Vol.2020-SE-206 No.3
2020/11/27

ランダムフォレストと相関ルール分析の組み合わせによるソフトウェアバグ予測の試行

瀬戸 俊輝[1]　門田 暁人[1]

概要：従来よりソフトウェアの品質確保とテスト効率化のためにバグを含むモジュールを予測（判別）する手法が盛んに研究されている．本研究では現在有力とされているランダムフォレストによる判別モデルと相関ルール分析を組み合わせた手法による判別精度向上を目的とする．相関ルール分析を用いることで解釈が容易になること，またバグを含むの条件とその信頼度を確認できることから適切に用いることで判別精度の向上が期待できる．本手法ではプロジェクトの過去のデータに対して相関ルール分析を適用することでルールを抽出する．ルールの選択にはリフト値を用い，得られたルールを現在進行中のバージョンに適用することでバグの有無を判別する．いずれのルールにも合致しないモジュールについてはランダムフォレストを用いる．評価実験を行ったところ，ランダムフォレストのみによる判別と比較して，9 件のプロジェクトのうち 6 件の判別精度が改善され，1 件が同等，2 件が悪化する結果となった．

キーワード：ソフトウェア品質，メトリクス，機械学習，ルールベース予測

An Attempt to Discriminate Software Faults by Combining Random Forest and Association Rule Analysis

Abstract: Methods for discriminating modules containing fault are being actively researched to ensure software quality and improve the test efficiency. The purpose of this study is to improve the discrimination accuracy by a method that combines the discrimination model by random forest, which is currently considered to be promising, and the association rule analysis. It can be expected that interpretation will be easier by using association rule analysis. In addition, since the conditions and confidence of fault occurrence can be confirmed by association rules, it is expected that the discrimination accuracy will be improved by using them properly. In this method, the association rule analysis is applied to the past data of the project and the rules are extracted. The lift value is used to select the rules, and faults are discriminated by applying the obtained rules to the version currently in progress. For modules that do not match any rule, we use a random forest to discriminate them. As a result of the evaluation experiment, compared to the random forest only method, the discrimination accuracy of 6 out of 9 projects were improved, 1 was equivalent, and 2 was worse.

Keywords: Software quality, Metrics, Machine learning, Rule-based prediction

1. はじめに

ソフトウェアのテスト及び保守工程においては，限られたリソースで十分な品質を確保することが重要である．その要求を満たすためにはバグを含む可能性の高いモジュールを効率よくテストする必要があり，これを実現するためにモジュールから得られるメトリクスを元にバグが含まれているか否かを予測する手法が多数提案されている．例を挙げると，ロジスティック回帰分析 [1]，線形判別分析，ランダムフォレスト [2] などである．特にランダムフォレストは近年の研究で有力であることが示されており [3]，モデル構築用データに過適合しないことや各説明変数がどの程度結果に影響しているかを算出することができるという特徴を持つ．さらにバグを含む度合い（確率や期待値）を知ることができるという点から，開発現場において柔軟な使い方ができる．

一方で非モデルベースの方法として，相関ルール分析

[1] 岡山大学大学院自然科学研究科

を用いたバグモジュール判別手法が提案されている [5][1]. バグを含む（もしくは含まない）条件をルールとして出力することができるため，解釈が容易であるという特徴を持つ．例えば「(WMC=大) & (TLOC=大) →バグあり」ならば，モジュール（クラス）のメソッド数が大カテゴリかつソースコード行数が大カテゴリならばバグありと判定できる．このようにルールベースの判別手法はバグの混入を判別する指標としてわかりやすい．しかし，ルールベースの分類では，いずれのルールにも合致しないモジュールを判別することはできない．そこで，任意のモジュールを判別可能なランダムフォレストを併用することで相関ルールでは判別することができないモジュールの推定を行う．

