BookKeeper:
https://github.com/AlessDea/ISW2_milestone1
https://sonarcloud.io/summary/overall?id=AlessDea_ISW2_milestone1

Syncope:
https://github.com/AlessDea/ISW2_milestone2
https://sonarcloud.io/summary/overall?id=AlessDea_ISW2_milestone2



# Machine Learning for Software Engineering

**De Angelis Alessandro**
**0317176**

# Introduzione

- L'obiettivo dello studio è quello di misurare l'andamento delle prestazioni di tre classificatori nei casi di applicazione di tecniche di feature selection, balancing e sensitivity.
- I classificatori considerati sono:
  - **RandomForest**
  - **NaiveBayes**
  - **Ibk**
- Per quanto riguarda le tecniche, sono state considerate:
  - **Feature Selection**: No feature Selection e Best First
  - **Balancing**: Undersampling
  - **Sensitivity**: Sensitive Learning

# Introduzione

- Sono stati analizzati due progetti di Apache Foundation:
  - **BookKeeper**
  - **Syncope**
- Sono stati creati quindi due **dataset**, uno per progetto, che riportano la buggyness delle classi
- Per la creazione dei dataset sono state utilizzate:
  - la libreria **JGit** che permette di ottenere informazioni da Git
  - **Jira API** per ottenere I ticket relativi a bug da Jira

# Progettazione

- Gli step eseguiti per la creazione del dataset sono:

1. retrieve delle release da **Jira** e da **Git** (tag)
2. filtraggio delle release fra quelle ottenute da Jira e quelle ottenute da Git: vengono mantenute solo le release ottenute da Git che sono presenti anche su Jira
3. Per ogni release prendi tutte le **classi Java** e calcola le **metriche**
4. Retrieve dei ticket da Jira
    1. Scarta i ticket che non hanno una **FV**
    2. Assegna ad ogni release i ticket che hanno, in quella release, la FV
5. Per ogni versione e per ogni file setta la **buggyness**
    1. se un ticket non ha la IV allora viene calcolata utilizzando la **Proportion**
6. Viene scartata l'ultima metà delle release

# Progettazione

- Sono stati considerati solo ticket relativi a Bug con risoluzione fixed e stato resolved o closed.
- Su jira la FV corrisponde all'**AV**
- Un ticket di Jira può avere più **IV** e più **FV**, per questo sono state prese la più vecchia IV e la più recente FV.
- La **Proportion** è stata applicata a quei ticket che non presentavano una IV
  - in particolare è stata utilizzata l'**Incremental Proportion**.

# Progettazione

- Le metriche considerate per la creazione del dataset sono:
  - **LOCs:** numero di linee di codice
    - Maggior numero di LOC maggiore probabilità di bug
  - **Churn:** added - deleted LOCs
    - Un churn alto significa che sono state effettuate molte modifiche su quel file quindi c'è alta probabilità di introduzione di bug
  - **Age:** età della classe
    - Una classe più nuova potrebbe avere dei bug
  - **Weighted Age:** età della classe pesata sul numero di LOCs
    - Come sopra (Age) ma con un numero di LOCs alto la probabilità di avere bug aumenta
  - **Number of Authors:** numero di autori per una determinata classe
    - Più autori hanno toccato una classe e maggiore è la probabilità che che possano essere stati introdotti dei bug

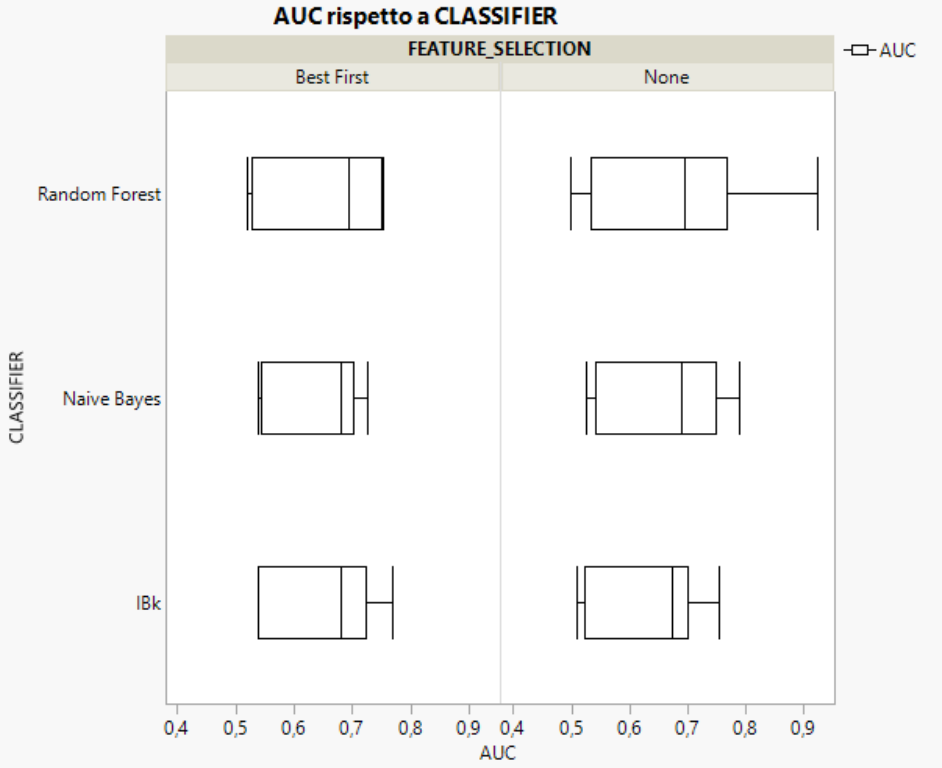
# Progettazione

- **Revisions**: numero di commit in cui la classe è stata toccata
  - Se una classe è stata toccata da un alto numero di commit durante una release è più probabile che possano essere stati introdotti dei bug
- **LOC Touched**: numero di LOCs toccate
  - Pià LOCs sono state toccate e più è probabile che siano stati introdotti dei bug
- **LOC Added**: numero di LOCs aggiunte
  - Con l'inserimento di molti LOCs c'è più probabilità di introduzione di bug
- **Avg Set Size**: numero medio di file che sono stati toccati da commit insieme al file specifico
  - Molti file in un commit possono significare un grande cambiamento nel codice e introduzione di bug
- **Number of Fix**: numero dei fix dei bug subito dalla classe
  - Un alto numero di fix non esclude l'assenza di altri bug

