Machine Learning for Software Engineering

Matteo Conti - 0323728

Università degli Studi di Roma Tor Vergata

Ingegneria del Software 2 - A.A. 22/23

Indice

Introduzione-Contesto

Nell'ambito dell'industria IT è fondamentale garantire qualità ed affidabilità del prodotto software, questo tipicamente viene fatto tramite attività di V&V e QA.

Problema:

Il software non può essere testato in maniera totalmente esaustiva per motivi di tempo e budget.
Su quali parti del software indirizzare il budget delle attività di V&V e QA?

Soluzione:

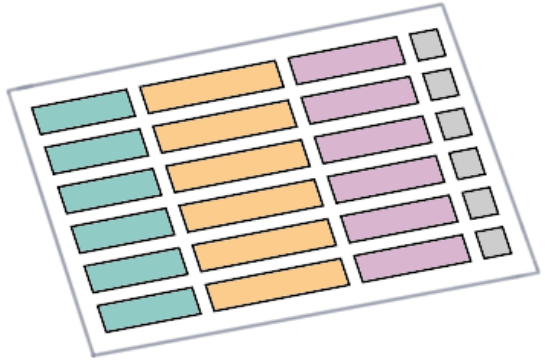
Applicare tecniche di machine learning per predire la difettosità delle varie componenti software e concentrare le attività di V&V e QA sulle componenti più a rischio di essere difettose.

Introduzione-Obbiettivi

L’obbiettivo di tale progetto è, nell’ambito dei software Apache BookKeeper ed Apache ZooKeeper, identificare quali classificatori e quali tecniche si comportano meglio al fine di predire la difettosità delle classi Java che compongono tali software.

Metodologia-Costruzione del dataset (1)

Per raggiungere gli obbiettivi della nostra analisi è necessario avere un dataset che, per ogni release, possiede uno snapshot delle classi che compongono il software al termine di tale release.
Le feature associate ad ogni classe devono caratterizzare le proprietà che sono potenzialmente correlate alla difettosità delle stesse (e.g LOC, numero di autori etc.).
E' inoltre necessario avere una label binaria che indica se la classe è difettosa (buggy) o meno nella release considerata.



Per costruire tali feature sono state utilizzate due fonti:

- **Jira**, per la raccolta dei dati riguardanti le release ed i ticket di bug
- **Github**, per la raccolta dei dati riguardanti le commit tramite le Github API

Metodologia-Costruzione del dataset (2)

Le feature sono state ottenute misurando il codice tramite i dati presenti nelle commit e nei ticket, in particolare sono state utilizzate le seguenti metriche:

- **LOC**, cioè la taglia della classe in termini di linee di codice
- **#Authors***, cioè il numero di autori della classe
- **AvgChurn***, cioè la media della differenza tra righe di codice aggiunte e tolte alla classe
- **LOCAdded***, cioè il numero medio di righe di codice aggiunte alla classe
- **#Revision***, cioè il numero di revisioni della classe
- **#BugFix***, cioè il numero di commit che hanno coinvolto la classe il cui messaggio è relativo ad un ticket di bug in Jira

(*) Tali metriche sono presenti in due versioni «in-release» e «from-start» cioè la misurazione è stata effettuata rispettivamente nel contesto della singola release e cumulativamente dalla prima release.

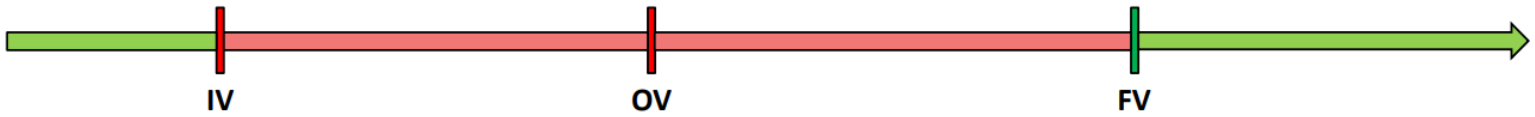
Metodologia-Costruzione del dataset (3)

Per assegnare la label di buggyness alle classi è necessario tenere conto del ciclo di vita di un bug, il quale è caratterizzato dai seguenti istanti temporali:

- **IV** o inject version, cioè la release in cui il bug è stato immesso nel codice
- **OV** o opening version, cioè la release in cui è stato aperto un ticket associato a tale bug
- **FV** o fix version, cioè la release in cui il bug viene risolto ed il ticket ad esso associato chiuso

Le release tra la IV e FV vengono dette **AV** o affected versions, un bug tra la IV e OV viene detto **dormiente** ed una classe che contiene un bug dormiente viene detta affetta da **snoring**.

