Software Analysis

Giulia Boccuccia 0354446

A.A. 2023/2024

Agenda 

Introduzione 

Ai giorni nostri, i sistemi software devono garantire alta qualità e minimizzazione dei costi, sia in fase di produzione che in fase di mantenimento.
Come possiamo far coincidere queste necessità con l'attività di software testing che risulta essere molto dispendiosa?
Dobbiamo trovare degli algoritmi di predizione per individuare le classi in cui possono trovarsi gli errori.

Strategia 

Usiamo le tecniche di **Machine Learning** per capire quali sono le principali caratteristiche che rendono le classi buggy, analizzando la loro storia passata. Tramite i classificatori, possiamo prevedere quali classi contengono dei difetti e concentrare i nostri sforzi solo su di loro.

Obiettivi 

Analizzare due progetti di Apache Software Foundation, **Bookkeeper** e **Storm**. Misurare l'accuratezza di tre **classificatori** usando diverse **tecniche** per migliorare la qualità delle predizioni. Infine stabilire l'impatto di tali tecniche sulle predizioni.

Classificatori

- Random Forest
- Naive Bayes
- IBK

Tecniche

- Feature Selection (*Best First*)
- Sampling (*Undersamplig, Oversamplig, SMOTE*)
- Cost sensitive (*Sensitive Learning*)

Metodologia 1

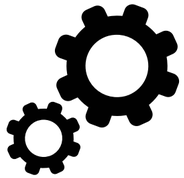
Stabiliamo come prima cosa il ciclo di vita di un bug all'interno di una classe.

La **Affected Version AV** racchiude tutte le versioni a partire dalla **IV** (inclusa), in cui era presente il bug, fino ad arrivare alla **FV** (esclusa) in cui il bug è stato risolto.

Metodologia 2

Il processo di misurazione è stato effettuato attraverso due strumenti principali:

- **JIRA:** Issue tracking system per la raccolta di tutte le release;
- **GIT:** Version control system per la raccolta di tutti i commit di una specifica release.

Metodologia 3 

Recuperiamo i ticket di tipo: "*buggy-fixed-closed*" ne estraiamo le AV. Per i ticket il cui campo AV risulta essere vuoto ricorriamo al metodo **Proportion (P).**

Idea: la distanza tra la OV e la FV è proporzionale alla distanza fra la IV e la FV.

In formule: $P = (FV-IV)/(FV-OV)$

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

Nelle prime iterazioni utilizziamo la tecnica del **cold start** per calcolare la mediana di P basandoci su altri progetti Apache simili.

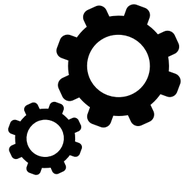
Successivemente, usiamo l'approccio **incrementale** per ottenere una media delle proportion calcolate sui bug "*fixed*" delle versioni precedenti .

Metodologia 4 

Per valutare il dataset dobbiamo fare attenzione allo **Snoring.**
Come sappiamo i difetti non possono essere individuati prima che si manifestino, questo comporta la presenza di molti bug dormienti nel sistema, che portano ad una prestazione peggiore dei classificatori. La soluzione a questo problema è evitare di usare dati troppo recenti e dati troppo passati.
Nella pratica quindi scartiamo la seconda metà delle release, aumentando l'affidabilità dei dati.

Metriche

Metrica	Descrizione
Size:	numero linee di codice della classe
LOC_added:	numero linee codice aggiunte durante le revisioni
AVG_LOC_added:	numero medio linee di codice aggiunte
LOC_deleted:	numero linee codice rimosse durante le revisioni
AVG_LOC_deleted:	numero medio linee codice rimosse
Churn:	somma dei moduli delle differenze tra LOC aggiunti e rimossi
AVG_Churn:	valore medio del churn
Fixed_defects:	numero difetti risolti
Number_of_commits:	numero commit all'interno della release
Number_of_authors:	numero di sviluppatori che hanno toccato la classe

Metodologia 5 

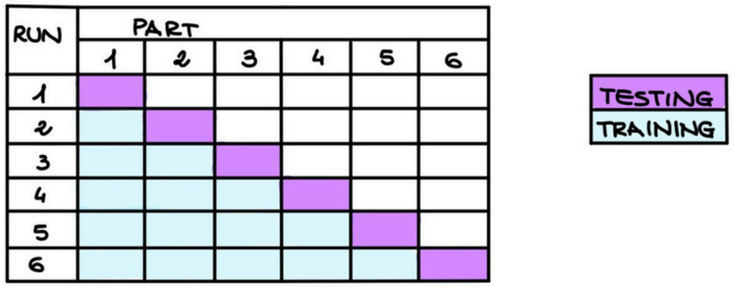
Dividiamo il dataset in **training set** e **testing set.**

- **Training set:** racchiude informazioni fornite per addestrare i classificatori.
- **Testing set:** racchiude le informazioni usate per valutare l'esattezza delle previsioni.

