MACHINE LEARNING FOR SOFTWARE ENGINEERING: REPORT

Mancini Francesco - 0285816

Sommario

Deliverable 1

Introduzione

L'obiettivo di questa Deliverable è analizzare lo storico dei ticket di una specifica categoria (i.e. "Bug Fix", "New Features" o "Fixed Tickets") di progetti open-source dell'Apache Software Foundation. L'analisi dei tickets avviene tramite l'elaborazione di un grafico di tipo Process Control Chart: tale grafico permette di determinare la stabilità del processo software associato, di controllarlo, di prevederne comportamenti futuri e di individuare problemi (attuali oppure passati), analizzandone le cause (Root Cause Analysis).

In questo report si analizzeranno i ticket di tipo **"Bug Fix"** relativi al progetto **"Mahout"**.

Progettazione

Si è sviluppato un programma Java per collezionare i ticket necessari all'analisi. La sua esecuzione è caratterizzata da tre fasi: nella prima colleziona i ticket dalla piattaforma Jira, utilizzando le API REST offerte dalla stessa; nella fase successiva, il programma ottiene tutti i commit del repository GitHub del progetto da analizzare, e tale operazione è eseguita utilizzando la libreria "jgit". Nell'ultima fase, il programma analizza i commit per individuare la data di chiusura dei ticket raccolti: per ciascun commit, l'eseguibile ricerca all'interno del commento i codici identificativi dei ticket di Jira (i.e. <nome progetto>-<numero ticket>) e utilizza la data associata al commit come data di chiusura del ticket. In caso ci siano più commit che fanno riferimento allo stesso ticket, il programma considera la data del commit più recente come data di chiusura del ticket. Dopo aver collezionato i ticket, il programma li salva in un apposito file in formato csv. *Il programma è stato progettato in modo tale da poter analizzare sia molteplici progetti che qualsiasi categoria di ticket: la selezione dei progetti, e delle tipologie di ticket da analizzare, possono essere scelti tramite un apposito file di configurazione (config.json).*

Risultati

Il progetto "Mahout" ha un totale di 511 ticket "Bug Fix" conclusi: 110 di questi ticket non hanno alcuna data di chiusura e, quindi, sono stati esclusi dalla generazione del Process Control Chart. Tali dati mancanti, che costituiscono il 18% dell'informazione totale [Figura 1], diminuiscono la precisione dell'analisi: il processo, infatti, potrebbe avere dei periodi di instabilità che non possono essere individuati a causa della mancanza di questi dati.

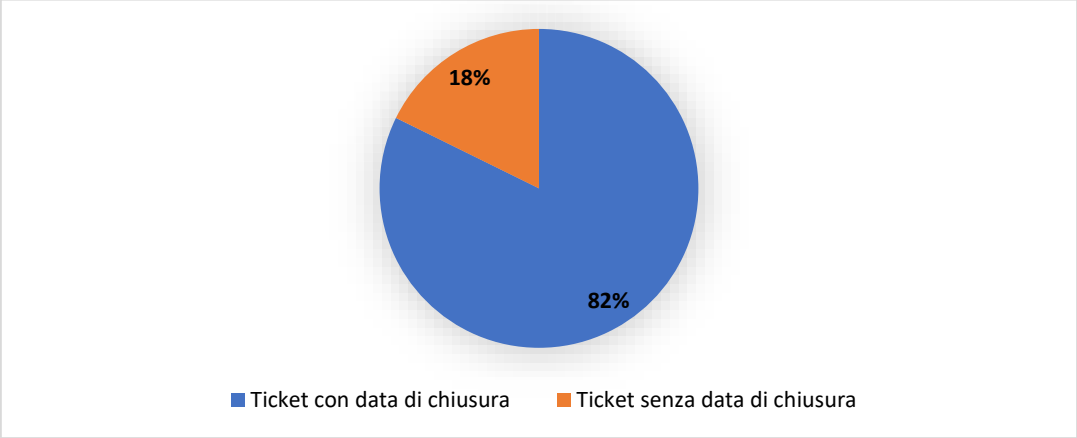

Figura 1: Percentuale dei ticket senza data di chiusura e dei ticket con data di chiusura

I ticket analizzati ricoprono il periodo che va da aprile 2008 a febbraio 2020, mentre il numero medio di ticket chiusi al mese, è pari a 3,599. Per il calcolo del valore di Upper Limit e il valore di Lower Limit, si sono utilizzate le seguenti formule:

$$UpperLimit = MEAN + 3 * STDDEV = 17{,}51$$

$$LowerLimit = MEAN - 3 * STDDEV = -10.31$$

Il valore di Lower Limit risultava essere negativo inizialmente, quindi, è stato spostato a 0 in quanto è impossibile avere un numero di ticket chiusi negativo.

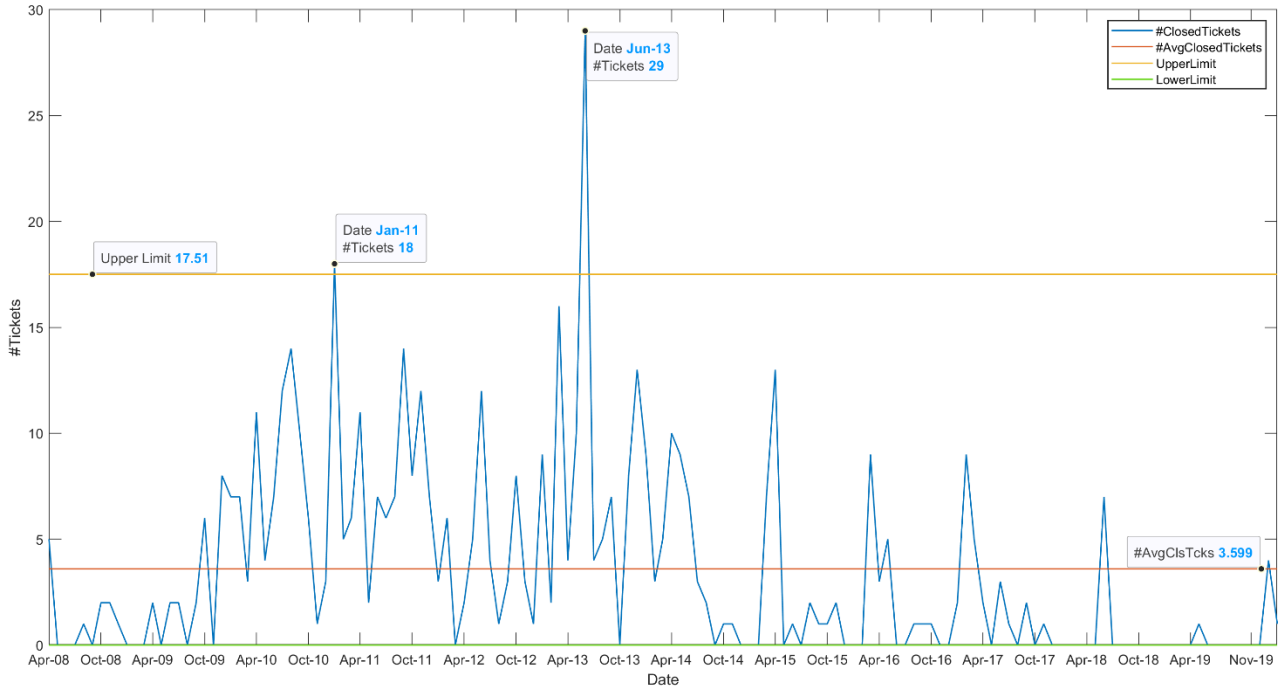

Figura 2: Process Control Chart dei ticket "Bug Fix" del progetto "Mahout"