Se una classe in una certa release viene toccata da una commit di fix associata ad un ticket di bug, significa che tale classe fino a quella release era buggy; il problema principale è determinare le AV, le quali non sempre sono presenti nei ticket di Jira, quando esse non sono disponibili viene utilizzata la tecnica **proportion** per cercare di stimare l'IV e da essa ricavare le AV.



Metodologia-Costruzione del dataset (4)

Come accennato in precedenza quando è necessario stimare l'IV utilizziamo proportion, la quale è una tecnica che si basa sull'idea dell'esistenza di una proporzionalità tra il tempo che trascorre tra FV ed IV e quello che trascorre tra FV ed OV, in particolare si definisce proportion come:

$$p = \frac{FV - IV}{FV - OV}$$

La stima del valore di proportion è stata fatta in due modi:

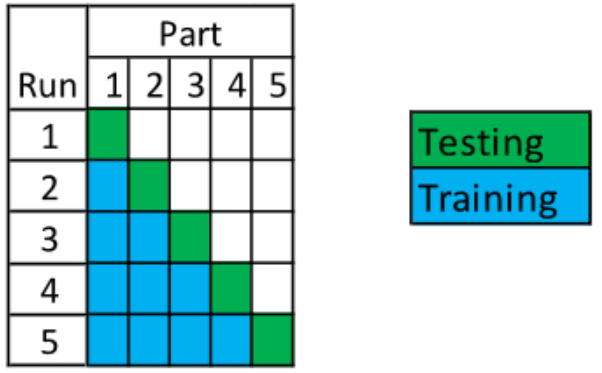
- **Cold start**, cioè P è stata stimata come la sua media in altri progetti simili (Apache Accumulo, Avro, Kafka, Pig, Syncope e Tajo). Questa P è stata utilizzata nelle prime 2 release e nelle release con meno di 5 ticket a disposizione
- **Incremental**, cioè P è stata stimata come la sua media nelle release precedenti a quella che si sta analizzando correntemente (e.g in release 3 considero i dati delle release 1 e 2)

Metodologia-Costruzione del dataset (5)

Quanto descritto fino ad ora è stato utilizzato per creare un insieme di N (numero di release) dataset che rispecchiano la tecnica di validazione utilizzata, cioè **walk forward.**

In particolare il dataset k-esimo è costruito nel seguente modo:

- I dati sulle classi nelle prime k-1 release fanno parte del training set mentre i dati delle classi nella release k-esima costituiscono il testing set
- Per assegnare le label del training set vengono utilizzati solo i dati disponibili fino alla release k-esima, in modo da avere un training set realistico
- Per assegnare le label del testing set vengono utilizzati tutti i dati disponibili fino alla release N-esima, in modo da avere un testing set accurato



Metodologia-Classificatori e tecniche

Il problema che stiamo tentando di risolvere è un problema di classificazione binaria, a tale scopo i classificatori utilizzati sono:

- **IBK**
- **Naive Bayes**
- **Random Forest**

Oltre al classificatore «vanilla» sono state utilizzate diverse tecniche per cercare di migliorare le performance dello stesso, in particolare sono state usate le tecniche:

- **Best first bidirectional** come feature selection
- **Under sampling** ed **Over sampling** come balancing
- **Cost sensitive threshold** e **Cost sensitive learning** (CFN=10*CFP) come cost sensitive classifiers

Le suddette tecniche sono state utilizzate sia singolarmente che combinate tra di loro, in particolare è stata combinata la feature selection con il balancing ed i cost sensitive classifiers.

