Machine Learning for Software Engineering

Luigi Ciuffreda – 0351898 - Università di Roma
Tor Vergata a.a. 2023/2024

AGENDA

- **Introduzione** (Contesto, obiettivo dello studio)
- **Progettazione** (Strumenti utilizzati, etc..)
- **Variabili** (Classificatori, tecniche, metriche)
- **Risultati**
- **Conclusioni**
- **Minacce alla validità**
- **Link utili**

INTRODUZIONE - Contesto

Nel settore dello sviluppo software, assicurare elevati standard di qualità e ridurre al minimo i costi sono obiettivi primari. Il software si evolve rapidamente e l'introduzione continua di nuove funzionalità, insieme a rilasci frequenti delle versioni, aumenta il rischio di errori. Come possiamo mitigare questo rischio e prevenire l'insorgenza di **bug**?

- **Soluzione**: usare il machine learning per rilevare e prevenire i bug all’interno del software. Questo approccio non solo migliora la qualità complessiva del prodotto, ma permette anche di ottimizzare i processi di sviluppo rendendoli più efficienti e mirati.

INTRODUZIONE – Obiettivi dello Studio

Costuire un dataset che unisca tutte le informazioni sulla storia passata.

Valutare le prestazioni dei modelli al variare delle tecniche utilizzate.

Individuare quali sono le tecniche migliori che aumentano l’accuratezza dei classificatori.

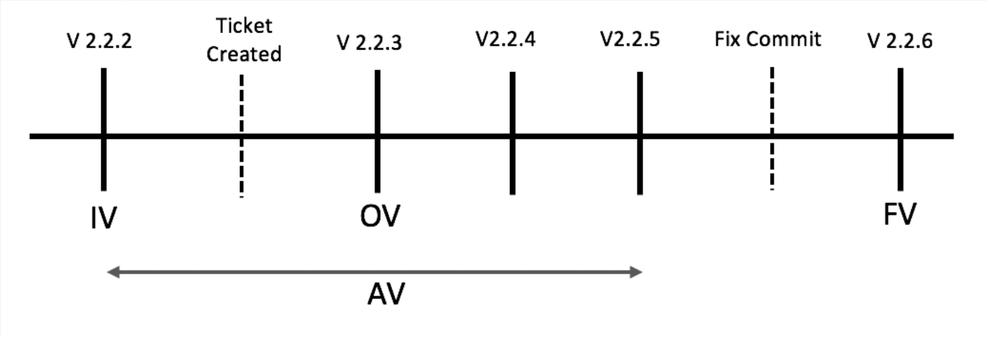
PROGETTAZIONE – Strumenti Utilizzati

Per sapere se una classe è stata buggy oppure no, possiamo utilizzare vari strumenti che ci permettono di analizzare e tracciare i problemi nei progetti.
Per analizzare i progetti di **BOOKKEEPER** e **ZOOKEEPER** utilizziamo i seguenti strumenti:

- **JIRA:** Fornisce una visione dettagliata delle revisioni e delle modifiche che hanno risolto i bug, aiutando a identificare le classi che sono state buggy.
- **Git:** version control system usato per raccogliere i commit.
- **WEKA:** software utilizzato per la creazione e implementazione di modelli di machine learning
- **ACUME :** utilizzato per il calcolo della metrica NPofB20

PROGETTAZIONE - Individuazione Classi Buggy

- Come otteniamo i dati relativi alla bugginess delle classi?
  - i progetti considerati utilizzando il sistema di ***issue tracking*** JIRA, che riporta informazioni riguardanti tutti i bug scoperti e la loro soluzione in dei ticket.
- Ogni bug ha un ciclo di vita:



- Le classi in cui il bug è presente sono buggy dalla relase **IV** alla realease **FV** .
- L'insieme di release che vanno dalla **IV** ad **OV** viene chiamato Affected Version (**AV**)
- Ogni classe che viene associata ad un ticket, deve essere etichettata come buggy dall'introduzione del bug (**IV**) fino al momento in cui viene risolto (**FV**).
- I valori di **OV** e di **FV** sono riportati sempre nei ticket di Jira, visto che indicano la creazione del ticket e la risoluzione dello stesso.

PROGETTAZIONE - Individuazione Classi Buggy

- **PROBLEMA**: non tutti i ticket JIRA contengono l'indicazione relativa alla IV, che però è necessaria per il labeling delle classi.
- **SOLUZIONE**: assumiamo esista una proporzionalità fissa tra l'intervallo di tempo **(IV,OV)** e l'intervallo (**OV,FV).**
- La tecnica che ci permette di calcolare il valore di IV viene chiamata **proportion**, che permette di calcolare una costante di proporzionalità *p*, per tutti i ticket in cui IV è nota viene definita nel seguente modo:

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

- Per tutti gli altri ticket la IV la calcoliamo così:

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

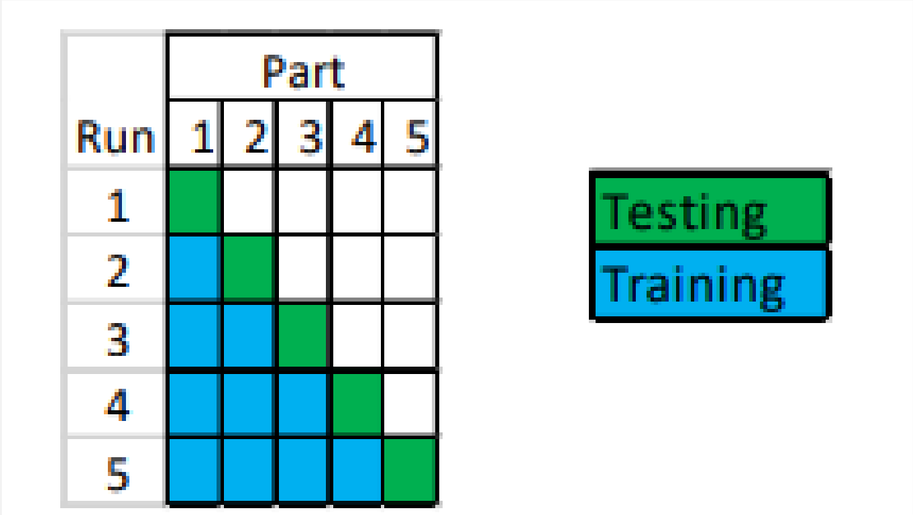
PROGETTAZIONE – Metriche Scelte

Dopo aver identificato le versioni con i bug e aver eseguito il labeling associando le classi buggy alle relative release, si procede con il calcolo delle metriche per la predizione. Con l'obiettivo di verificare empiricamente la validità del concetto "*less is more*," sono state selezionate diverse metriche:

Nome	Descrizione
Size(LOC)	Numero di linee di codice.
LOC Touched	Somma delle LOC modificate.
NR	Numero di revisioni.
Nfix	Numero di correzioni di difetti.
Max LOC Touched	Massimo delle linee di codice aggiunte in una singola revisione.
Churn	Valore assoluto della differenza tra linee aggiunte e rimosse
maxChurn	Massima variazione di linee di codice su tutte le revisioni.
AverageChurn	Media della variazione di linee di codice su tutte le revisioni.
Max LOC Added	Massimo delle linee di codice aggiunte in una singola revisione.
Age	Età della classe.

