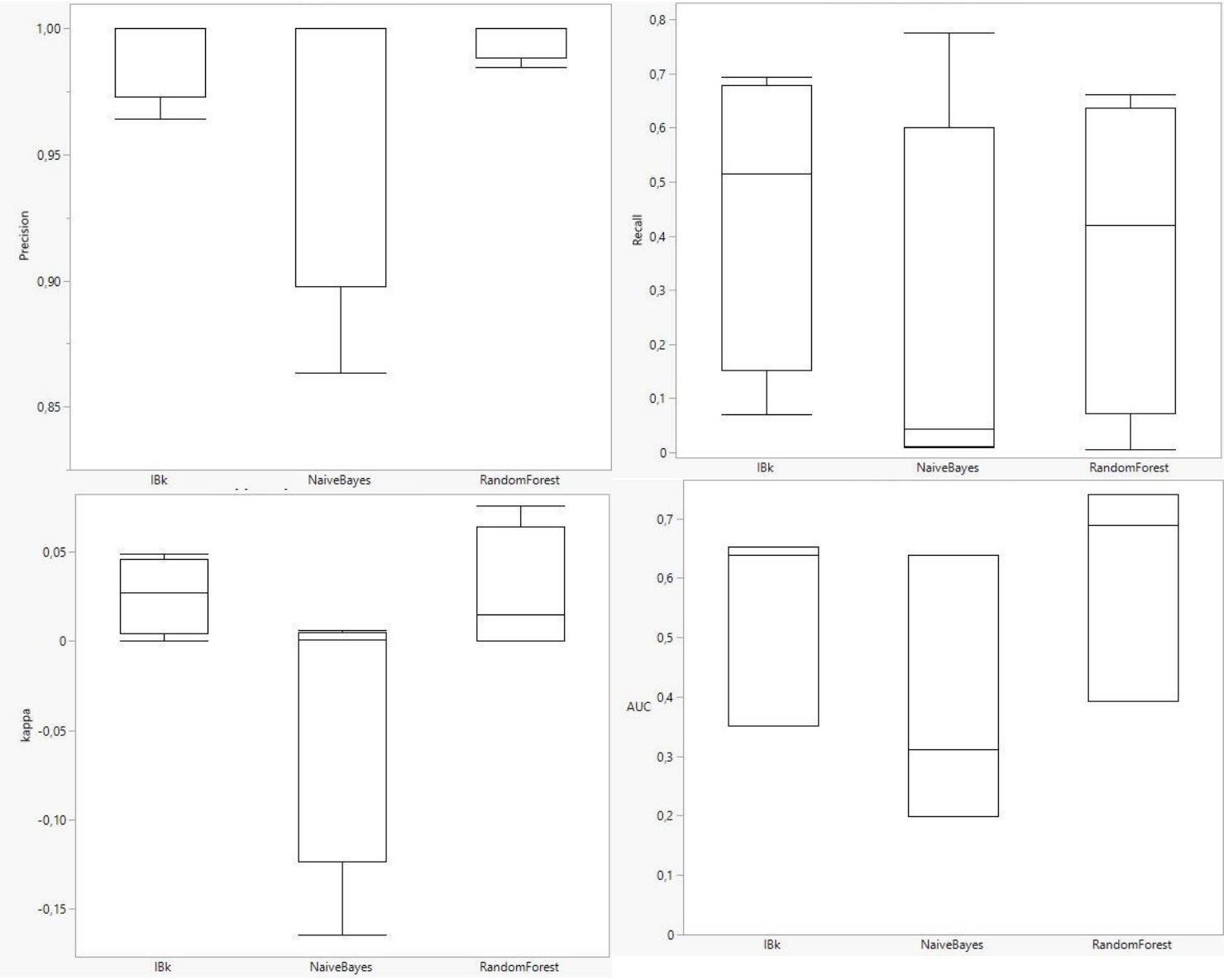
Risultati – Bookeepper con feature Selection e sampling SMOTE





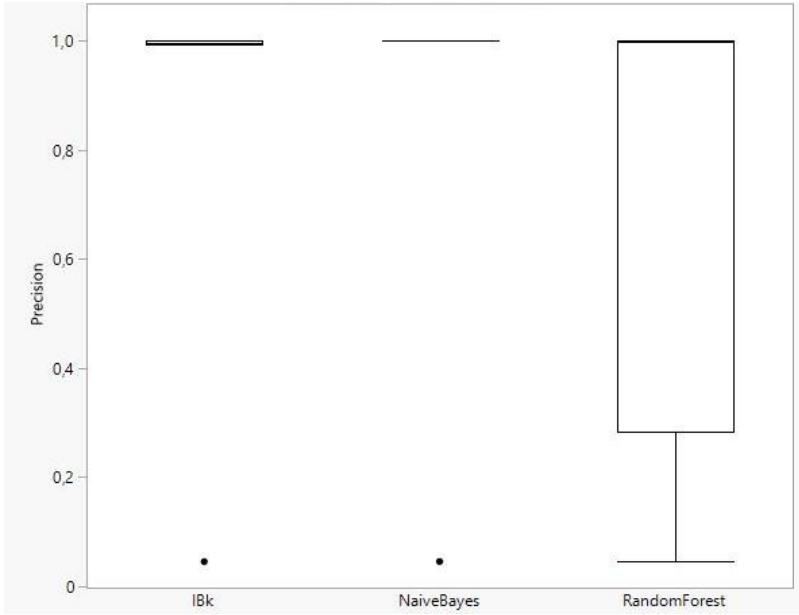
Non c'è
un classificatore
che si distingue
Particolarmente.

