Studio empirico finalizzato a misurare l'effetto di tecniche di sampling sull'accuratezza dei modelli predittivi.

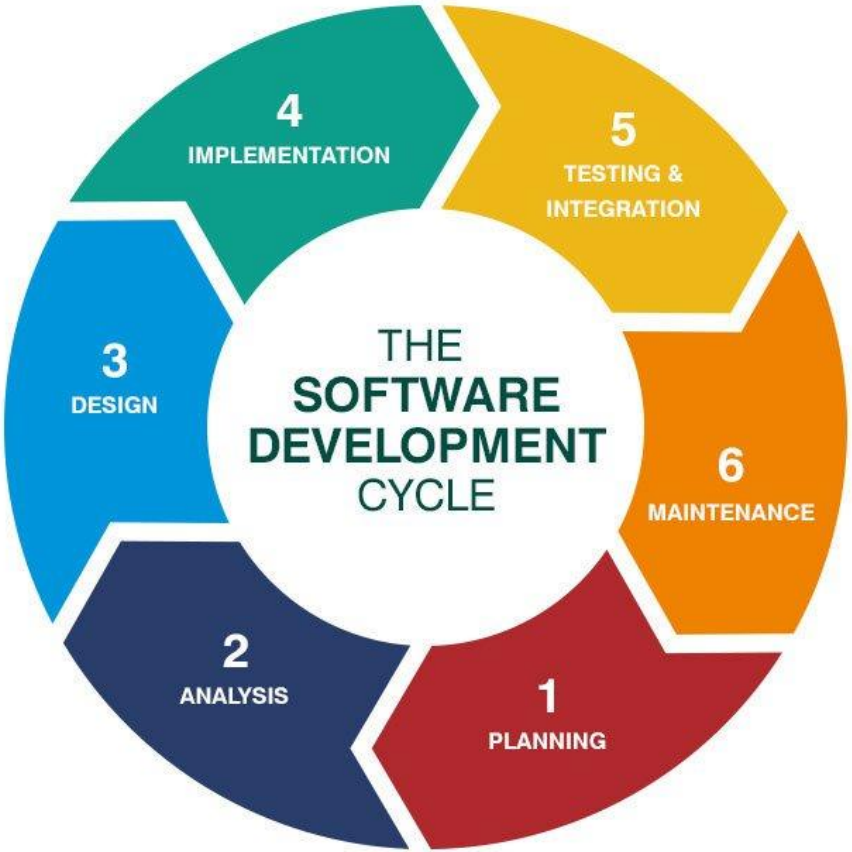
Introduzione di due nuove metriche e analisi della precisione condizionata dallo snoring.

Luca Fiscariello
0318266

Roadmap

- Introduzione
- Metodologia
- Analisi risultati
- Conclusioni

Introduzione : Contesto



In un contesto di sviluppo software la fase di test di un prodotto potrebbe essere molto onerosa.

Difficile individuare le entità da testare.

Si vorrebbe poter ottimizzare i tempi e avere uno strumento che indichi quali classi più probabilmente sono affette da bug.

Applicazione Machine Learning

Introduzione : problemi e obiettivi

Attraverso il machine learning è possibile costruire un modello predittivo attraverso una fase di training e una di test.

L'addestramento del modello avviene tramite dati raccolti in vario modo. Tuttavia la creazione del dataset presenta varie insidie.

Problemi

- Difficile individuare le metriche utili all'addestramento del modello
- I dataset creati possono essere fortemente sbilanciati
- Qualità dei dati impatta l'accuratezza del predittore

Obiettivi

- Introduzione di due metriche e valutazione della loro utilità
- Studio dell'effetto delle tecniche di sampling sui modelli predittivi
- Quantificare l'errore di predizione indotto dallo snoring.

Metodologia: creazione del dataset

Per la creazione del dataset sono stati incrociati dati provenienti da due fonti. I dati relativi alle classi sono stati prelevati da Git , quelli relativi alle classi difettose sono stati prelevati da Jira.

Raccolta dei dati è stata guidata da 11 metriche :

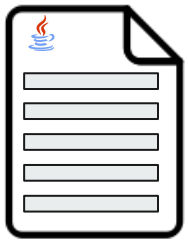
- Numero di autori di una classe
- LOC
- Churn
- Numero bug fixati per una classe
- Age
- Numero di revisioni
- Numero di file committati con la classe di interesse
- LOC touched : Numero di righe aggiunte+rimosse+modificate
- LOC added : Numero di righe aggiunte

- LOC medio dei **metodi di una classe**
- **Varianza** LOC touched

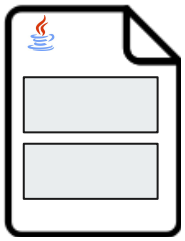
Metodologia: introduzione di due metriche

LOC medio dei metodi di una classe

Idea di base : Importante capire come le righe di codice di una classe sono organizzate . A parità di dimensione(LOC) possono verificarsi i seguenti casi:



Potrei avere una classe con tanti metodi, quindi i metodi sono di dimensione controllabile.



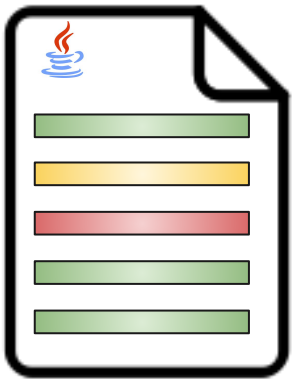
Potrei avere una classe con pochi metodi. La dimensione dei singoli metodi è considerevole.

- Si calcola quindi questa metrica effettuando il rapporto tra LOC e il numero di metodi contenuti nella classe.
- Lo svantaggio considerevole è che bisogna analizzare il sorgente di ogni file rendendo la raccolta delle metriche **MOLTO** lenta.

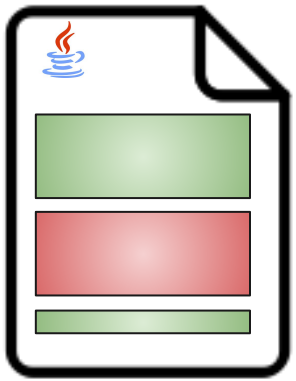
Metodologia: introduzione di due metriche

Varianza LOC touched

Idea: Non fermarsi ad analizzare i momenti del primo ordine calcolata su alcune metriche. La varianza calcolata su opportune metriche potrebbe essere indice di stabilità della classe.



Se la varianza delle linee di codice modificata è bassa, allora la classe potrebbe essere più stabile.



Se la varianza delle linee di codice modificata è alta, allora la classe è soggetta a più stravolgimenti.

Approccio non troppo pesante da un punto di vista computazionale

Metodologia: calcolo delle metriche

Come calcolare le metriche ?