PROGETTAZIONE - Valutazione dei Classificatori

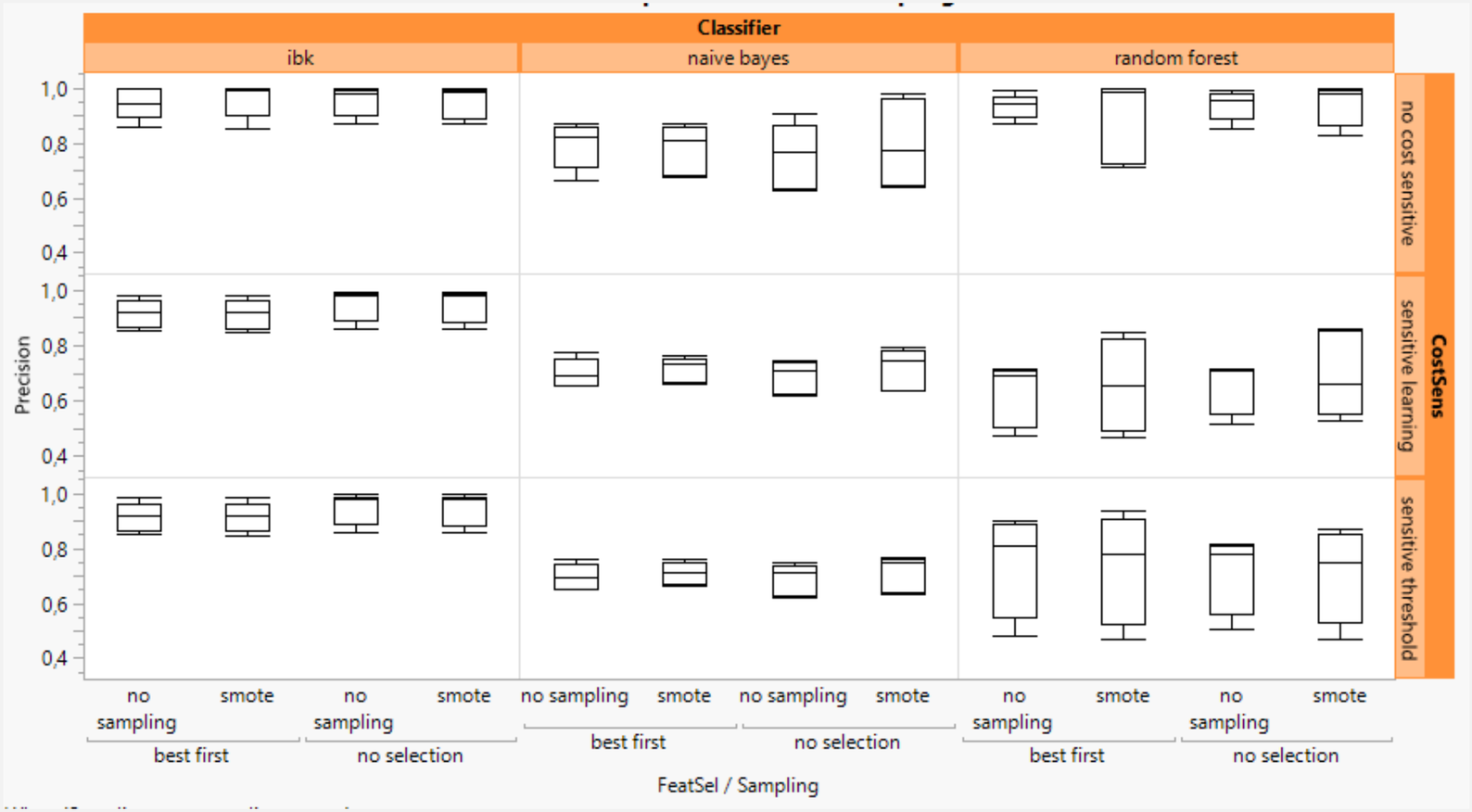
- Per valutare i classificatori, utilizziamo la tecnica di validazione «**walk-forward»,** adatta per dati temporali come le serie storiche. Il dataset viene suddiviso in gruppi ordinati temporalmente secondo le release: in ogni iterazione, il training set (insieme di dati utilizzato per addestrare il modello) comprende le prime (k-1) release, mentre il test set (insieme di dati utilizzato per valutare la performance del modello) è costituito dalla k-esima release.



VARIABILI – Classificatori, tecniche, metriche

Classificatori	Tecniche	Metriche
IBK	Nessun Filtro	Precision
Naive Bayes	Feature Selection (Best First)	Recall
Random Forest	Sampling (SMOTE)	AUC
	Sensitive Learning (CFN = 10*CFP)	Kappa
	Sensitive Threshold	NPofB20

