Deliverable 2: Applicazione di tecniche di ML su progetti open source

Pierciro Caliandro

Università degli studi di Roma Tor Vergata

July 14, 2021

Indice

Introduzione

- Lo scopo della presentazione è quello di mostrare i risultati a seguito dell'applicazione di tecniche di sampling, classificazioni sensibili al costo, e feature selection su modelli di ML. In particolare, ci si concentra su come tali tecniche impattano sulle metriche di accuratezza per i seguenti classificatori:
  - NaiveBayes
  - RandomForest
  - Ibk
- I dati forniti ai classificatori per il training ed il testing sono stati raccolti da due progetti open source della Apache Software Foundation:
  - Apache BookKeeper
  - Apache ZooKeeper
- Le analisi partono dai risultati ottenuti, per tutti i classificatori, senza l'applicazione di alcuna tecnica per poi vedere con quali tecniche e per quali classificatori si ottengono risultati migliori

Derivazione dei dataset

- Come prima cosa, per entrambi i progetti, è stata utlizzata la rest API di Jira per poter ottenere tutti i Ticket di tipo bug relativi ad entrambi i progetti
- Dopo aver ordinati i ticket in base alle loro date di risoluzione, sono stati trovati ed ordinati i corrispettivi commit andando a consultare il log ottenibile da git, per fare ciò si è utilizzato il tool JGit
- A questo punto, scorrendo le varie release, sono state calcolate le 9 metriche scelte per entrambi i progetti, andando poi a riempire un file csv per ogni singolo progetto. Per ciascun progetto, vengono riportati solo i dati relativi alla prima metà del totale delle release

Struttura del dataset

- La struttura di entrambi i dataset è la seguente:

Project name	Release	Class name	Size	NR	NAuth	Age	MAX_LOC_ADDED	LOC_ADDED	AVG_LOC_ADDDED	Churn	AVG_CHURN	buggyness
BOOKKEEPER	4.0.0	bookkeeper-server/src/main/java/org/apache/bookkeeper/bookie/Bookie.java	1026	15	3	35	545	1791	119	1026	68	yes
BOOKKEEPER	4.0.0	bookkeeper-server/src/main/java/org/apache/bookkeeper/bookie/LedgerEntryPage.java	157	4	3	35	151	315	78	157	39	yes
BOOKKEEPER	4.0.0	bookkeeper-server/src/main/java/org/apache/bookkeeper/bookie/BookieException.java	92	4	3	35	81	202	50	92	23	no
BOOKKEEPER	4.0.0	bookkeeper-server/src/main/java/org/apache/bookkeeper/bookie/EntryLogger.java	471	10	3	35	487	1120	112	471	47	yes
BOOKKEEPER	4.0.0	bookkeeper-server/src/main/java/org/apache/bookkeeper/bookie/BufferedChannel.java	180	5	2	35	168	363	72	180	36	no
BOOKKEEPER	4.0.0	bookkeeper-server/src/main/java/org/apache/bookkeeper/bookie/FileInfo.java	204	6	3	35	124	334	55	204	34	yes
BOOKKEEPER	4.0.0	bookkeeper-server/src/main/java/org/apache/bookkeeper/bookie/LedgerDescriptor.java	149	6	3	35	133	301	50	149	24	yes
BOOKKEEPER	4.0.0	bookkeeper-server/src/main/java/org/apache/bookkeeper/bookie/LedgerCache.java	551	8	3	35	536	1157	144	551	68	no

Figure: Header per i file csv

- In entrambi i dataset sono stati riportati 9 metriche, relative alla singola classe, usate poi dai classificatori per stimare se una classe presentasse un bug in una determinata release:
  - Size: numero di LOC
  - NR: Numero di revisioni
  - NAuth: Numero di autori della classe
  - Age: 'età', in settimane
  - MAX_LOC_ADDED: massimo numero di LOC aggiunte in una release
  - LOC_ADDED: LOC aggiunte in una release
  - AVG_LOC_ADDED: media di LOC aggiunte in una release
  - Churn: differenza fra LOC aggiunte e rimosse
  - AVG_Churn: media della differenza fra LOC aggiunte e rimosse
- L'ultimo attributo del dataset è la buggyness nella release corrente, calcolata usando le Affected Version quando disponibili dai ticket di Jira, o altrimenti applicando il metodo proportion per stimare le affected version

Tecnica di classificazione

- La tecnica di calssificazione utilizzata è Walk Forward
- Il training set è stato incrementato di volta in volta, andando ad aggiungere sempre i dati relativi alla successiva release
- Per il testing set si usa sempre la prima release non ancora inclusa nel training set
- Ad esempio, per la prima run si avrà il training set contenente la release 1 ed il testing set formato dalla release 2. Nella run successiva il training set sarà costituito dalle release 1 e 2, mentre il testing set dalla release 3
- Per la classificazione, è stato usato il tool weka, sfruttando sia la API che la versione stand alone.
- I valori riportati nei due dataset sono stati calcolati dall'API, e confrontati con quanto veniva riportato dall'esecuzione sugli stessi dati con la versione stand alone

Tecniche utilizzate

- Per cercare di migliorare i valori ottenuti dalla prima analisi, sono state applicate alcune tecniche:
  - Feature Selection, utilizzando Best First come tecnica
  - Sampling, utilizzando
    - under-sampling: vengono diminuite le istanze della classe maggioritaria fino a pareggiare quelle della classe minoritaria
    - over-samplig: vengono aumentate le istanze della classe minoritaria fino a pareggiare quelle della classe maggioritaria
    - SMOTE: vengono create istanze aggiuntive per la classe minoritaria in maniera "sintetica"
  - Cost sensitive valuation, usando:
    - sensitive threshold, viene aggiustato il valore della threshold
    - sensitive learning, le classi vengono replicate in base al peso, quindi è come se venissero ripesate

  In entrambe i casi, la matrice dei costi prevede un costo 10 volte maggiore per un falso negativo rispetto a quello per un falso positivo