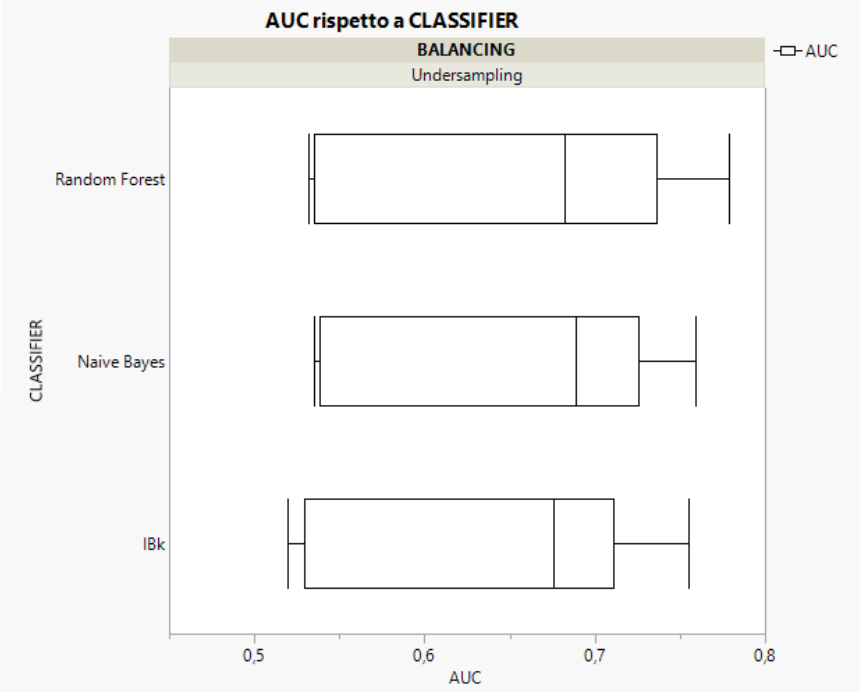
# Progettazione

- Il dataset è un file **CSV**, viene quindi trasformato nel formato **ARFF** per poterlo analizzare con **Weka**.
- Prima di poter effettuare lo studio c'è bisogno di **validare** il dataset.
  - In particolare è stata utilizzata la tecnica del **WalkForward** in quanto i dati sono strettamente legati ad **eventi temporali**.
  - Il dataset è stato quindi diviso in **training** e **testing** e in particolare sono state effettuate **N-1 run** di WalkForward
    - quindi ci sono N-1 training set e N-1 testing set
- A questo punto è stato possibile analizzare i dati ottenuti.
- Per ogni classificatore e tecniche sono state calcolate e analizzate:
  - **AUC**
  - **Kappa**
  - **Precision**
  - **Recall**

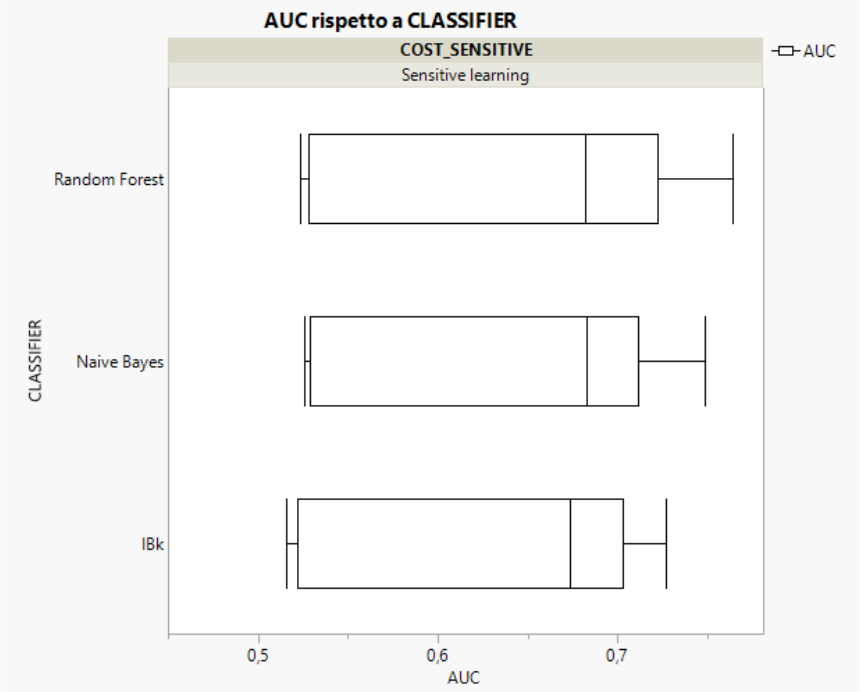
# Discussione – BookKeeper - AUC




- Random Forest è il classificatore che sembra comportarsi meglio in entrambi i casi.
- Random Forest nel caso senza feature selection sembra comportarsi meglio rispetto a quando viene applicata
- Naive Bayes non viene pressochè influenzato dall'applicazione del Best First

