MACHINE LEARNING COME SUPPORTO AL TESTING

MARTINA LUPINI 0344256

CONTENUTI

INTRODUZIONE

L’attività di testing risulta essere dispensiosa in termini di tempo e denaro. Tuttavia è indispensabile per garantire la qualità del software.

23%

budget IT annuale speso in testing*

2.8 MILIARDI

persi dalle aziende software nel 2020 a causa di bassa qualità software**

20-40%

percentuale di tempo speso nel testing durante lo sviluppo software***

*https://www.statista.com/statistics/500641/worldwide-qa-budget-allocation-as-percent-it-spend/
**https://www.it-cisq.org/the-cost-of-poor-software-quality-in-the-us-a-2020-report/
***https://techjury.net/blog/software-testing-statistics/

INTRODUZIONE

Adottare una strategia risk-based permette l'ottimizzazione del testing

Nel lavoro di oggi valuteremo l'applicazione di modelli di machine learning per la predizione della buggyness di una classe

Lo studio si è basato sull'analisi delle performance dei classificatori su due progetti Apache: Avro e Bookkeeper

METODOLOGIA

Si parte dall'assunzione che il futuro sia simile al passato.

Studiando le caratteristiche delle classi buggy rilevate nel passato è possibile predirre se una classe sia buggy o meno.

STEP 1

Raccolta dati

STEP 2

Selezione metriche

STEP 3

Scelta modelli e tecniche ML

STEP 4

Valutazione dei modelli

RACCOLTA DATI

Tramite Jira sono stati recuperati tutti i ticket
type= bug, status=closed (o resolved) e resolution=fixed

↳ i ticket sono importanti perchè hanno informazioni riguardo alle versioni

Da Git sono stati recuperati tutti i commit dei due progetti.
Tra questi sono stati rilevati i commit relativi ai ticket estratti sopra

↳ i commit sono importanti perchè ci permettono di capire quali classi sono state modificate

RACCOLTA DATI - PROPORTION

I ticket hanno informazioni riguardo:

- OV=Opening Version
- FV=Fixed Version
- AV=Affected Version(s)

Se le AV non sono presenti è possibile ricavare IV applicando la tecnica della proportion*:

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

Le variazioni usate sono:

- increment
- cold start (qualora non si avessero abbastanza dati per utilizzare increment)

*https://dl.acm.org/doi/abs/10.1145/3433928

METRICHE

Calcolate per ogni release

- LOC: linee di codice
- LOC TOUCHED: somma delle linee modificate nelle revisioni
- NUMBER OF REVISIONS: numero di revisioni
- NUMBER OF FIXES: numero di difetti di cui è stata fatta la fix
- NUMBER OF AUTHORS: numero di autori
- LOC ADDED: linee aggiunte nelle revisioni
- MAX LOC ADDED: numero massimo di linee aggiunte nelle revisioni
- AVG LOC ADDED: numero medio di linee aggiunte nelle revisioni
- CHURN: valore assoluto della differenza tra linee aggiunte e rimosse
- MAX CHURN: churn massimo nelle revisioni
- AVG CHURN: churn medio nelle revisioni
- COMMENT LINES: numero di linee di commenti (metrica personalizzata)

METRICHE - variabile target

La varibile target è buggy e viene calcolata nel seguente modo nel TESTING SET:

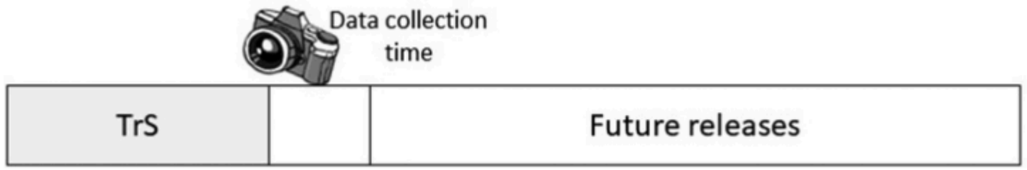
- si considerano tutti i ticket fino ad oggi e a partire dai commit ad essi associati si vanno a recuperare i nomi delle classi modificate. Ogni classe toccata viene labellata come buggy (in tutte le AV).

- il testing set deve essere il più corretto possibile.

