Machine Learning for Software Engineering

ISW2-2022/2023

Arianna Quinci

0325090

Agenda:

- Contesto e scopo del lavoro
- Progettazione:
  - Costruzione del dataset:
    - Raccolta dei dati
    - Impostazione della buggyness delle classi
    - Scelta delle metriche per predire la buggyness di una classe
  - Scelta di classificatori e filtri
  - Metodo di validazione
- Analisi dei risultati e conclusioni:
  - Bookkeeper
  - Tajo
  - Bookkeeper vs Tajo
- Link Github e Sonarcloud

Contesto e scopo del lavoro (1/2):

- Nella software engineering è importante prevedere sempre un'attività di testing per assicurarsi che il software sviluppato sia adeguato
- Il testing è costoso in quanto bisogna capire verso quali classi è meglio indirizzarsi per trovare bug
- Ci sono caratteristiche delle classi che le rendono più predisposte alla presenza di bug rispetto ad altre
- IDEA: Utilizziamo modelli di ML per identificare le classi che potrebbero essere soggette ad un bug, così da indirizzare l'attenzione dei tester

Contesto e scopo del lavoro (2/2):

- I modelli di machine learning vengono «allenati» su un training set e svolgono la loro previsione su un testing set

- Due concetti fondamentali:
  - Istanza→ campione indipendente da cui il modello impara
  - Attributo→ aspetto «interessante» dell'istanza

- L'approccio utilizzato in questo lavoro è di tipo black box, si utilizzano modelli di ML senza preoccuparsi della loro implementazione

- Il lavoro in questione è stato effettuato andando a raccogliere dati su due progetti opensource: Bookkeeper e Tajo

Progettazione:
costruzione del dataset (1/3)

- Raccolta dei dati:
  - Ticket su Jira per determinare «injected version», «opening version» e «fixed version»
  - Qualora l'IV non fosse presente, essa è stata calcolata tramite «Proportion» con:
    - cold start per i primi tickets a partire da altri progetti Opensource di Apache: Avro, OpenJPA,Syncope e Storm.
    - moving window con taglia della finestra pari al 5% per i tickets successivi
  - Github per ottenere i commit relativi ad un ticket Jira e le informazioni sulle modifiche effettuate

Progettazione:
costruzione del dataset (2/3)

- Impostazione della buggyness di una classe:
  - Per impostare la buggyness di una classe sono state utilizzate due metodologie differenti per le coppie classe-release di training set e testing set
  - Training set→ deve essere coerente con le informazioni a disposizione nella data release, l'attributo «buggy» è stato impostato true quando: OV<=release<FV
  - Testing set→ non deve essere affetto da snoring. Sono stati utilizzati tutti i ticket a disposizione, una classe è stata considerata buggy quando: IV<=release<FV

Progettazione:
costruzione del dataset (3/3)

- Scelta delle metriche:

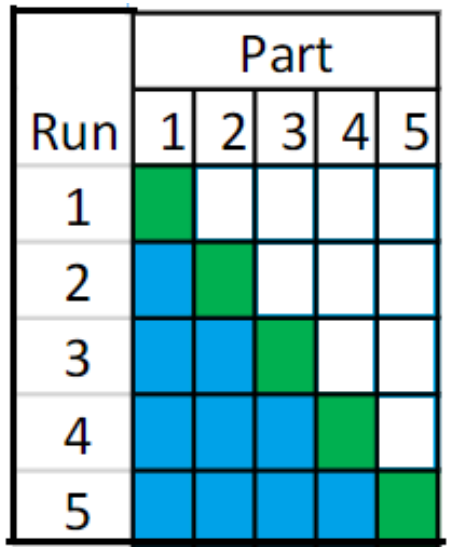
  - Le metriche scelte come attributi per la classe sono:
    - Size→ taglia della classe
    - Numero di autori
    - NR→ numero di revisioni dalla release 0
    - Age→ età della classe in termini di mesi
    - Metodi pubblici
    - Numero di commenti
    - Change set size→ numero di file coinvolti dallo stesso commit
    - Churn→ somma delle righe di codice aggiunte e rimosse nei commit

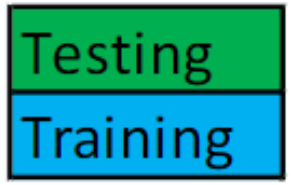
Progettazione:
scelta di classificatori e metodologie

- I classificatori presi in esame sono naive Bayes, ibk e random forest
- Qui la lista delle combinazioni utilizzate di sampling, feature selection (con backward search) e cost sensitive classifier:
  - No feature selection e no sampling
  - Feature selection
  - Simple oversampling
  - SMOTE
  - Feature selection e SMOTE
  - Cost sensitive classification (sensitive learning)