従来，ロジスティック回帰分析と相関ルール分析を組み合わせた方法が提案されており，2 つのデータセットを用いた評価が行われている [1]．本稿では，ロジスティック回帰分析の代わりに，より性能の良いランダムフォレストを用い，より多くのデータセットを用いた評価を行う点が異なる．

相関ルールの導出において，一般に，学習データから得られる相関ルールは非常に多く，予測に寄与しないものも含まれる．そのため，ルールの選別を行う必要がある．ルール選別ための指標としては，支持度，信頼度，リフト値の 3 つがよく用いられる．バグモジュール予測においては支持度とリフト値によるルールの選択が行われており [1]，本稿においてもそれに準ずることとする．

以降 2 章ではバグ予測に用いられる手法の説明を行う．3 章ではランダムフォレストと相関ルール分析を組み合わせた手法について説明し，4 章では複数のデータセットを用いて組み合わせ手法の評価を行う．5 章では考察について述べ，6 章で本研究のまとめを記す．

2. バグモジュール予測

本実験におけるバグモジュール予測とは，ソフトウェアを構成する各モジュール（本稿ではソースファイル）に含まれるバグの有無を予測することである．予測モデルにおける説明変数として，各モジュールから得られるメトリクス（ソースコード行数，変更行数，メソッド数，サイクロマティック数など）とを用いる．モデルの学習には，同一のソフトウェアプロダクトの過去のバージョン（旧バージョン）を用いる．つまり，過去の実績データに基づいて予測モデルを構築し，現在開発中のバージョン（新バージョン）のバグを予測することを想定する．バグを含む，もしくは，バグを含む可能性が高いと予測されたモジュールに対しては，より多くのテスト工数の割り当て，そうでないモジュールにはより少ないテスト工数を割り当てることで，テスト工数の削減と多数のバグ発見の両立を目指すことになる．

バグモジュール予測モデルの構築手法はこれまで多数提案されている．例えば線形判別分析，分類木，ニューラルネットワーク，ロジスティック回帰分析などがある．このように多数の手法があるものの，近年の研究ではランダムフォレストによる判別モデルが有力とされている [3].

2.1 ランダムフォレスト

ランダムフォレストは多数の分類木を用いて集団学習を行う手法であり，2001 年に Breiman によって提案された．モデル構築用のデータセットに対して繰り返しランダムサンプリングを行い，得られたサンプル群から多数の分類木を構築する．そして，各分類木の出力の多数決により最終的な判別結果を得る．従来の集団学習と異なり，ランダムフォレストでは変数をランダムサンプリングしたサブセットを用いるので，高次元データ解析に向いている特徴をもつ．その他，分類に用いる変数の重要度が推定可能であることや分類問題における各群の個体数がアンバランスであってもエラーのバランスが保たれるといった長所をもつ [4].

2.2 相関ルール分析

相関ルール分析は Agrawal らによって提案されたデータ分析手法である [6]．データベースから共起関係にある組み合わせを網羅的に抽出することができ，「$A \Rightarrow B$」のような簡潔なルールを網羅的に得ることが可能である．これは A が出現したときに B も現れることを示す．A を前提部（条件部）と呼び，B を結論部（帰結部）と呼ぶ.

一般に，相関ルール分析を行うことで大量のルールが得られるため，次に示すようなルールの重要性を表す指標を用いることで取捨選択を行う．

$$\text{支持度}\ (A,B) = \frac{\text{A と B が同時に出現したケース数}}{\text{全ケース数}}$$

$$\text{信頼度}\ (A,B) = \frac{\text{A の発生時に B が出現したケース数}}{\text{A が発生したケース数}}$$

$$\text{リフト値}\ (A,B) = \frac{\text{信頼度 (A,B)}}{\text{全ケースにおける B の発生割合}}$$

支持度が大きいほど頻繁におこる事象を表す．信頼度とリフト値は A と B の関連の強さを表す.

