Machine Learning

For software engineering

Elisa Verza - 0311317

A.A. 2022-2023

AGENDA

01
STRUTTURA DEL PROGETTO

Mappa strutturale e tecnologie usate

02
DATA RETRIEVE

Processi e metodi per la raccolta dei dati

03
DATA ANALYSIS

Processo di analisi dei dati raccolti

04
DATA VISUALIZATION

Grafici ottenuti dall’output prodotto

01 Struttura Del Progetto

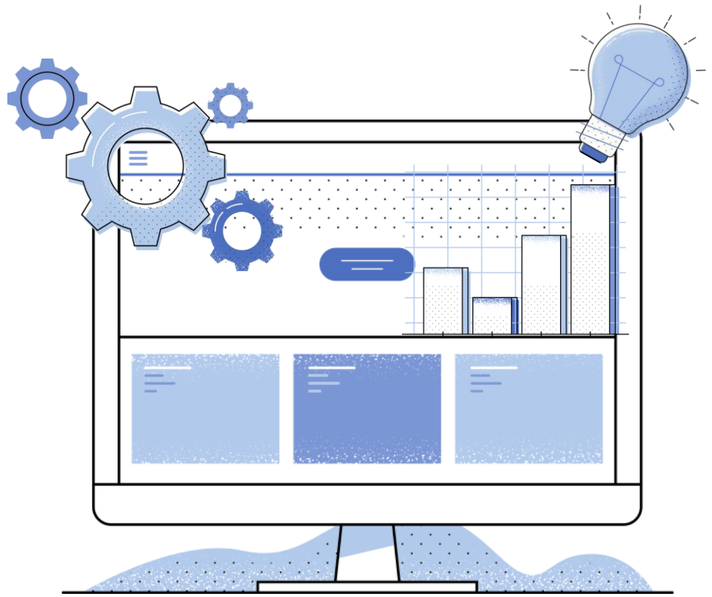
- Introduzione
- Schema del progetto
- Fasi del progetto

OBIETTIVI

Il progetto ha lo scopo di utilizzare tecniche di machine learning per la predizione, in progetti open source java, delle classi buggy

VANTAGGI

Utilizzando la predizione è possibile individuare i bug prima che si manifestino, inoltre è possibile concentrare l’attività di testing su classi più ad alto rischio.

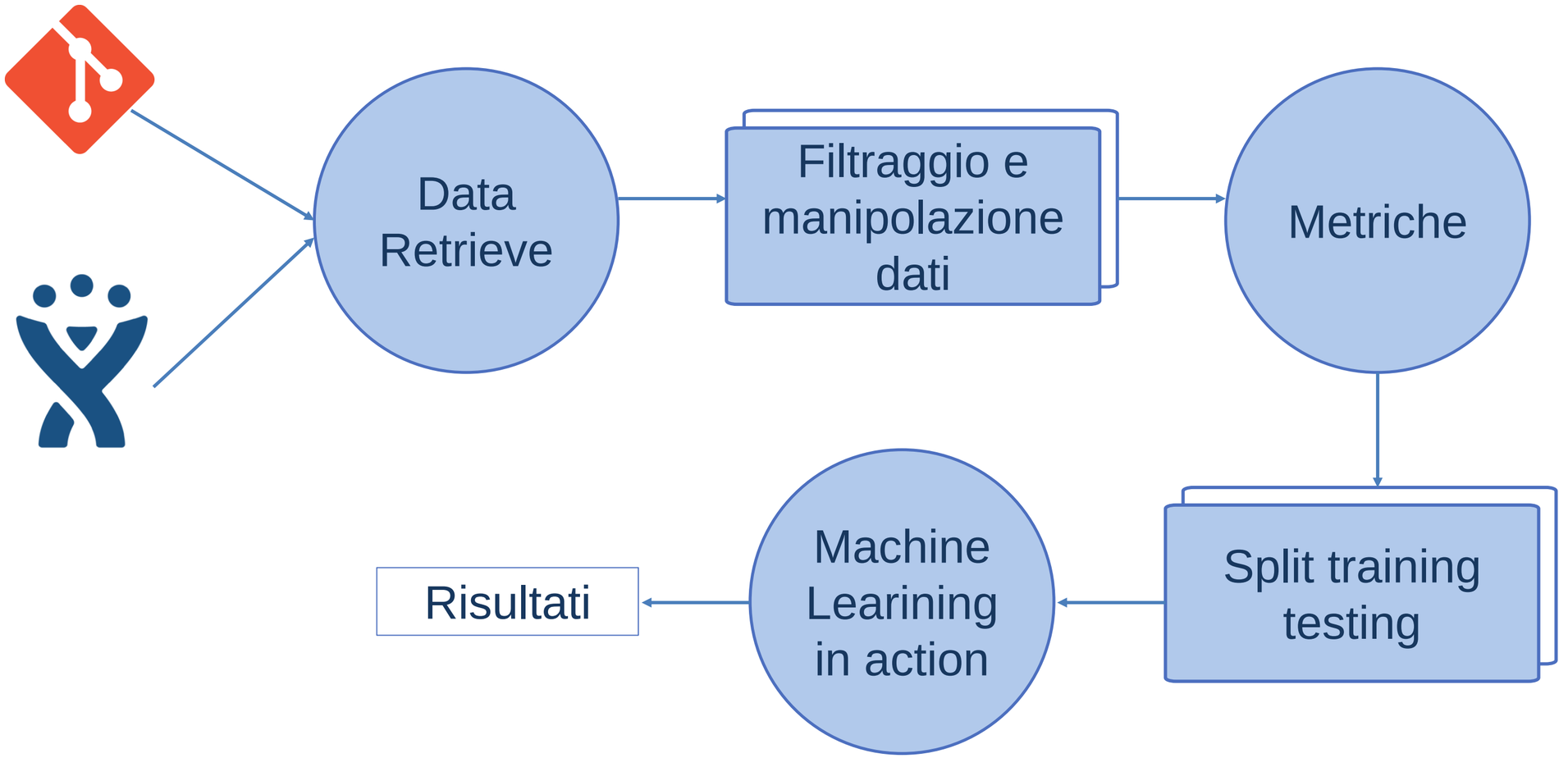


FOCUS

Nell’ambito del progetto sono stati analizzati due progetti open source di Apache: Bookkeeper e Syncope.