Calcolo incrementale delle metriche

Perché questa scelta?

La scelta è condizionata dall'uso della metrica age. Se non si usasse un approccio incrementale si perderebbe la storia passata della classe. Lo stato passato della classe influenza quello futuro.

Esempio

- Linee aggiunte
- Numero di autori

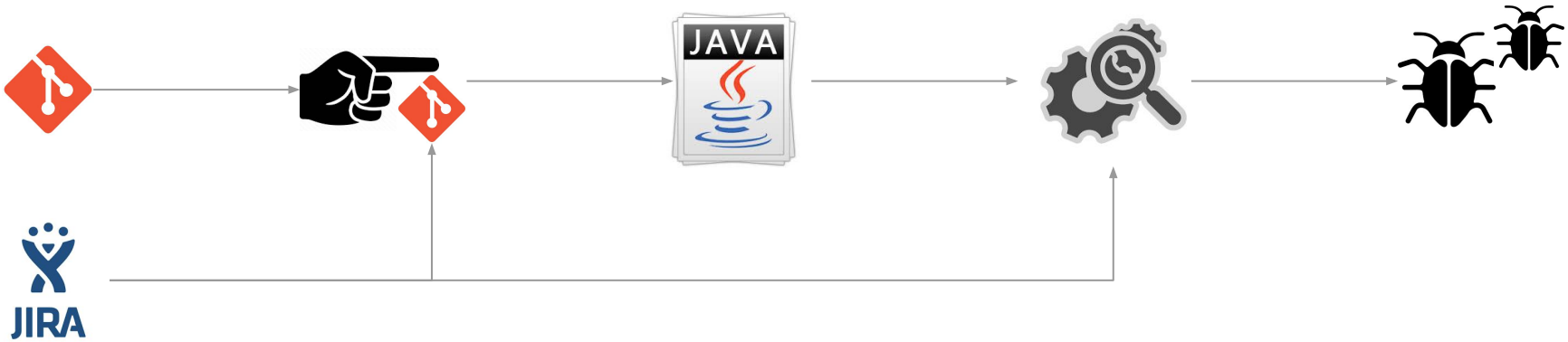
Svantaggio

Una classe appartenente a due release diverse **con le stesse metriche** potrebbe essere buggy in una release e no buggy nell'altra.

Metodologia: labeling

Il labeling è stato realizzato seguendo due diverse tecniche

1. SZZ + Rational
2. SZZ + Proportion



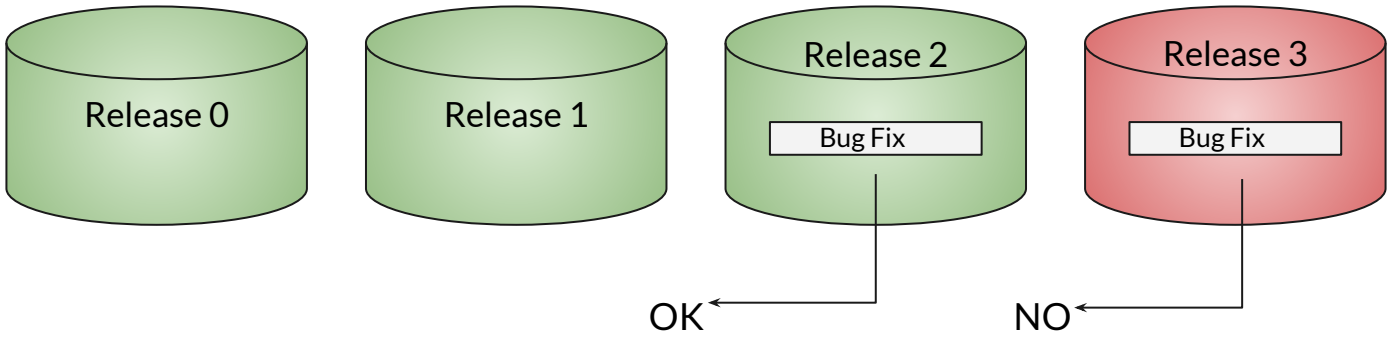
Di base è stato utilizzato Rational per individuare IV. Quando le informazioni relative all'IV non sono fornite allora è stato utilizzato Proportion. La versione di proportion realizzata è incremental training.

Metodologia: incremental training proportion

Il valore della proportion è calcolato in maniera incrementale. L'algoritmo riceve in input una release e calcola il valore con tutti i dati dalla release 0 fino a quella specificata. Eventuali bug che si potrebbero scoprire utilizzando release successive non vengono considerati.

Esempio

- Input algoritmo: Release 2
- Esecuzione : applicazione della formula per calcolare P considerando solo le prime 3 release



Metodologia: bilanciamento ed addestramento

Balancing

I dataset ottenuti sono fortemente sbilanciati con una dominanza in percentuali delle classi no buggy. Si è deciso quindi di bilanciare il dataset attraverso le tre diverse tecniche:

- Over Sampling
- Under Sampling
- SMOTE

Queste tecniche non sono state implementate da 0, ma sono stati utilizzati i filtri messi a disposizione da Weka.

Addestramento

Il modello è stato validato attraverso walk-forward validation. Progressivamente è stato incrementato l'insieme delle release utilizzate per il training del modello. Per il test sono state utilizzate le due release immediatamente successive all'ultima release di training

Analisi risultati : metriche ideate Avro

Per valutare l'effettiva utilità delle metriche sono stati creati 4 modelli predittivi addestrati in maniera differente:

1. Dataset privo delle due metriche ideate
2. Dataset contenente solo Loc metodo medio
3. Dataset contenente solo Varianza loc touched
4. Dataset con entrambe le metriche

METRICHE	AUC	KAPPA	RECALL	PRECISION
Nessuna	0.957236984062	0.586419951156	0.801302931596	0.495967741935
Loc methods	0.94340642616	0.50875248125	0.80456026058	0.40894039735
Var Loc touched	0.95266832266	0.55115612538	0.78827361563	0.45920303605
Entrambe	0.94134391145	0.48546189910	0.75570032573	0.39658119658

