



UNIVERSITA' DI TOR VERGATA

# Machine learning for software engineering

---

**Martorelli Luca**

Matricola 0354445

**Anno accademico 2023-24**

# Agenda

---

- Contesto
- Scopo e Obiettivi
- Metodologia
- Risultati e Valutazioni
- Conclusioni
- Minacce alla Validità

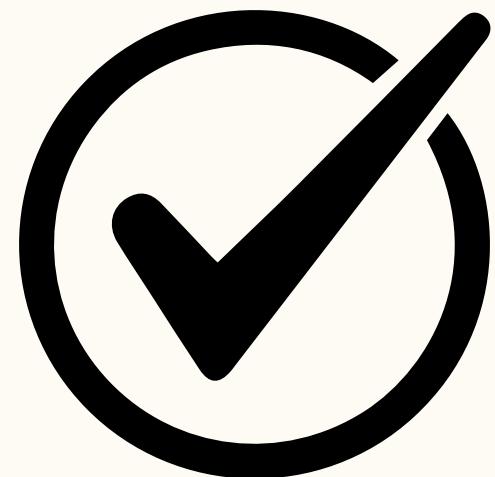
# Contesto

- Attualmente, ogni progetto di Ingegneria del Software necessita di attività di testing del software. Tuttavia, questo processo sta diventando sempre più costoso in termini di risorse.



È necessario individuare un sottoinsieme di classi da testare. Ma come stabilire quali classi hanno più probabilità di avere malfunzionamenti?

# Contesto



## Usando il Machine Learning!

Il machine learning cu permette di ottenere previsioni dettagliate e accurate sulla bugginess delle diverse classi, utilizzando dati del passato.

---

In questo modo, con l'uso di particolari algoritmi, detti classificatori, è possibile predire quali classi contengono difetti e concentrarsi soltanto su di loro.

# Scopo

Analisi di due progetti di Apache Software Foundation:

- BookKeeper
- OpenJPA

## Obiettivi

- Misurazione dell'accuratezza, attraverso l'utilizzo di tre classificatori.
- Sperimentare diverse tecniche al fine di migliorare la qualità delle predizioni.
- Stabilire come e in che misura le tecniche utilizzate hanno influenzato le predizioni.

# Metodologia

- Raccolta di **metriche**.
- Costruzione dei **dataset** di training e testing.
- Applicazione di algoritmi di Machine Learning per eseguire **predizioni**:
  - Attraverso l'uso di tre **classificatori**:
    1. Random Forest
    2. Naive Bayes
    3. IBK
  - Attraverso combinazione di tre **tecniche**:
    1. Feature Selection
    2. Balancing
    3. Cost Sensitive

# Metodologia

## Misurazioni 1

Il processo di misurazione è stato condotto utilizzando principalmente due strumenti:

- **Jira:** un sistema di tracciamento delle problematiche progettato per la raccolta e la gestione dei ticket relativi alle diverse versioni del software.
- **Git:** un sistema di controllo delle versioni distribuito ampiamente utilizzato per la gestione del codice sorgente.

# Metodologia

## Misurazioni 2

Le AV non sono sempre presenti in modo completo sui ticket estratti da Jira:

- Si utilizza il calcolo delle AV attraverso la tecnica del **proportion incremental**, basata sulla media delle proportion dei ticket disponibili fino a quel momento.
- Se non ce ne fossero abbastanza (impostando una threshold di '5'), si adotta l'approccio **cold-start**, prendendo la mediana tra i valori medi delle proporzioni calcolate sui ticket consistenti (con informazioni sulle AV) di altri progetti Apache ritenuti significativi per questo tipo di stima.

# Metodologia

## Misurazioni 3

La tecnica del proportion viene impiegata per stimare l'IV di un bug. Si è notato che il tempo tra IV e OV tende ad essere proporzionale al tempo tra OV e FV:

$$p = \frac{FV - IV}{FV - OV} \quad IV = FV - (FV - OV) * p$$

# Metodologia

## Metriche 1

<b>LinesOfCode</b>	Dimensione in LOC della classe.
<b>AddedLOC*</b>	Somma delle LOC aggiunte.
<b>TouchedLOC</b>	Somma delle LOC modificate.
<b>Churn*</b>	Fattore di aggiunta e rimozione di LOC.
<b>NumberOfAuthors</b>	Numero di autori.
<b>NumberOfRevisions</b>	Numero di revisioni della classe.
<b>Buggy</b>	Metrica binaria relativa alla bugginess della classe nella release considerata.

\*calcolato anche i valori \_ max e \_ avg, ma non sono stati inseriti nella tabella per mantenere la leggibilità

# Metodologia

## Metriche 2

La prima metà delle metriche riguarda principalmente le **loc**. Questa scelta è stata fatta anche per osservare l'effetto della tecnica di **feature selection** in questi casi.

L'ultima metrica relativa alla **bugginess** viene calcolata nel seguente modo:

- Vengono presi tutti i ticket di tipo 'bug' con risoluzione 'fixed' e con stato 'closed' o 'resolved' da jira.
- Ogni classe java modificata da un commit collegato a uno dei ticket estratti viene etichettata come buggy per tutte le versioni interessate.

# Metodologia

## Dataset 1

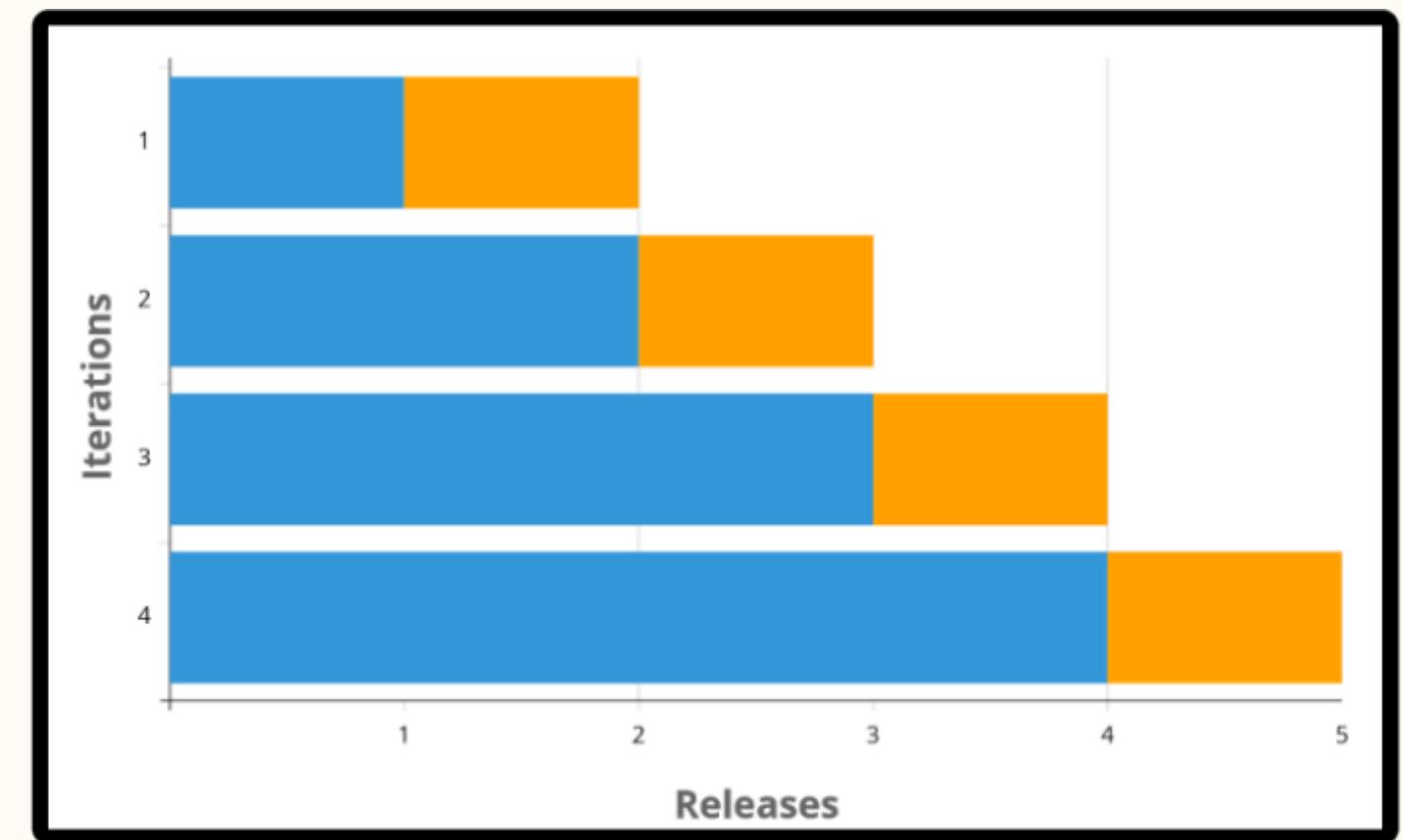
- I classificatori devono essere addestrati utilizzando **training datasets**, che rappresenta le informazioni fornite al sistema per "addestrarlo" e permettergli di "imparare" a effettuare previsioni.
- Per valutare i classificatori, il sistema verifica la precisione delle predizioni utilizzando **testing datasets**.

