MACHINE LEARNING FOR SOFTWARE ENGINEERING

Matteo Federico-0321569

Università degli studi di Roma- Tor Vergata

Anno 2021-2022

Obbiettivi

- Il progetto sviluppato ha l'obbiettivo di studiare gli effetti che hanno diverse tecniche di filtraggio sull'accuratezza di tre classificatori di Weka.
- Lo scopo dei tre classificatori è quello di riuscire a stabilire se una classe è buggy o meno, un bug è un difetto nel progetto che porterà a un malfunzionamento
- I tre classificatori studiati e valutati:
  1. Naive Bayes
  2. IBk (con k=1)
  3. Random Forest
- I dati sono stati prelevati da due progetti open source di Apache:
  1. Avro
  2. Bookkeeper

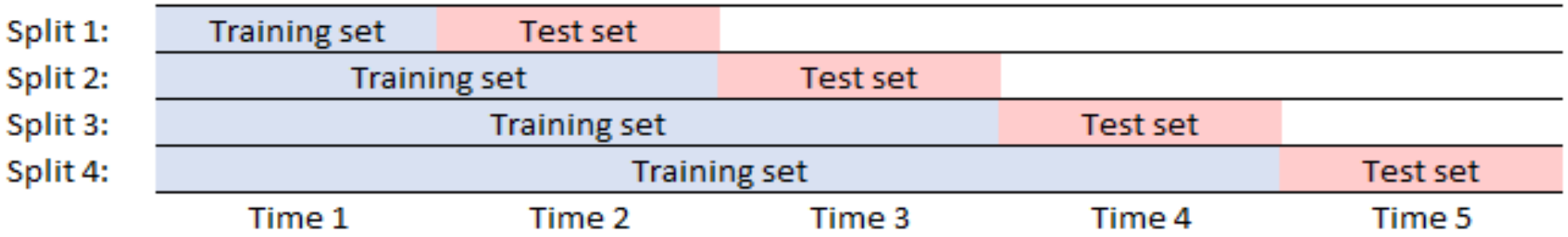
DATASET

- Per ottenere i dati, ho creato un programma in java, che permette di scoprire se una classe in una determinata release ha avuto qualche report che segnalava la presenza di bug.
- Per far ciò è stato usato JIRA, che permette di ottenere le affected version, le fixed version e le opening version di ogni bug scoperto per i progetti.
- Sono stati considerati per entrambi i progetti solo la prima metà delle release per ridurre al minimo la presenza di classi affette da snoring.
- Nel caso in cui l' affected version non fosse presente o consistente è stato utilizzato un metodo di proportion incrementale per ottenere un dataset sensato.
- Tutti i bug sono stati collegati alle classi tramite gli ID dei commit che andavano a correggere un determinato bug.

N.B.
Il programma non è capace di scoprire bug o dire se una classe è buggy o meno, è capace solo di vedere se un bug scoperto e correttamente riportato è correlato ad una classe, sotto determinate ipotesi

Divisione del DATASET

- Il dataset è stato diviso con la tecnica walk-forward:
- Inizialmente la prima versione, con la buggyness calcolata con le informazioni conosciute al momento della release, fa da training set, e la seconda, con la buggyness aggiornata all'ultima release, da testing set
- All'iterazione n, le prime n versioni costituiscono il training set e la (n+1)-esima il testing set. La buggyness del testing set è calcolata a partire dall'ultima release disponibile.
- L'accuratezza è calcolata come la media sulle varie run.



Metriche

Ho scelto le variabili in base alle seguenti ipotesi:

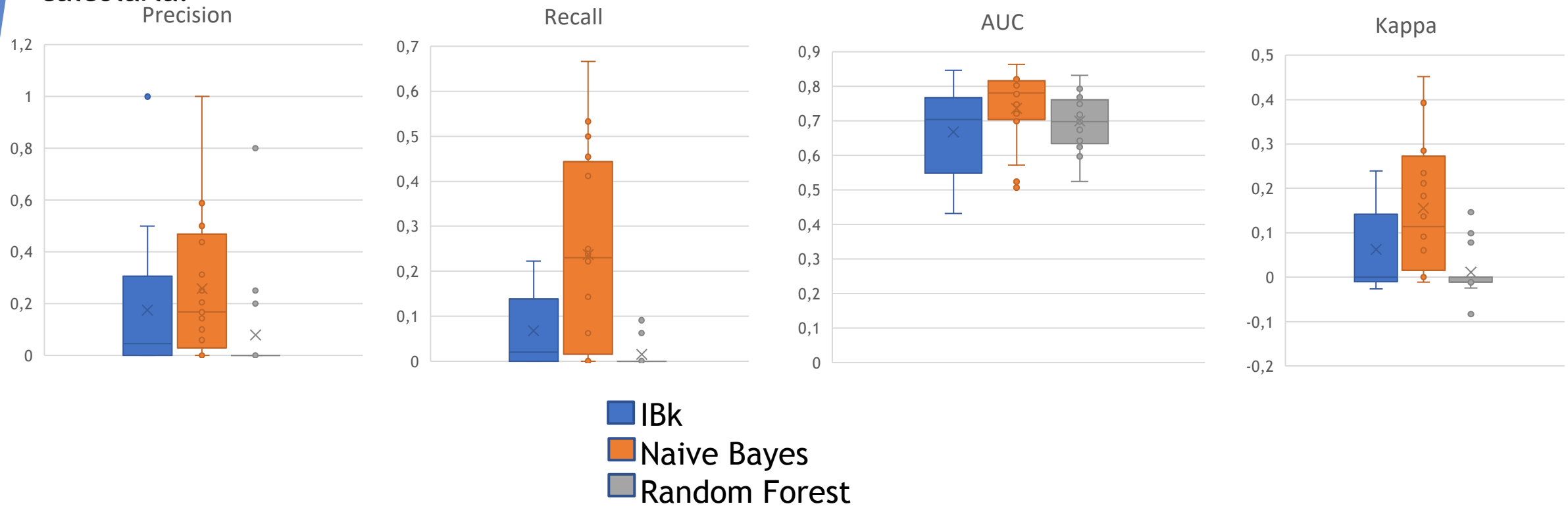
1. Piu una classe è grande più è possibile che la classe sia buggy
2. Piu una classe è stata modificata più è possibile che la classe sia buggy

Ho considerato le metriche per release eccetto per che per il numero di autori. Per il numero di commit è stato considerato sia globalmente che solo per release

- Size: taglia della classe al momento della release
- Commit number: numero di commit relativi alla classe
- Commit number release: numero di commit relativi alla classe in una determinata release
- LOC touched: linee di codice modificate dai commit
- LOC added: linee di codice aggiunte alla classe
- Max LOC added: maggior numero di linee di codice aggiunte in un unico commit
- Avg LOC added: media delle linee di codice aggiunte per commit
- Churn: differenza tra le linee aggiunte o rimosse dalla classe all'interno della release
- Max churn: massimo churn ottenuto in un unico commit
- Avg churn: churn medio per commit
- Authors' number: numero di autori che hanno lavorato sulla classe

Avro

Per il progetto Avro sono stati considerati solo i bug considerati chiusi o risolti. In totale sono stati trovati 183 bug di cui solo in 130 vi era coerenza tra affected version, fixed version e opening version. Tutti i bug in cui l'opening version non era riportata non sono stati considerati al fine di creare il dataset. In caso in cui l'affected version non era presente o non fosse stata coerente, è stato applicato un metodo di proportion per calcolarla.