Il Process Control Chart [Figura 2] nel complesso risulta essere stabile: il numero di ticket chiusi al mese, infatti, non oltrepassa mai i limiti di stabilità (i.e. Upper limit e Lower limit). Vi sono solamente due eccezioni: una nel periodo di gennaio 2011, dove il numero di ticket “Bug Fix” chiusi è 18, e un’altra nel periodo di giugno 2013, dove il numero di ticket “Bug Fix” chiusi è 29 (maggiore di ben 11 unità rispetto all’Upper Limit). tali anomalie, tuttavia, possono essere sfruttate per individuare problemi nel processo software e risolverli, in modo tale da impedire che si possano verificare ulteriori instabilità nel futuro. In particolare, l’analisi delle cause delle anomalie dovrebbe concentrarsi nei periodi antecedenti al verificarsi degli eventi di instabilità (e.g. [Gen-13 : Jun-13]).

Conclusioni

Attraverso l’utilizzo del Process Control Chart, sono state individuate delle anomalie nel processo di sviluppo del progetto “Mahout”: tali eventi anomali sono una fonte utilissima di informazioni, perché permettono di individuare dei sottoperiodi in cui ricercare dei problemi nel processo software. Applicando, inoltre, metodi di statistica inferenziale, si potrebbe prevedere la stabilità del processo in futuro ed eventualmente, eseguire azioni correttive.

Link

- GitHub: https://github.com/francesco1997/isw2-project-deliverable1
- SonarCloud: https://sonarcloud.io/dashboard?id=francesco1997_isw2-project-deliverable1

Deliverable 2

Introduzione

Il machine learning è un utile strumento che può migliorare il processo di sviluppo software. Grazie a tale strumento si può prevedere quali classi possono essere difettose: in tal modo, si possono ottimizzare le risorse (tempo, soldi…) impiegate nelle attività di testing e si può prevenire la formazione di difetti, ponendo più attenzione su quelle classi che sono più a rischio di essere difettose. I classificatori utilizzati come strumenti di predizione devono essere molto accurati affinché possano dare un contributo valido nel processo di sviluppo software. Tecniche di sampling e feature selection possono essere di aiuto a tale scopo: manipolano il contenuto del dataset di training del classificatore, cioè, aggiungono dati in base a quelli presenti oppure eliminano caratteristiche dei dati che non forniscono informazioni

utili a ciò che si vuole predire. Le tecniche suddette potrebbero migliorare l'accuratezza dei classificatori utilizzati: l'obiettivo di questa Deliverable è analizzare empiricamente l'effetto di tali tecniche sull'accuratezza di modelli predittivi di localizzazione di bug nel codice di grandi applicazioni open-source.

Due sono i progetti oggetto dello studio, appartenenti ad Apache Software Foundation:

- Bookkeeper
- OpenJPA

Le tecniche di feature selection analizzate sono:

- No selection
- Best First

Le tecniche di sampling analizzate sono:

- No sampling
- Oversampling
- Undersampling
- SMOTE

I classificatori considerati nello studio sono i seguenti:

- Random Forest
- Naive Bayes
- Ibk

Per valutare l'accuratezza dei classificatori si è utilizzata la tecnica Walk Forward.

Progettazione

Si è sviluppato un programma Java che permette di eseguire la valutazione dell'accuratezza dei classificatori. Il programma è costituito da due componenti:

- Generazione del dataset: raccolta ed elaborazione dei dati necessari per il training e il testing dei modelli predittivi.
- Training e test dei classificatori: training dei classificatori, applicando le tecniche di feature selection e sampling, e valutazione della loro accuratezza.

Il programma è stato progettato in modo tale da poter analizzare molteplici progetti: la loro configurazione avviene in un apposito file (config.json).

Generazione del dataset

Il programma determina per ogni release del progetto le classi di quella release, delle misurazioni associate a quelle classi e la loro difettosità.

La generazione del dataset è suddivisa in due fasi: nella prima vengono raccolti i dati necessari, mentre nella seconda vengono elaborati. Le fonti utilizzate sono due: Jira, per l'individuazione dello storico dei difetti, GitHub, per l'individuazione delle classi del progetto*. Entrambe le fonti forniscono le versioni del progetto, tuttavia, queste sono contrastanti: si è deciso, infatti, di considerare solo le versioni presenti in entrambe le sorgenti. Tale scelta permette di avere dei dati più precisi e accurati, sebbene possano essere di minore quantità. I risultati ottenuti, quindi, saranno più attendibili.*

Inizialmente Il programma determina, per ciascuna release, la difettosità delle classi e, a tal scopo, attinge ai ticket di tipo bug fix di Jira. Questi ticket forniscono teoricamente le versioni che sono state affette da un bug (Affected Versions - AV), inoltre, permettono di individuare le classi coinvolte da questi, essendo quindi difettose. Ciascun commit, relativo alla risoluzione di un ticket, viene contrassegnato nel suo commento dal codice identificativo dello stesso. Le classi difettose sono quelle classi coinvolte in commit associati ad un ticket di tipo bug fix.

Spesso accade che i ticket non abbiano le informazioni relative alle versioni coinvolte (AV), per questo motivo, tali dati devono essere ricavati e utilizzando la tecnica "Proportion". L'intuizione, che sta alla base di questa tecnica, è che vi sia una relazione di proporzionalità nella distanza che intercorre fra la Opening Version/Injected version e quella fra Fixed Version/Injected Version. Tale proporzionalità dovrebbe essere simile tra tutti i ticket di un medesimo progetto, quindi si calcola il valore di proporzionalità dai ticket completi di tutte le informazioni. In questo modo si possono determinare le informazioni mancanti nei restanti ticket.