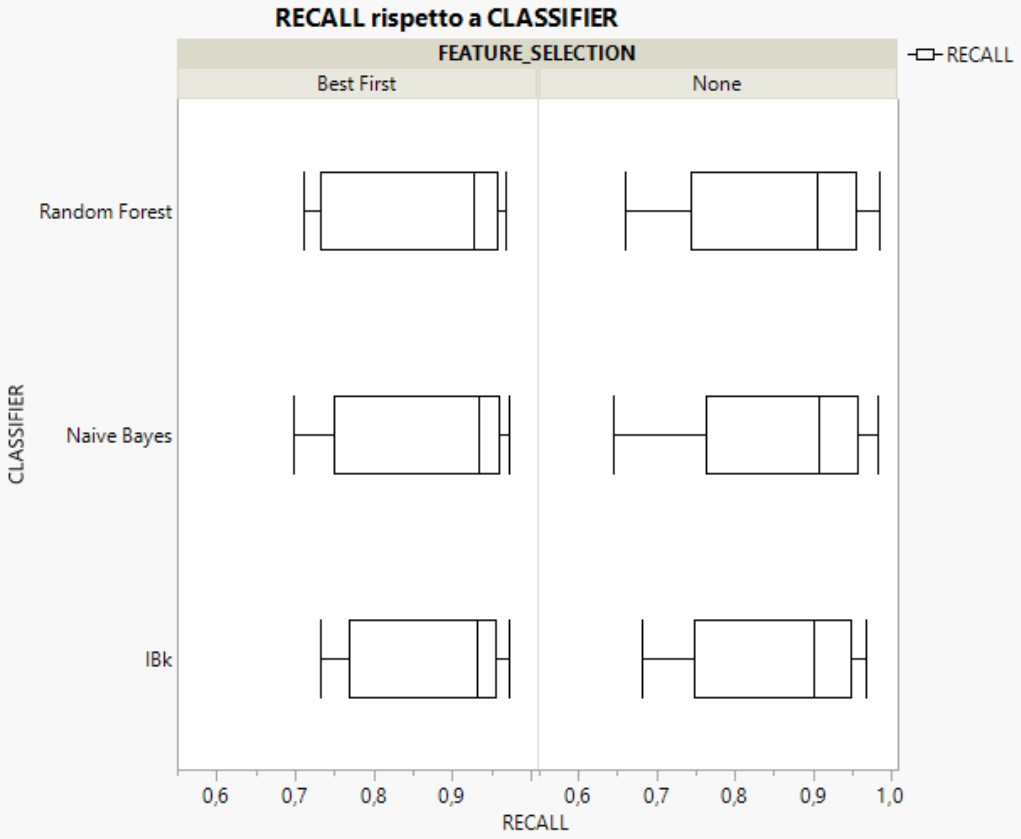



Random Forest si comporta meglio in caso di Undersampling

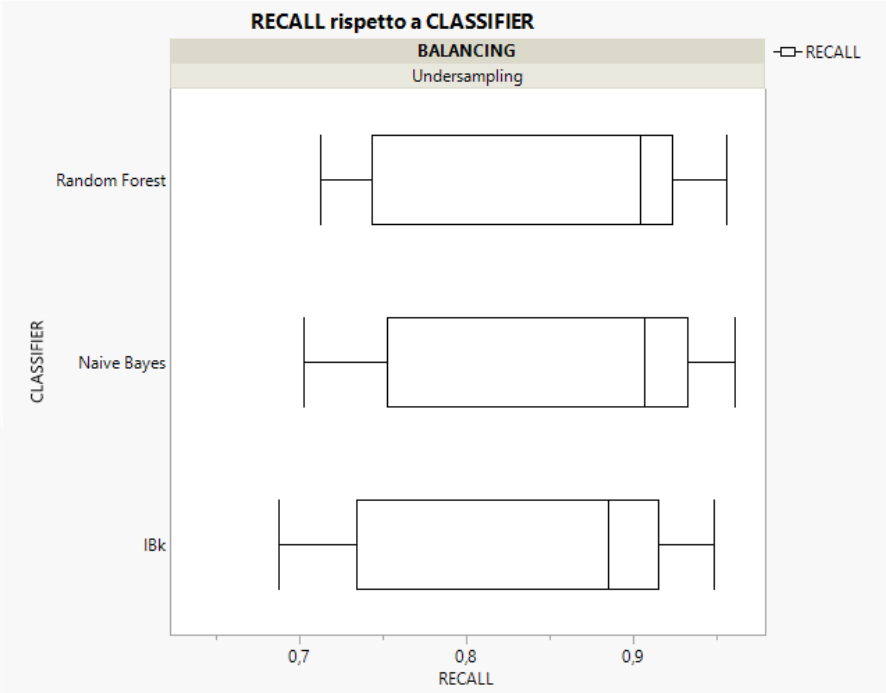


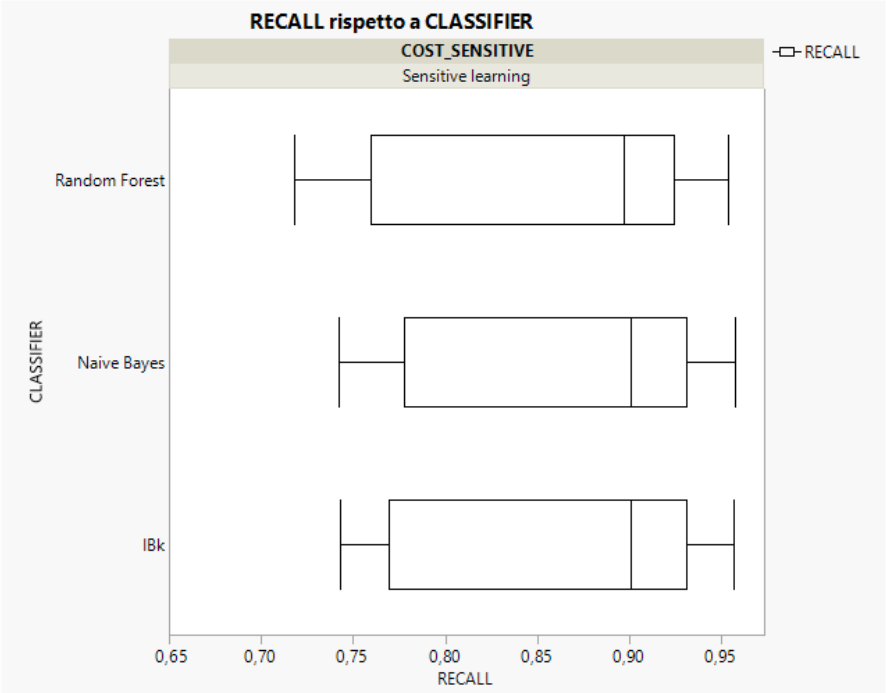

Con Sensitive learning il classificatore che sembra comportarsi meglio è Random Forest

# Discussione – BookKeeper - Recall




- In caso di no feature selection il classificatore che sembra comportarsi meglio è Ibk
- In caso di feature selection Ibk ha comunque prestazioni leggermente migliori
- Bisogna notare però che l'andamento generale è quasi invariato, questo perchè la tecnica di feature selection serve principalmente a ridurre i costi computazionali dell'apprendimento

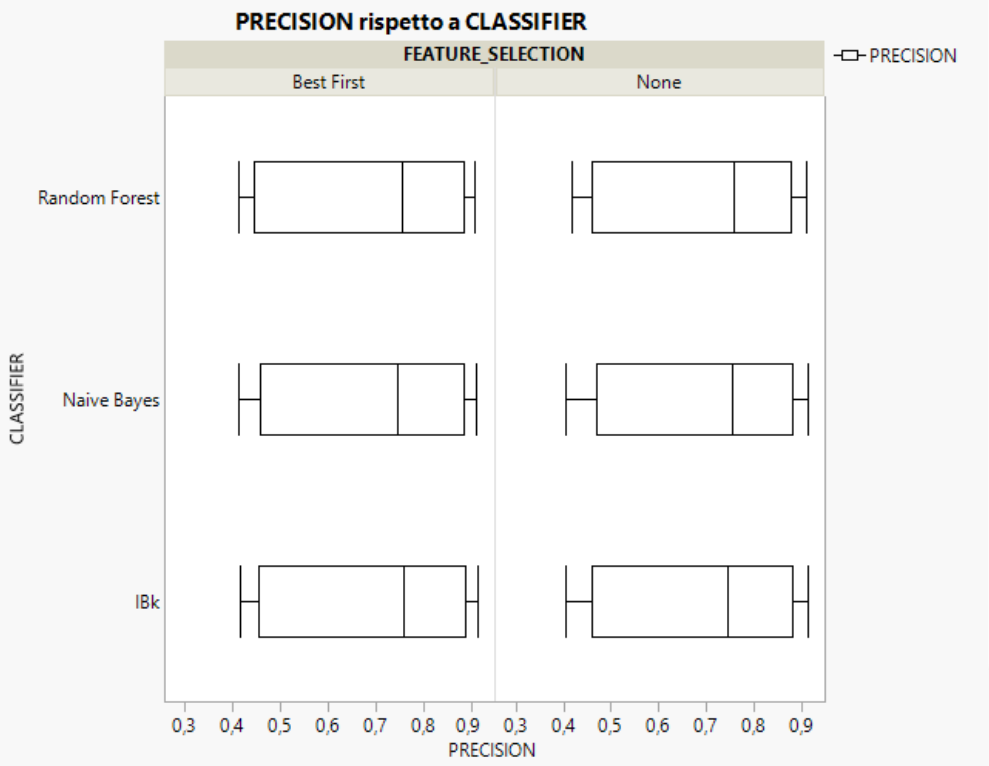


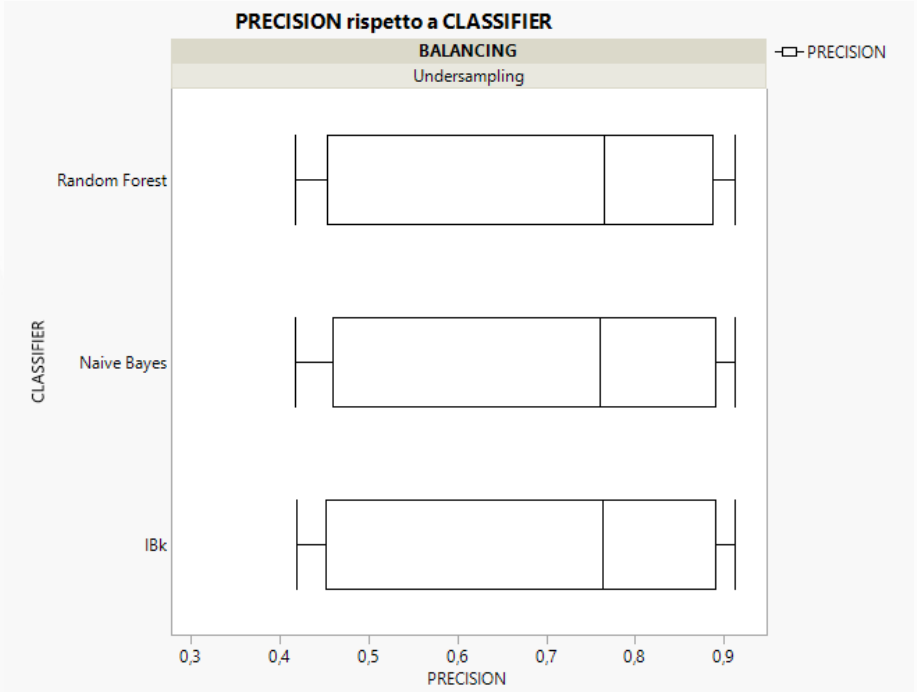

Naive Bayes sembra comportarsi meglio caso di Undersampling

Anche con cost sensitive Naive Bayes si comporta meglio. Come ci si aspetta le performance, applicando cost sensitive, migliorano poichè è stata abbassata la threshold e di conseguenza sono diminuiti i falsi negativi