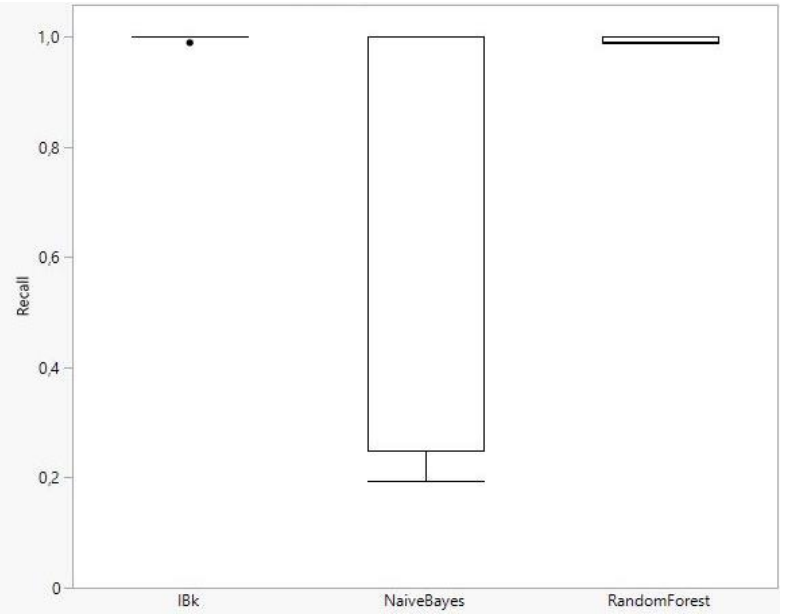
Risultati – Bookeepper con feature Selection e cost sensitive

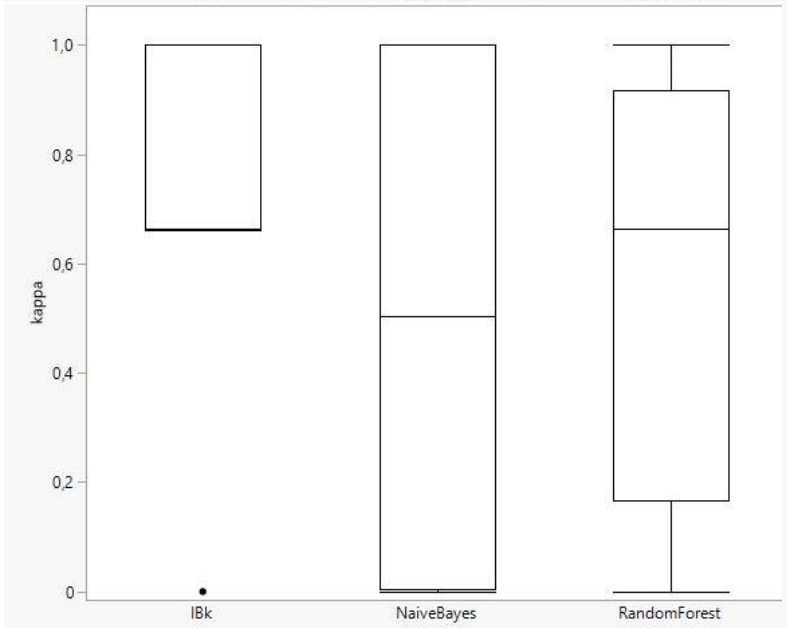


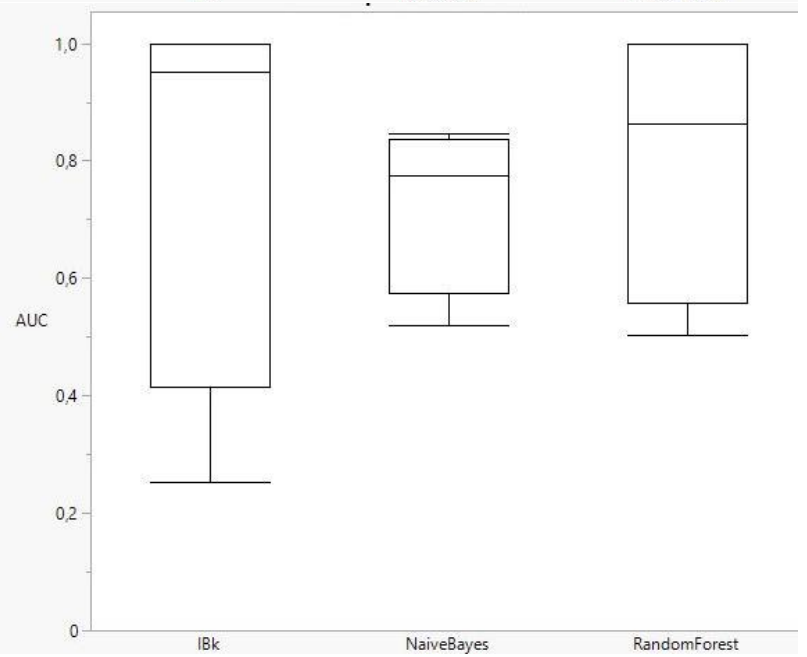
Applicando come tipo di sensitività ai costi, Sensitive Learning (con CFN = 10 * CFP), le prestazioni del classificatore NaiveBayes calano drasticamente.

