



Machine Learning applicato all'Ingegneria del Software

Ar dovino Luca
0345008



INDICE DEI CONTENUTI

- CONTESTO
- PROGETTAZIONE
- RISULTATI BOOKKEEPER
- RISULTATI OPENJPA
- LINK




CONTESTO: MISURARE IL SOFTWARE

- Nell'ingegneria del software è importante il concetto di **misurazione**, perché ciò che non può essere misurato non può essere controllato
- Spesso, la misurazione aumenta la sua utilità se considerata in funzione della **predizione**
- La misurazione deve affrontare problematiche legati ai costi, tra cui costi legati ai **bug** (i costi dei bug sfiorano i trilioni di dollari)



CONTESTO: COME PREVENIAMO I BUG?

- Un'analisi di qualità del software è dispendiosa e le risorse sono finite (esistono progetti con budget diversi per la qualità)
- Vari approcci limitano parzialmente il problema (es. code review)
- Oltre ai limiti economici, bisogna tener conto anche limiti legati al tempo (test troppo lunghi non vanno bene per tutti i software)



CONTESTO: L' OBIETTIVO E' UN TESTING EFFICIENTE

- Bisogna favorire approcci intelligenti per ottimizzare le risorse in modo da ottenere un processo di testing meno dispendioso
- Una soluzione è l'applicazione di modelli di **Machine Learning** per predire le classi «buggy» (bug localization)
- Nell'ambito del progetto, si valutano le prestazioni di alcuni modelli di Machine Learning applicati a dataset diversi



PROGETTAZIONE

- Progetti di riferimento: apache Bookkeeper, apache Openjpa
- Fonte dei dati: Jira(ticket e release) + Git (commit)
- Tecnica di labeling: Proportion (Cold start + Increment)
- Tecnica di valutazione: Walk Forward
- Classificatori: RandomForest, IBK, Naive Bayes
- Tecniche per valutare i classificatori: Feature Selection, Sampling, Cost-Sensitive Classification
- Comparare l'accuratezza (Precision, Recall, AUC, Kappa, NPofB20)



PROGETTAZIONE: ASSUNZIONI

- Per entrambi i progetti è stata presa in considerazione solamente la prima metà delle release per evitare lo «snoring»
- Se un bug è stato risolto nella release in cui è stato introdotto, non viene considerato
- I ticket senza commit associati non sono stati presi in considerazione
- Se una release non ha commit associati, viene comunque considerata nel calcolo delle metriche

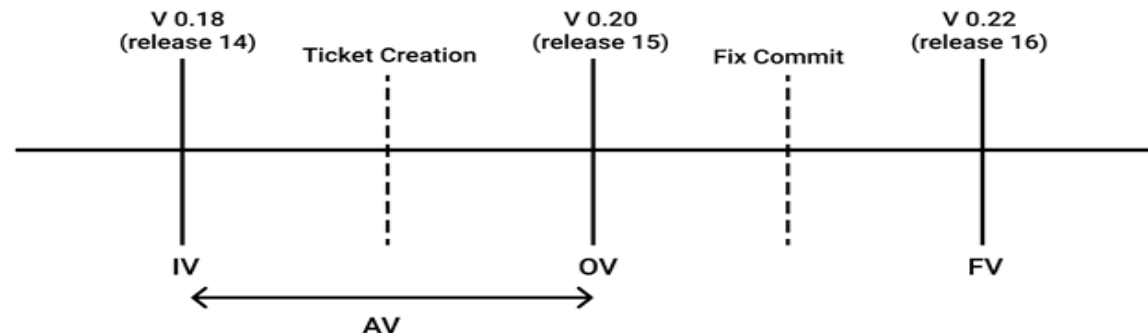


PROGETTAZIONE: COSTRUZIONE DEL DATASET

- Da **Jira** sono stati raccolti l'elenco delle release di ogni progetto, tutti i ticket relativi ai bug chiusi (cioè che sono stati risolti)
- Da **Git** sono stati presi, per ogni release, i commit relativi ai ticket d'interesse. Se una release non ha commit, si considerano validi quelli della release precedente
- Per tutte le classi di ogni progetto sono state calcolate le metriche scelte ed è avvenuta l'**etichettatura** in «buggy» per tali classi

PROGETTAZIONE: QUANDO UNA CLASSE E' CONSIERATA «BUGGY»?

