Machine Learning in Software Engineering

MACCARI GIANLUCA 0339210

Agenda

- Contesto e obiettivi
- Progettazione
- Variabili
- Risultati
- Scelte di progettazione
- Link Sonar Cloud e GitHub

Contesto

Nel processo di sviluppo di un prodotto software è fondamentale effettuare attività di testing volte all'individuazione e risoluzione di errori.

Le attività di testing ha però un costo elevato e spesso nello sviluppo di un progetto software non si possono impiegare troppe risorse nel testing.

Esiste quindi uno strumento che può velocizzare l'individuazione dei bug per ridurre il costo del testing, e tale strumento è il **Machine Learning**.

Per utilizzare questo strumento occorre misurare il codice per ottenere dei dati da dare in pasto ai classificatori. Inoltre, è necessario individuare le classi che presentano bug ed etichettarle correttamente, al fine di addestrare correttamente i classificatori.

Obiettivi

La domanda che viene posta in questo studio è quali tecniche tra Feature Selection, Sampling e Cost Sensitive Learning aumenta l'accuratezza dei classificatori?

Quindi occorre:

- Costruire il **dataset** di partenza
- Addestrare i classificatori
- Analizzare i risultati

Progettazione: introduzione

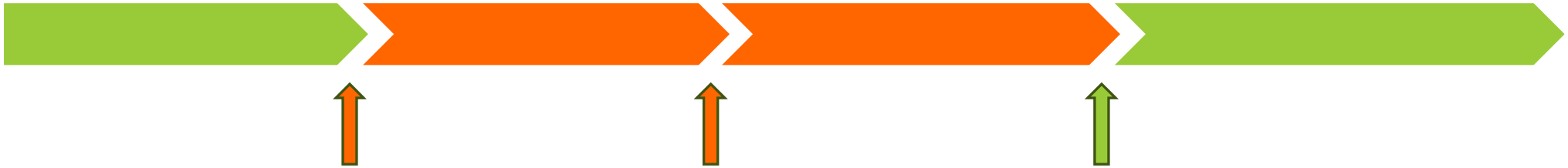
Per questo studio sono stati analizzati i progetti BOOKKEEPER e OPENJPA del gruppo Apache. Altri dati sono stati estratti dai seguenti progetti: AVRO, STORM, SYNCOPE, ZOOKEEPER e TAJO.

Per entrambi i progetti sono state utilizzate le repository GitHub da cui sono stati ottenuti il codice e le sue versioni.

Dalla piattaforma Jira sono stati ottenuti i ticket.

Infine i dati raccolti sono stati manipolati con la piattaforma Weka, dai cui sono stati tratti i risultati finali.

Progettazione: versions



Ogni bug ha il proprio ciclo di vita. Dal disegno si può intuire che il verde rappresenta le fasi temporali del progetto prive del bug e il rosso le fasi affette dal bug.
Nello scorrimento temporale si possono identificare 3 momenti:

- Injected version, versione in cui è stato inserito il bug
- Opening version, versione in cui è stata osservata la failure relativa al bug. Apertura del ticket
- Fixed version, versione in cui il bug è stato corretto

Progettazione: Jira

OV e FV possono essere **sempre** ottenuti dai dati forniti da Jira.

Da Jira sono stati presi tutti i ticket relativi a bug con stato **closed** o **resolved** e resolution **fixed**.

Ognuno di questi ticket riporta data di creazione del ticket (assunta come OV) e data di chiusura del ticket (assunta come FV).

Dai ticket si può anche ottenere la IV, però molto spesso tale informazione non è presente.

Come fare per ottenere la IV?

Progettazione: proportion

Idea: è possibile supporre che il tempo tra l'stante in cui il bug è stato trovato e l'istante in cui viene risolto, sia proporzionale al tempo tra istante in cui il bug è stato inserito e istante in cui viene risolto.

Questa tecnica è chiamata **proportion**, per cui è definita la seguente costante di proporzione:

$P = (FV – IV) / (FV – OV)$

Il valore di P può essere calcolato dai ticket che presentano IV e successivamente può essere applicato per calcolare la IV dei ticket che non presentano tale informazione:

$IV = FV - P (FV – OV)$

Progettazione: proportion (2)

In questo studio sono state applicate due metodologie per calcolare il valore del proportion:

- **Cold Start**, si calcola P come la mediana tra il valore P degli altri progetti del gruppo apache.
- **Increment**, fissato un istante di tempo, si calcola P con tutte le informazioni del progetto precedenti all'istante considerato.

Cold Start è stato applicato in caso il progetto in esame non avesse a disposizione almeno 5 ticket con IV. In particolare per tutto BOOKKEEPER e per le iterazioni iniziali di OPENJPA come vedremo avanti.

Progettazione: misurazione

Per applicare il Machine Learning è necessario individuare delle **metriche** misurabili con cui i modelli possano apprendere.

Le metriche considerate sono le seguenti:

- Complessità ciclomatica
- LOC, numero di linee di codice
- LOC no comment, come LOC ma escluse le linee di codice
- Linee di codice senza commenti
- NR, numero di revision della classe
- N_Auth, numero di
- Age, tempo di vita della classe in settimane

Progettazione: misurazione (2)

- Churn, somma delle linee aggiunte ed eliminate dai commit
- AVG Churn, valore medio del churn sul numero di commit che hanno modificato la classe
- MAX Churn, valore massimo di churn
- LOC_added, somma di tutte le linee aggiunte dai commit
- MAX_LOC_added, massimo numero di linee aggiunte da un commit

Progettazione: evaluation dei classificatori

Come valutare i classificatori?

Release

Run	1	2	3	4	5
1					
2					
3					
4					
5					

Walk-Forward è la tecnica applicata per valutare l'accuratezza dei classificatori. Si suddivide il dataset in parti (in questo studio una parte corrisponde ad una Release), la parte temporalmente più recente è usata come **test set** (verde in figura) e tutte le parti meno recenti sono utilizzate come **train set** (rosso in figura). L'accuratezza finale è calcolata come media tra i valori ottenuti nelle run.

Progettazione: evaluation dei classificatori (2)

Per ridurre gli effetti dello **snoring**, sono state scartate la metà delle release più recenti dei progetti e ad ogni run viene scartata la release precedente al test set.

Nota importante: train set e test set seguendo regole differenti.

Train set: deve essere il più fedele alla realtà di utilizzo; non può essere definito con dati o informazioni disponibili nel futuro; nonostante gli scarti effettuati, è affetto da snoring; l'etichetta buggy è assegnata in base ai dati disponibili nel momento corrente.

Test set: deve contenere i dati reali e corretti; costituisce l'oracolo; non deve essere affetto da snoring; l'etichetta buggy è assegnata in base a tutti i dati del progetto.

