Machine Learning for Software Engineering

Marco Ferri
0299859

Università degli studi di
Roma Tor Vergata

Agenda

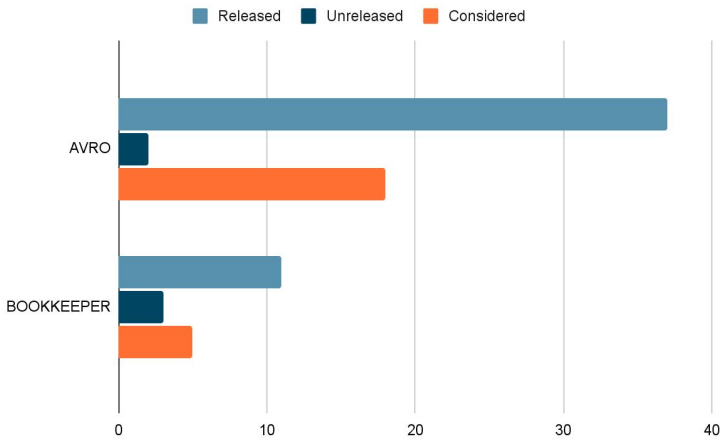
- Introduzione
- Scelte implementative
- Risultati
- Minacce alla validità
- Conclusioni e Links

Introduzione

- **CONTESTO:**
  - L' avvento del **Machine Learning** ha introdotto innumerevoli innovazioni negli ambiti più svariati.
  - Nell' ambito dell'**Ingegneria del Software** un aspetto fondamentale è quello di ridurre il **costo** della fase di **Software Testing** in termini economici e di tempo impiegato.
  - **E' possibile predire quali classi di un sistema hanno maggior possibilità di essere Buggy per concentrare la fase di Testing su di esse?**
- **SCOPO:**
  - Effettuare uno studio empirico che si pone l'obiettivo di valutare l'accuratezza di **Classificatori** applicando tecniche di **Feature Selection**, **Sampling** e **Cost Sensitive Classification** al fine di **localizzare classi Buggy** in progetti Open Source.
  - Lo studio prende sotto esame due progetti **APACHE**:
    - **AVRO**
    - **BOOKKEEPER**

Scelte implementative - Release

- Le **Release** sono ottenute con la **RestAPI** di **Jira**.
- Esse vengono categorizzate in:
  - **Released**: tutte le **Release** che sono state rilasciate, ossia quelle che hanno una data di rilascio ottenibile da Jira o GitHub.
  - **Unreleased**: le **Release** che non sono state rilasciate, ossia di cui non è possibile ottenere una data di rilascio.
- Durante questa fase, è stata evidenziata un' **inconsistenza** tra i nomi delle **Release** su Jira e GitHub.
  - Per risolvere tale inconsistenza, è stato implementato un **adattatore** che traduce il nome della **Release** da Jira a GitHub.
- Per lo studio è stato considerato il primo 50% delle **Release** rilasciate per evitare effetti di **Snoring**.
- Il secondo 50% e le **Release** non rilasciate sono state mantenute per gestire eventuali riferimenti ad esse.

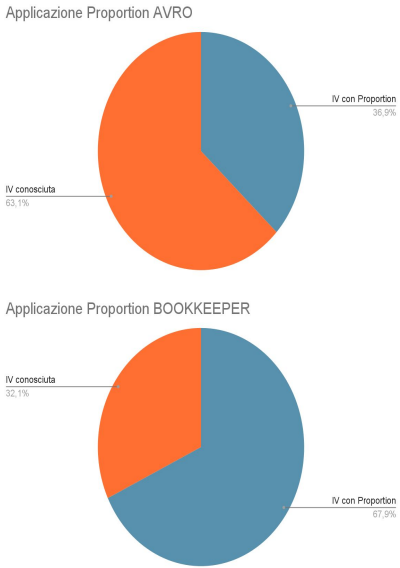


Scelte implementative - Bug

- I **Bug** sono ottenuti con la **RestAPI** di **Jira**.
  - Sono state esplicitamente richieste con l' API Issue di tipo "**Bug**" con risoluzione "**fixed**" e stato "**resolved**" o "**closed**".
- Per ogni **Bug** è stata considerata come:
  - **Fixed Version (FV)**: la Release temporalmente più recente delle Fixed Version restituite dall' API.
  - **Opening Version (OV)**: la Release immediatamente successiva alla data di creazione dell'Issue.
  - **Injected Version (IV)**: la Release temporalmente meno recente delle Affected Version restituite dall' API.
- Alla lista dei **Bug** viene applicato un **processo di sanificazione** prima e dopo aver applicato Proportion dove vengono scartati i **Bug** con:
  - **Inconsistenza dei dati**: IV > OV, IV > FV, OV > FV
  - **Nessun impatto su Proportion e classificazione Buggy**: IV=FV
  - **Bug senza commit relativi o classi Java modificate**: **solo** dopo aver applicato Proportion

Scelte implementative - Proportion

- Per ottenere la **Injected Version** dei Bug che ne sono privi viene utilizzata la tecnica di **Proportic**
- Tra le diverse varianti di **Proportion** è stata utilizzata **Moving Window**:
  - Rispetto a **Cold Start** vengono utilizzati dati passati del progetto sotto esame che sono sicuramente più rappresentativi rispetto a dati di altri progetti.
  - Rispetto ad **Incremental** potendo gestire la dimensione della "finestra" è possibile catturare in maniera migliore lo stato attuale del progetto rispetto ad uno stato troppo passato.
- La **dimensione della finestra** per Moving Window è stata scelta in base ad un compromesso tra la qualità e la quantità di Bug considerati. Tale valore corrisponde al **3%** dei Bug considerati.
  - La **qualità** ha rappresentato un problema perché in molti casi si è verificata la condizione OV=FV e l'impatto sul valore **P** è stato praticamente nullo.
  - La **quantità** ha rappresentato un problema per non considerare troppi Bug e ricadere nel caso della variante **Incremental**.
- **P** viene calcolato come la media dei rapporti **(FV - IV) / (FV - OV)** per i Bug che rientrano nella finestra.
- In un Bug la **IV incognita** viene calcolata come **FV - (FV-OV) * P** dove FV e OV sono Fixed Version ed Opening Version del Bug considerato.