Risultati-Metriche

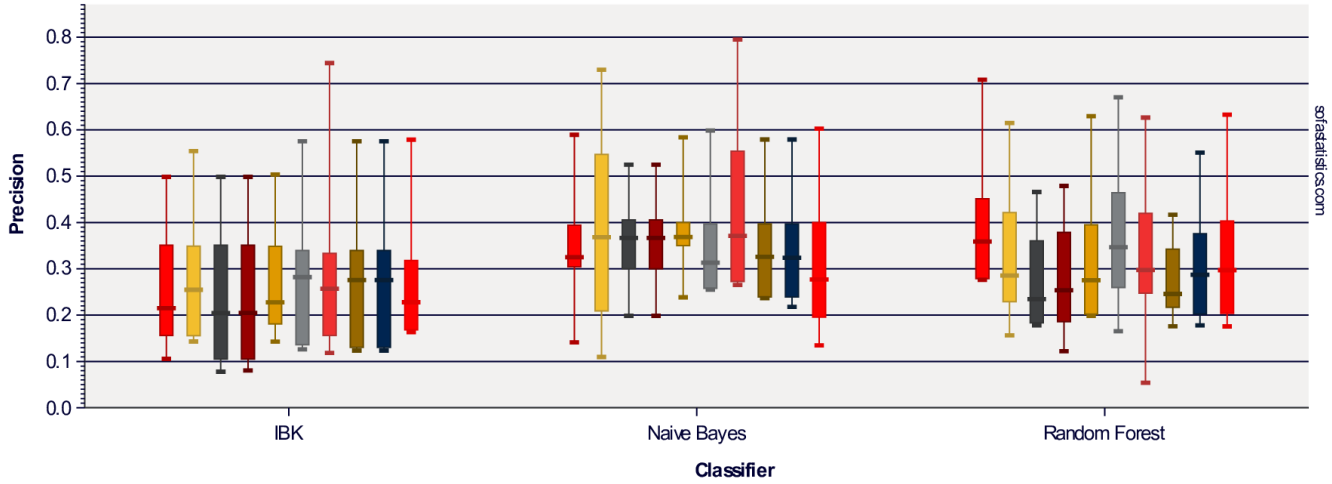
Quanto descritto fino ad ora è stato implementato in Java utilizzando le API del tool Weka, di seguito sono riportati i risultati ottenuti negli esperimenti.

Le metriche di accuratezza considerate negli esperimenti sono le seguenti:

- Kappa
- Precision
- Recall
- AUC

Risultati-BookKeeper(1)

Precision Bookkeeper




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

Search Method:
        Best first.
        Start set: no attributes
        Search direction: bi-directional
        Stale search after 5 node expansions
        Total number of subsets evaluated: 99
        Merit of best subset found:    0.139

Attribute Subset Evaluator (supervised, Class (nominal): 11 Buggy):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 2,3,5,7,9 : 5
                     LOC
                     AvgChurnInRelease
                     AvgLOCAddedInRelease
                     #RevisionInRelease
                     #BugFixInRelease
```

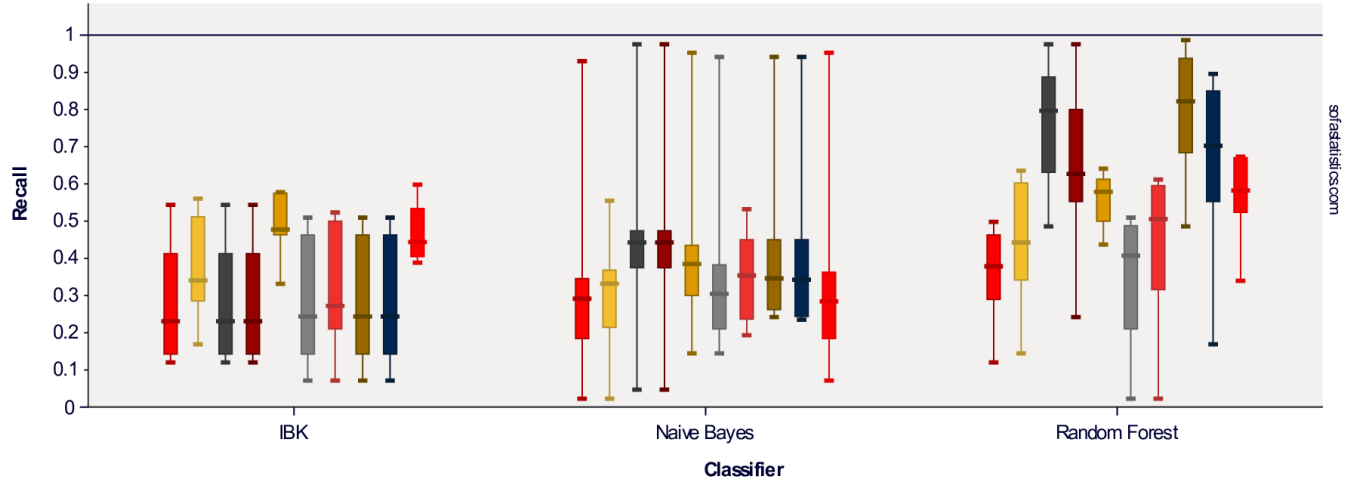
Addons:

Best_First_Bidirectional | Best_First_Bidirectional+Over_Sampling | Best_First_Bidirectional+Sensitive_Classifier_Threshold_CFP_1_CFN_10 | Best_First_Bidirectional+Sensitive_Learning_CFP_1_CFN_10 | Best_First_Bidirectional+Under_Sampling | None | Over_Sampling | Sensitive_Classifier_Threshold_CFP_1_CFN_10 | Sensitive_Learning_CFP_1_CFN_10 | Under_Sampling

Whiskers are at the minimum and maximum values

Risultati-BookKeeper(2)

Recall Bookkeeper




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

Search Method:
        Best first.
        Start set: no attributes
        Search direction: bi-directional
        Stale search after 5 node expansions
        Total number of subsets evaluated: 99
        Merit of best subset found:    0.139

Attribute Subset Evaluator (supervised, Class (nominal): 11 Buggy):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 2,3,5,7,9 : 5
                     LOC
                     AvgChurnInRelease
                     AvgLOCAddedInRelease
                     #RevisionInRelease
                     #BugFixInRelease
```

