NAIST-IS-DD0761011

博士論文

ソフトウェアモジュールの信頼性予測の精度向上に関する研究

亀井 靖高

2009年5月7日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

亀井 靖高

審査委員：

松本 健一 教授	(主指導教員)
関 浩之 教授	(副指導教員)
門田 暁人 准教授	(副指導教員)
楠本 真二 教授	(大阪大学)
Pattara Leelaprute 博士	(Kasetsart 大学)

ソフトウェアモジュールの信頼性予測の精度向上に関する研究*

亀井 靖高

内容梗概

ソフトウェアテストでは，限られたリソースで信頼性を確保するために，ソフトウェア中の各モジュールの信頼性（fault の有無）を予測し，fault を含むと判断されたモジュールにテスト工数を重点的に割り当てることが求められる．その手段として，従来，モジュールの特性を表すメトリクス値の集合から fault の有無を判別するモデル（以降，fault-prone モジュール判別モデル）が数多く提案されてきた．

本論文では，fault-prone モジュール判別モデルの性能向上を目的とし，(1) モデル構築用データ (フィットデータ) の偏りを解消するサンプリング法の適用，(2) モデル自体の表現能力の限界を補うためのルールベース判別との組み合わせ方法の提案，を行った．これによって，多くのテスト現場において，モデル構築に適したフィットデータの準備が容易になるとともに，fault の有無の判別精度が向上することで，より効果的なテストが行えることが見込まれる．本論文の具体的な成果は次のとおりである．

(1) サンプリング法適用の効果

従来，フィットデータに偏りがある，すなわち，fault を含むモジュールの個数が著しく少ない場合に，性能のよい判別モデルが構築できないということが課題であった．本論文ではフィットデータにサンプリング法を適用し，fault を含むモジュールの個数を制御することで，フィットデータの偏りを解消し，その効果を

* 奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DD0761011, 2009 年 5 月 7 日.

実験的に評価した．実験では，主要な 4 種類のサンプリング法（ROS，SMOTE，RUS，ONESS）適用の効果を，4 種類の判別モデル（線形判別分析，ロジスティック回帰分析，ニューラルネット，分類木) について評価した．その結果，線形判別分析とロジスティック回帰分析にはサンプリング法適用の効果があること，また，ONSSS は他の 3 つのサンプリング法よりも効果が小さいことがわかった．線形判別分析とロジスティック回帰分析の判別モデルに，ONESS 以外のサンプリング法を適用した場合，判別精度を表す F1 値が平均 0.125 向上した．

(2) ルールベース判別とモデルベース判別の組み合わせ方法

Fault-prone モジュール判別モデルは，それぞれモデル式の表現形式が決まっており，それらの表現上の制約のために，fault の有無を正しく判別できるモジュールは限られている．本論文では，fault の混入という事象を（一つのモデル式ではなく）多数のルールの集合として捉えるルールベース判別に着目し，ルールベース判別（相関ルール分析）と既存のモデルベース判別（ロジスティック回帰分析）の組み合わせ方法を提案した．提案手法では，与えられたモジュールに対し，重要なルール（支持度，信頼度，または，リフト値の大きなルール）が存在する場合は相関ルール分析によって判別し，そうでない場合は，ロジスティック回帰分析によって判別する．複数の重要なルールが存在する場合には，判別結果の多数決を行う．提案手法の判別性能を評価するために，3 つの代表的な fault-prone 判別モデル（ロジスティック回帰分析，線形判別分析，分類木）の性能と提案手法の性能を比較する実験を行った．その結果，重要とみなすルールの選択にはリフト値が適していることが分かり，リフト値に閾値を設けてルールを選定することで，F1 値が従来手法と比較して 0.163 向上した．

キーワード

Fault-prone モジュール判別，オーバーサンプリング，アンダーサンプリング，相関ルール分析

Performance Improvements for Reliability Prediction of Software Modules*

Yasutaka Kamei

Abstract

Prediction of reliability of software modules (fault-prone or not) and preferential allocation of test effort to modules identified as fault-prone is necessary for improving software reliability with a limited testing resource. So far, various multivariate models (fault-proneness models) that classify a module into either fault-prone or not fault-prone based on module metrics have been proposed.