Risultati – Avro senza filtri



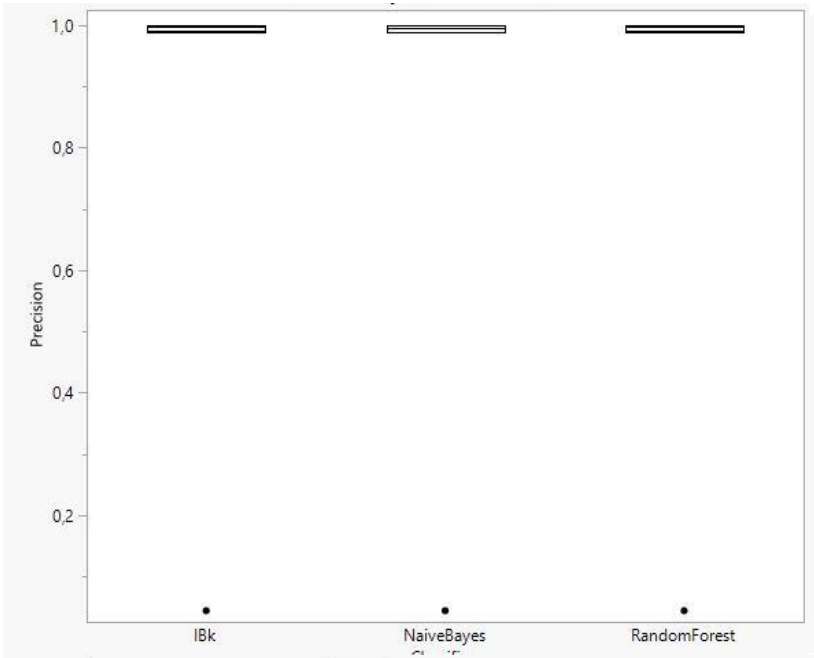


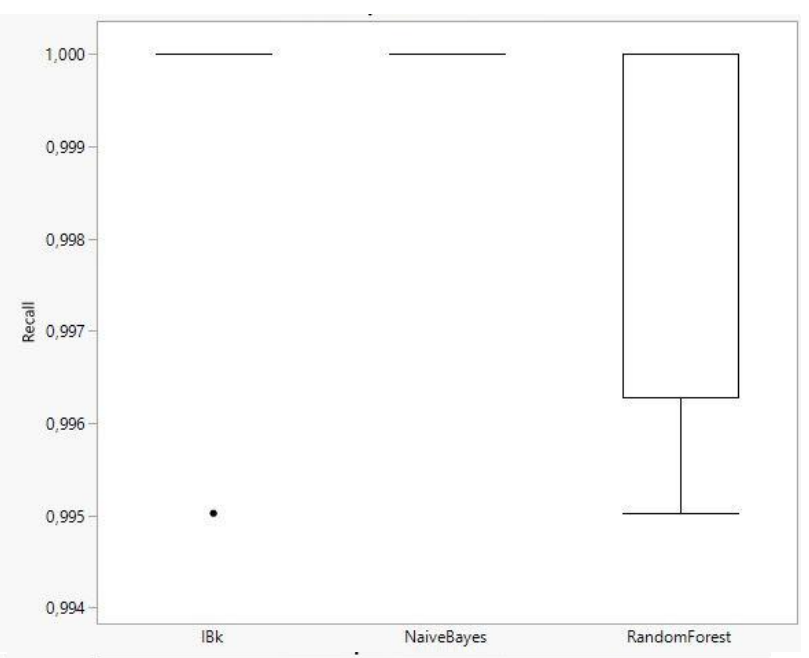


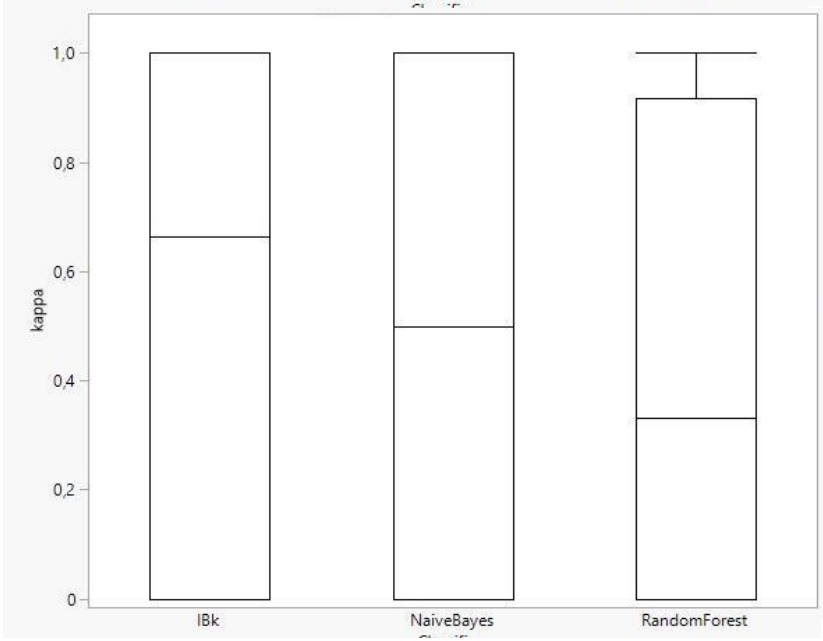


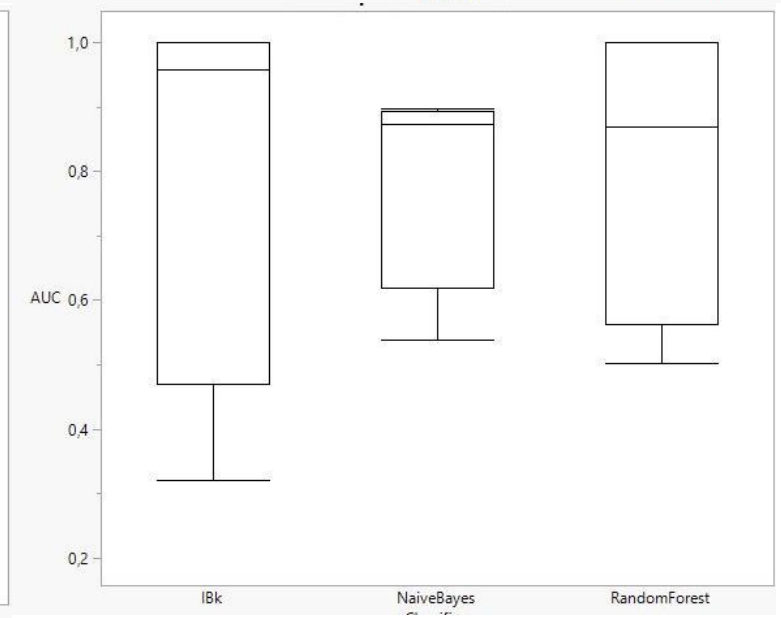