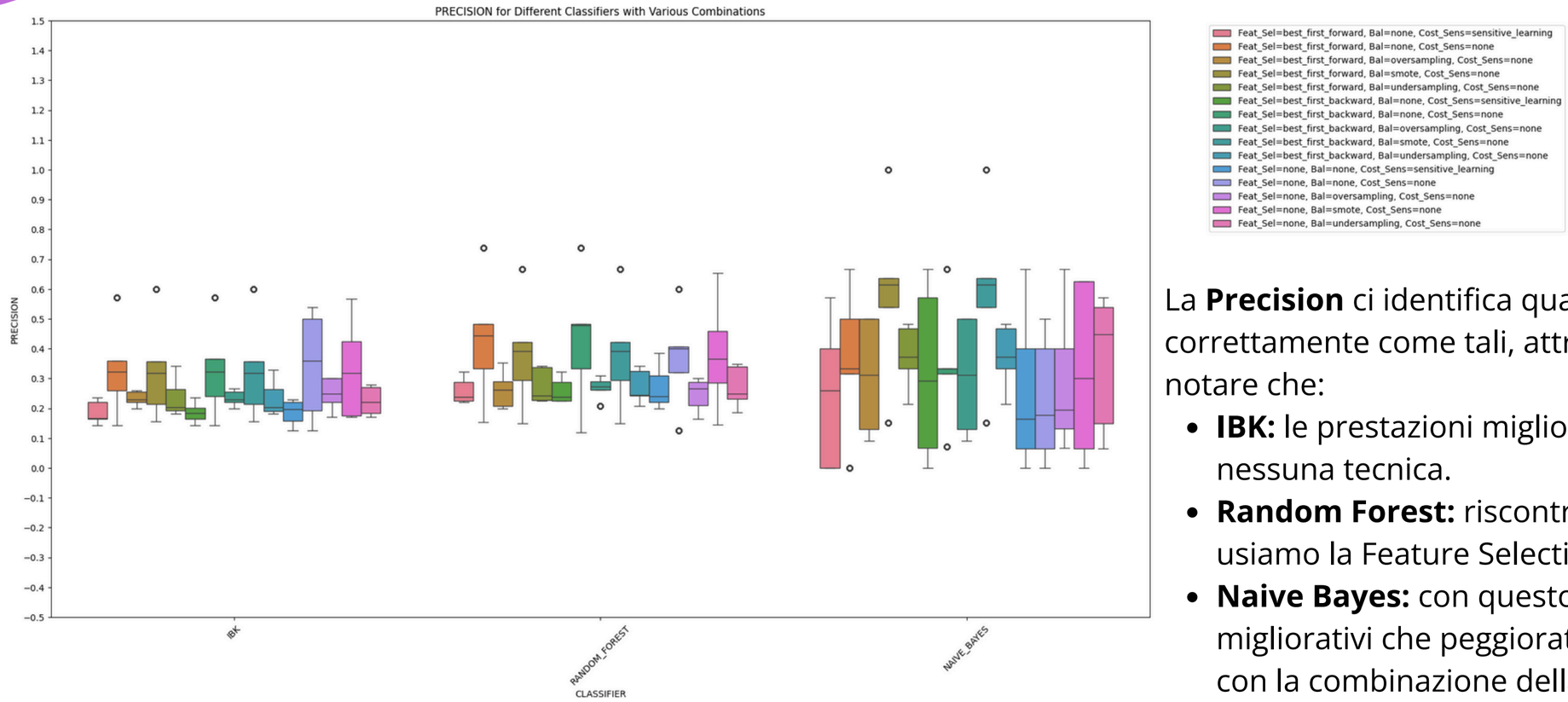
Seguendo l'assioma **size matters** possiamo affermare che: più sarà grande il training set migliore sarà il classificatore; più sarà grande il testing set invece, più accurata sarà la stima dell'errore.

Metodologia 6

Per validare i classifcatori usiamo la tecnica del **Walk Forward.** Questa tecnica è ampiamente usata per contesti *time-sensitive*. Il dataset viene diviso in due parti, ordinate cronologicamente. In ogni run tutti i dati disponibili vengono usati per valutare, quindi vanno a creare il training set; la parte da predire invece è usata come testing set.