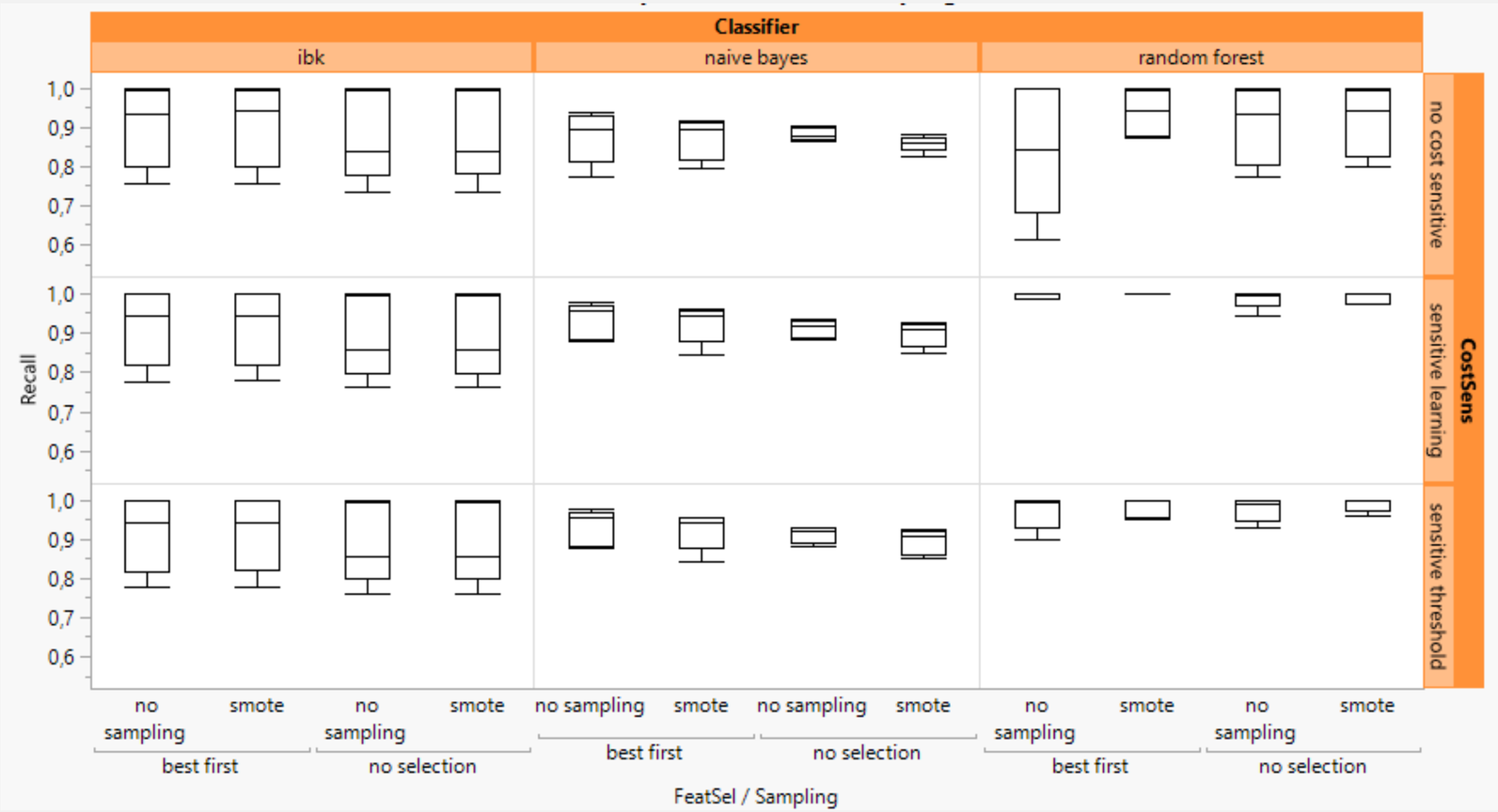
RISULTATI – Bookkeeper: Precision



Le **combinazioni migliori** sono:
IBK con NO_SAMPLING e SMOTE, con COST_SENSITIVE e SENZA, NO_SELECTION.
RF con NO_SELECTION, con NO_SAMPLING , SMOTE, NO_COST_SENSITIVE

Le **peggiori combinazioni** sono:
NB con NO_SAMPLING, NO_SELECTION, SENSITIVE_LEARNING; RF con SMOTE, NO_SELECTION e BEST_FIRST, SENSITIVE_THRESHOLD e SENSITIVE_LEARNING

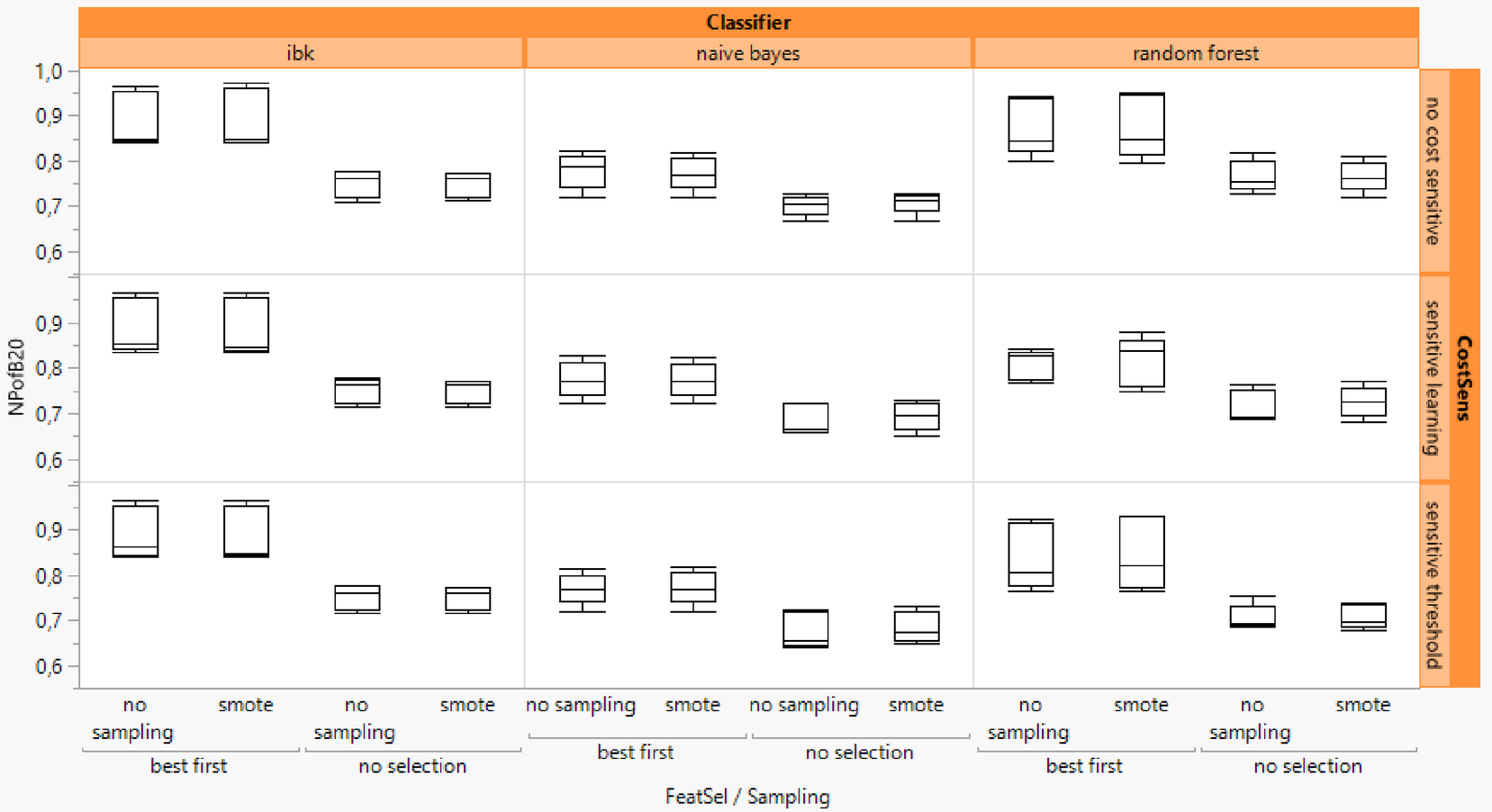
RISULTATI – Bookkeeper: Recall



Le **migliori combinazioni** sono:
RF con BEST_FIRST, NO_SELECTION, SENSITIVE_LEARNING, SENSITIVE_THRESHOLD.