- Una classe è considerata buggata dalla release introduzione del bug (IV, **Injected Version**) alla release in cui il bug viene «fixato» (FV, **Fixed Version**)
- Le release che vanno dalla IV alla OV (**Opening Version**) sono le cosiddette AV (**Affected Version**)



PROGETTAZIONE: PROPORTION

- In Jira non è sempre presente l'IV, il problema è superabile assumendo una proporzionalità diretta tra il tempo che passa tra FV ed IV ed il tempo che passa tra FV e OV
- **Proportion** è la tecnica per calcolare l'IV sfruttando tale proporzionalità

$$P = (FV - IV) / (FV - OV)$$

$$\text{Predicted IV} = FV - (FV - OV) * P$$

PROGETTAZIONE: PROPORTION (2)

- Nel progetto sono state applicate due varianti di Proportion per il calcolo di p : Cold Start e Increment
- **Cold Start:** si calcola il p che serve prendendo in considerazione i valori di p per un certo numero di progetti simili. Nel caso del progetto, tale tecnica è stata usata quando i valori disponibili nel progetto d'interesse erano meno di 5, e i progetti presi in considerazione sono alcuni progetti Apache.
- **Increment:** si calcola p usando facendo la media di tutti i fix precedenti. La scelta è dovuta alla vicinanza temporale tra le release

PROGETTAZIONE: METRICHE SCELTE

Metrica	Descrizione
Linee di codice (LOC)	Numero totale di righe del codice (ultima release)
Linee di commenti	Numero totale di righe di commenti (ultima release)
Numero di autori	Numero di sviluppatori che hanno toccato la classe
Numero di commit	Numero di commit in cui compare la classe
Numero di fix	Numero di fix-commit in cui compare la classe
ChangeSetSize*	Numero di file modificati insieme in un commit
Età	Anzianità della release
Numero di linee di codice aggiunte*	Numero di righe di codice totali aggiunte

Delle metriche segnate con * è stato calcolato anche il massimo

PROGETTAZIONE: WALK FORWARD

- Il **Walk-Forward** è una tecnica di validazione «time-series» e quindi si basa sull'ordine temporale dei dati
- Per questo ha bisogno di molteplici «run» per funzionare, perché sfrutta le release precedenti come training per fare la predizione su quella attuale

Run	Part				
	1	2	3	4	5
1	Testing				
2	Training	Testing			
3	Training	Training	Testing		
4	Training	Training	Training	Testing	
5	Training	Training	Training	Training	Testing

