MACHINE LEARNING APPLICATO ALL'INGEGNERIA DEL SOFTWARE

Luca Capotombolo
Matricola: 0316166

AGENDA

- Contesto
- Scopo
- Progettazione
- Discussione dei risultati per BookKeeper
- Discussione dei risultati per ZooKeeper
- Link del progetto per Sonarcloud e per GitHub

CONTESTO

- Il testing rappresenta una fase fondamentale nella produzione di un sistema software. In un progetto di grandi dimensioni il testing richiede una quantità di risorse non trascurabili.
- Nella pratica, il budget a disposizione per testare le componenti non è sufficiente per coprire l'intero sistema. Sarebbe opportuno testare per primi gli artefatti software che hanno maggiore probabilità di avere difetti e che rivestono un ruolo importante all'interno del sistema. All'interno di sistemi complessi, il testing non può essere fatto in maniera esaustiva.
- Tuttavia, potrebbe risultare complesso stabilire quali classi testare per prime senza avere a disposizione determinati strumenti. Il **Machine Learning** applicato all'Ingegneria Del Software può rappresentare una possibile soluzione al problema.
- Nella fase di testing l'obiettivo è quello di trovare il maggior numero di bug minimizzando lo sforzo e i costi.
- Un bug viene iniettato nel codice a causa di un errore e potrebbe determinare una failure.

SCOPO

- Lo scopo del progetto è quello di valutare le prestazioni dei classificatori IBK, NAIVE BAYES e RANDOM FOREST nella predizione della bugginess per le classi java all'interno di una release per i progetti Apache BookKeeper e Apache ZooKeeper.
- Nel progetto sono state utilizzate tecniche di balancing, di feature selection e di sensitivity per entrambi i progetti.
- E' stato condotto uno studio sugli effetti dell'applicazione di queste tecniche per verificare se si manifesta un miglioramento nelle prestazioni per alcuni classificatori.

PROGETTAZIONE

- Le **performance metrics** utilizzate per lo studio dell'accuratezza dei classificatori sono:

- Precision: mi da informazioni su quante volte il modello ha dato positivo ed effettivamente era positivo rispetto a tutte le volte che ha dato positivo→ TP / (TP + FP)
- Recall: mi da informazioni su quanti positivi è riuscito a classificare → TP / (TP + FN)
- AUC: area sottesa alla curva ROC.
- Kappa: mi da informazioni su quanto il classificatore si è comportato meglio rispetto ad un classificatore dummy.

PROGETTAZIONE: PASSI

- Per raggiungere lo scopo del progetto, sono stati seguiti i seguenti passi per entrambi i progetti:

1. Recupero delle informazioni necessarie da JIRA.
2. Recupero dei file per ogni release che appartiene alla prima metà delle release del progetto e recupero dei commit relativi ai vari bug.
3. Scelta e calcolo delle metriche per ogni classe.
4. Labeling dei training set e dei testing set necessari per il Walk-Forward.
5. Esecuzione del Walk-Forward.

PROGETTAZIONE: JIRA

- Le informazioni sulle release e sui BUG relative ai due progetti sono state ricavate utilizzando le API REST di JIRA. Ad esempio, per ottenere le informazioni sulle versioni dei due progetti ho utilizzato le seguenti api rest:

https://issues.apache.org/jira/rest/api/2/project/BOOKKEEPER/versions

https://issues.apache.org/jira/rest/api/2/project/ZOOKEEPER/versions

- Sono state scartate le release appartenenti alla seconda metà per via dello snoring.
- Per ottenere i BUG dei progetti sono stati utilizzati i seguenti filtri:

Type == 'defect' AND (status == 'Closed' OR status == 'Resolved') AND Resolution == 'Fixed'

- Sono stati scartati tutti i BUG che non hanno su JIRA la Fix Version, quelli che hanno la Opening Version inconsistente (e.g., OV > FV) e quelli che hanno le tre versioni in relazione tra di loro in modo scorretto (e.g., IV > OV > FV).

PROGETTAZIONE: FILE E COMMIT

- Una volta recuperate le release da JIRA, sono stati ottenuti i file java relativi a tali release. Per lo scopo del progetto, sono stati scartati tutti i file non java.
- Per ogni BUG, ho ricavato tutti i commit ad esso relativi. Per fare ciò, ho osservato che la maggior parte dei commenti nei commit contenevano l'informazione della key del BUG (e.g., BOOKKEEPER-123).
- La libreria utilizzata per interagire con git è JGIT:

https://www.eclipse.org/jgit/

PROGETTAZIONE: SCELTA DELLE METRICHE

- Con lo scopo di aiutare il classificatore, ho deciso di scegliere le seguenti metriche:

- **Numero di revisioni**: ragionevolmente, ho supposto che maggiore è il numero delle revisioni e più è probabile che la classe sia difettosa. Ho deciso di calcolare questa metrica partendo dalla release attuale.
- **Loc Added**: rappresenta il numero di linee di codice aggiunte. Ho deciso di calcolare questa metrica partendo dalla release attuale.
- **Max Loc Added**: mi permette di riconoscere i grandi cambiamenti fatti su una classe in una release.
- **Avg Loc Added:** mi da informazioni su quante linee di codice vengono aggiunte in media nelle varie revisioni in una release.
- **Size:** ragionevolmente, maggiore è il numero di righe di codice della classe e maggiore è la probabilità che ci possano essere bug all'interno della classe stessa.
- **Authors:** se ho molti sviluppatori che hanno partecipato all'implementazione della classe, allora la classe avrà stili di programmazione differenti fusi al suo interno.