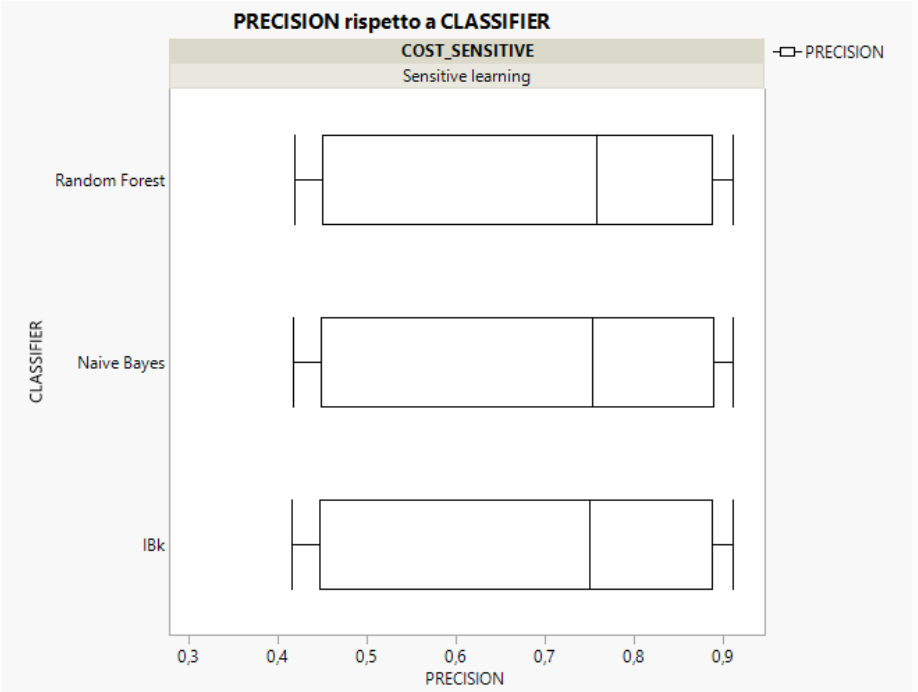
# Discussione – BookKeeper - Precision




- Per la precision i tre classificatori sembrano comportarsi pressochè in maniera molto simile, quello che ha prestazioni leggermente migliori è Ibk in caso di no feature selection
- In caso di feature selection Ibk è comunque quello che si comporta meglio



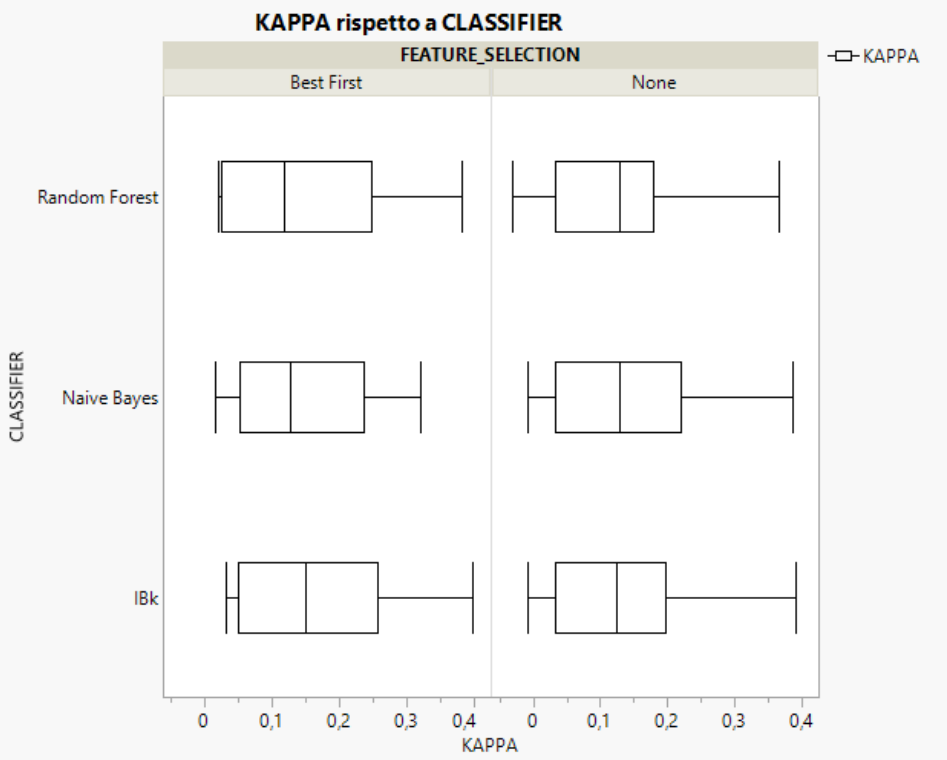

Ibk sembra comportarsi meglio in caso di Undersampling



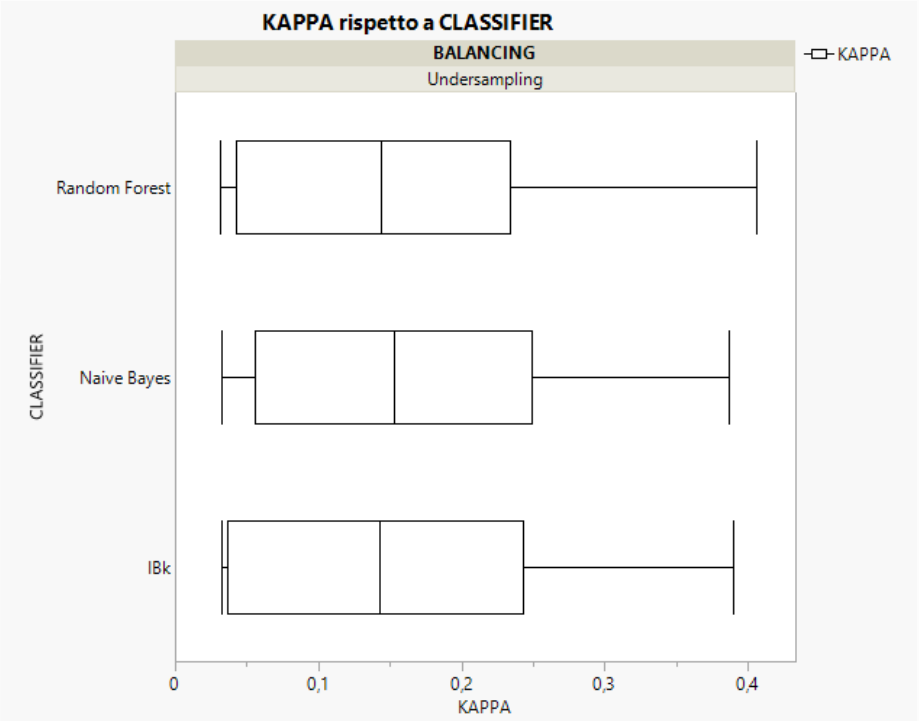

In caso di sensitive learning invece random forest ha una prestazione leggermente migliore

Entrambe le tecniche non sembrano apportare troppi benefici, questo perchè la precision è già abbastanza alta senza l'applicazione di nessuna tecnica