E’ stata comunque necessaria, come illustarto di seguito, l’analisi parziale di altri tre progetti: Zookeeper, Tajo e Avro

Struttura del progetto



Fasi del Progetto

1 Fonti dati

Incrociando i ticket di Jira rigurdanti i bug con stato resolved o closed e risoluzione fixed con i dati dei commit di git che presentavano nel messaggio l'ID del ticket risolto è stato possibile ottenere i dati grezzi.

2 Manipolazione dati

Si è proceduto ad: eliminare i dati non utilizzabili ai fini del progetto, eseguire il labeling dei dati assenti, assegnare i commit risolutivi dei ticket alle versioni dei progetti e calcolare quindi le affected version.

3 Calcolo delle metriche

Con i dati puliti è stato possibile calcolare le seguenti metriche per i file di ogni release: LOC touched, LOC added, Max LOC added, Avg LOC added, Churn, Max churn, Avg churn, Change set size, Max change set, Avg change set.

4 Training dei modelli

Il dataset è stato diviso in training e testing set con walkforward. Sono state valutete: precision, recall, AUC e kappa. Sono stati utilizzati tre modelli: NaiveBayes, RandomForest e IBK. Sono state adottate tecniche si samplig, feature selection e applicata matrice dei costi ai classificatori.

02 Data Retrieve

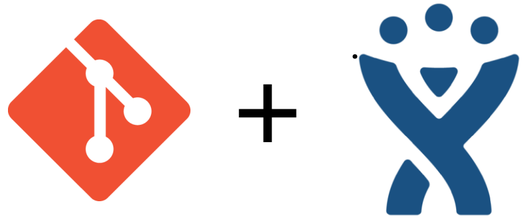
- Da dove vengono i dati?
- Quali dati scartare?
- Il labeling
- Calcolo delle metriche

Da dove vengono i dati?

Git

Tramite le API di git sono stati scaricati tutti i commit da cui sono state salvate le informazioni rilevanti: data del commit, sha e messaggio dal quale si ottine l'ID del ticket di Jira di cui il commit è risolutivo.

Per ogni versione vengono presi tutti i file java del progetto e tramite i commit della versione per ogni file sommate le righe di codice aggiunte e rimosse.



Viene fatto il join dei dati sull'ID del ticket Jira trovato nel commento dei commit di Git

Jira

Tramite le API di jira vengono presi tutti i ticket di interesse con la data di fix, la data di opening e la prima affected version.

Vengono scaricate anche tutte le versioni del progetto che risultano released con relativa data in modo da poterle ordinare temporalmente.

È così possibile assegnare il numero di versione alla data di fixed ed opening del ticket

Quali dati scartare?

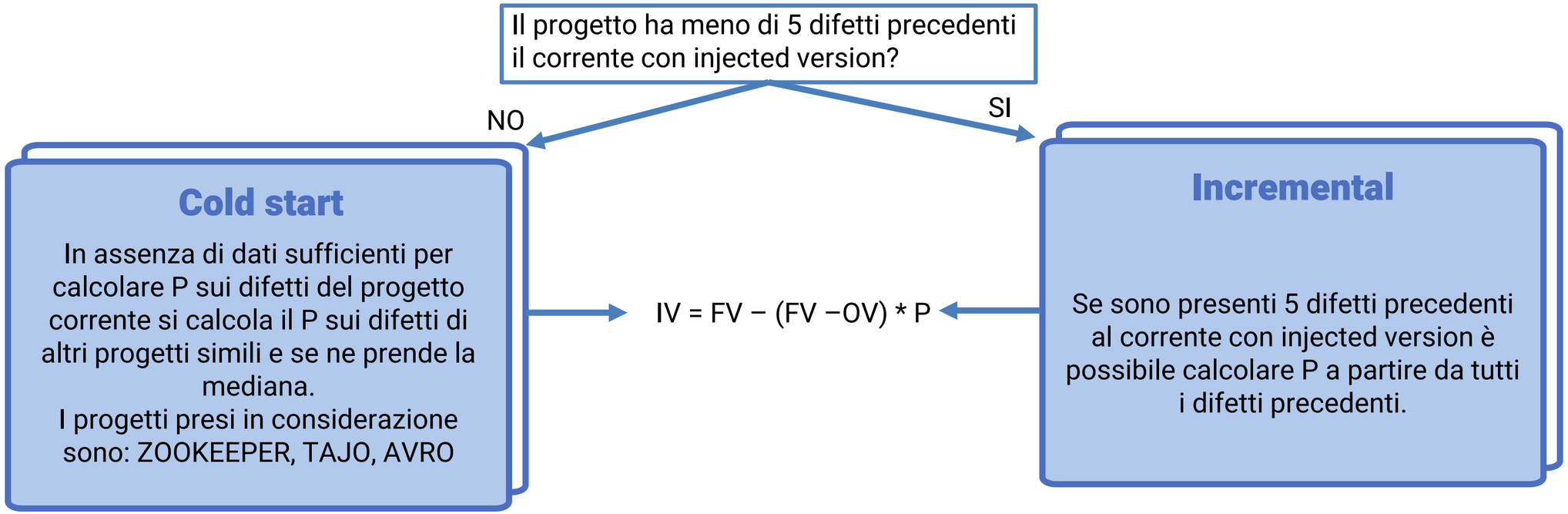
Non sono post release bug	IV = OV = FV	Sono bug nati e risolti prima del rilascio della nuova versione
Fixed date assente	FV = ??	Dai ticket jira sono stete prese le fix date e non le versioni, si tratta di un ticket malformato
Opening date assente	OV = ??	Ticket malformato
Injected version maggiore di opening	IV > OV ≥ FV	Ticket malformato

Cosa succede se non esistono nel ticket injected version ed affected version?

Labeling

Injected version non presente

In questo caso si procede alla stima del suo valore tramite una variabile chiamata proportion e calcolata nel seguente modo:
(FV – IV)/(FV – OV)
Il parametro può essere calcolato con dati diversi a seconda di quelli messi a disposizione:



Metriche

Le metriche considerate e calcolate sui dati precedentemente ottenuti sono inserite nella tabella sottostante. Sono state calcolate per ogni release e non su tutto il progetto, saranno le feature che serviranno ad addestrare i modelli alla predizione della bugginess.

METRICA	DESCRIZIONE	METRICA	DESCRIZIONE
LOC touched	Somme linee di codice aggiunte ed eliminate	**Avg churn**	Numero medio di churn
LOC added	Somma linee di codice aggiunte	**Max churn**	Numero massimo di churn
Max LOC added	Numero massimo di linee di codice aggiunte	**Change set size**	Numero di file committati insieme
Avg LOC added	Numero medio di linee di codice aggiunte	**Max change set**	Numero massimo di file committati insieme
Churn	Differenza tra righe aggiunte o eliminate	**Avg change set**	Numero medio di file committati insieme

03
DATA ANALYSIS

Preparazione alla predizione

Pulizia training set in corso...