Metriche analizzate

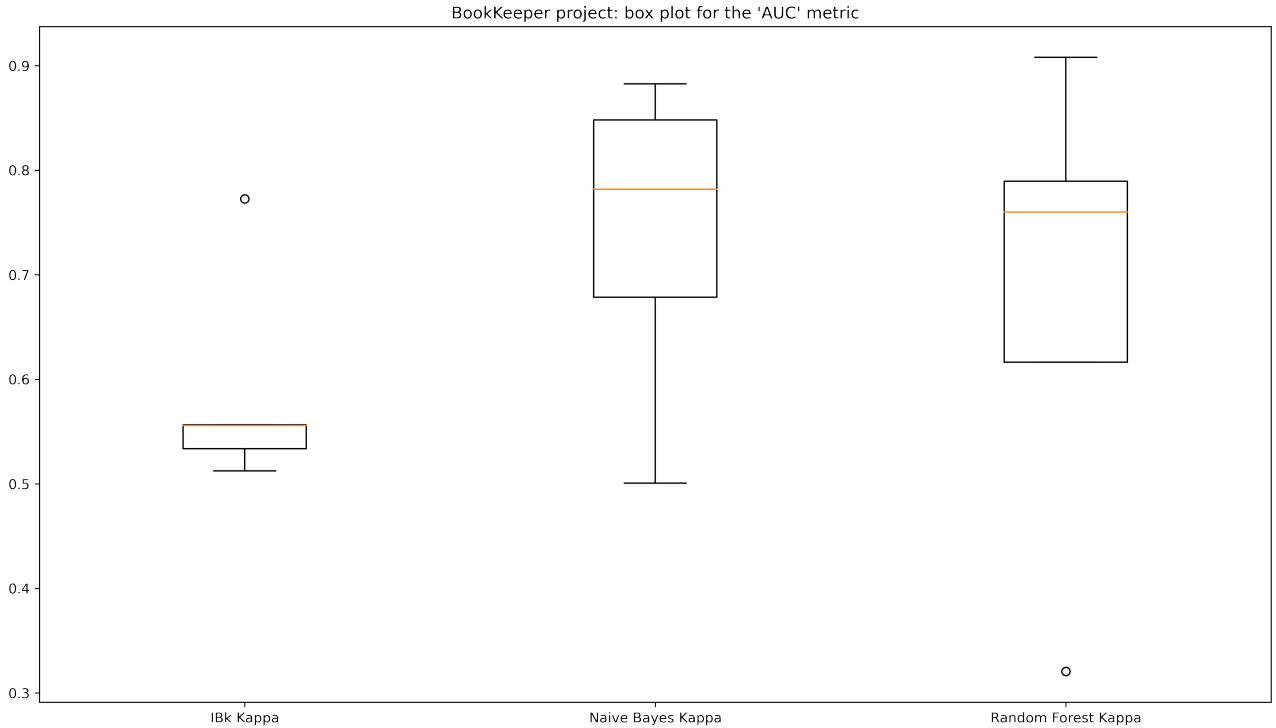
- Sono stati considerati ed analizzati i valori per le seguenti metriche di performance:
  - AUC: Area Under The Curve, area sottesa alla ROC che da una misura di quanto il classificatore è in grado di predirre correttamente una istanza positiva scelta casualmente con un valore più alto di uno per una istanza negativa presa casualmente
  - Recall: definita come $\frac{TP}{TP+FN}$, che fornisce una misura relativamente a quanti valori positivi sono stati classificati su quanti effettivamente ce ne erano
  - Precision: definita come $\frac{TP}{TP+FP}$, da una misura dell'errore che si commette nello stimare un positivo, quindi aiuta nel poter capire quanto è attendibile il valore di Recall.
  - Kappa: metrica che definisce quanto il classificatore è meglio rispetto ad un classificatore dummy, ovvero puramente randomico

Analisi dei valori

- Per alcune metriche, l'analisi dei valori è stata guidata dal confronto con i valori che si avrebbero se si usasse un classificatore "dummy", ovvero random
- Per la AUC, avere un valore pari a 0.5 vuol dire che il classificatore si comporta come uno random, mentre averlo minore di 0.5 indica un comportamento peggiore
- La metrica Kappa è un indice di quanto il classificatore va meglio rispetto ad uno random: valori pari a 0 indicano un comportamento del classificatore analogo a quello di uno random, mentre valori minori di 0 ne indicano un comportamento peggiore
- Precision e Recall vengono analizzate insieme, in quanto la Precision da una indicazione di quanto i valori ottenuti per la Recall siano "affidabili"

Analisi della AUC per BookKeeper

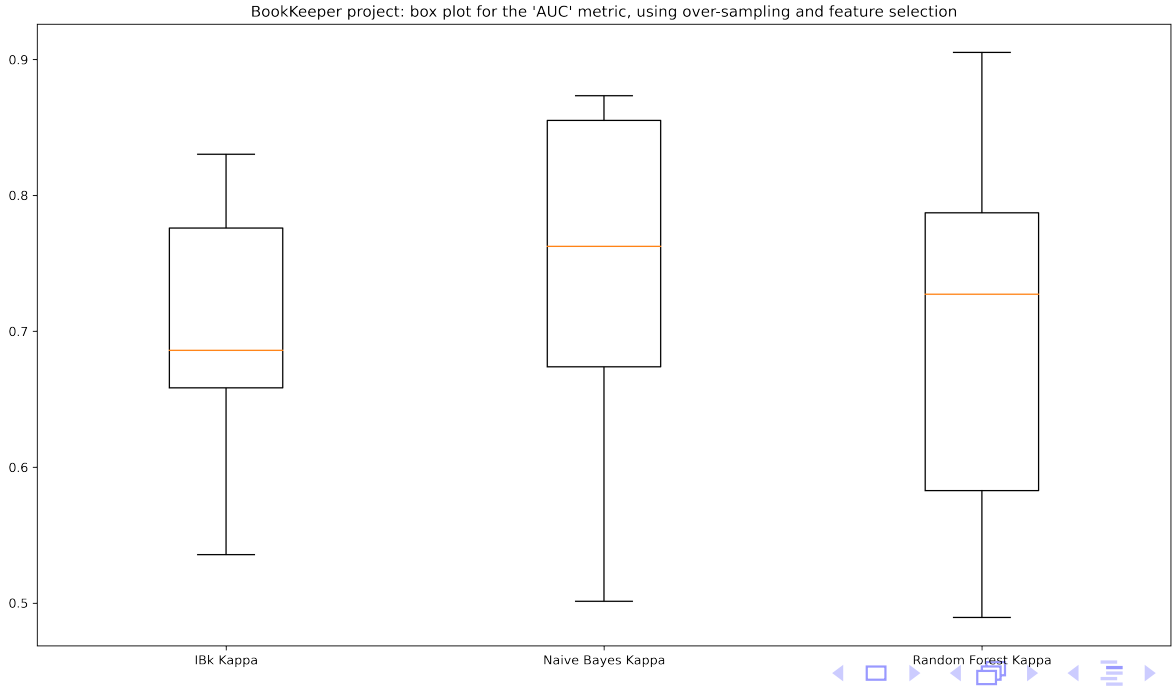
- Dall'analisi dei valori per la AUC dei tre classificatori, considerando il progetto BookKeeper, è stato estrapolato il seguente box plot:



- IBk presenta un una distribuzione della AUC con valori migliori di quelli di un classificatore random, anche se di poco, con un outlier intorno al valore 0.77
- Anche Random Forest ha una distribuzione dei valori migliore di quella di un classificatore random, pur presentando un outliers nel punto 0.32
- Naive Bayes risulta il classificatore con la migliore distribuzione per la metrica AUC