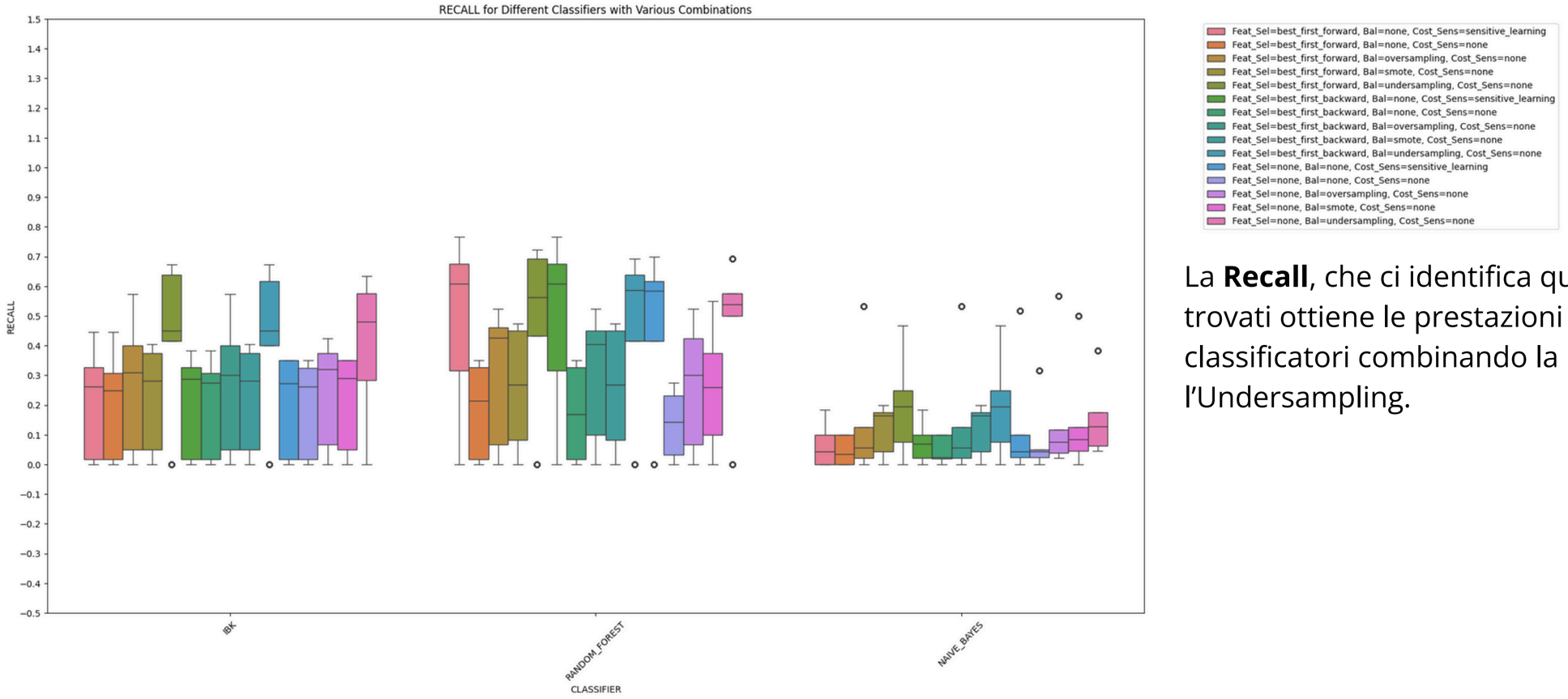
BOOKKEEPER - Precision



La **Precision** ci identifica quanti positivi sono stati classificati correttamente come tali, attraverso questo grafico possiamo notare che:

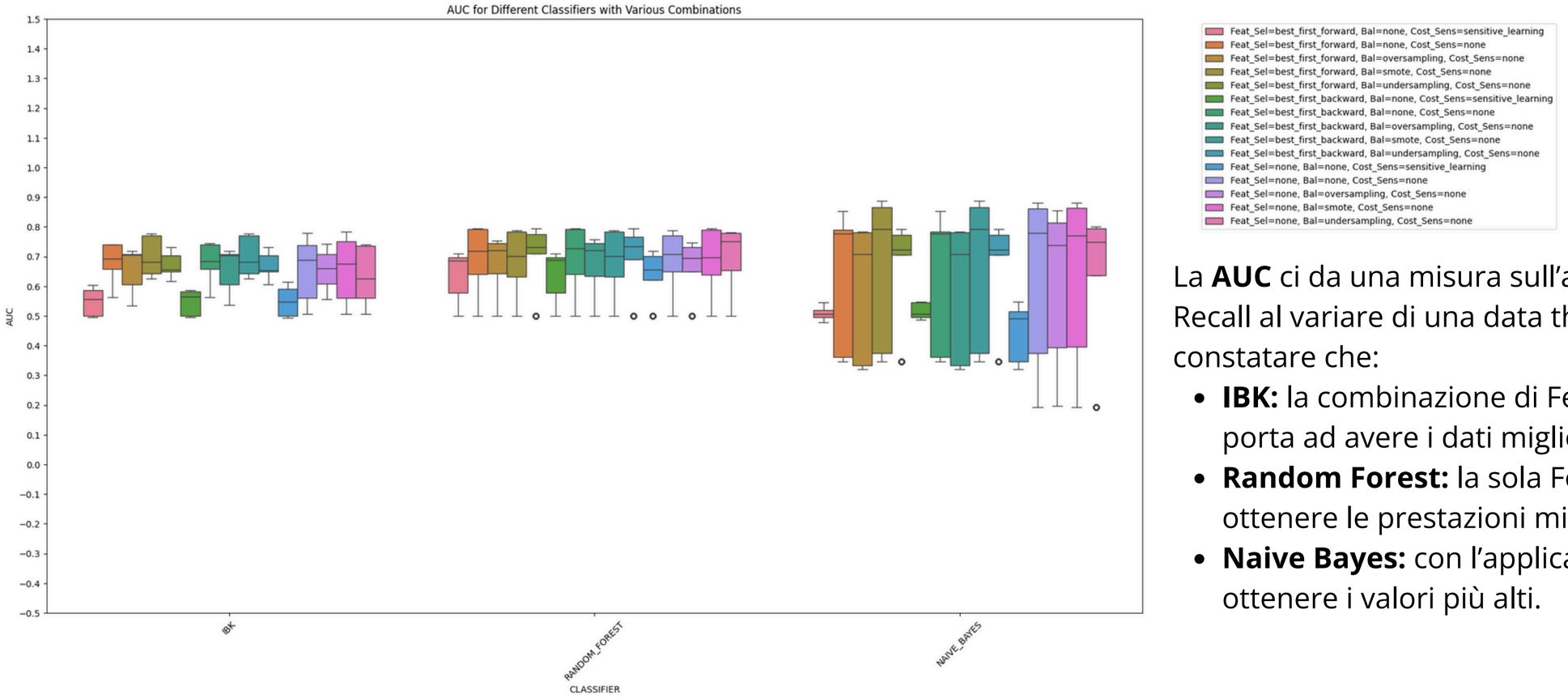
- **IBK:** le prestazioni migliori si hanno quando non viene usata nessuna tecnica.
- **Random Forest:** riscontriamo miglioramenti quando usiamo la Feature Selection.
- **Naive Bayes:** con questo classificatore abbiamo sia risultati migliorativi che peggiorativi. Il caso peggiore lo riscontriamo con la combinazione della Feature Selection e del Sensitive Learning. I casi migliori invece li otteniamo combinando la Feature Selection con SMOTE.

BOOKKEEPER - Recall



La **Recall**, che ci identifica quanti positivi sono stati trovati ottiene le prestazioni migliori con tutti i classificatori combinando la Feature Selection con l'Undersampling.