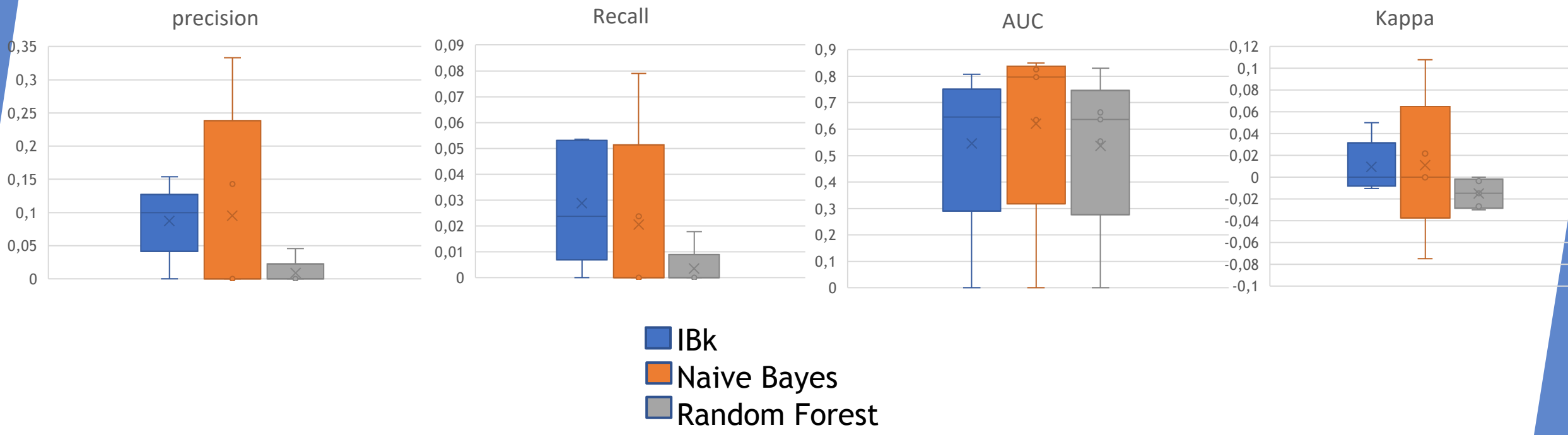


Dai grafici si può notare che, senza alcun filtraggio, Naive Bayes si comporta in media meglio sotto tutte le metriche considerate mentre Random Forest è quello che si comporta peggio.

Bookkeeper

Per il progetto Bookkeeper sono stati trovati 443 bug considerati chiusi, di cui solo 229 correttamente riportati. Anche per Bookkeeper è stato utilizzato il metodo della proportion incrementale per calcolare le affected version inconsistenti.

Bookkeeper a differenza di Avro anche se ha un numero di commit e di file molto più ha un numero di release veramente basso. Infatti anche se sono stati considerati ben 1700 commit sono state considerate solo 5 release



Dai grafici si può notare che senza alcun filtraggio Naive Bayes e IBk si comportano entrambi meglio rispetto a Random Forest ma è difficile stabile chi dei due ha il comportamento migliore

Filtri Valutati

Sampling:

- No Sampling
- Under Sampling
- Over Sampling
- Smote

Feature Selection:

- No Feature Selection
- Best First

Cost Sensitivity:

- No cost Sensitive
- Sensitive Threshold
- Sensitive Learning

Sampling

Il sampling è necessario per evitare che ci siano forti sbilanciamenti tra le istanze buggy e non buggy presentate al classificatore, che potrebbero portarlo ad avere una maggiore accuratezza nel predirre istanze della classe prevalente e una accuratezza minore per la classe minoritaria

Under sampling

Vengono considerate tutte le istanze minoritarie e solo una parte di quelle maggioritarie

Over sampling

Le istanze minoritarie vengono replicate fino ad avere un numero pari di istanze maggioritarie e minoritarie

Smote

Si introducono delle istanze fittizie che vengono create in base alle distanze medie dei punti generati dalle istanze già presenti

Feature Selection

- Lo scopo della feature selection è considerare solo il più piccolo sottoinsieme di attributi significativo invece che l'intero insieme di variabili nel fare le predizioni, in modo da poter ridurre i costi dell'apprendimento e migliorare i risultati.

- Per effettuare feature selection è necessario selezionare un metodo di ricerca degli attributi, e un algoritmo per la loro valutazione.

- Attribute evaluator: CfsSubsetEval, search method: Best First

Cost Sensitivity

Un classificatore cost sensitive è un classificatore che assegna costi diversi agli errori. Nel nostro caso, ogni falso negativo costa 10 volte in più rispetto a un falso positivo, essendo un errore più grave per i nostri scopi.

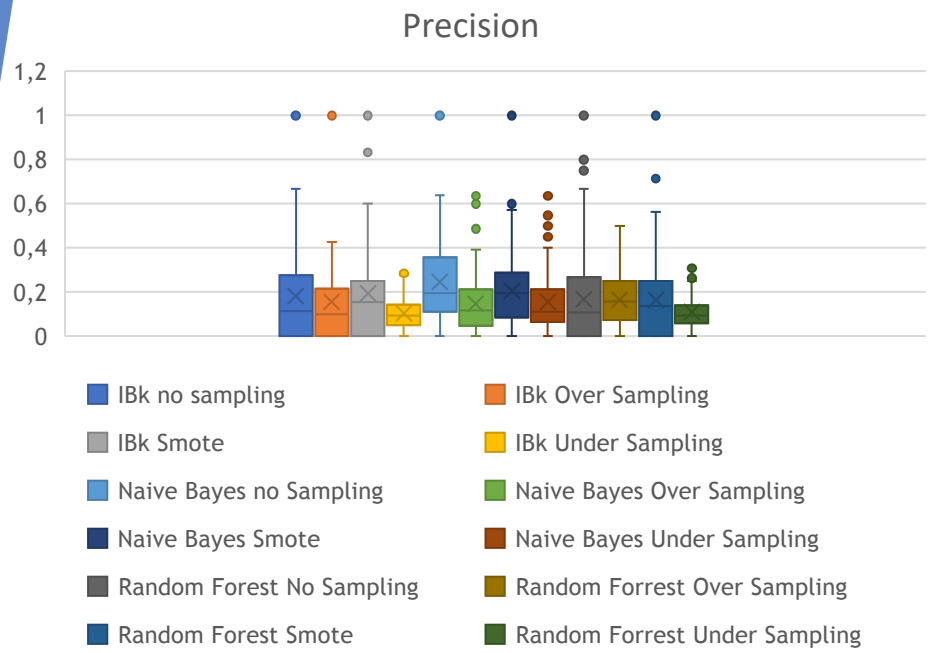
Sensitive learning

Le istanze vengono replicate tante volte quant'è il loro costo

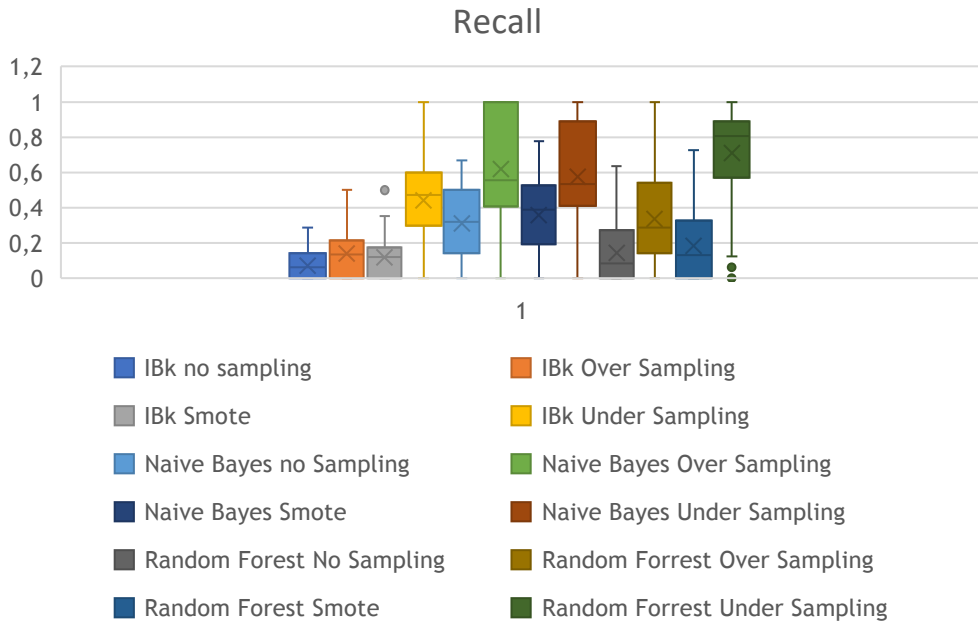
Sensitive threshold

La soglia sulla probabilità di appartenere a una classe invece che essere lo standard 0. 5 viene calcolata come CFP/(CFP+CFN), ovvero 0. 09