Il calcolo del valore proporzionale prevede la presenza di tre valori associati al ticket: l'Opening Version (OV), versione in cui è stato rilevato il bug; la Fixed Version (FV), versione in cui è stato risolto il bug; la Injected Version (IV), versione in cui si è formato il bug. L'opening Version si individua tramite un'operazione di interpolazione tra la data di apertura del ticket, elemento sempre presente, e le date di rilascio delle versioni di progetto. L'Opening Version è la versione che ha come data di rilascio quella subito successiva alla data di apertura del ticket. La Fixed Version si individua similmente al caso precedente, l'unica differenza è che non si utilizza la data contenuta nel ticket perché presente un altro valore più accurato. Tale valore corrisponde alla data del commit che risolve il problema riportato nel ticket. Spesso vi possono essere associati più commit ad un solo ticket, per questo motivo si considera la data del commit più recente per indicare la chiusura del ticket. Per quanto concerne l'Injected Version, i ticket non possiedono direttamente questa informazione, hanno, tuttavia, una lista di tutte le versioni che sono state affette dal bug. In questo studio è stata considerata l'Injected version come minima tra quelle affette. Il valore proporzionale P è calcolato utilizzando i ticket che fornivano i valori di Injected Version tramite la seguente formula:

$$P = \frac{FV - IV}{FV - OP}$$

Per i ticket che non forniscono l'Injected Version, si predice tale valore utilizzando la media dei valori di P dell'1% dei ticket precedenti (Moving Window), potendo eseguire così una stima più accurata. La formula utilizzata è la seguente:

$$Predicted\ IV = FV - (FV - OV) * P$$

L'insieme delle versioni affette da un bug sono quelle che intercorrono dalla Injected Version alla Fixed Version esclusa: $[IV, FV)$.

Dopo aver analizzato la difettosità delle classi, il programma esegue delle misurazioni su di esse. ***Sono state selezionate dieci metriche:***

- Size: numero di linee di codice (LOC).
- LOC touched: numero di righe di codice aggiunte o eliminate.
- NR: numero di revisioni in cui è stata modificata la classe.
- NAuth: numero di autori che hanno modificato la classe.
- LOC_added: numero di linee di codice aggiunte alla classe.
- MAX_LOC_added: numero massimo di linee di codice aggiunte alla classe tra tutte le revisioni.
- AVG_LOC_added: numero medio di linee di codice aggiunte alla classe per revisione.
- Churn: differenza tra le linee di codice aggiunte alla classe e quelle eliminate.
- MAX_Churn: Valore massimo della differenza tra le linee di codice aggiunte alla classe e quelle eliminate tra tutte le revisioni.
- AVG_Churn: Valore medio della differenza tra le linee di codice aggiunte alla classe e quelle eliminate per revisione.

Non sono state considerate tutte le versioni del progetto per ridurre l'effetto Snoring, ossia la presenza di classi difettose non ancora individuate. Tali classi fornirebbero informazioni errate (classi difettose che non sono considerate tali) rendendo i risultati dello studio meno accurati e affidabili. Considerando solo la prima metà delle versioni, la probabilità di avere delle classi "dormienti" scende al di sotto del 10%. *Per questo studio è stata considerata la prima metà delle versioni e il programma offre comunque la possibilità di configurare la percentuale di versioni da analizzare.*

La misurazione delle classi avviene utilizzando il codice sorgente presente su GitHub. Per ciascuna release si analizzano le revisioni associate, utilizzando una scansione lineare. Il programma analizza i dati necessari per ciascuna revisione ed aggiorna i valori associati alle misurazioni di ciascuna classe; questo è reso possibile perché le misurazioni

possono essere eseguite in modo incrementale (Size esclusa). Per quanto concerne la misurazione della dimensione delle classi, si è deciso di considerare nel conteggio pure il numero di linee di commento, poiché rendono più semplice la comprensione delle funzionalità offerte e della loro implementazione. *Il programma tiene pure conto di operazioni di spostamento, ridenominazione e copia delle classi in una release, eseguendo così delle misurazioni più precise.*

Per ciascun progetto analizzato, i dati ricavati vengono salvati in un apposito file csv.

Training e test dei classificatori

Per lo sviluppo di questa componente del programma sono state utilizzate le API Java offerte dal Software WEKA. Il programma riceve come input uno o molteplici file arff che contengono per ciascuna classe le seguenti informazioni:

1. Release della classe
2. Una o molteplici misurazioni associate alla classe
3. Difettosità o meno della classe (Yes/No)

Il programma offre la possibilità di convertire i file csv generati dal precedente componente in file arff.

Il programma esegue training e test dei classificatori citati nell'introduzione utilizzando tutte le possibili combinazioni di tecniche di feature selection e sampling. Dai risultati dei test, il programma analizza l'accuratezza del classificatore utilizzando le seguenti metriche:

- Recall
- Precision
- AUC
- Kappa

Per ciascun progetto, il programma restituisce un file csv contenente i dati di ciascun test:

- Nome del progetto
- Numero di release considerate nel training
- Percentuale di dati utilizzati nel training
- Percentuale di classi difettose nel dataset di training
- Percentuale di classi difettose nel dataset di testing
- Classificatore utilizzato
- Tecnica di sampling utilizzata
- Tecnica di feature selection utilizzata
- Numero di veri positivi (True Positive - TP)
- Numero di falsi positivi (False Positive - FP)
- Numero di veri negativi (True Negative - TN)
- Numero di falsi negativi (False Negative - FN)
- Tasso di veri positivi (True Positive Rate - TPR)
- Tasso di falsi positivi (False Positive Rate - FPR)
- Tasso di veri negativi (True Negative Rate - TNR)
- Tasso di falsi negativi (False Negative Rate - FNR)
- Metriche di accuratezza

Risultati

L'analisi dei risultati si concentra nel valutare inizialmente l'efficacia delle tecniche di sampling, indipendentemente dalle tecniche di feature selection, e viceversa. Lo studio viene successivamente approfondito analizzando l'efficacia delle possibili combinazioni delle suddette tecniche.