Scelte implementative - Dataset

- Gli attributi del **Dataset** sono:
  - **Size**
  - **LOC Touched**
  - **Number of revisions (NR)**
  - **Number of defect fixes (NFix)**
  - **Number of authors (Nauth)**
  - **LOC added, Max LOC added, Avg LOC added**
  - **Churn, Max Churn, Avg Churn**
  - **Change Set Size, Max Change Set Size, Avg Change Set Size**
  - **Age**
  - **Weighted Age**

Scelte implementative - Dataset(1)

- La maggior parte delle metriche sono state calcolate con i comandi "**Git**" eseguiti come processi.
- La size è stata calcolata con "**Tokei**", un sistema Open Source con Repository su GitHub.
- Una classe è definita **Buggy** se esiste un commit relativo ad un Bug che agisce su di essa.
  - Una classe è definita Buggy da **IV inclusa** a **FV esclusa**.
- Problema di **inconsistenza** tra i nomi delle Issue "Bug" ed il commento dei Commit Git.
  - Soluzione tramite utilizzo di **espressioni regolari RegEx**.

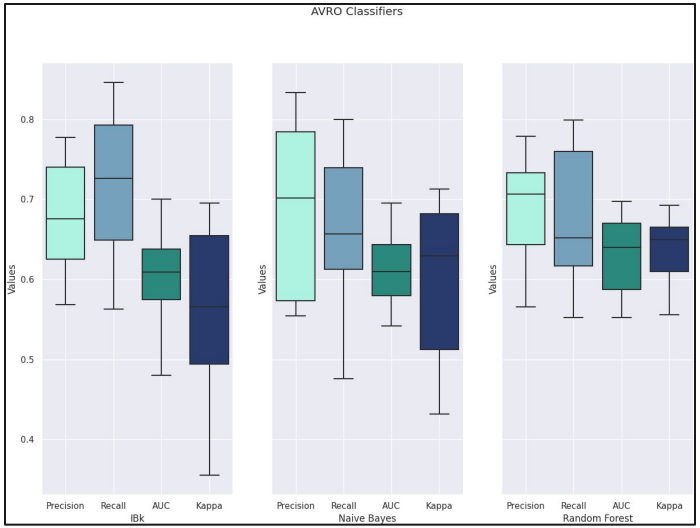
- **Come influisce la Best Practice di mantenere le classi ben commentate sulla Buggy?**
  - Generazione di un **secondo Dataset** che contiene come attributo aggiuntivo la **percentuale di commenti** sulla dimensione totale della classe.
  - Dimensione dei commenti ottenuta tramite "**Tokei**".
  - I risultati su questo Dataset vengono valutati **solo sui Classificatori** e comparati con i risultati del Dataset standard.

Scelte implementative - Machine Learning

- Per la fase di valutazione dello studio è stato utilizzato **WEKA**.
- Come tecnica di valutazione dei classificazione è stato utilizzato **Walk Forward**:
  - Viene mantenuto l'**ordinamento temporale** delle **Release**.
  - Al **passo n-esimo** vengono utilizzate:
    - Le prime n-1 Release come **Training Set**.
    - La n-esima Release come **Testing Set**.
  - Viene **evitata** l'esecuzione del **primo passo** in cui vengono utilizzate **Release per il Training Set**.
- Lo studio prevede l'utilizzo di:
  - **Classificatori** (RandomForest, NaiveBayes, IBK).
  - Tecniche di **Feature Selection** (No selection, Best First).
  - Tecniche di **Sampling** (No sampling, Oversampling, Undersampling, SMOTE).
  - Tecniche di **Cost Sensitive Classification** (No cost sensitive, Sensitive Threshold, Sensitive Learning).
- I risultati ottenuti sono stati rappresentati tramite **Box Plot** generati tramite la libreria **Python Seaborn**.