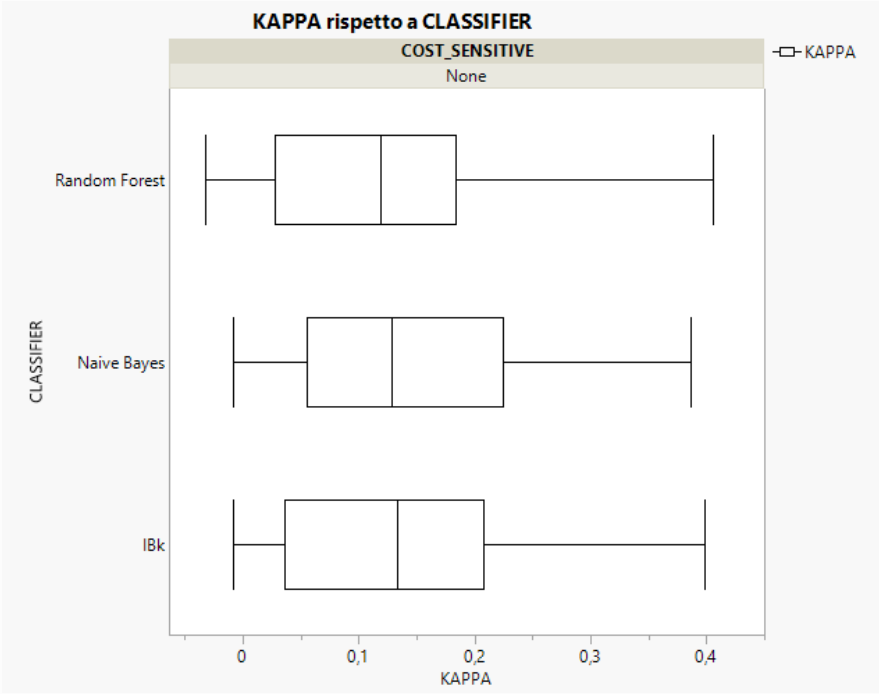
# Discussione – BookKeeper - Kappa




- Ibk si comporta meglio in caso di feature selection
- Senza feature selection invece Naive Bayes ha prestazioni migliori