Bookkeeper

Sampling

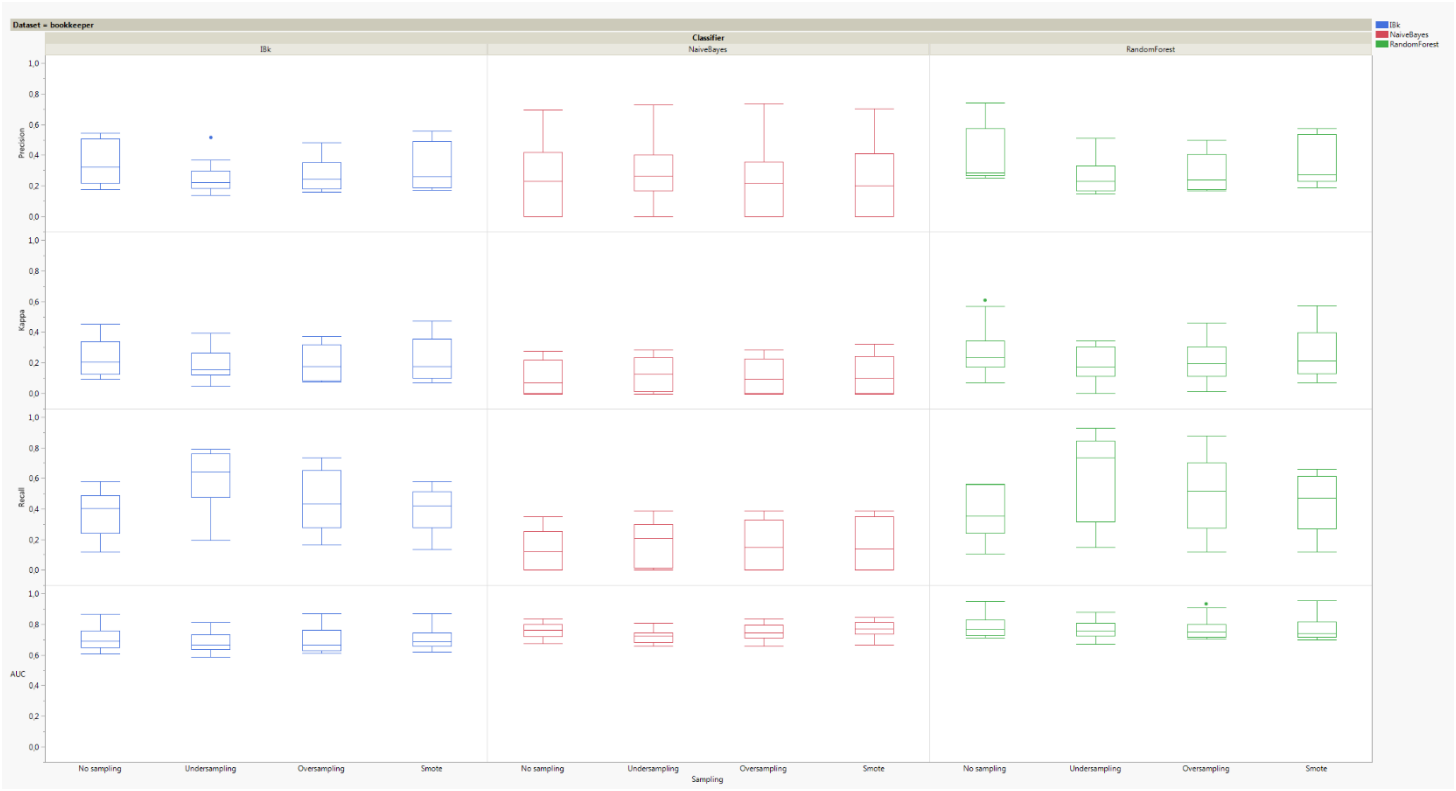

Figura 3: Accuratezza dei classificatori al variare delle tecniche di sampling per il dataset di Bookkeeper

Per quanto riguarda il classificatore Ibk, l'assenza di una tecnica di sampling ne migliora la Precision: il valore mediano (o mediana) è superiore rispetto alle altre mediane, così come il massimo. Anche l'utilizzo della tecnica SMOTE restituisce un risultato simile, mentre la tecnica Undersampling peggiora la Precision del classificatore. Non utilizzando alcuna tecnica di sampling oppure la tecnica SMOTE, si aumenta anche il valore di Kappa e pure il valore di AUC. Il valore di Recall ha un comportamento opposto rispetto alle altre metriche di accuratezza: le tecniche di Undersampling e Oversampling ne aumentano il valore, mentre le altre lo diminuiscono. A tal proposito, la tecnica di Undersampling, sebbene aumenti il valore di Recall, ne diminuisce quello di Precision: crea, quindi, un classificatore che tende facilmente a predire un caso come positivo, anche qualora non lo fosse (elevato numero di falsi positivi). Il motivo di tale comportamento è dovuto probabilmente al fatto che questa tecnica elimina delle informazioni dal dataset, informazioni che diminuiscono la capacità del classificatore nel distinguere correttamente casi positivi da quelli negativi. In base ai dati ottenuti, la tecnica SMOTE restituisce dei buoni risultati; d'altronde, tali risultati possono essere raggiunti senza l'utilizzo di alcuna tecnica di sampling, che comporterebbe costi computazionali aggiuntivi. Viene, in tal modo, massimizzato il risultato, minimizzando lo sforzo.

Per quanto riguarda il classificatore Naive Bayes, la sua Precision migliora nettamente utilizzando la tecnica di Undersampling; il primo quartile del boxplot associato ha, infatti, un valore superiore alla mediana degli altri, mentre le altre tre tecniche determinano dei risultati abbastanza simili tra loro. È difficile, invece, valutare l'efficacia delle tecniche di sampling utilizzando la metrica Kappa: la tecnica SMOTE ha un valore massimo di Kappa maggiore di tutti gli altri, mentre la tecnica di Undersampling ha un valore minimo e una mediana leggermente maggiori degli altri. La tecnica di Undersampling primeggia pure per quanto riguarda la metrica Recall, che è seguita da quella SMOTE. Si presenta una situazione diversa per la metrica AUC: la tecnica di Undersampling registra dei valori che sono evidentemente peggiori rispetto agli altri ottenuti con le altre tecniche e, in questo caso, la tecnica SMOTE riporta i risultati migliori. Alla luce dei risultati ottenuti, non vi è un'unica tecnica in grado di migliorare completamente l'accuratezza del classificatore Naive Bayes, bensì ve ne sono due, ognuna con dei pregi e dei difetti. La prima è la tecnica Undersampling, che migliora la Precision e la Recall del suddetto classificatore peggiorandone, tuttavia, l'AUC; il classificatore, quindi, ha più difficoltà nel distinguere i veri positivi dai veri negativi. La seconda tecnica è SMOTE, che riesce a migliorare l'AUC e la Kappa

penalizzano la Precision e la Recall. Il classificatore è in grado, pertanto, di individuare i veri positivi in modo più efficacie rispetto ad un banale classificatore, tuttavia, tende a stimare più falsi negativi e falsi positivi.

Per quanto riguarda il classificatore Random Forest, si nota che l'utilizzo di una tecnica di sampling ne peggiora la Precision. Tra le tecniche utilizzate, quella SMOTE restituisce il risultato migliore e si presenta una situazione analoga anche per la metrica Kappa. L'utilizzo di tecniche di sampling determina pure un aumento della Recall, ma in modo del tutto differente dai precedenti casi: Undersampling, infatti, restituisce i risultati migliori seguita da Oversampling e SMOTE. La ragione per cui si verifica tutto ciò è presumibilmente l'eliminazione di molti dati di classi non difettose dal dataset di training [Figura 4]. Tale eliminazione, infatti, diminuisce la conoscenza che il classificatore ha riguardo alle classi non difettose, aumentando il peso dei dati delle classi difettose nelle decisioni. Il classificatore tenderà a stimare una classe come difettosa piuttosto che non difettosa; tutto ciò determina un aumento del numero di veri positivi e falsi positivi e una diminuzione di veri negativi e falsi negativi [Figura 5], portando ad un aumento della Recall, diminuendo tuttavia la Precision. Le tecniche di sampling tendono a diminuire pure il valore della metrica AUC, e l'utilizzo di una tecnica di sampling non è idonea per il classificatore Random Forest, in quanto peggiora quasi tutte le metriche di accuratezza.