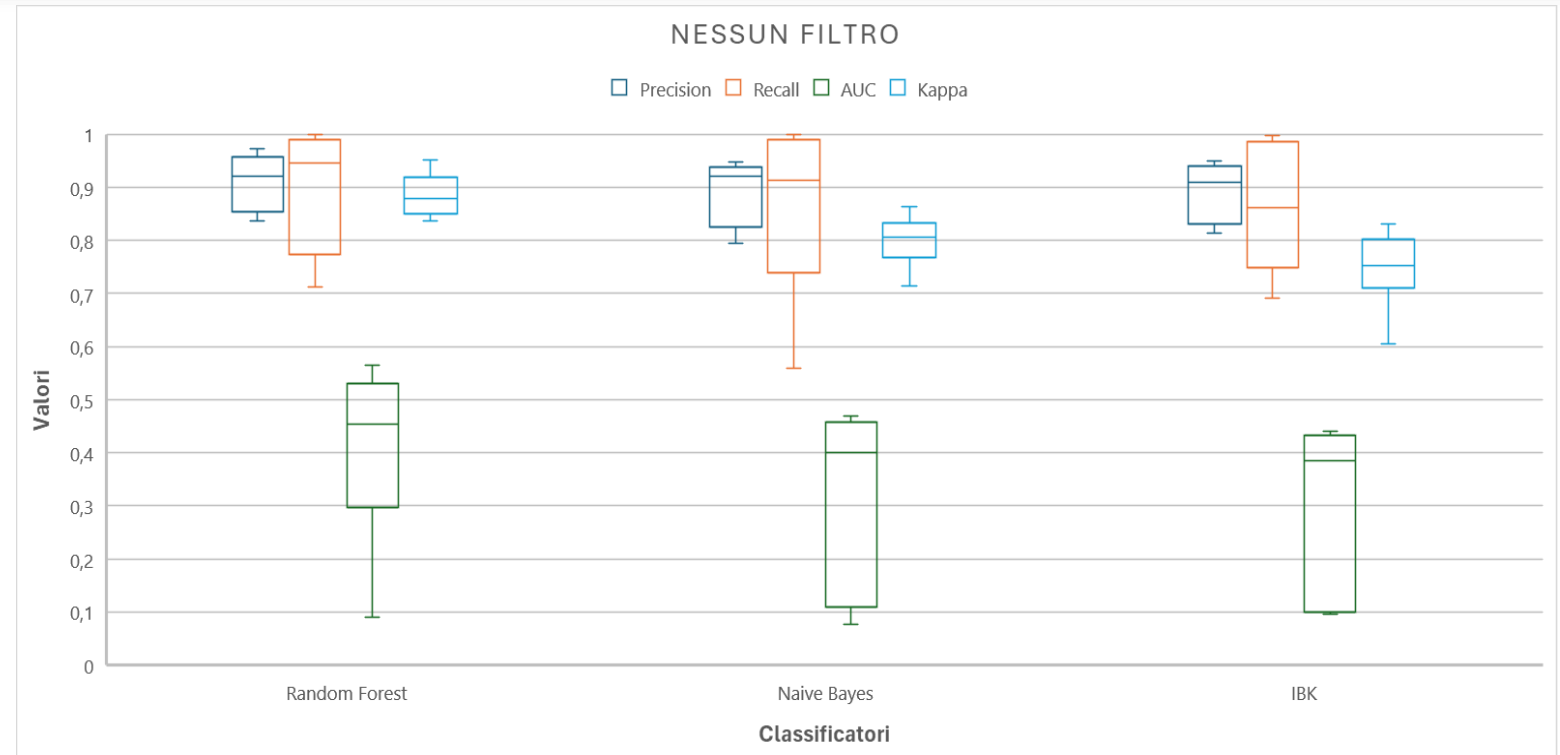


PROGETTAZIONE: CONFIGURAZIONI DEI CLASSIFICATORI

- I classificatori utilizzati sono stati IBK, Naive Bayes e Random Forest
- Tali classificatori sono stati applicati ad entrambi i progetti combinati con le tecniche di Feature Selection, Sampling e Cost-Sensitive Classifier
- Le configurazioni applicate ai progetti sono state:
 - Configurazione 1: Nessun filtro
 - Configurazione 2: Solo Feature Selection
 - Configurazione 3: Feature Selection + Undersampling
 - Configurazione 4: Feature Selection + Cost Sensitive ($CFN = 10 * CFP$)