Progettazione:
validazione dei classificatori

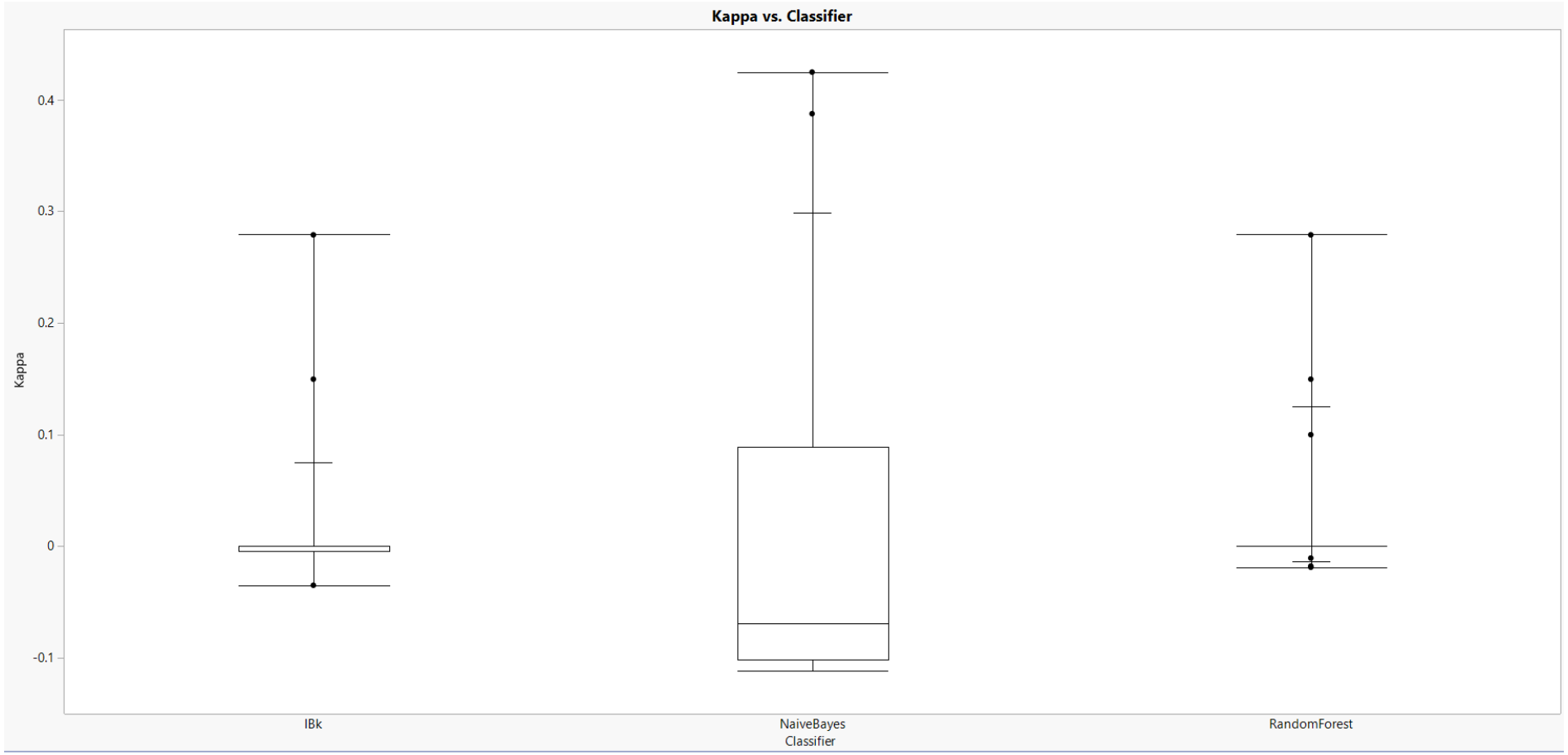
- Tecnica di validazione: walk forward
- Essendo una tecnica time-series essa tiene conto del tempo
- Si realizza andando a dividere il dataset in testing set e training set come segue:





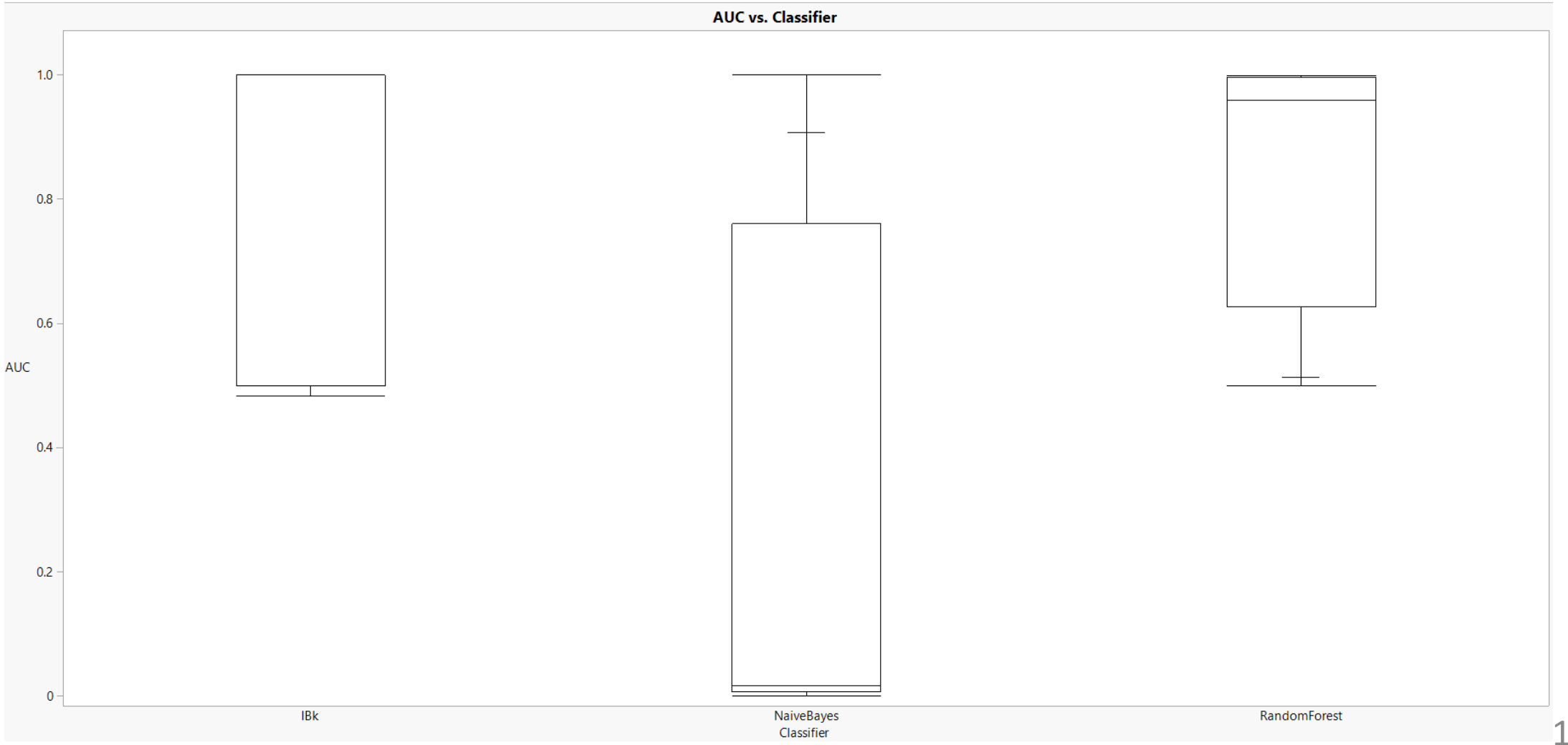
Analisi dei risultati e conclusioni: Bookkeeper (1/9)

- Di seguito il grafico di kappa per i tre classificatori considerati
- K indica quanto il classificatore è migliore o peggiore rispetto ad un classificatore dummy