相関ルール分析でデータを扱う際に前提部として用いるメトリクスは質的変数が想定されている．しかし本研究で対象とするようなソフトウェア工学分野におけるデータセットに関しては数値データを多く含むことから変換を行う必要がある．本実験で用いた Eclipse_jdt データセット中の総行数を意味する TLOC を例にとると，「$TLOC < 83, 83 \leqq TLOC < 213, 213 \leqq TLOC$」のような変換を行う．このケースは各メトリクスを昇順に並べ，サンプル数が均等になるように分割されている．

相関ルール分析の利点としては欠損値を含むデータセットに適用できること及び，他の分析方法と比べて得られる情報が平易で理解しやすいことが挙げられる．

3. 組み合わせ手法

本研究では相関ルール分析とランダムフォレストを組み合わせた手法によりバグモジュールの判別を行う．この手法では，バグの有無を判別したいモジュール群が与えられたときに，相関ルールの条件部に合致するモジュールについては，当該ルールに基づいてバグの有無を判別する．いずれの相関ルールにも合致しないモジュールについては，ランダムフォレストによりバグの有無を判別する．

判別のための相関ルールは，バグの有無と各種メトリクスを含む旧バージョンのデータを使用してルールの抽出を行い，リフト値を用いてルールを選択する．この際に，結論部が「バグあり」となるルールのみを用い，「バグなし」となるルールは採用しない．バグありルールのみであっても，バグの有無の判別に十分であることが文献 [7] における実験で示されているためである．

本組み合わせ手法の有効性は，判別に用いるルールを選定する基準（リフト値の閾値）に依存すると考えられる．また，望ましい閾値は，データセットにより異なる可能性がある．そこで，本研究では，リフト値の閾値を，選定候補となるルール群の上位 5%，10%，20%，40%，60%のように変動させ，実験的に評価を行う．

本組み合わせ手法の利点としては，判別精度の向上が見込めるこてい加えて，相関ルールを用いることによる解釈の容易性がある．バグにつながる条件の参照を可能とすることで，開発現場において受け入れられやすいと期待される．

4. 実験

本実験の目的はバグモジュール判別手法として相関ルール分析とランダムフォレスト法を組み合わせることで判別精度の向上が得られるか否かを評価することである．

4.1 データセット

実験ではオープンソースソフトウェア（OSS）プロジェクトから得られた 9 種類のデータセットを用いた．実験で用いたデータセットの概要を**表 1** に示す．Project Name がプロジェクトの名前を表している．各プロジェクトのうち Release 列の中で数字の小さいものが旧バージョンを表しており，学習データとして用いるデータセットである．Modules 列は各プロジェクトの当該バージョンに含まれているモジュールの総数を示す．Defect content(%) はモジュール総数に対してバグを含んだモジュールの割合を表す．

表 1 実験に用いたデータセットの概要
Table 1 Overview of the dataset used in the experiment

Project Name	Release	Modules	Defect content(%)
Ant	1.5	293	10.9
	1.6	350	26.3
Camel	1.4	856	16.8
	1.6	945	19.9
Eclipse JDT	3.1	3192	16.5
	3.2	3408	10.9
Eclipse PDE	3.1	228	21.5
	3.2	309	25.6
Ivy	1.4	241	6.6
	2.0	352	11.4
Jedit	4.2	367	13.1
	4.3	492	2.2
Log4j	1.1	109	33.9
	1.2	205	92.2
Netbeans	4.0	4660	19.3
	5.0	9332	14.6
Synapse	1.0	157	10.2
	1.1	222	27.0

4.2 メトリクス

本節ではバグの有無を判定するための説明変数となるメトリクスについて説明する．メトリクスにはプロダクトメトリクスとプロセスメトリクスがある．プロダクトメトリクスは開発された成果物から計測されるメトリクスであり，プロセスメトリクスは開発時のプロセス上の作業を定量化したものである．本実験で用いたデータセットは含まれるメトリクスの違いから 2 種類に大別される．Eclipse_pde，Eclipse_jdt，Netbeans のデータ収集や測定方法については文献 [7] で述べられており，**表 2** に示す 15 種類のプロダクトメトリクスと 8 種類のプロセスメトリクスが含まれている．もう一方の Ant，Camel，Ivy，Jedit，Log4j，Synapse は tera-PROMISE repository[8] において過去に公開されていたデータセットであり，**表 3** に示す 20 種類のプロダクトメトリクスが含まれている．モデル作成およびルールの作成に用いる学習データに対して変数選択を行い，互いに著しく相関の高い（相関係数が 0.9 以上）メトリクスは冗長なルールを生成する原因になると考え一方を削除した．