Modelli e risultati

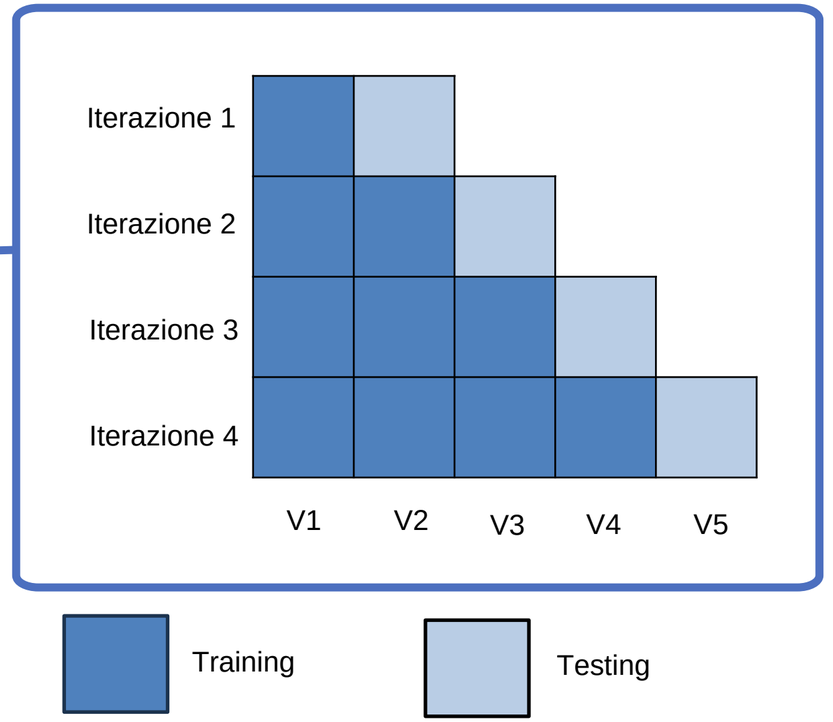
Preparazione alla predizione

Limitare lo snoring

Le classi snoring hanno dei bug, che si manifesteranno dopo alcune release, quindi le più recenti con più probabilità avranno un maggior numero di classi dormienti. Per questo motivo si scarta la metà dei dati più recente

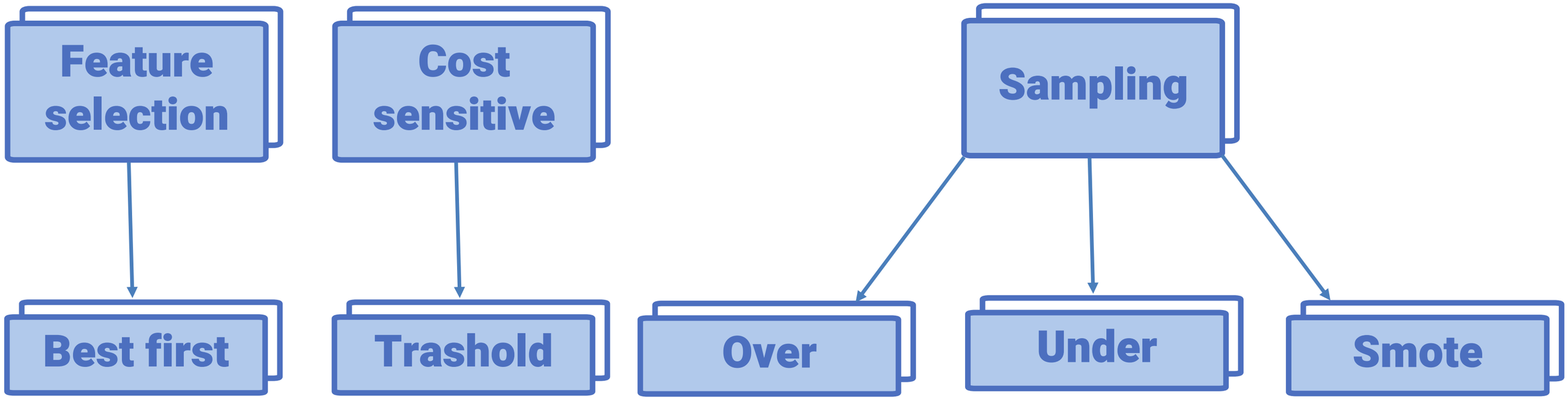
Walk forward

Come dividere il dataset in training e testing?
Il walk forward divide il dataset in porzioni, in questo caso versioni in ordine temporale.
Partendo dalla prima versione per il training e la successiva come testing, le iterazioni successive aggiungono una versione al training e fanno scalare il testing di una



Pulizia training set in corso...

I modelli sono stati addestrati con dataset raw, quindi senza effettuare alcun tipo di preprocessing o utilizzando una combinazione di tecniche di sampling, feature selection e applicando la matrice dei costi al classificatore.



Modelli e risultati

I modelli utilizzati sono stati:

- **NaiveBayes**
- **RandomForest**
- **IBK**

Sono state calcolate le seguenti metriche:

Recall	TP/(TP + FN)
Precision	TP/(TP + FP)
AUC	Area sotto la ROC curve
Kappa	Quante volte si ha risultato migliore del classificatore random