PROGETTAZIONE: SCELTA DELLE METRICHE

- **ChgSetSize**: ragionevolmente, se una classe viene toccata nei commit insieme a molte altre classi, allora è più probabile che vengano iniettati dei bug. Ho deciso di calcolare questa metrica per release.
- **AvgChgSetSize:** mi dice quanti file mediamente vengono toccati insieme alla classe di riferimento nei vari commit per la release corrente.
- **MaxChgSetSize**: mi dice il numero massimo di file che sono stati toccati insieme alla classe di riferimento per la release corrente.
  - Le tre metriche della ChgSetSize sono a mio parere informazioni rilevanti da dare al modello. Il fatto che vengano modificate insieme più classi, potrebbe implicare (ma non è certo) un alto accoppiamento per la classe di riferimento.
  - Le metriche scelte sono correlate con la bugginess delle classi in una release. L'obiettivo è stato quello di scegliere delle metriche evitando di portare fuori strada il modello.

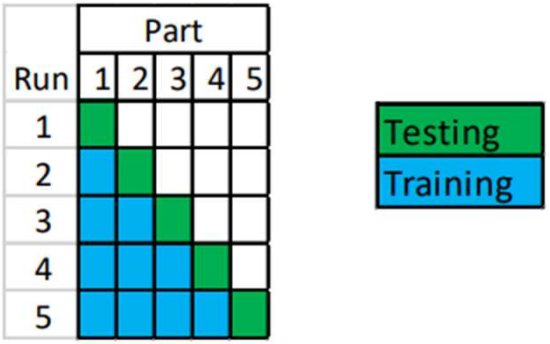
PROGETTAZIONE: LABELING

- Numerosi bug non hanno l'AV su JIRA e di conseguenza non posso determinare la loro IV. Per stimare il valore della IV ho utilizzato la tecnica **Proportion**.
- Nel progetto sono state utilizzate due varianti di proportion per stimare l'IV:

1. **Training Set Proportion**: dato un dataset di training, utilizzo tutte le informazioni che ho a disposizione all'interno del training set stesso. Non vengono utilizzate le informazioni che sono future al training set. Questa variante viene utilizzata nel processo di labeling delle classi delle release che fanno parte del training set.
2. **Global Proportion**: utilizzo tutte le informazioni che ho a disposizione con l'intento di ottenere una conoscenza che sia la più ampia possibile. Questa variante viene utilizzata nel processo di labeling delle classi della release che costituisce il testing set.

PROGETTAZIONE: WALK-FORWARD

- Nel progetto è stata utilizzata la tecnica del Walk-Forward. Il dataset viene diviso in **unità** che corrispondono alle **release** del progetto. Supponendo che per il progetto considerato abbiamo N release, allora il numero di esecuzioni è pari a N – 1, uno in meno rispetto al numero delle release.
- Ad ogni iterazione, il testing set è costituito da una singola release.
- Il training set viene incrementato ad ogni nuova iterazione andando ad aggiungere i dati relativi alla release che ha svolto il ruolo di testing set nella iterazione precedente.