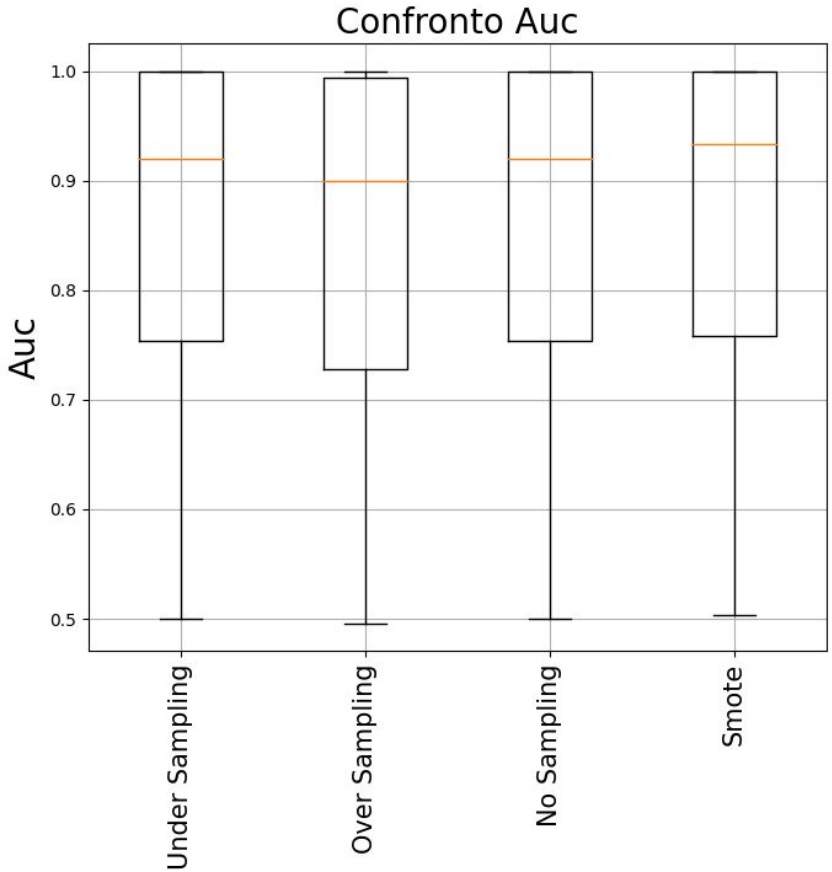
Analisi risultati : metriche ideate Bookkeeper

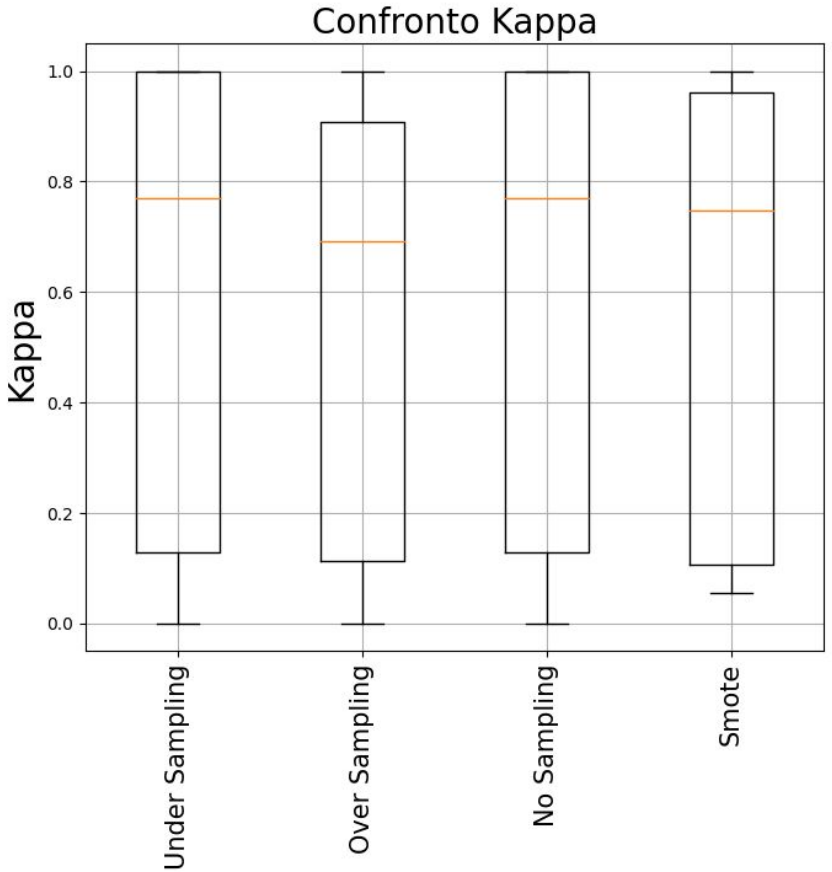
Per valutare l'effettiva utilità delle metriche sono stati creati 4 modelli predittivi addestrati in maniera differente:

1. Dataset privo delle due metriche ideate
2. Dataset contenente solo Loc metodo medio
3. Dataset contenente solo Varianza loc touched
4. Dataset con entrambe le metriche

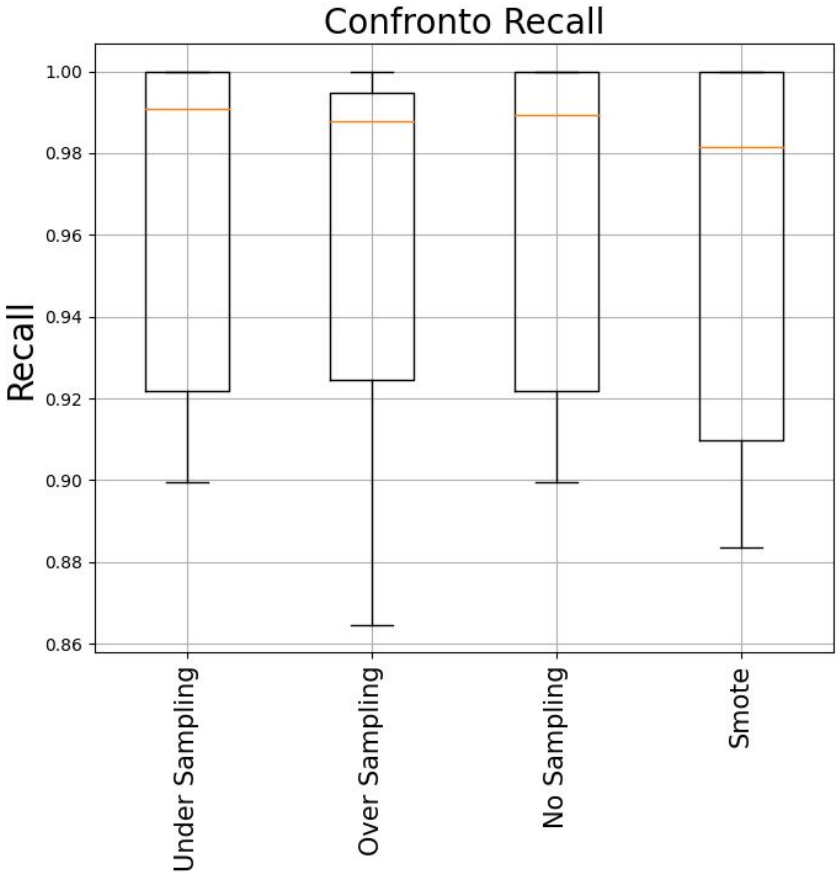
METRICHE	AUC	KAPPA	RECALL	PRECISION
Nessuna	0.922936404343	0.324019014007	0.382113821138	0.338942307692
Loc methods	0.92027345532	0.31636068484	0.37127371273	0.33414634146
Var Loc touched	0.92171863859	0.37572647607	0.49051490514	0.35420743639
Entrambe	0.91940687689	0.35991432739	0.46070460704	0.34693877551