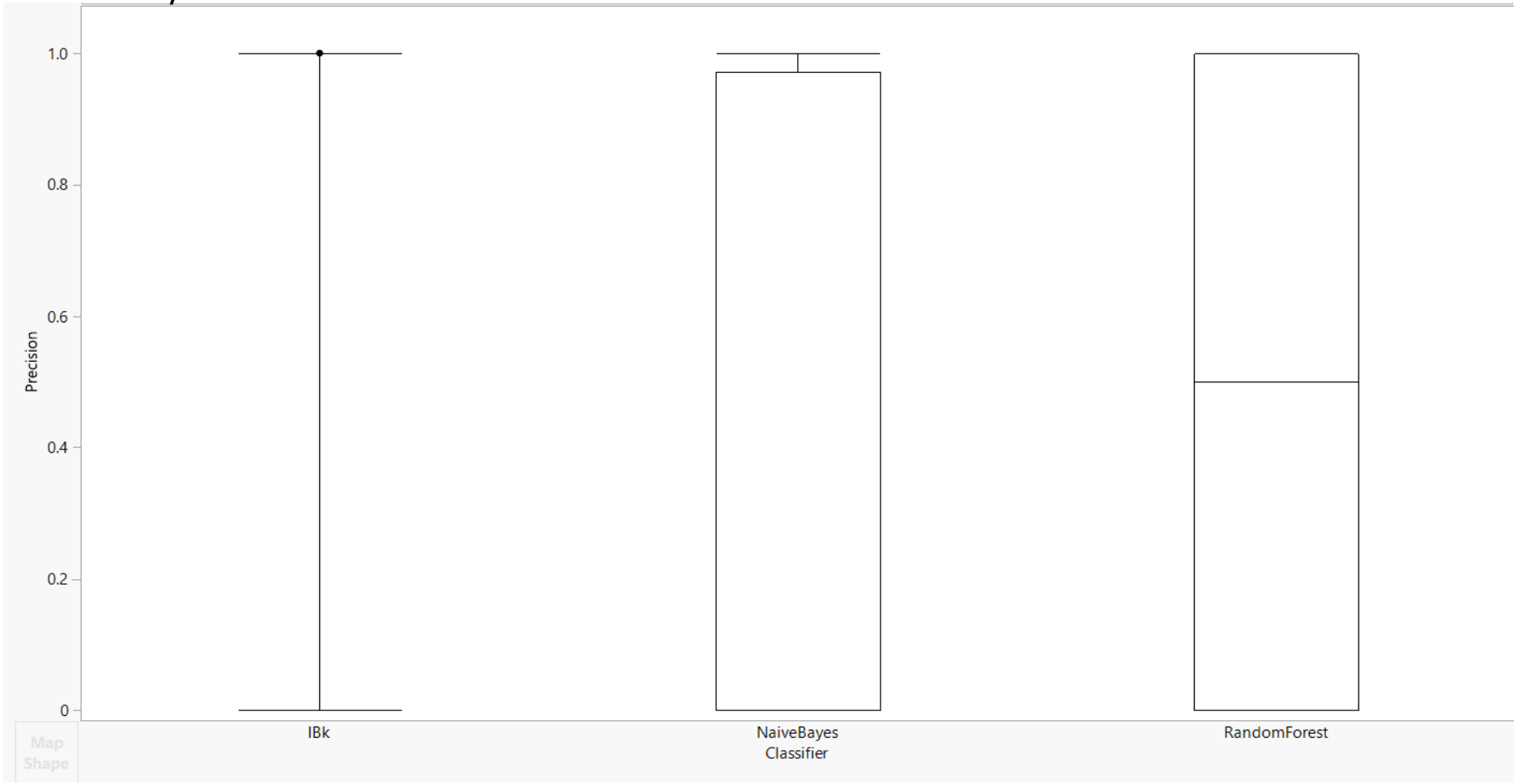
Analisi dei risultati e conclusioni: Bookkeeper (2/9)

- AUC: rappresenta l'area sotto la ROC
- ROC: curva che mostra l'andamento del classificatore per ogni threshold. Traccia true positives e false positives.



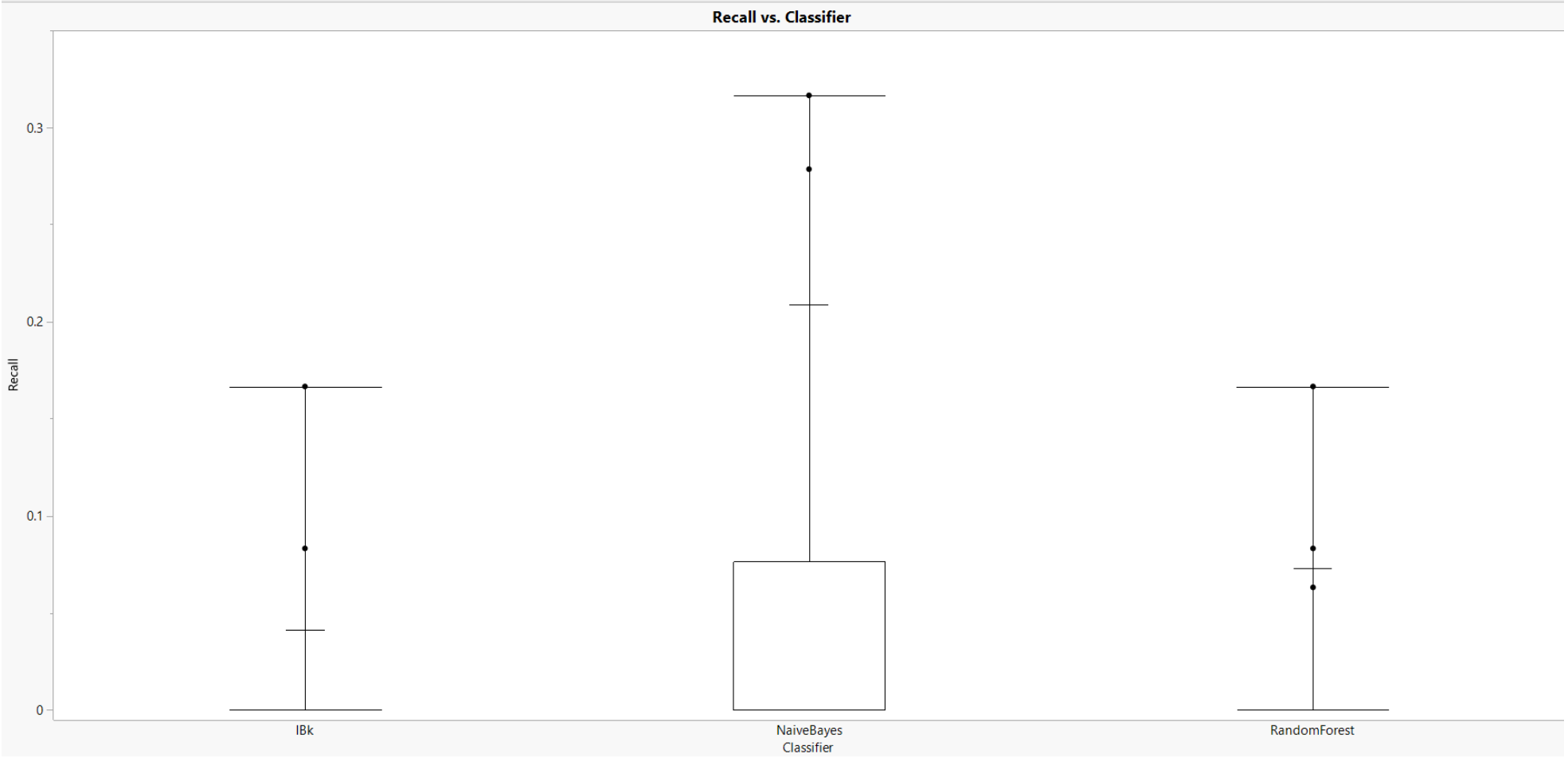
Analisi dei risultati e conclusioni: Bookkeeper (3/9)