Figura 5: Percentuale media di classi difettose e non nel dataset di training di Bookkeeper

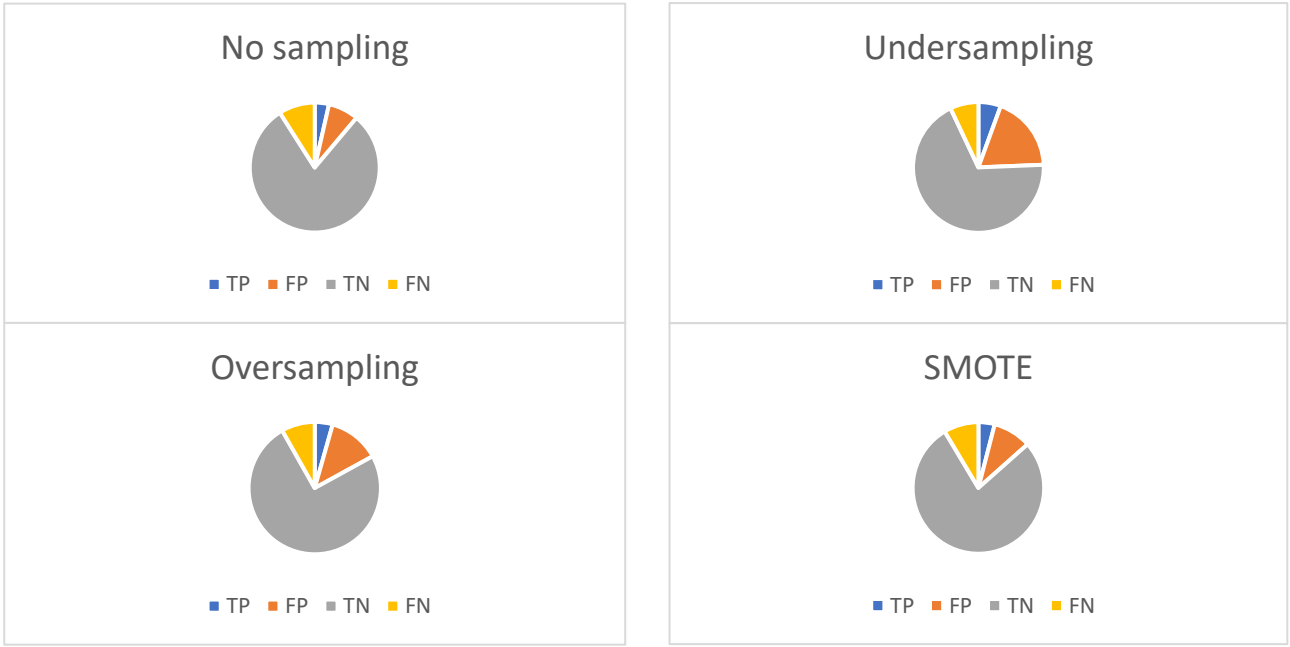

Figura 4: Numero di veri positivi, falsi positivi, veri negativi e falsi negativi medi

In base a quanto evidenziato dai risultati delle analisi, le tecniche di sampling non determinano dei miglioramenti dell'accuratezza nelle previsioni. La tecnica SMOTE, tuttavia, permette di ottenere dei risultati simili nel caso in cui non si utilizzasse alcuna tecnica mentre, in altri casi, ottiene dei risultati migliori.

Feature selection

Figura 6: Accuratezza dei classificatori al variare delle tecniche di feature selection per il dataset di Bookkeeper

Per quanto riguarda il classificatore Ibk, la tecnica Best First determina un miglioramento netto della Precision. Un leggero miglioramento si ha pure nella metrica Kappa: Best First innalza sia il valore del massimo che del minimo sebbene abbassi il valore della mediana. L'utilizzo della tecnica di feature selection migliora, inoltre, la metrica Recall: ottiene un risultato meno variabile rispetto al caso in cui non si utilizzi Best First, e si ha pure un innalzamento del valore minimo. I valori di AUC non variano eccessivamente se si utilizza o meno una tecnica di feature selection. Alla luce di quanto detto, la tecnica Best First determina dei leggeri miglioramenti dell'accuratezza di Ibk.

Riguardo al classificatore Naive Bayes, si evince dai risultati che l'utilizzo della tecnica di feature selection migliora in modo evidente soltanto la metrica Precision. Nelle altre metriche si hanno delle piccole variazioni, alcune delle quali peggiorano il risultato: nella metrica Kappa si ha una diminuzione della mediana e un aumento della variabilità, sebbene aumenti leggermente il massimo; nella metrica Recall e AUC si ha un abbassamento della mediana e del minimo. L'utilizzo della tecnica di feature selection, per il classificatore Naive Bayes, non migliora le metriche di accuratezza ma, in alcuni casi, le peggiora.

Per quanto concerne il classificatore Random Forest, la metrica Precision non varia eccessivamente se si utilizza o meno una tecnica di feature selection: Best First diminuisce leggermente il massimo e la mediana, ma innalza il valore del minimo. Situazione differente si presenta per la metrica Kappa: l'utilizzo di una tecnica di feature selection ne migliora sensibilmente i valori e, in egual modo, avviene nella metrica Recall. La tecnica Best First, tuttavia, peggiora leggermente i valori della metrica AUC. In base ai risultati ottenuti, la tecnica di feature selection migliora effettivamente l'accuratezza del classificatore, aumentando i valori di quasi tutte le metriche, in particolar modo la metrica Kappa, sebbene diminuisca leggermente la metrica AUC.

Per la maggior parte dei casi si è evidenziato che la tecnica Best First, indipendentemente dalla tecnica di sampling, migliora l'accuratezza del classificatore nel predire la difettosità delle classi. L'unica eccezione è data dal classificatore Naive Bayes.