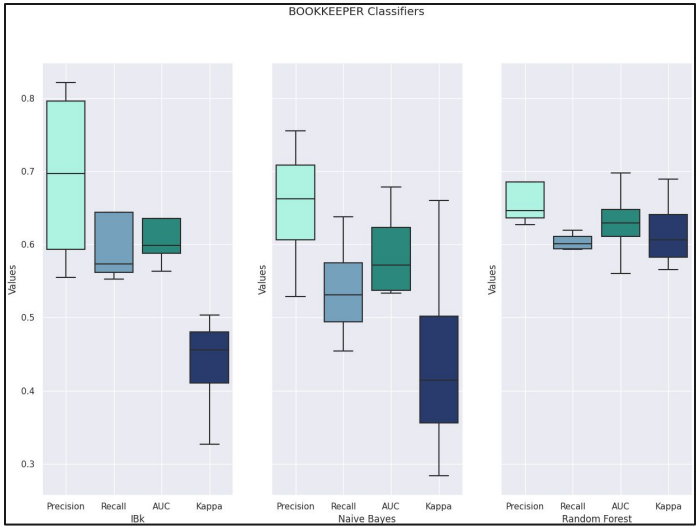
Risultati - Classificatori: Avro

- In termini di mediana **Random Forest** risulta essere il classificatore con i risultati migliori per tutte metriche esclusa la Recall.
- I valori delle mediane di **Random Forest** e **Naive Bayes** non presentano differenze nette. Come **Random Forest** presenta valori di Precision, AUC e Kappa leggermente migliori, **Naive Bayes** presenta una Recall leggermente più alta.
- Tra **Random Forest** ed **IBk** le differenze sono più apprezzabili. In **Random Forest** Precision, AUC e Kappa sono maggiori rispetto a quelli di **IBk** come la Recall risulta ancora una volta minore.

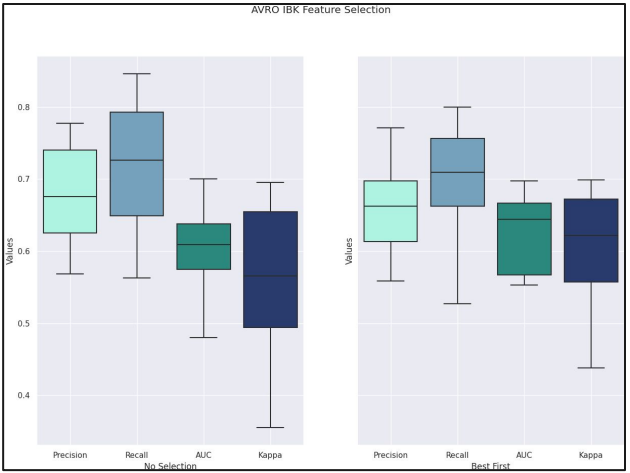


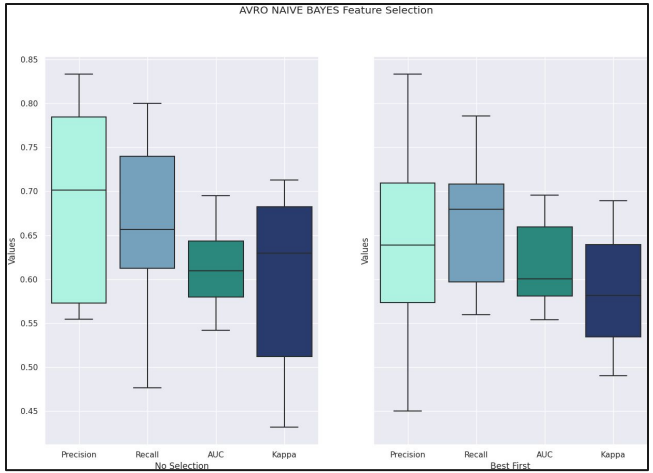
Risultati - Classificatori: Bookkeeper

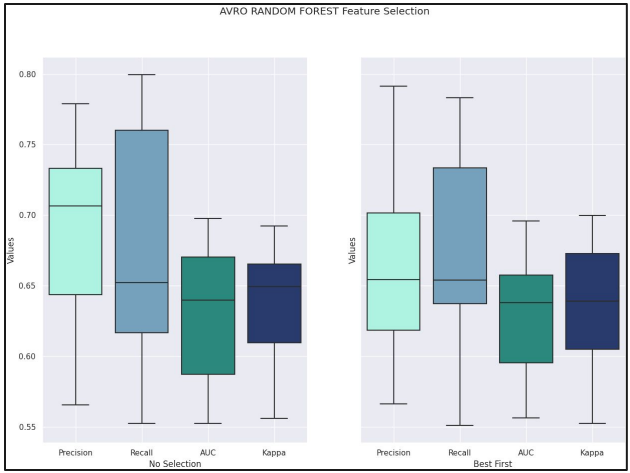
- In termini di mediana **Random Forest** risulta essere il Classificatore con i risultati migliori per tutte metriche esclusa la Precision.
- **IBk** risulta migliore di **Naive Bayes** per ciascuna metrica.
- **Random Forest** presenta un Kappa nettamente migliore rispetto a **Naive Bayes** e **IBk**.
    - **Random Forest** si comporta in maniera migliore di un Dummy Classifier rispetto a quanto facciano **Naive Bayes** e **IBk**.
- Da notare come in **Random Forest** per ogni metrica 1Q e 3Q siano molto ravvicinati.
    - Le metriche risultano essere molto “stabili” in quanto assumono in maggioranza valori in un range ristretto.