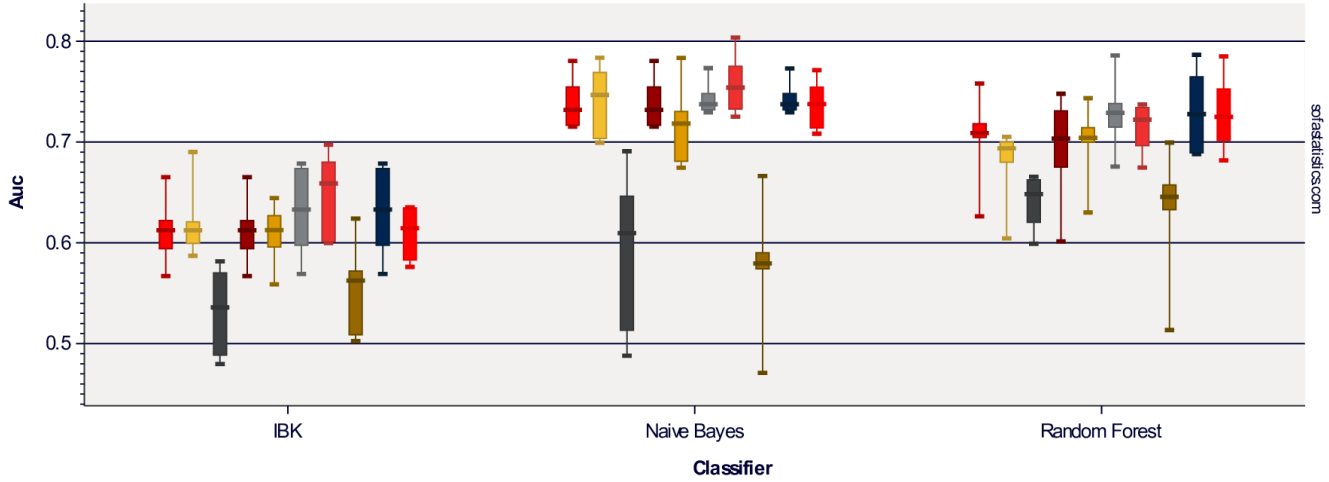
Addons:

Best_First_Bidirectional	Best_First_Bidirectional+Over_Sampling	Best_First_Bidirectional+Sensitive_Classifier_Threshold_CFP_1_CFN_10	Best_First_Bidirectional+Sensitive_Learning_CFP_1_CFN_10
Best_First_Bidirectional+Under_Sampling	None	Over_Sampling	Sensitive_Classifier_Threshold_CFP_1_CFN_10
Sensitive_Learning_CFP_1_CFN_10	Under_Sampling		

Whiskers are at the minimum and maximum values

Risultati-BookKeeper(3)

AUC Bookkeeper




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

Search Method:
        Best first.
        Start set: no attributes
        Search direction: bi-directional
        Stale search after 5 node expansions
        Total number of subsets evaluated: 99
        Merit of best subset found:    0.139