4.3 実験手順

以下に実験の手順を説明する．

(1) 学習データを用いてルールを作成する．このとき量的変数の変換と変数の選択を行う．最低支持度は 0.01，ルールの最大結合数は 3 とする．
(2) バグありルールのリフト値上位 5%，10%，20%，40%，60%を採用する．
(3) テストデータに対してルールを適用し，バグを含む特徴をもつモジュールを抽出．
(4) ルールでの判別ができなかったモジュールをランダムフォレストにより判別．本実験では R のランダムフォレスト専用パッケージである randomForest を使用．

表 2 メトリクス：Eclipse_pde, Eclipse_jdt, Netbeans
Table 2 Metrics : Eclipse_pde, Eclipse_jdt, Netbeans

分類	名称	定義
プロダクトメトリクス	WMC	重み付きメソッド数
	FOUT	ファンアウト総数
	NBD	ネストの最大深さ
	PAR	パラメータ総数
	VG	サイクロマチック複雑度の総数
	NOF	フィールドの総数
	NOM	メソッドの総数
	NSF	制定フィールドの総数
	NSM	静的メソッドの総数
	CBO	メソッドやインスタンス変数を参照・実行している他のクラス数
	NOC	直接のサブクラス数
	RFC	オブジェクトが受け取るメッセージに応答して実行される合計メソッド数
	DIT	継承ツリーの深さ
	LCOM	メソッド凝集性欠如の度合い
	TLOC	ソースコード行数
プロセスメトリクス	ADD	コードの追加行数
	DEL	コードの削除行数
	CHRN	ADD+DEL
	TPC	事前変更の総数
	AGE	モジュールの年数
	BFC	リリース前 3 か月間でバグ修正トランザクションに関与した回数
	PRF	リリース前 3 か月間のバグ数
	REFAC	リリース前 3 か月間でリファクタリングトランザクションに関与した回数

表 3 メトリクス：Ant, Camel, Ivy, Jedit, Log4j, Synapse
Table 3 Metrics : Ant, Camel, Ivy, Jedit, Log4j, Synapse

分類	名称	定義
プロダクトメトリクス	WMC	重み付きメソッド数
	DIT	継承ツリーの深さ
	NOC	直接のサブクラス数
	CBO	メソッドやインスタンス変数を参照・実行している他のクラス数
	RFC	オブジェクトが受け取るメッセージに応答して実行される合計メソッド数
	LCOM	メソッド凝集性欠如の度合い
	LCOM3	メソッド凝集性欠如の度合い
	NPM	パブリックメソッドの総数
	DAM	データアクセスメトリクス
	MOA	集約の計測
	MFA	機能体抽象化の計測
	CAM	クラスのメソッド間凝集性
	IC	継承結合度
	CBM	メソッド間結合度
	AMC	平均メソッド複雑度
	Ca	パッケージ内のクラスに依存するパッケージ外のクラス数
	Ce	パッケージ外のクラスに依存するパッケージ内のクラス数
	MaxCC	サイクロマティック複雑度の最大値
	AvgCC	サイクロマティック複雑度の平均値
	TLOC	ソースコード行数

引数はデフォルトの値を使用した．
(5) ランダムフォレストのみによる手法と F1 値を比較．

4.4 評価手法

バグモジュール判別の評価指標として適合率 (precision)，再現率（recall），F1 値を用いる．これらの指標は混同行列と呼ばれる分類の出力結果をまとめた**表 4** から算出される．