# Metodologia

## Dataset 2

Per costruire i set utilizziamo la tecnica **walkforward**, una tecnica time-series molto diffusa dove il dataset viene diviso in “parti” che vengono ordinate **cronologicamente**:

- Costruzione del **Training Set**: si utilizzano le prime 'K' release in ogni iterazione, con labeling basato sulle informazioni disponibili fino a quel momento.
- Costruzione del **Testing Set**: contiene le informazioni delle classi della release  $k+1$ -esima, con labeling basato su tutte le informazioni disponibili.



# Metodologia

## Predizioni

Per come abbiamo costruito i dataset:

- Il training set sarà soggetto a eventuali disturbi da **snoring** e su di esso verranno applicate diverse tecniche di **selection** e **balancing**.
- Al contrario, il testing set non presenterà disturbi da **snoring** e sfrutterà tutti i dati di cui siamo a conoscenza, senza subire alcuna trasformazione.

---

Per ridurre gli effetti dello snoring, è stata rimossa dal dataset la metà delle release più recenti prima della valutazione da parte del classificatore.

# Risultati

Si mettono a confronto i risultati ottenuti dai classificatori attraverso varie metriche: **Precision**, **Recall**, **KAPPA**, **AUC** e **Npofb20**. Questo confronto rende evidenti le differenze tra le tecniche utilizzate.

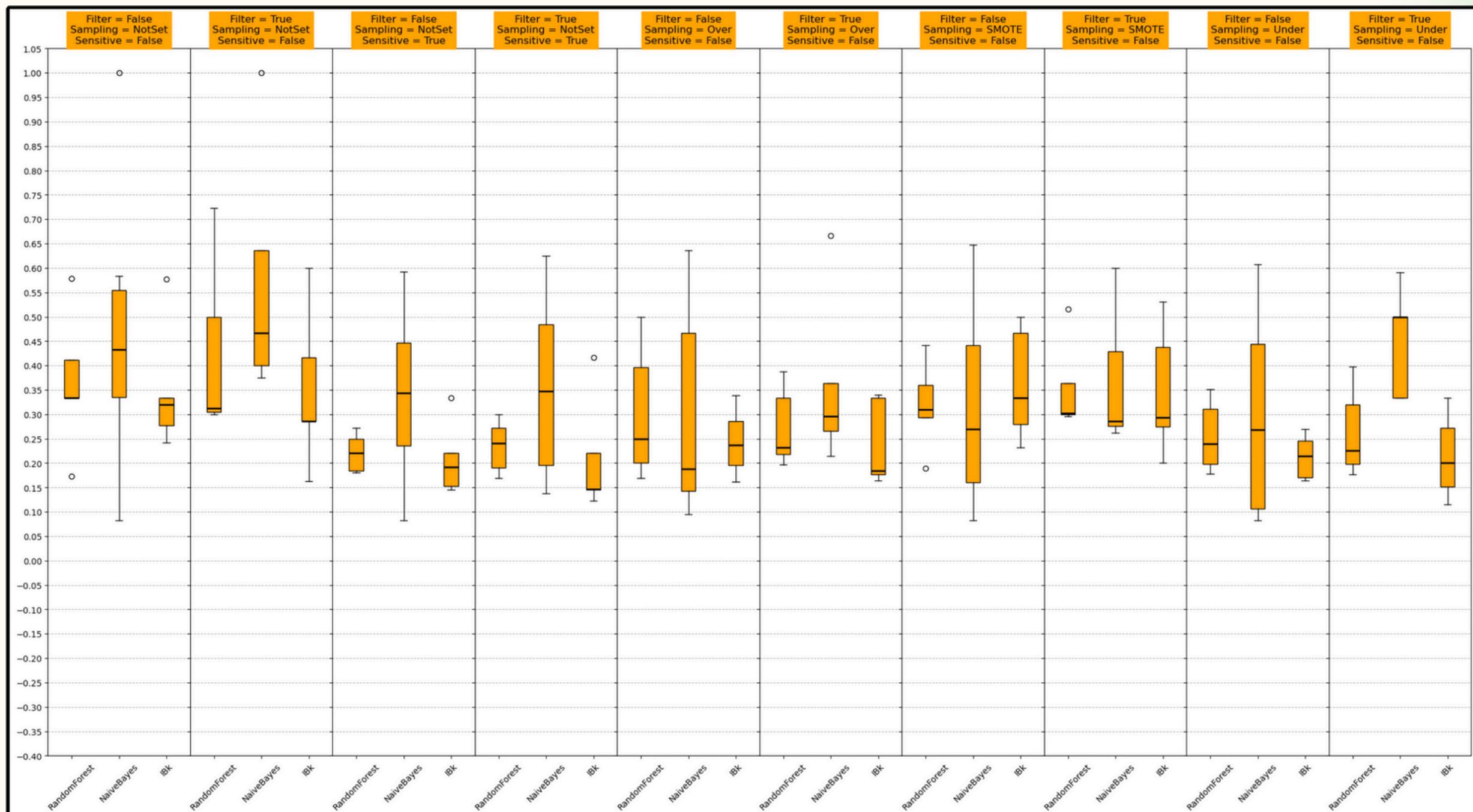
Considerando, inoltre, le diverse tipologie di classificatori, descritte pocanzi, Random Forest, Naive Bayes e IBk, sono state scelte alcune tipologie, per ogni tecnica:

- **Feature selection**: Best first backward
- **Sampling**: undersampling, oversampling e SMOTE
- **Cost Sensitive**: Sensitive Learning con le matrici dei costi ( $CFN = 10 * CFP$ )