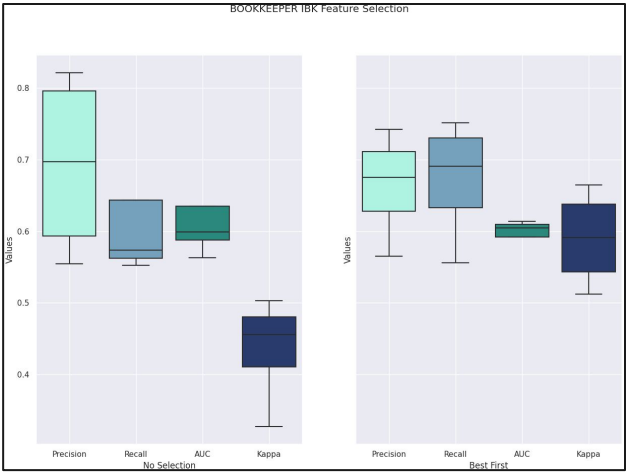
Risultati - Feature Selection: Avro

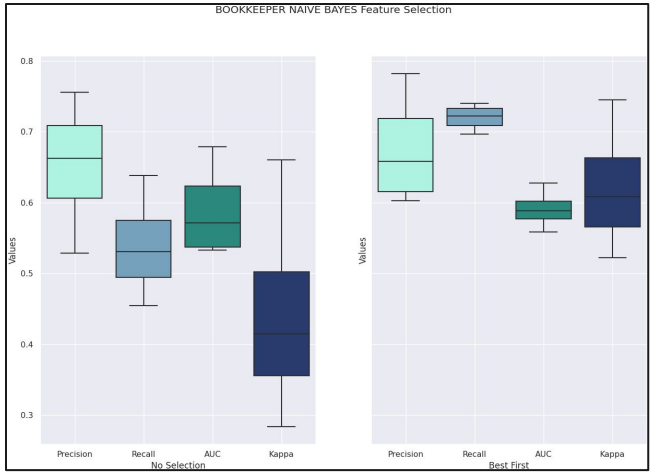


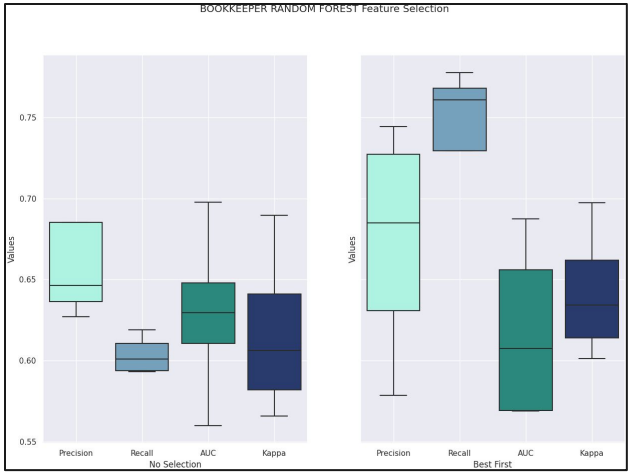




Risultati - Feature Selection: Bookkeeper



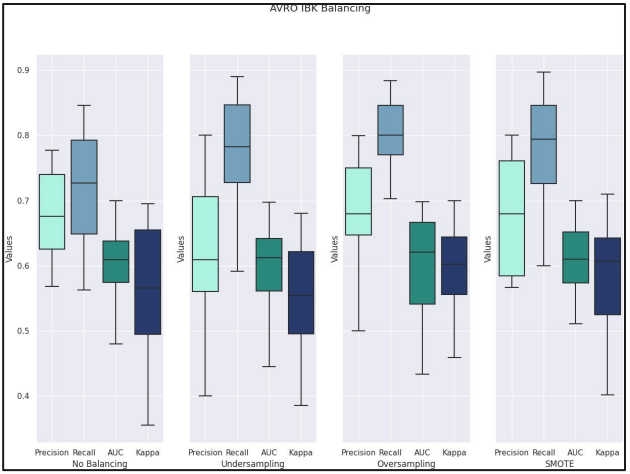


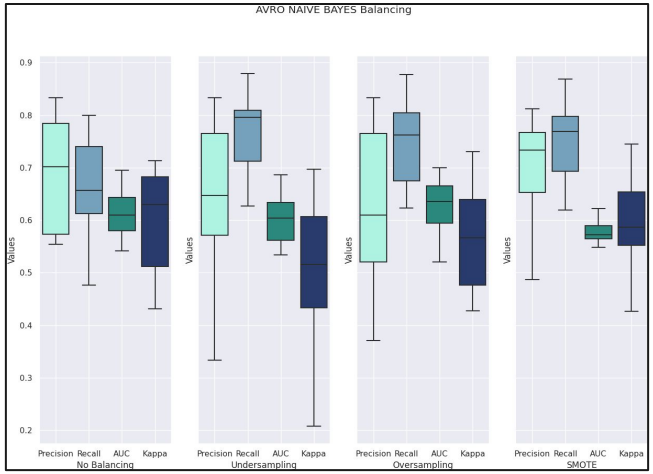


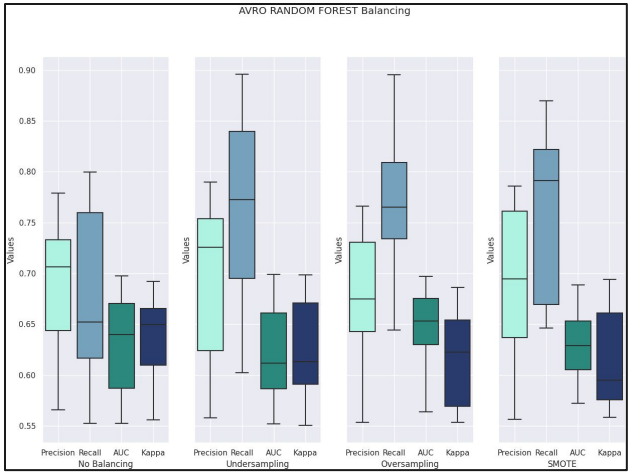
Risultati - Feature Selection

- **AVRO:**
  - **IBk** risulta essere il classificatore che trae maggior beneficio dalla Feature Selection. Considerando i valori delle mediane peggiorano leggermente i valori di Precision e Recall, ma aumentano sia AUC che Kappa. Per tale classificatore 16 attributi risultano ridondanti e correlati tra loro.
  - **Random Forest** è invece il classificatore che peggiora o mantiene costanti i valori delle mediane non ottenendo quindi alcun vantaggio. In questo caso eliminando attributi si riducono le informazioni.
  - **Naive Bayes** migliora la Recall, ma peggiora tutte le altre metriche.
- **BOOKKEEPER:**
  - Sia **IBk** che **Naive Bayes** presentano miglioramenti su tutte le metriche. In entrambi i casi le mediane di Recall e Kappa sono quelle che riscontrano un incremento maggiore.
  - Anche **Random Forest** incrementa tutte le metriche esclusa l'AUC che diminuisce.
  - Questo secondo esempio conferma il principio no silver bullet. La stessa tecnica applicata a due Dataset distinti, ma con gli stessi attributi ottiene risultati molto differenti.