04 DATA VISUAL IZATION

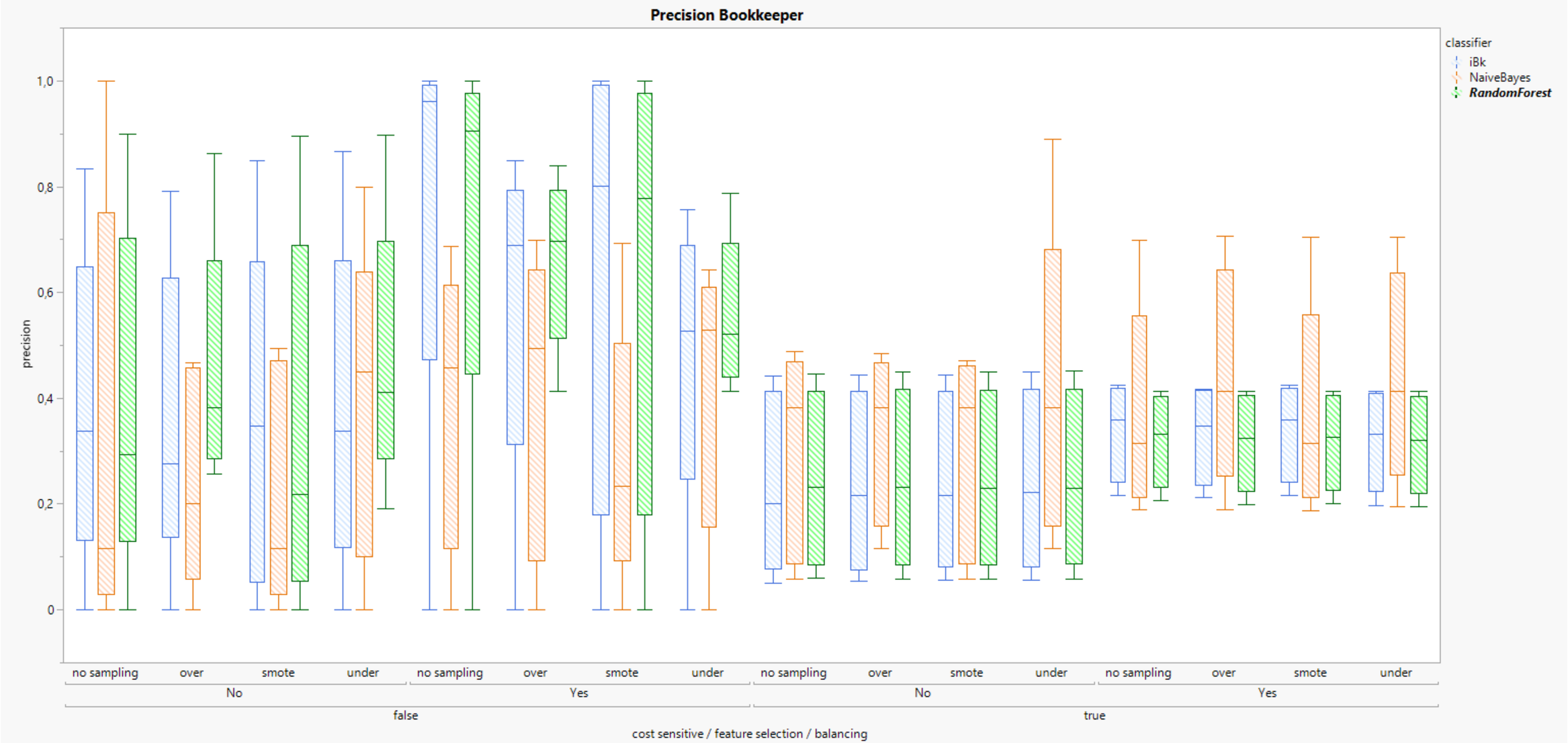
Precision, recall, AUC, Kappa - Bookkpper