DISCUSSIONE DEI RISULTATI: BOOKKEEPER

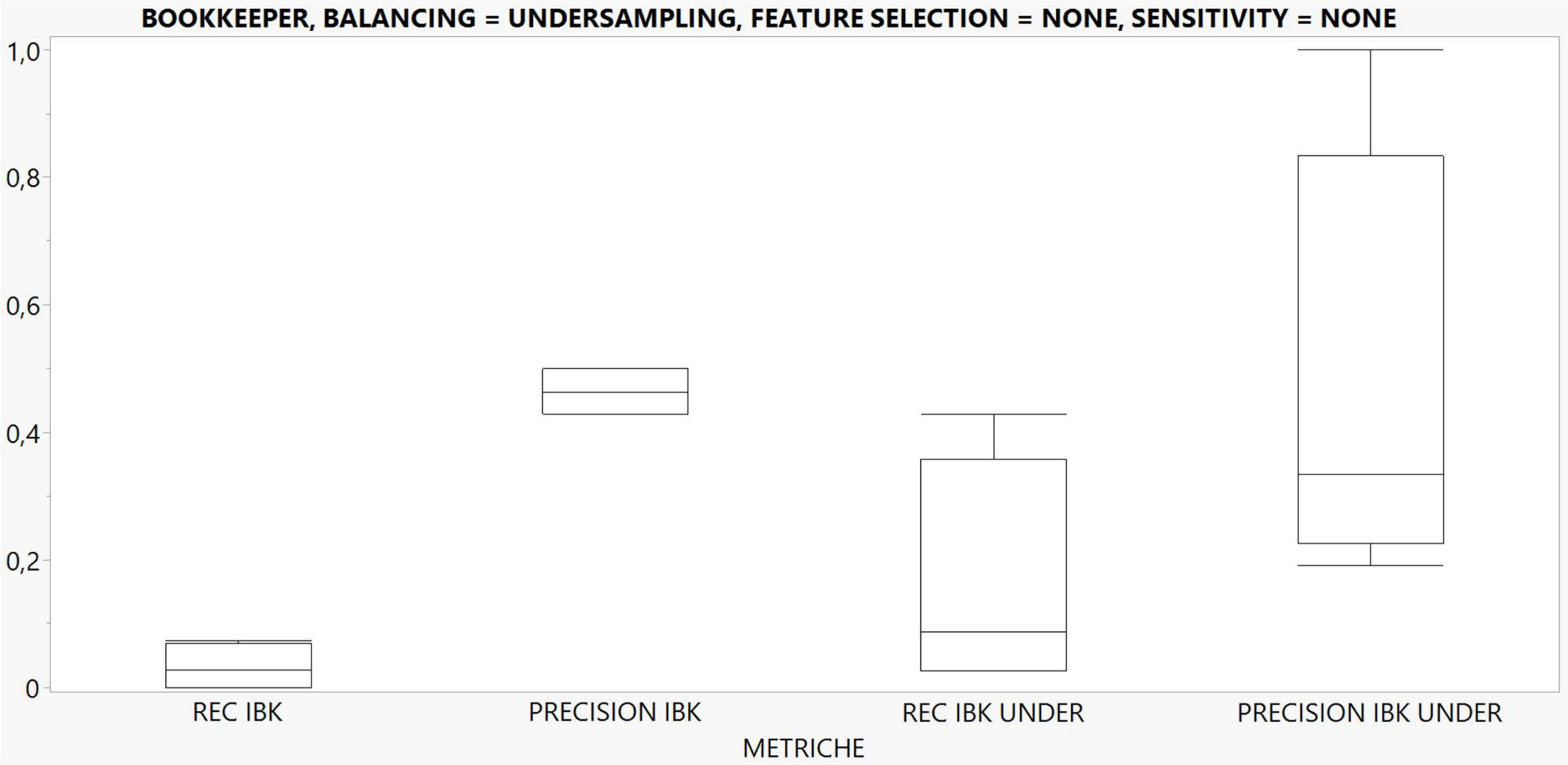
- La percentuale di classi difettose all'interno del training set nelle varie iterazioni del Walk-Forward è mediamente sotto al 10%. Di conseguenza, è ragionevole aspettarsi che i classificatori dicano quasi sempre negativo (classe non difettosa). Inoltre, osservo che la bassa percentuale di classi difettose all'interno del training set è dovuta principalmente allo snoring. Ho a che fare con un dataset molto rumoroso.
- Un altro svantaggio riscontrato per il progetto Apache BookKeeper è rappresentato dal basso numero di iterazioni nel Walk-Forward e di conseguenza sono pochi i valori che vado a rappresentare graficamente.
- Dall'analisi dei risultati **senza l'applicazione di alcuna tecnica** si osserva che i classificatori dicono quasi sempre negativo se non per un numero molto basso di volte, principalmente nelle ultime iterazioni. Probabilmente, ciò è dovuto al fatto che nelle iterazioni finali il numero delle classi difettose all'interno del training set è maggiore e il classificatore è portato a dire più volte positivo. In generale, il valore della Recall è molto basso per tutti e tre i classificatori ma considerando insieme anche il valore della Precision osservo che i classificatori IBK e Naive Bayes si comportano meglio rispetto a Random Forest. IBK è il classificatore che dice più volte positivo nelle varie iterazioni ed è quello con la Recall maggiore tra i tre classificatori.