METRICHE - variabile target

La varibile target è buggy e viene calcolata nel seguente modo nel TRAINING SET:

- si usa la stessa procedura del testing set ma si considerano solo i ticket la cui FV è minore o uguale all'ultima release del training set.
- il training set deve essere realistico
- è affetto da snoring.



MODELLI E TECNICHE

CLASSIFICATORI	BALANCING	FEATURE SELECTION	COST SENSITIVE CLASSIFIERS
RANDOM FOREST	SMOTE	BEST FIRST (BACKWARD)	THRESHOLD 0.91
IBK			
NAIVE BAYES			

I classificatori sono stati valutati:

- senza l'applicazione di tecniche
- applicando le tecniche di sampling, feature selection e cost sensitive singolarmente
- applicando insieme le tecniche di feature selection e sampling
- applicando insieme le tecniche di feature selection e cost sensitive

MODELLI E TECNICHE - feature selection

Per quanto riguarda la feature selection possiamo vedere tramite Weka GUI quali sono gli attributi selezionati:

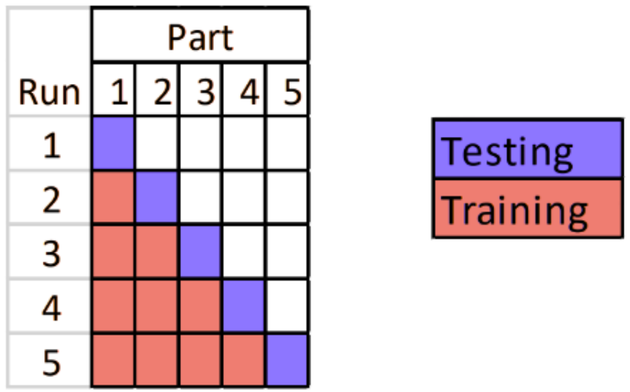
```
=== Attribute Selection on all input data ===

Search Method:
        Best first.
        Start set: 1,2,3,4,5,6,7,8,9,10,11,12,
        Search direction: backward
        Stale search after 5 node expansions
        Total number of subsets evaluated: 83
        Merit of best subset found:    0.134

Attribute Subset Evaluator (supervised, Class (nominal): 13 buggy):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 1,2,3 : 3
                     LOC
                     num_comments
                     num_revisions
```

In entrambi i progetti le feature selezionate sono:

- **LOC: cattura la complessità della classe.**
- **num comments: cattura quanto il codice è stato studiato.**
- **num revisions: cattura l'"instabilità" della classe.**

VALUTAZIONE MODELLI

Come tecnica di valutazione è stata utilizzata walk-forward in quanto è una tecnica time-series (ossia preserva l'ordine temporale).



Per limitare il più possibile lo snoring:

- **l'ultima metà delle release non viene considerata**
- **il testing set parte dalla release 3**
- **vengono considerati solo le classi buggy dell'ultima release del training set (quella più affetta da snoring)**

VALUTAZIONE MODELLI

Per ogni classificatore sono state valutate le seguenti metriche:

- COSTO
- AUC
- F1
- NPOFB30

NPofB30 indica la percentuale di classi buggy trovate ispezionando il 30% delle linee di codice.
Le classi sono ordinate in base alla probabilità di predizione normalizzata sulla size della classe.
Per calcolarlo si è fatto uso del tool ACUME.