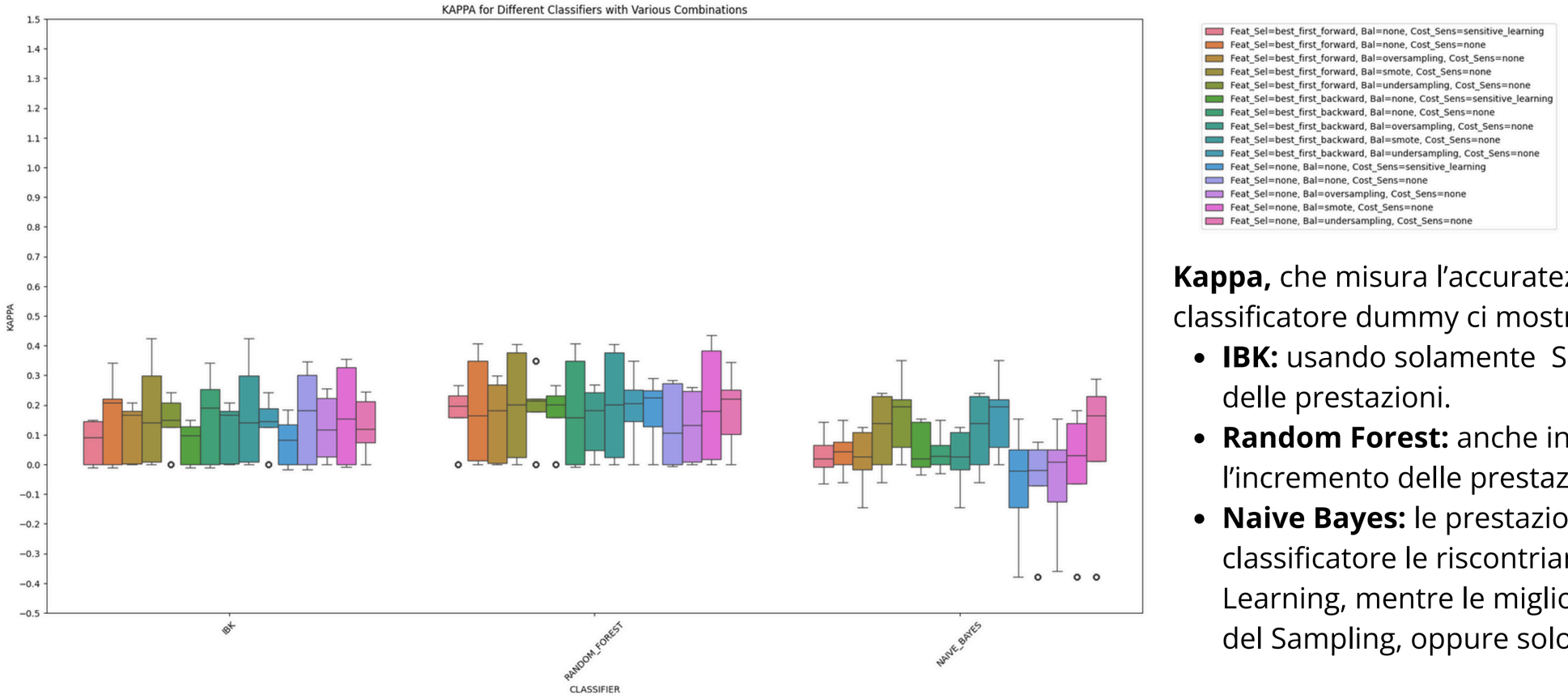
BOOKKEEPER - AUC



La **AUC** ci da una misura sull'andamento di Precision e Recall al variare di una data threshold. Possiamo constatare che:

- **IBK:** la combinazione di Feature Selection con SMOTE ci porta ad avere i dati migliori nei valori di AUC.
- **Random Forest:** la sola Feature Selection riesce a farci ottenere le prestazioni migliori.
- **Naive Bayes:** con l'applicazione di SMOTE riusciamo ad ottenere i valori più alti.