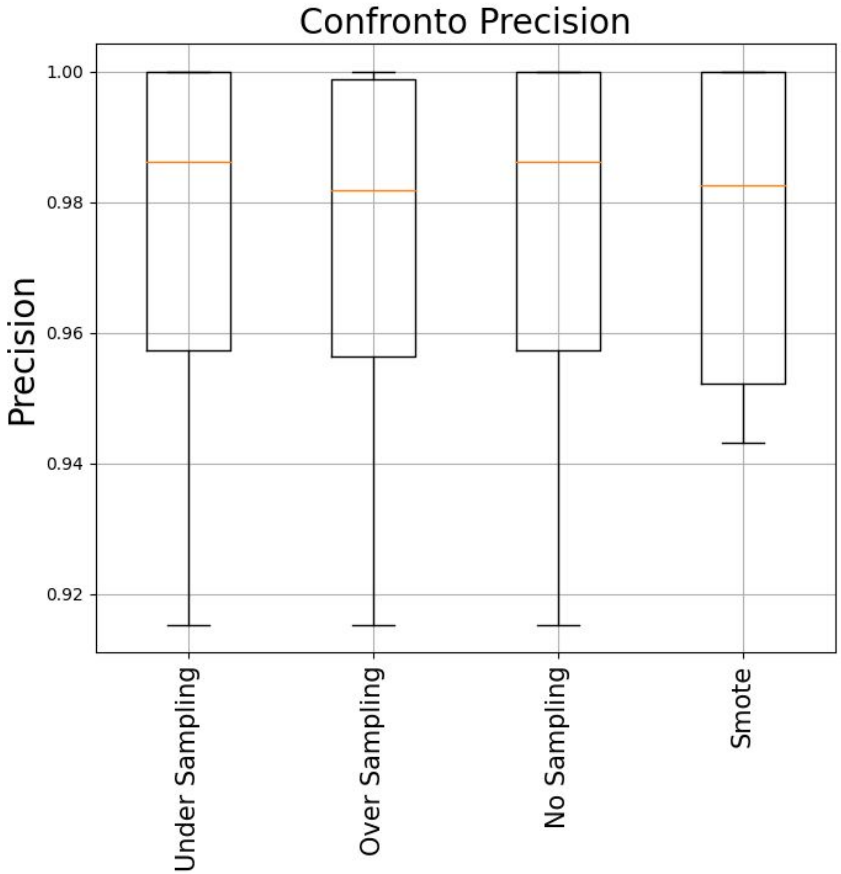
Analisi risultati : sampling AVRO



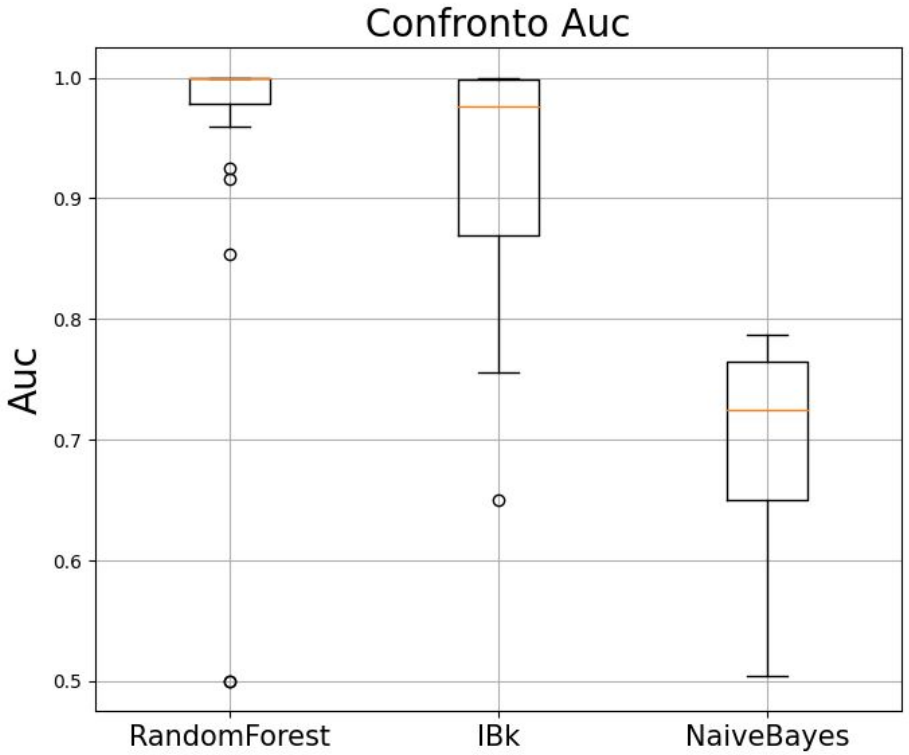


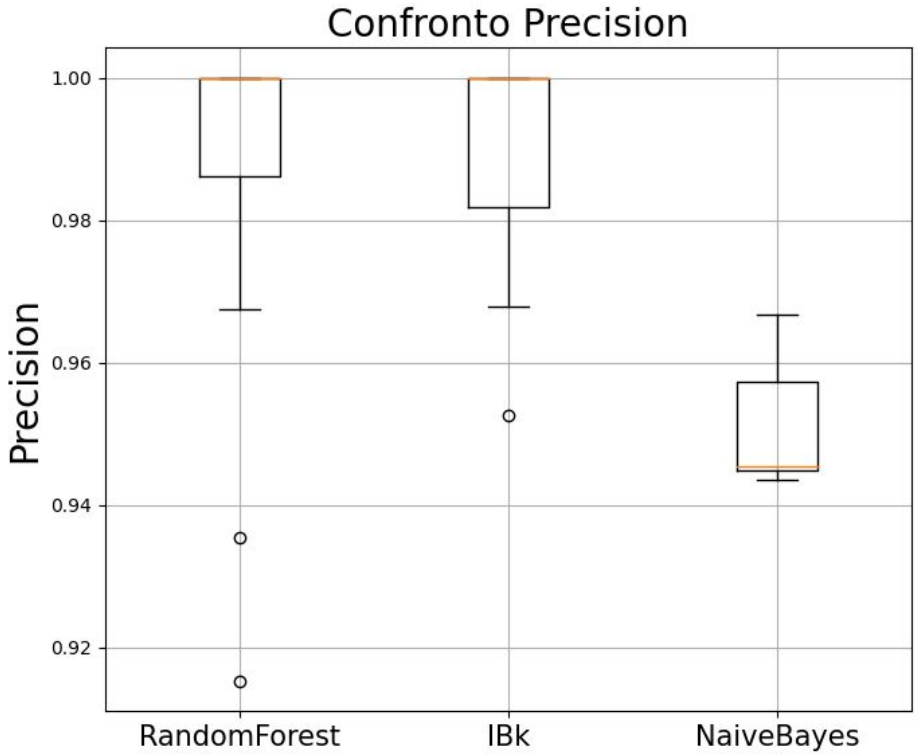