- Qui sotto i grafici della precision
- Precision=TP/TP+FP



Analisi dei risultati e conclusioni: Bookkeeper (4/9)

- Qui sotto i grafici della recall
- Recall=TP/TP+FN

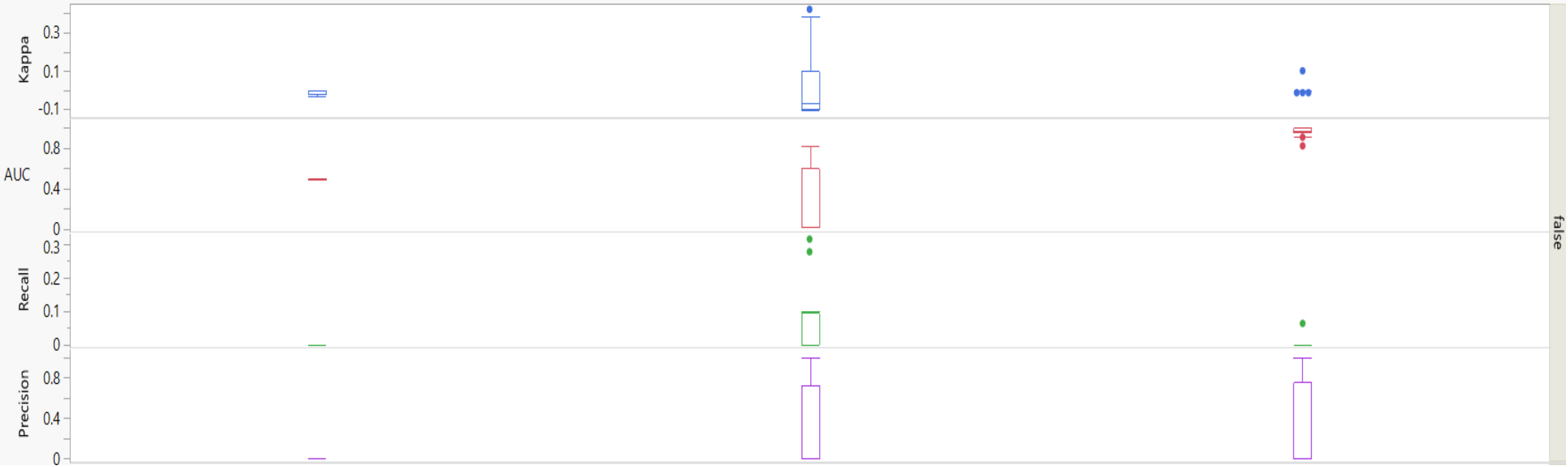


Analisi dei risultati e conclusioni: Bookkeeper (5/9)

- E’ difficile identificare un classificatore dominante
- Naive Bayes ha un range di valori più ampio degli altri due classificatori
- Per capire se c’è qualche classificatore dominante bisogna andare ad interpretare i grafici relativi alle diverse combinazioni di metodologie usate

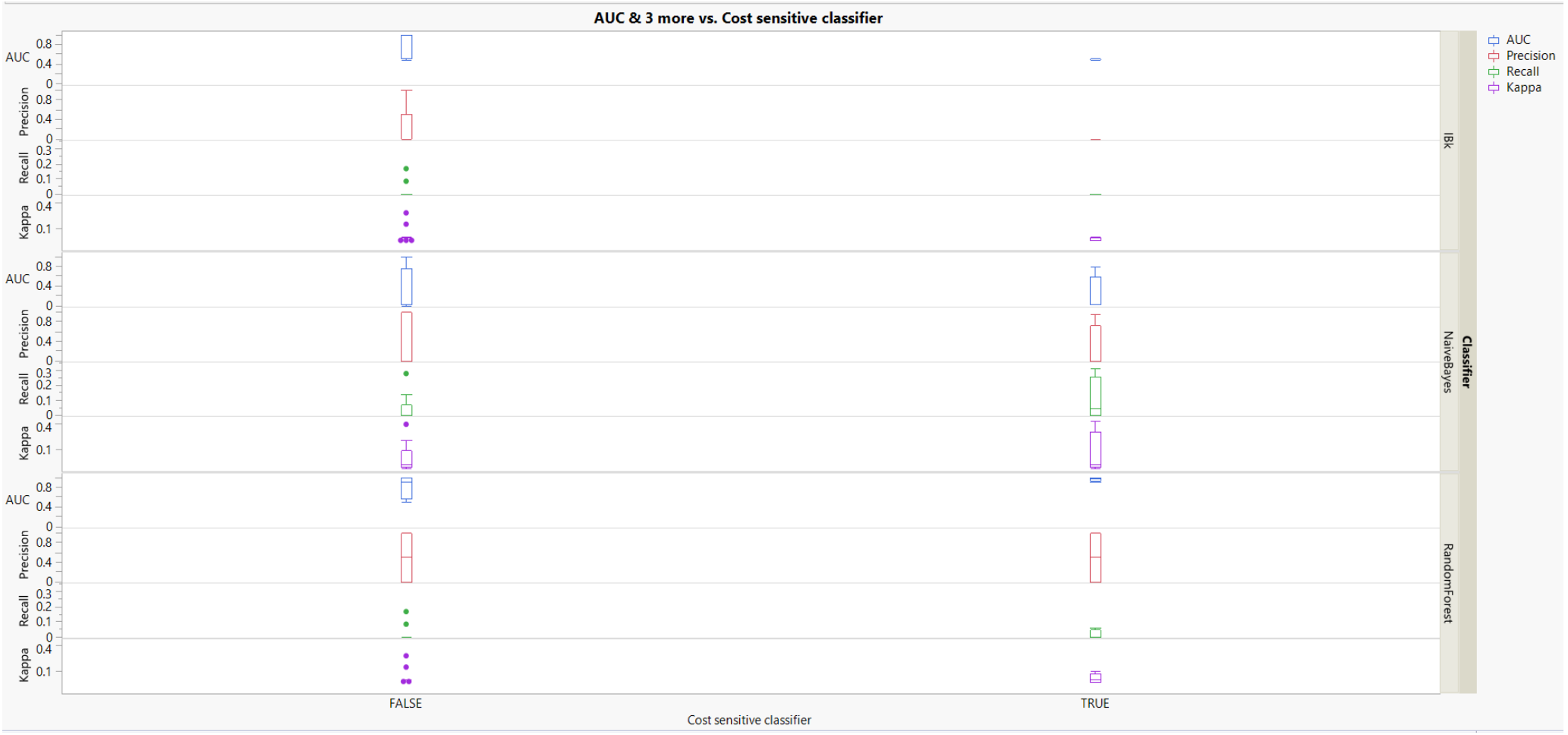
Analisi dei risultati e conclusioni: Bookkeeper (6/9)