適合率（precision）：バグありと判別した件数の内，実際にバグを含むモジュールであった件数を示す．

$$precision = \frac{TP}{TP + FP} \tag{1}$$

再現率（recall）：実際にバグを含むモジュール総数の内，正しくバグありと予測した件数を示す．

$$recall = \frac{TP}{TP + FN} \tag{2}$$

F1 値：適合率と再現率の調和平均を示す．判別モデルを評価するためには適合率と再現率のバランスが重要である．F1 値は適合率と再現率が同じように高い場合に良い評価を算出する．値域は [0,1] となっており，1 に近づくほど精度が高い．

$$F_1 = \frac{2 \times recall \times precision}{recall + precision} \tag{3}$$

5. 結果と考察

5.1 データセット毎の予測精度

実験結果を**図 1** から**図 9** に示す．9 件のデータセットに対して組み合わせ手法を適用した結果，Ant，Eclipse_pde，Ivy，Log4j，Netbeans，Synapse の 6 件で F1 値が向上した．一方で Camel は同等，Eclipse_jdt，Jedit では悪化する傾向がみられた．

5.2 考察

この節では結果に対する考察を行う．まず F1 値が改善したデータセットについて考えるにあたり，Ant の詳細結果を**図 11** に示す．precision に着目した場合，ランダムフォレストのみの手法が優れている．組み合わせ手法に関してはリフト値の閾値を大きくするほど低下しており，平均で 18.3%の悪化がみられた．一方で recall については閾値を大きくするほど改善されることがわかり，平均で 33.0%の改善がみられた．この傾向は F1 値が改善された他のデータセットでも同様であることが確認できている．