Variabili

I classificatori analizzati sono: IBk, Naive Bayes e Random Forest.

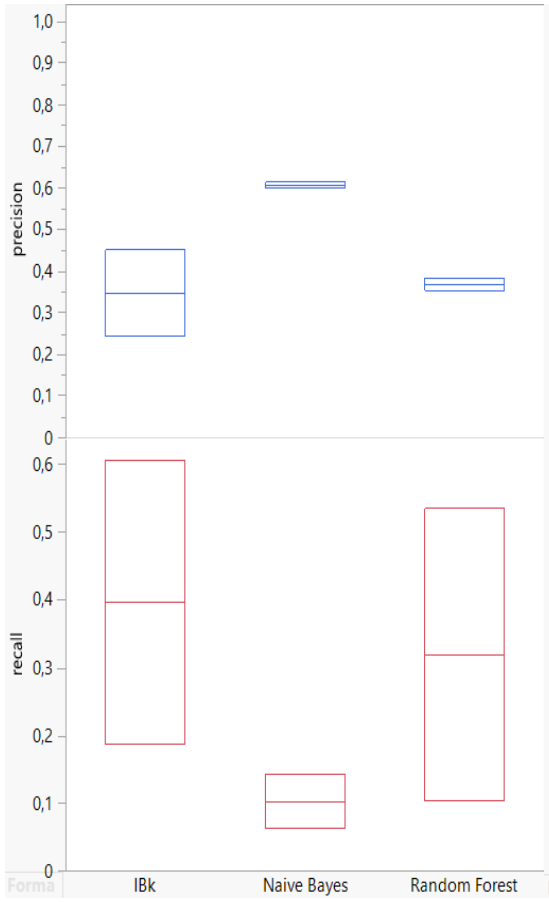
Le tecniche analizzate sono: Feature Selection, Under Sampling, Over Sampling e Cost Sensitive Learning.

Le tecniche sono anche state combinate: Feature Selection con Under Sampling e Feature Selection con Cost Sensitive Learning.

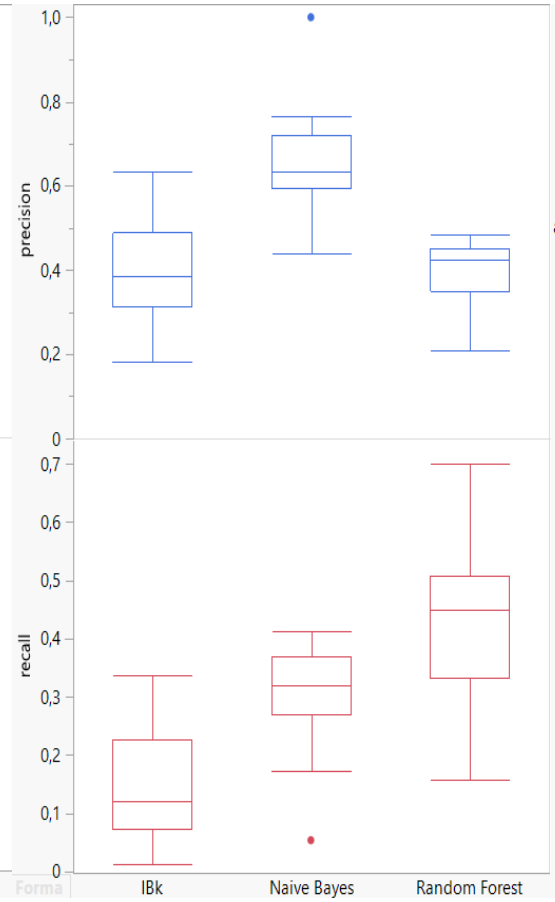
Le metriche utilizzate per le analisi dei risultati sono: precison, recall, auc e kappa.