# Risultati - BOOKKEEPER

## Precision

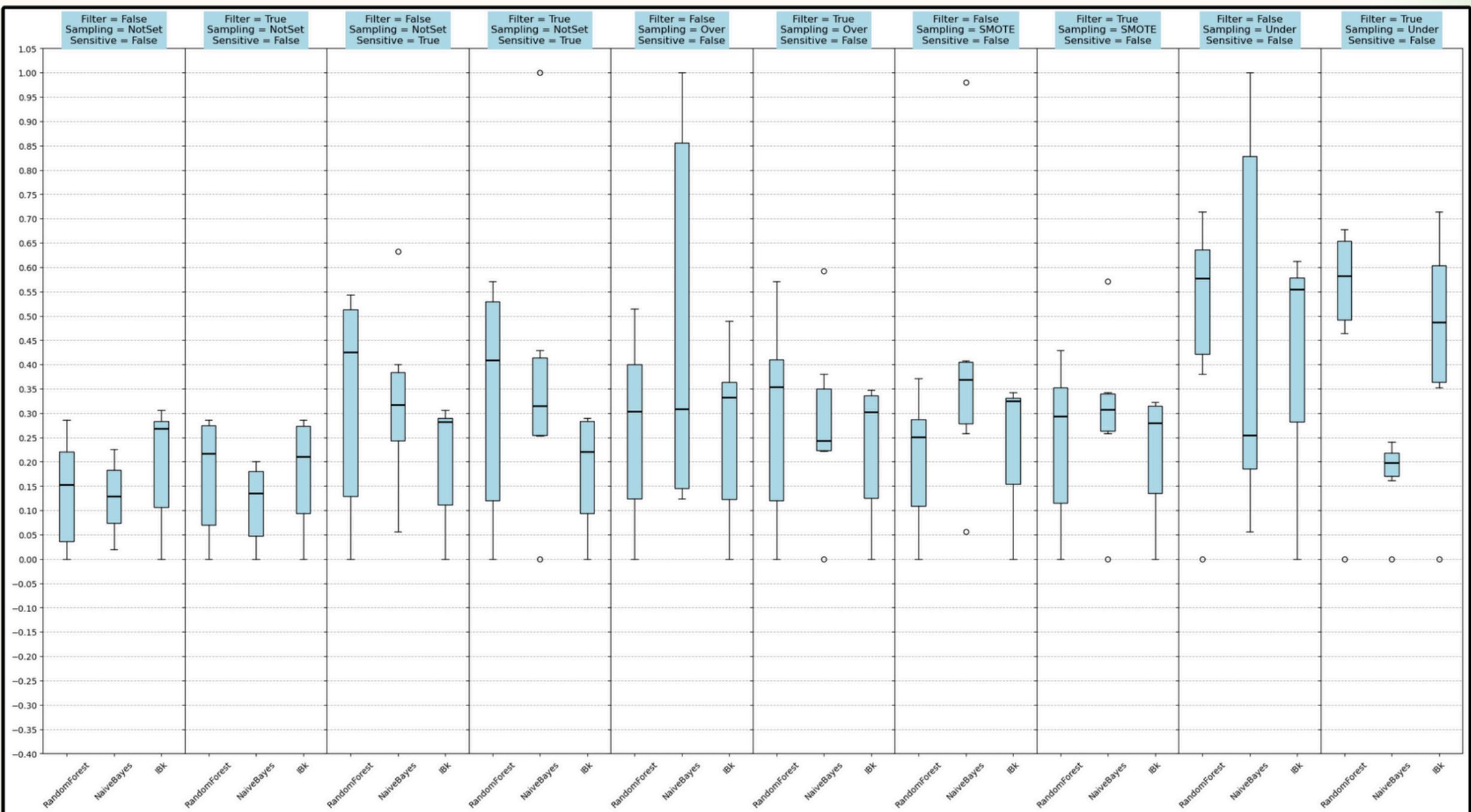
- **Feature selection:** migliora la precision del classificatore Naive Bayes; ma peggiora quella di IBK e di Random Forest.
- **Oversampling:** riduce la precisione di tutti i classificatori, e con **feature selection** si registra un decremento dei classificatori, tranne che per Naive Bayes.
- **Smote:** peggiora soprattutto la precisione in Naive Bayes, con l'aggiunta di **feature selection**, la precision peggiora per IBK.
- **Undersampling:** peggiora la precisione all'unanimità, ma con l'inclusione di **feature selection** migliora in modo significativo quella di Naive Bayes.
- **Sensitive learning:** la precisione crolla vertiginosamente, e combinando **feature selection**, la situazione non cambia.



# Risultati - BOOKKEEPER

## Recall

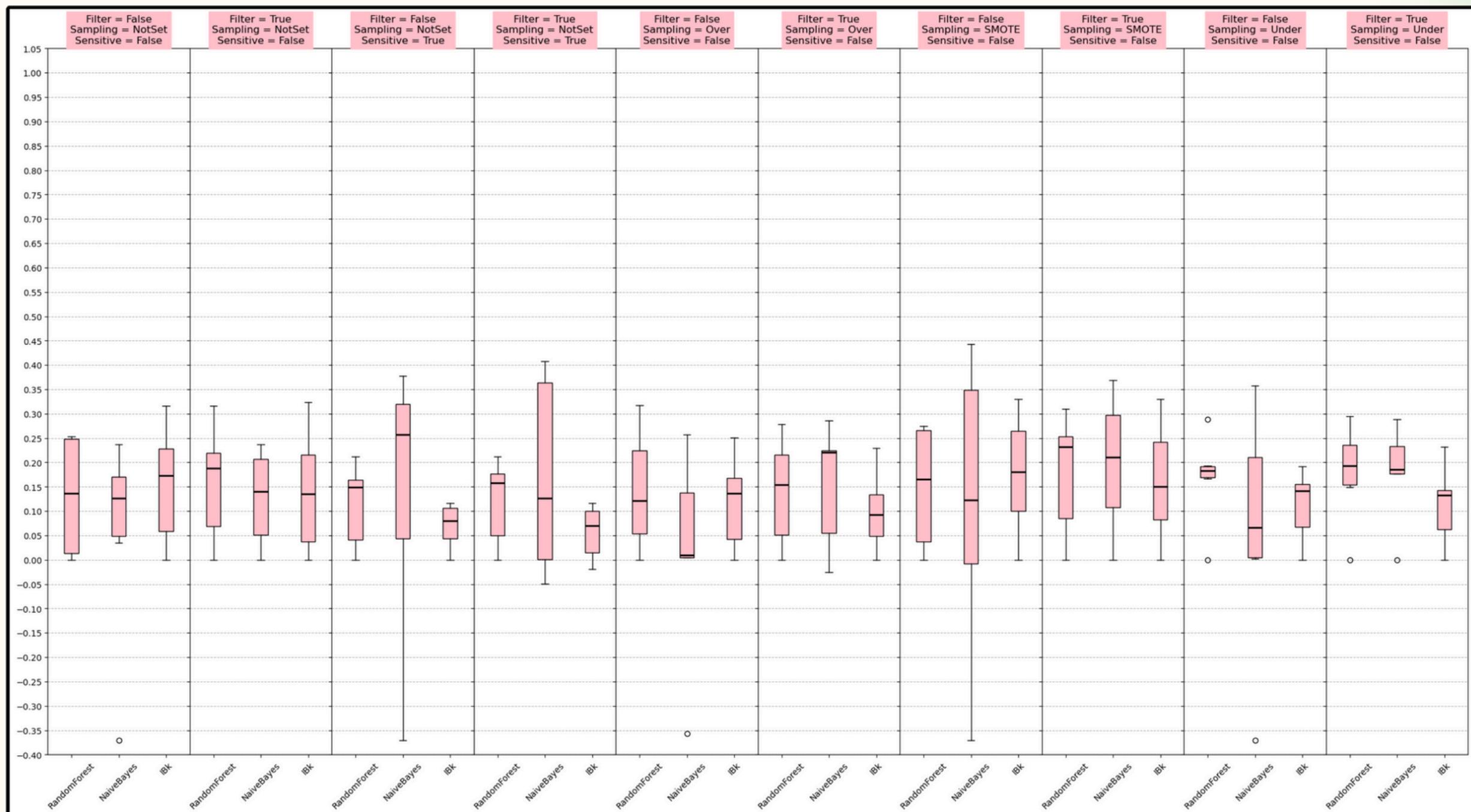
- **Feature selection:** migliora la recall del classificatore Random Forest, peggiora quella di IBK; mentre Naive Bayes resta invariata.
- **Oversampling:** aumenta la recall di tutti i classificatori; combinando **feature selection** si constata un ulteriore incremento per Random Forest, ma un leggero calo per gli altri classificatori.
- **Smote:** registra gli stessi effetti del caso precedente (Oversampling).
- **Undersampling:** aumenta notevolmente la recall dei classificatori, ma con l'inclusione di **feature selection** il miglioramento iniziale cala all'unanimità.
- **Sensitive learning:** la recall di Naive Bayes e Random Forest aumenta sensibilmente, ma con l'inclusione di **feature selection** la crescita iniziale si ridimensiona.