Sampling on Avro 1

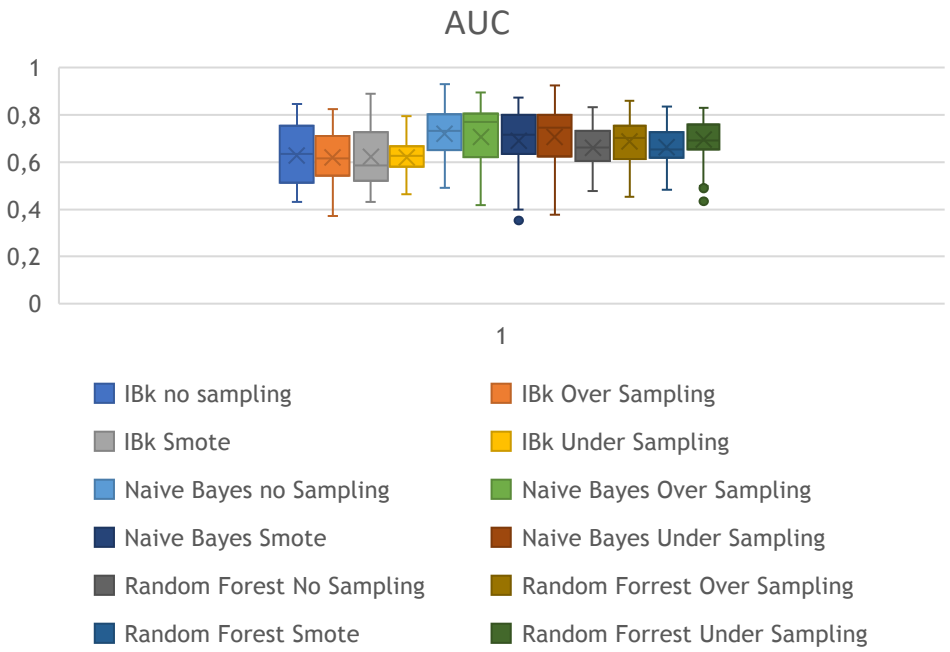


In ogni caso, eccetto che per Random Forest con Over Sampling e Smote, la precision in media diminuisce.

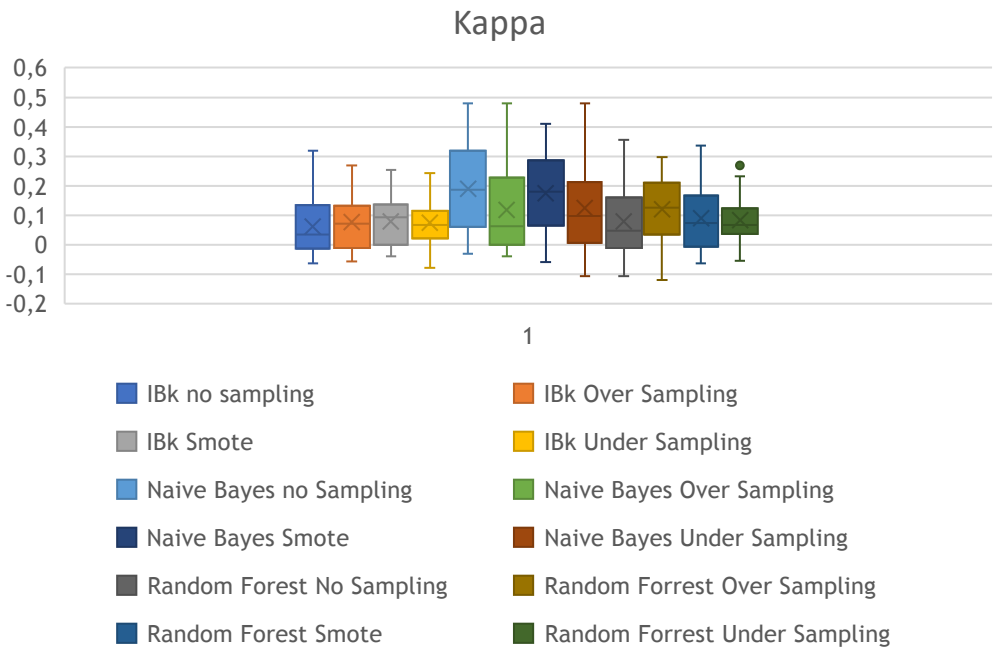


Il sampling aiuta fortemente la recall su ognuno dei classificatori, per qualsiasi tecnica di sampling, da questo punto di vista la recall migliore si ottiene in media con un Random Forest Under Sampling, che ha anche però una grande varianza

Sampling on Avro 2

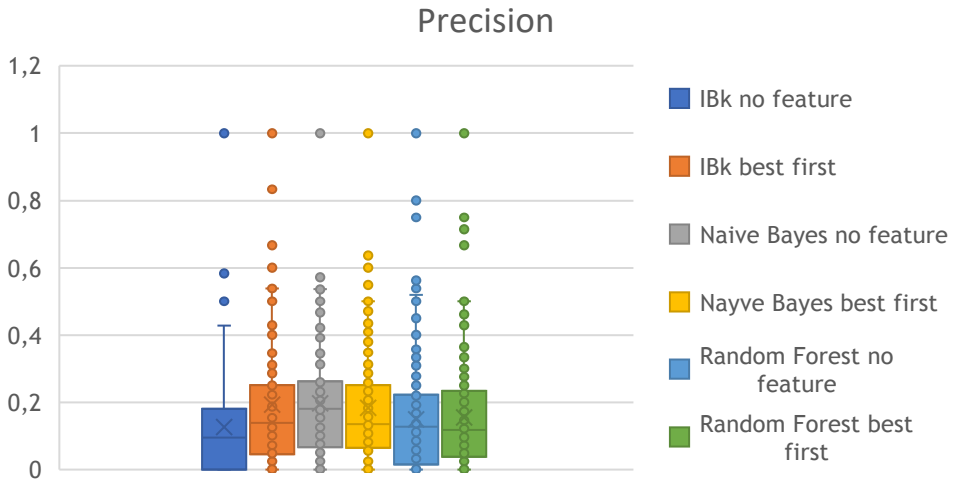


Si riscontra che per tutti i classificatori usare tecniche di sampling influisce lievemente rispetto al parametro di AUC, che può sia aumentare che diminuire in modo lieve