Risultati: BOOKKEEPER e OPEN

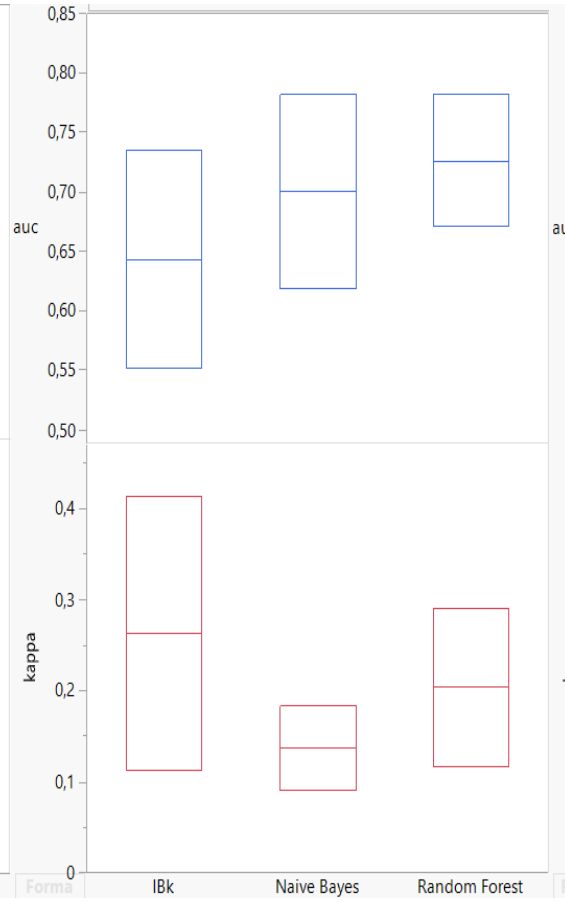
BOOKKEEPER



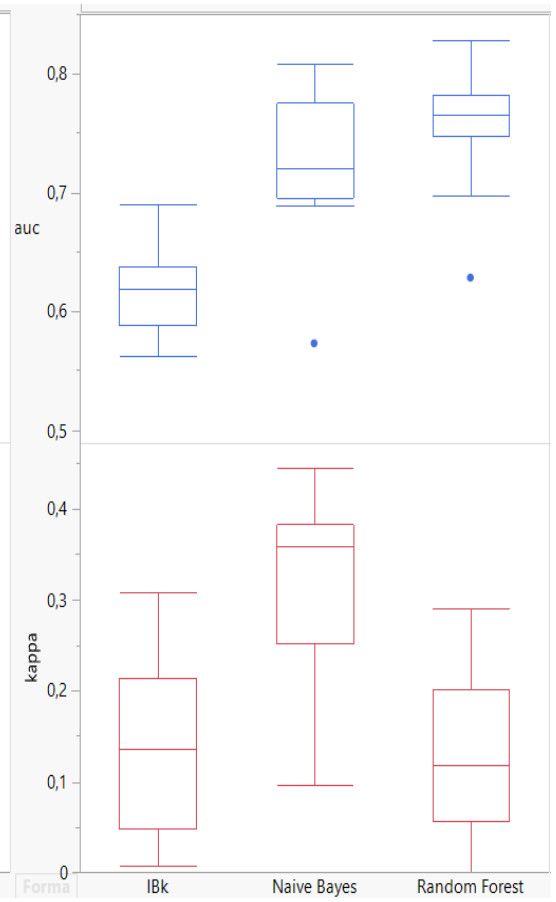
OPENJPA



BOOKKEEPER

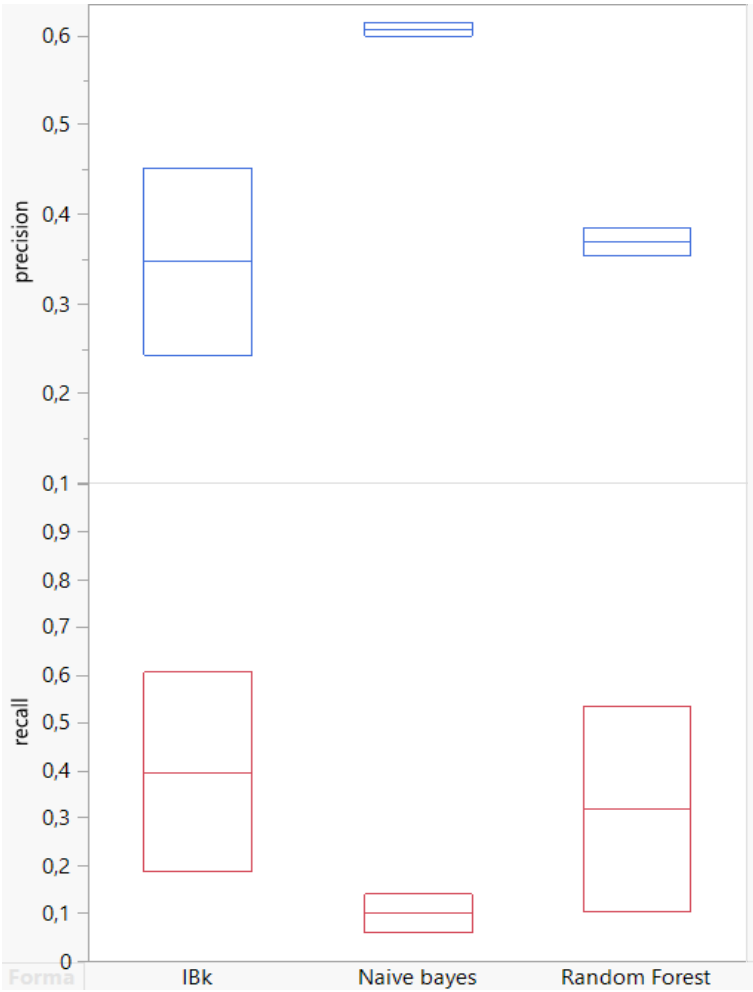


OPENJPA

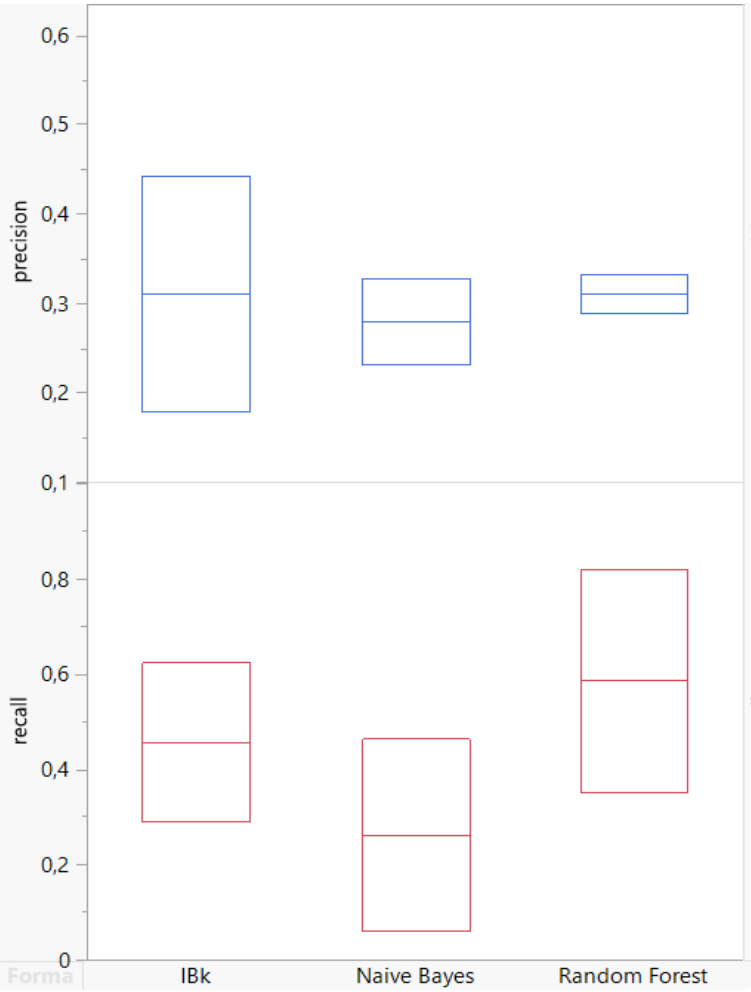


Risultati: BOOKKEEPER

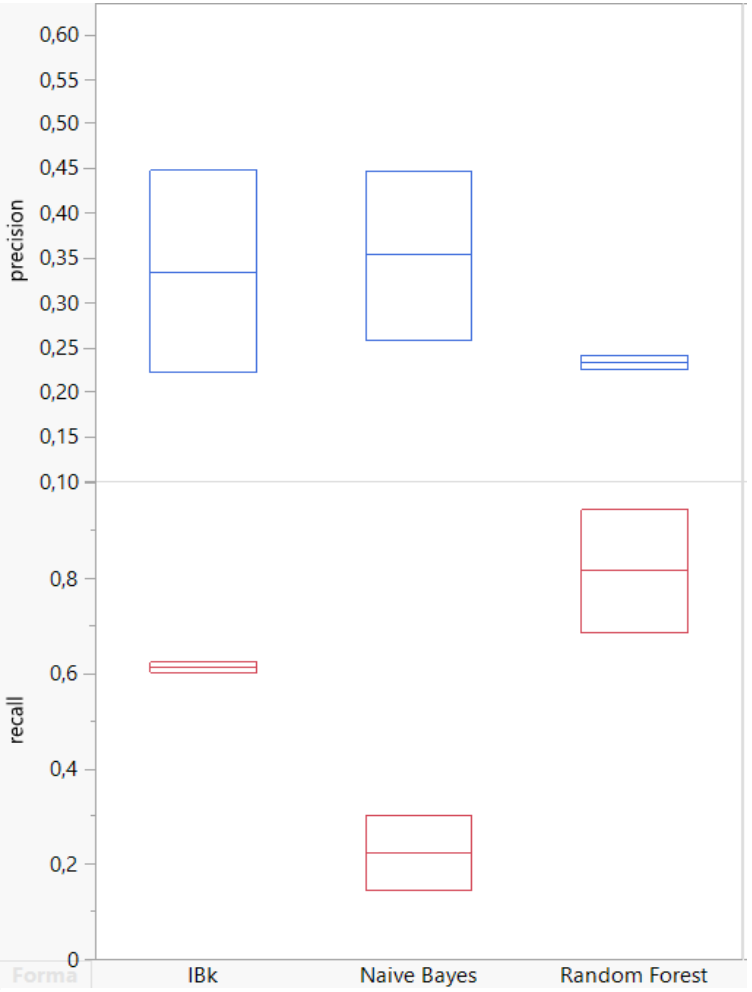
Nessuna tecnica



Over Sampling



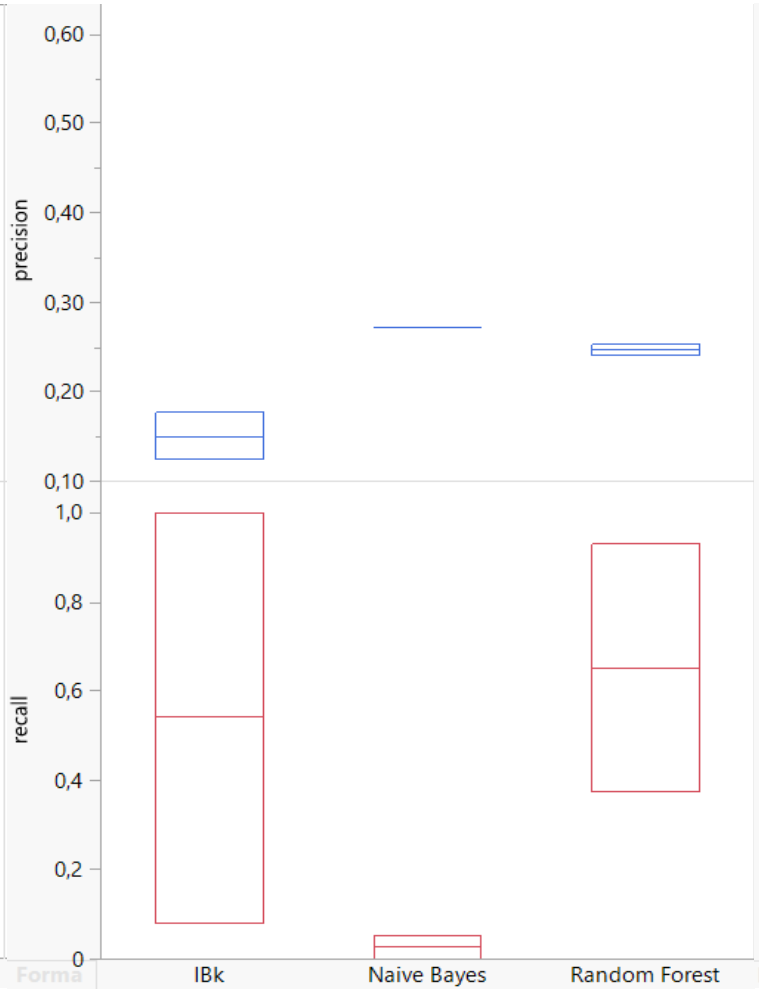
Under Sampling



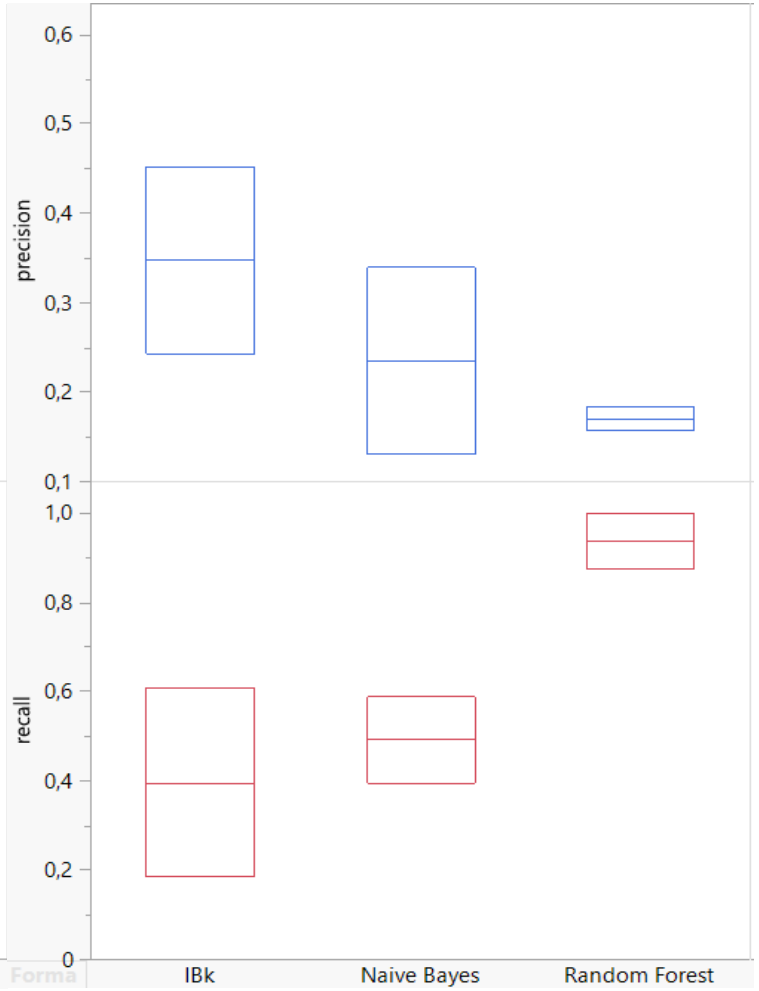
Risultati: BOOKKEEPER

Nessuna tecnica

Feature Selection



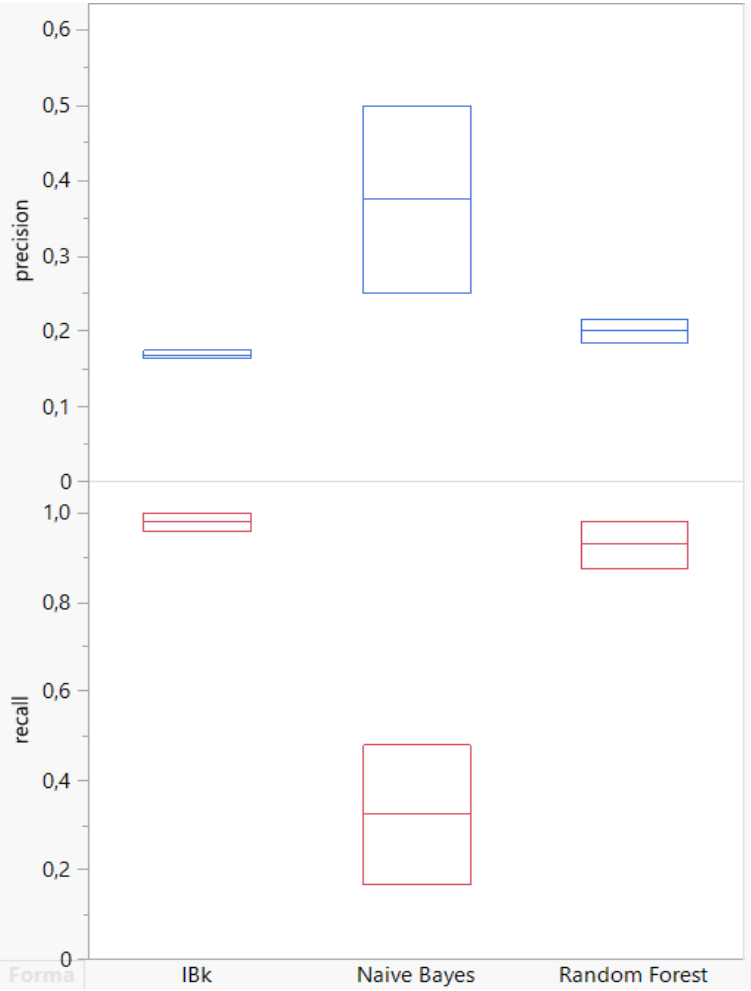
Cost Sensitive



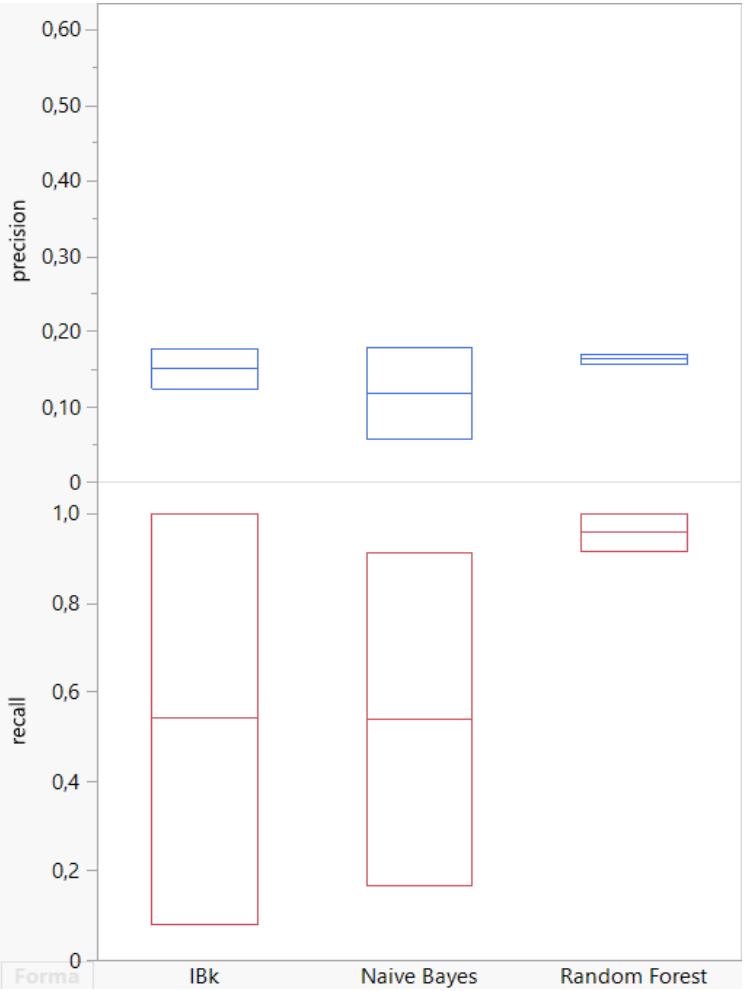
Risultati: BOOKKEEPER

Nessuna tecnica

FS e Under Sampling

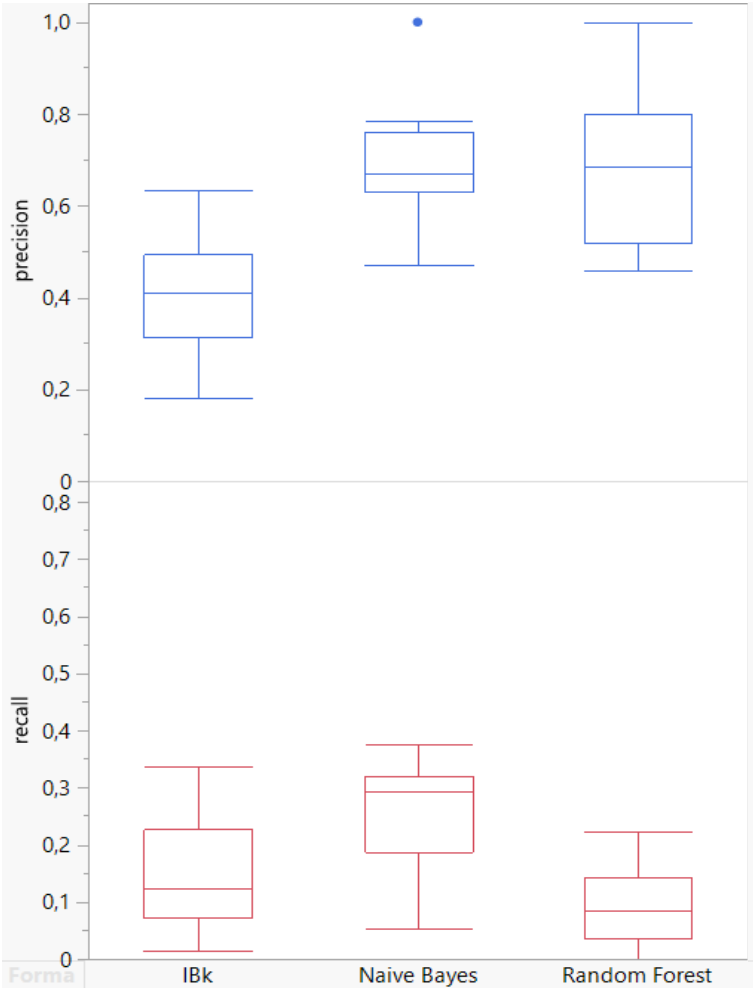


FS e Cost Sensitive

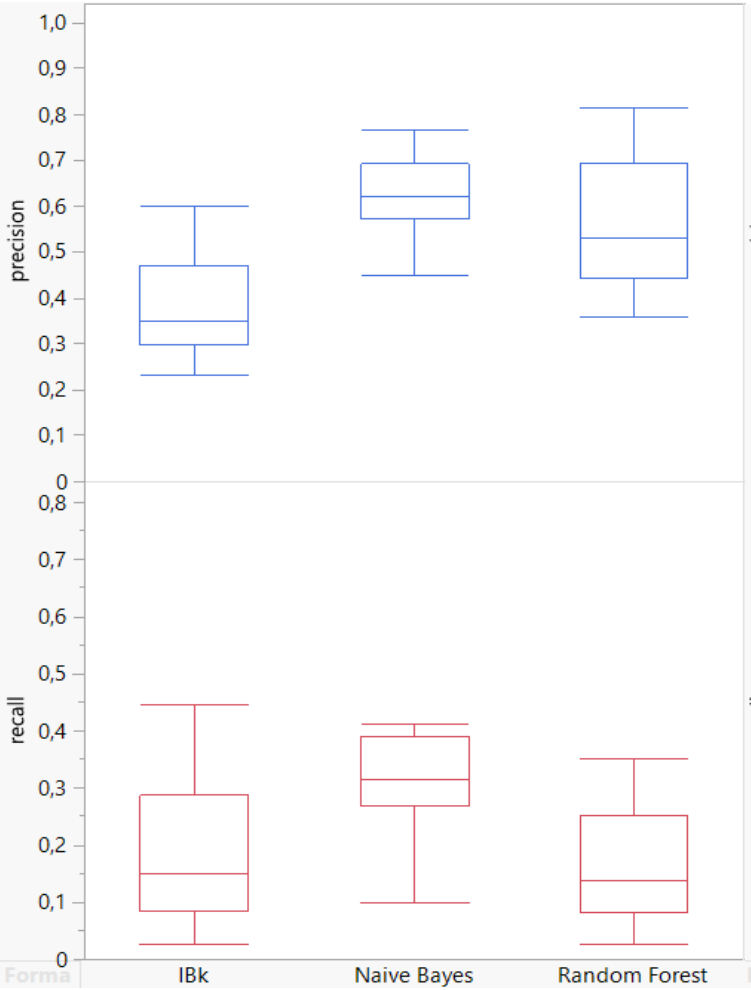