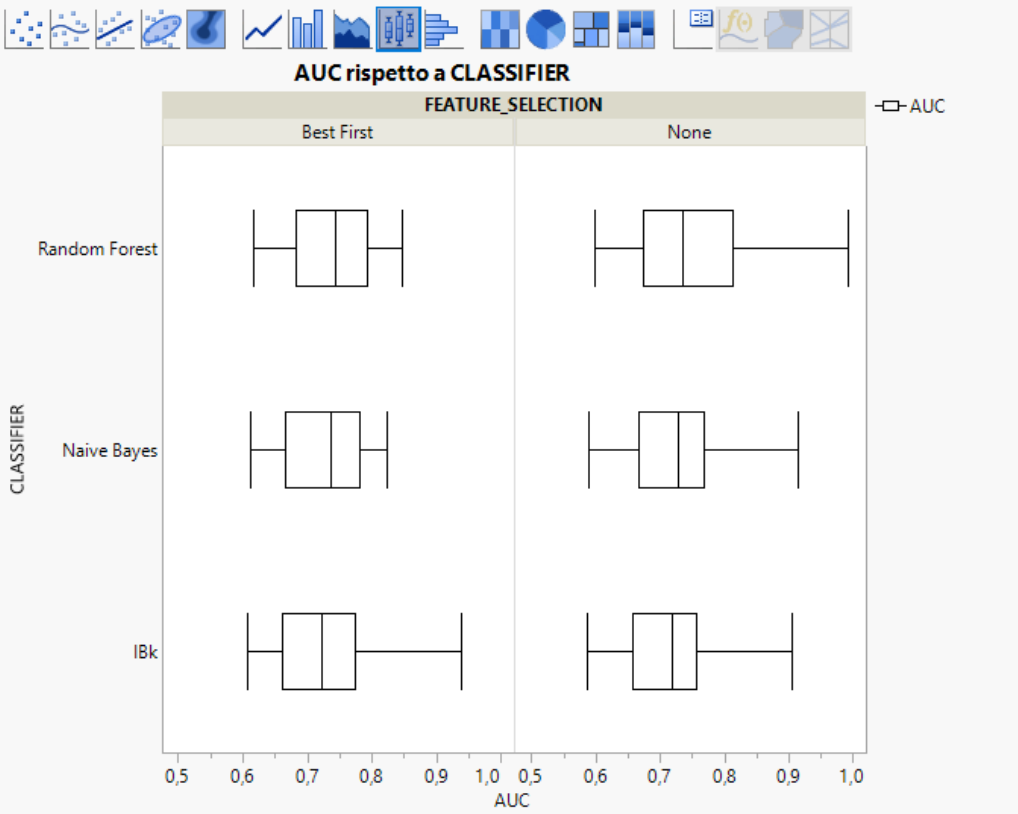



Naive Bayes si comporta meglio in caso di undersampling

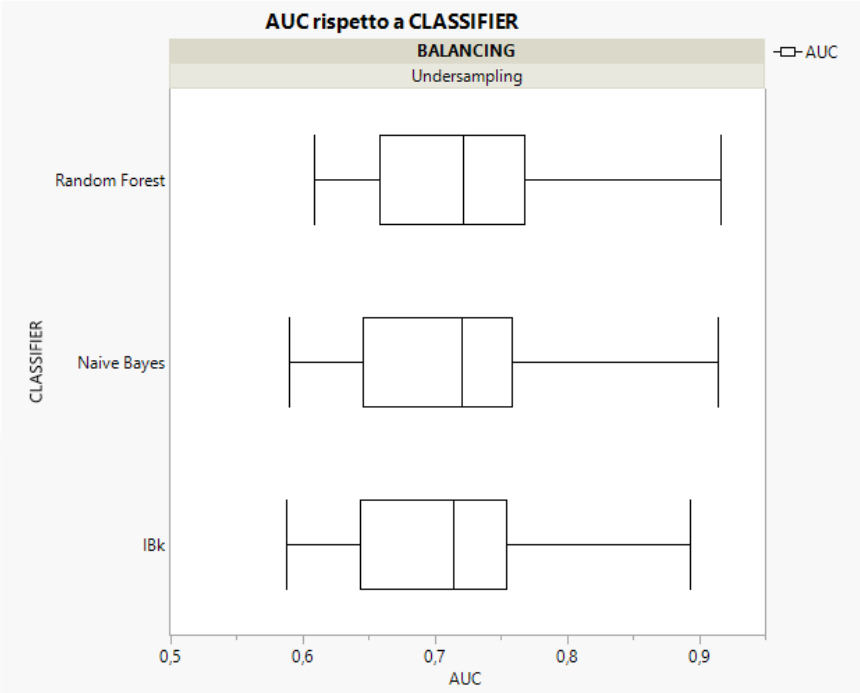



Naive Bayes ha prestazioni migliori rispetto con

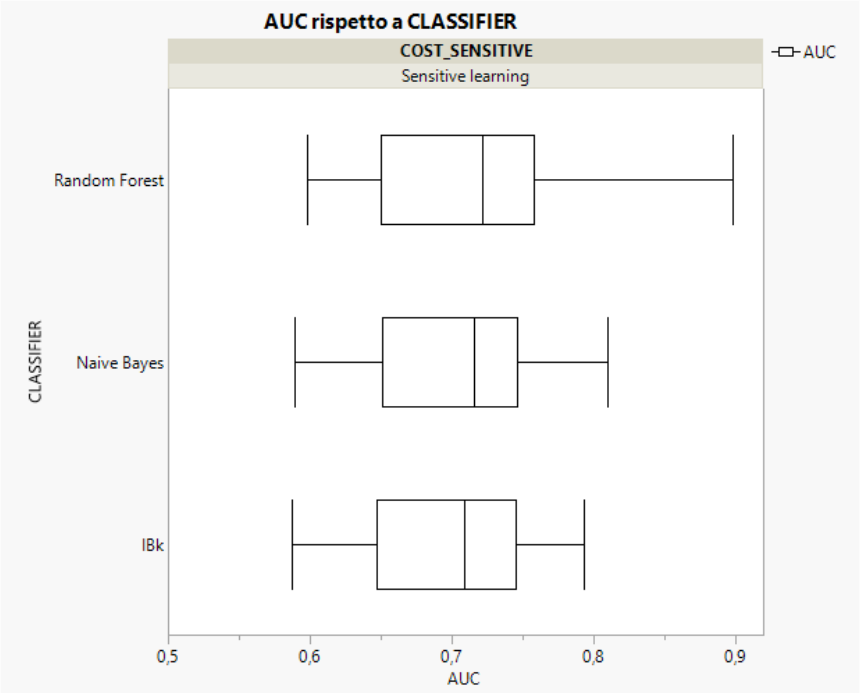
# Discussione – Syncope - AUC




- Random forest ha prestazioni migliori in entrambi i casi
- In caso di no feature selection, i tre classificatori arrivano a toccare valori più alti di AUC ma in generale l'accuratezza dell'AUC resta pressochè invariata proprio come ci si aspettava

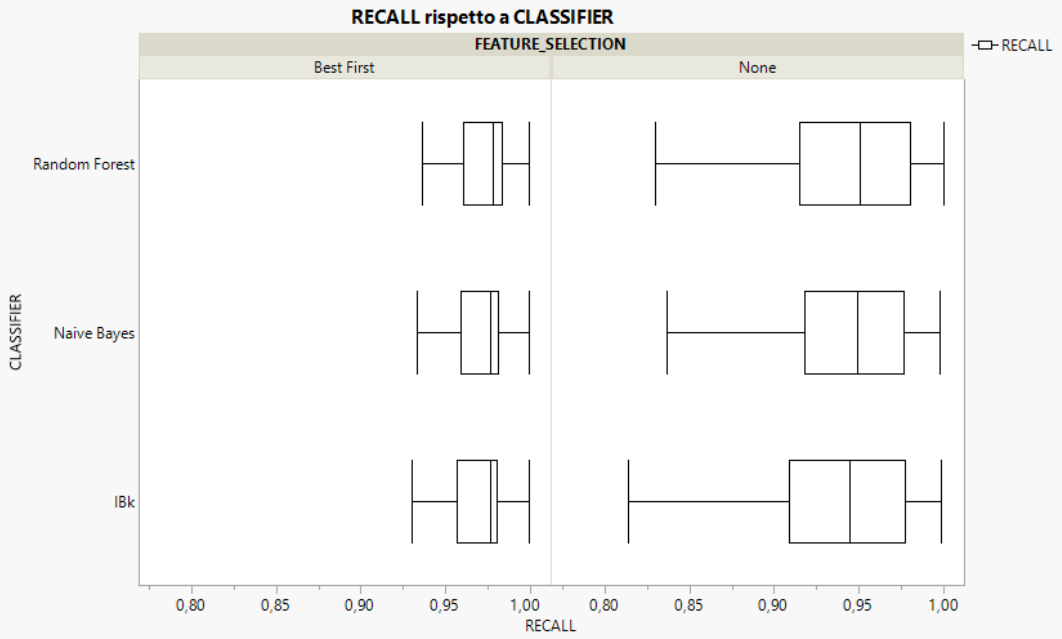



Random forest ha prestazioni migliori rispetto agli altri classificatori

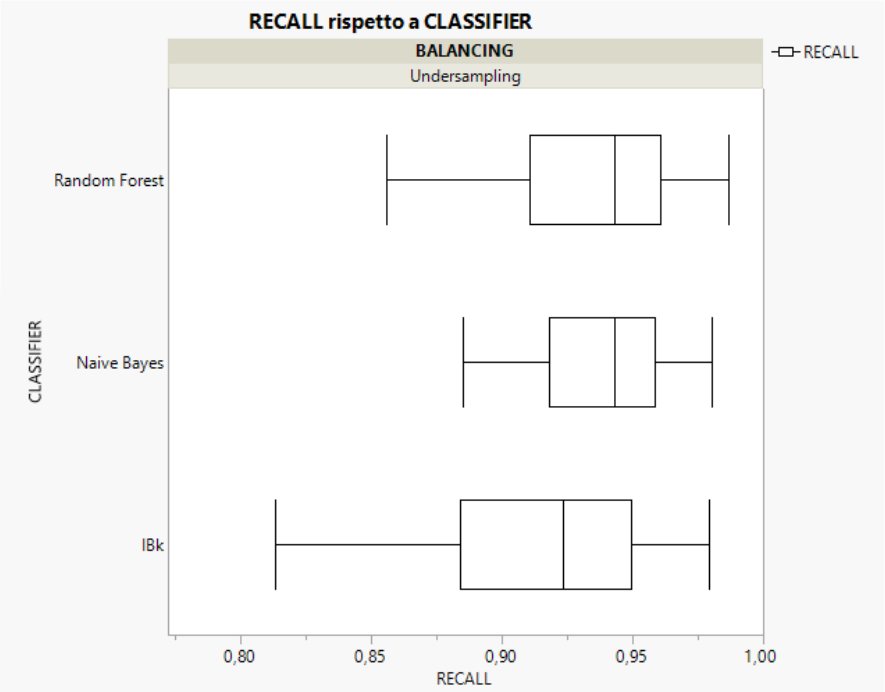



Anche con Sensitive learning Random Forest ha prestazioni migliori

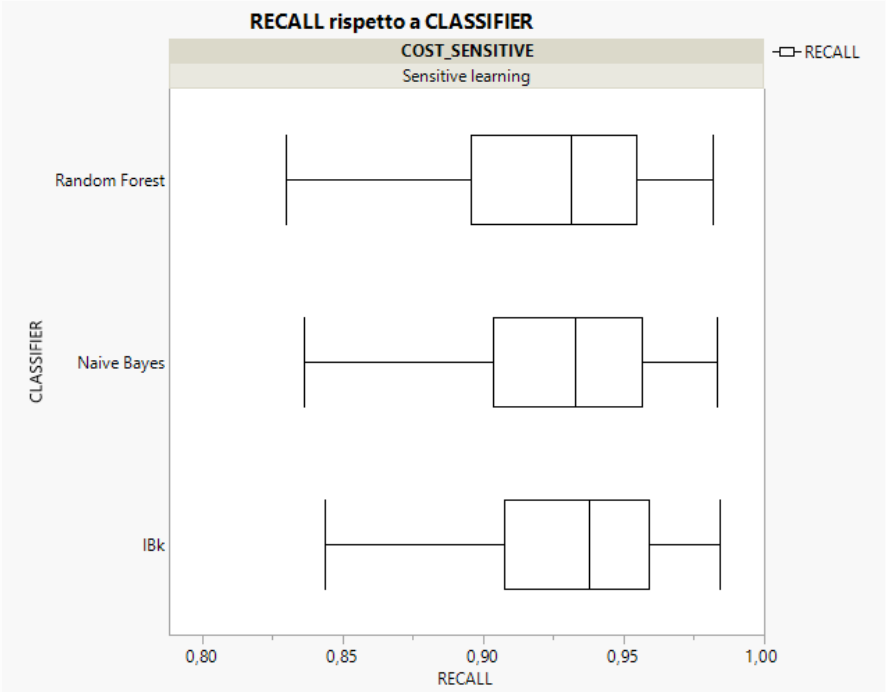
# Discussione – Syncope - Recall




- Con Best first, Random forest si comporta meglio
- Stessa cosa vale anche quando non viene appliccata la feature selection
- Notare come, senza applicazione di feature selection, tutti i classificatori arrivano a toccare valori più bassi di Recall rispetto a quando viene applicata Best First