Analisi dei grafici

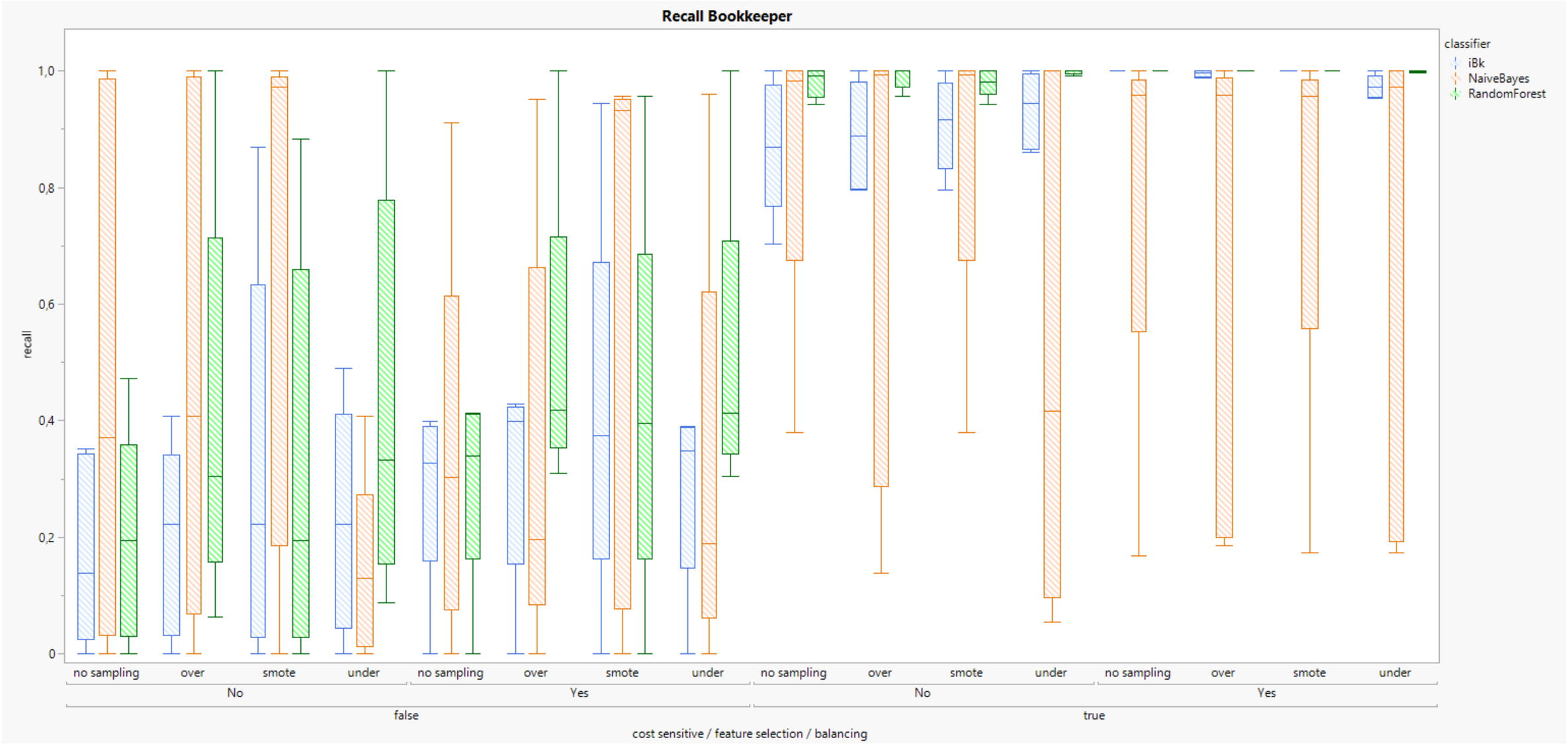
Precision, recall, AUC, Kappa - Syncope

Analisi dei grafici

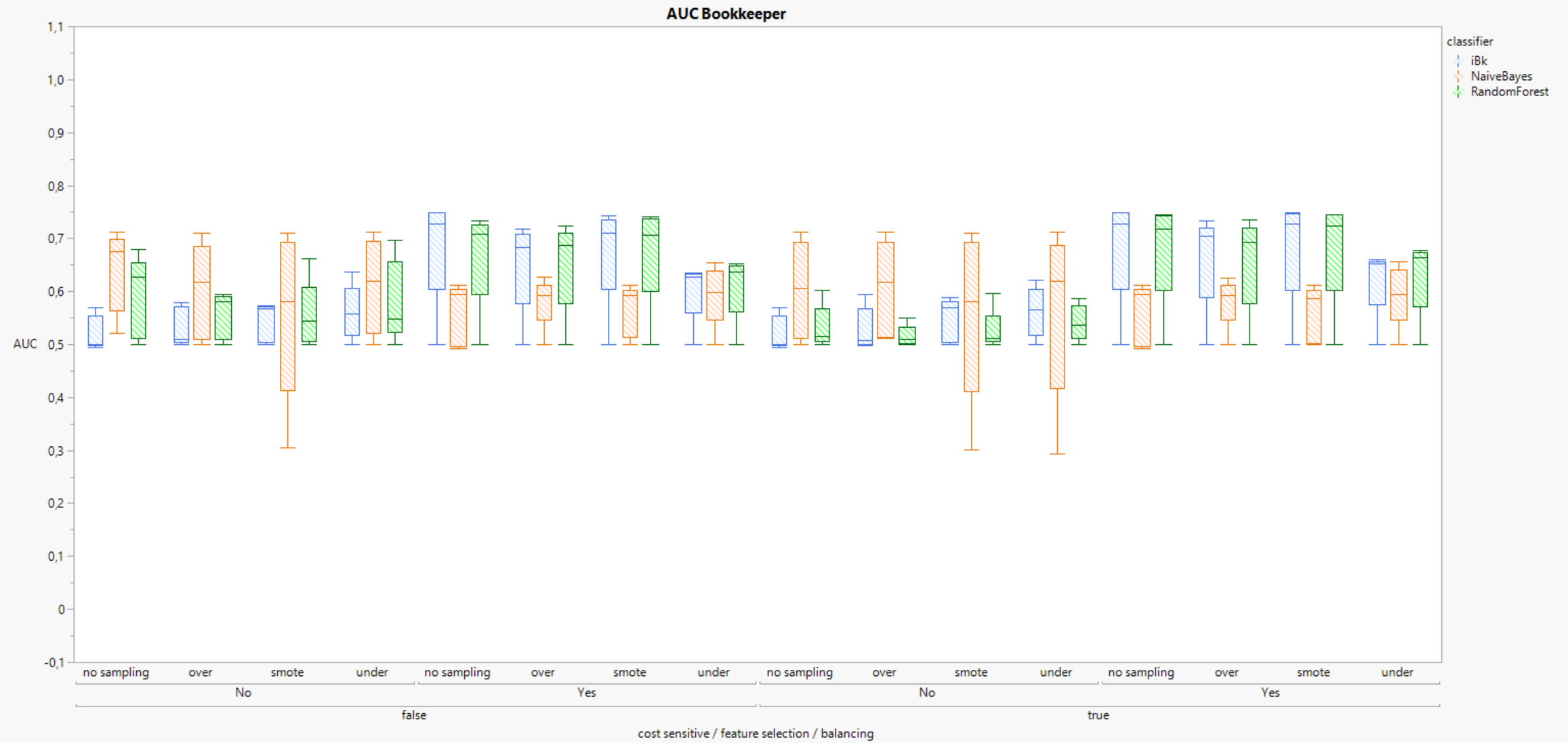
Precision - Bookkeeper



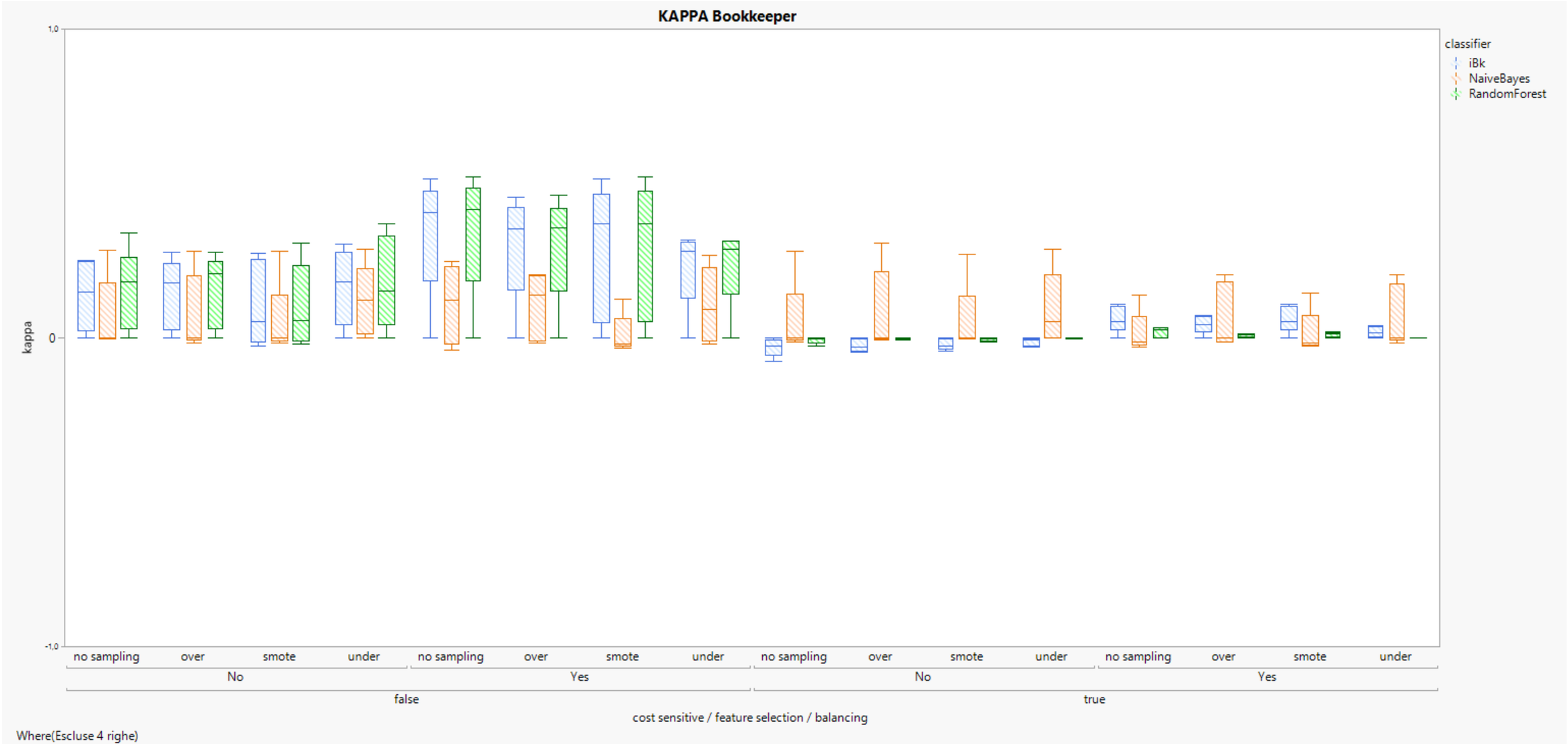
Recall - Bookkeeper



AUC - Bookkeeper



Kappa - Bookkeeper



Where(Escluse 4 righe)

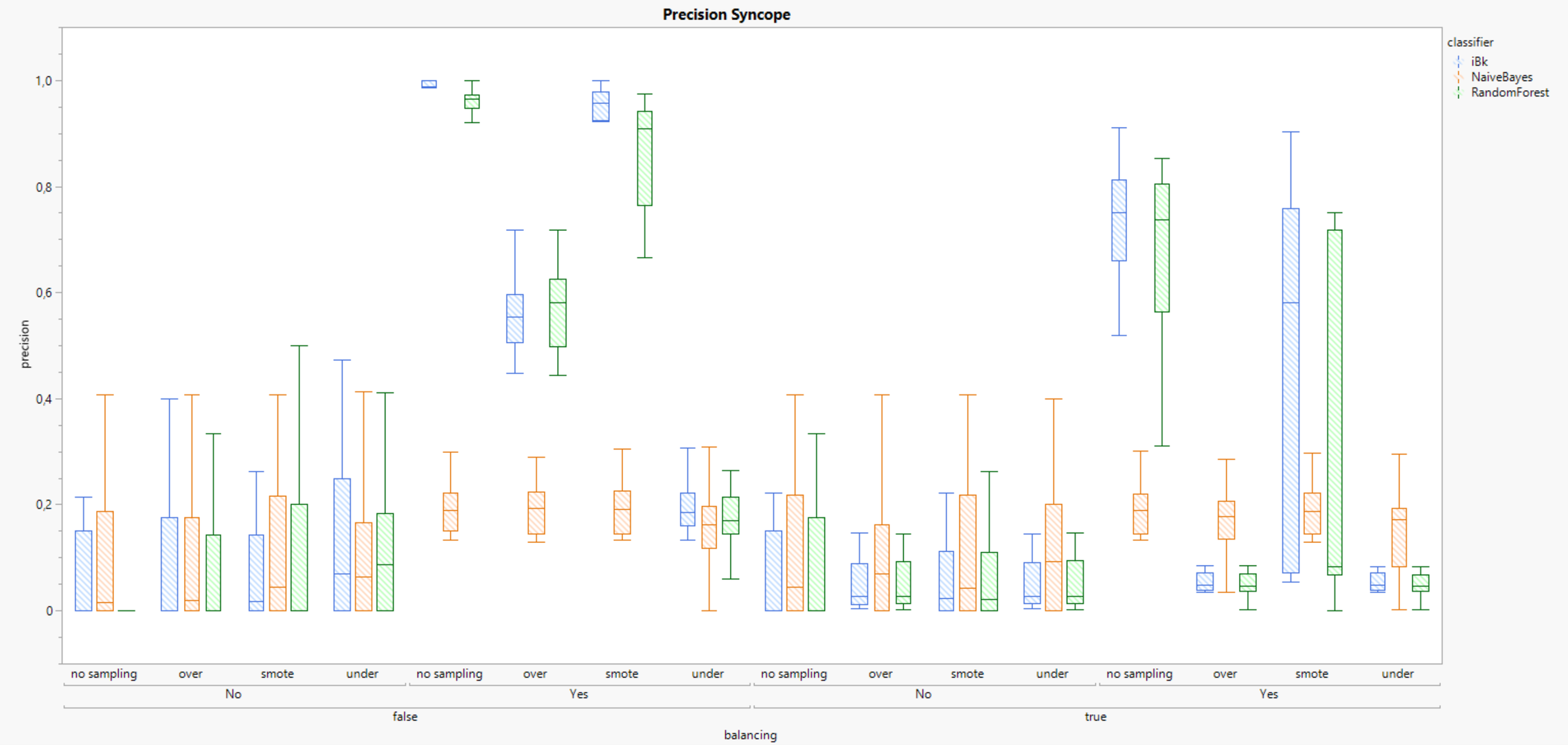
Analisi dei grafici

La precision con mediana più alta si ottiene senza balancing, facendo feature selection e con un modello IBK senza cost sensitive, però presenta anche un'alta dispersione, sia totale che dei quantili, con una precision con mediana 20% inferiore si trova il classificatore random forest, con oversampling, senza feature selection e non cost sensitive. Nonostante il valore inferiore presenta una dispersione nettamente inferiore.