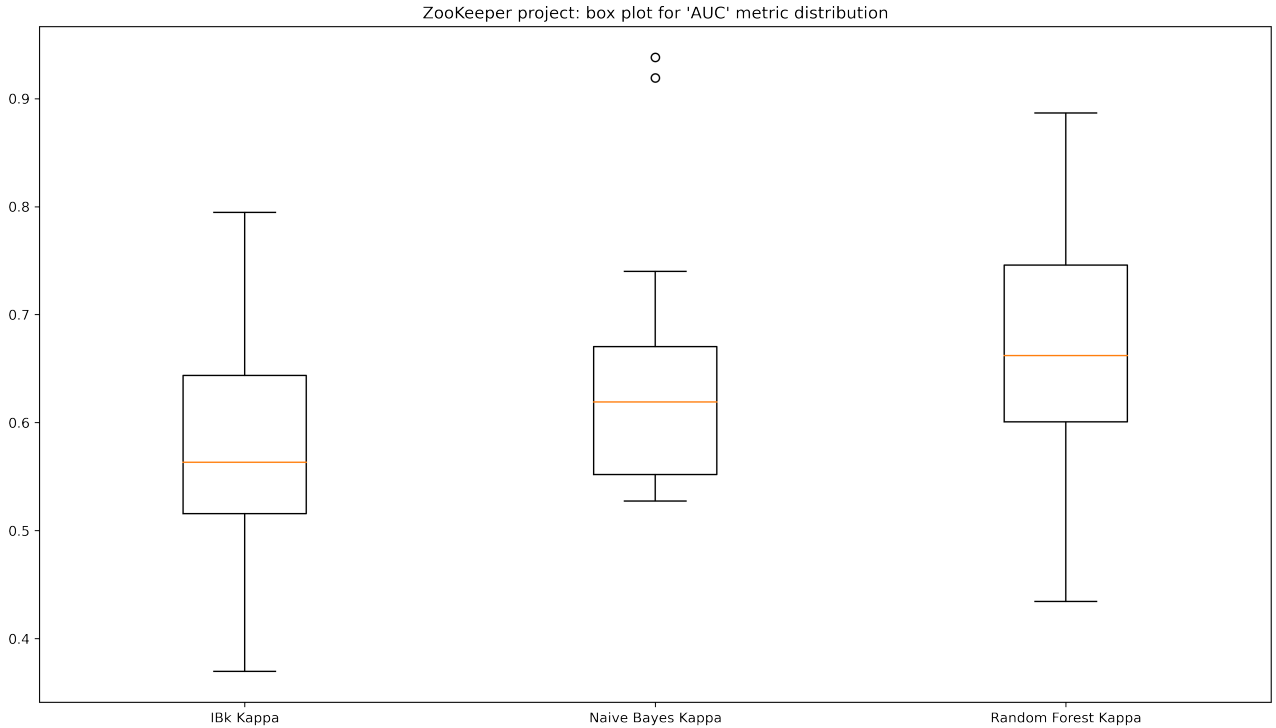
Analisi della AUC per BookKeeper - miglioramento dei valori

- L'analisi successiva dei valori era volta a capire quale tecnica migliorasse il valore per la metrica di AUC per i singoli classificatori
- Da una prima analisi, risulta che cambiare il valore della metrica peggiora sempre quando viene utilizzato un classificatore cost sensitive
- Fra tutti, il classificatore su cui ci si è concentrati per l'aumento della metrica è IBk, che mostra i valori peggiori
- Risulta che l'applicazione di over-sampling e di feature selection migliorano di molto i valori per la metrica IBk, con anche un leggero miglioramento per i valori di Random Forest



Analisi della AUC per ZooKeeper

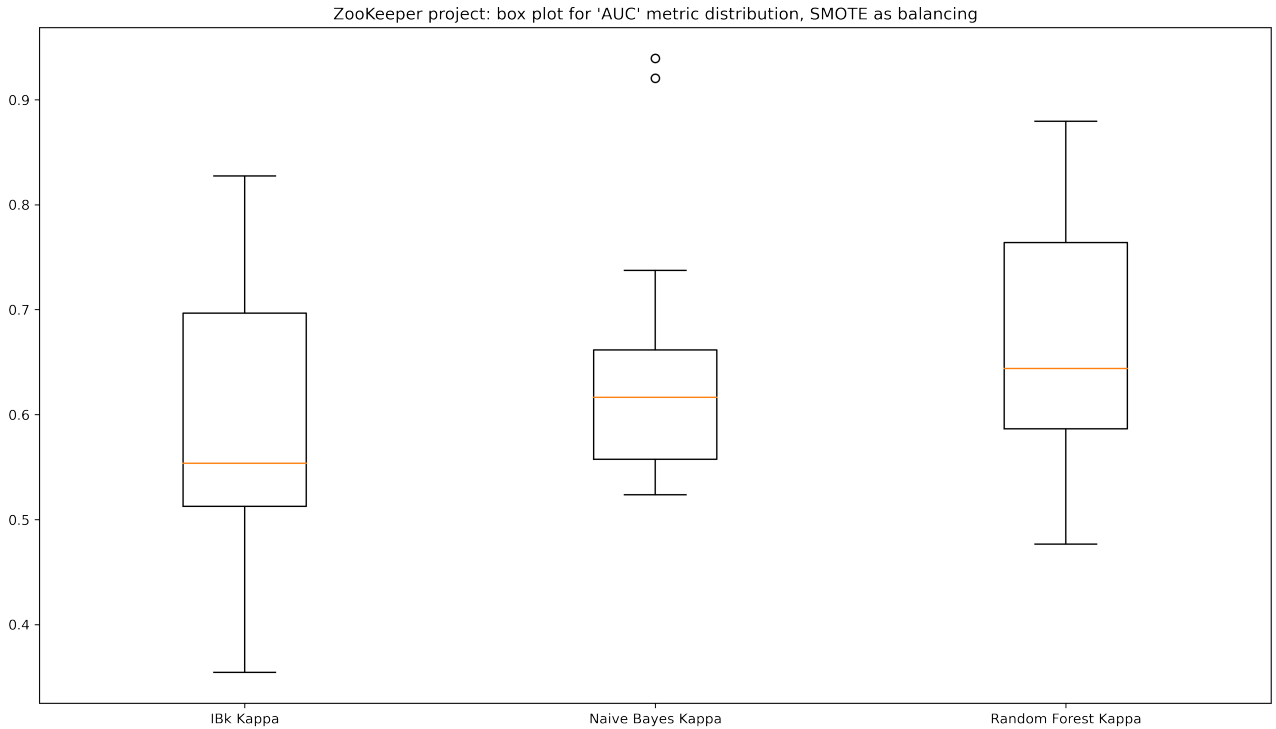
- Per determinate run, alcuni dei classificatori presentavano NaN come valore della metrica, quindi tali valori sono stati scartati
- La prima analisi per la AUC sul progetto ZooKeeper fornisce i seguenti risultati:



- In questo caso, Random Forest è il classificatore con la distribuzione dei valori migliore, mentre il peggiore è IBk.
- Sia IBk che Random Forest mostrano, per alcune run, valori peggior del caso di un classificatore random

Analisi della AUC per ZooKeeper - miglioramento dei valori

- Anche in questo caso, il primo classificatore di cui si cerca di migliorare i valori per la metrica di AUC è IBk
- Confrontando le diverse tecniche, si evince che sia per IBk i valori della distribuzione della metrica migliorano usando SMOTE come filtro per il sampling
- Il box plot sottostante mostra i risultati ottenuti