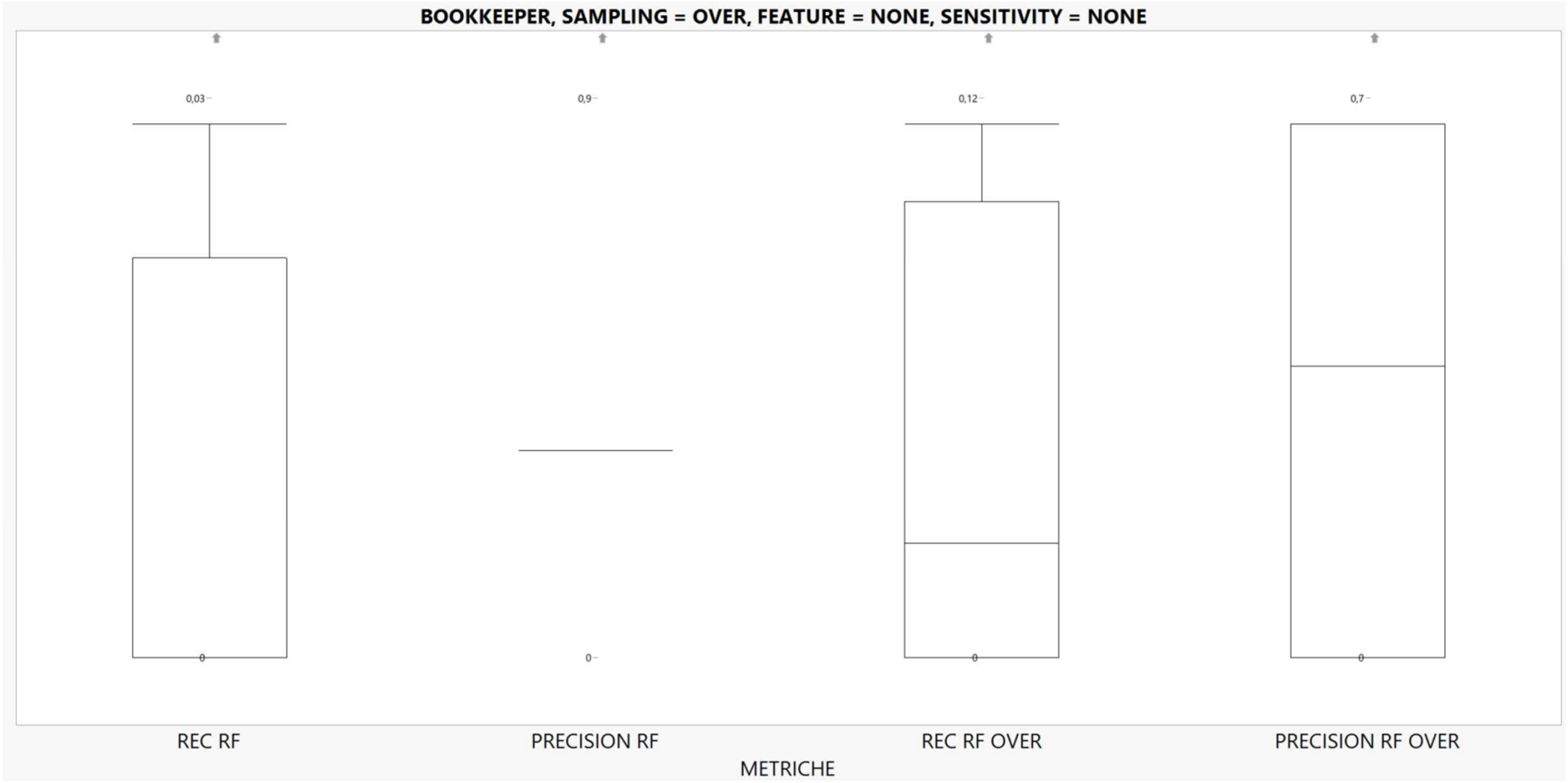
DISCUSSIONE DEI RISULTATI: BOOKKEEPER

- Siccome il training set nelle varie iterazioni è molto sbilanciato, ho deciso di applicare le tecniche di balancing UNDERSAMPLING e OVERSAMPLING. Dall'applicazione di queste tecniche, le prestazioni di alcuni classificatori potrebbero migliorare in quanto con entrambe si arriva ad avere un dataset bilanciato. Dall'analisi dei risultati osservo che per tutti i classificatori il numero dei positivi stimati è aumentato. Questo è il risultato che mi aspettavo in quanto nella fase di addestramento viene passato al modello un training set che ha una percentuale di positivi maggiore.

- Come mostrato nella slide successiva, con la tecnica di **UNDERSAMPLING** sono riuscito a migliorare la Recall e la Precision per alcune iterazioni per il classificatore IBK rispetto al caso senza alcuna tecnica. Con le etichette **«qualcosa» UNDER** rappresento i risultati nel caso dell'applicazione dell'UNDERSAMPLING mentre le rimanenti sono relative al caso senza alcuna tecnica. Inoltre, a differenza del caso in cui non viene applicata alcuna tecnica, per IBK sono riuscito ad avere valori diversi da zero per le due metriche in tutte le iterazioni del Walk-Forward. Sono riuscito a migliorare la Recall e la Precision in alcune iterazioni per quanto riguarda il classificatore Random Forest. Mentre nel caso del classificatore Naive Bayes, il valore delle due metriche nell'insieme è peggiorato nel 75% delle iterazioni e il valore della KAPPA è diventato negativo in alcune iterazioni.

- Come mostrato nella slide 16, con la tecnica di **OVERSAMPLING** sono riuscito a migliorare le metriche Recall e Precision per il classificatore Random Forest (RF) nelle singole iterazioni. Con le etichette **«qualcosa» OVER** rappresento i risultati per l'OVERSAMPLING mentre le rimanenti sono relative al caso senza alcuna tecnica. Inoltre, è possibile osservare anche un miglioramento per il classificatore Naive Bayes mentre per il classificatore IBK il valore della Recall nelle varie iterazioni rimane inalterato e il valore della Precision diminuisce poiché vengono detti più positivi ma sono tutti FP.