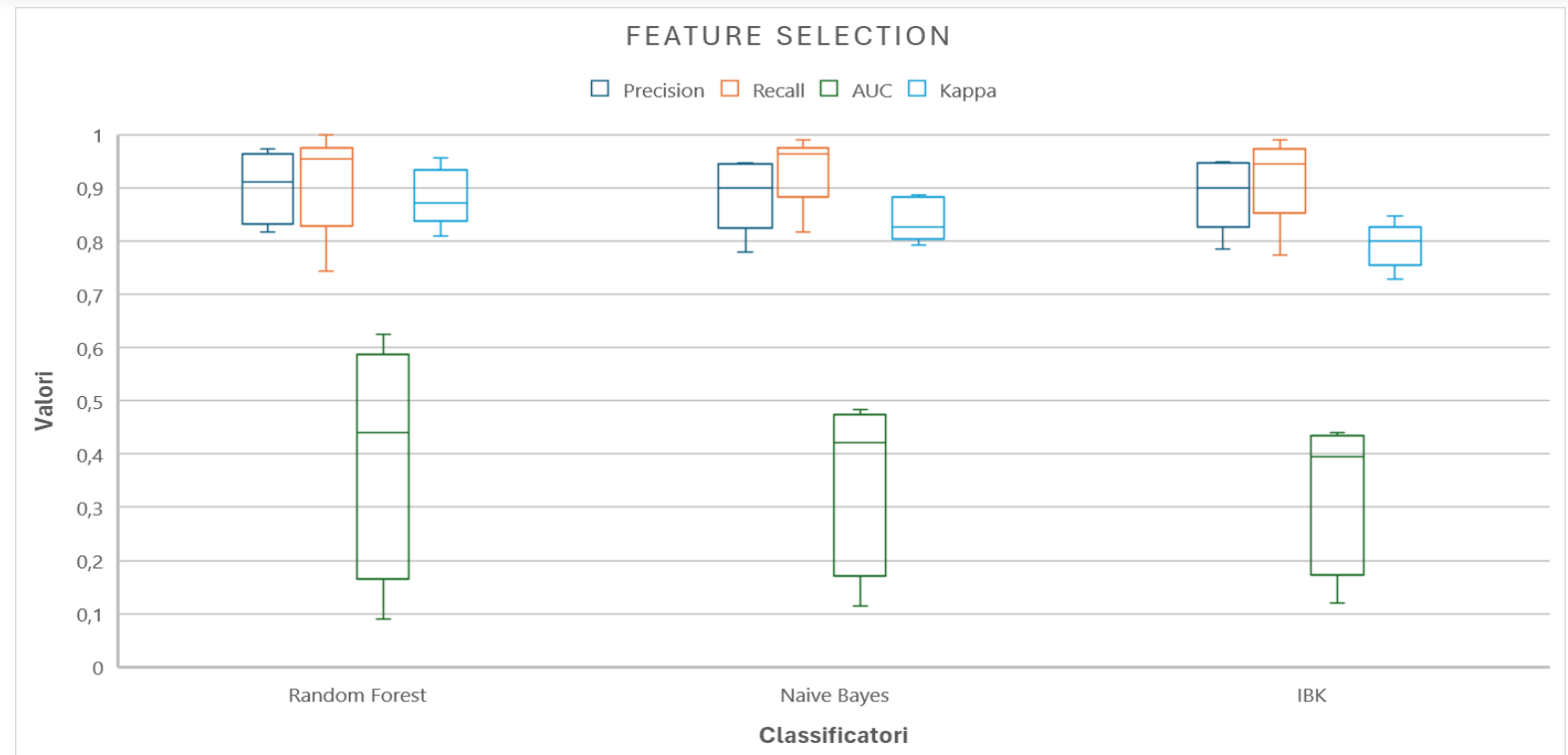
RISULTATI BOOKKEEPER: NESSUN FILTRO

- Random Forest ha miglior Precision (ex-equo con Naive Bayes), Recall, AUC e Kappa ed NPofB20 (mediana ≈ 0.65)