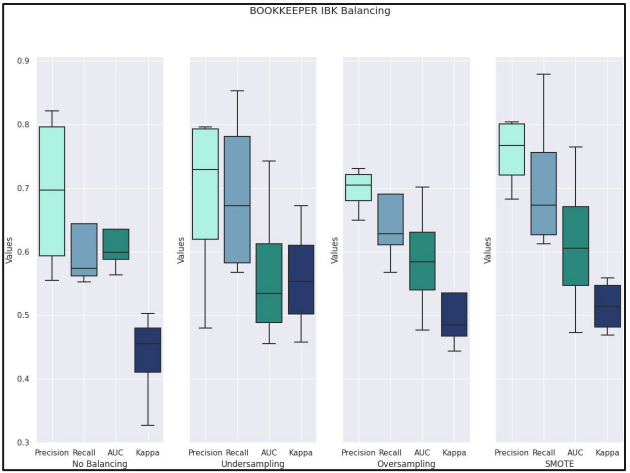
Risultati - Balancing: Avro

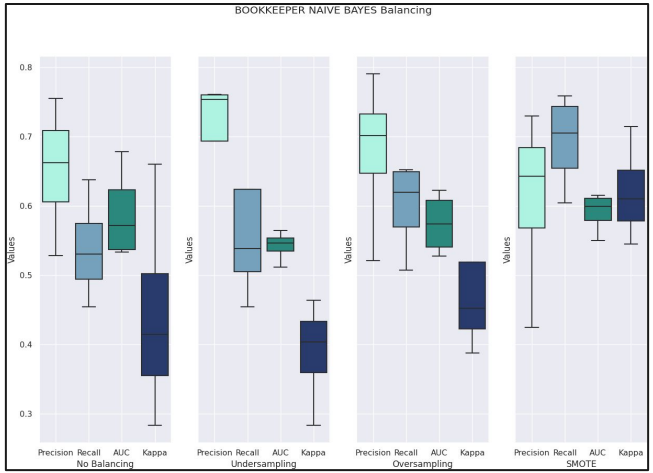


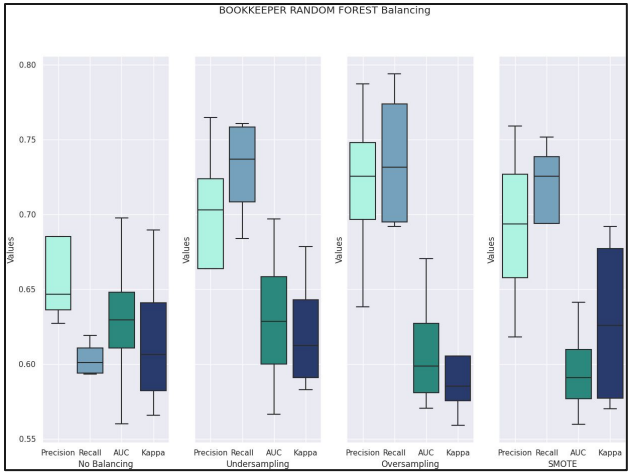




Risultati - Balancing: Bookkeeper



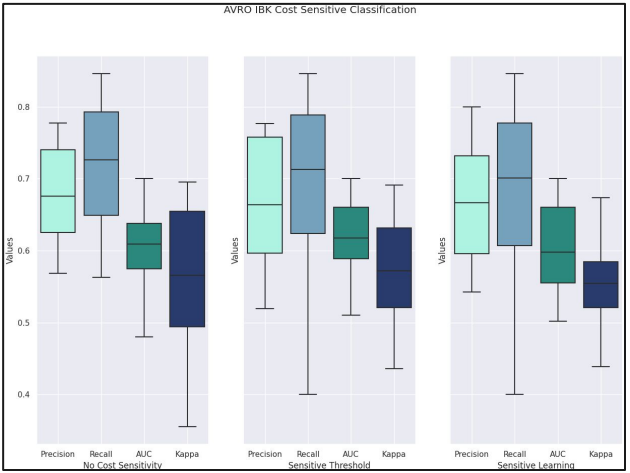


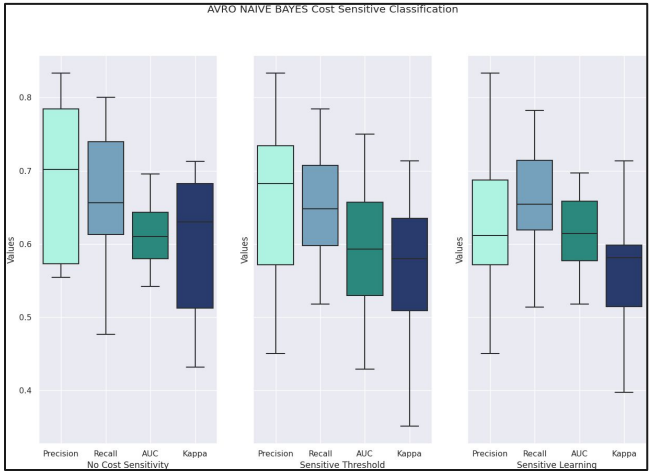


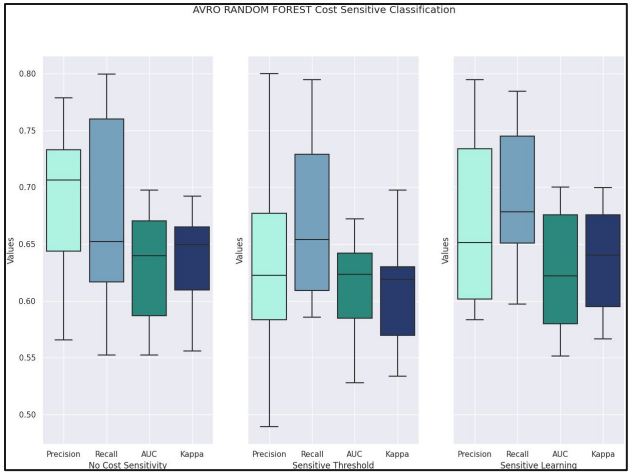
Risultati - Balancing

- **AVRO:**
  - La Recall aumenta in tutti i Classificatori per tutte le tecniche di Balancing applicate. Questo risultato era prevedibile dalla teoria e conferma la bontà del Dataset che presenta nella fase di valutazione quelli che sono i comportamenti attesi.
  - Per tutti i Classificatori sia Kappa che AUC non presentano alterazioni importanti.