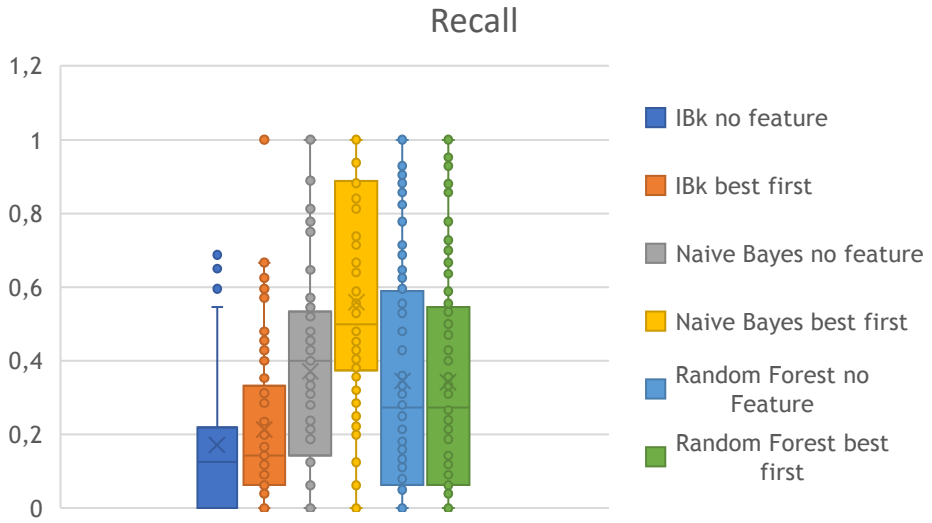


L'utilizzo della tecnica di Under Sampling sia per IBk che per Random Forest porta a un miglioramento del kappa medio e una diminuzione della sua varianza, mentre non vale lo stesso per Naive Bayes, che peggiora con l'utilizzo di queste tecniche

Feature Selection on Avro 1

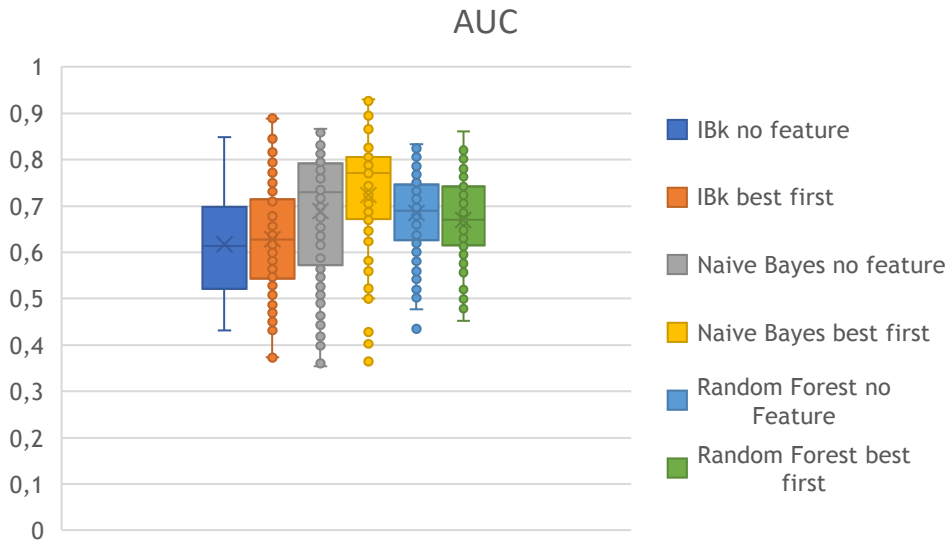


L'utilizzo della feature selection tende a migliorare la precision solo per IBk, mentre non porta grandi variazioni negli altri casi

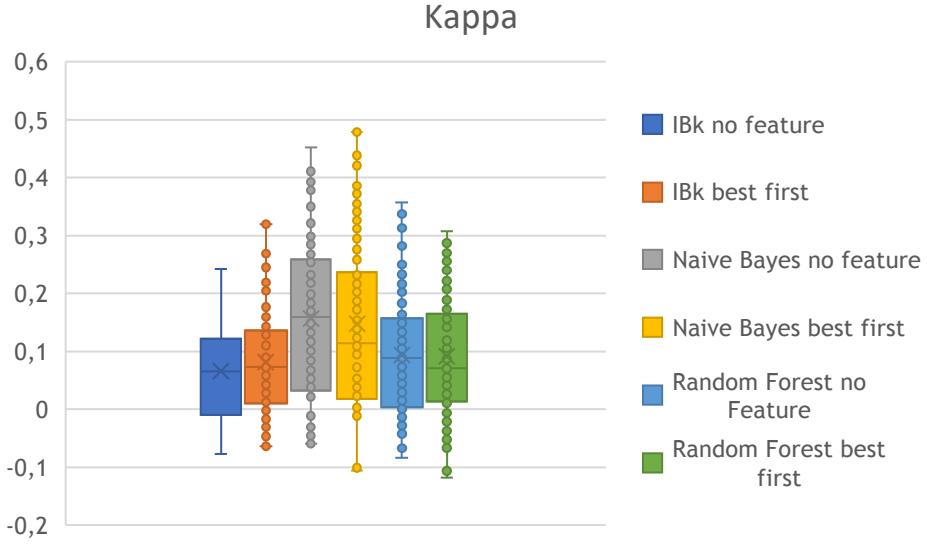


Effettuare feature selection su Naive Bayes muove verso valori tendenti ad 1 la distribuzione della sua recall, anche se la media non aumenta della stessa quantità, mentre sia Random Forest sia IBk non subiscono grandi variazioni dall'utilizzo di questa tecnica

Feature Selection on Avro 2

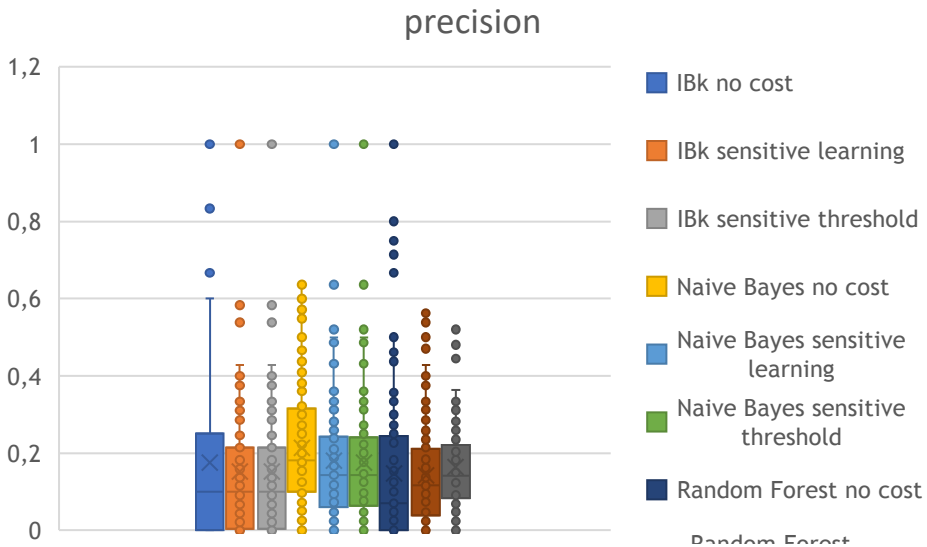


Rispetto agli altri due parametri precedentemente citati, usare feature selection su Naive Bayes porta un aumento in media dell'AUC, mentre non porta alcun miglioramento per quanto riguarda IBk e Random Forest

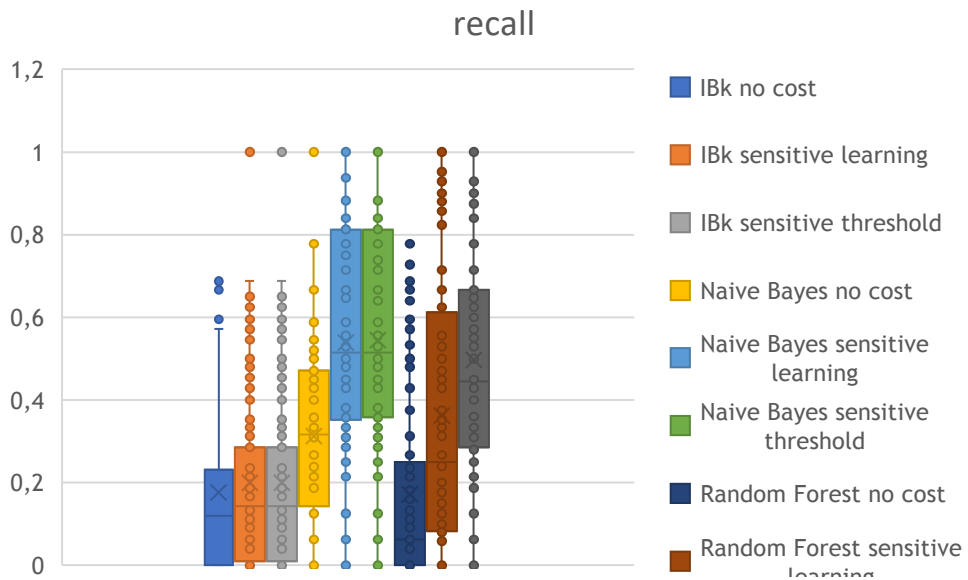


Il valore di kappa non aumenta per nessun classificatore, con l'aggiunta di questo filtro, anzi per Naive Bayes addirittura peggiora