Sampling e Feature Selection

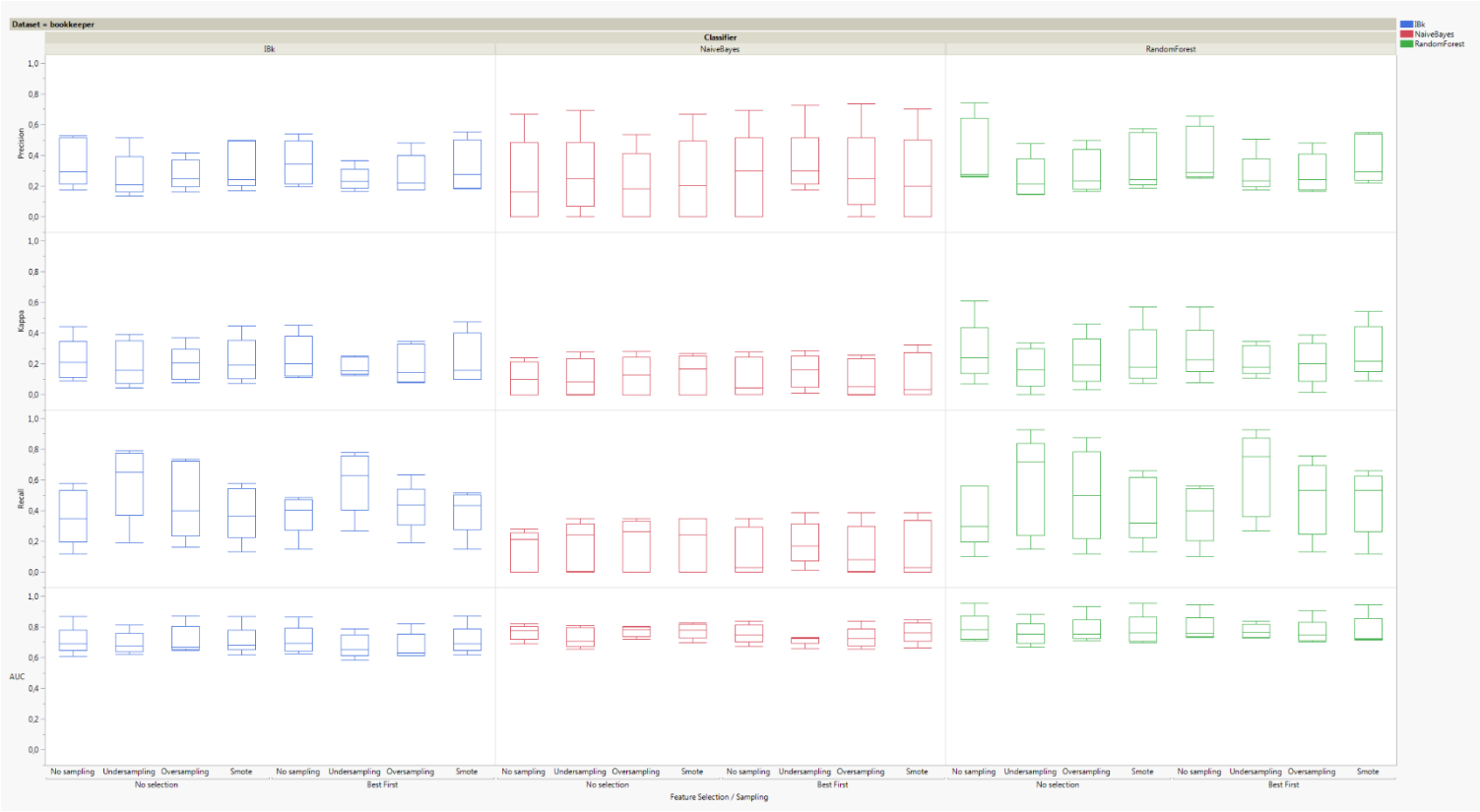

Figure 7: Accuratezza dei classificatori al variare delle tecniche di feature selection e sampling per il dataset di Bookkeeper

Considerando il classificatore Ibk, il solo utilizzo della tecnica Best First, migliora notevolmente la Precision: ne aumenta il valore del minimo, del massimo e della mediana. La medesima situazione si presenta anche per quanto riguarda le metriche Kappa e AUC. Bisogna, invece, combinare la tecnica di Undersampling a quella Best First per massimizzare i risultati della metrica Recall. I risultati ottenuti sono in linea con quelli precedenti, confermando l'utilizzo della sola tecnica Best First per migliorare notevolmente l'accuratezza nelle previsioni del classificatore. Tale tecnica, inoltre, diminuisce il numero di dati dei campioni del dataset e il costo computazionale necessario al training.

L'analisi dell'applicazione delle tecniche di feature selection e sampling al classificatore Naive Bayes è abbastanza complessa. Da uno studio preliminare sembra che l'utilizzo combinato delle tecniche Best First e Undersampling determini il risultato migliore: queste aumentano visibilmente i valori delle metriche Precision, Kappa e Recall. D'altronde, tali tecniche peggiorano notevolmente la metrica AUC, diminuendo la capacità del classificatore nel distinguere i veri positivi dai veri negativi. Un'altra possibile alternativa sarebbe utilizzare solamente la tecnica SMOTE: questa massimizza il valore della metrica AUC a scapito delle altre (e.g. Precision) poiché ne diminuisce i valori di minimi. I risultati rispettano quanto ottenuto dalle analisi precedenti, evidenziando il fatto che non vi è una tecnica di sampling o feature selection che migliora l'accuratezza del classificatore indipendentemente dall'utilizzo o meno di altre tecniche.

In merito al classificatore Random Forest, una qualsiasi combinazione di tecniche di feature selection e sampling peggiora i valori delle metriche di accuratezza. L'unica eccezione si ha nella metrica Recall, dove l'utilizzo combinato di Best First e Undersampling migliora notevolmente i valori ottenuti. I risultati di questo studio sono coerenti a quelli dell'analisi dei metodi di sampling ma, si contrappongono a quelli dell'analisi dei metodi di feature selection: questi, infatti, evidenziano Best First come tecnica migliore. Gli effetti delle singole tecniche, quindi, non tendono a sommarsi quando si combinano.

Nel complesso, le tecniche di feature selection e sampling aiutano a migliorare l'accuratezza del classificatore: non vi sono, però, delle tecniche che migliorino allo stesso modo tutti i classificatori.