# Risultati - BOOKKEEPER

## KAPPA

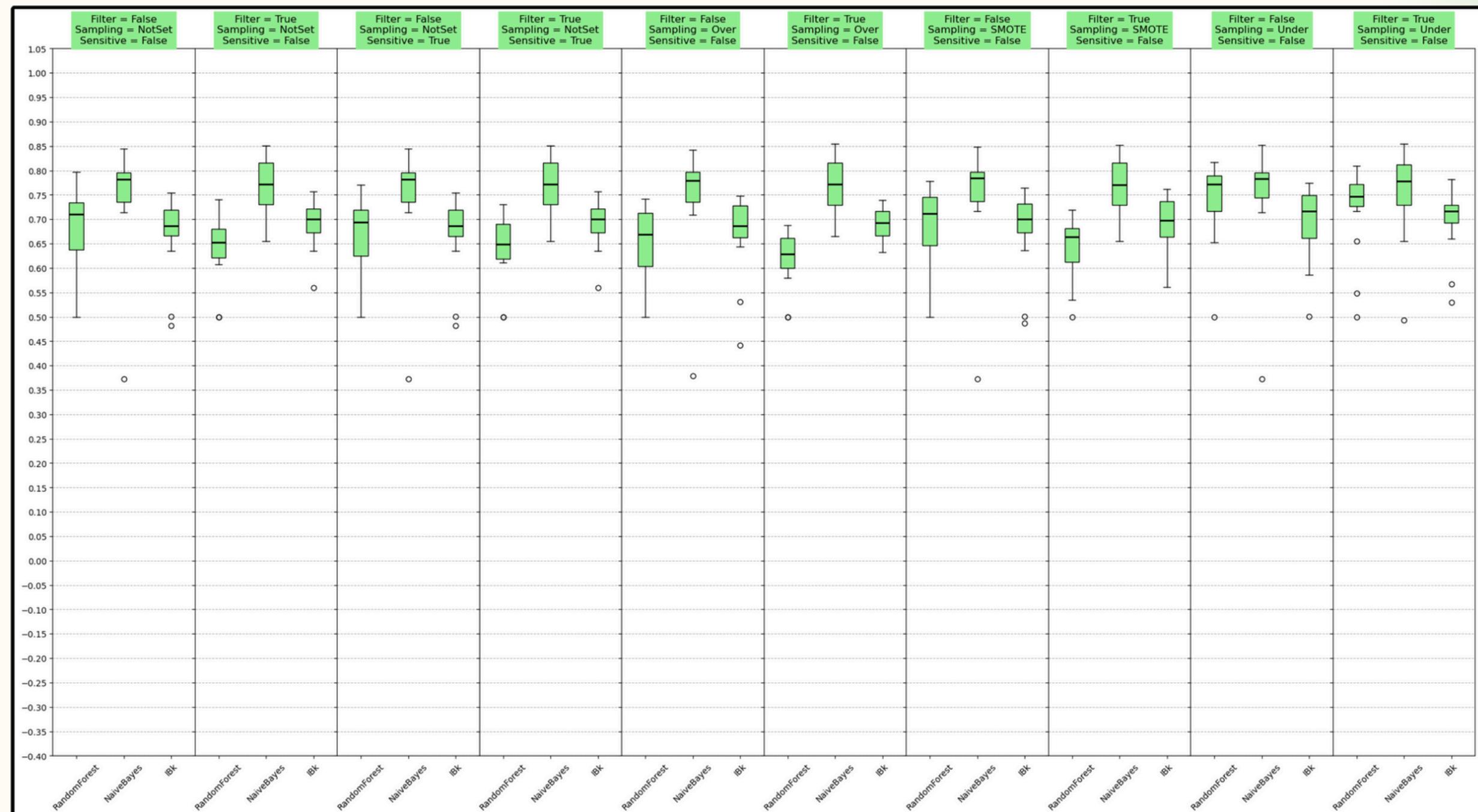
- **Feature selection:** migliora il valore di Kappa di Random Forest, ma peggiora quella di IBK; mentre Naive Bayes resta invariata.
- **Oversampling:** diminuisce il valore di Kappa di tutti i classificatori e, con l'inclusione di **feature selection** si nota una leggera ripresa.
- **Smote:** registra gli stessi effetti del caso precedente (Oversampling).
- **Undersampling:** aumenta il valore di Kappa di Random Forest, peggiorandolo però per gli altri due classificatori, ma combinando **feature selection** notiamo un netto miglioramento per Naive Bayes.
- **Sensitive learning:** il valore di Kappa di Naive Bayes e Random Forest aumenta, ma con l'inclusione di **feature selection** Naive Bayes e IBk peggiorano, mentre Random Forest aumenta ulteriormente.