Cost Sensitivity on Avro 1

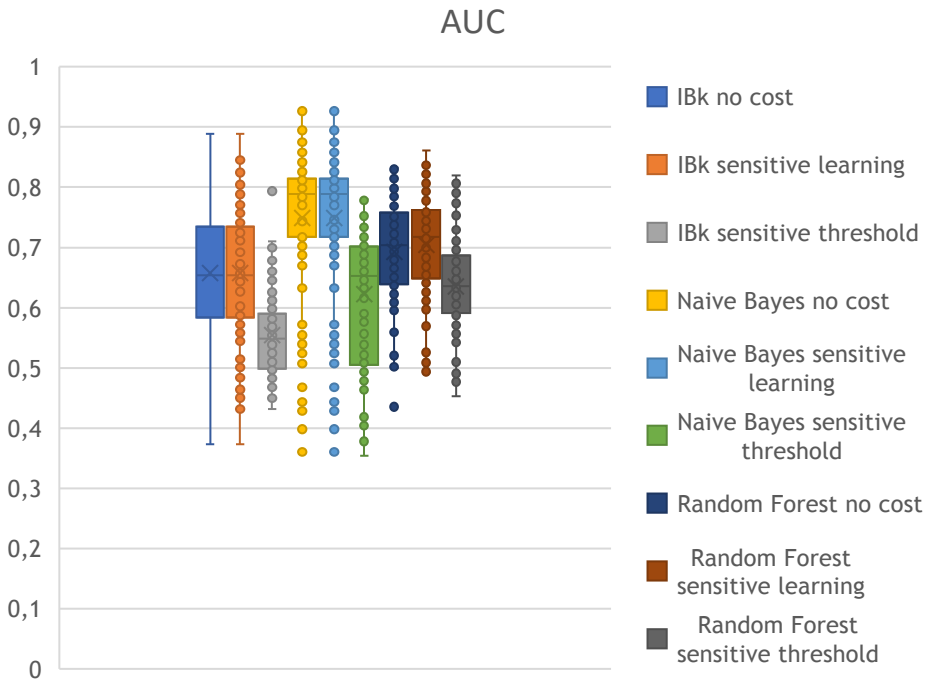


Le due tecniche di cost sensitivity producono il medesimo risultato sia su IBk che su Random Forest, ovvero fanno diminuire la varianza della precision anche se mantenendo la stessa media. Cosa diversa accade a Naive Bayes che al posto di migliorare peggiora

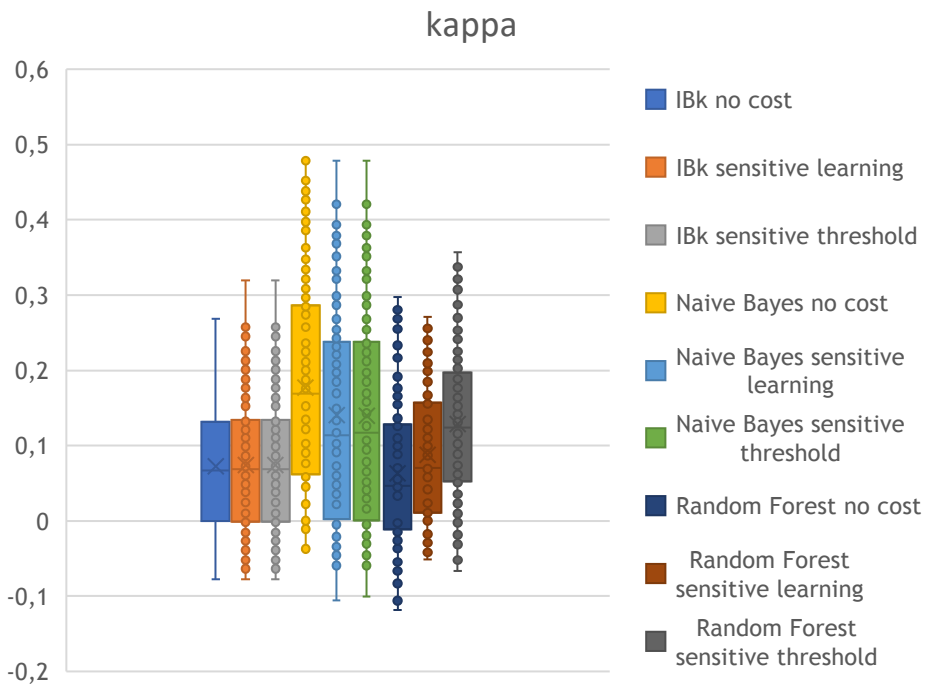


Per quanto riguarda la metrica della recall, l'utilizzo di tecniche di cost sensitivity la fa aumentare per tutti i classificatori e in certi casi anche di molto

Cost Sensitivity on Avro 2

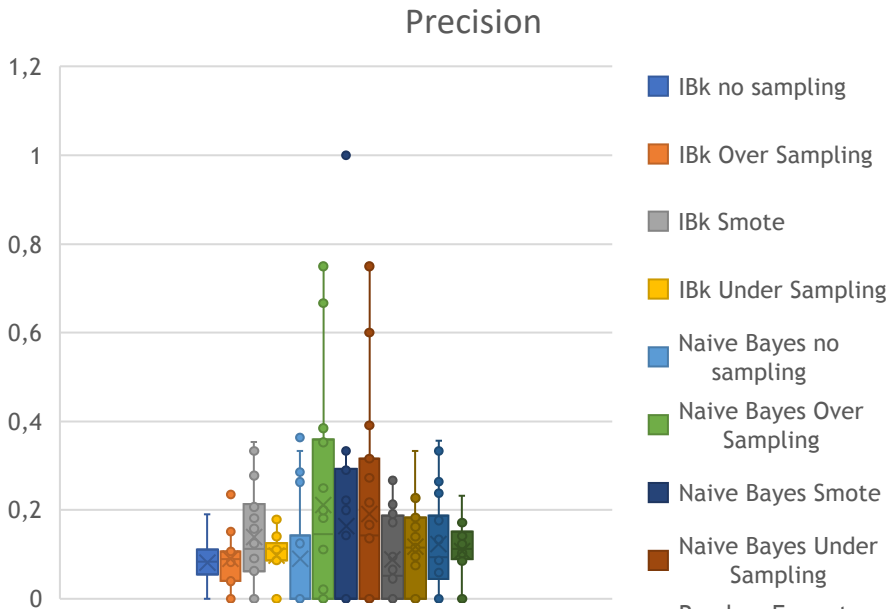


Nel caso del AUC, l'utilizzo di sensitive learning non fa variare per nulla o al massimo di poco questa metrica, mentre utilizzo della tecnica di sensitive threshold la peggiora