Ibk sembra comportarsi meglio

Risultati – Avro con feature Selection

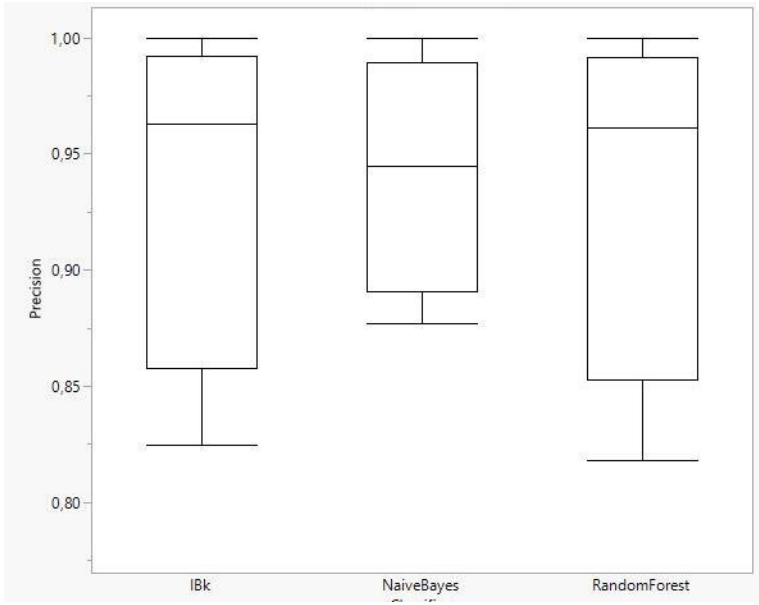


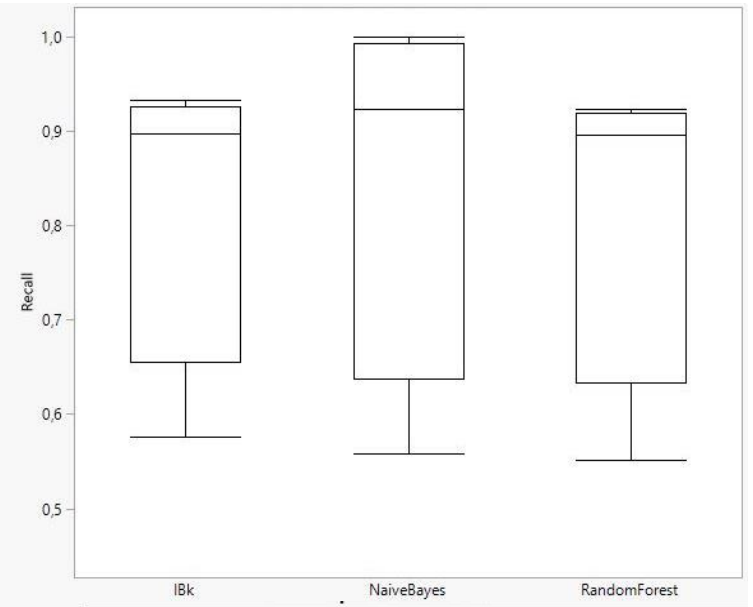


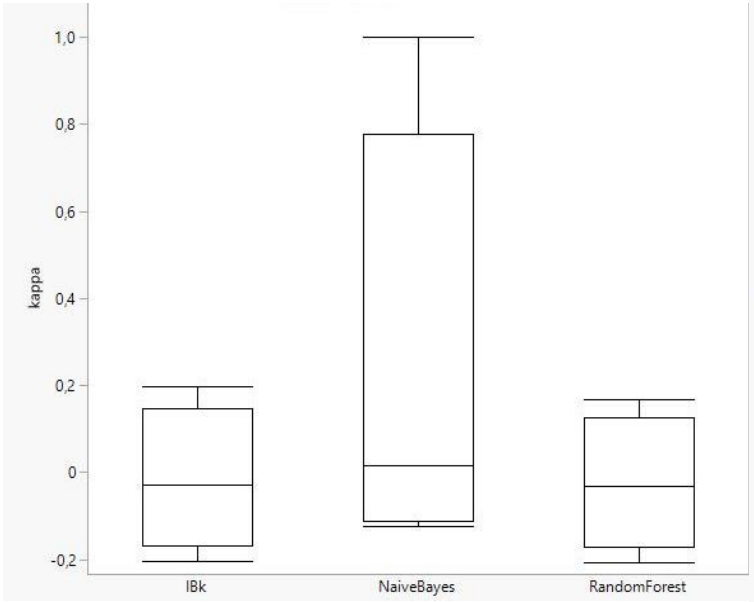