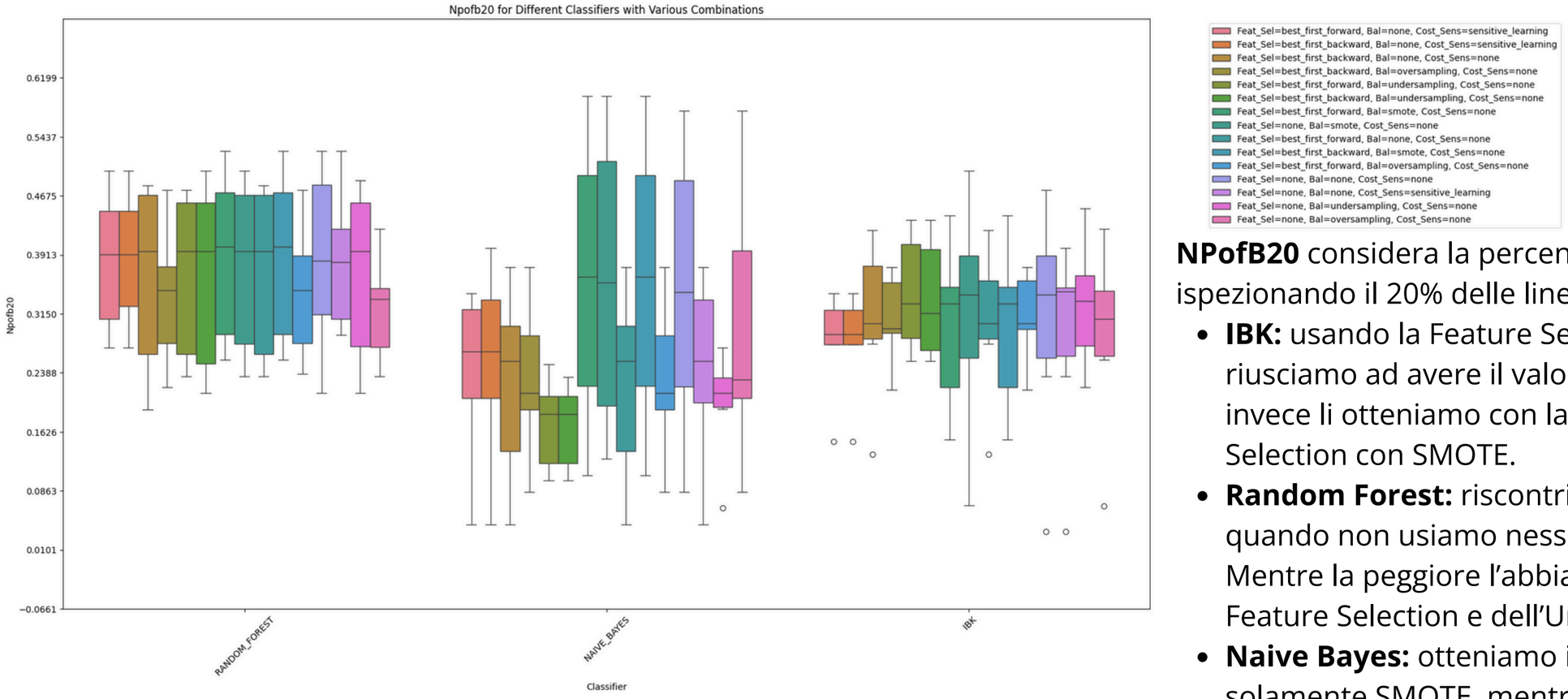
BOOKKEEPER - Kappa



Kappa, che misura l'accuratezza rispetto ad un classificatore dummy ci mostra che:

- **IBK:** usando solamente SMOTE otteniamo il massimo delle prestazioni.
- **Random Forest:** anche in questo caso SMOTE aiuta l'incremento delle prestazioni.
- **Naive Bayes:** le prestazioni peggiori con questo classificatore le riscontriamo con l'uso del Sensitive Learning, mentre le migliori le abbiamo o solo con l'uso del Sampling, oppure solo con l'uso di SMOTE.

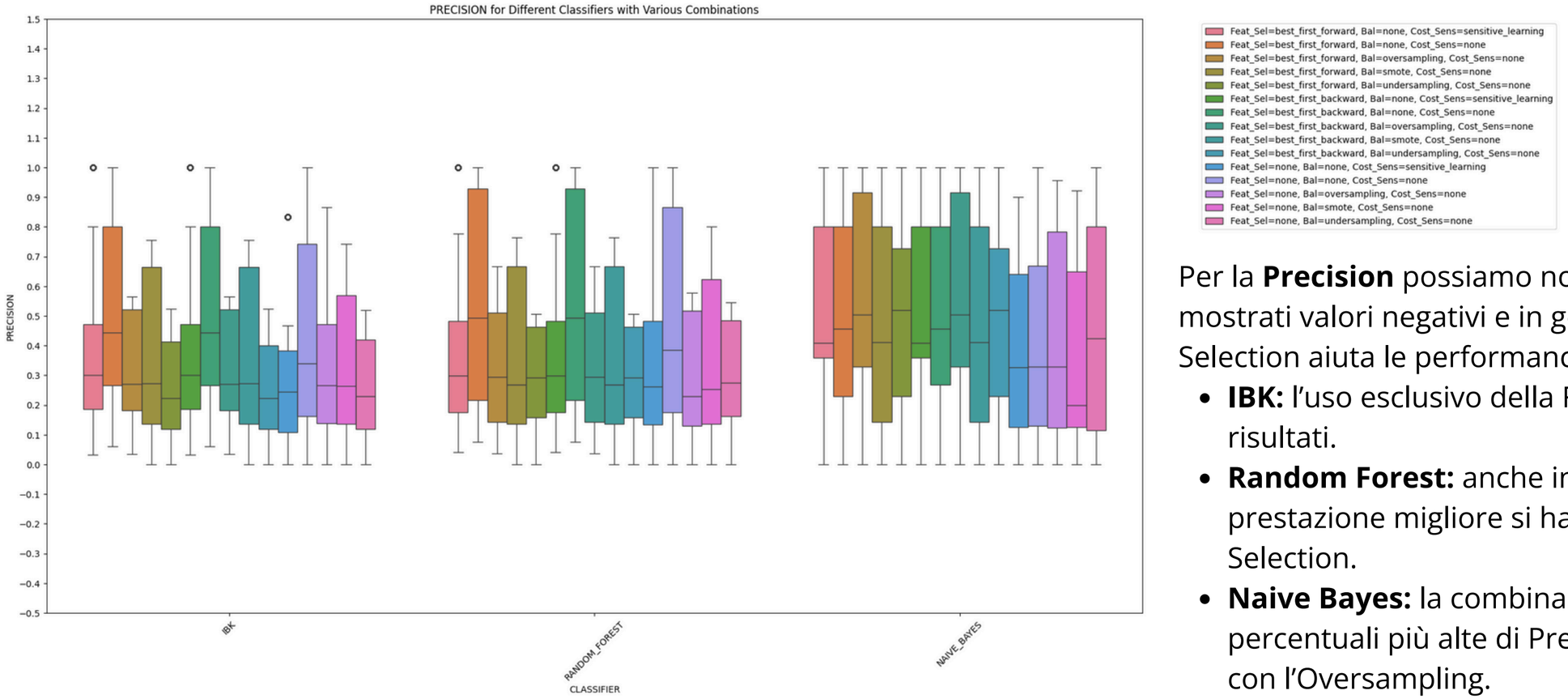
BOOKKEEPER - NpofB20



NPofB20 considera la percentuale delle entità difettose ispezionando il 20% delle linee di codice mostrando che:

- **IBK:** usando la Feature Selection e l'Undersampling riusciamo ad avere il valore migliore. I valori peggiori invece li otteniamo con la combinazione della Feature Selection con SMOTE.
- **Random Forest:** riscontriamo le migliori prestazioni quando non usiamo nessuna delle tecniche studiate. Mentre la peggiore l'abbiamo in corrispondenza della Feature Selection e dell'Undersampling.
- **Naive Bayes:** otteniamo il risultato migliore utilizzando solamente SMOTE, mentre i valori peggiori risultano uscire dalla combinazione della Feature Selection con l'Undersampling.