Analisi risultati : sampling AVRO



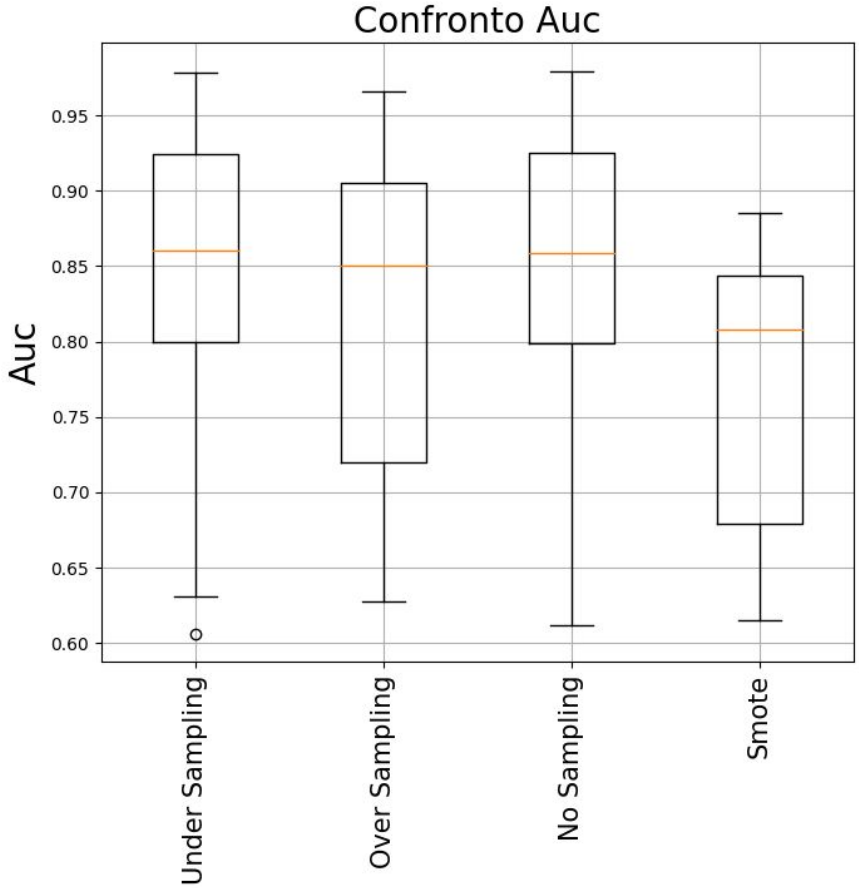


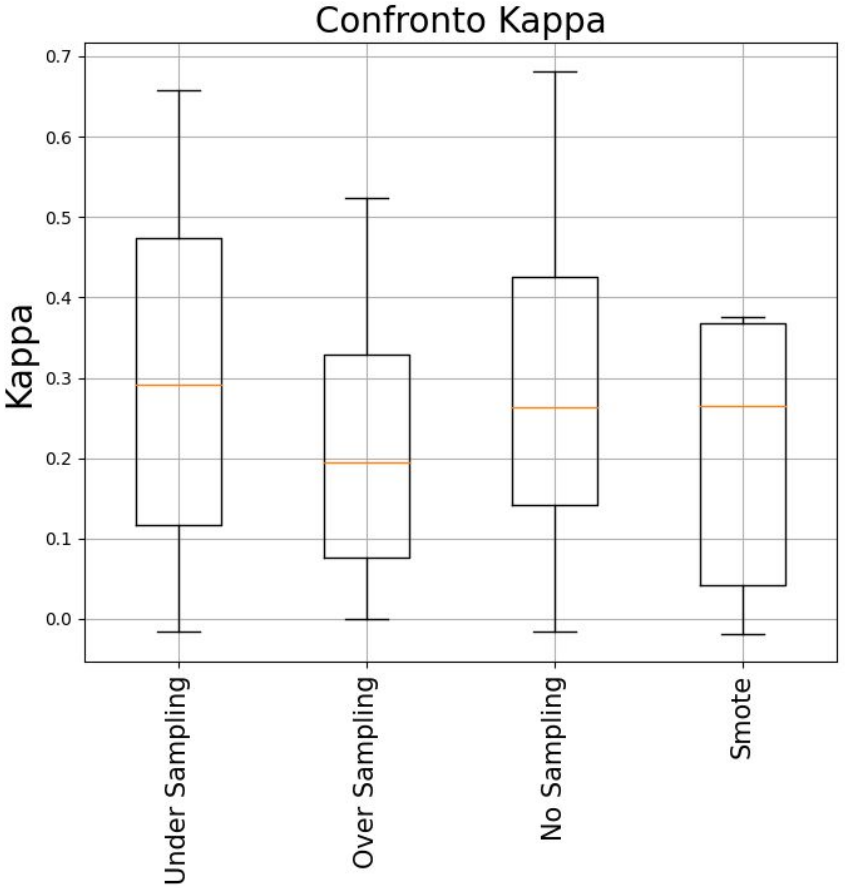
Analisi risultati : classificatore AVRO