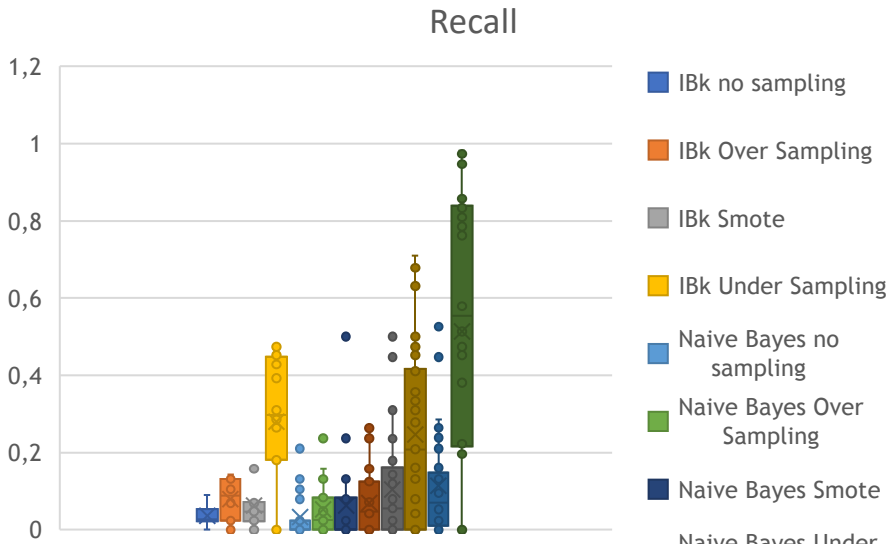


Per IBk l'utilizzo di tecniche di cost sensitivity non fa variare la media dei valori di kappa mentre per Naive Bayes la peggiora e per Random Forest la migliora

Sampling on Bookkeeper 1

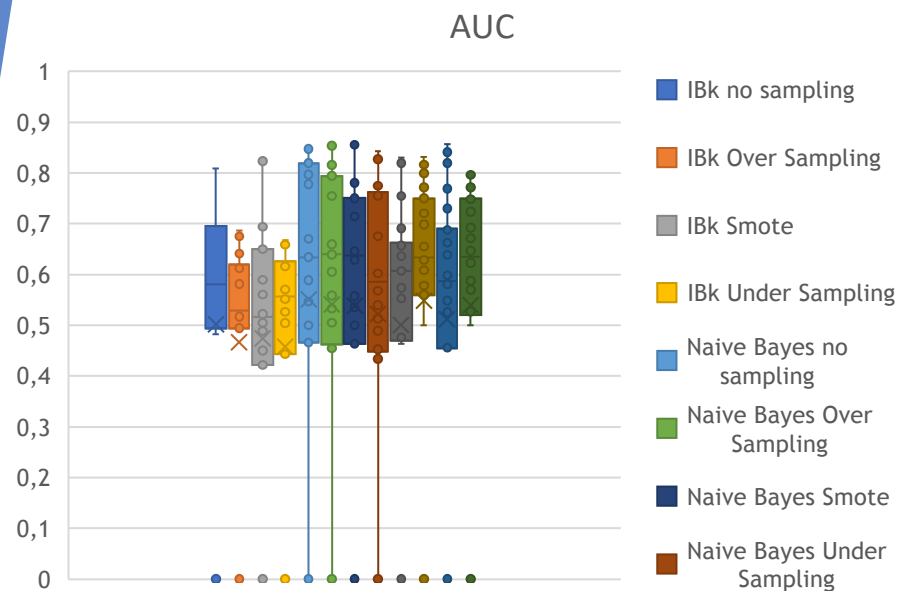


Su IBk, la tecnica di Smote in media riesce a comportarsi meglio delle altre.
Per Naive Bayes tutte e tre le tecniche di sampling hanno lo stesso impatto, e migliorano la precisione media ma aumentano la variabilità di essa.
Per nessuna tecnica di sampling cambia la precisione di Random Forest

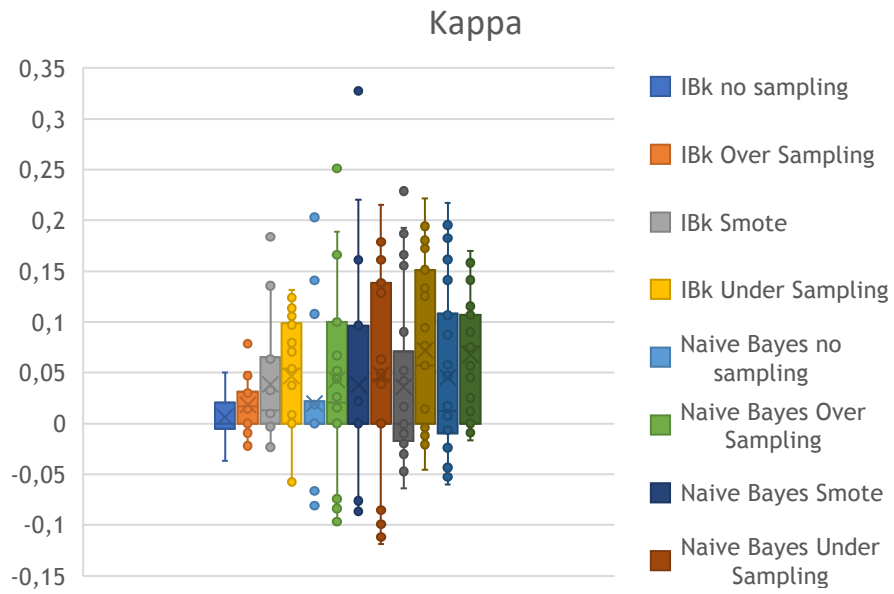


La recall per ogni tecnica di sampling aumenta, in particolare cresce per tutti e tre i classificatori, nel miglior modo con l'Under sampling

Sampling on Bookkeeper 2

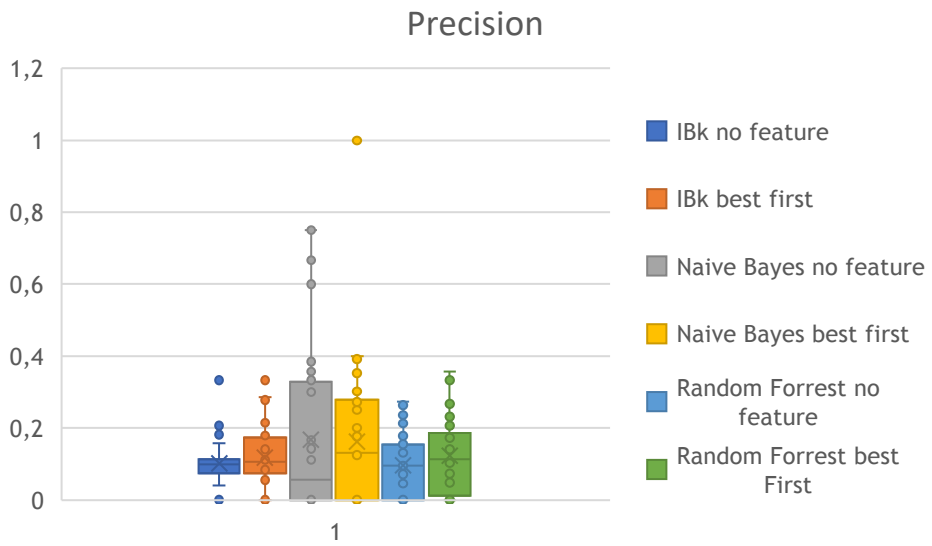


Le tecniche di sampling peggiorano tutte l'AUC dei classificatori tranne che per Random Forest per cui sia la tecnica di under Sampling che quella di Over Sampling portano un aumento del parametro