  - I Classificatori migliori risultano essere **Naive Bayes** e **Random Forest** che presentano comportamenti simili. Tuttavia non esiste una tecnica di Balancing migliore. Deve essere scelta in base al trade-off tra Precision e Recall più adatto al caso di studio.
- **BOOKKEEPER:**
  - Anche in questo caso si può osservare l'aumento di Recall atteso.
  - **Random Forest** presenta valori di Kappa e AUC bassi per tutte le tecniche di Balancing.
  - **IBk** aumenta tutte le metriche rispetto alla versione senza Balancing, ma presenta valori di Kappa bassi.
  - **Naive Bayes** è il classificatore che ottiene i risultati migliori in particolare la versione che utilizza SMOTE aumenta in maniera apprezzabile anche Kappa.

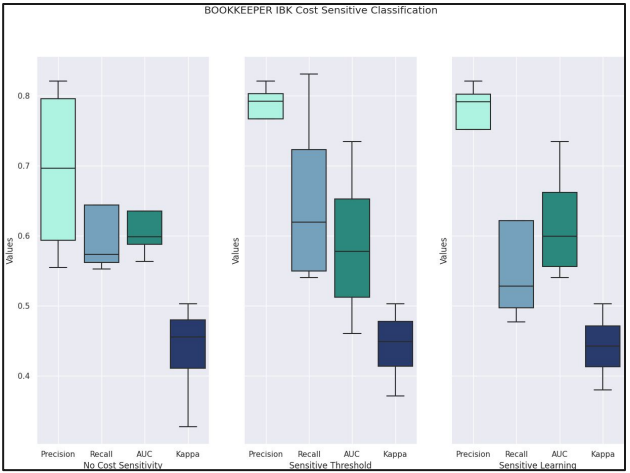
Risultati - Cost Sensitive Classification: Avro