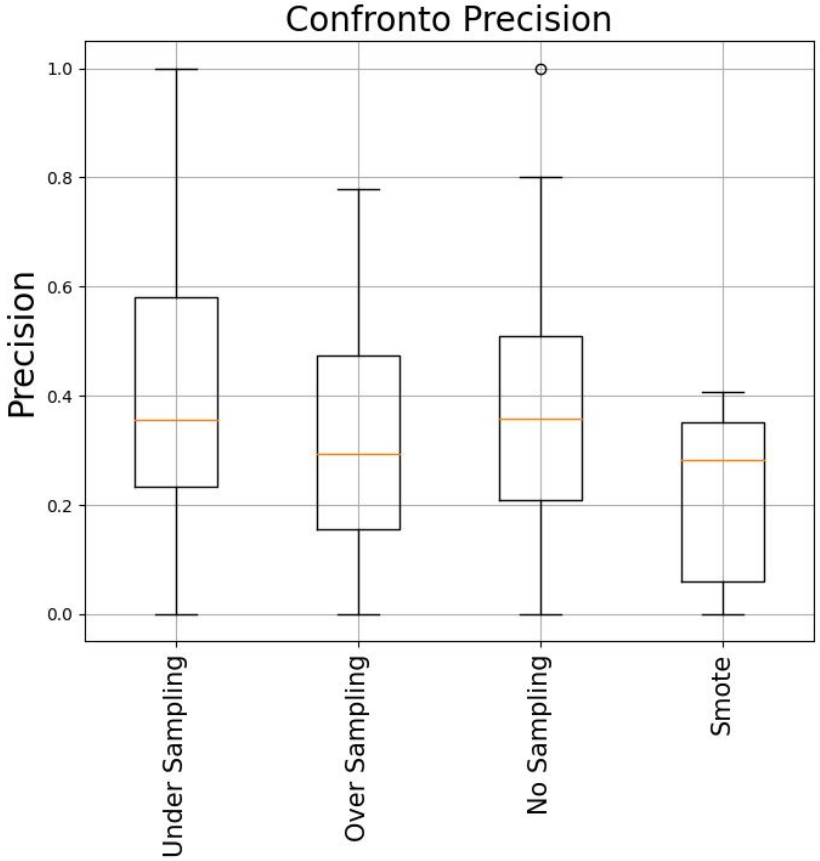


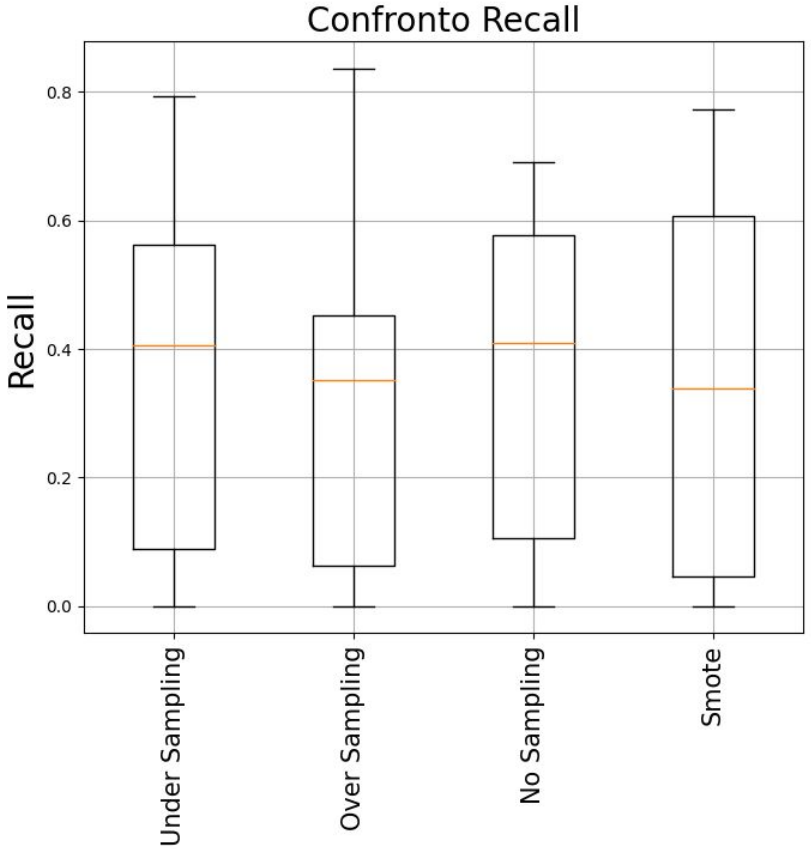
Analisi risultati : sampling BOOKKEEPER



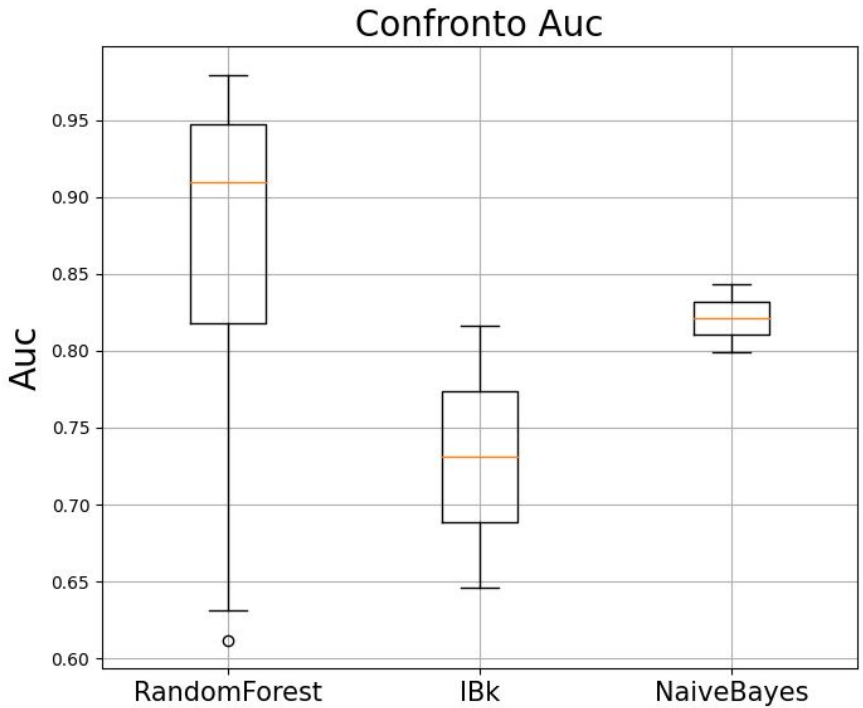