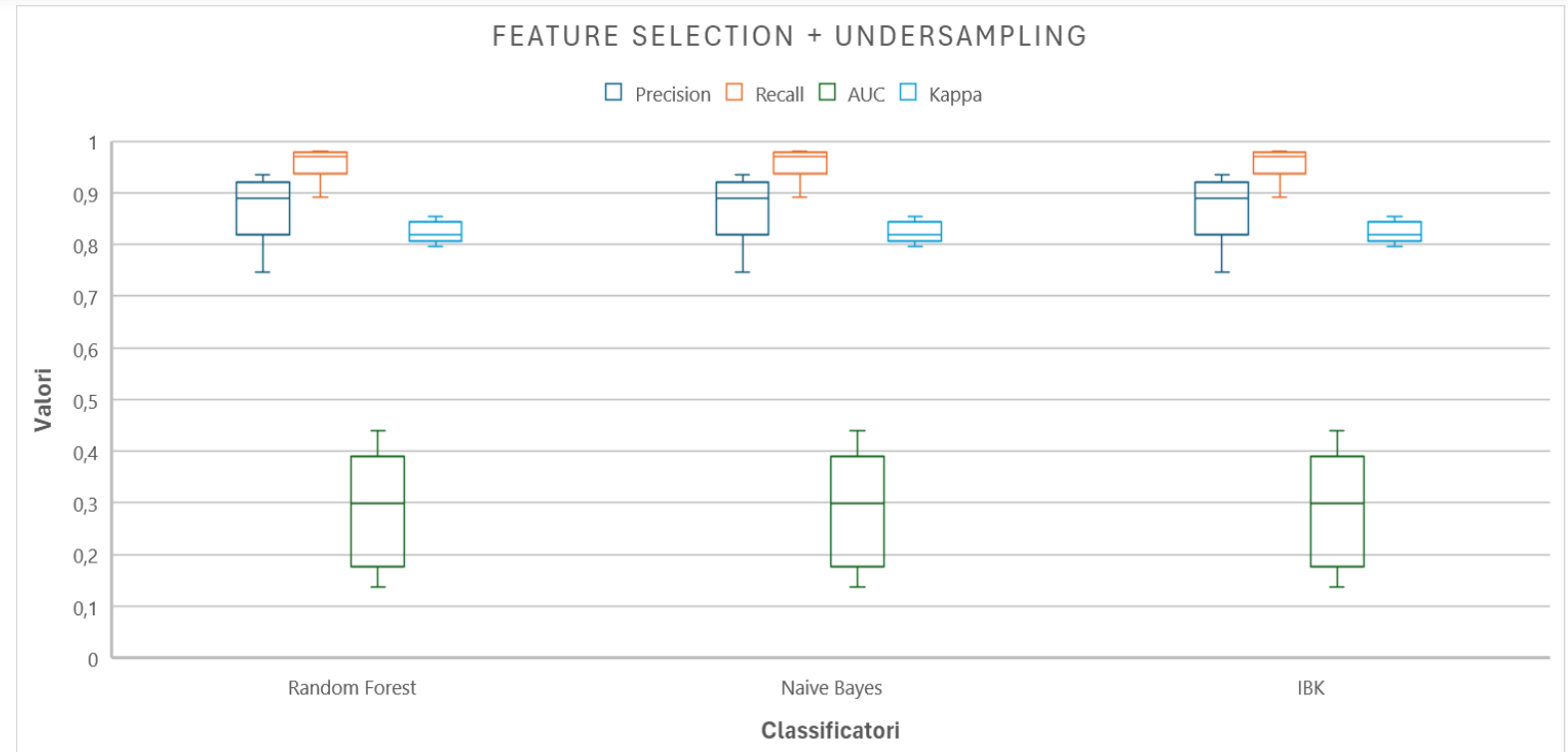
RISULTATI BOOKKEEPER: SOLO FEATURE SELECTION

- Random Forest ha miglior Precision, Kappa, AUC ed NPofB20 (mediana ≈ 0.66)
- Naive Bayes ha la miglior Recall