# Risultati - BOOKKEEPER

## AUC

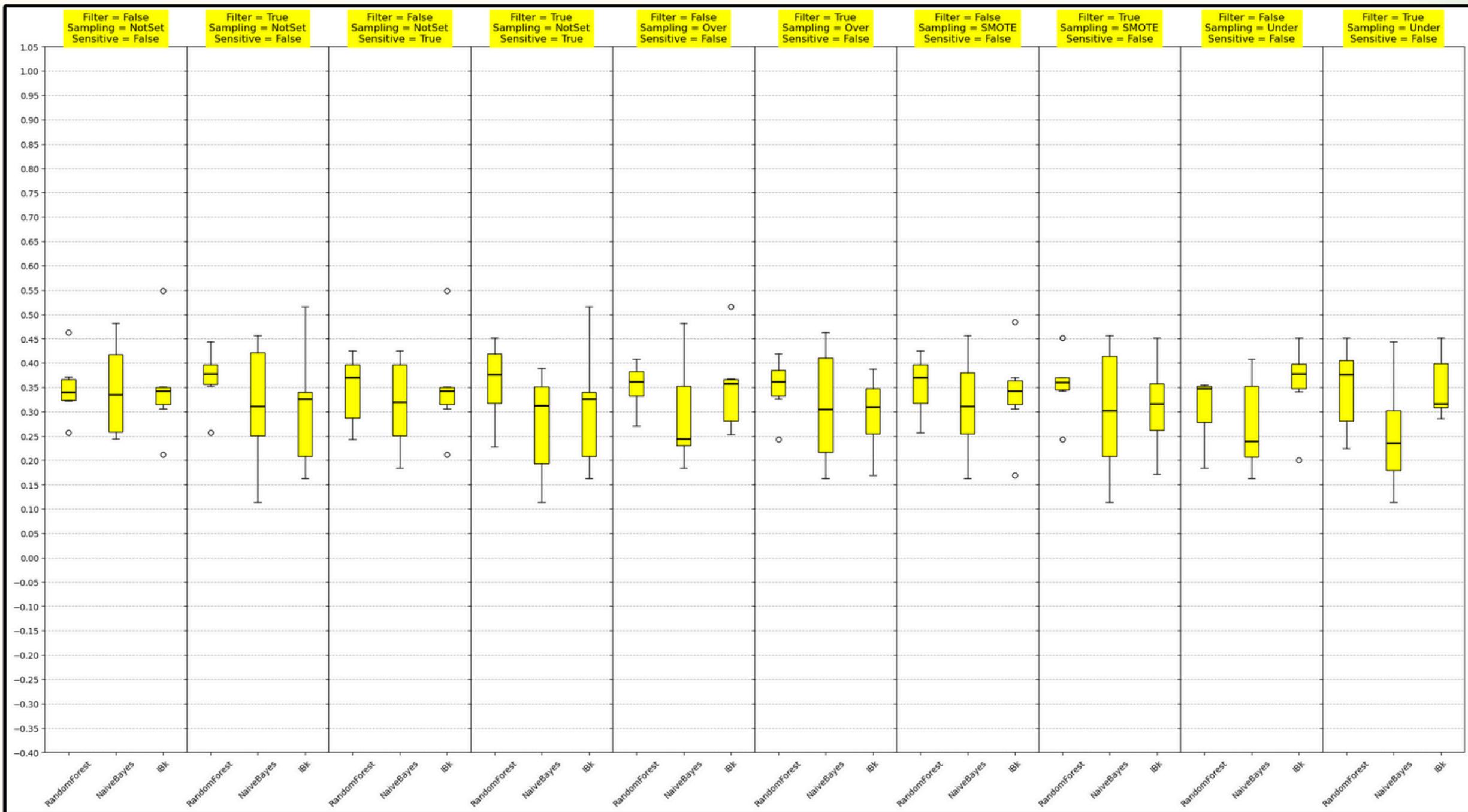
- **Feature selection:** migliora il valore di AUC di IBK, ma peggiora quella di Random Forest e Naive Bayes.
- **Oversampling:** diminuisce il valore di AUC di Random Forest, lasciando invariati gli altri classificatori e, includendo **feature selection** si verifica un ulteriore peggioramento su tutti i classificatori.
- **Smote:** registra i medesimi valori di AUC per tutti i classificatori, ma combinandolo con **feature selection** si vede come tali valori subiscono un leggero peggioramento.
- **Undersampling & Sensitive learning:** registrano i medesimi effetti del caso precedente (Smote).



# Risultati - BOOKKEEPER

## Npofb20

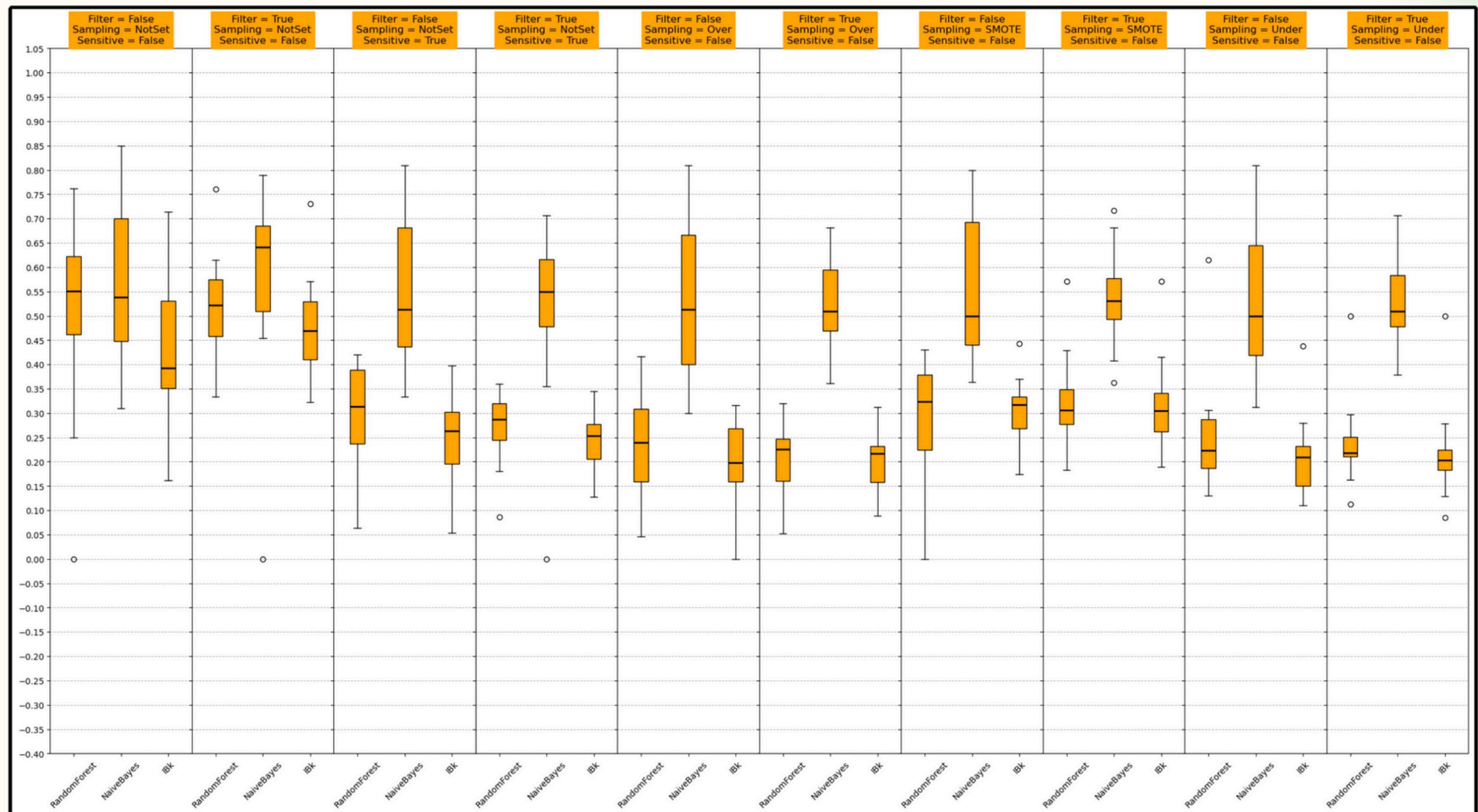
- **Feature selection:** migliora il valore di Npofb20 di Random Forest, ma peggiora quella di IBK e Naive Bayes.
- **Oversampling:** riduce il valore di Npofb20 per il classificatore Naive Bayes, mentre aumenta i valori per gli altri classificatori. Includendo **feature selection**, si rileva un aumento di Npofb20 per Naive Bayes, ma un non per IBK.
- **Smote:** registra un miglioramento di Npofb20 per Random Forest, e un leggero calo degli altri classificatori. In combinazione con **feature selection** si vede come tali valori subiscono un leggerissimo peggioramento.
- **Undersampling:** registra un peggioramento di Npofb20 per Naive Bayes, e un miglioramento degli altri classificatori, e con **feature selection**, notiamo un miglioramento per Random Forest e un peggioramento per IBK.
- **Sensitive learning:** constata un incremento del valore di Npofb20 per Random Forest, ma in combinazione con **feature selection**, emerge un leggero con per IBK e Naive Bayes.



# Risultati - OPENJPA

## Precision

- **Feature selection:** migliora la precision di IBK e Naive Bayes, ma peggiora quella di Random Forest.
- **Oversampling:** peggiora gravemente la precisione di Random Forest e IBK e, includendo **feature selection** si verifica un ulteriore, leggero peggioramento su tutti i classificatori.
- **Smote:** registrano i medesimi effetti del caso precedente (Oversampling).
- **Undersampling:** peggiora molto la precisione di IBK e Random Forest. Con l'aggiunta di feature selection, la situazione non cambia.
- **Sensitive learning:** notevolmente la precisione di tutti i classificatori tranne Naive Bayes e, applicando **feature selection**, la situazione peggiora ulteriormente, tranne che per Naive Bayes, dove si registra un leggero miglioramento.