Attribute Subset Evaluator (supervised, Class (nominal): 11 Buggy):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 2,3,5,7,9 : 5
                     LOC
                     AvgChurnInRelease
                     AvgLOCAddedInRelease
                     #RevisionInRelease
                     #BugFixInRelease
```

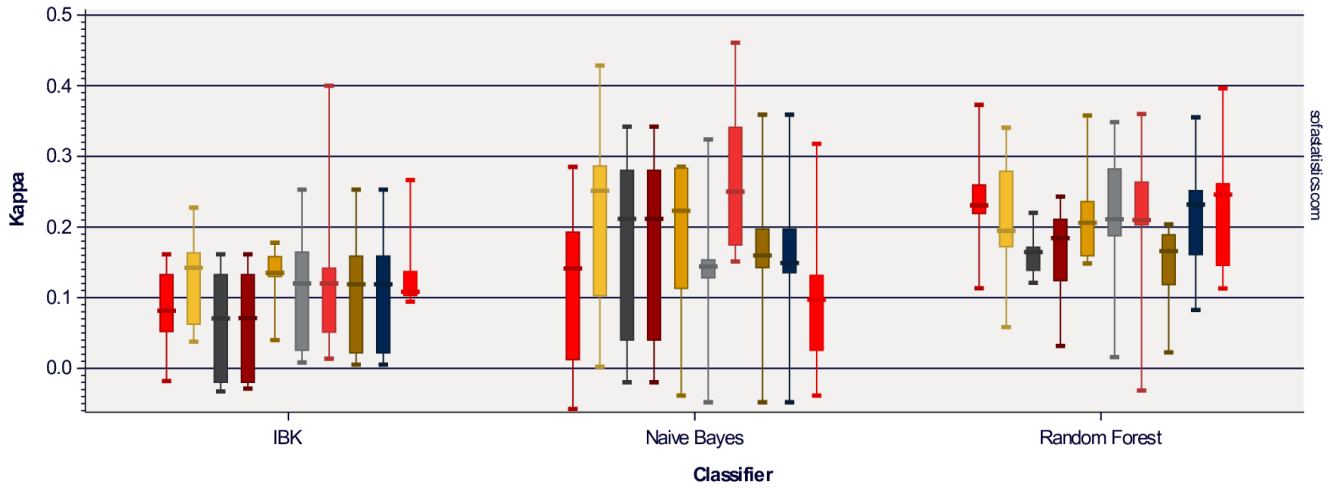
Addons:

Best_First_Bidirectional | Best_First_Bidirectional+Over_Sampling | Best_First_Bidirectional+Sensitive_Classifier_Threshold_CFP_1_CFN_10 | Best_First_Bidirectional+Sensitive_Learning_CFP_1_CFN_10 | Best_First_Bidirectional+Under_Sampling | None | Over_Sampling | Sensitive_Classifier_Threshold_CFP_1_CFN_10 | Sensitive_Learning_CFP_1_CFN_10 | Under_Sampling

Whiskers are at the minimum and maximum values

Risultati-BookKeeper(4)

Kappa Bookkeeper




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

Search Method:
        Best first.
        Start set: no attributes
        Search direction: bi-directional
        Stale search after 5 node expansions
        Total number of subsets evaluated: 99
        Merit of best subset found:    0.139

Attribute Subset Evaluator (supervised, Class (nominal): 11 Buggy):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 2,3,5,7,9 : 5
                     LOC
                     AvgChurnInRelease
                     AvgLOCAddedInRelease
                     #RevisionInRelease
                     #BugFixInRelease
```

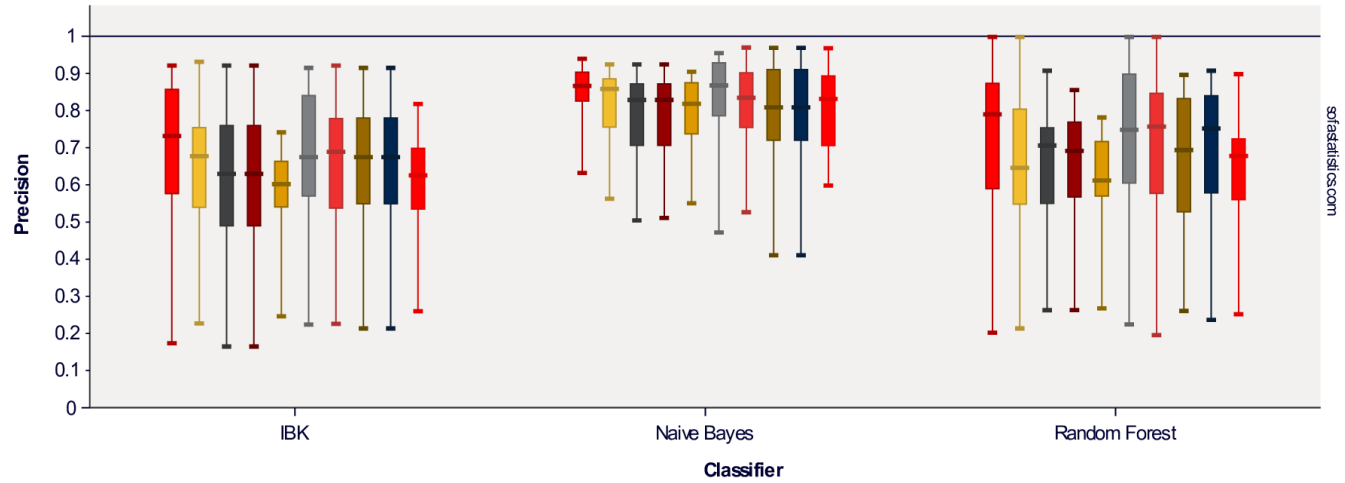
Addons:

Best_First_Bidirectional | Best_First_Bidirectional+Over_Sampling | Best_First_Bidirectional+Sensitive_Classifier_Threshold_CFP_1_CFN_10 | Best_First_Bidirectional+Sensitive_Learning_CFP_1_CFN_10 | Best_First_Bidirectional+Under_Sampling | None | Over_Sampling | Sensitive_Classifier_Threshold_CFP_1_CFN_10 | Sensitive_Learning_CFP_1_CFN_10 | Under_Sampling

Whiskers are at the minimum and maximum values

Risultati-ZooKeeper(1)

Precision Zookeeper




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

Search Method:
        Best first.
        Start set: no attributes
        Search direction: bi-directional
        Stale search after 5 node expansions
        Total number of subsets evaluated: 90
        Merit of best subset found:    0.124

Attribute Subset Evaluator (supervised, Class (nominal): 11 Buggy):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 2,8,9,10 : 4
                     LOC
                     #RevisionFromStart
                     #BugFixInRelease
                     #BugFixFromStart
```

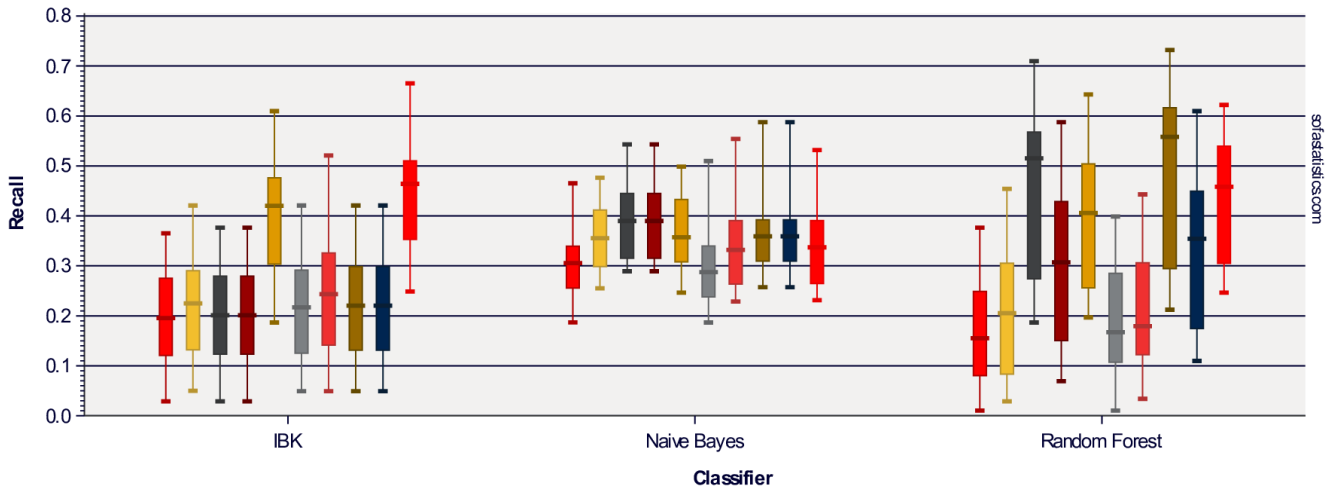
Addons:

Best_First_Bidirectional | Best_First_Bidirectional+Over_Sampling | Best_First_Bidirectional+Sensitive_Classifier_Threshold_CFP_1_CFN_10 | Best_First_Bidirectional+Sensitive_Learning_CFP_1_CFN_10 | Best_First_Bidirectional+Under_Sampling | None | Over_Sampling | Sensitive_Classifier_Threshold_CFP_1_CFN_10 | Sensitive_Learning_CFP_1_CFN_10 | Under_Sampling

Whiskers are at the minimum and maximum values

Risultati-ZooKeeper(2)

Recall Zookeeper




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

Search Method:
        Best first.
        Start set: no attributes
        Search direction: bi-directional
        Stale search after 5 node expansions
        Total number of subsets evaluated: 90
        Merit of best subset found:    0.124

Attribute Subset Evaluator (supervised, Class (nominal): 11 Buggy):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 2,8,9,10 : 4
                     LOC
                     #RevisionFromStart
                     #BugFixInRelease
                     #BugFixFromStart
```