表 4 混同行列
Table 4 Confusion matrix

	予測：バグあり	予測：バグなし
実際：バグあり	TP（真陽性）	FN（偽陰性）
実際：バグなし	FP（偽陽性）	TN（真陰性）

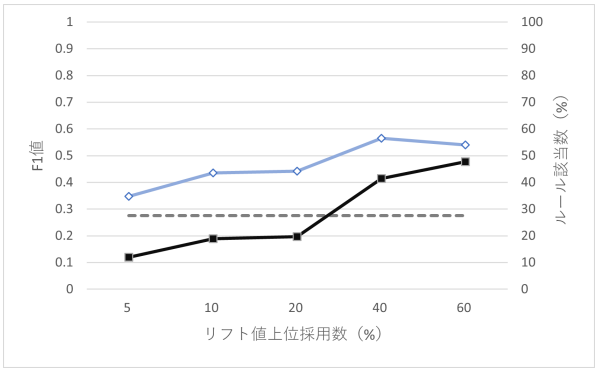

図 1 Ant の判別結果
Fig. 1 Discrimination result of Ant

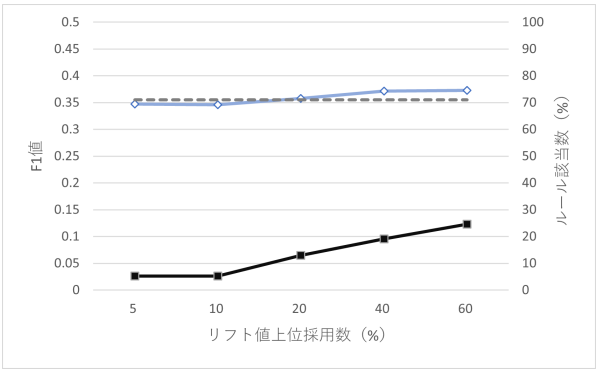

図 2 Camel の判別結果
Fig. 2 Discrimination result of Camel

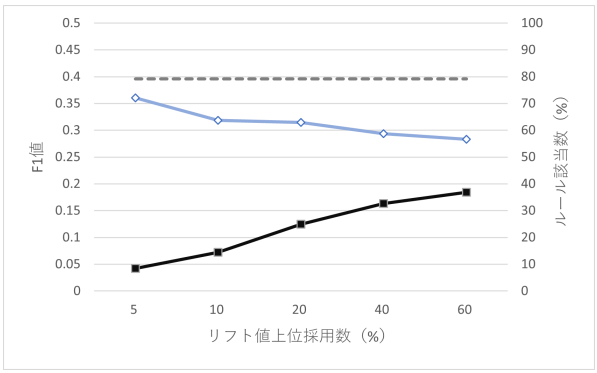

図 3 Eclipse_jdt の判別結果
Fig. 3 Discrimination result of Eclipse_jdt

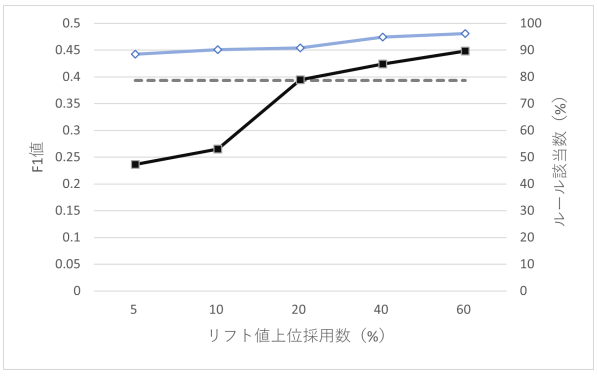

図 4 Eclipse_pde の判別結果
Fig. 4 Discrimination result of Eclipse_pde

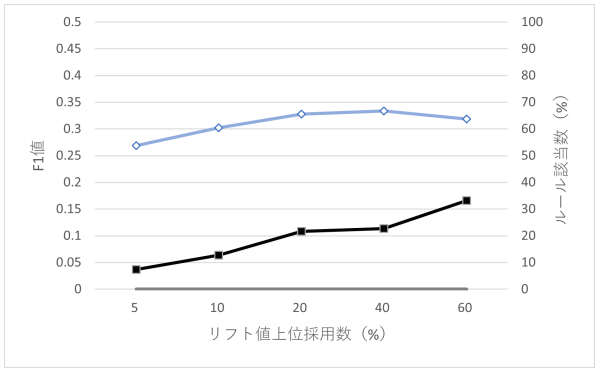

図 5 Ivy の判別結果
Fig. 5 Discrimination result of Ivy

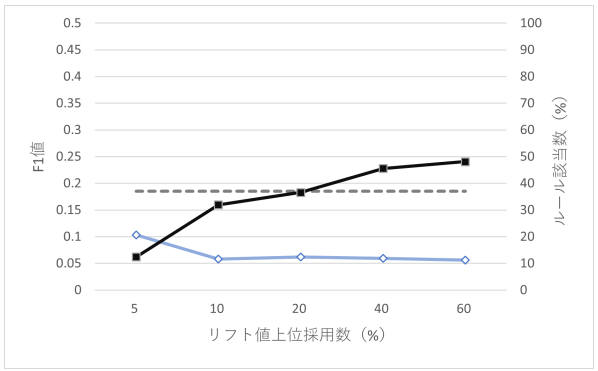

図 6 jedit の判別結果
Fig. 6 Discrimination result of jedit

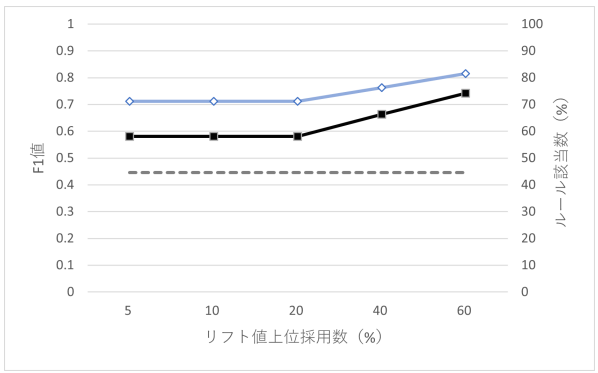

図 7 Log4j の判別結果
Fig. 7 Discrimination result of Log4j

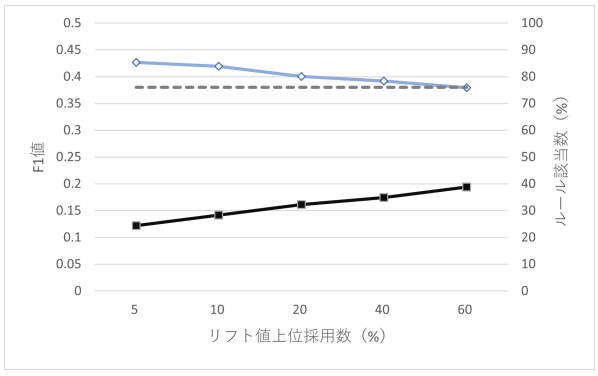

図 8 Netbeans の判別結果
Fig. 8 Discrimination result of Netbeans

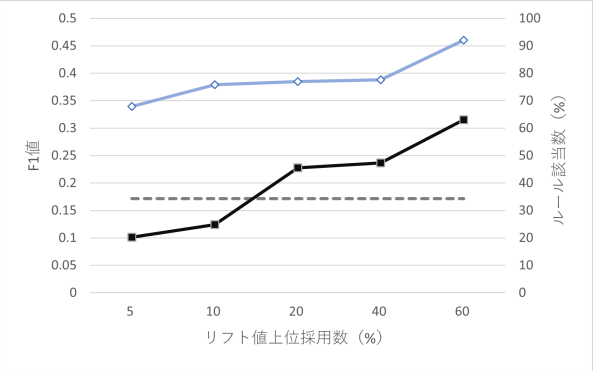

図 9 Synapse の判別結果
Fig. 9 Discrimination result of Synapse