OpenJPA

Sampling

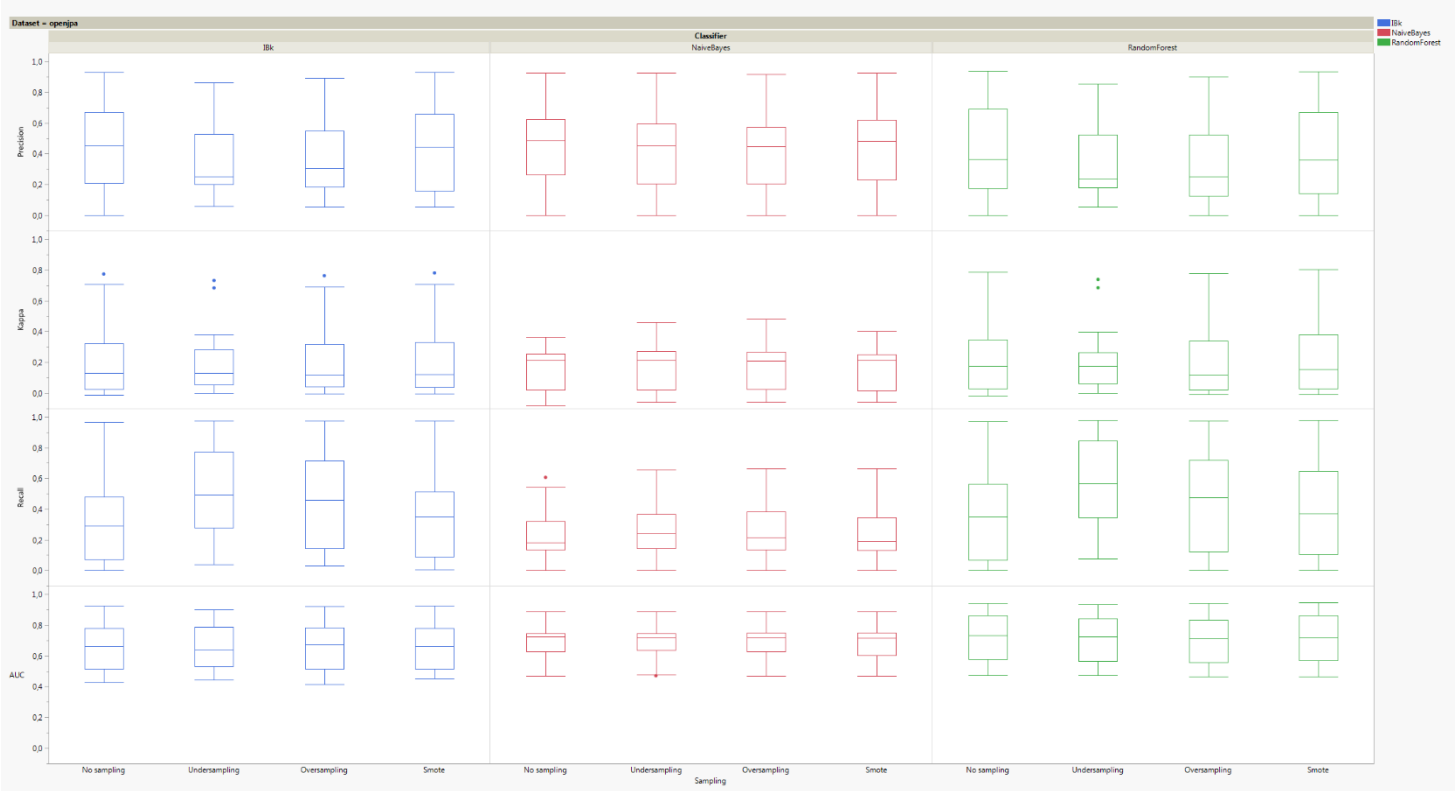

Figure 8: Accuratezza dei classificatori al variare delle tecniche di sampling per il dataset di OpenJPA

Per quanto concerne il classificatore Ibk, la tecnica SMOTE migliora complessivamente le metriche di accuratezza rispetto al caso in cui non ne venga utilizzata nessuna. La tecnica Undersampling registra i valori migliori per la metrica Recall, ma quelli peggiori per la metrica Kappa: costruisce, così, un classificatore che rileva i veri positivi con poca più efficacia rispetto ad uno banale. La tecnica Oversampling migliora le metriche Recall e AUC, mentre peggiora leggermente la Precision. L'utilizzo delle tecniche di sampling, in particolare la tecnica SMOTE, migliora nel complesso l'accuratezza del classificatore Ibk.

Per quanto riguarda il classificatore Naive Bayes, l'utilizzo delle tecniche di sampling peggiora leggermente la Precision del classificatore. Queste tecniche, tuttavia, migliorano nettamente la metrica Kappa, che registra i valori migliori con la tecnica Undersampling e tale situazione si presenta in maniera analoga per la metrica Recall. Le tecniche di sampling, invece, non variano il valore della metrica AUC. In base ai risultati ottenuti, le tecniche di sampling migliorano l'accuratezza del classificatore, in particolare, la tecnica Undersampling determina complessivamente i valori migliori.

In merito al classificatore Random Forest, le tecniche si comportano in maniera differente in base alla metrica di accuratezza. L'utilizzo di tecniche di sampling, in particolare della tecnica Undersampling, peggiora il valore di Precision e di Kappa. Tali tecniche, tuttavia, migliorano nettamente il valore della metrica Recall e quella che ne determina il valore migliore è la tecnica Undersampling. L'utilizzo o meno di tecniche di sampling non variano il valore della metrica AUC. Nel complesso, le tecniche di sampling non migliorano significativamente l'accuratezza del classificatore Random Forest ma, in alcuni casi, la peggiorano.

In base ai risultati ottenuti, l'utilizzo delle tecniche di sampling migliora la capacità predittiva del classificatore, con l'unica eccezione di Random Forest. Tuttavia, l'efficacia delle tecniche è altamente influenzata dal classificatore utilizzato.

Feature Selection

Figura 9: Accuratezza dei classificatori al variare delle tecniche di feature selection per il dataset di OpenJPA

Per quanto riguarda il classificatore Ibk, l'utilizzo di una tecnica di feature selection peggiora visibilmente tutte le metriche di accuratezza; tutte le informazioni fornite dal dataset sono probabilmente fondamentali per prevedere più accuratamente la difettosità delle classi.

Per quanto riguarda il classificatore Naive Bayes, la tecnica Best First peggiora, anche in questo caso, l'accuratezza del classificatore: tutte le metriche, infatti, hanno dei valori tendenzialmente bassi ed altamente variabili. La metrica Kappa, ad esempio, ha un valore massimo visibilmente maggiore rispetto al caso in cui non si utilizzi una tecnica di feature selection, ma ha dei valori minimi che tendono ad essere negativi. Questi rappresentano il fatto che l'accuratezza del classificatore, nell'individuare i veri positivi, è peggiore di quella di un classificatore banale.

Per quanto concerne il classificatore Random Forest, si ha una situazione analoga ai casi precedenti: l'utilizzo della tecnica Best First peggiora tutte le metriche di accuratezza.

Alla luce dei risultati ottenuti, non è consigliabile utilizzare la tecnica di feature selection Best First per questo dataset perché peggiora sensibilmente le metriche di accuratezza del classificatore, indipendentemente da quale esso sia.