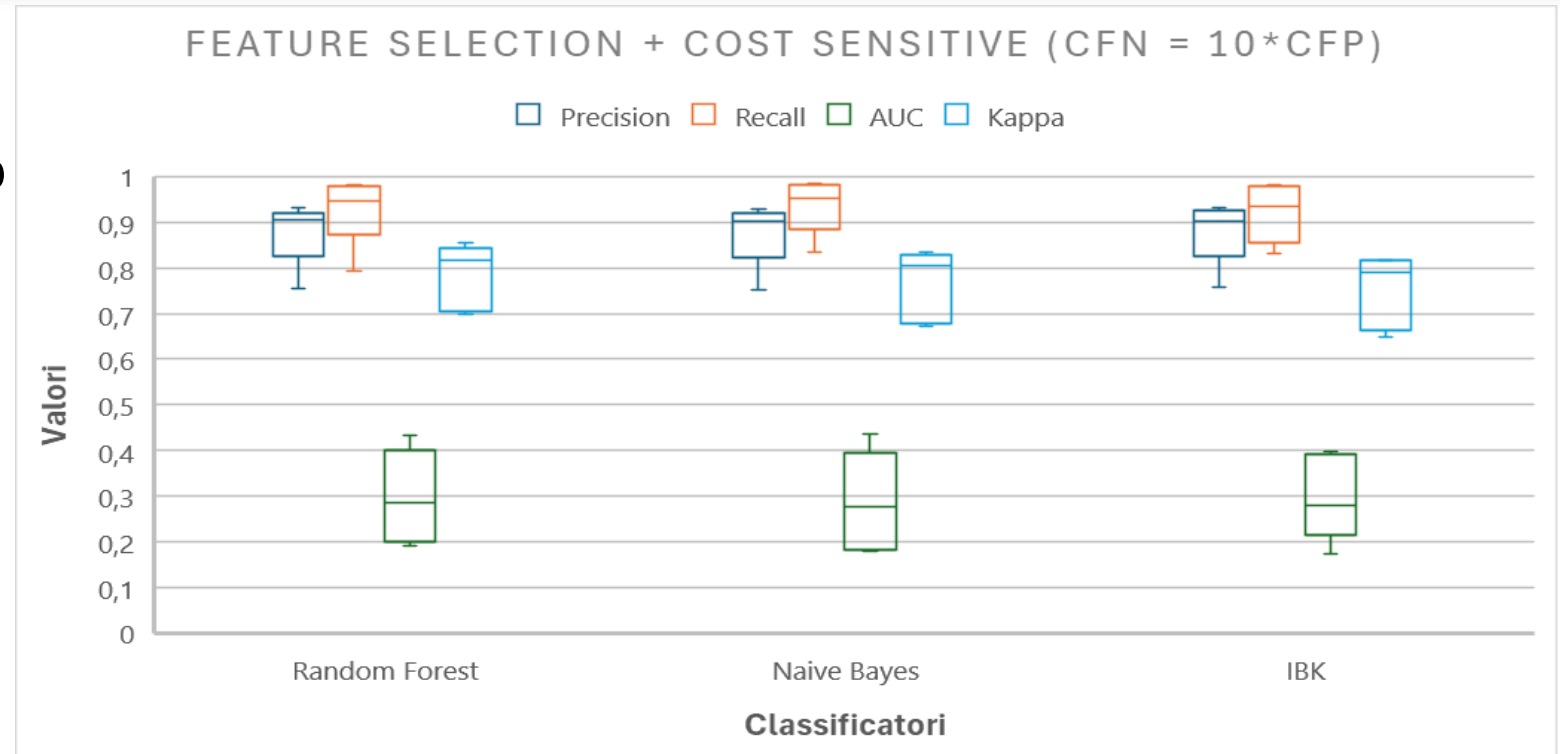
RISULTATI BOOKKEEPER: FEATURE SELECTION + UNDERSAMPLING

- In questo caso i classificatori presentano un equilibrio per Precision, Recall, Kappa e AUC
- Equilibrio anche per NPofB20 (mediana $\approx 0,6$)



RISULTATI BOOKKEEPER: FEATURE SELECTION + COST SENSITIVE (CFN = $10 \cdot \text{CFP}$)

- Random Forest ha miglior Recall (ex-equo con Naive Bayes), Kappa, AUC ed NPofB20 (mediana ≈ 0.66)
- Equilibrio su Precision