Medie dei valori per AUC

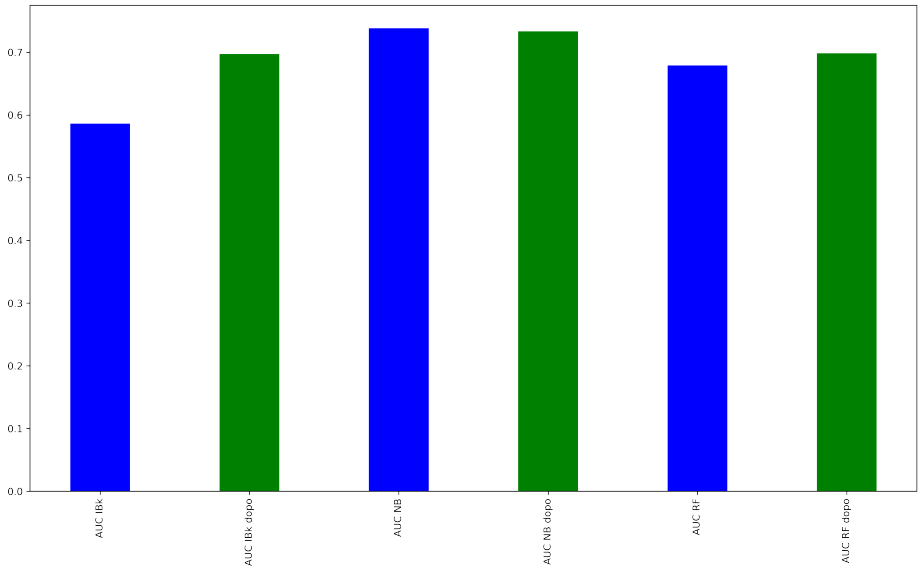


Figure: Confronto dei valori di AUC per il progetto BookKeeper

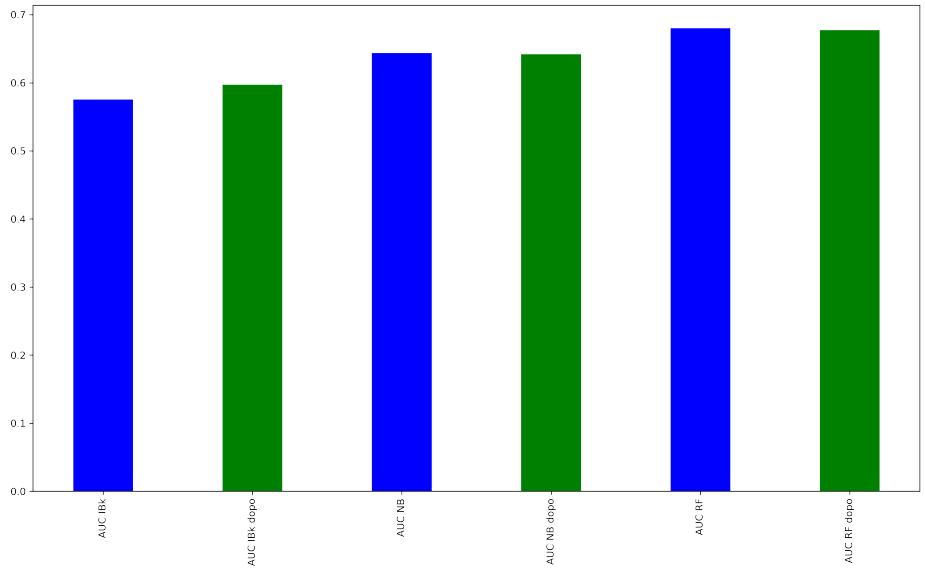
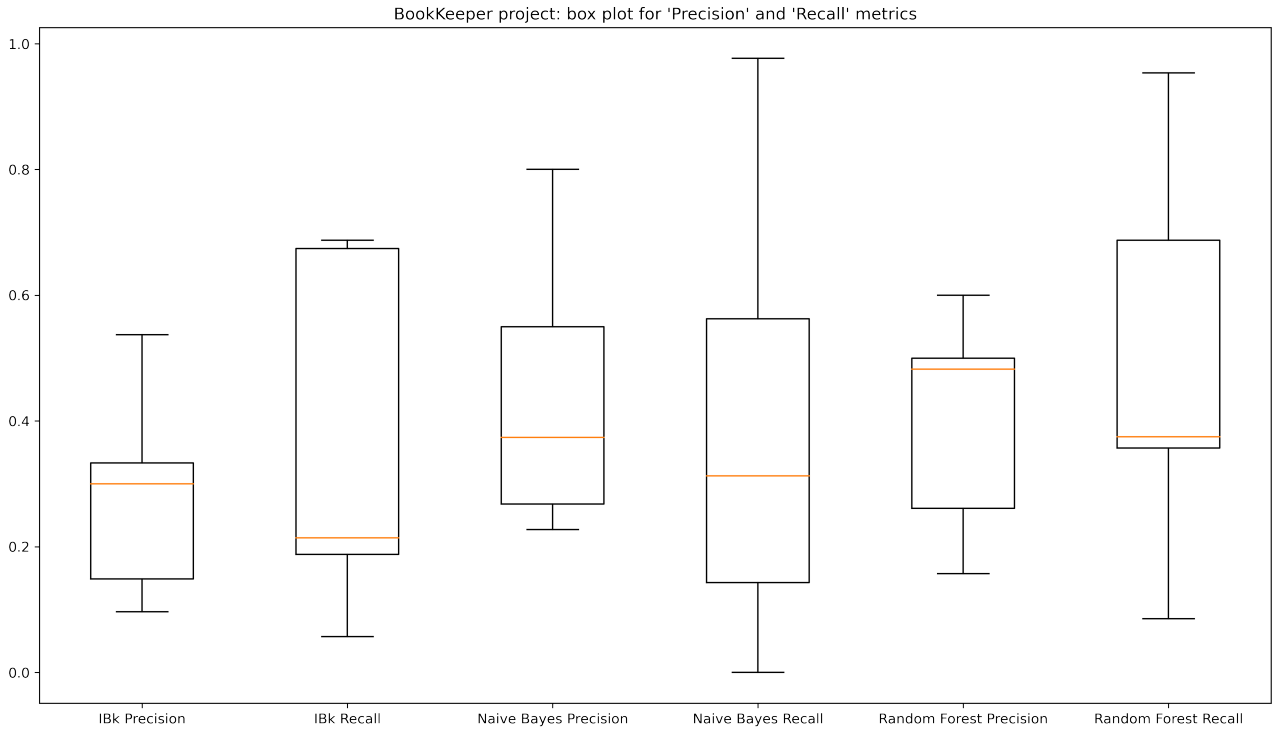