RISULTATI BOOKKEEPER - Costo

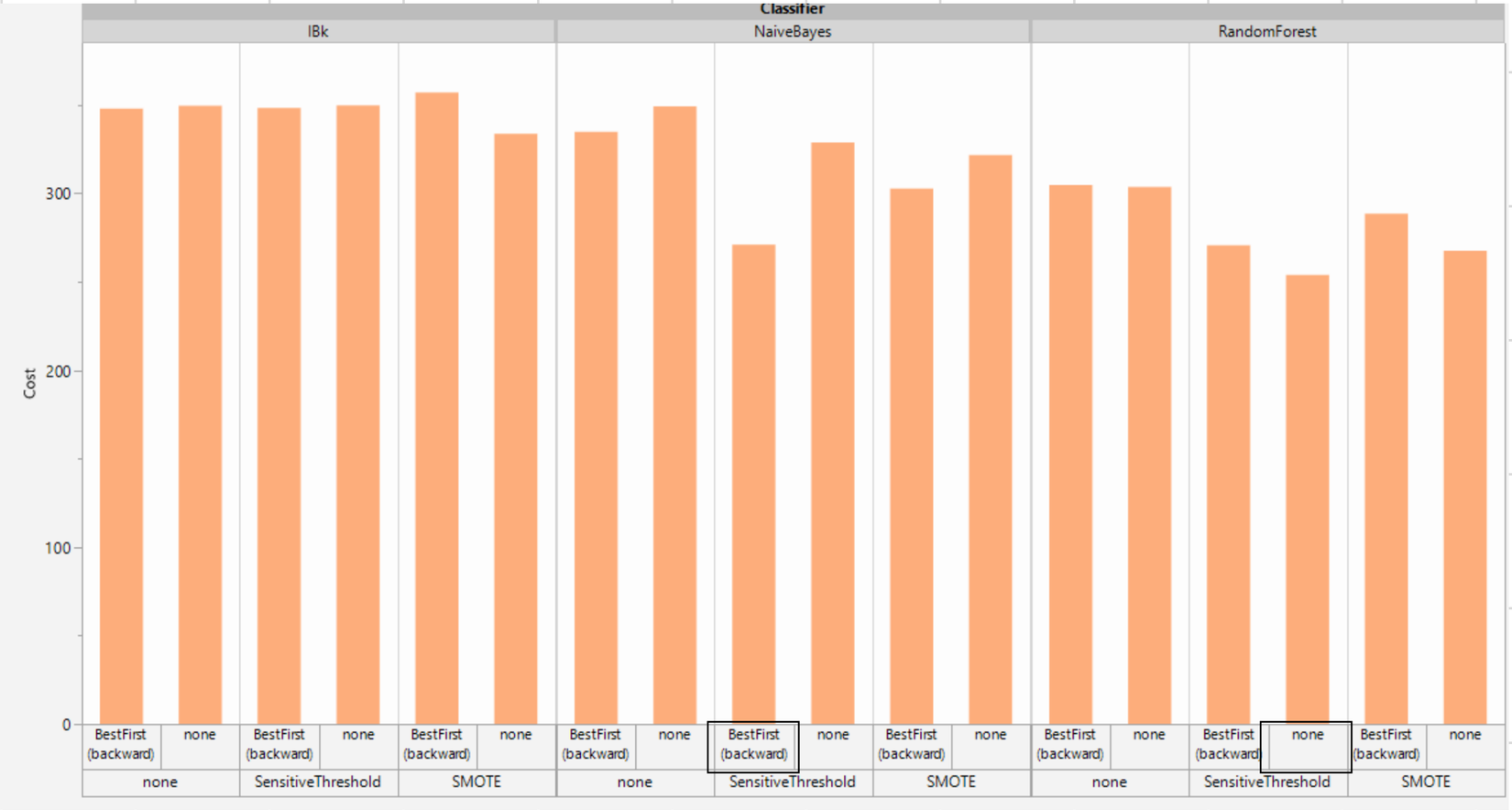


Fig: costo medio (asse y) per classificatore e tecniche applicate (asse x)

Tra tutti i classificatori quelli con costo minore sono Random Forest con sensitive threshold e Naive Bayes con sensitive threshold e feature selection.