- Di seguito il grafico dei vari classificatori (in ordine ibk, naive Bayes e random forest) senza feature selection:



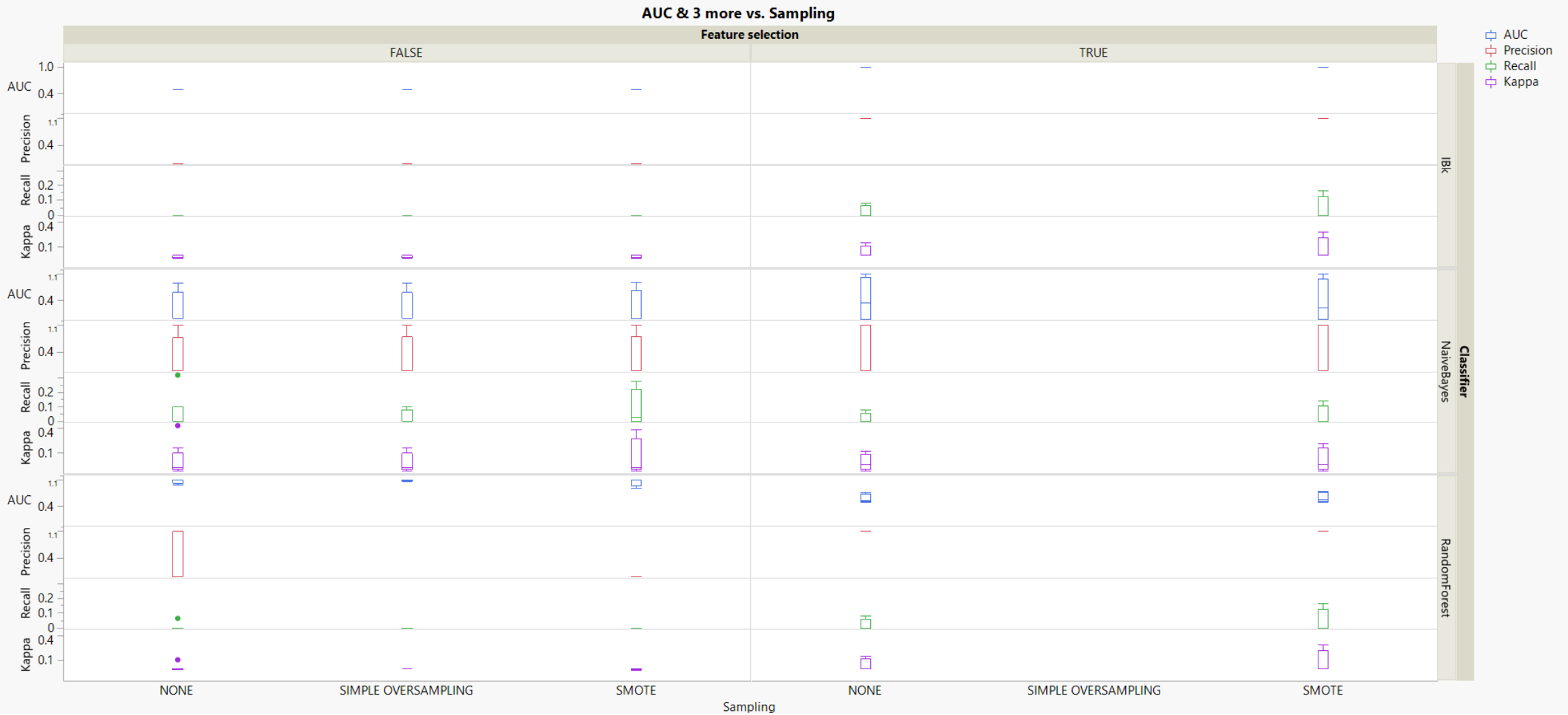
Analisi dei risultati e conclusioni: Bookkeeper (7/9)

- Di seguito i grafici dell'andamento dei tre classificatori con e senza cost sensitive classification
- La cost sensitive classification riduce gli outliers



Analisi dei risultati e conclusioni: Bookkeeper (8/9)

- Osservazione: la combinazione simple oversampling con feature selection non presenta grafico in quanto non è stata presa in considerazione in questo lavoro