Figure: Confronto dei valori di AUC per il progetto ZooKeeper

Analisi di Precision e Recall per BookKeeper

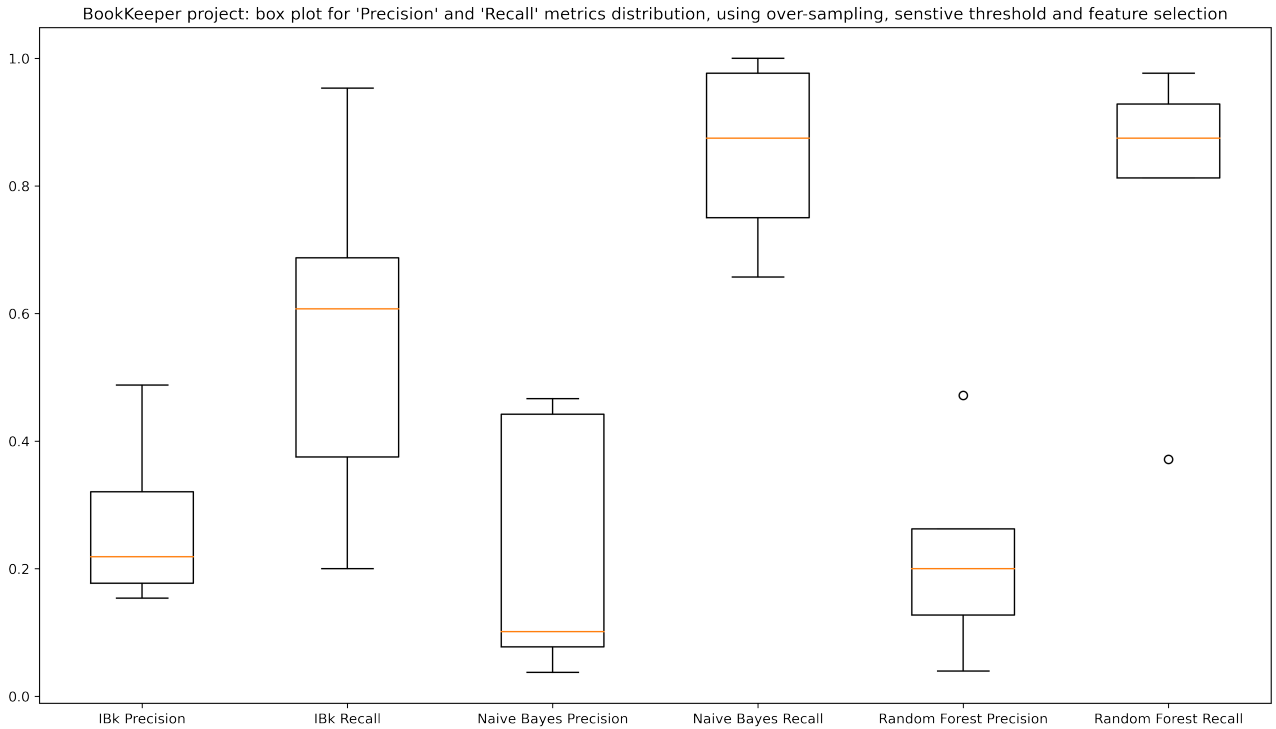
- Per il progetto BookKeeper, si ottengono i seguenti risultati:



- Tutti i classificatori sembrano mostrare delle metriche coerenti col fatto che il dataset è molto sbilanciato, essendo come già detto molto più presenti i valori "no" per l'attributo buggyness, che è quello che viene stimato
- Questo giustifica i bassi valori di Precision, in quanto riuscire a predirre correttamente una istanza positiva non è semplice, ed anche di Recall

Tentativo di miglioramento dei valori

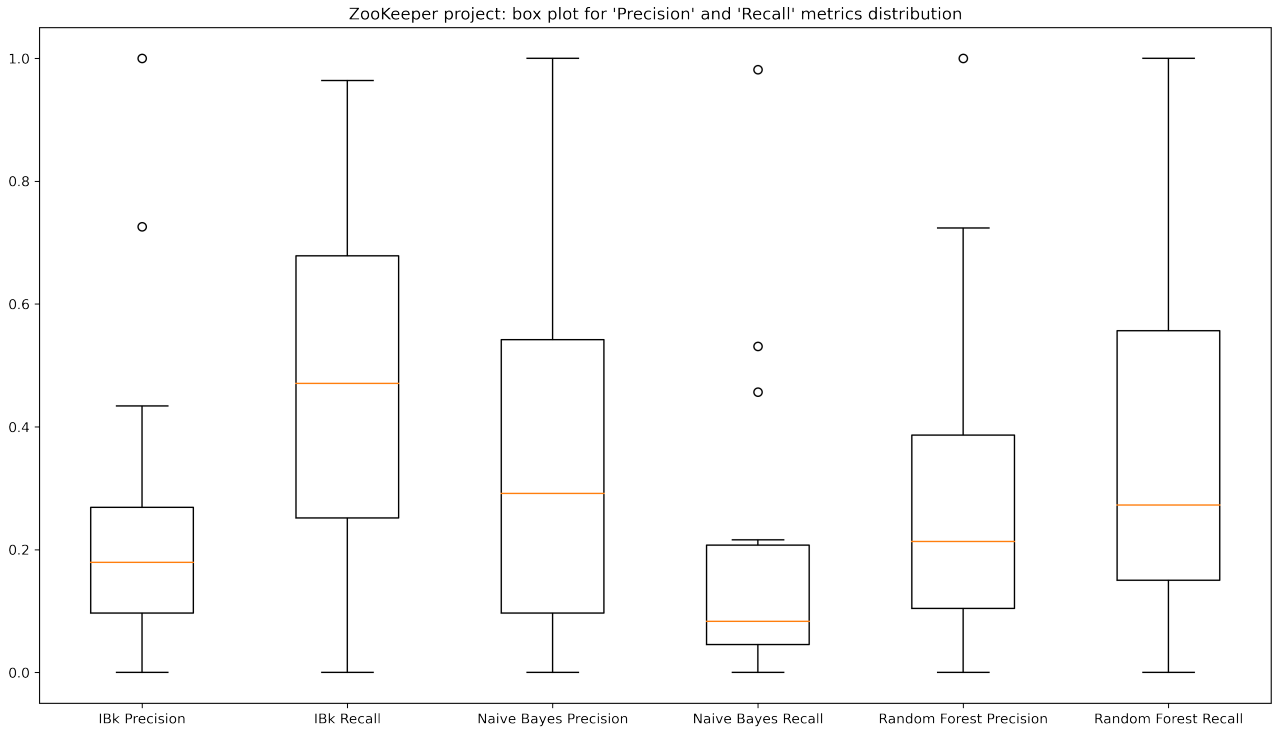
- Dall'analisi dei valori, si evince che applicando over sampling come meccanismo di balancing, feature selection e sensitive threshold, c'è una aumento molto importante dei valori delle distribuzioni per le metriche di Recall ottenute da Naive Bayes e Random Forest



- Avendo applicato over sampling, le istanze nella classe minoritaria stata aumentate e quindi questo spiega il vertiginoso aumento dei valori per le distribuzioni
- I valori di Precision si abbassano rispetto al caso precedente, questo in poiché c'è un elevato numero di istanze classificate come FP