RISULTATI BOOKKEEPER - F1 e AUC

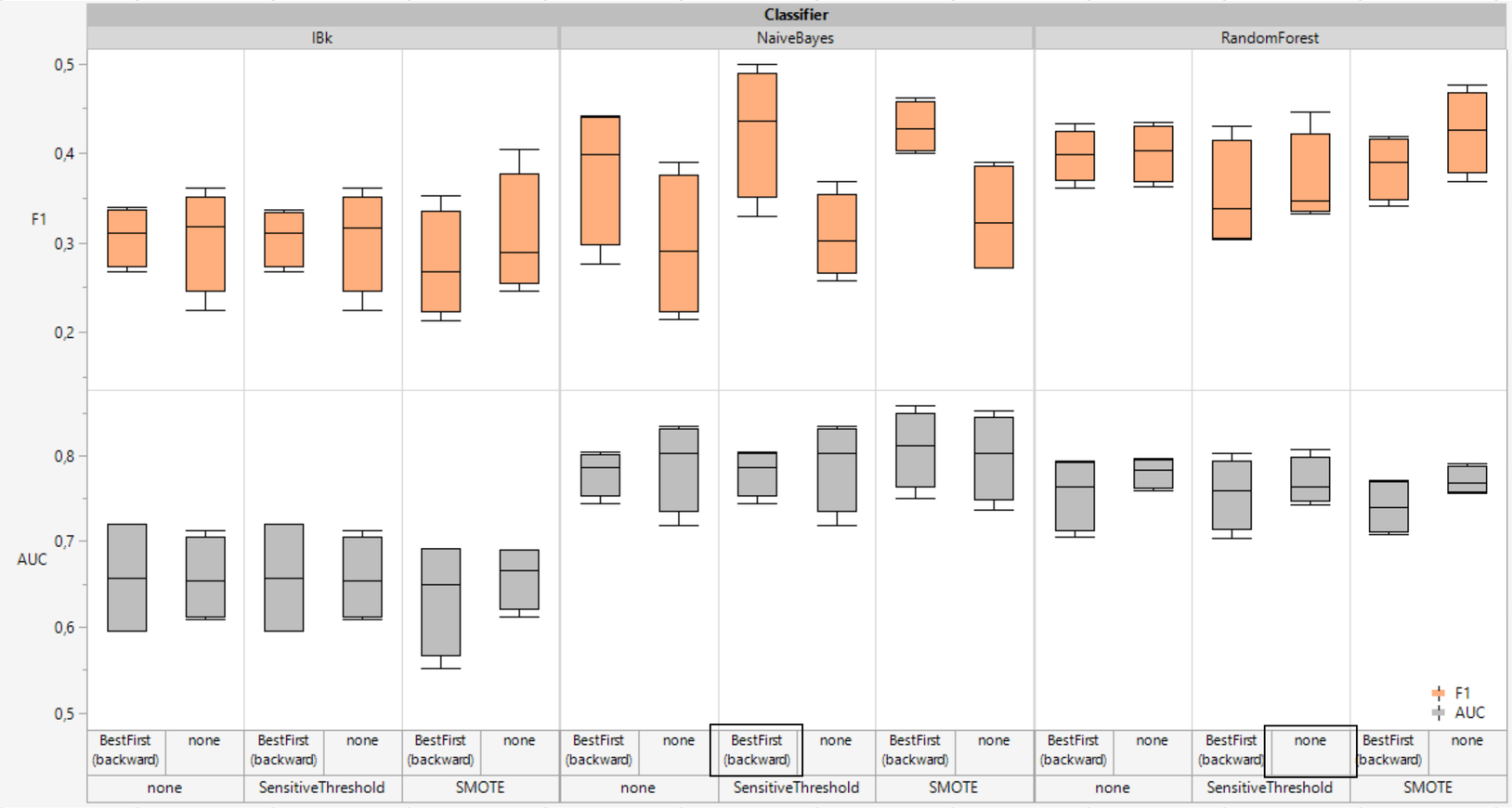