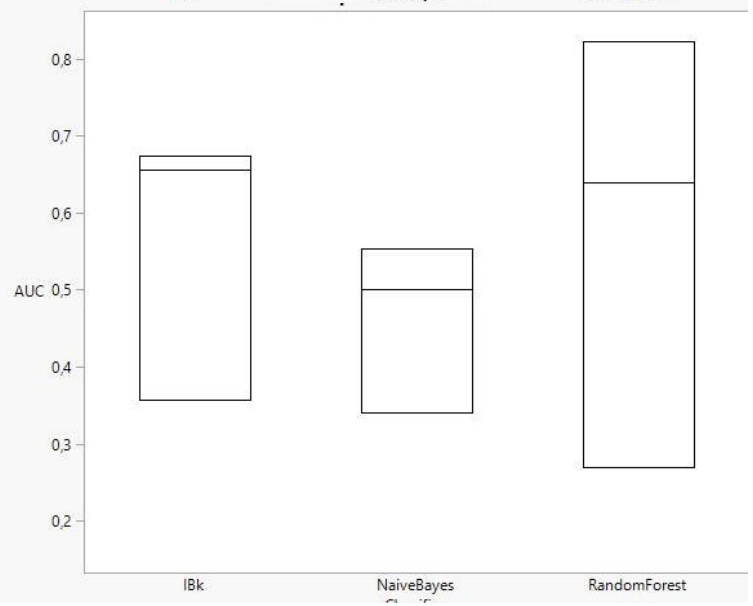


Il classificatore IBk sembra essere il più adeguato. Notare l'alta variabilità della metrica kappa. Il motivo di quest'ultima lo vedremo più avanti