Analisi di Precision e Recall per ZooKeeper

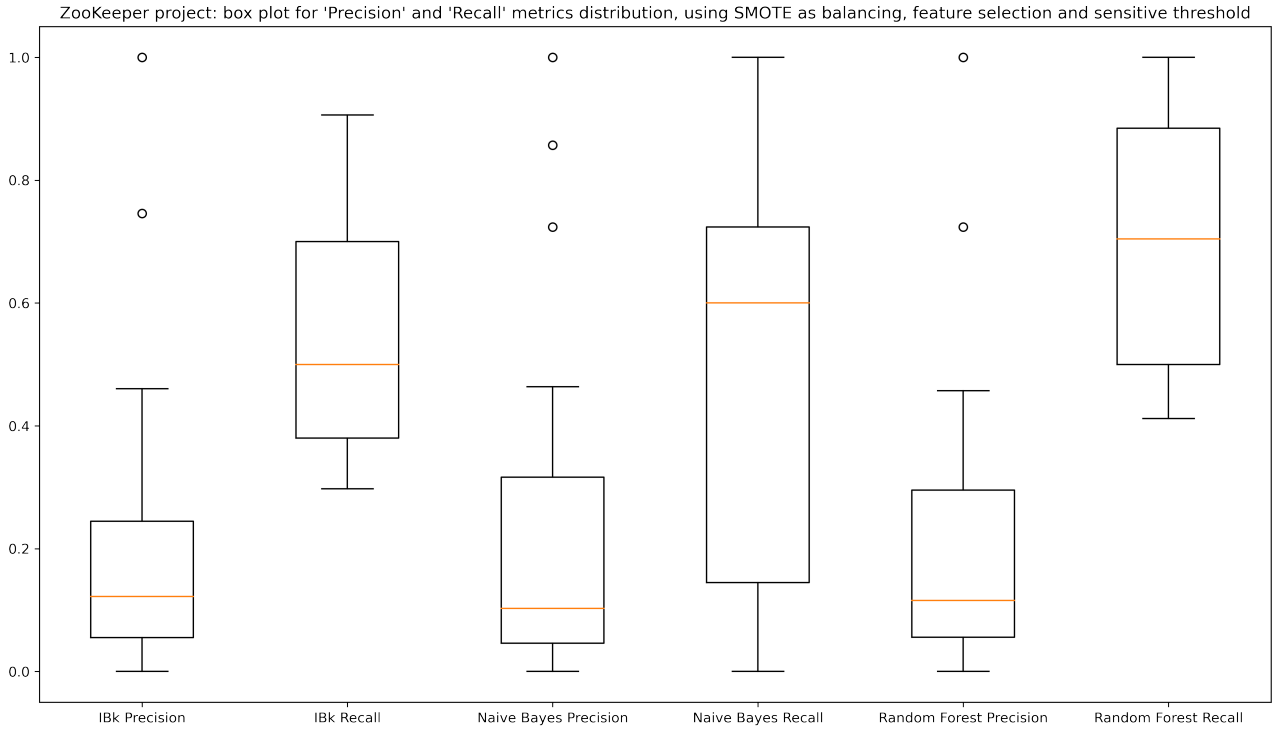
- Per il progetto ZooKeeper, si ottengono i seguenti risultati:



- In questo caso, il classificatore che mostra i valori peggiori per la metrica di Precision è Naive Bayes, ma con dei valori di Recall più alti rispetto agli altri classificatori
- Anche qui, l'obiettivo è quello di cercare di alzare i valori per le metriche di tutti i classificatori

Tentativo di miglioramento dei valori

- Anche in questo caso, lo scopo dell'analisi è cercare di applicare le tecniche viste per aumentare i valori di Precision e Recall
- Applicando feature selection, sensitive threshold e SMOTE come filtro di balancing, si ottengono dei valori di Precision più alti per i classificatori IBk e Random Forest, ma la Recall rimane bassa



- la varianza per la distribuzione dei valori di Recall per Naive Bayes è molto maggiore rispetto agli altri due classificatori, quindi non c'è miglioramento
- Random Forest mostra una distribuzione analoga al caso precedente, ovvero senza l'applicazione di filtri, balacing o cost sensitive

Risultati analitici e grafici

Classificatore	Media dei FP	Tecniche usate
IBk	18.95	Nessuna
IBk	38.68	SMOTE + FS + Sens. Thresh.
Naive Bayes	4.1	Nessuna
Naive Bayes	27.22	SMOTE + FS + Sens. Thresh.
Random Forest	13.13	Nessuna
Random Forest	48.12	SMOTE + FS + Sens. Thresh.

Table: Valori medi dei FP per il progetto ZooKeeper prima e dopo l'applicazione delle tecniche

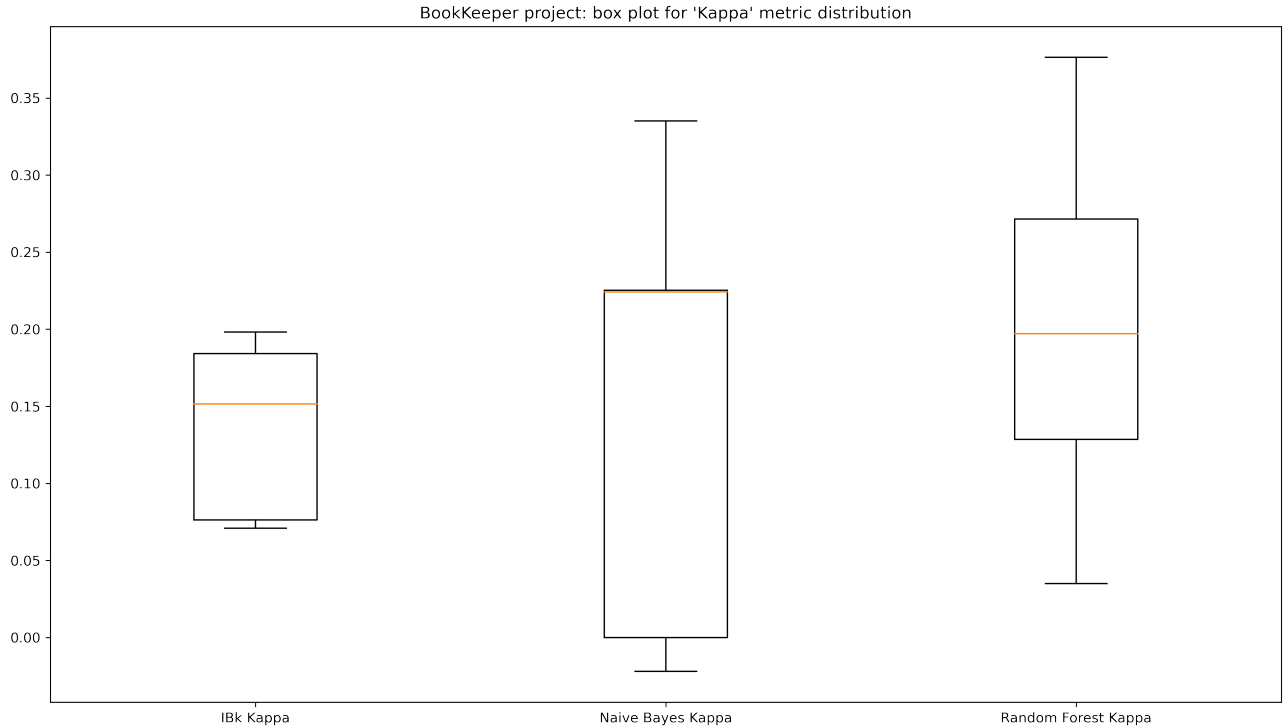
Classificatore	Media dei FP	Tecniche usate
IBk	8.93	Nessuna
IBk	12.53	over-sampl. + FS + Sens. Thresh.
Naive Bayes	5.93	Nessuna
Naive Bayes	56.07	over-sampl. + FS + Sens. Thresh.
Random Forest	8.8	Nessuna
Random Forest	38.87	over-sampl. + FS + Sens. Thresh.