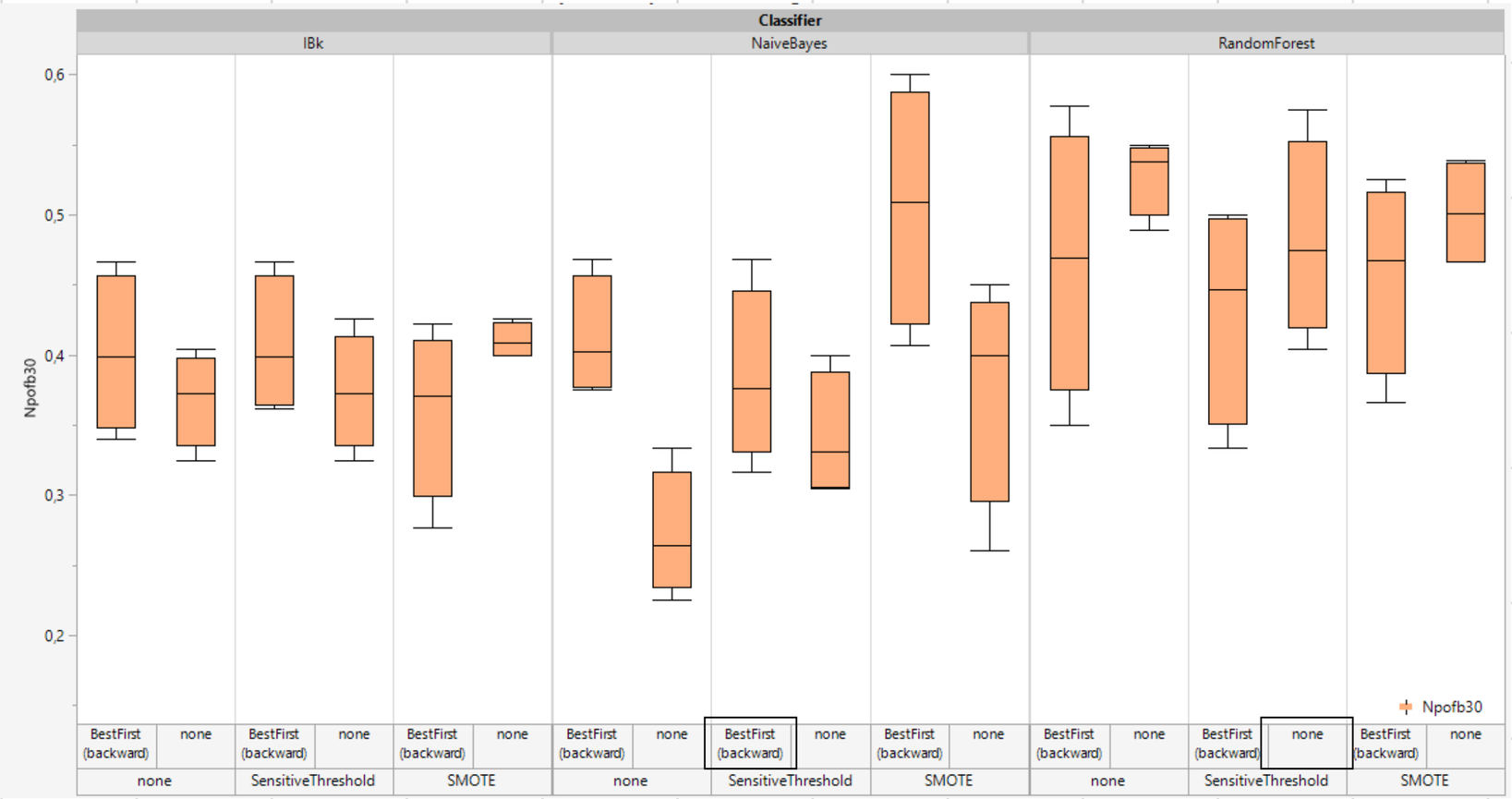