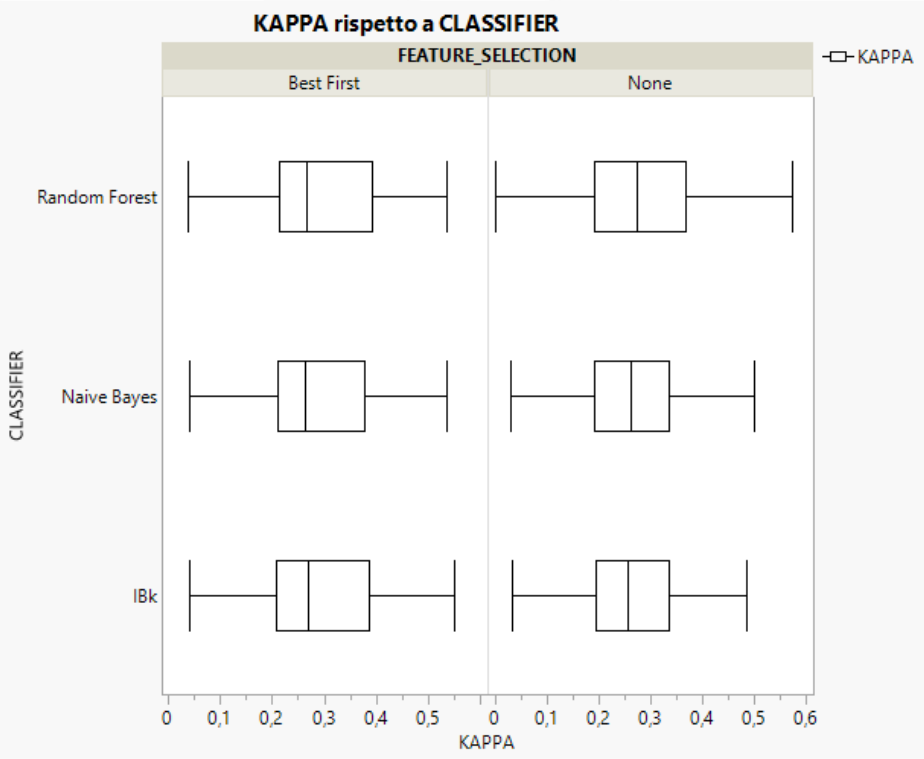

Naive Bayes ha prestazioni migliori



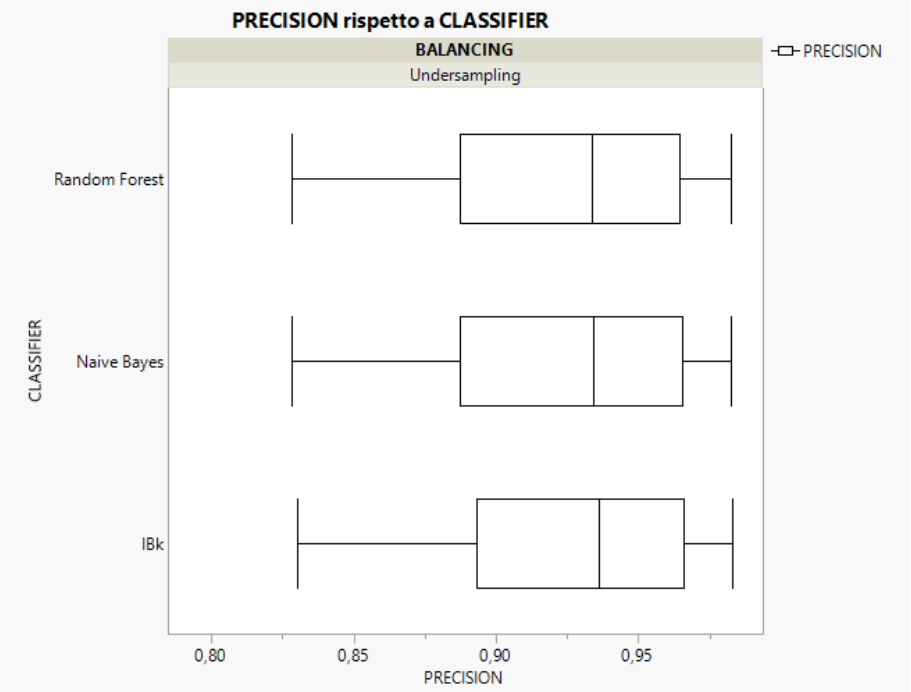

Con sensitive learning invece Ibk si comporta meglio degli altri

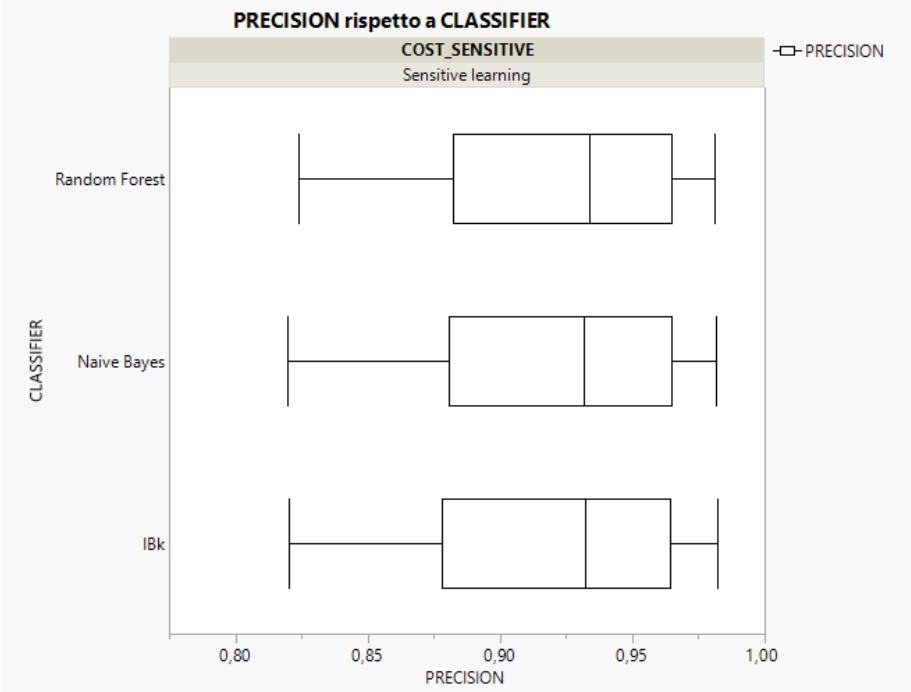
Con l'applicazione di cost sensitive le prestazioni migliorano in generale poichè si è abbassata la threshold e quindi vengono presi meno falsi negativi

# Discussione – Syncope - Precision




- Random forest si comporta meglio in entrambi i casi
- C’è anche un leggero miglioramento con l’applicazione di Best First