The goal of this dissertation is to improve the prediction performance of fault-proneness models, by (1) applying sampling methods to a fit dataset to correct the imbalance of the dataset and (2) combining rule-based prediction and model-based prediction to compensate the restrictions in model descriptions. These achievements would help software developers prepare a suitable fit dataset for building the models and conduct software test more effectively. Achievements of this dissertation are described below:

(1) Applying sampling methods to a fit dataset

In conventional fault-proneness models, the prediction accuracy of the minority class (fault-prone modules) usually becomes worse, when a fit dataset is imbalanced, i.e. there exists a large difference between the number of fault-prone modules and not-fault-prone modules. This dissertation applied sampling methods to a fit dataset and corrected the imbalance of the dataset by increasing or

* Doctoral Dissertation, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0761011, May 7, 2009.

decreasing the cases (modules) in the dataset. This dissertation evaluated the effects of four sampling methods (ROS, SMOTE, RUS, ONESS) applied to four fault-proneness models (linear discriminant analysis, logistic regression analysis, neural network and classification tree). All four sampling methods improved the prediction performance of the linear and logistic models, and the improvement by ONESS was less than that of the other sampling methods. The improvements of F1-values in linear and logistic models were 0.125 at the mean when three sampling methods (ROS, SMOTE and RUS) were applied to linear discriminant analysis and logistic regression analysis.

(2) Combining rule-based prediction and model-based prediction

In conventional fault-proneness models, some types of modules are inherently not correctly predicted because of the restrictions of model descriptions (formula). This dissertation focused on rule-based prediction that could characterize various types of faulty modules by a large set of rules, and proposed a fault-prone module prediction method that combined association rule mining with a logistic regression model. In the proposed method, if a module satisfies the premise (i.e. the condition in the antecedent part) of one of the important rules (i.e. support, confidence or lift of the rules is large), the module is classified by the rule as either fault-prone or not. Otherwise, the module is classified by the logistic model. If there exist two or more rules, then the module is classified by the majority of rules' conclusion. This dissertation experimentally evaluated the prediction performance of the proposed method with different thresholds of each rule interestingness measure, and compared it with three well-known fault-proneness models (logistic regression model, linear discriminant model and classification tree). The result showed that the lift was the most suitable measure to select useful association rules in the proposed method compared to other measures. The improvement of the F1-value of the proposed method with the lift was 0.163 at maximum compared to conventional models.

Keywords:

Fualt-prone Module Detection, Over-sampling, Under-sampling, Association Rule Mining

関連発表論文

学術論文誌

1. 亀井 靖高, 柗本 真佑, 柿元 健, 門田 暁人, 松本 健一. Fault-prone モジュール判別におけるサンプリング法適用の効果. 情報処理学会論文誌, Vol. 48, No. 8, pp. 2651-2662, August 2007.（第 2 章に関連する）

2. 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 モジュール判別手法. 情報処理学会論文誌, Vol. 49, No. 12, pp. 3954-3966, December 2008.（第 3 章に関連する）

国際会議発表

1. Yasutaka Kamei, Akito Monden, Shinsuke Matsumoto, Takeshi Kakimoto, and Ken-ichi Matsumoto. The effects of over and under sampling on fault-prone module detection. In Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM2007), pp. 196-204, September 2007.（第 2 章に関連する）

2. Yasutaka Kamei, Akito Monden, Shuji Morisaki, and Ken-ichi Matsumoto. A hybrid faulty module prediction using association rule mining and logistic regression analysis. In Proceedings of the 2nd International Symposium on Empirical Software Engineering and Measurement (ESEM2008), pp. 279-281, October 2008.（第 3 章に関連する）

査読付き国内研究集会発表

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用いた Fault-Prone モジュール予測手法の提案. 岸知二, 野田夏子 (編), ソフトウェア工学の基礎 XIV -日本ソフトウェア科学会 FOSE2007-, pp. 125-130, 近代科学社, November 2007.

国内研究集会発表

1. 亀井 靖高, 門田 暁人, 松本 健一. Fault-Proneness モデルへのオーバーサンプリング法の適用. ソフトウェア信頼性研究会 第 3 回ワークショップ, pp. 97-103, July 2006.（第 2 章に関連する）

その他の発表論文

学術論文誌

1. 柗本 真佑, 亀井 靖高, 門田 暁人, 松本 健一. Fault-prone モジュール判別における外れ値除去法の比較. 情報処理学会論文誌, Vol. 49, No. 3, pp. 1341-1351, March 2008.