Fig: Distrubuzione dei valori AUC e F1 (asse y) per classificatore e tecniche applicate (asse x)

Confrontando F1 notiamo che tra i due Naive Bayes risulta migliore.

Le AUC sono molto simili anche se Naive Bayes risulta comunque avere una media migliore.

RISULTATI BOOKKEEPER - NPofB30



Random Forest risulta avere un NpofB30 migliore.

Fig: Distribuzione valori NPofB30 (asse y) per classificatore e tecniche applicate (asse x)

RISULTATI AVRO - costo

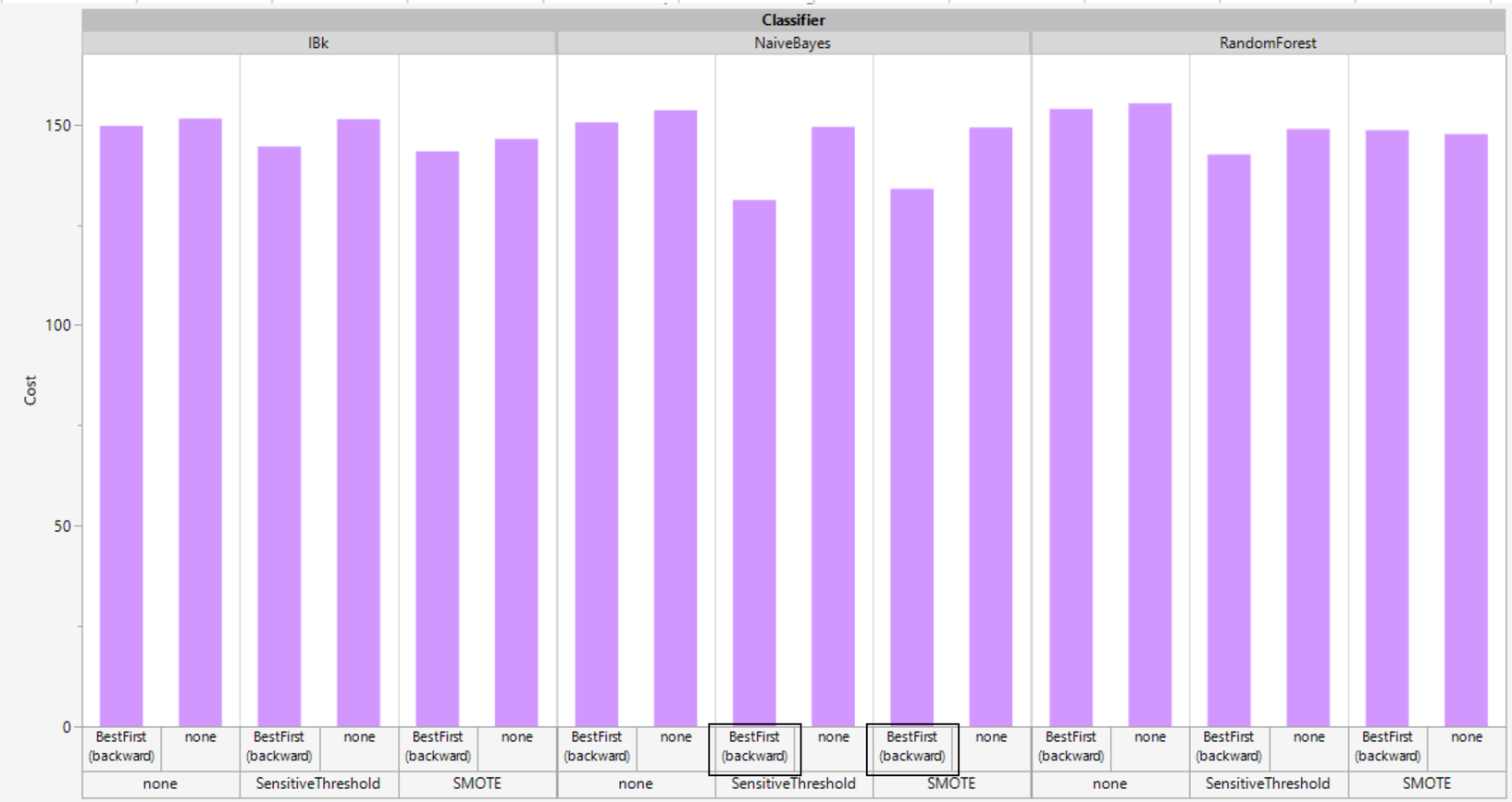


Fig: costo medio (asse y) per classificatore e tecniche applicate (asse x)

Tra tutti i classificatori quelli con costo minore sono Naive Bayes con sensitive threshold e SMOTE (entrambi con feature selection)