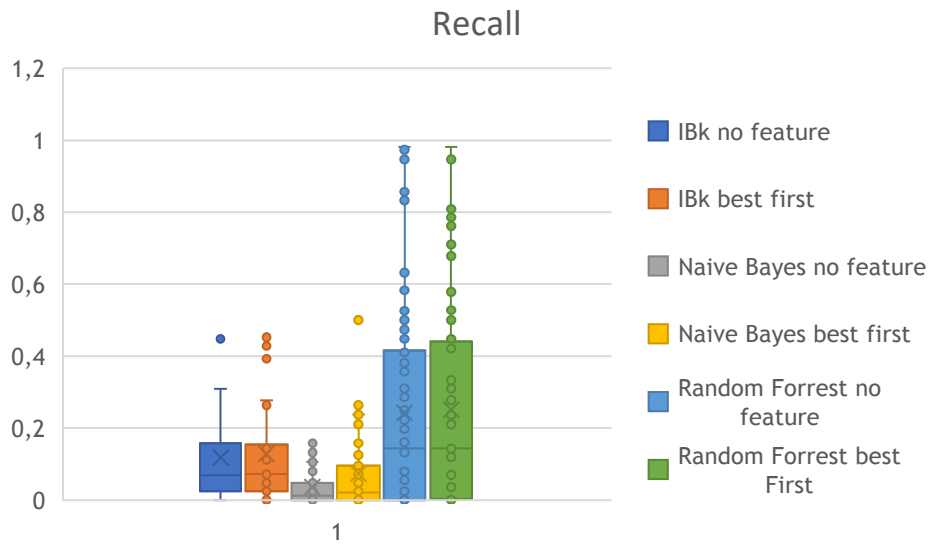


Per tutti i classificatori vi è un aumento del kappa, in particolare, per IBk e Naive Bayes la tecnica migliore è Under Sampling.
Per Random Forrest invece la migliore tecnica è Over sampling

Feature Selection on Bookkeeper 1

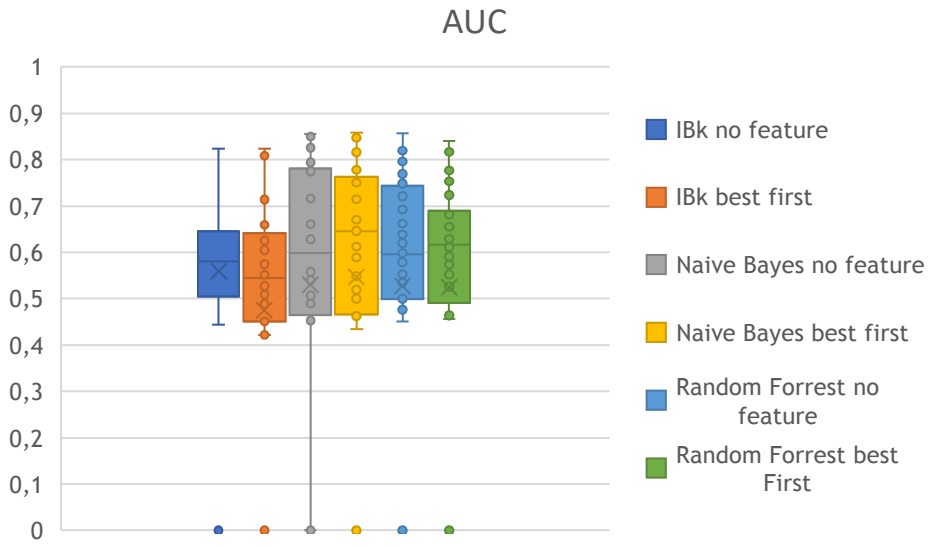


Effettuare feature selection porta un lieve aumento in media della precision per tutti e tre i classificatori

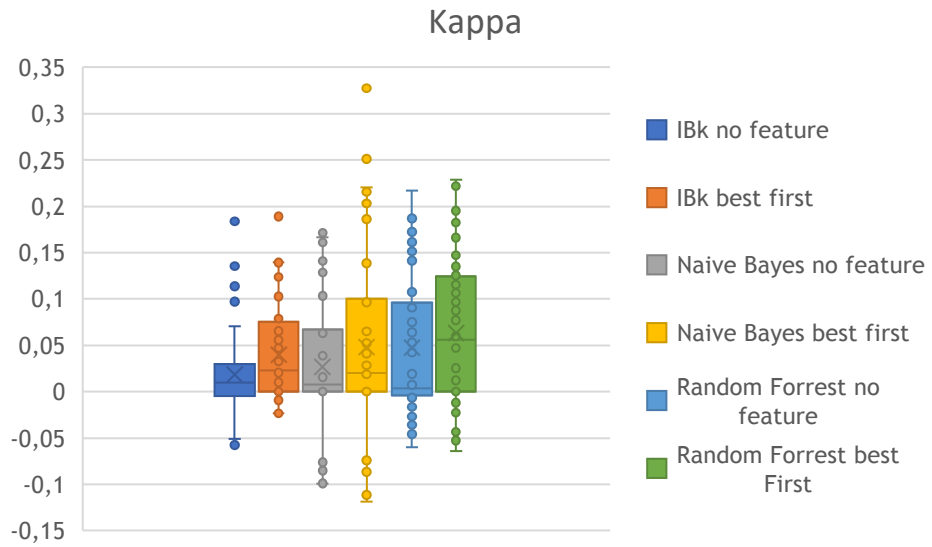


La recall per IBk e Random Forest rimane praticamente inalterata, mentre vi è un lieve aumento per Naive Bayes

Feature Selection on Bookkeper 2

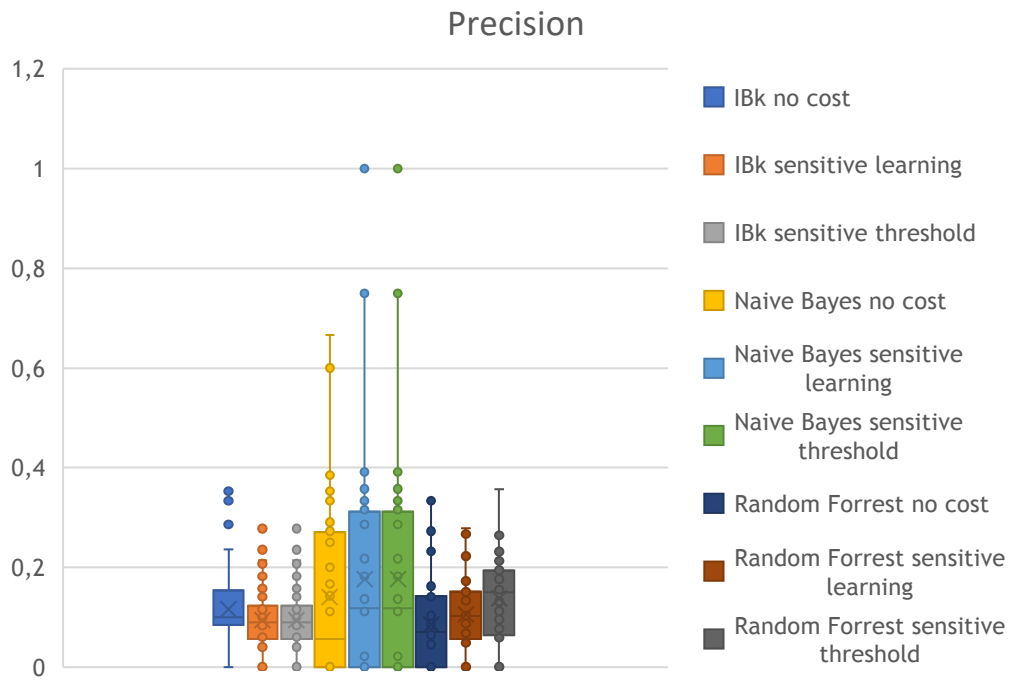


Per quanto riguarda l'AUC, applicare feature selection peggiora questa statistica per IBk, mentre per Random Forest e Naive Bayes migliora leggermente la media