Le **combinazioni peggiori** sono:
IBK sia con COST_SENSITIVE che senza, con BEST_FIRST, NO_SELECTION sia con NO_SAMPLING che con SMOTE

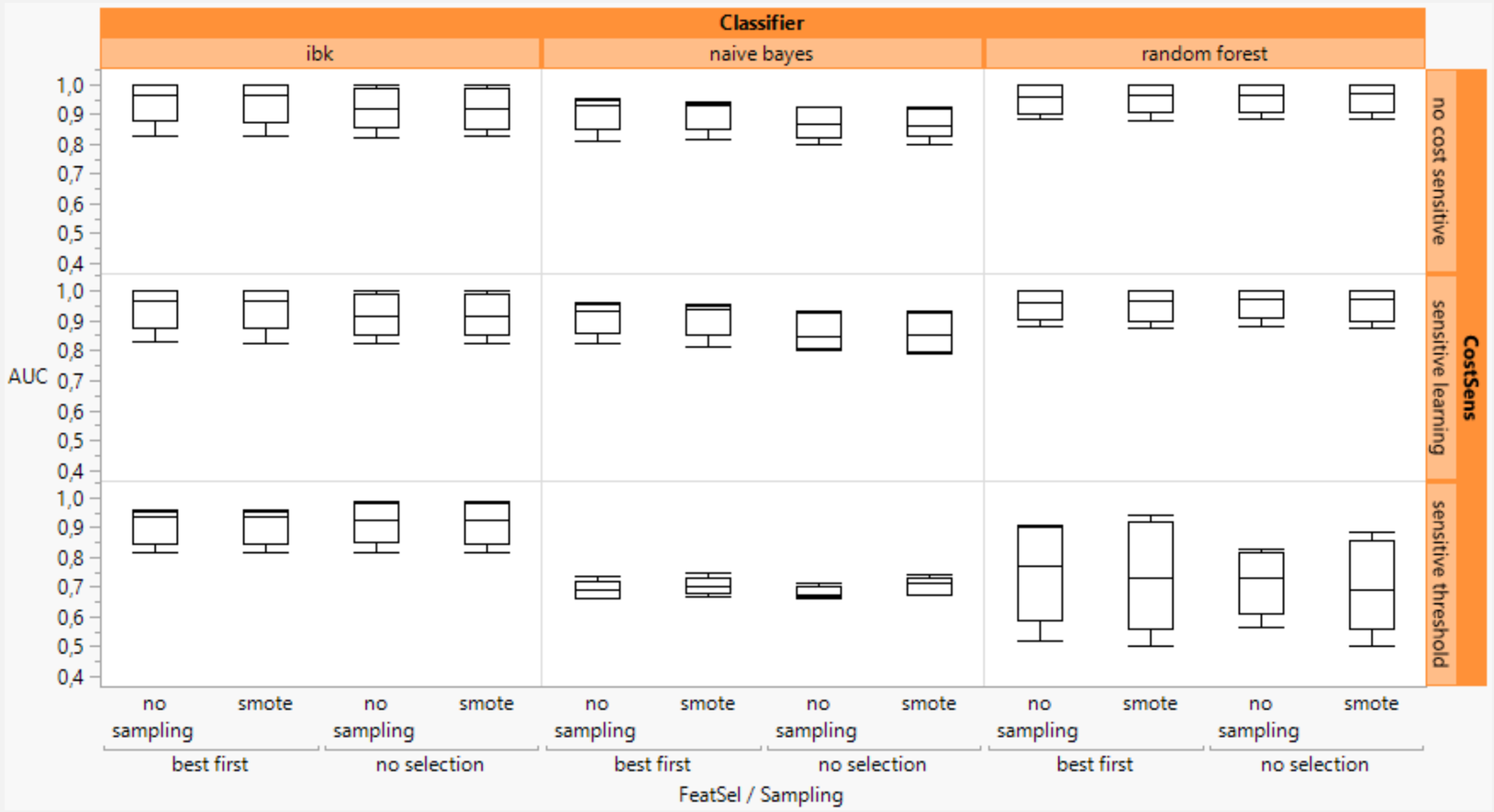
RISULTATI – Bookkeeper: NPofB20



Le **migliori combinazioni** sono:
IBK con BEST_FIRST, NO_SAMPLING, SMOTE, sia con COST_SENSITIVE che SENZA. RF con BEST_FIRST con NO_COST_SENSITIVE e SENSITIVE_THRESHOLD

Le **combinazioni peggiori** invece sono:
IBK con NO_SELECTION,NO_SAMPLING e SMOTE, SENSITIVE_LEARNING, NO_COST_SENSITIVE, SENSITIVE_THRESHOLD;
RF con NO_SELECTION, NO_SAMPLING e SMOTE, SENSITIVE_LEARNING, NO_COST_SENSITIVE, SENSITIVE_THRESHOLD