BOOKKEEPER, BALANCING = UNDERSAMPLING, FEATURE SELECTION = NONE, SENSITIVITY = NONE
1,0
0,8
0,6
0,4
0,2
0
REC IBK
PRECISION IBK
REC IBK UNDER
PRECISION IBK UNDER
METRICHE


BOOKKEEPER, SAMPLING = OVER, FEATURE = NONE, SENSITIVITY = NONE
0,03
0
0,9
0
0,12
0
0,7
0
REC RF
PRECISION RF
REC RF OVER
PRECISION RF OVER
METRICHE

DISCUSSIONE DEI RISULTATI: BOOKKEEPER

- Applicando la tecnica della **feature selection** ho avuto un peggioramento generale delle prestazioni dei classificatori. Probabilmente questo risultato è dovuto ad una scelta del sottoinsieme delle feature che non è quello ottimale. In ogni iterazione il valore della Recall risulta pari a zero sia per Random Forest che per Naive Bayes e i pochi positivi stimati sono stati tutti FP. Inoltre, per i due classificatori la Precision o vale zero oppure il classificatore non ha mai stimato un positivo nella iterazione. La Recall per IBK è diminuita in tutte le iterazioni rispetto al caso senza alcuna tecnica applicata. In generale, noto un peggioramento nelle iterazioni anche per la Precision di IBK.
- Applicando le tecniche di sensitivity, mi aspetto che il numero di positivi stimati cresca in alcune iterazioni e che migliorino le prestazioni di alcuni classificatori. Effettivamente, applicando entrambe le tecniche di sensitivity, **Sensitive Learning** e **Sensitive Threshold**, il numero dei positivi stimati è aumentato notevolmente.

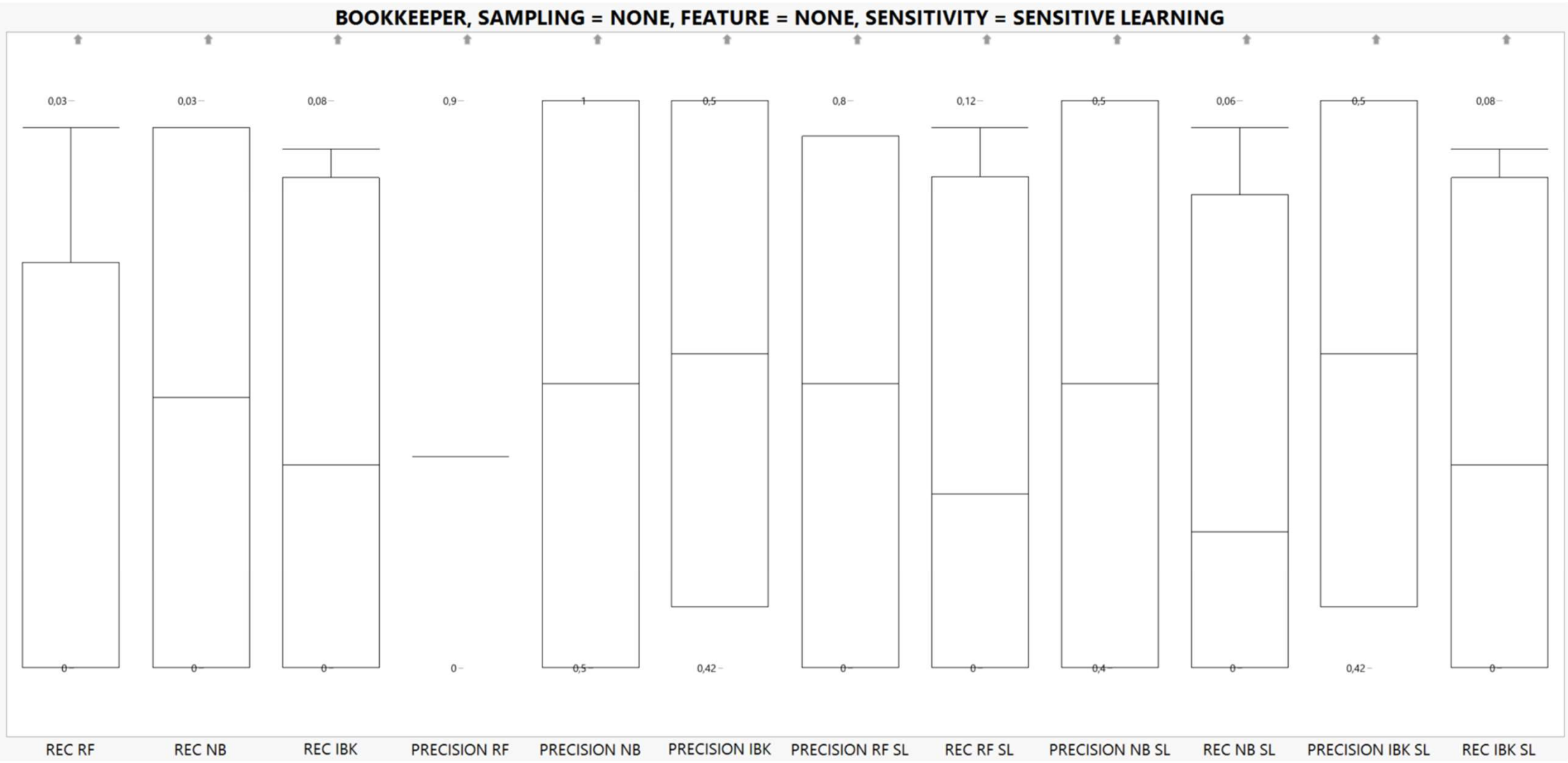
DISCUSSIONE DEI RISULTATI: BOOKKEEPER

- Nella slide successiva è raffigurato un box plot in cui le metriche **«qualcosa» SL** sono quelle relative all'applicazione della tecnica di **Sensitive Learning** mentre le altre sono relative al caso in cui non è stata applicata alcuna tecnica. Dal confronto viene fuori che:

- La Recall e la Precision per Random Forest sono aumentate in quasi tutte le iterazioni e in nessuna iterazione ho avuto un peggioramento. In generale, è possibile osservare un aumento del valore della KAPPA nelle varie iterazioni.

- La Recall per Naive Bayes mediamente non peggiora e aumenta in alcune iterazioni. Tuttavia, noto un calo per quanto riguarda la Precision. Nonostante il valore della Recall sia leggermente aumentato, a mio parere tale aumento non è sufficiente se considerata la diminuzione che si è avuta della Precision. Il valore della KAPPA mediamente rimane lo stesso nelle varie iterazioni.

- La Recall, la precision e la KAPPA per il classificatore IBK rimangono esattamente le stesse.


BOOKKEEPER, SAMPLING = NONE, FEATURE = NONE, SENSITIVITY = SENSITIVE LEARNING
0,03
0
0,03
0
0,08
0
0,9
0
1
0,5
0,5
0,42
0,8
0
0,12
0
0,5
0,4
0,06
0
0,5
0,42
0,08
0
REC RF
REC NB
REC IBK
PRECISION RF
PRECISION NB
PRECISION IBK
PRECISION RF SL
REC RF SL
PRECISION NB SL
REC NB SL
PRECISION IBK SL
REC IBK SL

DISCUSSIONE DEI RISULTATI: ZOOKEEPER

- Nel progetto Apache ZooKeeper osservo che la percentuale di classi difettose nel training set per le varie iterazioni è molto inferiore rispetto alla percentuale di classi difettose che ho nel testing set della rispettiva iterazione. Questo potrebbe rappresentare un problema per il classificatore in quanto la distribuzione delle classi difettose non è la stessa all'interno dei due dataset. Inoltre, ad eccezione delle ultime tre iterazioni, la percentuale di classi difettose all'interno del training set è di molto inferiore all'1%. Dati questi due fatti, senza applicare alcuna tecnica mi aspetto che il classificatore stimi un bassissimo numero di positivi e che le prestazioni del classificatore per la classe minoritaria siano molto basse.
- **Senza applicare alcuna tecnica**, osservo che i classificatori IBK e Random Forest stimano zero positivi in tutte le iterazioni ad eccezione dell'ultima e di conseguenza il valore della Precision non può essere calcolato per queste iterazioni. In tutte queste iterazioni il valore della KAPPA è pari a zero. Per quanto riguarda il classificatore Naive Bayes, a differenza degli altri due, nelle ultime iterazioni riesce ad avere dei buoni valori per la Recall e per la Precision con la KAPPA che riesce ad avvicinarsi al valore 0,58. Tra i tre classificatori, Naive Bayes è quello più accurato senza l'applicazione di alcuna tecnica.