Sampling e Feature Selection

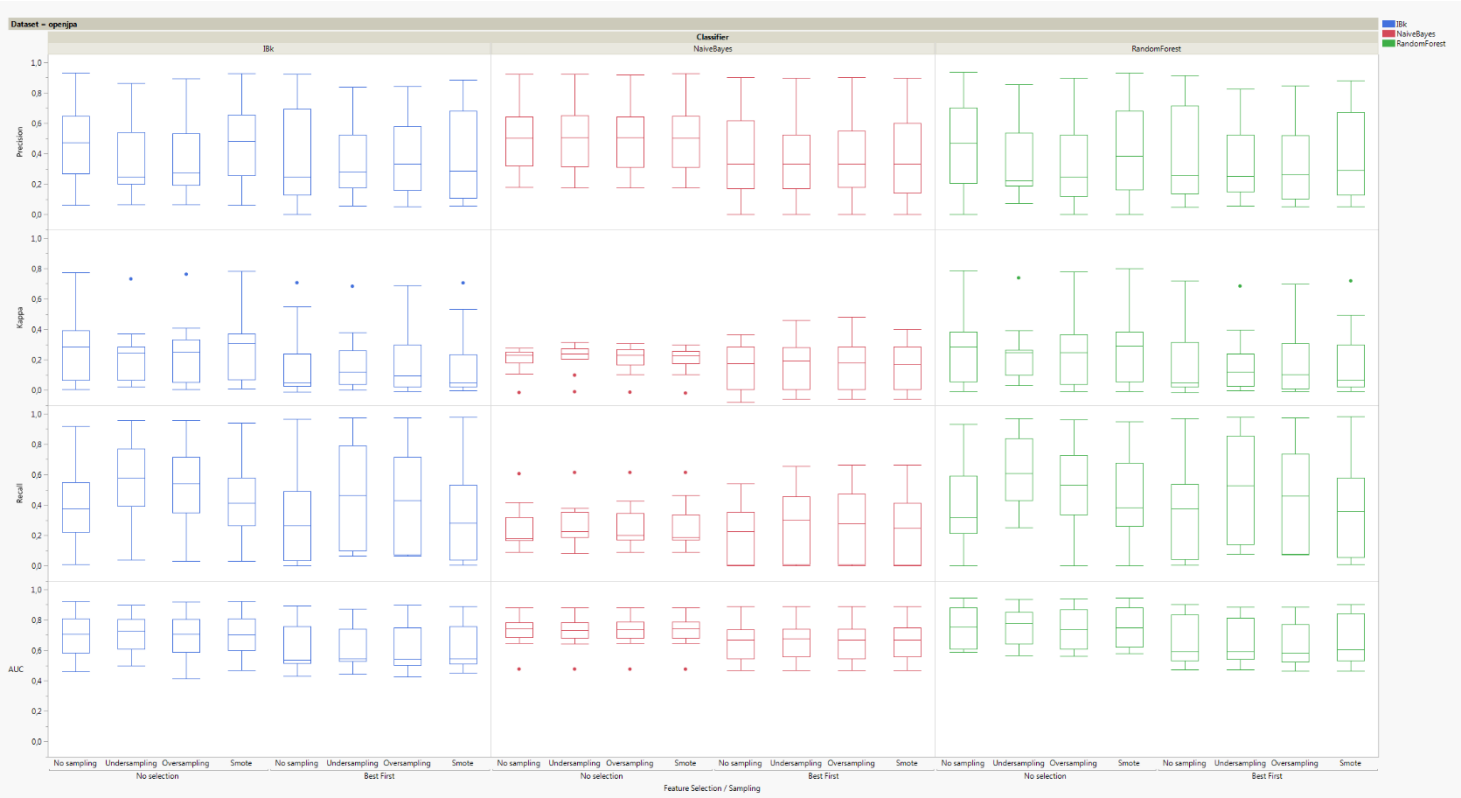

Figure 10: Accuratezza dei classificatori al variare delle tecniche di feature selection e sampling per il dataset di OpenJPA

Per quanto riguarda il classificatore Ibk, l'utilizzo della sola tecnica SMOTE migliora notevolmente l'accuratezza del classificatore: questa tecnica, infatti, accresce i valori di tutte le metriche di accuratezza del classificatore. Anche l'implementazione della sola tecnica Undersampling sembrerebbe una valida alternativa, in quanto aumenta notevolmente i valori di Recall e leggermente i valori di AUC. Tale tecnica, tuttavia, peggiora notevolmente Precision e Kappa, rendendola meno efficacie rispetto a SMOTE. I risultati ottenuti sono concordi a quelli precedenti: gli effetti delle tecniche di feature selection e sampling, quindi, tendono a sommarsi quando vengono combinate.

Per quanto concerne il classificatore Naive Bayes, l'utilizzo della sola tecnica Undersampling migliora notevolmente i valori delle metriche di accuratezza, in particolare Kappa. L'aggiunta della tecnica di feature selection Best First aumenta notevolmente i valori dei massimi di tutte le metriche, ma ne diminuisce allo stesso modo i valori dei minimi, rendendo i risultati altamente variabili ed instabili. Questo studio è concorde a quanto analizzato precedentemente.

Per quanto riguarda il classificatore Random Forest, l'utilizzo di tecniche di sampling e feature selection, combinate tra loro o non, variano di poco le metriche di accuratezza del classificatore e, in alcuni casi, le peggiorano. L'utilizzo della sola tecnica Undersampling migliora nettamente i valori di Recall, ma peggiora sensibilmente quelli di Kappa. Anche in questo caso i risultati ottenuti concordano con quanto elaborato precedentemente.

Le tecniche di feature selection peggiorano complessivamente le metriche di accuratezza di tutti i classificatori, mentre quelle di sampling aiutano a migliorarne la capacità predittiva, ad eccezione di Random Forest. Non vi è una tecnica di sampling comune che massimizza l'accuratezza di tutti i classificatori, bensì dipende dalla tipologia di classificatore.

Conclusioni

In base ai risultati delle analisi, la tecnica SMOTE utilizzata senza alcuna tecnica di feature selection, migliora l'accuratezza predittiva del classificatore Naive Bayes in Bookkeeper e del classificatore Ibk in OpenJPA. Anche la combinazione delle tecniche di Undersampling e Best First migliora l'accuratezza del

classificatore Naive Bayes in Bookkeeper. La tecnica di Oversampling, invece, migliora il classificatore Ibk in OpenJPA. Tutte le tecniche di sampling, in particolare Undersampling, hanno registrato degli ottimi risultati nel classificatore Naive Bayes: queste, infatti, migliorano generalmente i risultati delle misurazioni di accuratezza di questo. Le tecniche di feature selection, invece, non hanno migliorato l'accuratezza di nessun classificatore in nessun dataset, con l'unica eccezione di Ibk in Bookkeper: tutte le informazioni delle classi, probabilmente, sono utili per la predizione della presenza di difetti. Nessuna tecnica di sampling e feature selection migliora l'accuratezza del classificatore Random Forest, sia in Bookkeeper che in OpenJPA, ma ne peggiora solamente la capacità predittiva.

Link

- GitHub: https://github.com/francesco1997/isw2-project-deliverable2
- SonarCloud: https://sonarcloud.io/dashboard?id=francesco1997_isw2-project-deliverable2