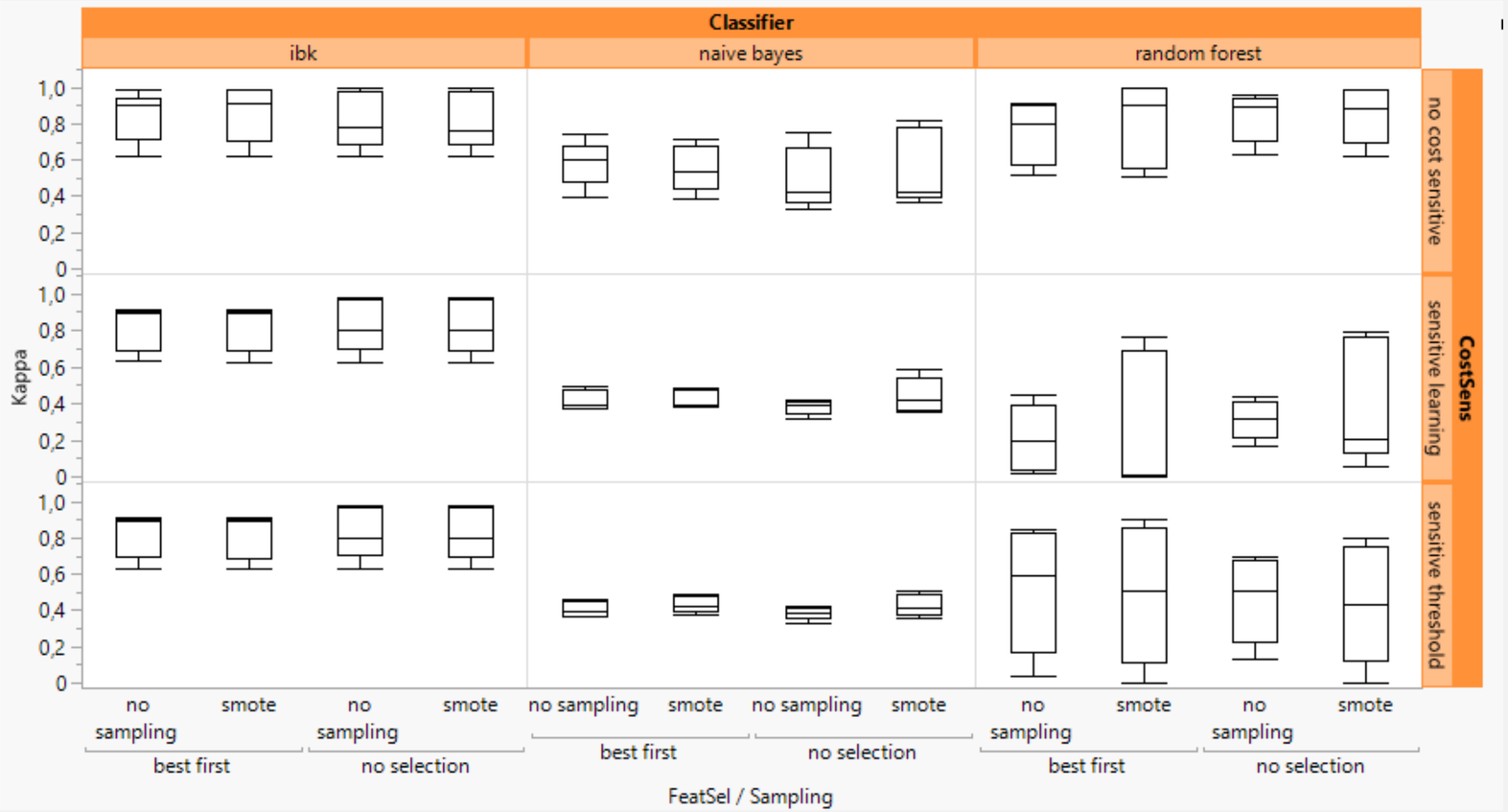
RISULTATI – Bookkeeper: AUC



Le **combinazioni migliori** sono:
RF con BEST_FIRST /NO_SELECTION, NO_SAMPLING e SMOTE, e NO_COST_SENSITIVE/SENSITIVE_LEARNING.

Le **combinazioni peggiori** sono:
NB con BEST_FIRST/NO_SELECTION, NO_SAMPLING/SMOTE, con SENSITIVE_THRESHOLD.

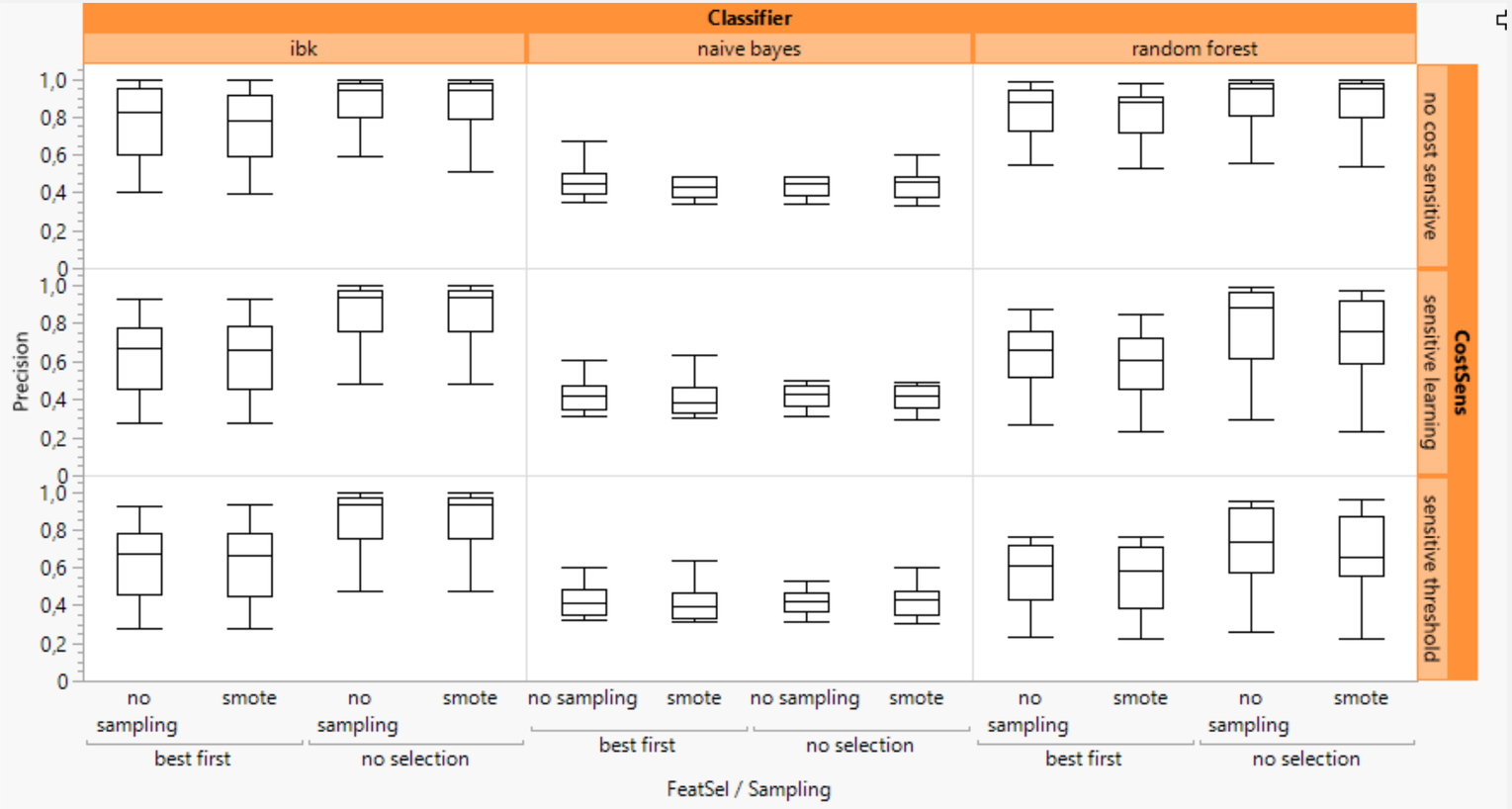
RISULTATI – Bookkeeper: Kappa



Le **combinazioni migliori** sono :
IBK con BEST_FIRST/ NO_SELECTION, NO_SAMPLING e SMOTE e sotto NO_COST_SENSITIVE,SENSITIVE_LEARNING, SENSITIVE_THRESHOLD. RF con NO_SAMPLING/ SMOTE con NO_COST_SENSITIVE.