2. 亀井 靖高, 角田 雅照, 柿元 健, 大杉 直樹, 門田 暁人, 松本 健一. ソフトウェアコンポーネント推薦における協調フィルタリングの効果. 情報処理学会論文誌, Vol. 50, No. 3, pp. 1139-1143, March 2009.

3. 柗本 真佑, 亀井 靖高, 大平 雅雄, 松本 健一. OSS コミュニティにおけるオープンコラボレーションの理解. 情報社会学会誌, , Vol. 3, No. 2, pp. 29-42, March 2009.

4. 柿元 健, 門田 暁人, 亀井 靖高, 柗本 真佑, 松本 健一, 楠本 真二. Fault-Prone モジュール判別におけるテスト工数割り当てとソフトウェア信頼性のモデル化. 情報処理学会論文誌, (to appear).

国際会議発表

1. Takeshi Kakimoto, Akito Monden, Yasutaka Kamei, Haruaki Tamada, Masateru Tsunoda, and Ken-ichi Matsumoto. Using software birthmarks to identify similar classes and major functionalities. In Proceedings of the 3rd International Workshop on Mining Software Repositories (MSR2006) Mining Challenge 2006, pp. 171-172, May 2006.

2. Masateru Tsunoda, Akito Monden, Takeshi Kakimoto, Yasutaka Kamei, and Ken-ichi Matsumoto. Analyzing OSS developers' working time using mailing lists archives. In Proceedings of the 3rd International Workshop on Mining Software Repositories (MSR2006) Mining Challenge , pp. 181-182, May 2006.

3. Takeshi Kakimoto, Yasutaka Kamei, Masao Ohira, and Ken-ichi Matsumoto. Social network analysis on communications for knowledge collaboration in OSS communities. In Proceedings of the 2nd International Workshop on Supporting Knowledge Collaboration in Software Development (KCSD2006), pp. 35-41, September 2006.

4. Yasutaka Kamei, Akito Monden, and Ken-ichi Matsumoto. Empirical evaluation of SVM-based software reliability model. In Proceedings of the 5th International Symposium on Empirical Software Engineering (ISESE2006), Vol. 2, pp. 39-41, September 2006.

5. Shinsuke Matsumoto, Yasutaka Kamei, Akito Monden, and Ken-ichi Matsumoto. Comparison of outlier detection methods in fault-proneness models. In Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM2007), pp. 461-463, September 2007.

6. Shinsuke Matsumoto, Yasutaka Kamei, Masao Ohira, and Ken-ichi Matsumoto. A comparison study on the coordination between developers and users in FOSS communities. In Proceedings of the 1st Socio-Technical Congruence (STC 2008) , No. CD-ROM-No. 8, pp. 1–9, May 2008.

7. Masateru Tsunoda, Kohei Mitsui, Kyohei Fushida, Yasutaka Kamei, Masahide Nakamura, Keita Goto, and Ken-ichi Matsumoto. An authentication method combining spatiotemporal information and actions. In Proceedings of the 4th International Conference on Mobile Computing and Ubiquitous Networking (ICMU 2008), pp. 41-49, June 2008.

8. Yasutaka Kamei, Shinsuke Matsumoto, Hirotaka Maeshima, Yoji Onishi, Masao Ohira, and Ken-ichi Matsumoto. Analysis of coordination between developers and users in the Apache community. In Proceedings of the 4th

International Conference on Open Source Systems (OSS2008), pp. 81-92, September 2008.

9. Yasutaka Kamei, Jacky Keung, Akito Monden, and Ken-ichi Matsumoto. An over-sampling method for analogy-based software effort estimation. In Proceedings of the 2nd International Symposium on Empirical Software Engineering and Measurement (ESEM2008), pp. 312-314, October 2008.

査読付き国内研究集会発表

1. 伏田 享平, 亀井 靖高, 川口 真司, 飯田 元. 定量的測定データの体系化に基づいた開発プロセステーラリング方式の提案. 満田 成紀，羽生田 栄一（編）, ソフトウェアエンジニアリング最前線 2006-SES2006-, pp. 51-58, 近代科学社, October 2006.