Table: Valori medi dei FP per il progetto BookKeeper prima e dopo l'applicazione delle tecniche

- I valori medi mostrano chiaramente i risultati ottenuti per Precision, ovvero l'abbassamento dei valori per tutti i classificatori

Analisi di Kappa per BookKeeper

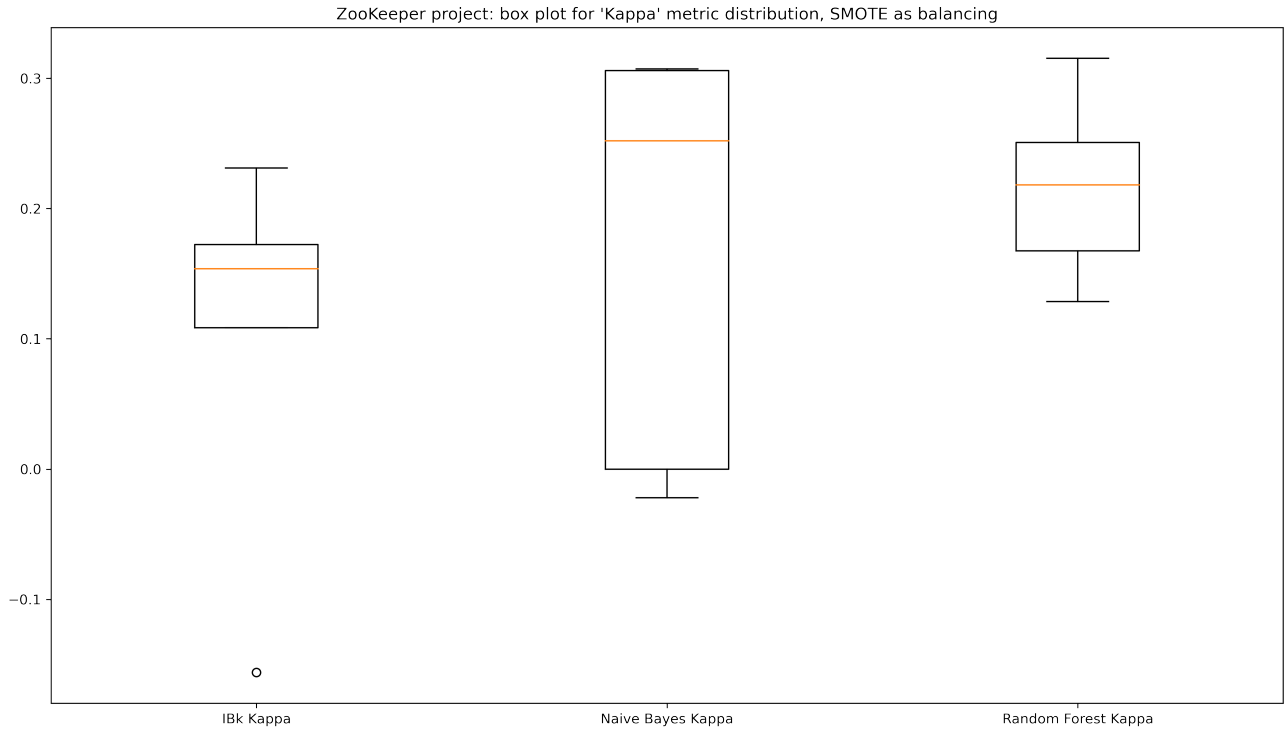
- Le distribuzioni dei valori per la metrica kappa, per tutti i classificatori, sul progetto BookKeeper senza l'utilizzo delle tecniche viste in precedenza sono mostrati nel box plot sottostante



- I classificatori che mostrano la distribuzione dei valori migliori sono IBk e Random Forest
- Tutti i classificatori presentano dei valori che risultano, per determinate run, valori peggiori o uguali ad un classificatore random, quindi il miglioramento è stato concentrato su tale classificatore

Analisi di Kappa per BookKeeper

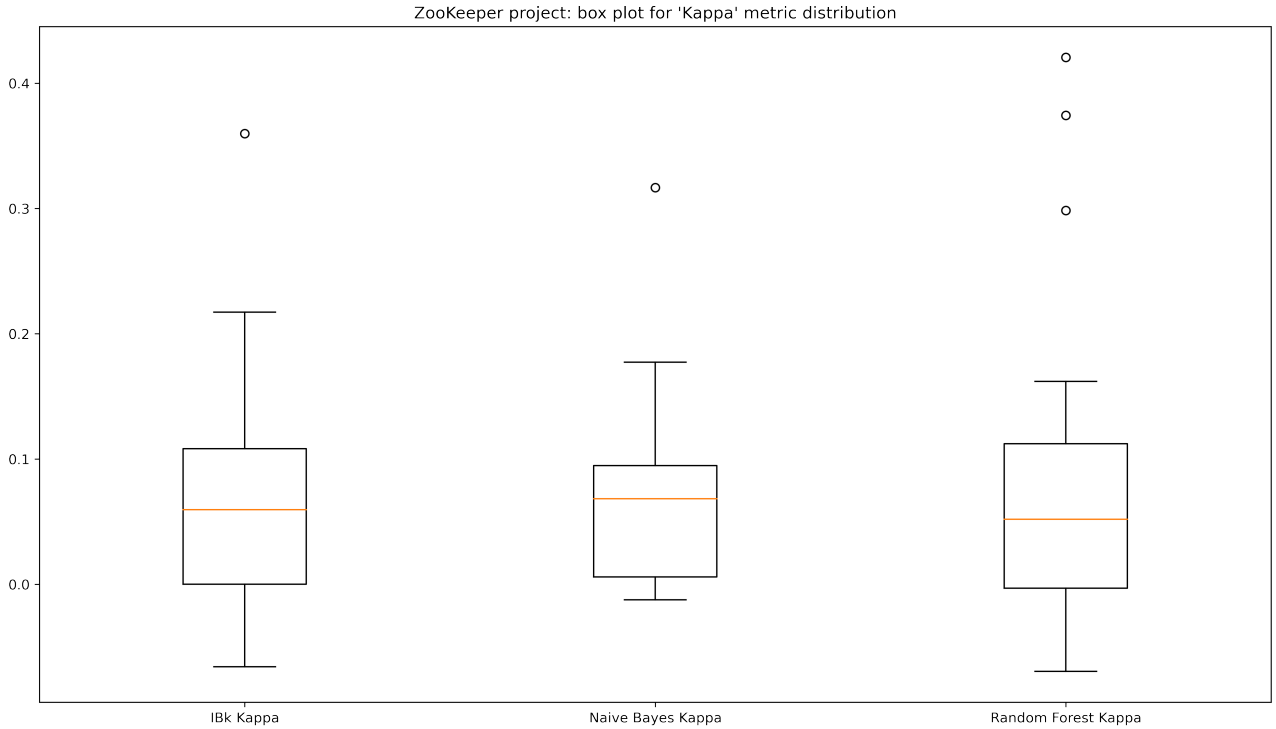
- Applicando SMOTE come filtro per il sampling e feature selection, la distribuzione dei valori per la kappa ottenuta da Naive Bayes migliorano di molto, così come anche quelli di Random Forest



- Per IBk invece, viene riscontrato un peggioramento dei valori della distribuzione