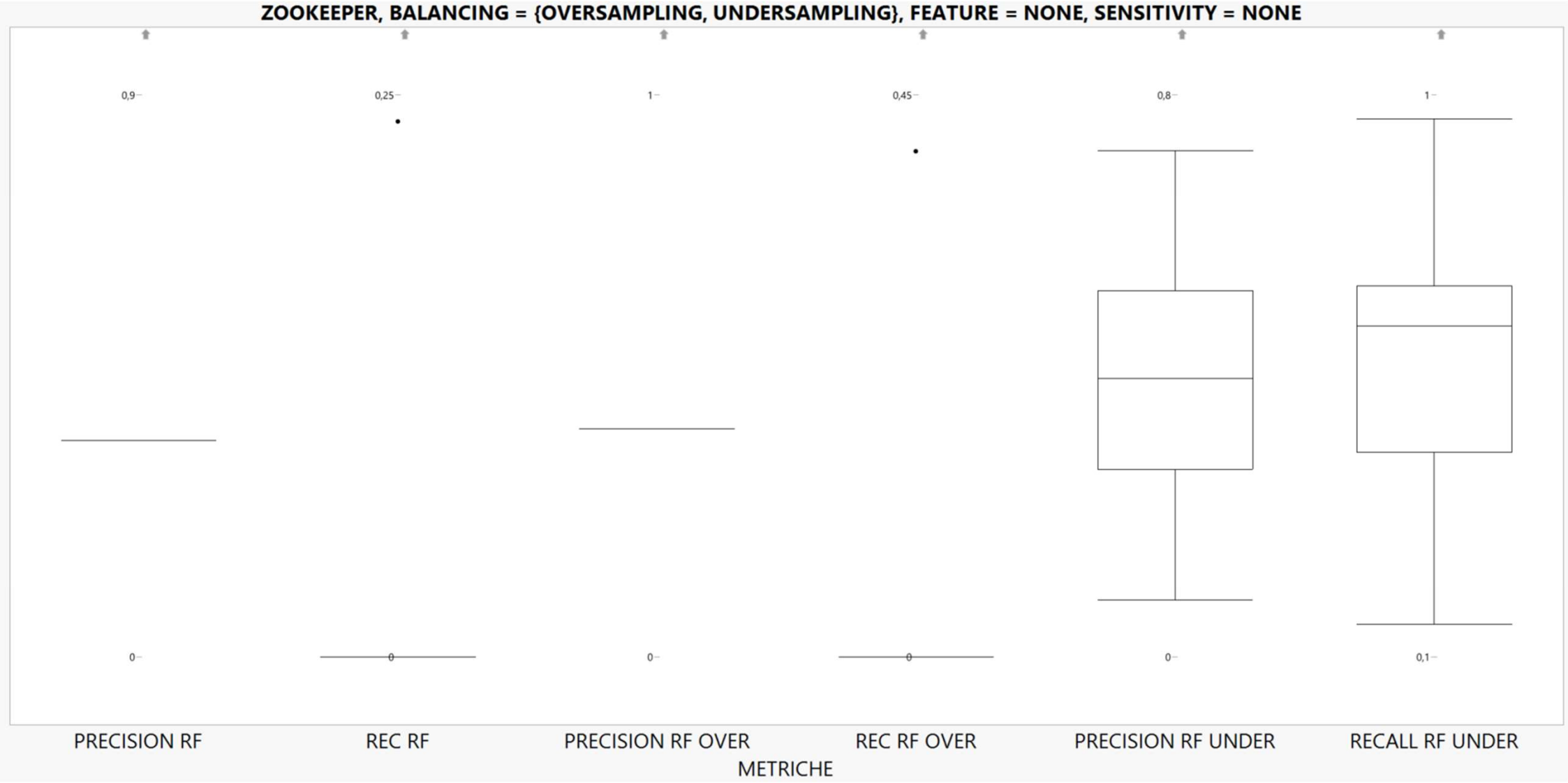
DISCUSSIONE DEI RISULTATI: ZOOKEEPER

- Siccome il training set è molto sbilanciato in tutte le iterazioni, ho deciso di applicare le tecniche di BALANCING. Nella slide successiva viene mostrato un box plot relativo al classificatore Random Forest (RF) in cui le etichette **«qualcosa» OVER** e **«qualcosa» UNDER** rappresentano rispettivamente le metriche nel caso dell'utilizzo della tecnica di OVERSAMPLING e dell'utilizzo della tecnica di UNDERSAMPLING. Le altre due etichette rappresentano il caso senza alcuna tecnica. Posso fare le seguenti considerazioni:

- Seppur per una sola iterazione, nel caso dell'OVERSAMPLING la Recall, la precision e la KAPPA aumentano. Questo significa che il classificatore riesce a prendere più positivi con una maggiore precisione rispetto al caso senza alcuna tecnica applicata, in quella iterazione.

- Nel caso dell'UNDERSAMPLING, sono riuscito ad avere valori diversi da zero sia per la Recall che per la Precision in tutte le iterazioni. Inoltre, osservo che ci sono iterazioni in cui i valori della Recall e della Precision sono abbastanza buoni. La tecnica dell'UNDERSAMPLING si è rivelata buona per migliorare le prestazioni del classificatore Random Forest.

- Una situazione molto simile a quella riscontrata con Random Forest si ha con IBK. Invece per Naive Bayes, considerando la tecnica di OVERSAMPLING, il valore della Precision diminuisce in ogni iterazione mentre aumenta il valore della Recall. Per alcune iterazioni potrebbe essere ragionevole accettare l'incremento della Recall a discapito della relativa diminuzione della Precision.