Risultati: OPENJPA

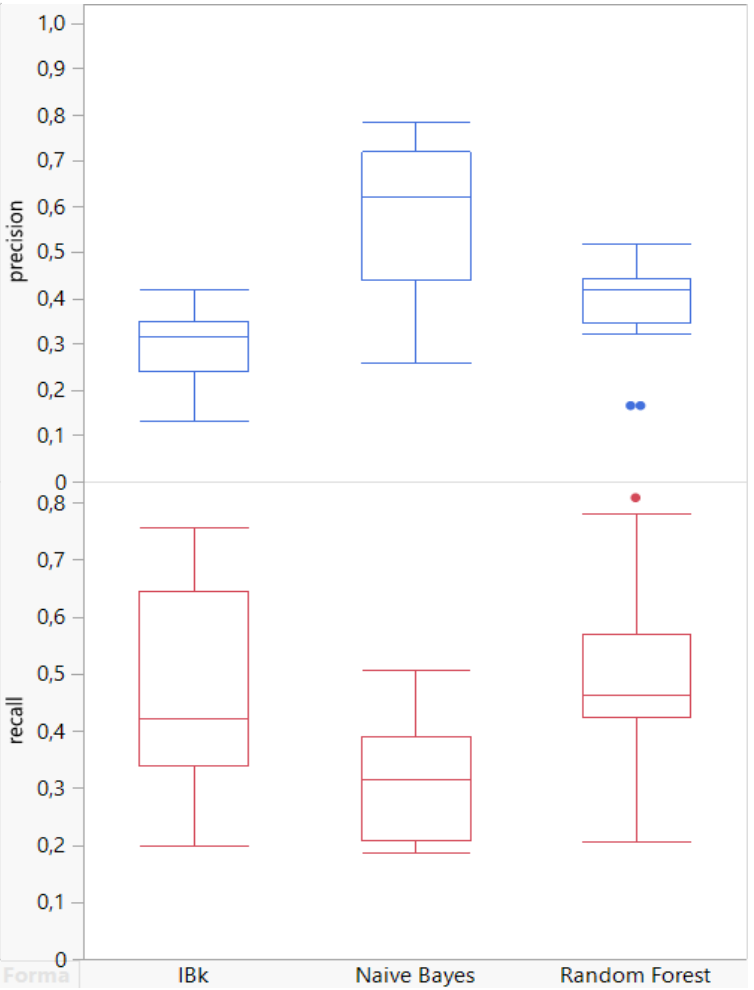
Nessuna tecnica



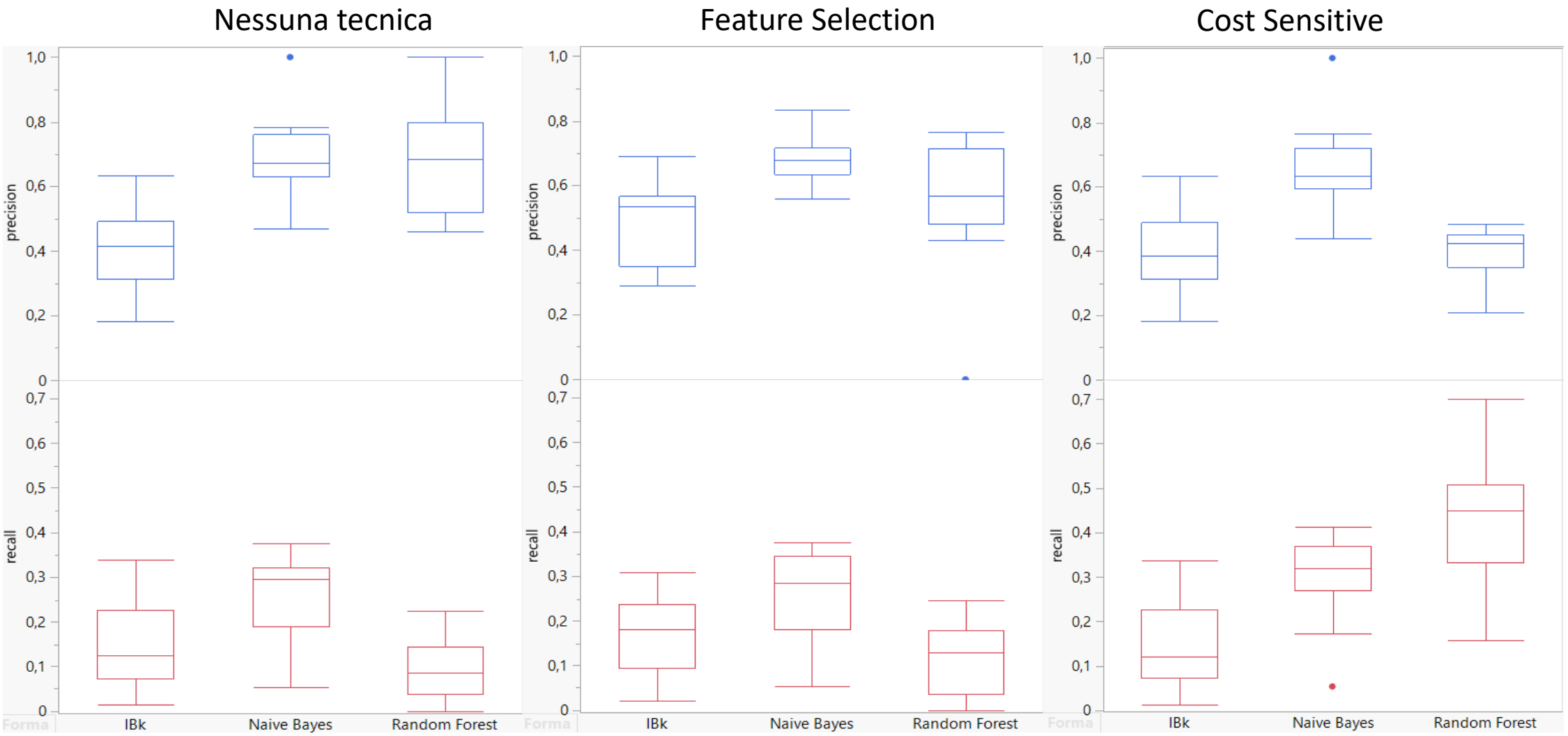
Over Sampling



Under Sampling

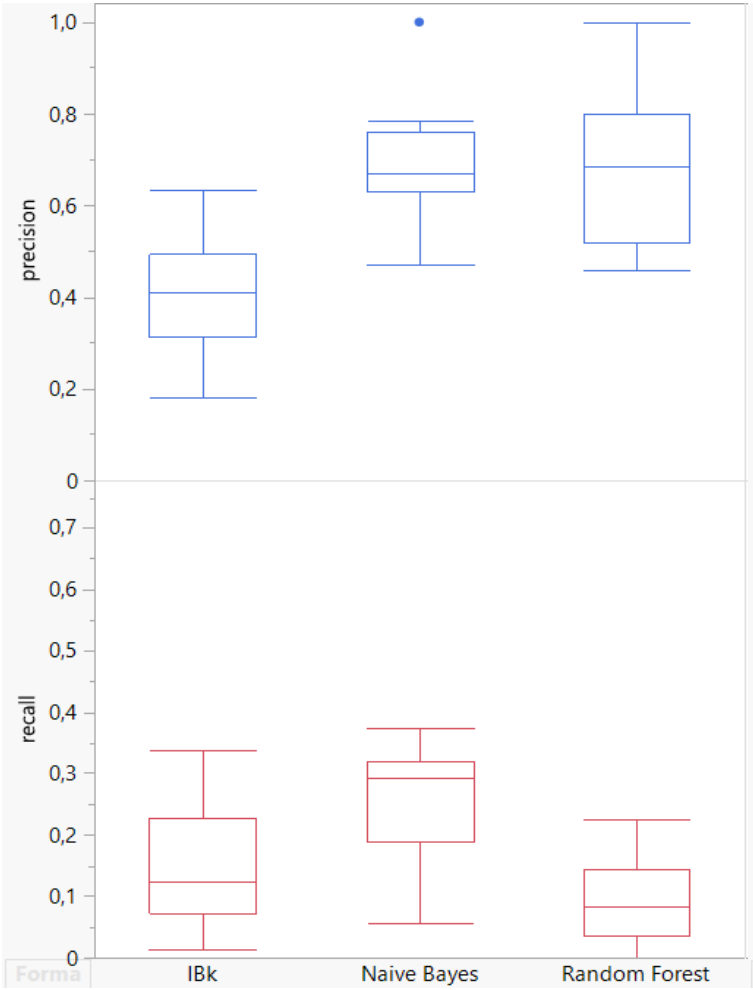


Risultati: OPENJPA

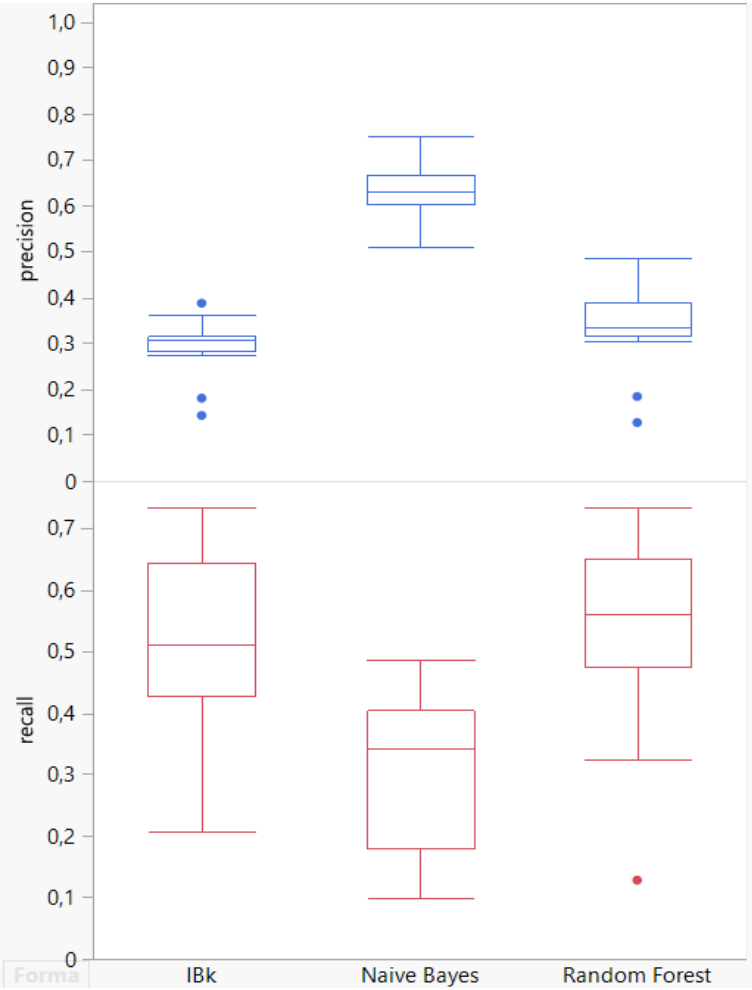


Risultati: OPENJPA

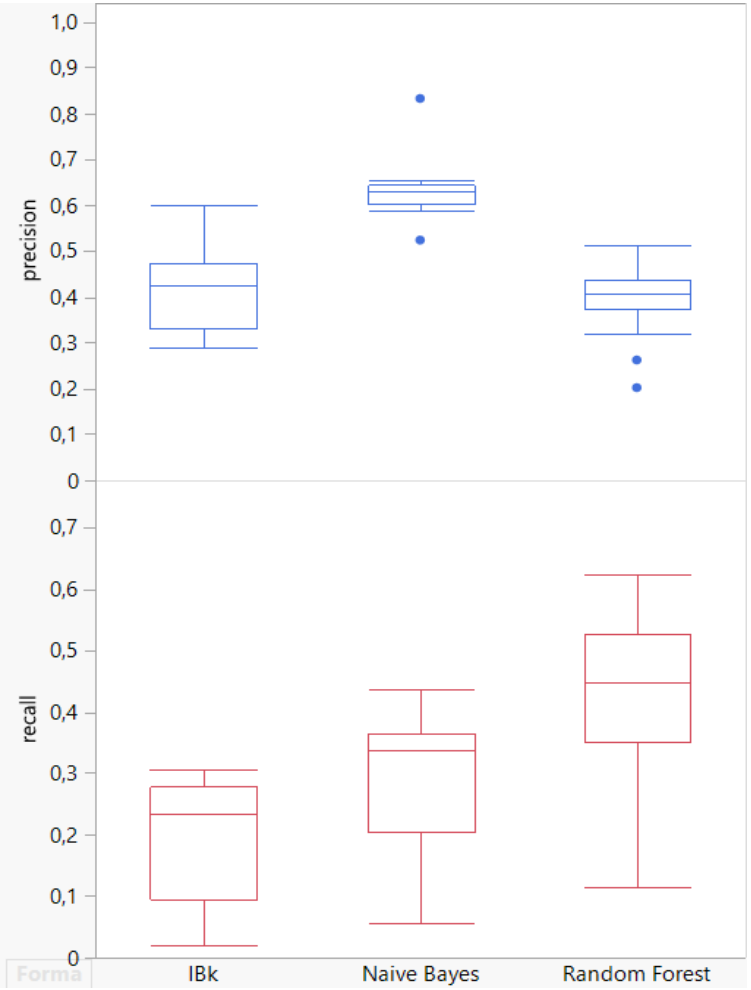
Nessuna tecnica



FS e Under Sampling



FS e Cost Sensitive



Analisi dei risultati

La slide 17 mostra i risultati dei classificatori senza applicare nessuna tecnica.

Dai risultati delle slide 18 e 21 si nota come entrambe le tecniche di sampling diminuiscano il valore di precision ma incrementino il valore della recall. Interessante è il comportamento di Naive Bayes e Random Forest nel progetto BOOKKEEPER. Il