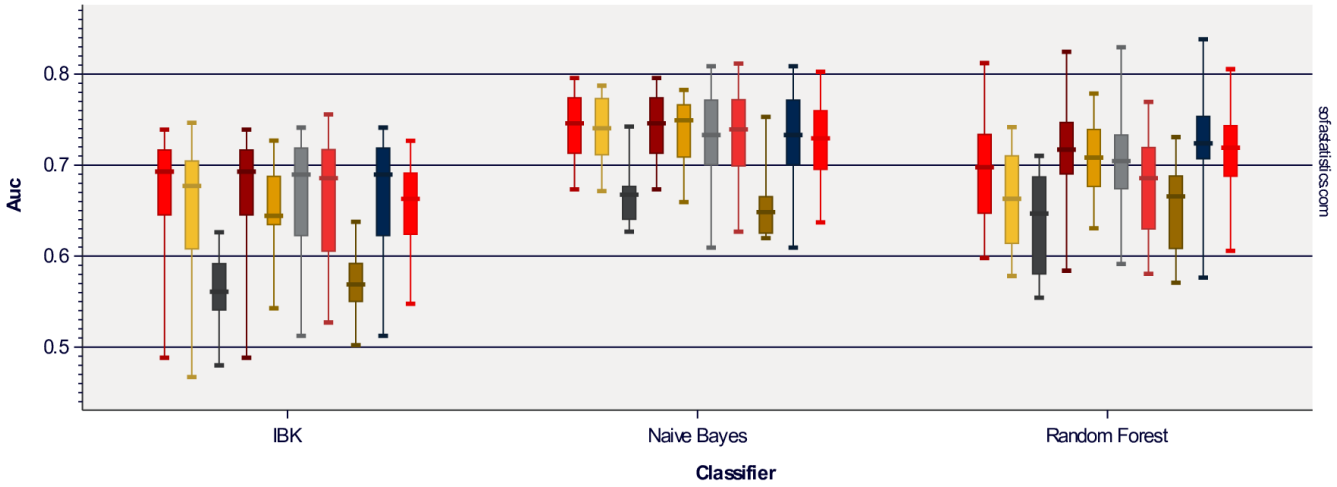
Addons:

Best_First_Bidirectional	Best_First_Bidirectional+Over_Sampling	Best_First_Bidirectional+Sensitive_Classifier_Threshold_CFP_1_CFN_10	Best_First_Bidirectional+Sensitive_Learning_CFP_1_CFN_10
Best_First_Bidirectional+Under_Sampling	None	Over_Sampling	Sensitive_Classifier_Threshold_CFP_1_CFN_10
Sensitive_Learning_CFP_1_CFN_10	Under_Sampling		

Whiskers are at the minimum and maximum values

Risultati-ZooKeeper(3)

AUC Zookeeper




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

Search Method:
        Best first.
        Start set: no attributes
        Search direction: bi-directional
        Stale search after 5 node expansions
        Total number of subsets evaluated: 90
        Merit of best subset found:    0.124

Attribute Subset Evaluator (supervised, Class (nominal): 11 Buggy):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 2,8,9,10 : 4
                     LOC
                     #RevisionFromStart
                     #BugFixInRelease
                     #BugFixFromStart
```

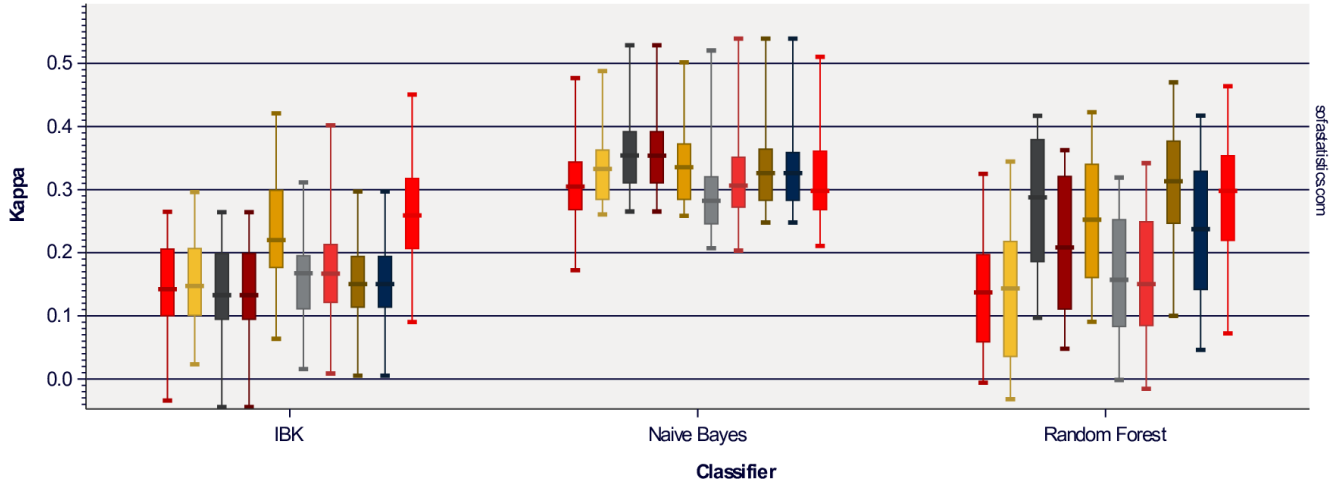
Addons:

Best_First_Bidirectional | Best_First_Bidirectional+Over_Sampling | Best_First_Bidirectional+Sensitive_Classifier_Threshold_CFP_1_CFN_10 | Best_First_Bidirectional+Sensitive_Learning_CFP_1_CFN_10 | Best_First_Bidirectional+Under_Sampling | None | Over_Sampling | Sensitive_Classifier_Threshold_CFP_1_CFN_10 | Sensitive_Learning_CFP_1_CFN_10 | Under_Sampling

Whiskers are at the minimum and maximum values

Risultati-ZooKeeper(4)

Kappa Zookeeper




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

Search Method:
        Best first.
        Start set: no attributes
        Search direction: bi-directional
        Stale search after 5 node expansions
        Total number of subsets evaluated: 90
        Merit of best subset found:    0.124

Attribute Subset Evaluator (supervised, Class (nominal): 11 Buggy):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 2,8,9,10 : 4
                     LOC
                     #RevisionFromStart
                     #BugFixInRelease
                     #BugFixFromStart
```