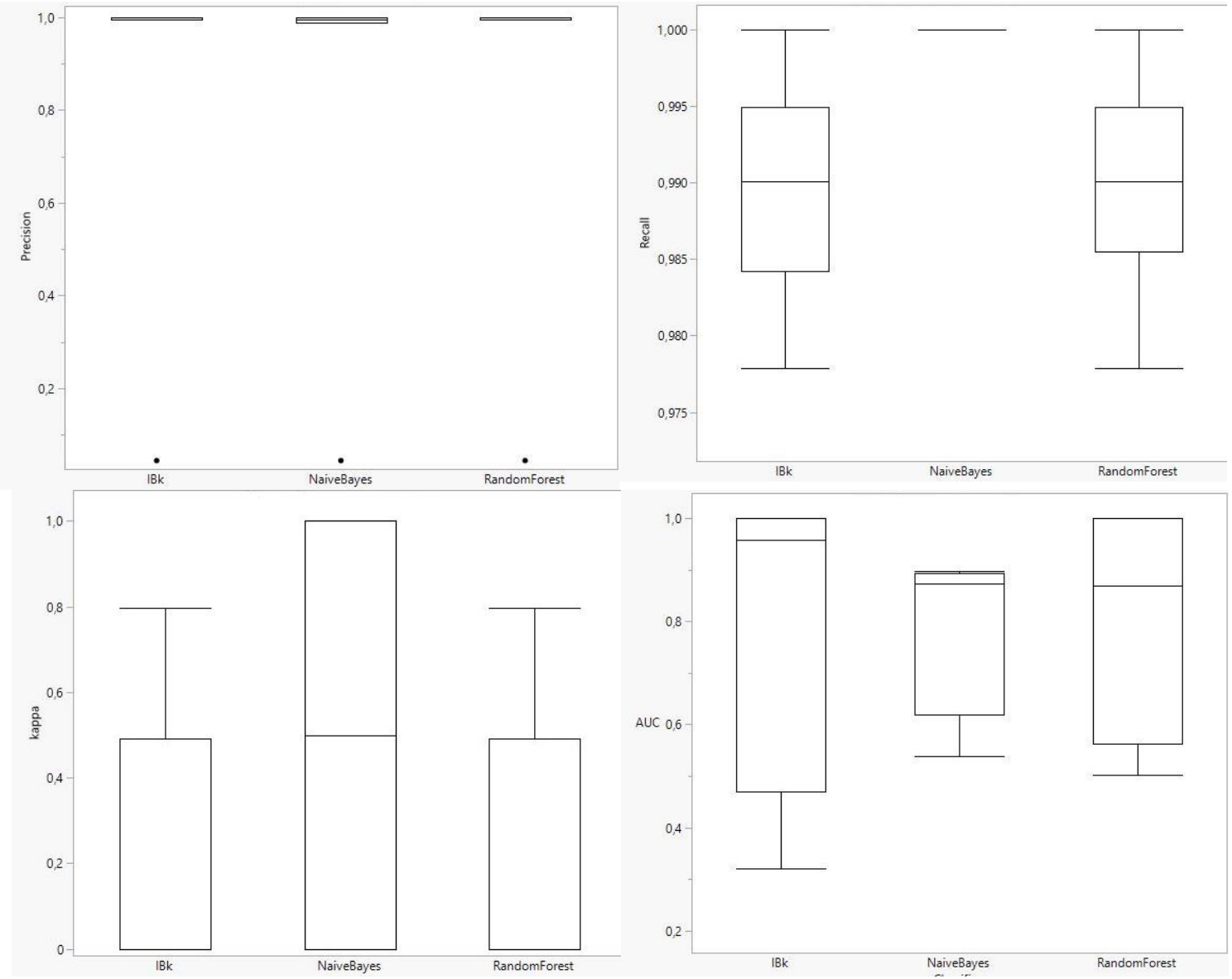








Notare che, con alla tecnica di sampling SMOTE, è stato fatto un ribilanciamento delle classi, e nonostante ciò, si mantiene un elevato grado di precision e recall per tutti i classificatori utilizzati.



il classificatore NaiveBayes risulta essere il migliore con il dataset Iniziale e utilizzando come combinazione Feature Selection e Cost Sensitive Learning (con CFN = 10* CFP).

CONCLUSIONI

Apache Bookkeeper

Nel progetto Bookkeeper, durante i test con i classificatori IBK, Naive Bayes e Random Forest, sono stati osservati i seguenti risultati:

1. Senza filtri, il classificatore predominante è Random Forest. Questo significa che senza l'applicazione di alcuna tecnica aggiuntiva, il Random Forest ha ottenuto prestazioni migliori rispetto agli altri classificatori in base alle metriche utilizzate.

2. Con l'utilizzo della feature selection, si è ottenuta una variabilità minore nelle metriche considerate, rispetto al caso senza feature selection. Ciò suggerisce che la selezione delle caratteristiche ha contribuito a migliorare la stabilità delle prestazioni dei classificatori.