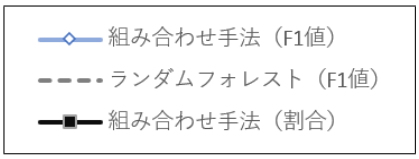

図 10 図 1〜図 9 における凡例
Fig. 10 Usage guide for Fig.1, . . . , Fig.9

このことから，バグモジュール発見の網羅性を高めるために相関ルール分析を用いることが有効に働く場合があるといえる．しかし懸念すべき点もある．実際にバグのないモジュールを誤ってバグありと判定するケースが recall に比例して増加することである．Ant の場合はランダムフォレストのみと比較して平均で 17.5%増加した．

続いて F1 値が悪化したデータセットについて考えるにあたり，Eclipse_jdt の詳細結果を**図 12** に示す．図 11 と同様の特徴がみられるものの，ルールでの判別を増やすほど F1 値は悪化することがわかった．precision に関してはランダムフォレストのみの手法と比較すると平均で 23.5%の悪化がみられ，recall に関しては平均で 16.6%の改善がみられた．Ant の場合と比較すると precision の低下率は大きく，recall の増加率が小さいことがわかる．F1 値が悪化するデータセットの共通の特徴として，テストデータとして用いた新バージョンのバグ含有率が他のデータセットと比較して低いことが共通している．このようなデータに対してルールベースでの分類を試みると網羅性の向上による恩恵を受けづらいと考えられる．

6. おわりに

本稿ではソフトウェアバグモジュール予測の精度向上を目的として相関ルール分析とランダムフォレストを組み合わせた手法を用いた．実験の結果，9 件のデータセットのうち 6 件で F1 値の向上が確認できた．得られた知見として，ルールベースの予測を行うことでおおまかな分類が行われ，見逃しが少なく網羅性が向上することがわかった．一方で適合率は低下する傾向にあり，これは複数あるメトリクスのうち最大でも 4 つのメトリクスのみでバグの発生する条件をつくるため，バグが発生しない要因を考慮できておらず例外的な処理ができていないと考えられる．またテストデータのバグ含有率が精度に影響することがわかった．バグ含有率が 20%を超えるデータに関しては，組み合わせることで得られる網羅性の利点を活かすことができる．