RISULTATI AVRO - F1 e AUC

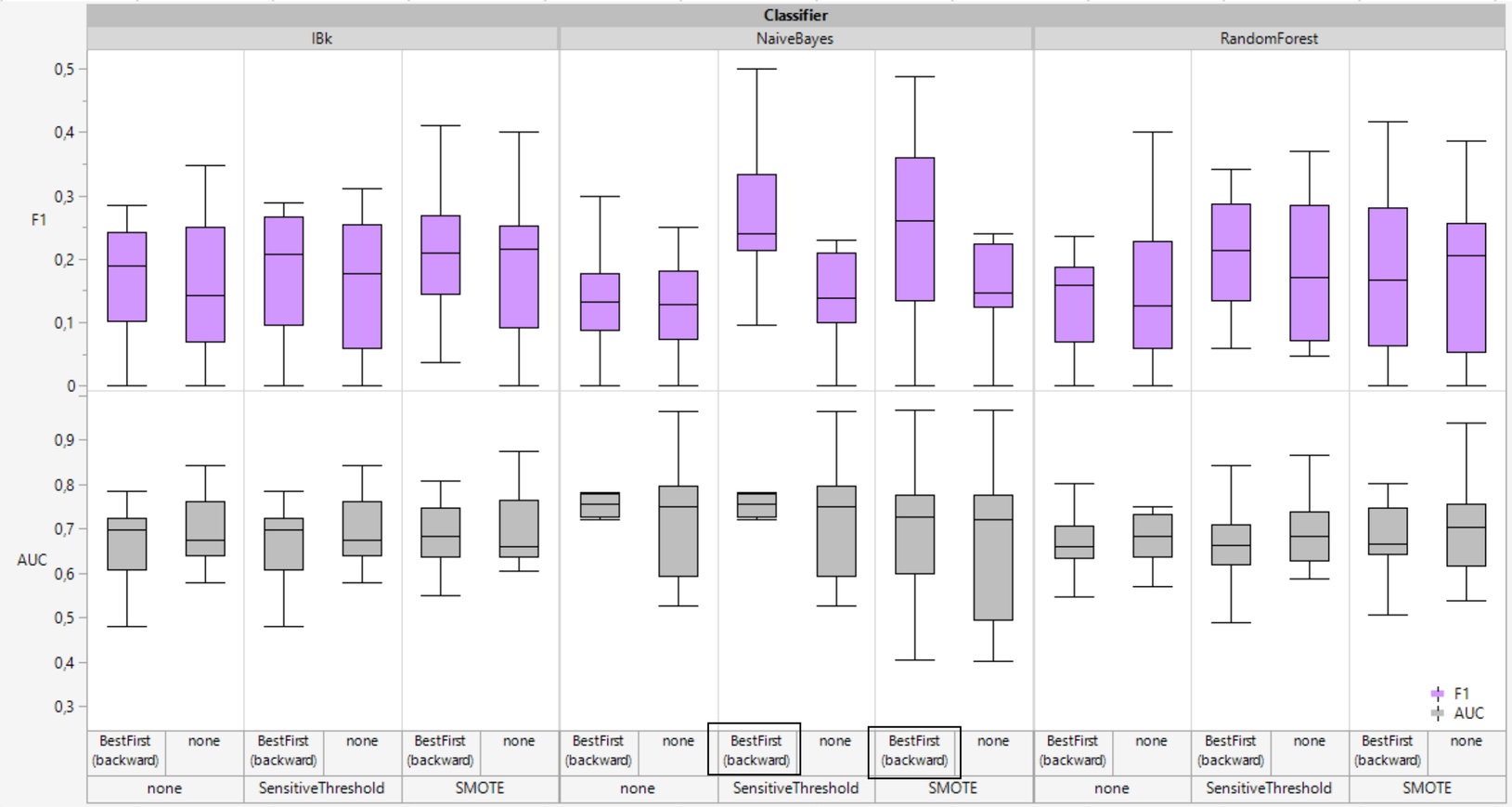


Fig: Distrubuzione dei valori AUC e F1 (asse y) per classificatore e tecniche applicate (asse x)

Le AUC e F1 hanno media simile ma il classificatore con sensitive threshold ha molta meno variabilità dalla media.