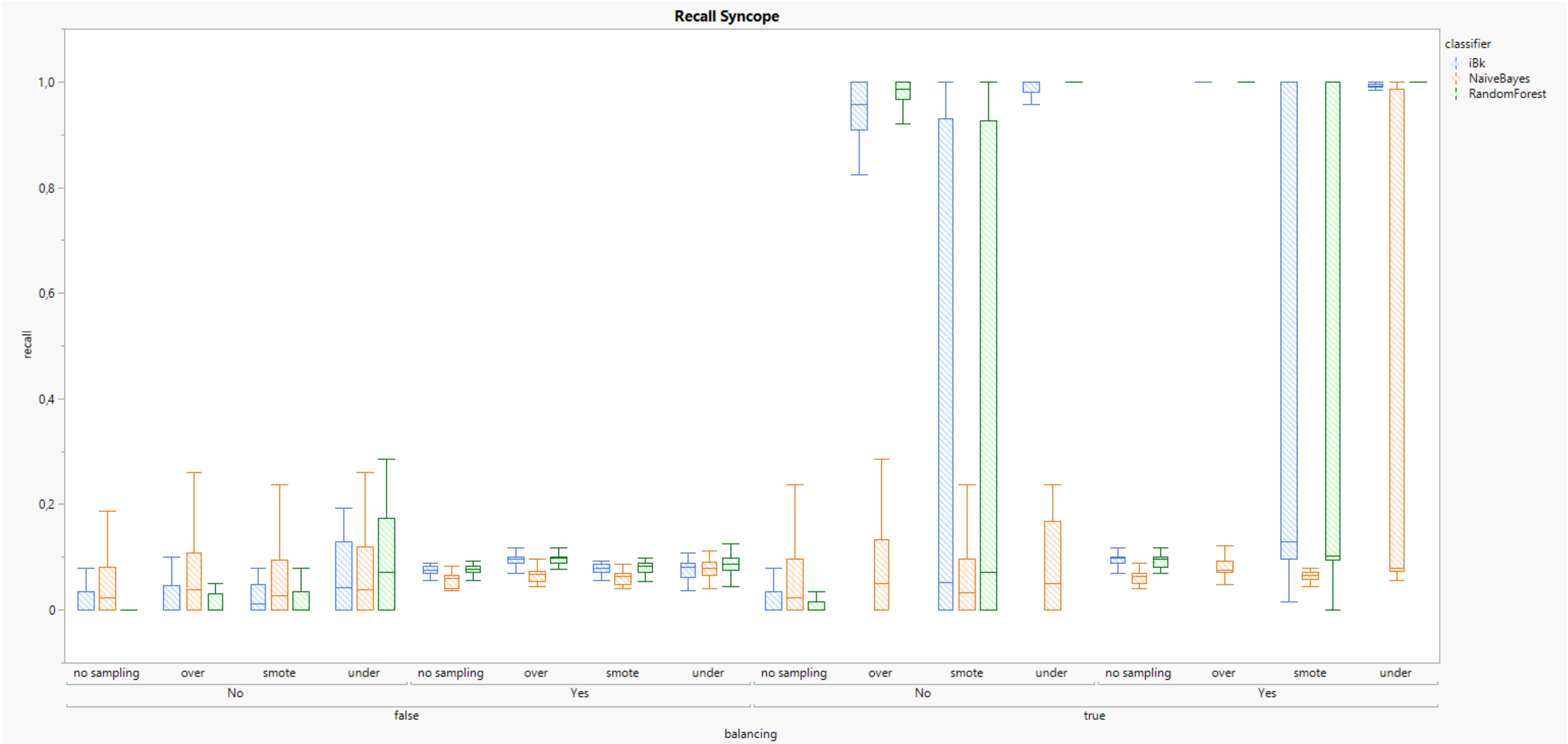
Per quanto riguarda la recall i migliori risultati si otterrebbero con il classificatore Random Forest, con feature selection e cost sensitive è indipendente la scelta del balancing, però le corrispondenti precision hanno mediana bassa, quindi si continua a preferire la seconda soluzione individuata nella precision, che ha mediana ancora abbastanza alta e una dispersione contenuta.

Sia per AUC che KAPPA i migliori risultano essere i classificatori con Random Forest, feature selection, non cost sensitive e senza o con sampling over o smote. Tra questi ricade anche il caso preso in considerazione fin'ora