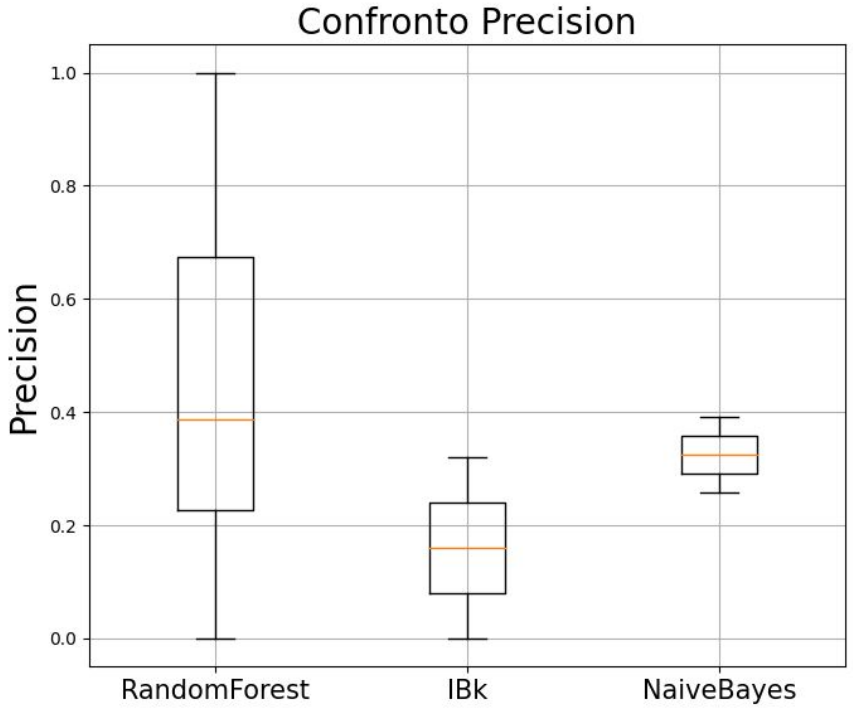
Analisi risultati : sampling BOOKKEEPER





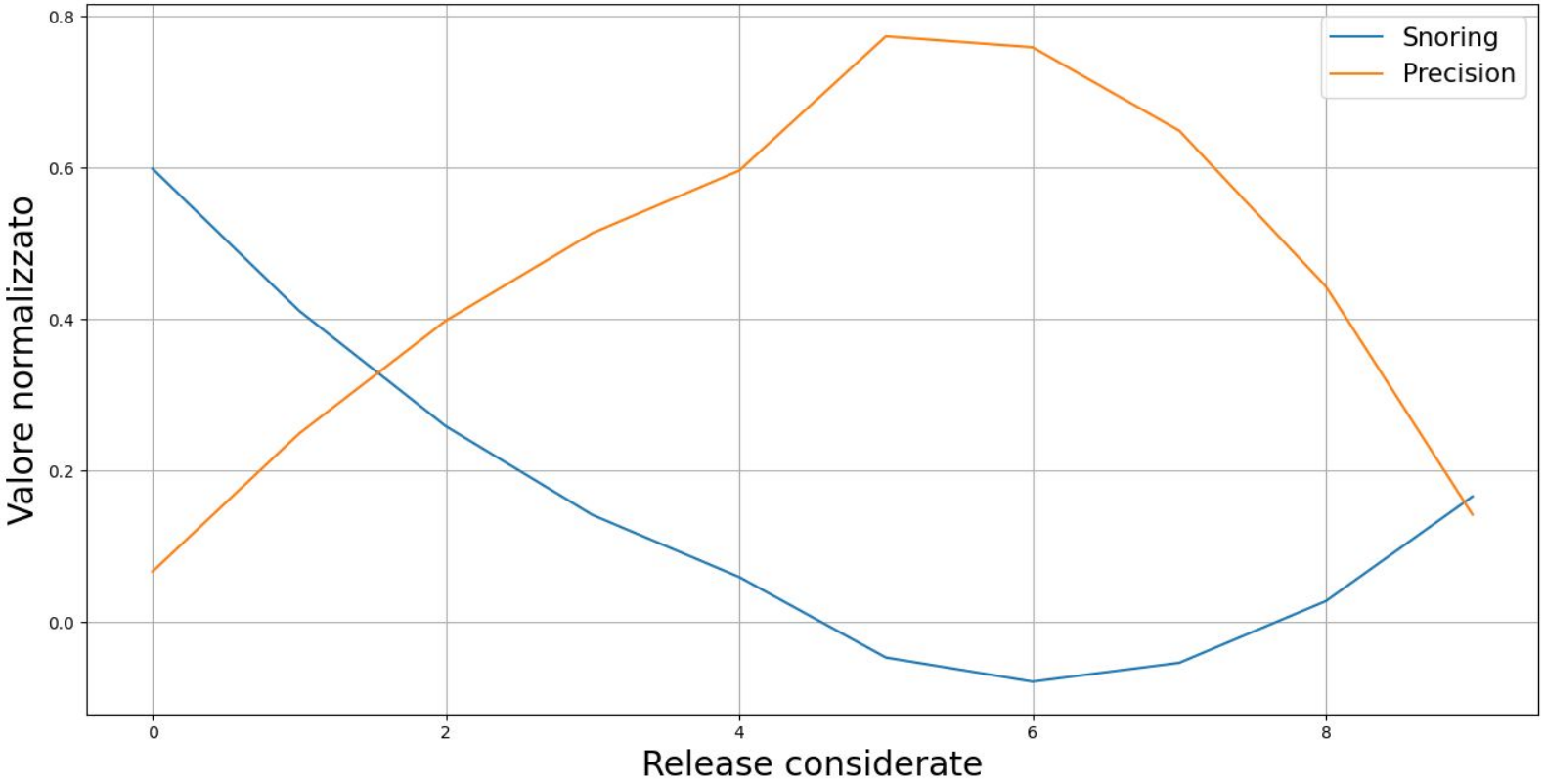
Analisi risultati : classificatore BOOKKEEPER