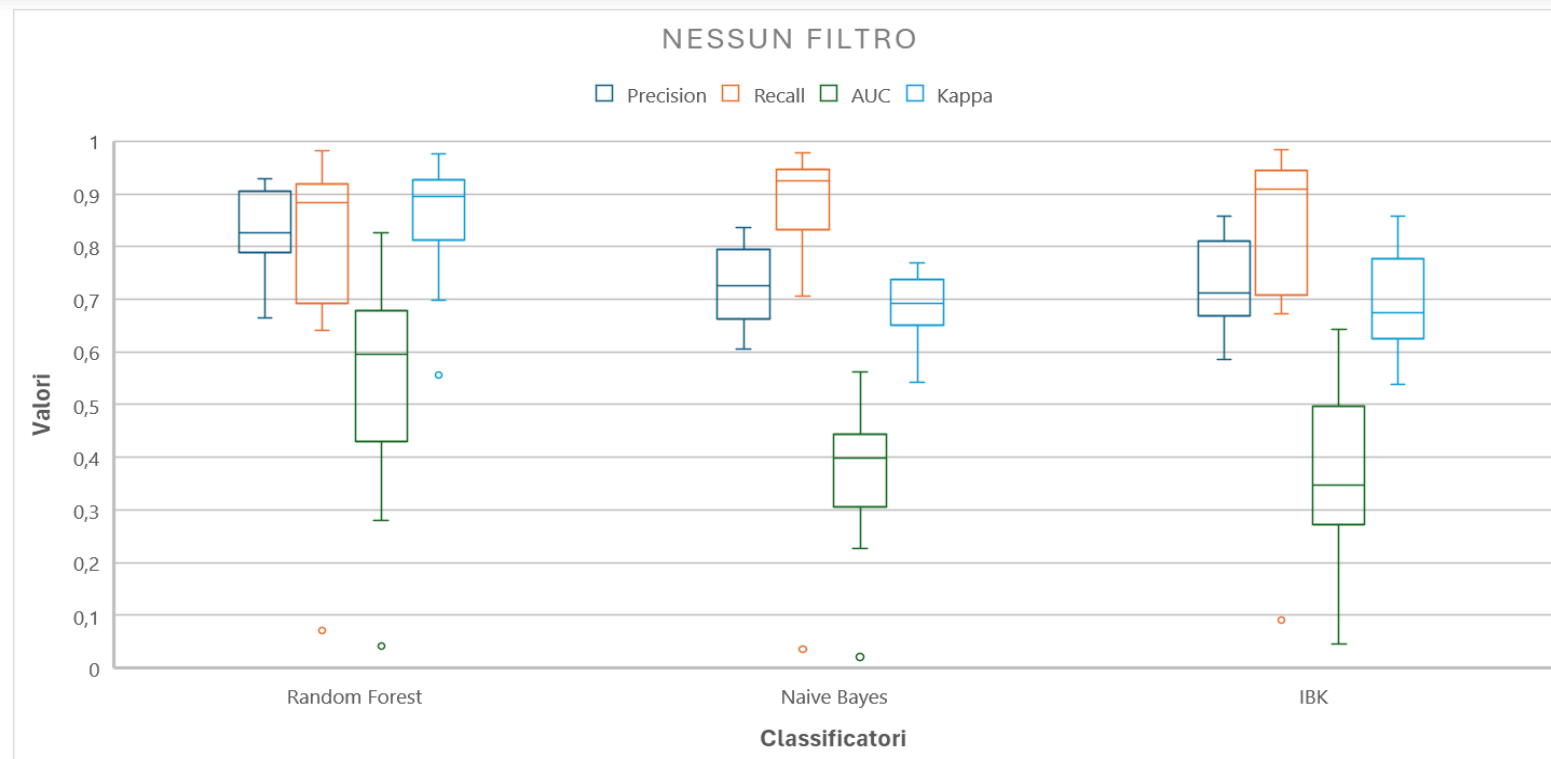


RISULTATI BOOKKEEPER: CONCLUSIONI

- Il miglior classificatore in generale risulta essere Random Forest, anche se di poco perché le prestazioni dei classificatori nelle varie configurazioni risultano equilibrate
- Non c'è una configurazione che spicca delle altre, ma si noti come in generale l'Undersampling e il Sensitive-Threshold vadano a migliorare la Recall andando ad abbassarne la dispersione

RISULTATI OPENJPA: NESSUN FILTRO

- Random Forest ha la miglior Precision, AUC, Kappa e NPofB20 (mediana $\approx 0,51$)
- Naive Bayes ha la miglior Recall