Le **peggiori combinazioni** invece sono:
RF con BEST_FIRST/NO_SELECTION, NO_SAMPLING/ SMOTE, SENSITIVE_LEARNING/SENSITIVE_THRESHOLD

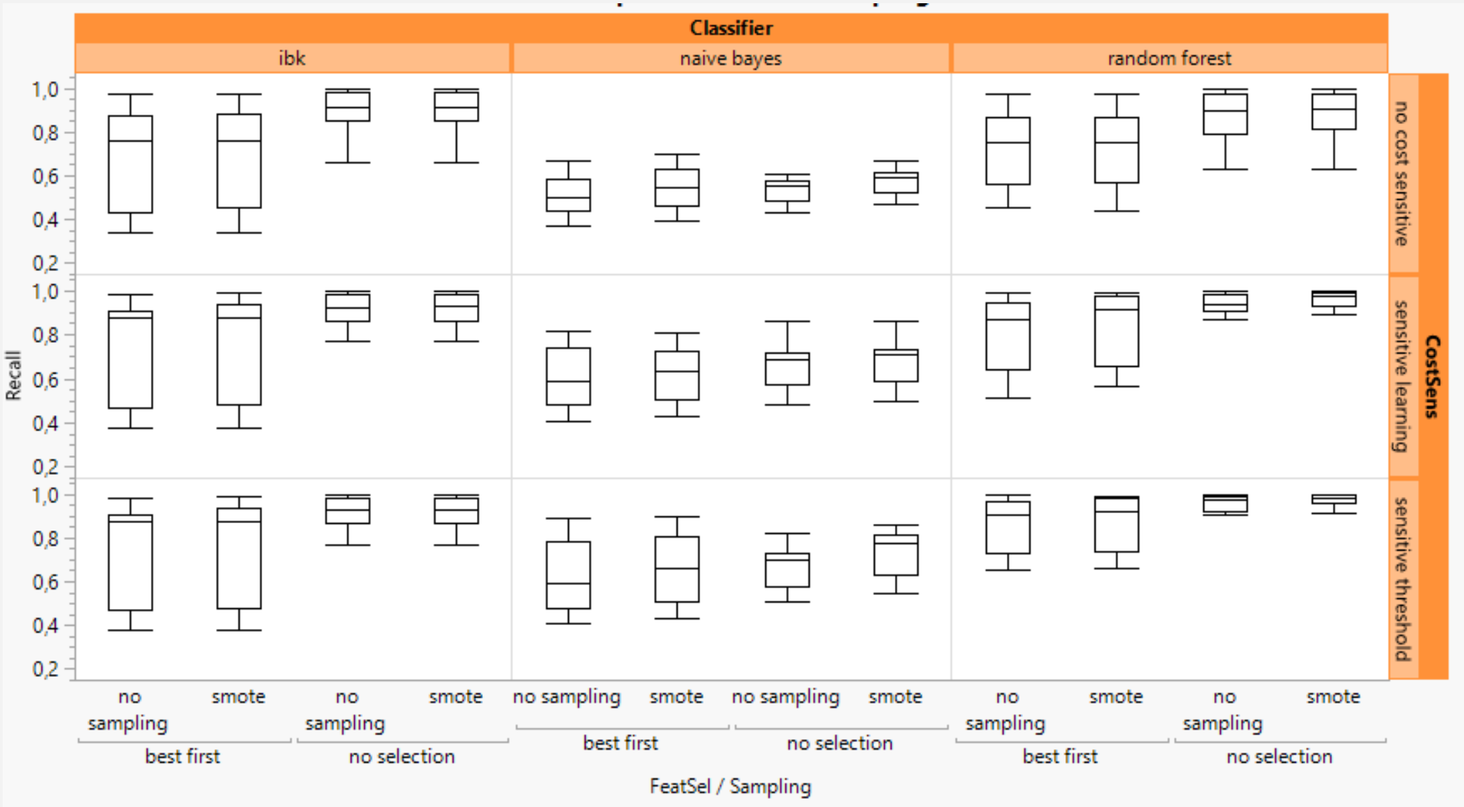
RISULTATI – ZooKeeper: Precision



Le **combinazioni migliori** sono:
IBK con NO_SELECTION,NO_SAMPLING/SMOTE, con NO_COST_SENSITIVE/SENSITIVE_LEARNING/SENSITIVE_THRESHOLD;
RF con NO_SELECTION, NO_SAMPLING/SMOTE e NO_COST_SENSITIVE

Invece le **combinazioni peggiori** le abbiamo con NB in tutte le possibili combinazioni; RF con SENSITIVE_THRESHOLD, BEST_FIRST, NO_SAMPLING e SMOTE.