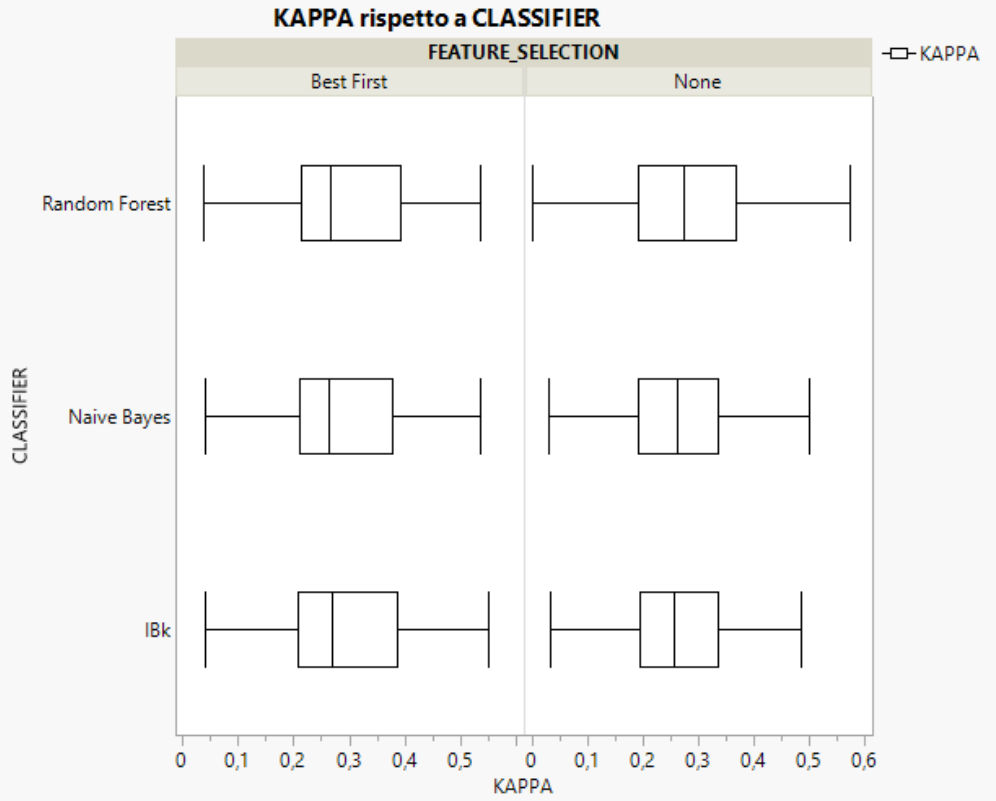




Ibk si comporta meglio rispetto agli altri

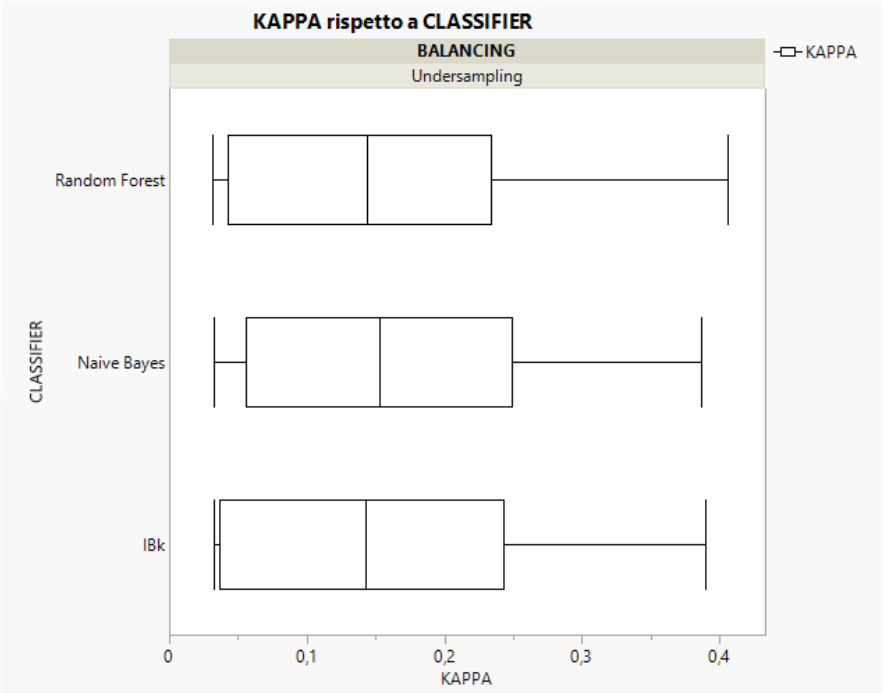
Random Forest si comporta meglio rispetto agli altri

Sia con Undesampling che con Sensitive Learning, la precision aumenta di molto rispetto al caso precedente. Questo perchè la variabile d'interesse è probabilmente presente in maniera superiore nel caso 'false', con undersapling quindi portiamo la cardinalità dei due casi ad uno stesso valore. Con il cost sensitive invece, abbassando la soglia, vengono presi meno falsi negativi (da notare anche che cost sensitive applica, al suo interno, una sorta di balancing)

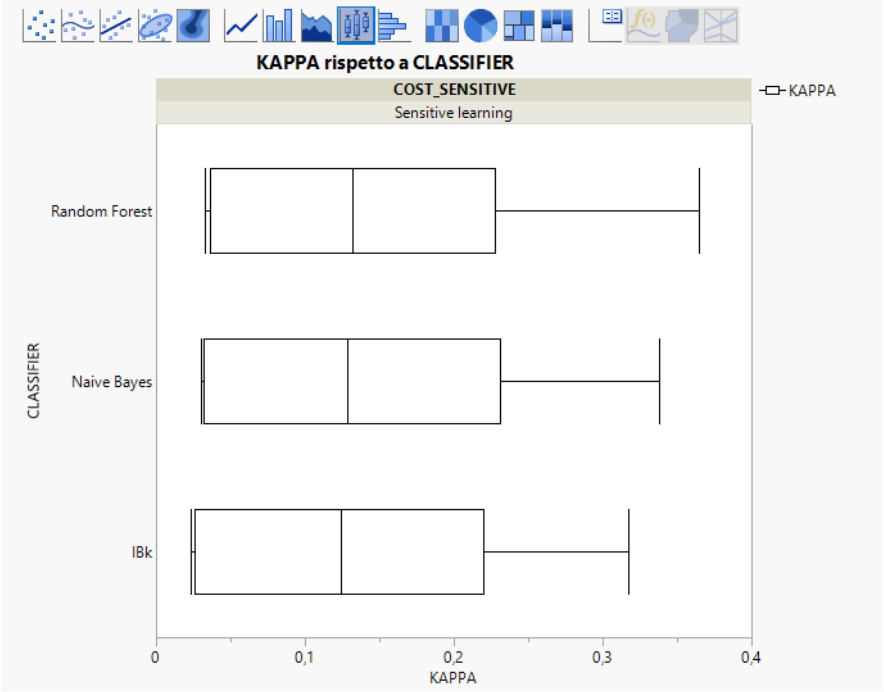
# Discussione – Syncope - Kappa




- Random Forest si comporta meglio in entrambi i casi
- L’applicazione del Best first aumenta leggermente le prestazioni di tutti e tre i classificatori




Naive Bayes ha prestazioni migliori in caso di Undersampling




In caso di Sensitive learning invece Random forest ha prestazioni leggermente migliori

# Conclusioni

- **BookKeeper:** i classificatori che hanno performance leggermente migliori sono Random Forest e Ibk
- Tra le tecniche utilizzate invece la feature selection è quella che apporta minori benefici ma questo era attendibile

- **Syncope:** il classificatore con migliori prestazioni è Random Forest
- L’undersampling e il cost sensitive learnin impattano in maniera migliore sulle prestazioni, questo è probabilmente dovuto alla bassa percentuale di classi buggy etichettate nel dataset