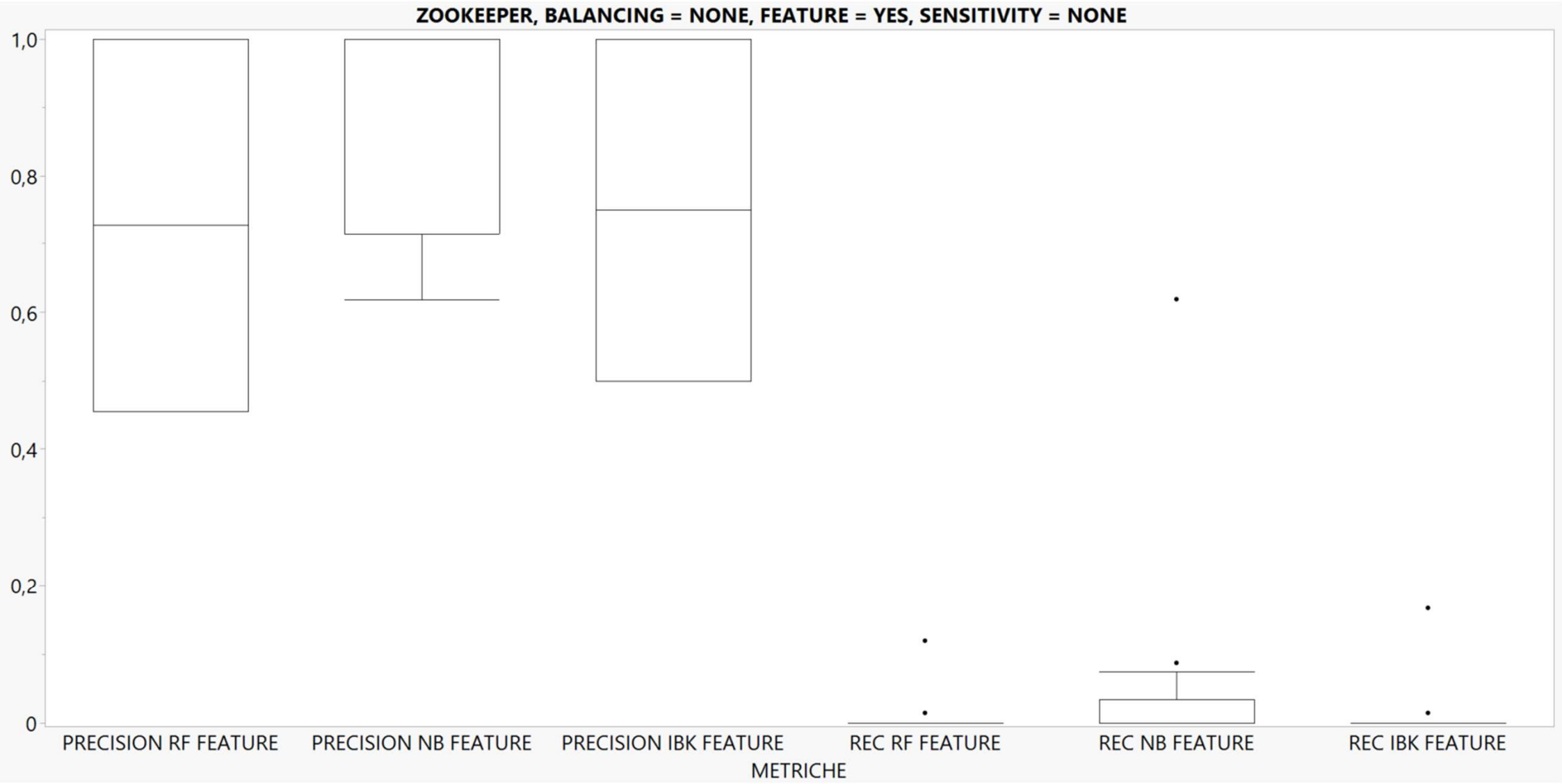

ZOOKEEPER, BALANCING = {OVERSAMPLING, UNDERSAMPLING}, FEATURE = NONE, SENSITIVITY = NONE
0,9
0
0,25
0
1
0
0,45
0
0,8
0
1
0,1
PRECISION RF
REC RF
PRECISION RF OVER
REC RF OVER
PRECISION RF UNDER
RECALL RF UNDER
METRICHE

DISCUSSIONE DEI RISULTATI: ZOOKEEPER

- Per il progetto Apache ZooKeeper ho deciso di applicare la tecnica di Feature Selection. Nella slide successiva è possibile vedere gli effetti dell'applicazione di tale tecnica. Per prima cosa viene rappresentata la Precision per i tre classificatori e successivamente viene rappresentata la Recall, in tutti i casi avendo applicato la feature selection.
- Posso fare le seguenti osservazioni:

- In generale, l'applicazione della feature selection ha migliorato la Precision nelle varie iterazioni per tutti e tre i classificatori. La Precision di IBK e di Naive Bayes sono quelle che risentono maggiormente in positivo dell'applicazione della tecnica.
- In generale, la Recall di Random Forest e di Naive Bayes risentono negativamente dell'applicazione della tecnica. Al contrario, ho un piccolo miglioramento per la Recall di IBK.

- In conclusione, con la feature selection riesco a migliorare la Precision per Random Forest e Naive Bayes a discapito di una riduzione della Recall. Al contrario, per IBK riesco ad aumentare la Precision e la Recall.


ZOOKEEPER, BALANCING = NONE, FEATURE = YES, SENSITIVITY = NONE
1,0
0,8
0,6
0,4
0,2
0
PRECISION RF FEATURE
PRECISION NB FEATURE
PRECISION IBK FEATURE
REC RF FEATURE
REC NB FEATURE
REC IBK FEATURE
METRICHE

LINKS

- Link del progetto per Sonarcloud con zero Code Smells:

Deliverable1 - luca-capotombolo (sonarcloud.io)

Nel caso il link sopra non funzionasse, di seguito è riportata la URL per SonarCloud →

https://sonarcloud.io/project/overview?id=luca-capotombolo_Deliverable1

- Link del codice del progetto per Github:

luca-capotombolo/Deliverable1 (github.com)

Nel caso il link sopra non funzionasse, di seguito è riportata la URL del progetto →

https://github.com/luca-capotombolo/Deliverable1