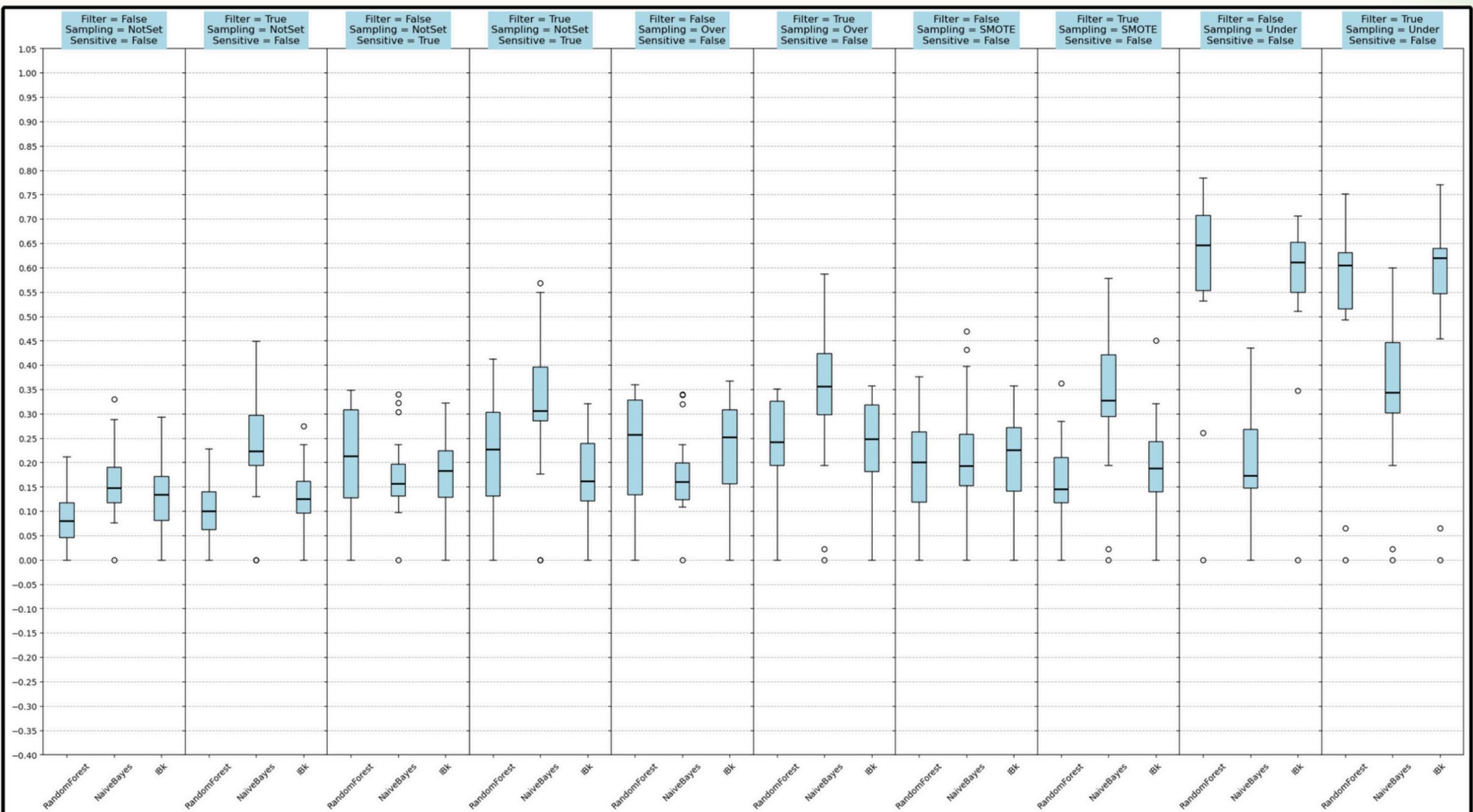


Possiamo dunque affermare un peggioramento per tutte le tecniche tranne la feature selection, i cui miglioramenti però non sono poi così significativi.

# Risultati - OPENJPA

## Recall

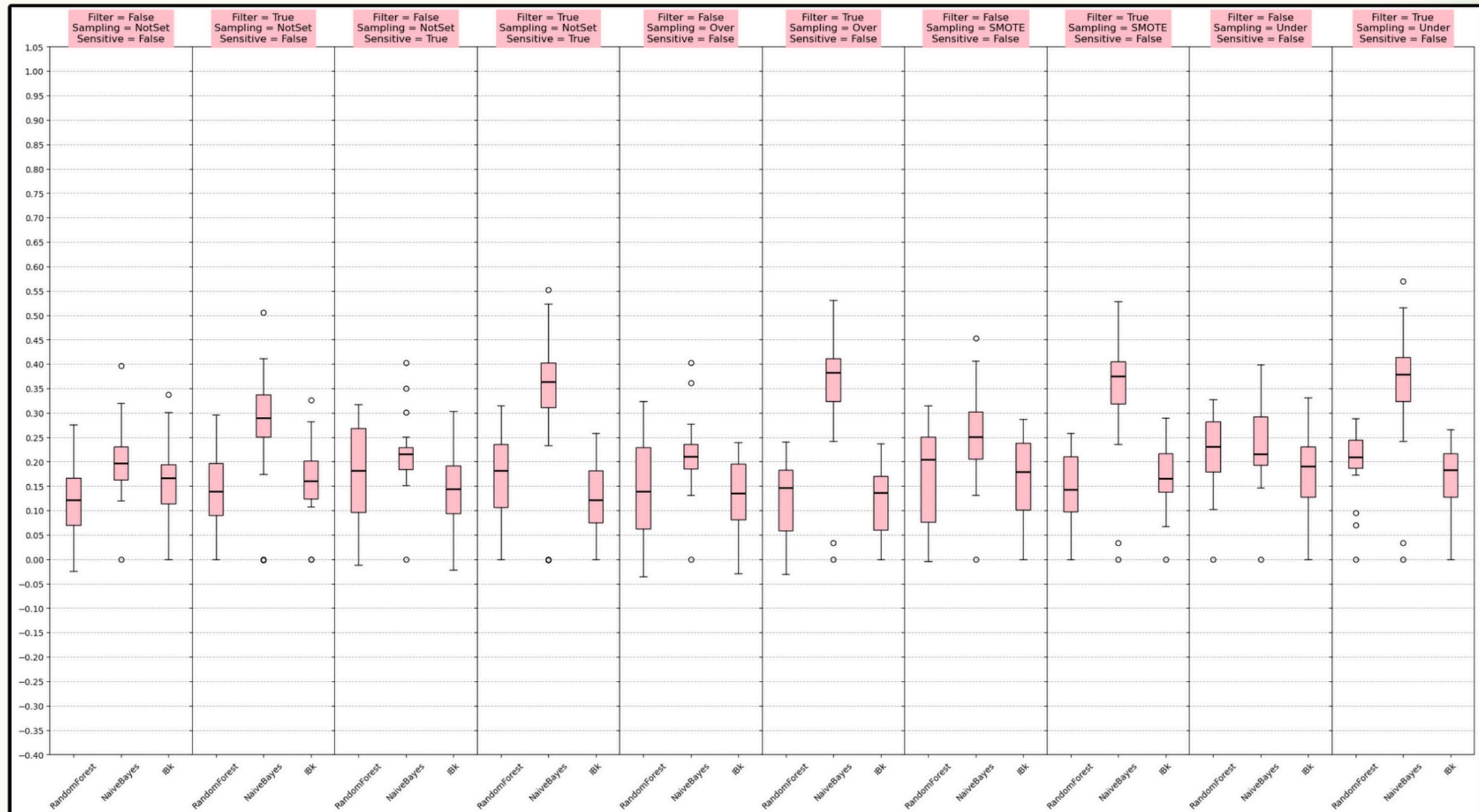
- **Feature selection:** migliora la recall di Naive Bayes e leggermente di Random Forest, ma peggiora quella di IBK.
- **Oversampling:** aumenta significativamente la recall in IBK e Random Forest. Inoltre, applicandola con **feature selection**, aumenta anche per Naive Bayes, che supera addirittura i valori registrati dagli altri due classificatori.
- **Smote:** aumenta la recall di tutti i classificatori. Tuttavia, con l'aggiunta di **feature selection**, l'aumento diminuisce per IBK e Random Forest, e aumenta significativamente per Naive Bayes.
- **Undersampling:** ha l'impatto più significativo sulla recall, raggiungendo valori massimi per IBK e Random Forest. Per Naive Bayes, invece, il miglioramento si ha quando tale tecnica viene combinata con la **feature selection**.
- **Sensitive learning:** aumenta la recall in IBK e Random Forest e con **feature selection**, si ottiene un miglioramento netto in Naive Bayes.