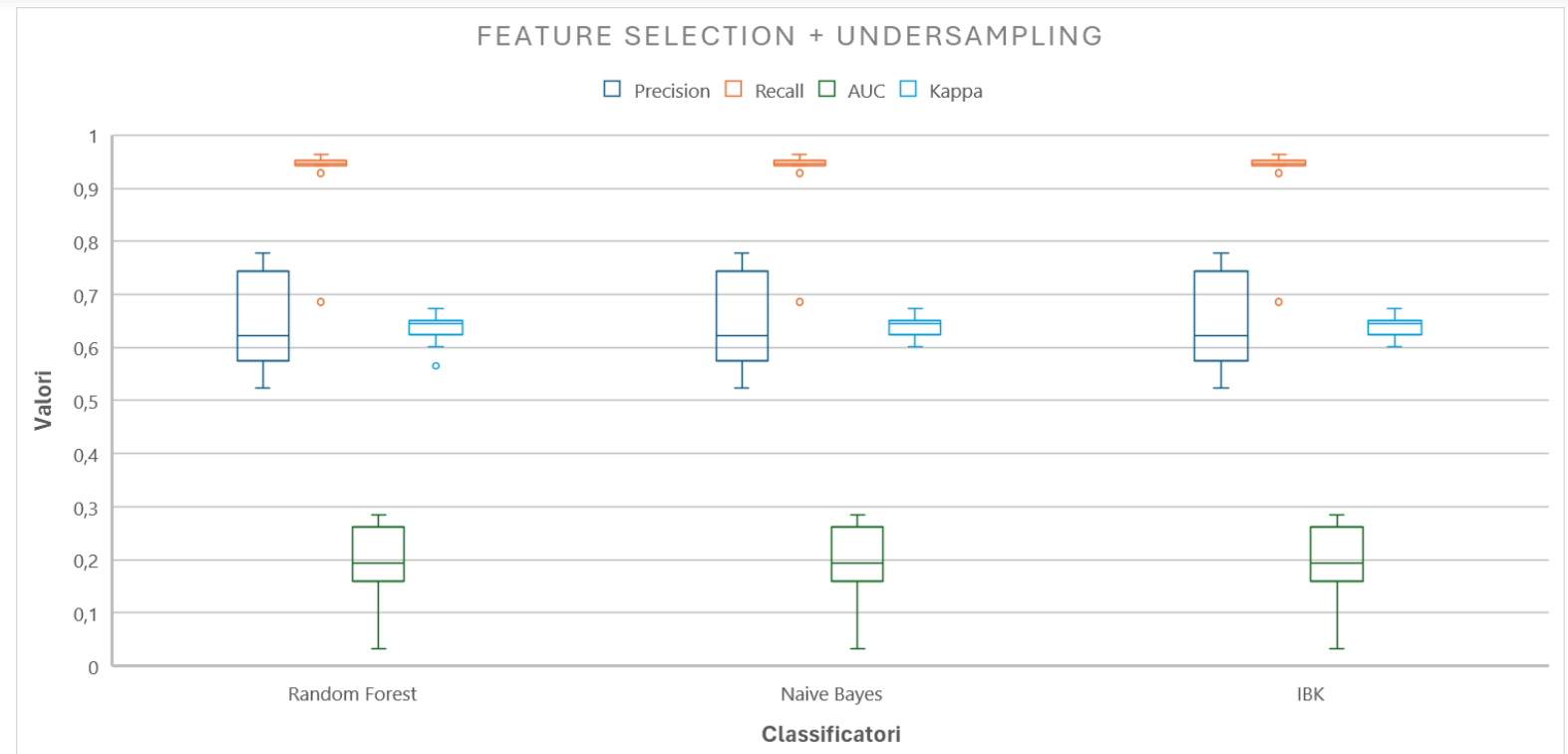
RISULTATI OPENJPA: SOLO FEATURE SELECTION

- Random Forest ha la miglior Precision, AUC, Kappa e NPofB20 (mediana $\approx 0,81$)
- Naive Bayes ha la miglior Recall



RISULTATI OPENJPA: FEATURE SELECTION + UNDERSAMPLING

- Equilibrio tra i classificatori sia per Precision, che per Recall, che per AUC, che per Kappa
- Equilibrio anche per NPofB20 (mediana $\approx 0,8$)



RISULTATI OPENJPA: FEATURE SELECTION + COST SENSITIVE (CFN = 10*CFP)

- IBK ha miglior AUC e Kappa
- Equilibrio su Precision e Recall
- Random Forest ha il miglior NPofB20 (mediana $\approx 0,81$)





RISULTATI OPENJPA: CONCLUSIONI

- In questo caso non c'è un classificatore migliore di altri, con tutti i classificatori che migliorano determinate metriche in base alla configurazione utilizzata. L' NPofB20 è l'unica metrica in cui RandomForest continua a primeggiare nel valore mediano
- Anche in questo caso UnderSampling e Cost sensitive hanno le migliori Recall con valori mediani quasi identici ed entrambi con dispersioni bassissime.
- Random Forest + Feature Selection è la configurazione con i valori mediani più alti per tutte le metriche



LINK

- Link Github: <https://github.com/Ardow22/lsw2Project>
- Link SonarCloud:
https://sonarcloud.io/summary/new_code?id=Ardow22_lsw2Project