Analisi di Kappa per ZooKeeper

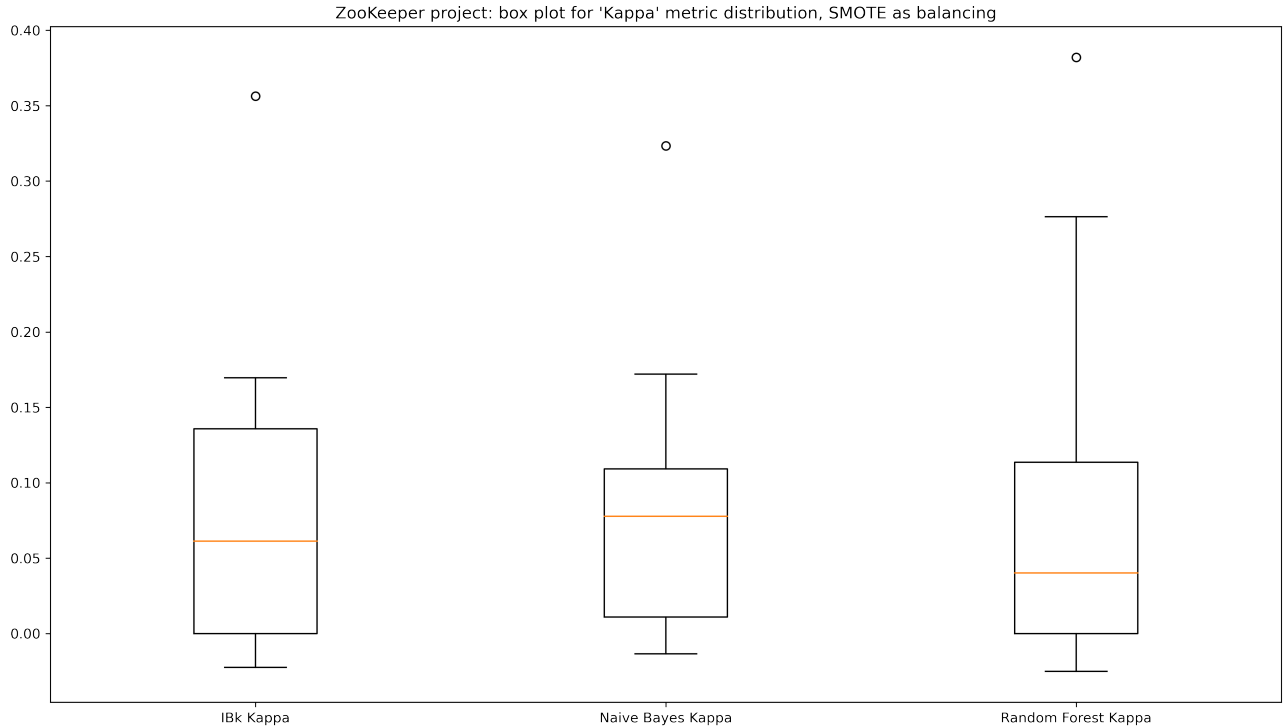
- Per il progetto Zookeeper, senza l'applicazione di alcuna tecnica, si ottengono i seguenti valori



- Per tutti i classificatori, ci sono valori della distribuzione che risultano peggiori di quanto si avrebbe con un classificatore random
- L'obiettivo è quello di cercare di migliorare i valori per tutti e 3 i classificatori

Miglioramento dei valori

- Applicando SMOTE come filtro per il balancing, si ottiene un modesto miglioramento nei valori per IBk e per Naive Bayes



- La distribuzione presenta alcuni valori negativi, quindi per le relative run i classificatori si comporta peggio fi uno random, ma questo è nuovamente dovuto al dataset usato per il testing set, che presenta pochi valori positivi

Medie dei valori per AUC

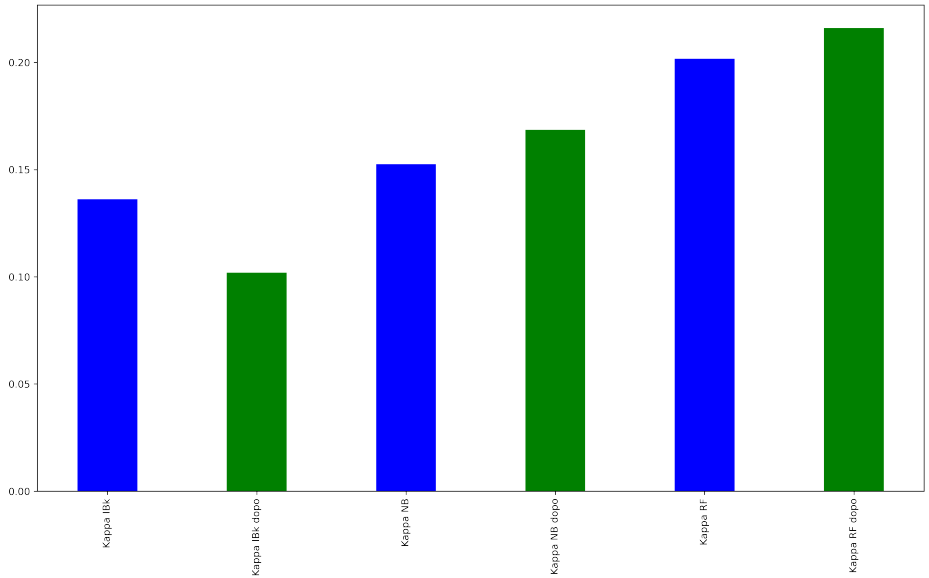


Figure: Confronto dei valori di Kappa per il progetto BookKeeper

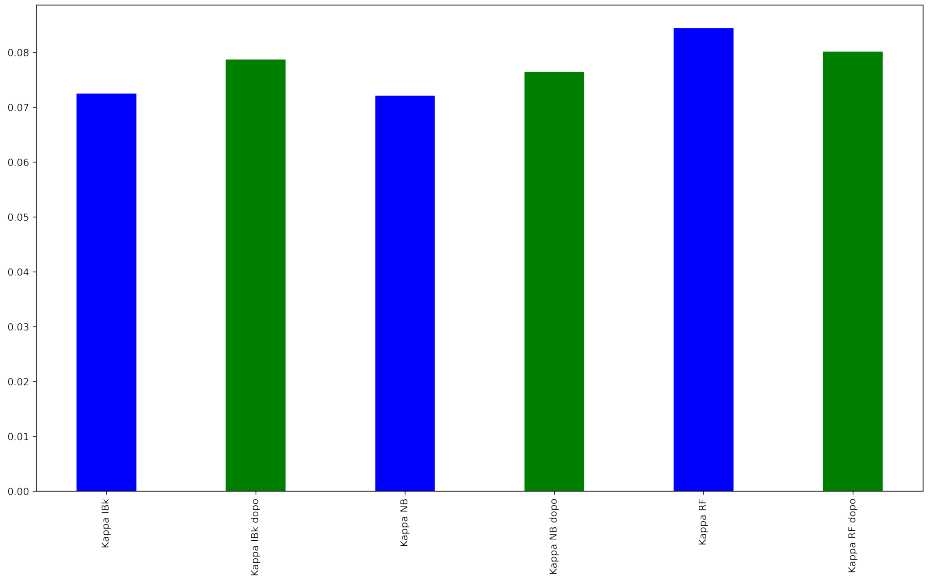


Figure: Confronto dei valori di Kappa per il progetto ZooKeeper