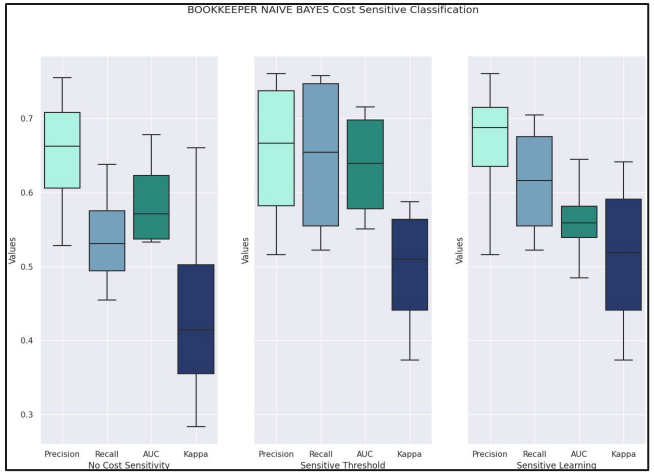


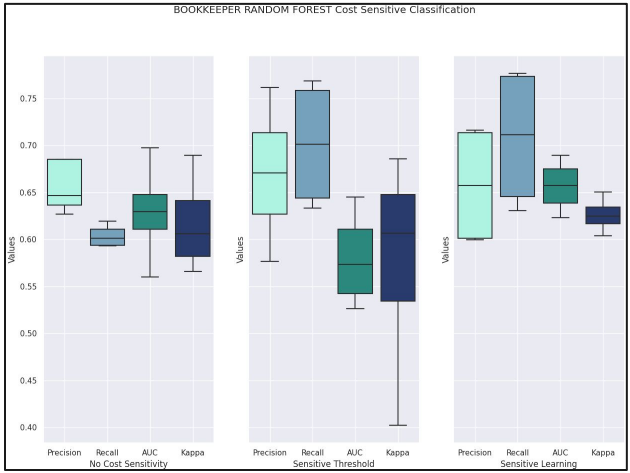




Risultati - Cost Sensitive Classification: Bookkeeper



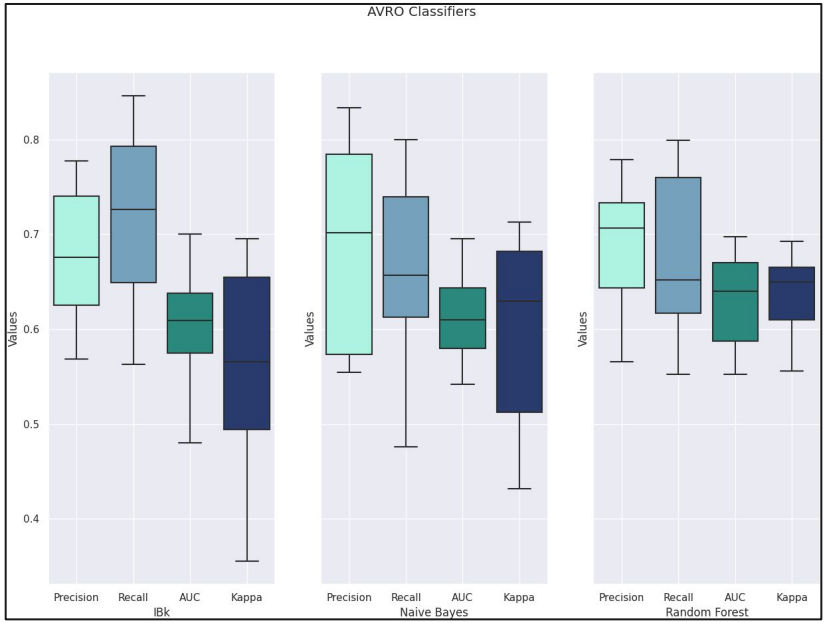


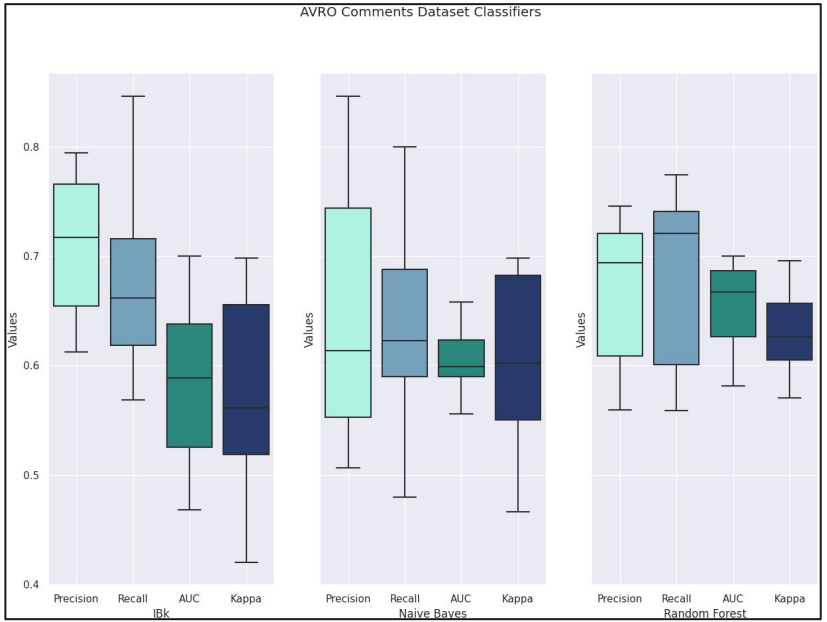


Risultati - Cost Sensitive Classification

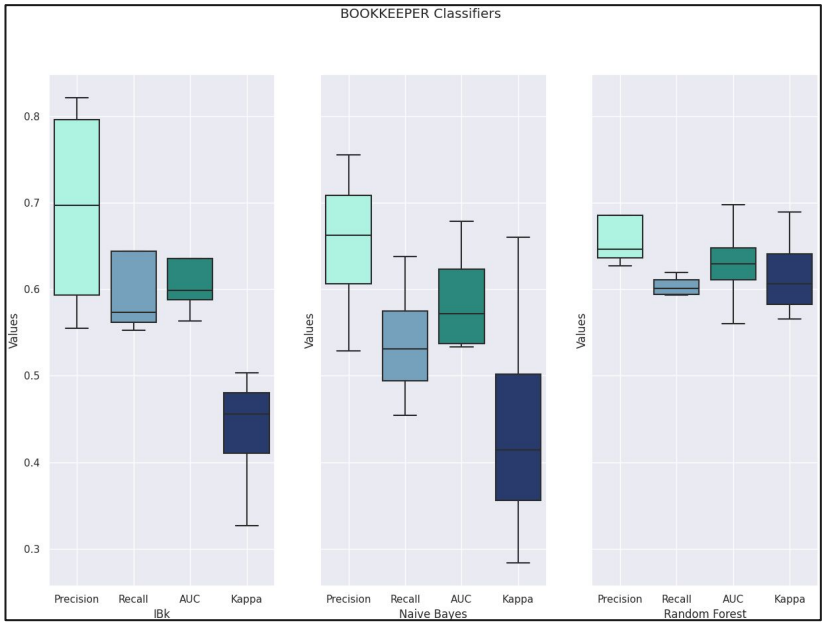
- **AVRO:**
  - La matrice dei costi utilizzata prevede il costo di True Positive e True Negative pari a 0. Il costo di False Positive pari a 1 mentre viene data molta più importanza ai False Negative che hanno costo pari a 10.
  - A prescindere dalla tecnica utilizzata in **IBk** i valori delle metriche non presentano alterazioni apprezzabili.
  - **Random Forest** e **Naive Bayes** presentano risultati molto simili. Per entrambe le tecniche applicate tendono a mantenere costante AUC, diminuiscono Kappa e Precision, ma aumenta la Recall.
- **BOOKKEEPER:**
  - Viene utilizzata la stessa matrice dei costi descritta precedentemente.
  - **IBk** assume lo stesso comportamento del caso precedente.
  - **Naive Bayes** e **Random Forest** tendono ad aumentare tutte le metriche.
  - **Random Forest** con Sensitive Learning presenta le metriche con i valori migliori.