Addons:

Best_First_Bidirectional	Best_First_Bidirectional+Over_Sampling	Best_First_Bidirectional+Sensitive_Classifier_Threshold_CFP_1_CFN_10	Best_First_Bidirectional+Sensitive_Learning_CFP_1_CFN_10
Best_First_Bidirectional+Under_Sampling	None	Over_Sampling	Sensitive_Classifier_Threshold_CFP_1_CFN_10
Sensitive_Learning_CFP_1_CFN_10	Under_Sampling		

Whiskers are at the minimum and maximum values

Risultati-Commenti

Dai risultati si vede che il classificatore che performa in modo peggiore risulta essere IBK, mentre Random Forest porta alle performance migliori sulla recall e Naive Bayes porta alle performance migliori sulle restanti metriche.

Per quanto riguarda le tecniche di balancing si vede che l'oversampling non è mai particolarmente efficace, mentre l'undersampling migliora significativamente la recall se usato con Random Forest e IBK.

Il classificatore cost sensitive è risultato essere efficace nel migliorare la recall solamente con il classificatore Random Forest. Inoltre, è possibile osservare che le performance ottenute sulla recall con la tecnica cost sensitive threshold risultano essere significativamente migliori delle performance ottenute con la tecnica cost sensitive learning.

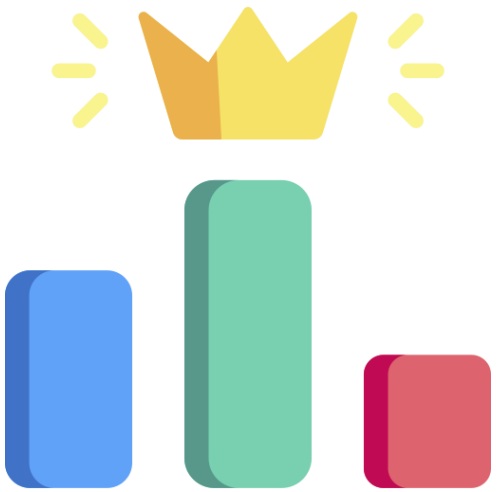
La feature selection da sola è risultata efficace mantenendo o migliorando tutte le metriche di accuratezza se usata con Naive Bayes e solamente la precision se usata con Random Forest. Se combinata con tecniche di sampling ha mantenuto o migliorato la precision e la kappa con IBK e Naive Bayes, mentre se combinata con i cost sensitive classifier ha mantenuto o migliorato tutte le metriche di accuratezza solamente se usata con Naive Bayes.

Conclusioni

Per i nostri scopi la metrica di accuratezza più importante è la recall, in quanto ci interessa principalmente che l'evento di interesse (classe buggy) venga catturato, anche al costo di testare qualche classe in più a causa di un maggior numero falsi positivi; pertanto la configurazione migliore per i nostri scopi risulta essere:

- **Random Forest**
- **Cost Sensitive Threshold**

In particolare questa configurazione ci permette di migliorare significativamente la recall facendo dei compromessi accettabili sulla precision.



Links

Assunzioni

https://github.com/matteo-conti-97/isw2-jira-git-measurement/blob/main/ASSUNZIONI.txt

Github

https://github.com/matteo-conti-97/isw2-jira-git-measurement

SonarCloud

https://sonarcloud.io/summary/overall?id=matteo-conti-97_isw2-jira-git-measurement