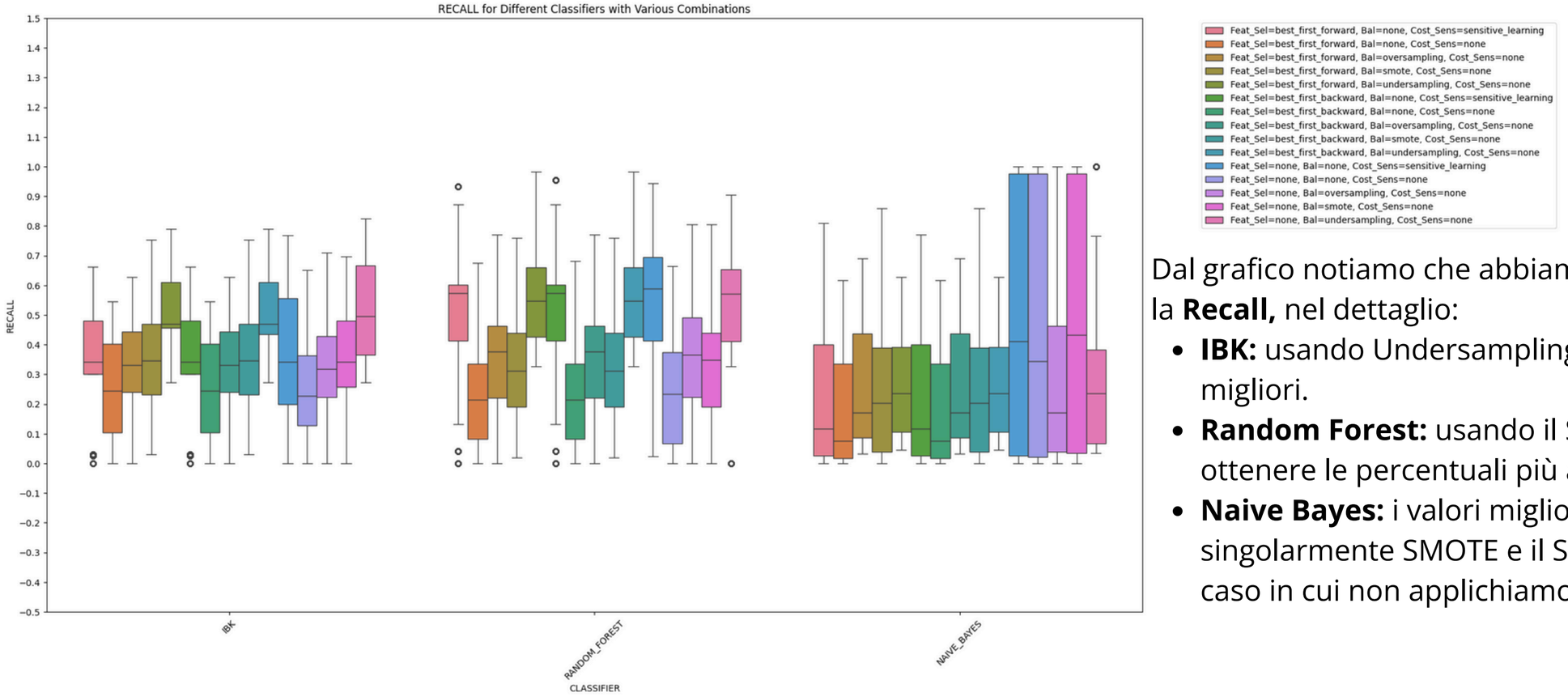
STORM - Precision



Per la **Precision** possiamo notare che non vengono mostrati valori negativi e in generale l'uso della Feature Selection aiuta le performances.

- **IBK:** l'uso esclusivo della Feature Selection migliora i risultati.
- **Random Forest:** anche in questo caso notiamo che la prestazione migliore si ha solo con l'uso della Feature Selection.
- **Naive Bayes:** la combinazione che ci porta ad avere percentuali più alte di Precision è la Feature Selection con l'Oversampling.