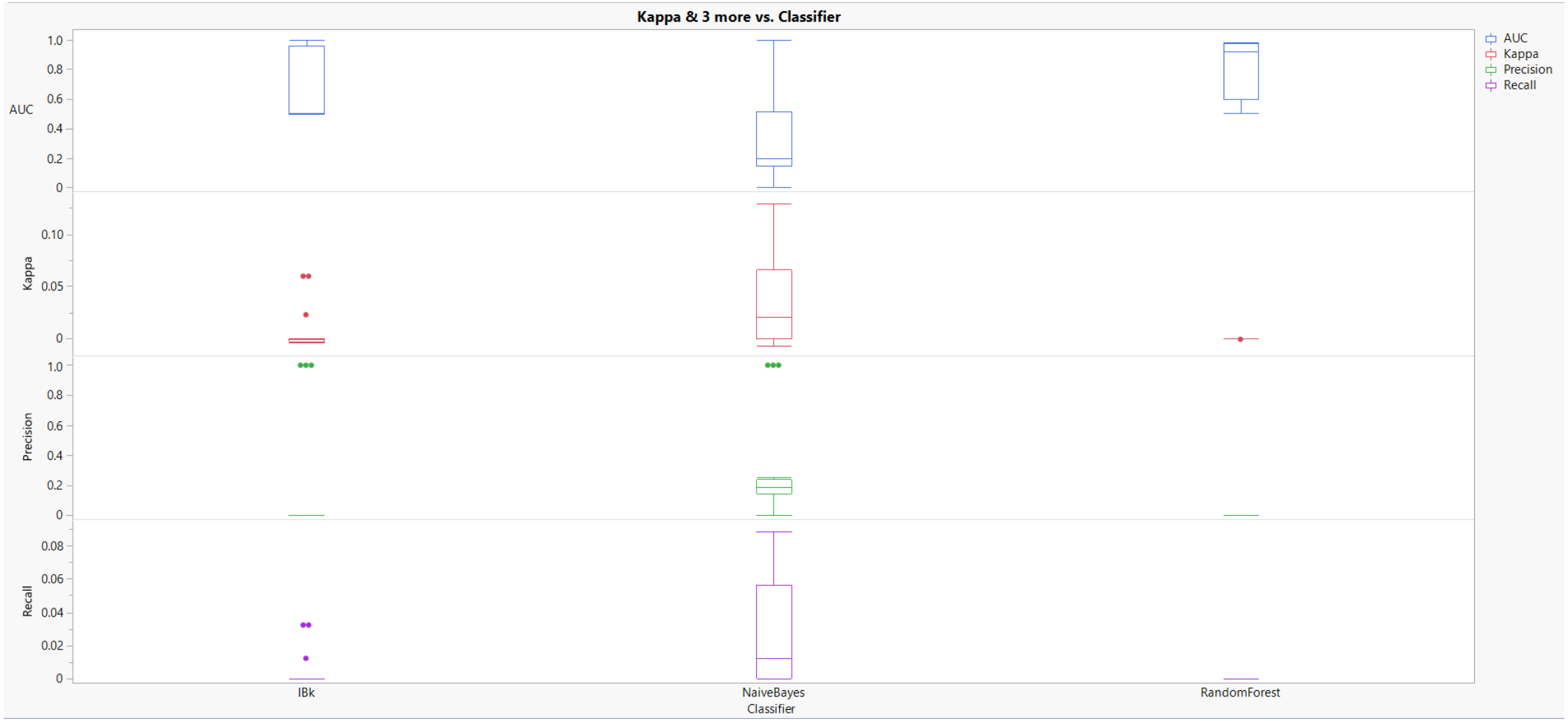


Analisi dei risultati e conclusioni: Bookkeeper (9/9)

- Andando a vedere il grafico con feature selection e sampling emerge che il classificatore migliore in questo caso potrebbe essere Ibk o random forest
- Gli outliers diminuiscono applicando queste tecniche
- In conclusione non si riesce ad identificare un classificatore dominante, ma se si utilizza feature selection i classificatori si comportano meglio

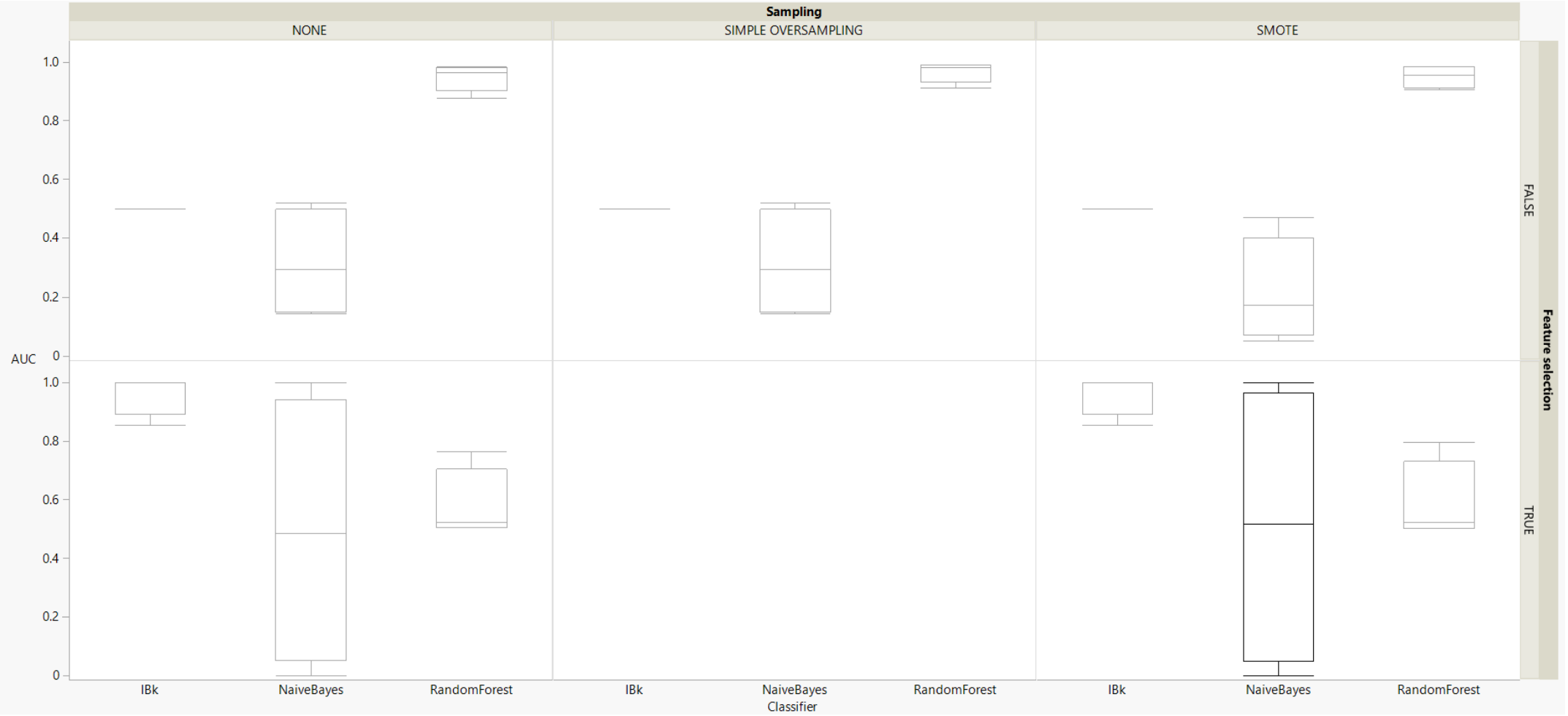
Analisi dei risultati e conclusioni: Tajo (1/5)