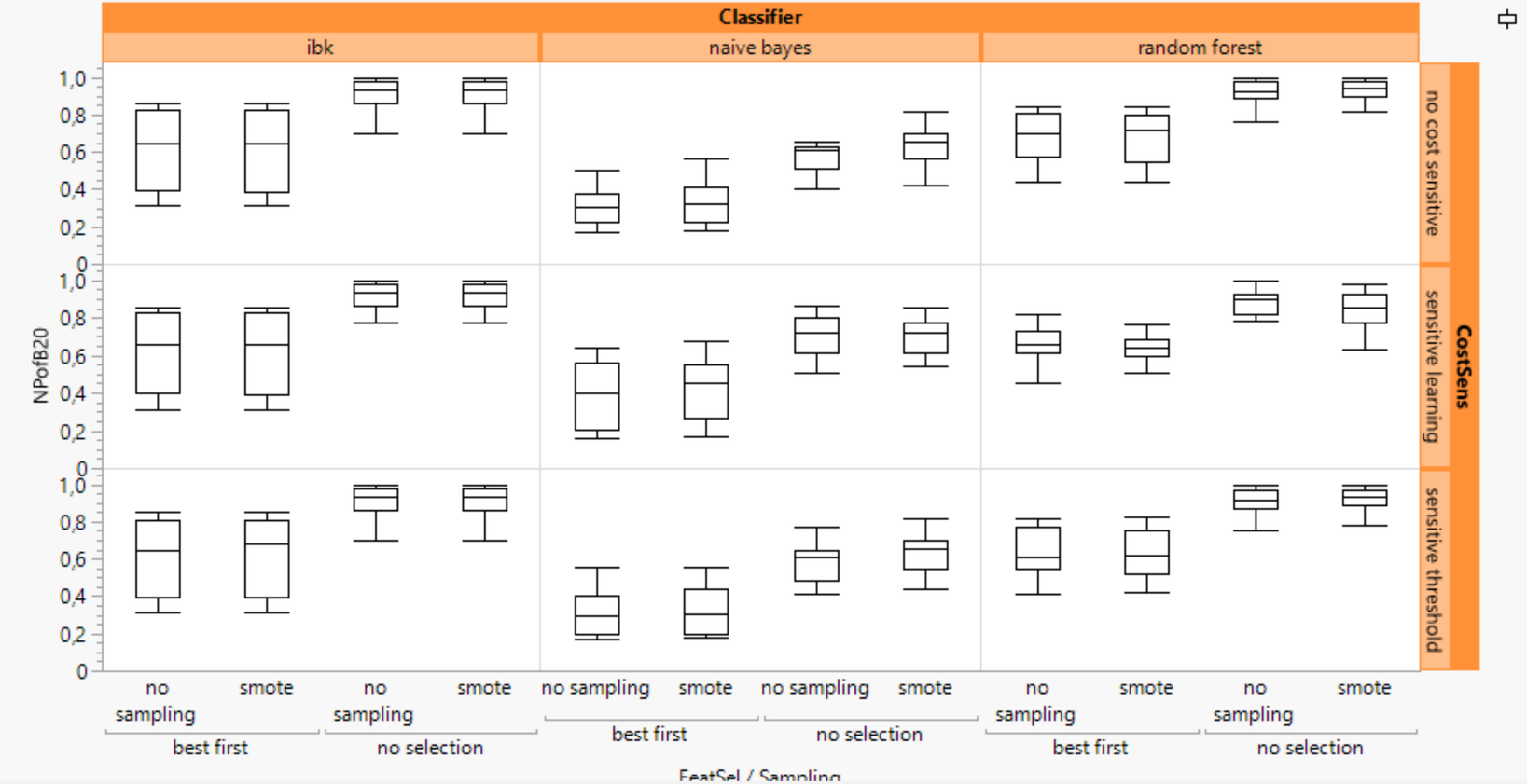
RISULTATI – ZooKeeper: Recall



Le **combinazioni migliori** sono:
RF con NO_SELECTION,NO_SAMPLING/SMOTE, SENSITIVE_THRESHOLD/SENSITIVE_LEARNING;
IBK con NO_SELECTION,NO_SAMPLING/SMOTE, SENSITIVE_THRESHOLD/SENSITIVE_LEARNING, NO_COST_SENSITIVE.

Le **combinazioni peggiori** invece sono:
IBK con BEST_FIRST,NO_SAMPLING/SMOTE, NO_COST_SENSITIVE, SENSITIVE_LEARNING, SENSITIVE_THRESHOLD;
NB con NO_SELECTION, NO_SAMPLING/ SMOTE, NO_COST_SENSITIVE, SENSITIVE_LEARNING, SENSITIVE_THRESHOLD.

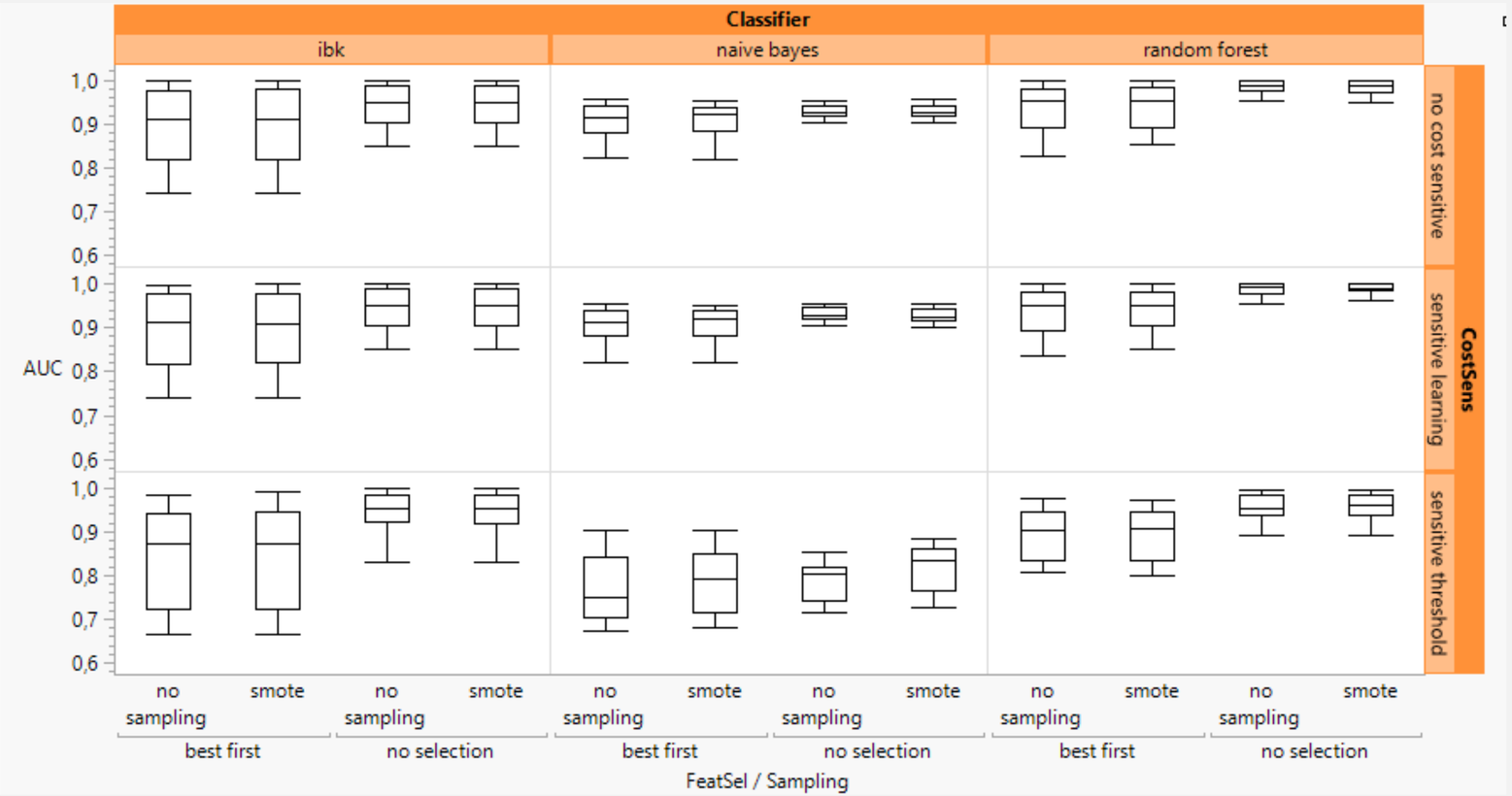
RISULTATI – ZooKeeper: NPofB20



Le **combinazioni migliori** sono:
IBK con NO_SELECTION, NO_SAMPLING/SMOTE, NO_COST_SENSITIVE_SENSITIVE LEARNING, SENSITIVE_THRESHOLD;
RF con NO_SELECTION, NO_SAMPLING/ SMOTE, NO_COST_SENSITIVE_SENSITIVE LEARNING, SENSITIVE_THRESHOLD.