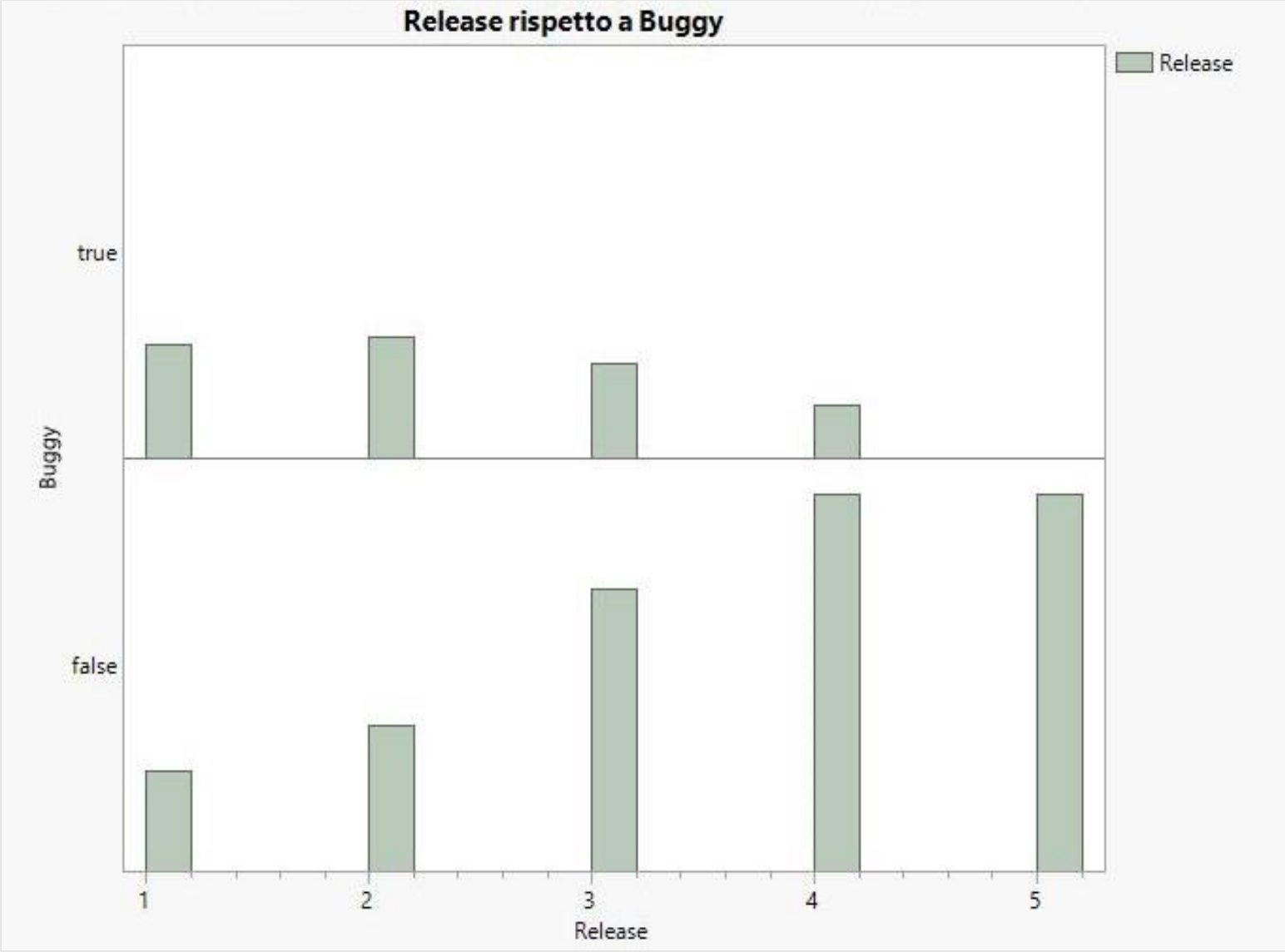
3. Applicando sia la feature selection che sampling SMOTE, i tre classificatori utilizzati non presentano grosse differenze in termini prestazionali, eccetto in qualche caso in cui un classificatore è meno variabile rispetto ad un altro nella metrica AUC o Kappa.

4. Infine, applicando la feature selection e il cost-sensitive learning, si è ottenuta una precisione più elevata, ma una recall media più bassa rispetto al solo utilizzo della feature selection. Le metriche AUC e kappa sono rimaste invariate e, il peggior classificatore è risultato essere Il Random Forest..

Andamento delle classi buggy e non buggy in funzione nel numero di release del progetto.

CONCLUSIONI

Apache Bookkeeper



CONCLUSIONI

Apache Avro

Nel progetto Avro, sono stati ottenuti i seguenti risultati utilizzando i classificatori IBK, Naive Bayes e Random Forest con diverse tecniche applicate:

1. Senza filtri, il classificatore IBK ha mostrato un andamento migliore rispetto agli altri due classificatori, con precisione e recall di media, pari ad 1.

2. Con l'applicazione della feature selection, i risultati sono stati simili al caso senza filtri con IBk che ancora sembra essere il più adeguato, con una varianza elevata per Kappa. Questa varianza per i tre classificatori potrebbe indicare una maggiore variabilità nelle prestazioni dei classificatori e la causa di ciò è dovuta al fatto che c'è uno sbilanciamento tra le classi.