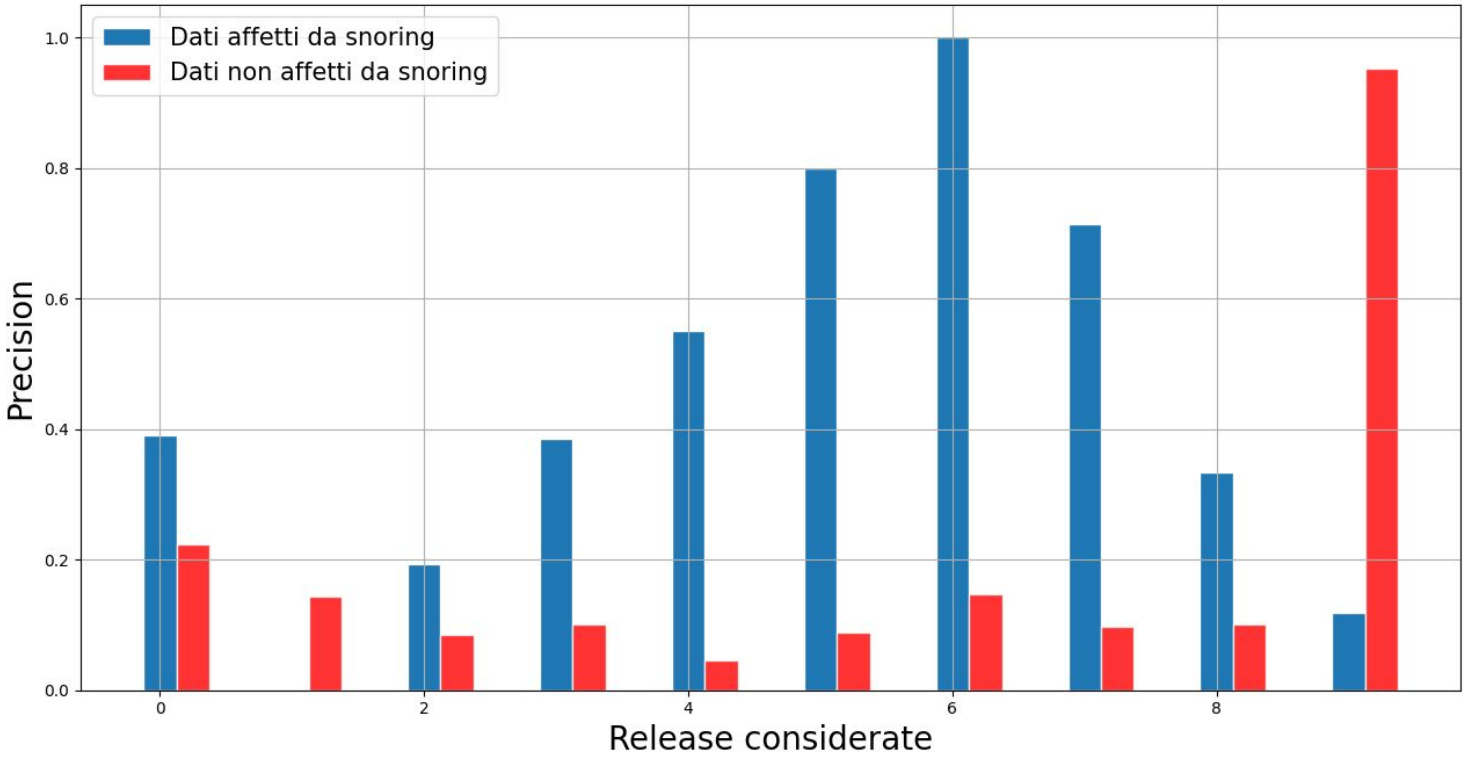


Analisi risultati : impatto snoring (1/3)

Check sui dati : all'aumentare del numero di release considerate la precision dovrebbe aumentare, il numero di classi affette da snoring dovrebbe diminuire.



Analisi risultati : impatto snoring(2/3)



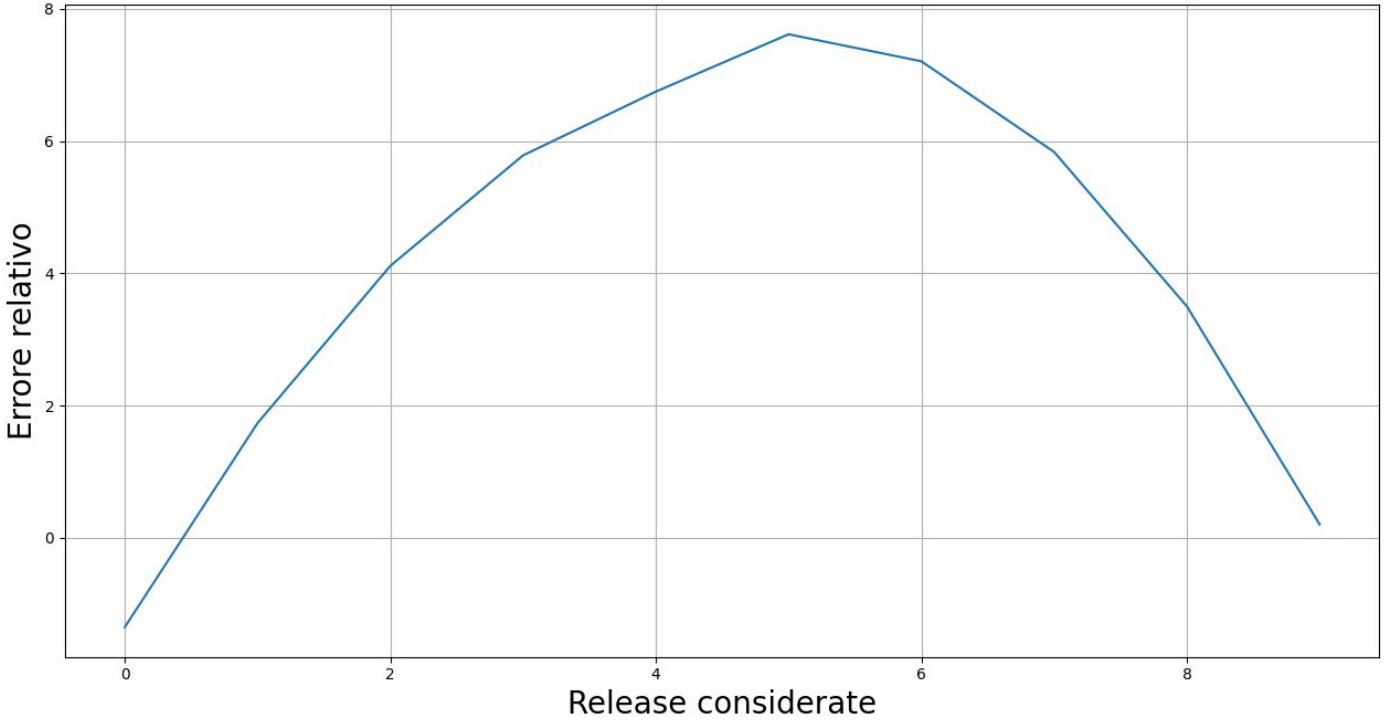
Sembrerebbe che con i dati affetti da snoring si riesca a costruire un predittore con precisione maggiore. Possibile una situazione del genere ?

Analisi risultati : impatto snoring(2/3)

A cosa è dovuto questo risultato ?

Una possibile causa è da rintracciare nel dataset fortemente sbilanciato. Con dataset affetto da snoring è più semplice fare predizioni. Con dataset non affetto da snoring aumentano le classi buggy e la predizione si complica.

Analisi risultati : impatto snoring(3/3)



Il massimo errore che si commette addestrando un modello con dati affetti da snoring è pari circa al 700%.

Conclusioni

- Le metriche ideate hanno mostrato un effettiva utilità solo in bookkeeper. In Avro i risultati non sono stati molto incoraggianti.
- Per ottenere prestazioni migliori è consigliato utilizzare per entrambi i progetti la combinazione **No sampling-Random forest**
- Addestrare un modello con dati affetti da snoring può comportare un errore relativo sulla precisione fino a un 700%