# Risultati - OPENJPA

## KAPPA

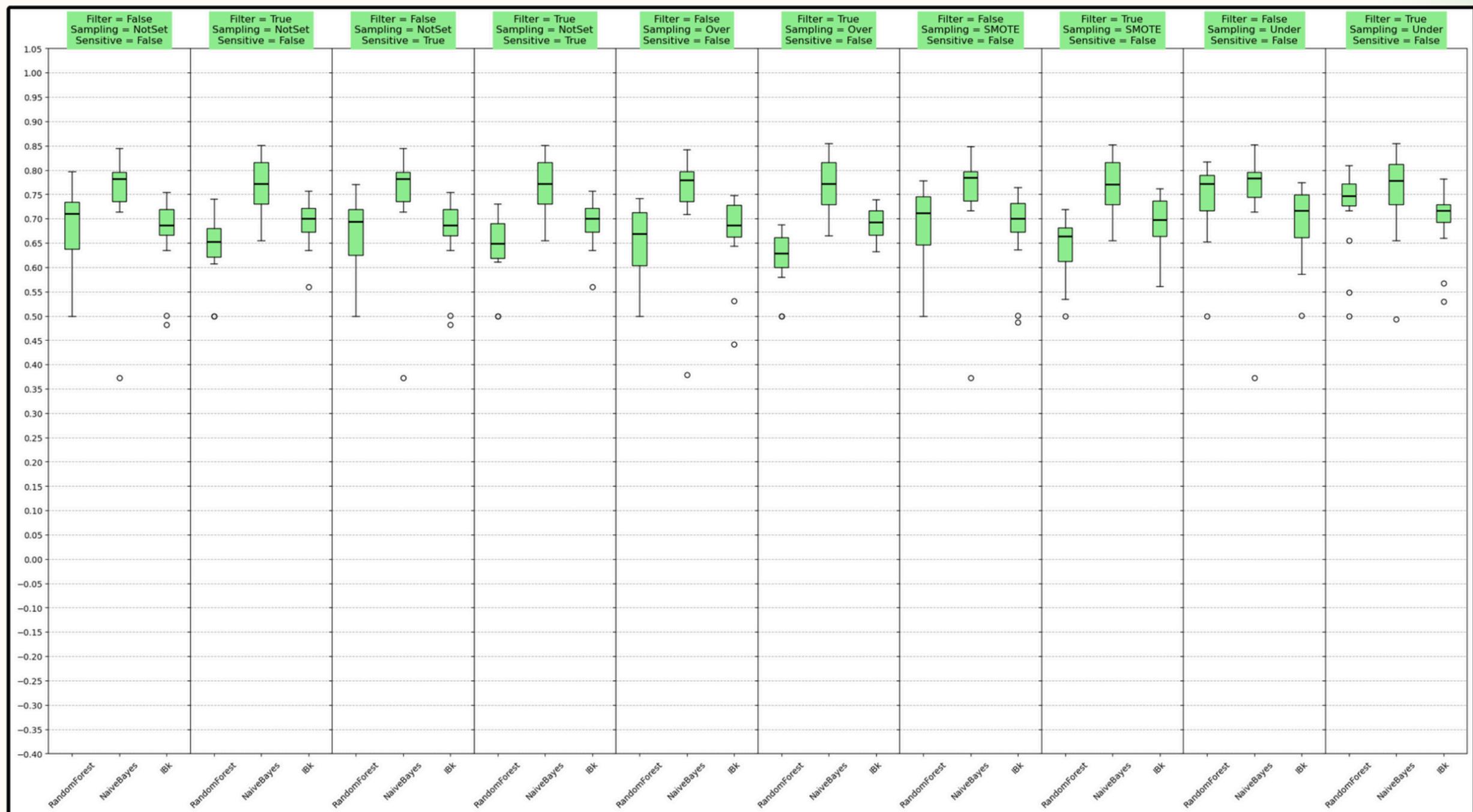
- **Feature selection:** migliora la Kappa del classificatore Naive Bayes e lievemente quella di Random Forest.
- **Oversampling:** peggiora la Kappa in IBK ma migliora lievemente in Random Forest e Naive Bayes, che poi registra un enorme incremento quando si considera anche **feature selection**.
- **Smote:** migliora la Kappa dei classificatori e, applicando anche **feature selection**, c'è un aumento significativo in Naive Bayes.
- **Undersampling:** incrementa i valori di Kappa in tutti i classificatori. In particolare, questo accade in Naive Bayes quando questa tecnica viene combinata con la **feature selection**.
- **Sensitive learning:** peggiora la Kappa in IBK e migliora lievemente quella in Random Forest e Naive Bayes. Insieme a **feature selection**, si ottiene un miglioramento rilevante della Kappa in Naive Bayes e lievemente in IBK.