2. 亀井 靖高, 門田 暁人, 松本 健一. SVM に基づくソフトウェア信頼性モデルの定量的評価. 岸知二, 野田夏子 (編), ソフトウェア工学の基礎 XIII -日本ソフトウェア科学会 FOSE2007-, pp. 65-70, 近代科学社, November 2006.

3. 前島 弘敬, 柗本 真佑, 亀井 靖高, 柿元 健, 大西 洋司, 大平 雅雄, 松本 健一. コーディネータのコミュニティ媒介性の評価指標の提案. 情報処理学会シンポジウム, Vol. 2007, No. 11, pp. 71-76, November 2007.

4. 柿元 健, 門田 暁人, 亀井 靖高, 柗本 真佑, 松本 健一. Fault-Prone モジュール判別における F1 値とソフトウェア信頼性の関係. 岸知二, 野田夏子 (編), ソフトウェア工学の基礎 XIV -日本ソフトウェア科学会 FOSE2007-, pp. 75-83, 近代科学社, November 2007.

5. 木浦 幹雄, 柗本 真佑, 亀井 靖高, 門田 暁人, 松本 健一. 異なるプロジェクト間における Fault-Prone モジュール判別の精度評価. ソフトウェア工学の基礎 XIV, 岸知二, 野田夏子 (編), ソフトウェア工学の基礎 XIV -日本ソフトウェア科学会 FOSE2007-, pp. 131-136, 近代科学社, November 2007.

6. 伊原 彰紀, 大平 雅雄, 柗本 真佑, 亀井 靖高, 松本 健一. 複数のサブコミュニティを有するOSSコミュニティを対象としたネットワーク分析. マルチメディア, 分散, 協調とモバイル (DICOMO2008) シンポジウム, pp. 297–303, July 2008.

7. 柿元 健, 亀井 靖高, 柗本 真佑, 門田 暁人, 松本 健一, 楠本 真二. バグ予測で信頼性はどれだけ向上するのか？-テスト工数割り当ての観点からの従来研究の評価. 松下誠, 川口真司 (編), ソフトウェア工学の基礎XV -日本ソフトウェア科学会 FOSE2008-, pp. 63-68, 近代科学社, November 2008.

8. 門田 暁人, 亀井 靖高, 上野 秀剛, 松本 健一. プロセス改善のためのソフトウェア開発タスク計測システム. 松下誠, 川口真司 (編), ソフトウェア工学の基礎XV -日本ソフトウェア科学会 FOSE2008-, pp. 123-128, 近代科学社, November 2008.

国内研究集会発表

1. 亀井 靖高, 角田 雅照, 柿元 健, 大杉 直樹, 門田 暁人, 松本 健一. 進行中のプロジェクトに有用なソフトウェアコンポーネントの推薦方法. 電子情報通信学会技術研究報告, ソフトウェアサイエンス研究会, Vol. 106, No. 16, pp. 25-30, April 2006.

2. 角田 雅照, 伏田 享平, 三井 康平, 亀井 靖高, 後藤 慶多, 中村 匡秀, 松本 健一. 位置と速度を利用した移動体向け認証方式の提案. 電子情報通信学会技術研究報告, モバイルマルチメディア通信研究専門委員会, No. MoMuC2006-55, pp. 11-16, November 2006.

3. 柗本 真佑, 亀井 靖高, 門田 暁人, 松本 健一. Fault-Proneモジュール判別モデルに対する外れ値除去法の適用効果. 情報処理学会研究報告, ソフトウェア工学研究会, Vol. 2007-SE-155, No. 33, pp. 49-56, March 2007.

4. 木浦 幹雄, 柗本 真佑, 亀井 靖高, 門田 暁人, 松本 健一. 他プロジェクトの実績データに基づくFault-Proneモジュール判別の試み. ソフトウェア信頼

性研究会 第 4 回ワークショップ, pp. 45-53, July 2007.

5. 左藤 裕紀, 亀井 靖高, 上野 秀剛, 川口 真司, 門田 暁人, 松本 健一. ソースコードの圧縮性とコードクローンの関係の分析. 日本ソフトウェア科学会第 24 回大会論文集 CD-ROM(講演番号 3B-3), September 2007.

6. 大平 雅雄, 柗本 真佑, 前島 弘敬, 亀井 靖高, 松本 健一. OSS コミュニティにおける共同作業プロセス理解のための中心性分析. 情報処理学会シンポジウムシリーズ, Vol. 2007, No. 11, pp. 7-12, November 2007.