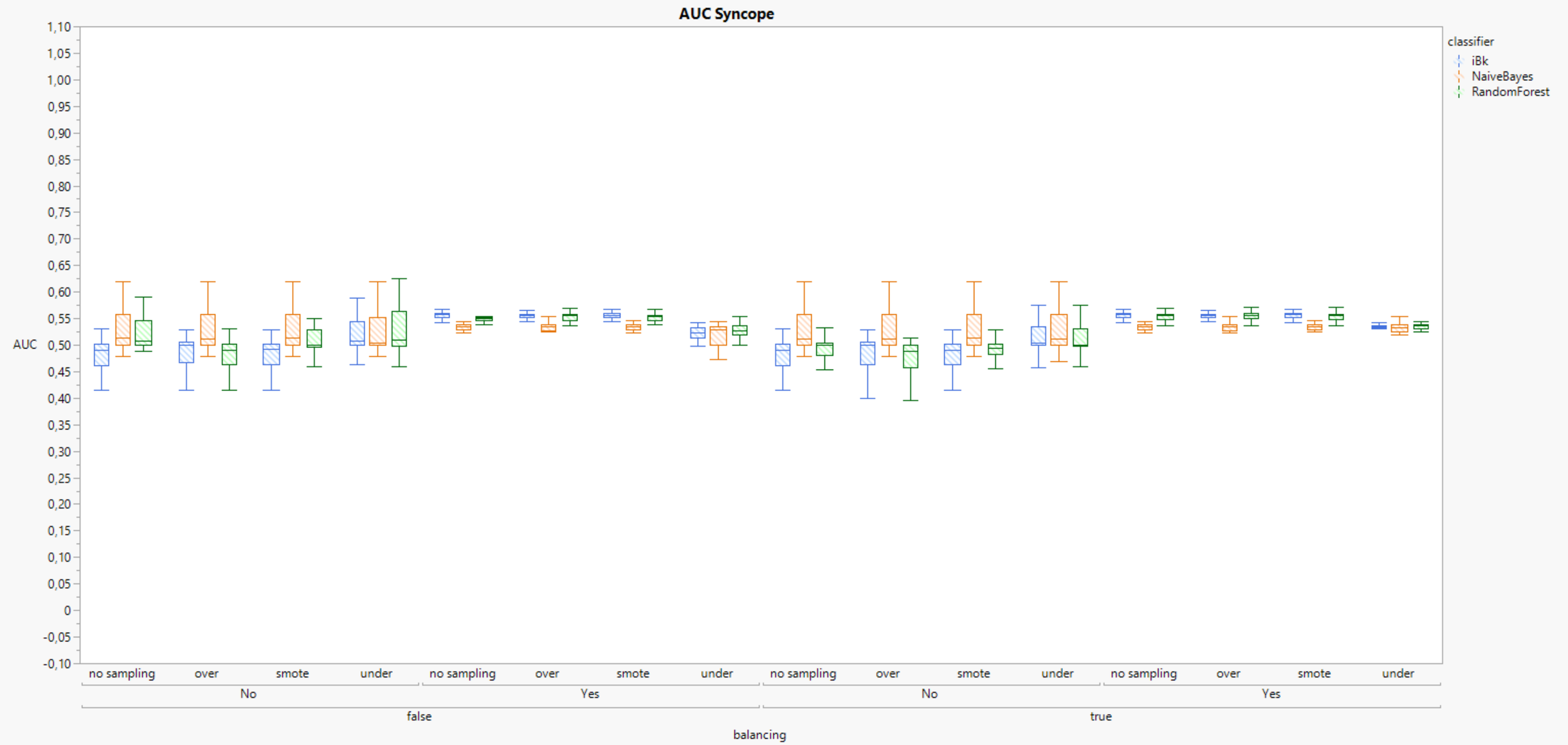
Precision - Syncope



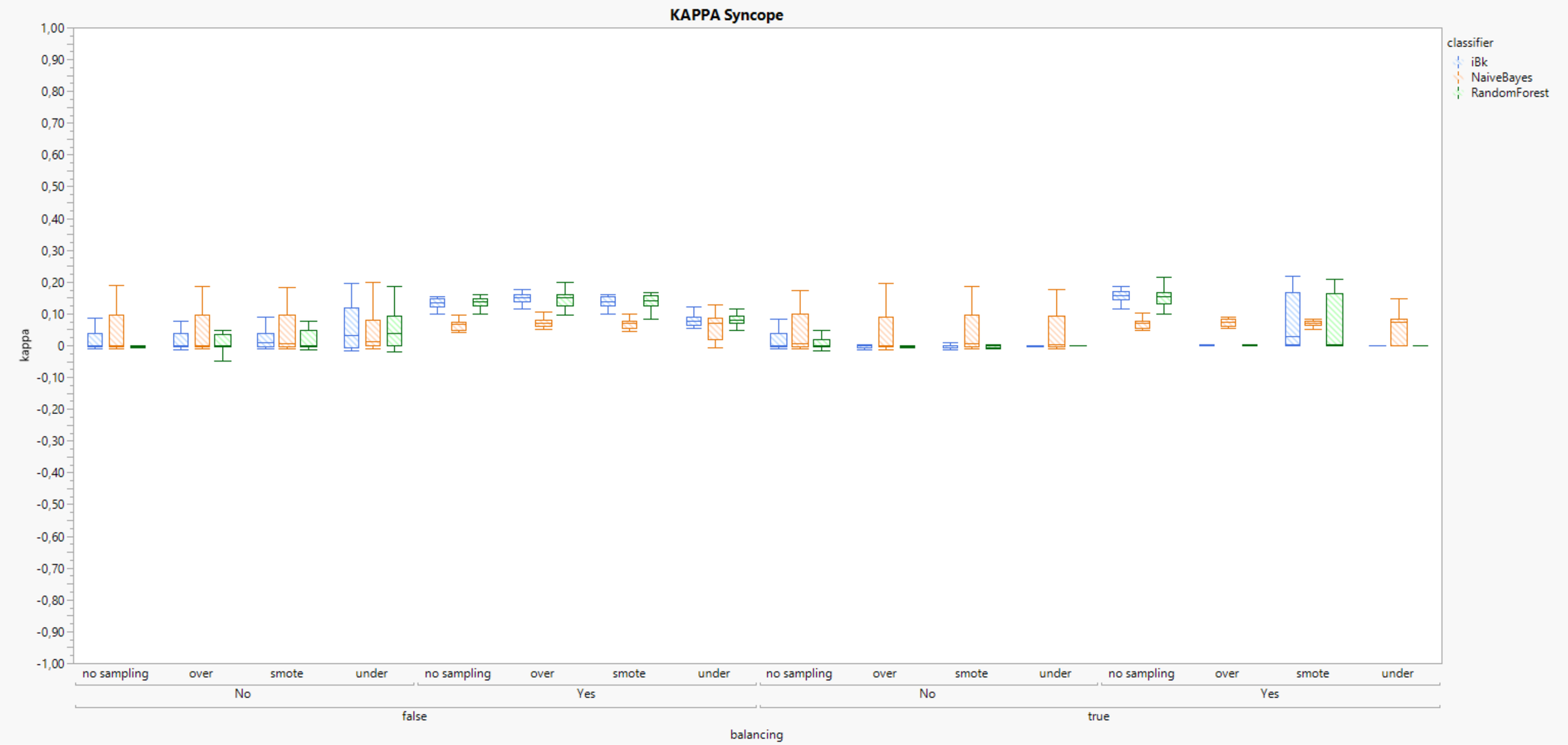
Recall - Syncope



AUC - Syncope


AUC Syncope
AUC
1,10
1,05
1,00
0,95
0,90
0,85
0,80
0,75
0,70
0,65
0,60
0,55
0,50
0,45
0,40
0,35
0,30
0,25
0,20
0,15
0,10
0,05
0
-0,05
-0,10
classifier
iBk
NaiveBayes
RandomForest
no sampling
over
smote
under
No
Yes
false
true
balancing

Kappa - Syncope



Analisi dei grafici

Per Syncope l'analisi dei risultati è partita dall'osservare AUC e KAPPA perché precision e recall presentano valori discordanti. Inoltre i risultati rispecchiano lo sbilanciamento trovato nei dati che presentano percentuali molto basse di classi buggy rendendo difficili le prime iterazioni del walk forward.

Per quanto riguarda AUC è costante in tutte le prove effettuate ed oscilla intorno allo 0,5. Discorso analogo per KAPPA che sta tra lo zero e 0,15 per quanto riguarda la mediana. In entrambi i casi la dispersione è piccola e il valore della mediana più alto risulta essere il classificatore Random forest, senza sampling, ma con fueature selection e cost sensitive.

Precision e recall nella zona del grafico dove non avviene feature selection e non è cost sensitive indiependentemente dal balancing e dal modello risultano essere molto bassi. Effettuando il feature selection, con smote e oversampling o senza, Naive Bayes presenta la minore disparità tra precision e recall che però risultano basse, stesso comportamento hanno tutti e tre i classificatori nel caso di undersampling. Con la cost sensitivity l'unico cambiamento è nel caso di smote balancing un incremento notevole della dispersione della recall, sempre con valori bassi di precision.

In conclusione, vista la scarsa presenza di esempi buggy, si ottengono al più classificatori con bassa recall e alta precision, che quindi assegnano la buggyness a poche classi con una precisione alta.

Link Utili:

sonarcloud

GitHub

THANK YOU