primo ha una drastica diminuzione della precision mentre il secondo ha l'incremento maggiore di recall tra tutti i classificatori.

Escluso il comportamento di Naive Bayes, la tecnica Over Sampling ha un impatto meno significativo rispetto alla tecnica Under Sampling. Quest'ultima garantisce un incremento significativo della recall a danno del valore di precision. Tuttavia anche con queste tecniche non è identificabile un classificatore dominante.

Dalle slide 19 e 22, si nota anche l'effetto della tecnica Cost Sensitive. Effetto quasi non visibile per il classificatore IBk, mentre per i restanti si nota una drastica diminuzione della precision e significativo aumento della recall.

Analisi dei risultati

Come si osserva dalle slide 19, la tecnica di feature selection comporta una diminuzione dell'accuratezza generale dei classificatori. Tuttavia possiamo notare che nel progetto OPENJPA, slide 22, la feature selection aiuta i classificatori ad eliminare le feature non significative guadagnando leggermente sul valore di precision e recall. Questo comportamento duale è anche dettato dal fatto che per il progetto BOOKKEEPER, sono state effettuate solo due run del walk forward, per mancanza di dati. Una ulteriore diminuzione dei dati comporta una scarsità eccessiva di informazioni per il training dei classificatori. Invece, il progetto OPENJPA possiede un gran numero di revisioni, il che potrebbe comportare un eccessivo carico di informazioni tra cui quelle non utili all'addestramento.