7. 伊原 彰紀, 前島 弘敬, 柗本 真佑, 亀井 靖高, 大平 雅雄, 松本 健一. 複数のサブコミュニティを有する OSS コミュニティにおけるコーディネータの分析. 情報処理学会シンポジウム　グループウェアとネットワークサービス・ワークショップ 2007 論文集, Vol. 2007, No. 11, pp. 13-18, November 2007.

8. 山科 隆伸, 上野 秀剛, 伏田 享平, 亀井 靖高, 名倉 正剛, 川口 真司, 飯田 元. レガシーソフトウェア保守プロセスにおける開発者によるコードクローン認識についての観察. 情報処理学会第 70 回全国大会 講演論文集, Vol. 5, pp. 411-412, March 2008.

9. 山科 隆伸, 上野 秀剛, 伏田 享平, 亀井 靖高, 名倉 正剛, 川口 真司, 飯田 元. コードクローンに着目したソフトウェア保守支援ツールの設計と実装. 電子情報通信学会技術研究報告, ソフトウェアサイエンス研究会, Vol. 108, No. 64, pp. 65-70, May 2008.

10. 田村 晃一, 亀井 靖高, 上野 秀剛, 森崎 修司, 松村 知子, 松本 健一. 見逃し欠陥の回帰テスト件数を考慮したコードレビュー手法. 電子情報通信学会技術研究報告, ソフトウェアサイエンス研究会, Vol. 108, No. 173, pp. 61-66, July 2008.

11. 亀井 靖高, 大平 雅雄, 柗本 真佑, 松本 健一. Apache コミュニティにおける開発者とユーザとのコーディネーションの分析. 飯田元, 山本里枝子（編）, ソフトウェアエンジニアリング最前線 2008-SES2008-, pp. 183-184, 近代科学社, September 2008.

12. 左藤 裕紀, 亀井 靖高, 上野 秀剛, 川口 真司, 名倉 正剛, 門田 暁人, 松本 健一, 飯田 元. コードクローンの長さとソフトウェア信頼性の関係の分析. 電子情報通信学会技術研究報告, ソフトウェアサイエンス研究会, Vol. 108, No. 242, pp. 43-48, October 2008.

13. 伊原 彰紀, 亀井 靖高, 大平 雅雄, 柗本 真佑, 松本 健一. OSS プロジェクトにおける障害に関する情報共有の分析. 平成 20 年度 情報処理学会関西支部支部大会 講演論文集, Vol. 2008, pp. 69-72, October 2008.

14. 友谷 有希, 内田 眞司, 左藤 裕紀, 亀井 靖高, 門田 暁人. コードクローンメトリクスを説明変数とした Fault-Prone モジュール判別の試み. 平成 20 年電気関係学会関西支部連合大会講演論文集 CD-ROM(講演番号 G11-26), November 2008.

テクニカルノート

1. Takanobu Yamashina, Hidetake Uwano, Kyohei Fushida, Yasutaka Kamei, Masataka Nagura, Shinji Kawaguchi, and Hajimu Iida. SHINOBI: a real-time code clone detection tool for software maintenance. NAIST-IS-TR2007011, March 2008.

目 次

図 目 次

表 目 次

第1章　はじめに

1．研究の背景と目的

ソフトウェア開発プロジェクトにおいて，限られた開発期間で信頼性を確保するために，ソフトウェアテストの効率化が求められている [27]．その一手段は，ソフトウェア中の各モジュールの信頼性（fault の有無）を予測し，fault を含むと判断されたモジュールにテスト工数を重点的に割り当て，信頼性の確保と無駄なテスト工数の削減を両立することである [21]．そのため，モジュールの fault の有無を判別するモデル（以降，fault-prone モジュール判別モデル）が多数提案されている．Fault-prone モジュール判別モデルは，モジュールのメトリクス値（プログラム行数，サイクロマティック数，変更行数など）を説明変数とし，モジュールの fault の有無を目的変数とする数学的モデルであり，例えば，線形判別分析，ロジスティック回帰分析，ニューラルネットなどの手法が用いられる [8] [15] [26] [31].