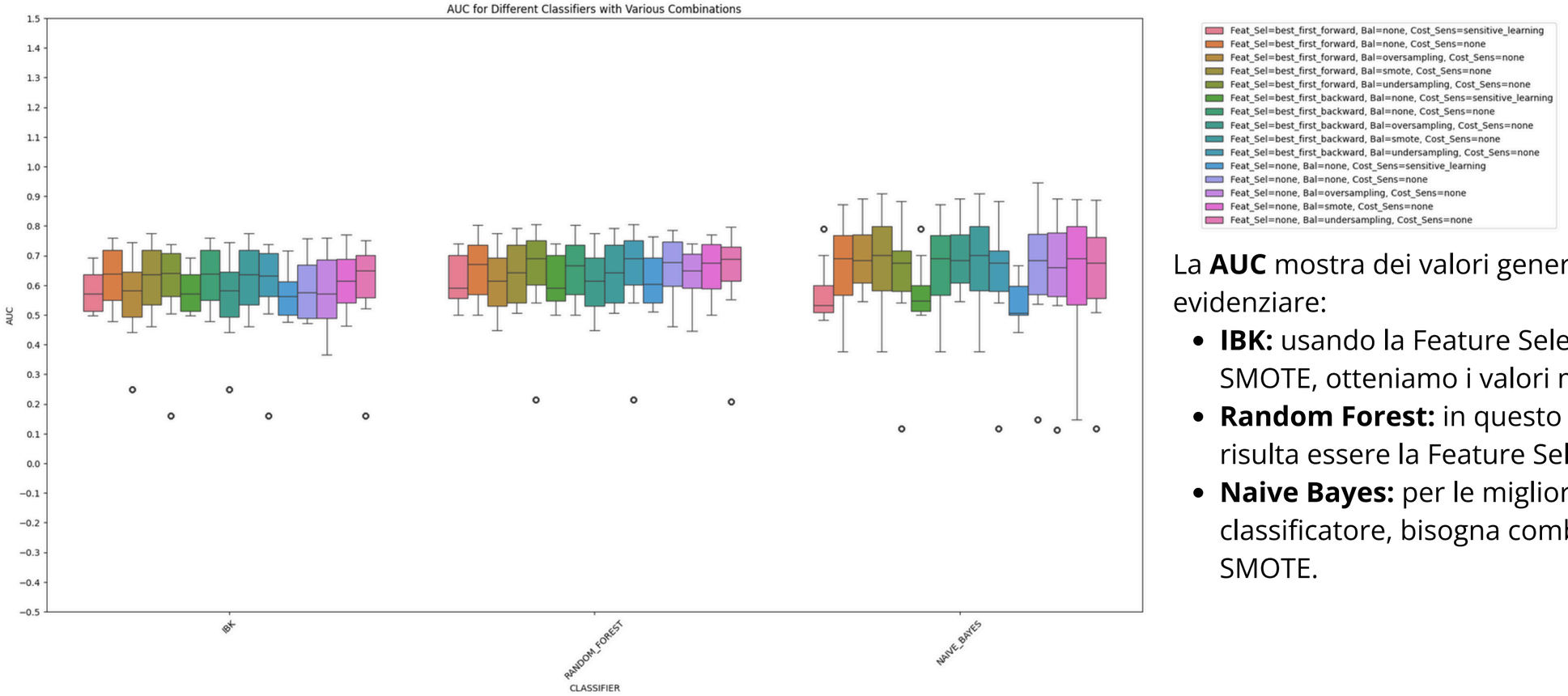
STORM - Recall



Dal grafico notiamo che abbiamo dei risultati molto vari per la **Recall,** nel dettaglio:

- **IBK:** usando Undersampling possiamo ottenere i risultati migliori.
- **Random Forest:** usando il Sensitive Learning possiamo ottenere le percentuali più alte.
- **Naive Bayes:** i valori migliorni li abbiamo valutando singolarmente SMOTE e il Sensitive Learning oppure nel caso in cui non applichiamo nessuna tecnica.

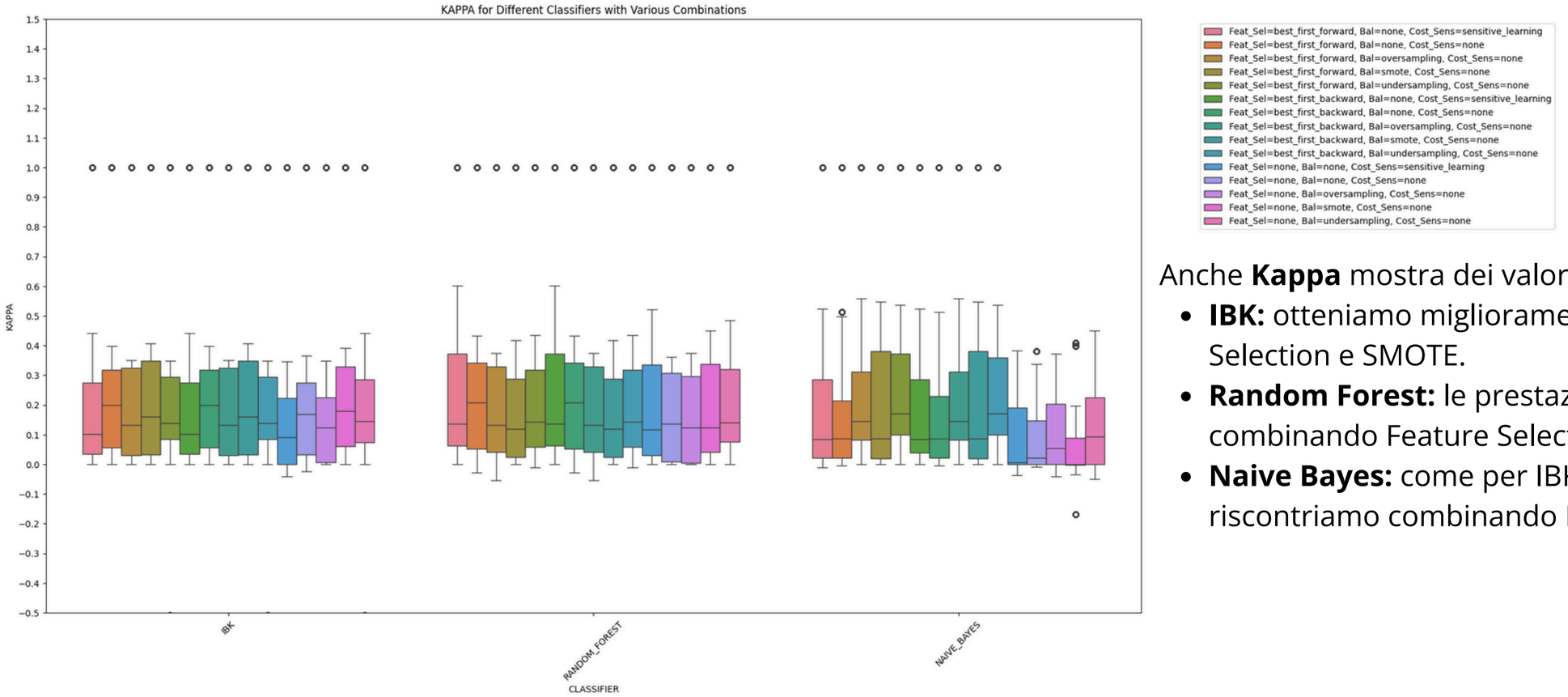
STORM - AUC



La **AUC** mostra dei valori generalemente stabili, possiamo evidenziare:

- **IBK:** usando la Feature Selection in combinazione con SMOTE, otteniamo i valori maggiori.
- **Random Forest:** in questo caso la combinazione migliore risulta essere la Feature Selection con l'Undersampling.
- **Naive Bayes:** per le migliori prestazioni di questo classificatore, bisogna combinare Feature Selection e SMOTE.