RISULTATI AVRO - NPofB30

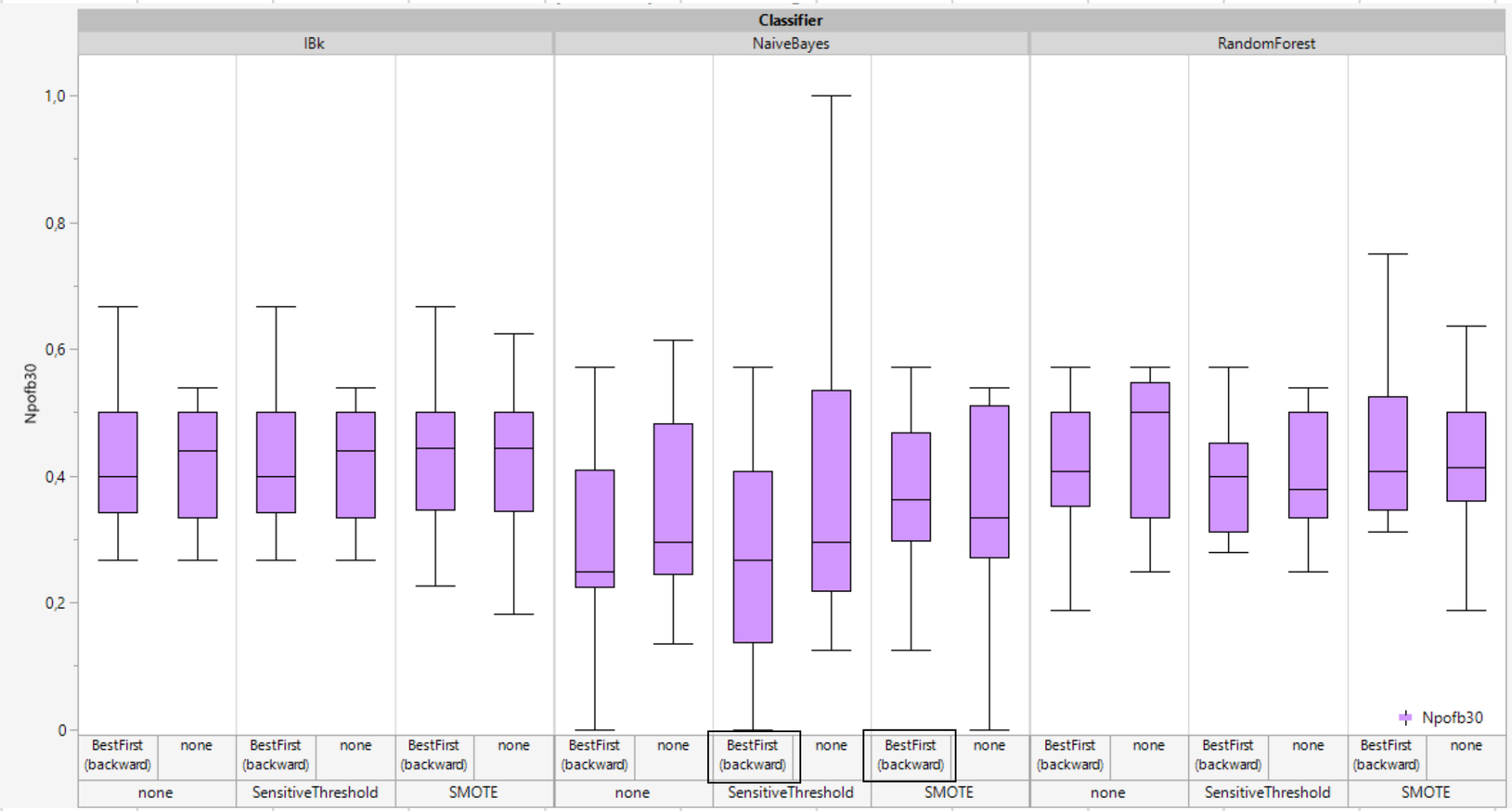
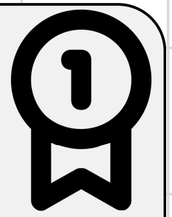


Fig: Distribuzione valori NPofB30 (asse y) per classificatore e tecniche applicate (asse x)

Tra sensitive threshold e SMOTE il secondo sembra essere migliore confrontando le medie.

CONCLUSIONI

BOOKKEEPER



Tra i classificatori si distingue Random Forest con sensitive threshold in termini di costi e NPofB30

AVRO

Tra i classificatori si preferisce Naive Bayes con SMOTE e feature selection, in quanto oltre ad avere un costo basso ha ottimi risultati in F1, AUC e NPofB30

CONCLUSIONI

Tuttavia, in ottica di un'applicabilità futura ad un nuovo progetto la scelta consigliata sarebbe Naive Bayes con SMOTE e feature selection in quanto riporta costi contenuti e degli ottimi valori per le metriche prese in considerazione in entrambi i progetti.
Inoltre, dal momento che utilizza feature selection si ha un'ulteriore riduzione dei costi.

CONCLUSIONI - approssimazioni

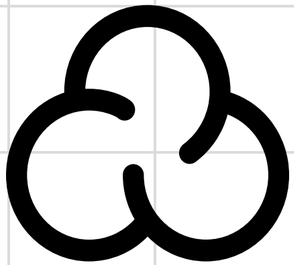
Le date di creazione e risoluzione dei ticket potrebbero non essere accurate

I ticket etichettati come “bug” potrebbero non essere effettivamente bug

Increment è un approccio conservativo e potrebbe sottostimare le performance dei classificatori

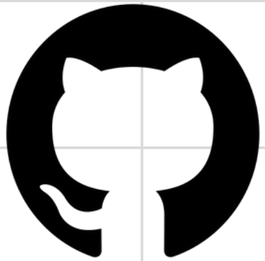
CONCLUSIONI

Potete trovare i riferimenti del progetto su:



Sonarcloud:

https://sonarcloud.io/project/overview?id=martinalupini_BugginessPredictor_ISW2



Github:

https://github.com/martinalupini/BugginessPredictor_ISW2

GRAZIE PER L'ATTENZIONE !