Fault-prone モジュール判別モデルの構築から，fault の有無の判別までの流れを図 1.1 に示す．図 1.1 に示すように，まず，(1) 過去に開発されたモジュールのメトリクス値と fault の有無から判別モデルを構築する．次に，構築したモデルに対して，(2) 判別対象のモジュールから計測したメトリクス値を入力することで，(3) 当該モジュールが fault を含んでいるか否かが判別される．その後，fault-prone と判別されたモジュールに対して，fault-prone と判別されなかったモジュールよりも多くの工数を割り当てることで，信頼性の確保と無駄なテスト工数の削減を図る．

本論文では，fault-prone モジュール判別における，次の 2 つの問題に取り組む．まず，フィットデータに含まれる fault を含むモジュールの個数が（fault を含ま

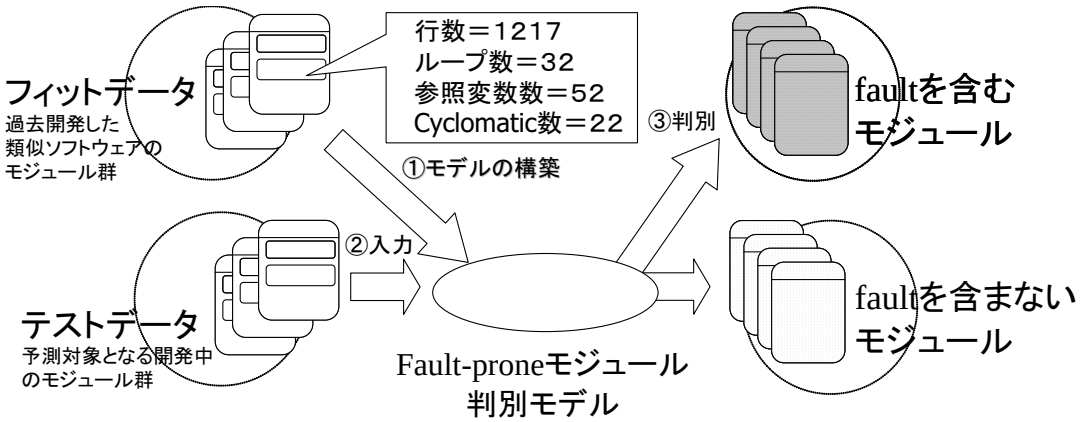


図 1.1 fault-prone モジュール判別の流れ

ないモジュールと比べて）著しく少ない場合，性能のよい判別モデルを構築することができない [16] という問題（**問題 1**）である．判別モデルでは，多数派の群をなす（fault を含まない）モジュールを正確に判別することが（モデル構築時の目的関数である）判別誤差の最小化に大きく寄与する．そのため，判別モデルでは少数派の群をなす（fault を含む）モジュールの判別の正確さが相対的に犠牲となる [36]．Fault を含むモジュールの個数が著しく少ない事例は，保守の現場で収集されるデータセットの中にしばしば見られる．例えば，NASA IV&V Facility Metrics Data Program[28] が公開しているデータセットの場合，11 件中 8 件のデータセットは fault モジュール含有率（fault を含むモジュールの数／全モジュール数）が 15%未満であった．

次に，fault-prone モジュール判別モデルは，それぞれモデル式の表現形式が決まっており，それらの表現上の制約のために，fault の有無を正しく判別できるモジュールが限られているという問題（**問題 2**）である．Fault-prone モジュール判別モデルは，過去に開発されたモジュールのメトリクス値と fault の有無の関係を抽象化し，単一のモデル式で表現するため，fault を含んでいる確率が高いモジュールの傾向を大まかに表すのに役立つ．しかしながら，単一のモデル式で表現するために，モジュールの性質と fault の有無の関係を必ずしもうまく抽象化できず，判別精度に限界がある．

本論文は，この 2 つの問題それぞれを解決し，fault-prone モジュール判別モデルの精度を向上することを目的とする．まず，**問題 1** の解決のために，サンプリング法に着目する．サンプリング法は，フィットデータに前処理を加えて偏りを解消する方法であり，今までに提案されてきた判別モデルの種類に依存せず，多くのモデルに適用できるという利点を持つ．前処理の具体的な内容は，(1)fault を含むモジュール（少数派の群）を人工的に増やす，もしくは (2)fault を含まないモジュール（多数派の群）のいくつかを削除することである．