- Naive Bayes sembra comportarsi meglio degli altri



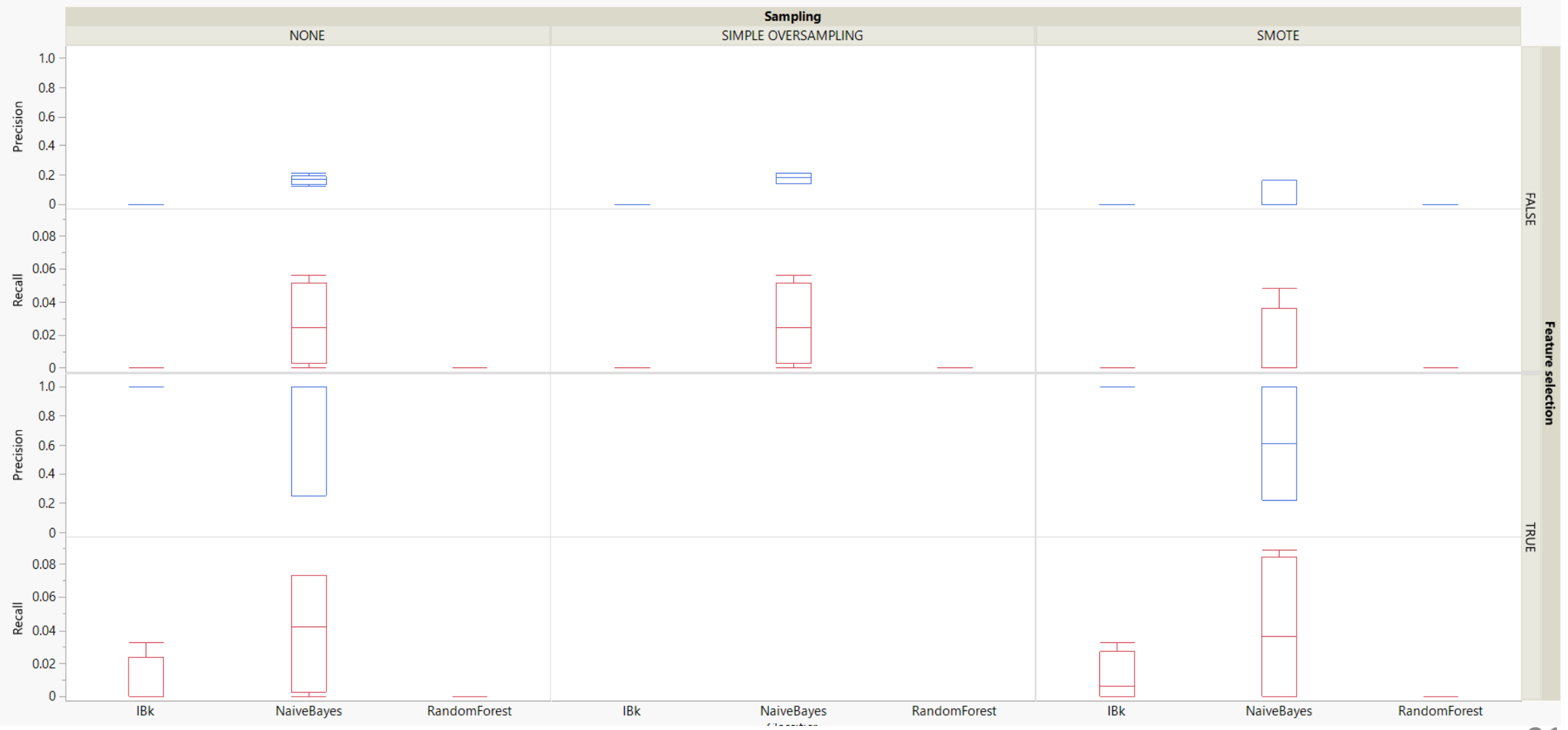
Analisi dei risultati e conclusioni: Tajo (2/5)

- Con feature selection AUC migliora per naive Bayes
- Il valore migliore per AUC con feature selection è quello di ibk



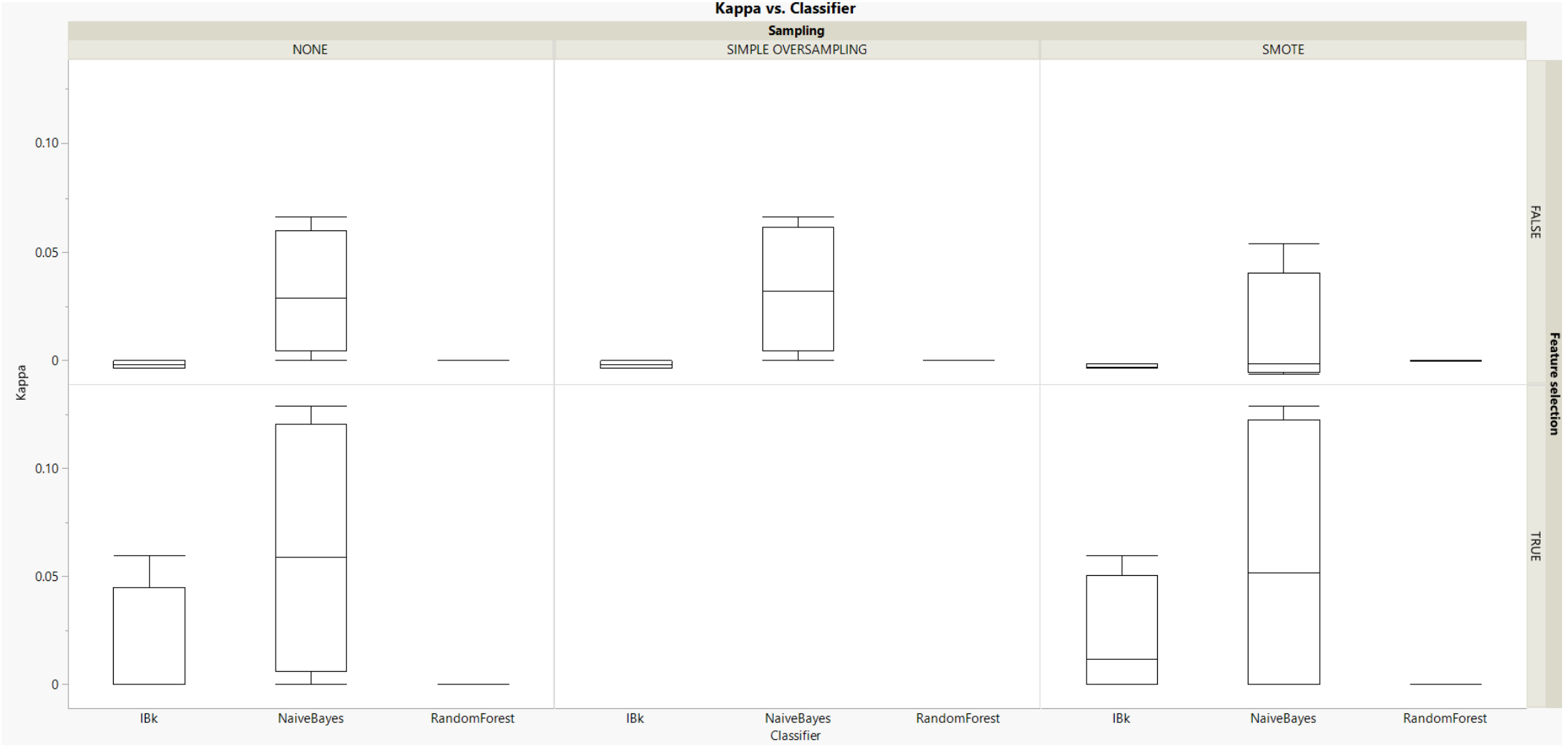
Analisi dei risultati e conclusioni: Tajo (3/5)