今後の課題としてはオーバーサンプリング等の前処理を行うことでモデル・ルール共に効果が得られるか否かを評価することや誤判定の割合を減らすこと，開発現場において提案手法が適しているかどうかを判断する指標について明らかにすることなどが考えられる．

参考文献

[1] 亀井靖高，森崎修司，門田暁人，松本健一：相関ルール分析とロジスティック回帰分析を組み合わせた fault-prone モジュール判別方法，情報処理学会論文誌，Vol.49, No.12, pp.3954-3699, Dec.2008.
[2] L. Breiman : Random Forests. Machine Learning 45, 5–32, 2001.
[3] S. Lessmann, B. Baesens, C. Mues, S. Pietsch : Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings,

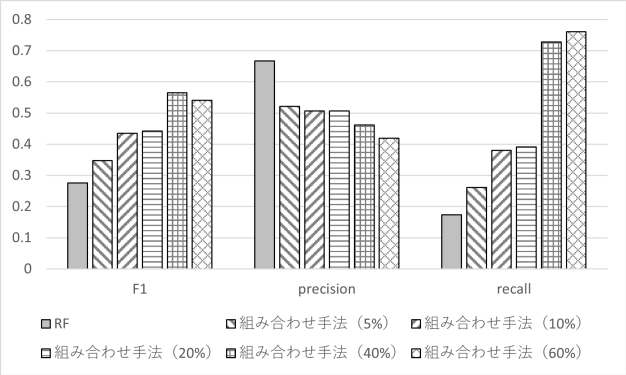

図 11 Ant に関する詳細結果（F1 値，precision，recall）
Fig. 11 Detailed results about Ant(F-measure, precision, recall)

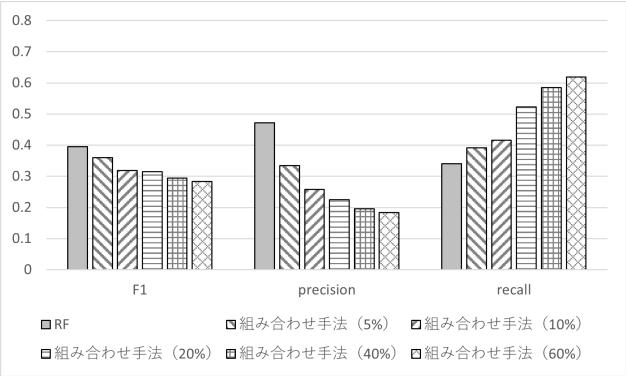

図 12 Eclipse_jdt に関する詳細結果（F1 値，precison，recall）
Fig. 12 Detailed results about Eclipse_jdt(F-measure, precision, recall)

IEEE Transactions on Software Engineering, Vol.34, No.4, pp.485-496, July-Aug. 2008.
[4] 金明哲：R によるデータサイエンス，森北出版株式会社
[5] Q. Song, M. Shepperd, M. Cartwright, C. Mair : Software defect association mining and defect correction effort prediction, IEEE Transactions on Software Engineering, Vol.32, No.2, pp.69-82, Feb. 2006.
[6] R. Agrawal, T. Imielinski, A. Swami : Mining association rules between sets of items in large databases, ACM SIGMOD Record, pp.207-216, June 1993.
[7] A. Monden, J. Keung, S. Morisaki, Y. Kamei, K. Matsumoto : A heuristic rule reduction approach to software fault-proneness prediction, Proc. Asia-Pacific Software Engineering Conference (APSEC2012), pp. 838-847, 2012.
[8] T. Menzies, R. Krishna, D. Pryor : The Promise Repository of Empirical Soft-ware Engineering Data, http://openscience.us/repo. North Carolina State University, Department of Computer Science, 2015.