Si può concludere affermando che le tecniche di Sampling e Cost Sensitive Learning aumentano la recall e diminuiscano la precision per entrambi i progetti, mentre la tecnica Feature Selection diminuisce l'accuratezza generale per il progetto BOOKKEEPER mentre aumenta di poco l'accuratezza per il progetto OPENJPA.

Scelte di progettazione

Durante lo studio sono stati scartati solamente i ticket che non sono citati da alcun commit.

Dai ticket, ove possibile, la IV è stata ottenuta prendendo la release più vecchia tra le versioni citate nel campo AV. Per la correttezza di AV, è stato verificato che nessuna release elencata in questo campo sia maggiore o uguale alla release di resolution del bug.

Nel caso in cui FV corrispondesse a OV, la formula per calcolare la IV tramite proportion utilizzata è la seguente: IV = FV - P

Per entrambi i progetti, per il train set sono state scartate metà delle release più recenti per diminuire l'effetto dello snoring, mentre per il test set sono state utilizzate tutte le release disponibili.

Per entrambi i progetti sono state scartate le prime due run della valutazione dei classificatori, poiché per tali run non era possibile la costruzione di un train set.

Il valore del proportion non è costante tra le iterazioni, ma viene calcolato ad ogni run con le informazioni disponibili in tale run.

Links

- **GitHub**: https://github.com/jack-mack15/ISW2_Project
- **SonarCloud**: https://sonarcloud.io/summary/overall?id=jack-mack15_ISW2_Project