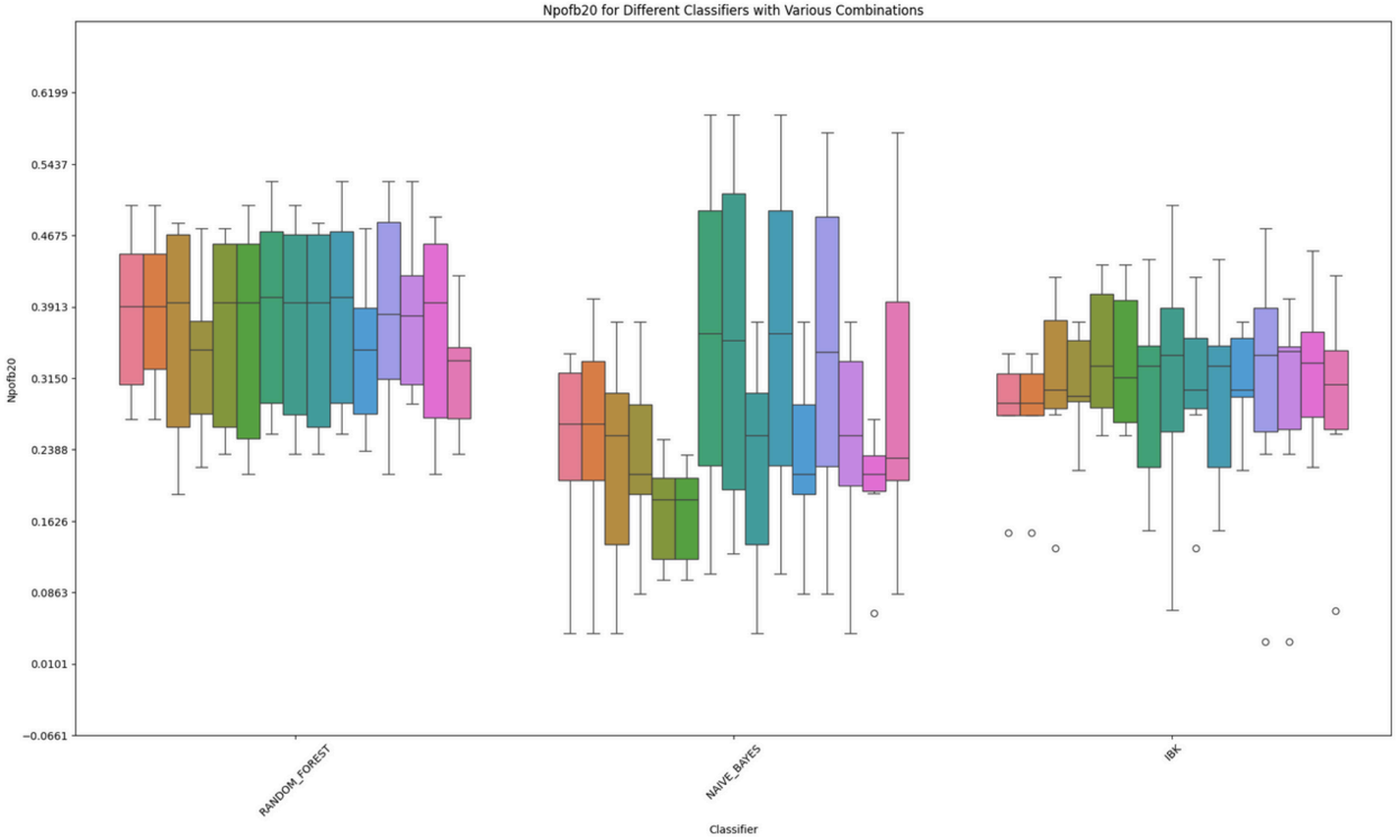
STORM - Kappa



Anche **Kappa** mostra dei valori generalemente uniformi.

- **IBK:** otteniamo miglioramenti applicando Feature Selection e SMOTE.
- **Random Forest:** le prestazioni migliori le otteniamo combinando Feature Selection con Sensitive Learning.
- **Naive Bayes:** come per IBK i miglioramenti li riscontriamo combinando Feature Selection e SMOTE.

STORM - NpofB20



- **IBK:** usando la Feature Selection e l'Undersampling otteniamo il valore migliore. I valori peggiori invece li abbiamo con Feature Selection e SMOTE.
- **Random Forest:** riscontriamo le migliori prestazioni quando non applichiamo nessuna delle tecniche precedenti. Mentre notiamo un peggioramento in corrispondenza della Feature Selection unita con l'Undersampling.
- **Naive Bayes:** otteniamo il risultato migliore utilizzando solamente SMOTE, mentre i valori minori risultano uscire dalla combinazione della Feature Selection con l'Undersampling.

Considerazioni 1 

Come era prevedibile la nostra analisi ci mostra che non c'è un classificatore migliore rispetto ad un altro. Le tecniche associate ad ognuno dei classificatori scelti possono portare a dei miglioramenti o a dei peggioramenti, quindi, ragionevolemente, possiamo affermare che non c'è una scelta assoluta per la coppia classificatore - tecnica. Tuttavia possiamo notare che:

- **Bookkeeper:** se dovessimo scegliare una coppia che risulti migliore rispetto alle altre possiamo affermare che è quella composta da **Feature Selection** e **SMOTE** che porta dei miglioramenti nella maggior parte dei classificatori.

Considerazioni 2

- **Storm:** possiamo notare che le migliori prestazioni per questo software vengono riscontrate quando usiamo la tecnica di **Feature Selection** con qualsiasi tipo di classificatore, in combinazione o meno con il **Sampling**.

Links 

GitHub: https://github.com/GBoc09/ISW2_Project

SonarCloud: https://sonarcloud.io/project/overview?id=GBoc09_ISW2_Project

Grazie per l’attenzione