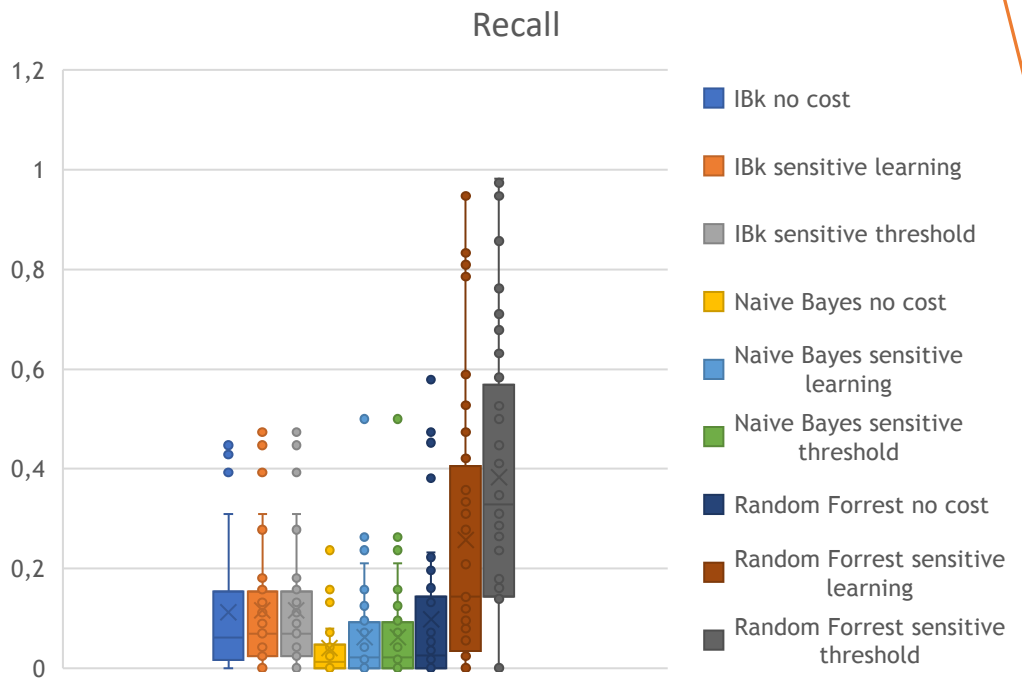


Il valore di kappa tende ad aumentare per tutti e tre i classificatori, in particolare Random Forest con Best First ottiene il miglior punteggio in media

Cost Sensitivity on Bookkeeper1

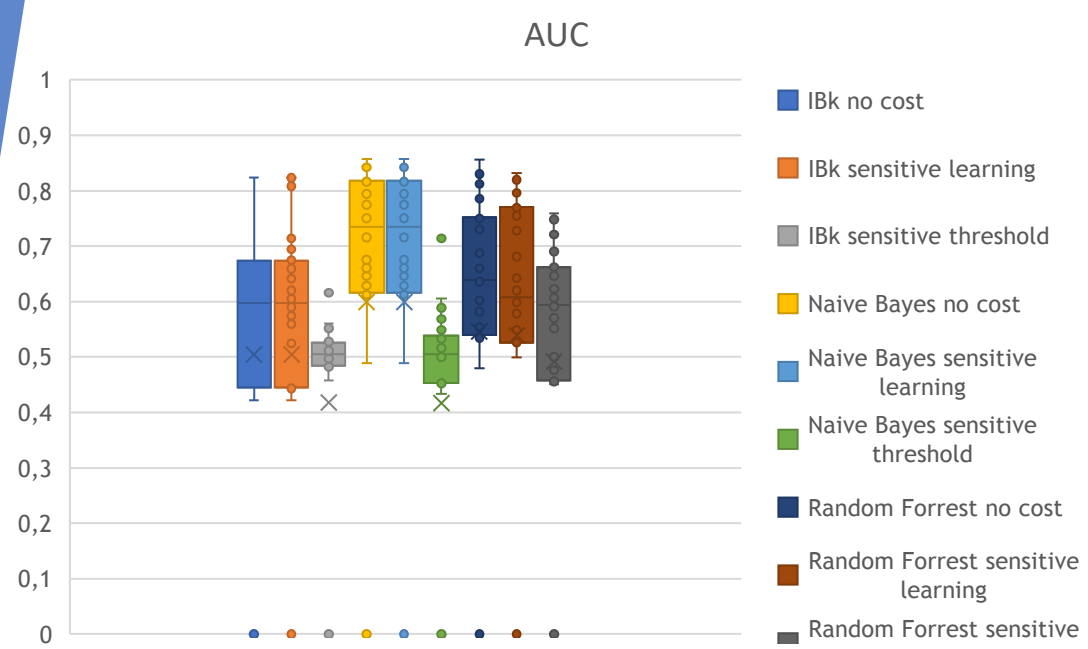


Applicazione delle tecniche di cost sensitivity aumenta in media la precision per quanto riguarda Naive Bayes e Random Forest, mentre l'utilizzo di esso per quanto riguarda IBk non porta ad alcun vantaggio

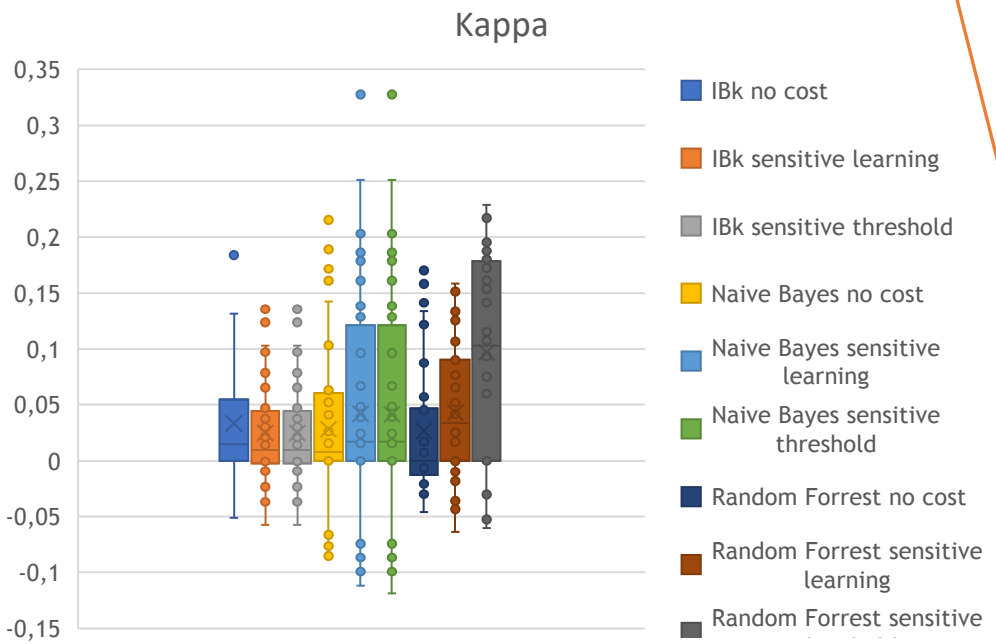


Per quanto riguarda la recall l'applicazione di esse non impatta i classificatore IBk e impatta lievemente Naive Bayes mentre per Random Forest vi è un grande impatto

Cost Sensitivity on Bookkeeper 2



L'utilizzo della tecnica di sensitive threshold fa diminuire notevolmente la distribuzione dell'AUC



Il parametro kappa sembra anche esso variare nello stesso modo in cui varia la precision per i tre classificatori

Conclusioni

- Nessun classificatore, con l'introduzione di uno o più filtri, migliora in maniera incisiva su tutti e quattro i parametri considerati.
- Per Avro quello che si può dire è, senza applicare nessun filtraggio dei dati, il miglior classificatore si conferma Naive Bayes, ma anche quello con più alta variabilità, mentre con l'introduzione di filtri, Naive Bayes sembra essere pareggiato dagli altri due classificatori.
- Per Bookkeeper la mia scelta ricadrebbe su Naive Bayes con under sampling, no feature selection e no cost sensitivity.

Grazie per l’attenzione!

Link

Il progetto realizzato è disponibile su GitHub e controllabile su SonarCloud:

Link a Sonarcloud

Link a GitHub