# Risultati - OPENJPA

## AUC

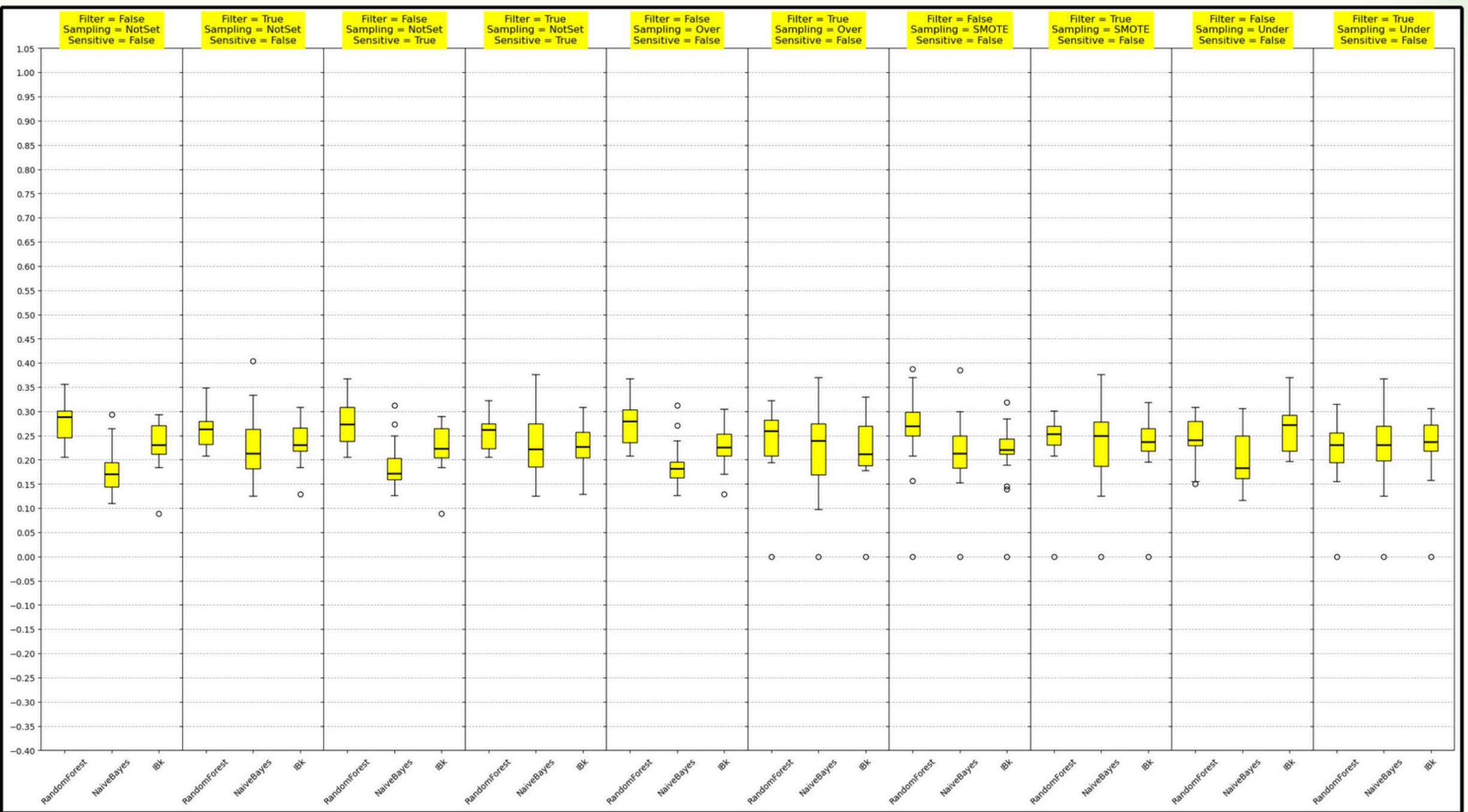
- **Feature selection:** diminuisce i valori di AUC in Random Forest e lievemente in Naive Bayes, migliorandoli in IBK.
- **Oversampling:** peggiora la AUC di Random Forest e con l'aggiunta di **feature selection**, si verifica un ulteriore peggioramento in Random Forest.
- **Smote:** migliora lievemente la AUC di IBK, e combinando **feature selection** c'è un lieve aumento della AUC in tutti i classificatori, tranne che per Random Forest, che peggiora.
- **Undersampling:** aumentare i valori di AUC in IBK e Random Forest, e quando viene aggiunta **feature selection**, si nota un peggioramento della AUC per Naive Bayes e Random Forest.
- **Sensitive learning:** diminuisce la AUC di Random Forest, e tale riduzione viene amplificata applicando la tecnica in combinazione con **feature selection**.



# Risultati - OPENJPA

## Npofb20

- **Feature selection:** migliora il valore di Npofb20 per Naive Bayes, ma peggiora per Random Forest.
- **Oversampling:** riduce il valore di Npofb20 per Random Forest e aumenta il valore per Naive Bayes. Includendo **feature selection**, si rileva lo stesso effetto, ma in misura maggiore.
- **Smote:** registra un miglioramento di Npofb20 per Naive Bayes, ma un decremento per gli altri classificatori. Combinato con **feature selection**, migliorano i valori sia di Naive Bayes che di IBK.
- **Undersampling:** registra un miglioramento di Npofb20 per tutti i classificatori, ma con **feature selection**, notiamo un peggioramento per Random Forest e IBK, mentre vediamo che per Naive Bayes, invece, aumenta Npofb20.
- **Sensitive learning:** constata un decremento del valore di Npofb20 per tutti i classificatori, ma in combinazione con **feature selection**, emerge un aumento per Naive Bayes.



# Conclusioni

## Techniques

L'analisi dei progetti BOOKKEEPER e OPENJPA evidenzia che le tecniche di preprocessing e selezione delle caratteristiche influenzano i classificatori in modi diversi:

- La Feature selection migliora alcuni classificatori (es. Naive Bayes su precision in BOOKKEEPER, IBK e Naive Bayes su recall in OPENJPA) ma ne peggiora altri (es. Random Forest su precision in OPENJPA).
- Oversampling e SMOTE tendono ad aumentare la recall ma spesso riducono anche la precisione. Il sensitive learning ha effetti molto variabili, ma generalmente, tende a peggiorare la precisione e migliorare la recall e Kappa.
- L'Undersampling migliora recall e Kappa, ma può peggiorare la precisione, ma risulta essere nel complesso la tecnica migliore, anche in combinazione con feature selection

---

## Classifiers

Scegliere il miglior classificatore dipende da diversi fattori, come la natura del problema e i dati disponibili. Nei progetti BOOKKEEPER e OPENJPA, Naive Bayes si è dimostrato stabile e affidabile per precisione, ma ha mostrato pessimi valori di recall, ad eccezione del caso in cui si utilizza la tecnica dell'undersampling. Inoltre, Naive Bayes permette di mantenere valori di NPofB2O più bassi rispetto agli altri classificatori, che hanno valori più alti. Al contrario, Random Forest si è rivelato versatile e robusto, ottenendo i migliori valori di recall e Kappa. IBK, invece, si è dimostrato il peggior classificatore, poiché presenta la peggiore precisione e il valore di kappa più basso. Inoltre, non eccelle in nessuna delle altre metriche, indipendentemente dalla combinazione di tecniche utilizzata.

In conclusione, possiamo affermare che la combinazione migliore è quella formata da **Naive Bayes e Undersampling**, poiché consente di ottenere un buon compromesso per una discreta recall (in OpenJPA raggiunge addirittura il valore maggiore), un'alta precisione e un'ottima AUC.

# Minacce alla validità

Dobbiamo però tenere conto di quelle che possono essere le minacce alla validità, che suddividiamo in:

- **Validità Interna**
  - Ordinamento delle versioni per data di rilascio.
  - Consideriamo un commit afferente ad una versione solo se a data del commit è compresa fra la data di rilascio della versione precedente e la data di rilascio della versione.
  - Scarto delle versioni senza commit, potrebbero essere esclusi ticket validi..
  - Gestione delle versioni rilasciate nella stessa data.
- **Validità Esterna**
  - Calcoliamo la P di proportion attraverso il metodo di Cold Start, la cui soglia potrebbe portare a stime inaffidabili.
- **Validità di Costrutto**
  - Le versioni rilasciate nella stessa data possono causare la perdita di commit rilevanti.
  - Se un file viene cancellato tra due versioni, non consideriamo le righe cancellate, il che porta una visione incompleta delle modifiche fatte.
  - Considerazione delle classi di una versione come quelle dell'ultimo commit.
- **Validità Statistica**
  - Se una classe ha zero commit in una versione significa che non viene modificata in quella versione e quindi l'unica misura non nulla sarà la size della classe. Però, ignorando classi non modificate potrebbe portare a una visione incompleta dello stato del progetto.

# Grazie per l'attenzione

## Links a Github e Sonarcloud



[https://github.com/MartorelliLuca/ISW2\\_CodeMetrics](https://github.com/MartorelliLuca/ISW2_CodeMetrics)



[https://sonarcloud.io/project/overview?id=MartorelliLuca\\_ISW2\\_CodeMetrics](https://sonarcloud.io/project/overview?id=MartorelliLuca_ISW2_CodeMetrics)