- Benchè ibk abbia un valore più alto per AUC, naive Bayes continua a risultare il migliore per precision e recall anche in presenza di SMOTE e feature selection



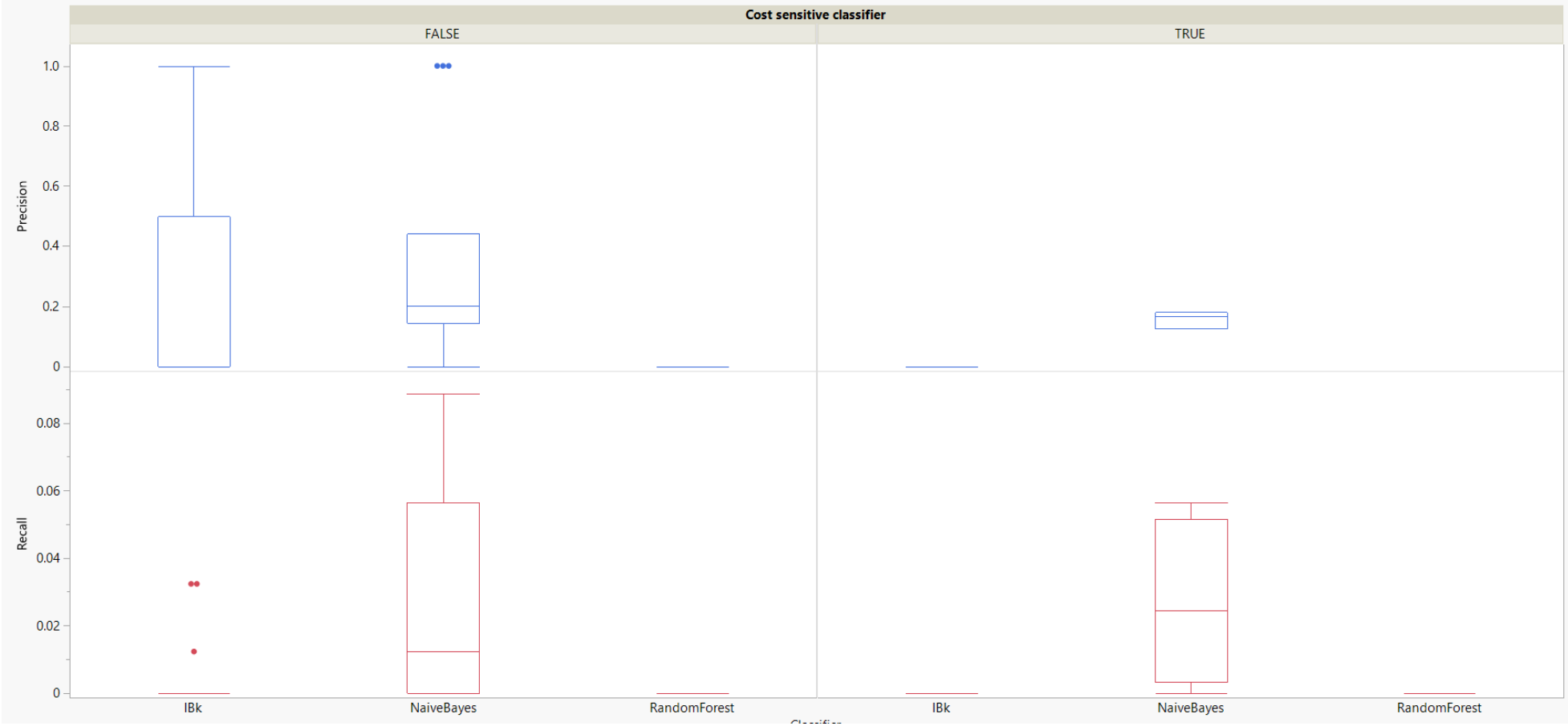
Analisi dei risultati e conclusioni: Tajo (4/5)

- Kappa è migliore in naive Bayes in presenza o in assenza sia di balancing che di feature selection



Analisi dei risultati e conclusioni: Tajo (5/5)

- Di seguito il grafico relativo a precision e recall in caso di cost sensitive classifier con sensitive learning (FN=10*FP)
- Non migliora le performance dei classificatori



- In definitiva il classificatore che risulta migliore è naive Bayes
- Miglioramenti si hanno in presenza di feature selection e balancing realizzato con tecnica SMOTE

Analisi dei risultati e conclusioni: Bookkeeper vs Tajo

- Per i due progetti sono stati adottati:
  - stessi classificatori
  - stesse combinazioni di sampling, feature selection e sensitive learning

- Si può concludere che i risultati ottenuti con Tajo sono migliori di quelli ottenuti con Bookkeeper, a cosa è dovuto?
  - A come sono fatti i due dataset: il dataset di Tajo contiene più dati (size does matter!!!)

Link Github e Sonarcloud:

- Github repository:
  - https://github.com/ariannaquinci/deliverables-isw2.git
- Sonarcloud:
  - https://sonarcloud.io/project/overview?id=ariannaquinci_deliverables-isw2