次に，**問題 2** の解決のために，複数の判別手法を組み合わせた fault-prone モジュール判別方法を提案する．本論文では，非モデルベース（ルールベース）の相関ルール分析を，最も利用されている判別モデルの 1 つであるロジスティック回帰分析と組み合わせた fault-prone モジュール判別方法を提案する．相関ルール分析の特長の一つは，fault の混入という事象を（一つのモデル式ではなく）多数のルールの集合として捉えるため，多様な fault 混入の要因を抽出し，判別に用いるのに適していることである．提案手法では，与えられたモジュールに対し，重要なルール（支持度，信頼度，または，リフト値の大きなルール）が存在する場合は相関ルール分析によって判別し，そうでない場合は，ロジスティック回帰分析によって判別する．複数の適用可能なルールが存在する場合には，判別結果の多数決を行う．

2. 論文構成

本論文の主要部分は大きく 2 つの章から構成される．

第 2 章では，**問題 1** に対して，サンプリング法をフィットデータに適用し，どの程度の効果が得られるのかを実験的に確かめる．実験では，日本のある企業で開発されたレガシーソフトウェアから得られた 2 つのモジュール群（fault モジュール含有率 5.37%，および 11.63%）を題材として，4 種類のサンプリング法（ROS, SMOTE，RUS，ONESS）適用の効果を，4 種類の判別モデル（線形判別分析，ロジスティック回帰分析，ニューラルネット，分類木) について評価した．

第 3 章では，**問題 2** に対して，複数の判別手法を組み合わせた判別手法の提

案を行う．提案手法の判別性能を評価するために，3 つの代表的な fault-prone 判別モデル（ロジスティック回帰分析，線形判別分析，分類木）の性能と提案手法の性能を比較する実験を行った．実験では，National Aeronautics and Space Administration/West Virginia University（NASA/WVU） の公開しているデータセットと，Eclipse プロジェクトから収集したデータセットを対象として評価を行った．

そして，本論文の最後の 4 章で論文全体のまとめを述べる．

第2章　サンプリング法を用いた fault-prone モジュール判別モデルの精度評価

1. はじめに

Fault-prone モジュール判別モデルを構築する上で，フィットデータに含まれる fault を含むモジュールの個数が（fault を含まないモジュールと比べて）著しく少ない場合，性能のよい判別モデルを構築することができない [16]．判別モデルでは，多数派の群をなす fault を含まないモジュールを正確に判別することが（モデル構築時の目的関数である）判別誤差の最小化に大きく寄与する．そのため，判別モデルでは少数派の群をなす fault を含むモジュールの判別の正確さが相対的に犠牲となる [36]．フィットデータに fault を含むモジュールの個数が著しく少ない事例は，保守の現場で収集されるデータセットの中にしばしば見られる．例えば，Khoshgoftaar らが用いたデータセットでは，全モジュールに対して保守工程で発見された fault を含むモジュールの割合（以降，fault モジュール含有率 P_{fp}）は約 6%であった [18]．また，NASA IV&V Facility Metrics Data Program[28] が公開しているデータセットの場合，11 件中 8 件のデータセットは P_{fp} が 15%未満であった．なお，保守工程では fault モジュール含有率 P_{fp} が低いだけでなく，ほとんどのモジュールから高々1 個しかバグが検出されないことが多い．そのため各モジュールに含まれる fault の数ではなく，fault の有無の判別が一般的に行われる．

fault モジュール含有率 P_{fp} の低いデータセットのために，fault-prone モジュール判別モデルを拡張する方法が提案されている [18]．Khoshgoftaar らは，少数派

の群の判別精度を向上させるために，誤分類のコストを考慮したモデル構築を行っている．しかしながら，fault モジュールを判別するデータセットに対して，分類木が常にモデルの中で最も精度がよいとは限らない．一般に，高い判別精度が見込まれるモデルの種類は，データセットごとに異なる [8].