Le **combinazioni peggiori** invece:
NB con BEST_FIRST,NO_SAMPLING/ SMOTE, NO_COST_SENSITIVE_SENSITIVE LEARNING, SENSITIVE_THRESHOLD;

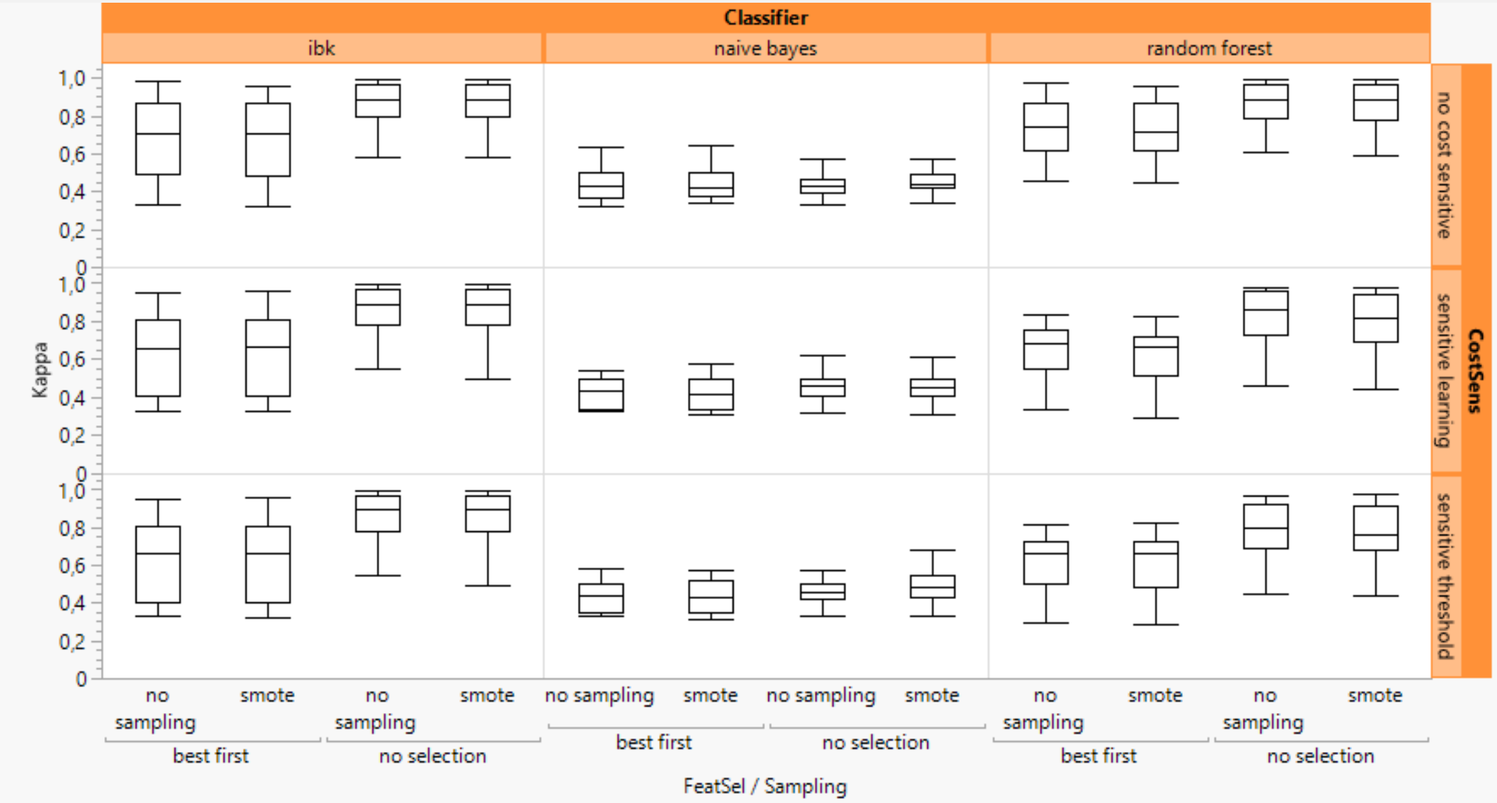
RISULTATI – ZooKeeper: AUC



Le **combinazioni migliori** sono le seguenti:
NB con NO_SELECTION, NO SAMPLING/ NO_SMOTE, NO_COST_SENSITIVE, SENSITIVE_LEARNING;
RF con NO_SELECTION, NO SAMPLING, /SMOTE, NO_COST_SENSITIVE, SENSITIVE_LEARNING.

Invece le **combinazioni peggiori** sono:
NB con BEST_FIRST, NO_SELECTION, NO_SAMPLING/SMOTE sotto SENSITIVE_LEARNING

RISULTATI – ZooKeeper: Kappa



Le **migliori combinazioni** sono le seguenti:
IBK con NO_SELECTION,NO_SAMPLING/ SMOTE, NO_COST_SENSITIVE, SENSITIVE_LEARNING,SENSITIVE_THRESHOLD;
RF con NO_SELECTION,NO_SAMPLING/ SMOTE, NO_COST_SENSITIVE, SENSITIVE_LEARNING,SENSITIVE_THRESHOLD

Le **peggiori combinazioni** invece sono tutte le possibili combinazioni fatte con il classificatore NB.

CONCLUSIONI: Bookkeper

In generale, possiamo affermare che i classificatori RF e IBK mostrano prestazioni superiori rispetto agli altri, anche se le differenze non sono particolarmente marcate. Poiché il nostro obiettivo principale è massimizzare il recall, ovvero ridurre il numero di falsi negativi (FN), alcune combinazioni specifiche si rivelano particolarmente efficaci.

- **BOOKKEEPER:** una combinazione ottimale è utilizzare il classificatore Random Forest (RF) con le seguenti caratteristiche:
    - **Feature Selection:** Best First.
    - **Sampling:** No Selection.
    - **Cost Sensitivity:** Sensitive Learning.

Questa configurazione non solo ottimizza il recall, ma fornisce anche buoni risultati per le altre metriche.

CONCLUSIONI: Zookeeper

- **ZOOKEEPER**: Per massimizzare il recall, le configurazioni raccomandate sono:
  - **Feature Selection :** No Selection.
  - **Sampling:** No Sampling o SMOTE.
  - **Cost Sensitivity:** Sensitive Threshold o Sensitive Learning.

Anche in questo caso, queste configurazioni garantiscono buone performance anche per le altre metriche.

Possiamo concludere che non esiste un classificatore nettamente migliore rispetto agli altri; non esiste una soluzione perfetta per ogni scenario ("*No Silver Bullet*"). Tuttavia, i risultati ottenuti su ZOOKEEPER risultano leggermente migliori, probabilmente a causa della diversa dimensione dei dataset ("*Size Matters*").

MINACCE ALLA VALIDITÀ

- Le tecniche di **undersampling** e **oversampling** non sono state incluse nella valutazione dei classificatori per evitare la perdita di informazioni e mantenere l'efficienza computazionale.

- Per limitare lo **snoring:**
  - L’ultima metà delle release non viene considerata
  - È stato utilizzato «incremental» che è un approccio conservativo e potrebbe approssimare le performance dei classificatori.

LINK UTILI

GITHUB

https://github.com/luigiciuf/ISW_2

SONAR CLOUD

https://sonarcloud.io/project/overview?id=luigiciuf_ISW_2

GRAZIE PER L’ATTENZIONE!