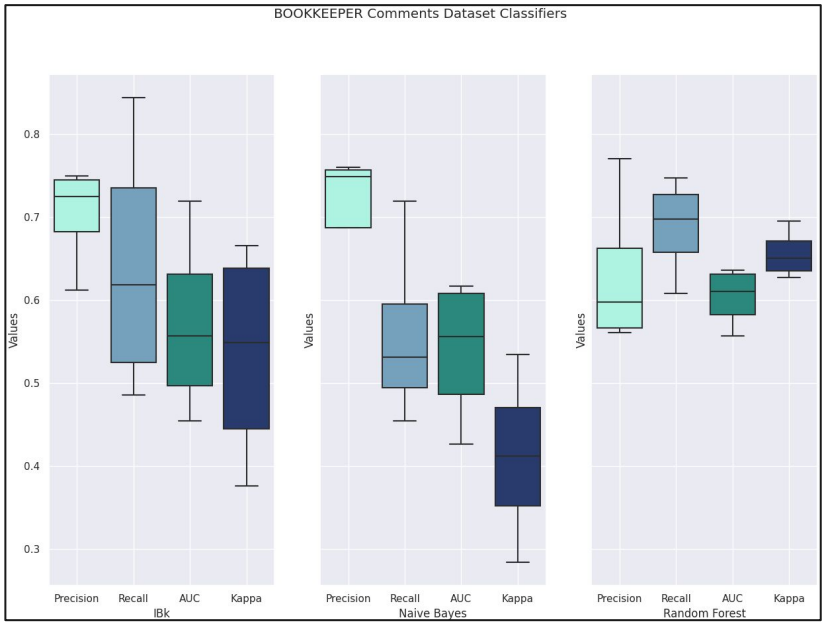
Risultati - Dataset “Comment”: Avro





Risultati - Dataset “Comment”: Bookkeeper





Risultati - Dataset “Comment”

- **AVRO:**
  - Utilizzando il Dataset che contiene l’attributo della percentuale dei commenti sia **IBk** che **Naive Bayes** tendono a peggiorare tutte le metriche.
    - Nel contesto di questa analisi l’attributo non sembra essere indicativo di contro peggiora le metriche rispetto al Dataset standard.
  - Nel caso di **Random Forest** si mantengono costanti AUC e Precision, diminuisce Kappa, ma aumenta significativamente la Recall.
- **BOOKKEEPER:**
  - Utilizzando il nuovo Dataset non si evincono peggioramenti delle metriche rispetto a tutti i Classificatori e all’utilizzo del Dataset standard.
    - In questo progetto tale metrica risulta essere indicativa.
  - **IBk** aumenta Precision, Recall e Kappa, ma diminuisce AUC, mentre **Naive Bayes** non mostra alterazioni.
  - **Random Forest** può essere considerato come il migliore per valori delle mediane.
    - La Precision diminuisce drasticamente, ma aumentano tutte le altre metriche che raggiungono i valori più alti ottenuti dagli altri Classificatori.

Minacce alla validità

- La principale minaccia alla validità di tale studio si ritrova nella **qualità** del Dataset utilizzato.
  - Principio **garbage in garbage out**.
- Durante più di una fase della generazione del Dataset sono stati riscontrate **inconsistenze** nei dati.
  - Soprattutto nel merge delle informazioni tra **Jira** e **Git**.
  - Molti progetti compresi quelli analizzati in tale studio stanno abbandonando **Jira** in favore di **GitHub**.
- I dati riportati nei Dataset sono stati **calcolati per Release** e non cumulativi per evitare di aggregare anche gli errori della fase di misurazione.

Conclusioni e Links

- Lo studio ha dimostrato gli **enormi vantaggi** che possono essere portati nel mondo dell'Ingegneria del Software dal Machine Learning.
- Sono stati riscontrati i principi che sono stati enfatizzati più volte a lezione:
  - **garbage in garbage out:** la quantità dei dati è importante tanto quanto la qualità.
  - **no silver bullet:** non esiste una configurazione ottimale, ma ogni scelta è dettata da trade off.
- I risultati ottenuti possono essere considerati **soddisfacenti**.
  - Kappa è sempre maggiore di 0. Ciò indica che ogni configurazione analizzata risulta migliore di un Classificatore Dummy.
  - E' importante avere **consapevolezza del livello di validità** del proprio studio e di quali fattori possono minarla.
- **Links**:
  - **GitHub:** https://github.com/IronMarken/ML-for-SE
  - **SonarCloud:** https://sonarcloud.io/project/overview?id=IronMarken_ML-for-SE