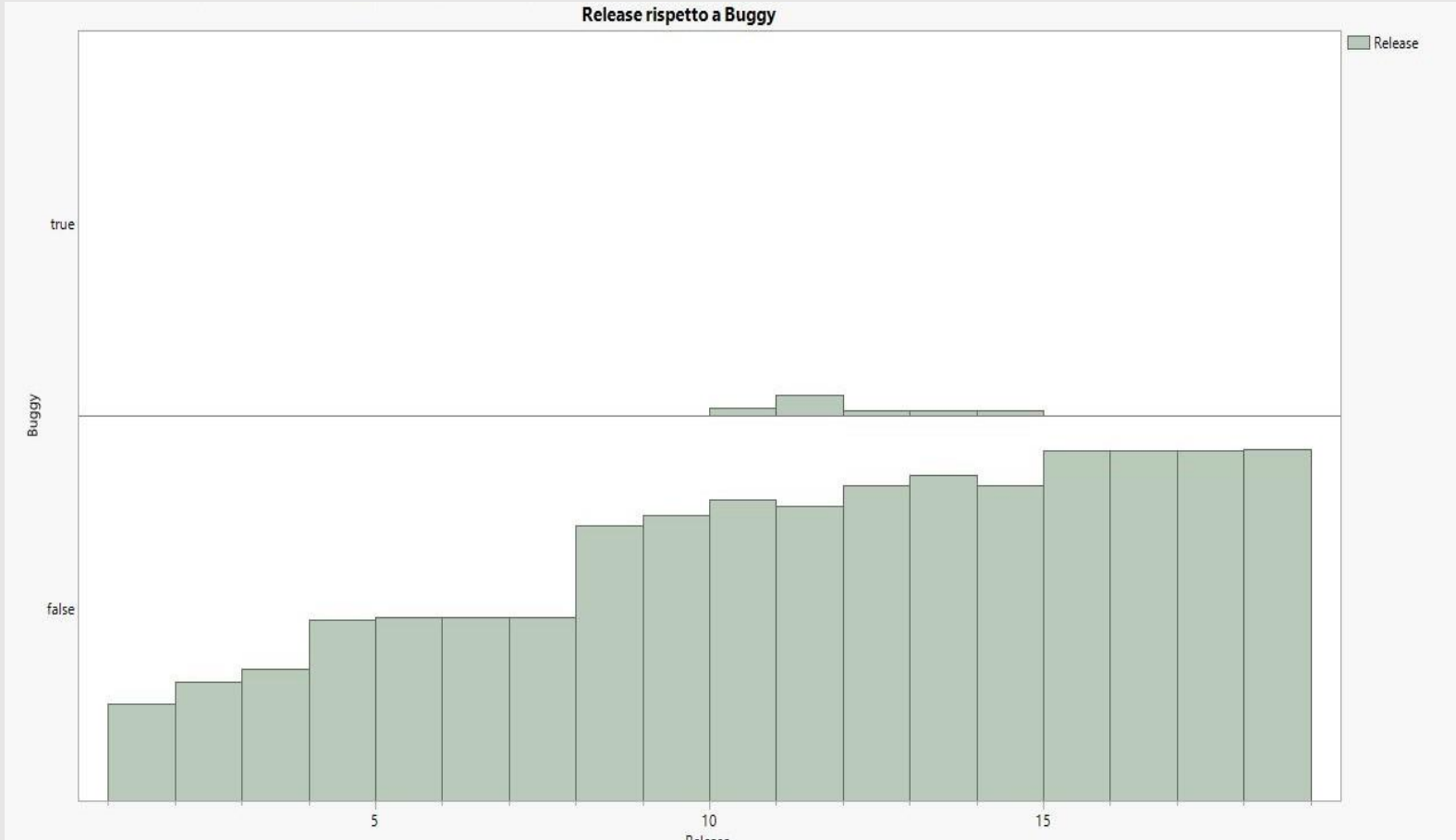
3. Applicando sia la feature selection che la tecnica di sampling SMOTE, nessuno dei tre classificatori si è distinto particolarmente. Ciò potrebbe indicare che l'applicazione del sampling SMOTE ha contribuito a bilanciare le prestazioni dei classificatori e ridurre le differenze tra di loro.

4. Infine, applicando la feature selection e il cost-sensitive learning, il classificatore NaiveBayes ha dimostrato di essere il più efficace con un risultato molto positivo che suggerisce un'alta capacità predittiva del modello. Tuttavia, la varianza, per il classificatore Naive Bayes per la metrica Kappa, è risultata drasticamente alta. Ciò potrebbe indicare che il modello ha avuto prestazioni molto diverse a seconda dei casi specifici, ma in generale ha dimostrato una buona capacità di gestire i costi relativi degli errori di classificazione.

CONCLUSIONI

Apache Avro

Nel progetto Avro, considerando solo la prima metà delle release del progetto, si è osservato che tra le prime 9 release e quelle successive alla 14, tutte le classi mostrano di non avere difetti, mentre nelle altre release la situazione sembra differire. Questa discrepanza potrebbe essere un problema poiché potrebbe influenzare le prestazioni del modello, portando a una diminuzione della precisione e della recall durante il test delle future release e compromettendo l'affidabilità complessiva del modello. Pertanto, si consiglia di utilizzare tecniche di ribilanciamento delle classi per affrontare questa situazione.



LINK

- Repository Github: https://github.com/Zudel/Deliverable-ISW2
- SonarCloud: https://sonarcloud.io/project/overview?id=Zudel_Deliverable-ISW2