本章では，判別モデル自身を拡張するのではなく，フィットデータに前処理を加えて偏りを予め解消する方法（サンプリング法）に着目する．この手法は，判別モデルの種類に依存しないため，多くのモデルに適用できるという利点を持つ．この手法は，(1)fault を含むモジュール（少数派の群）を人工的に増やすオーバーサンプリング，(2)fault を含まないモジュール（多数派の群）のいくつかを削除するアンダーサンプリング，の二通りに分類される [6][19]．ただし，fault-prone モジュール判別モデルにおいて，サンプリング法を適用した事例はなく，いずれのサンプリング法が適しているかは定かではない．また，数ある判別モデル（線形判別分析，ロジスティック回帰，ニューラルネットなど）のそれぞれについて，サンプリング法がどの程度有効であるかも定かではない．

そこで，本章では，日本のあるソフトウェア開発企業で開発されたソフトウェアの保守工程で収集されたデータを対象とし，4 つのモデル（線形判別分析，ロジスティック回帰分析，ニューラルネット，分類木）におけるサンプリング法の効果を実験的に評価した．実験では，よく用いられている 4 種類のサンプリング手法（ランダムオーバーサンプリング，SMOTE，ランダムアンダーサンプリング，ONESS）を用い，それぞれサンプリングする割合を変化させて，各 fault-prone モジュール判別モデルの判別精度に与える影響を評価した．実験では，あるリリース前の 3 年間に発見された fault を記録したフィットデータを用い，リリース後 3 年間に検出された fault の判別を試みた．モデルの説明変数として，ソースコードの複雑さメトリクス 16 個と変更履歴メトリクス 3 個を各モジュールについて計測したものを用いた．

以降，2 章 2 節では判別モデルについて，2 章 3 節では本章で用いたサンプリング法について説明する．2 章 4 節では評価実験の方法と手順について説明する．2 章 5 節では評価実験の結果と，考察を述べる．2 章 6 節では関連研究を説明する．最後に 2 章 7 節でまとめについて述べる．

2. Fault-prone モジュール判別モデル

Fault-prone モジュール判別モデルでは，過去に開発されたモジュールのメトリクス値と fault の有無から判別モデルを構築する．構築したモデルに対して，モジュールから計測したメトリクス値を入力することで，当該モジュールが fault を含んでいるかが判別される．

2.1 線形判別分析

線形判別分析では，判別モデル（判別関数）は，ケースの特性を表す説明変数の線形結合により表される．各ケースに対して p 個の説明変数の値が観測されているとき，線形判別関数は式 (2.1) の形となる．

$$Z = \alpha_1 x_1 + \cdots + \alpha_p x_p \tag{2.1}$$

ここで，x_i は説明変数，α_i は判別係数を表す定数である．本章では，判別値 Z が 0 以上の値を取る場合，fault を含むモジュールであると判定する．

2.2 ロジスティック回帰分析

ロジスティック回帰分析では，判別関数はロジスティック関数の形で表現される．各ケースに対して p 個の説明変数の値が観測されているとき，判別モデル（判別関数）は式 (2.2) の形となる．

$$P(y|x_1, \cdots, x_p) = \frac{1}{1+e^{-(\alpha_1 x_1 + \cdots + \alpha_p x_p + \beta)}} \tag{2.2}$$

ここで，$y \in \{0, 1\}$ は群を表す目的変数，x_i は説明変数，α_i は判別係数であり，$P(y|x_1, \cdots, x_p)$ は，説明変数の値の組 $x_1, \cdots, x_p$ に対して，y が 1 の値を取る条件付き確率である．本章では $P(y|x_1, \cdots, x_p)$ が 0.5 以上の値を取る場合，fault-prone モジュールであると判定する．

2.3 ニューラルネット

ニューラルネットワークモデルは，人間の脳内の神経細胞間の信号伝達を模したネットワーク構造により説明変数と目的変数の関係を表現する [2]．モデルの出力に閾値を設定することで，2 群判別に用いることができる．本章では，ニューラルネットワークモデルに 3 層のパーセプトロン（中間層のユニット数は，栗田の研究 [20] で精度の高かった 3 個とした.）を，学習アルゴリズムに誤差逆伝播法 [32] を用い，2 群を判別するための出力値が 0.5 以上の場合，fault を含むモジュールであると判定する．

2.4 分類木

分類木は，説明変数と目的変数の関係を木構造で表現したモデルである．木の各ノードは 2 つ以上の子ノードを持っており，説明変数の値によっていずれかの子ノードへと分岐する．リーフノードは，いずれかの群に割り当てられている．与えられたケースは，説明変数の値に従ってルートノードから木をたどることで，リーフノードにおいて群の判別が行われる．本章では，分類木を構築するためのアルゴリズムとして fault-prone モジュール判別で一般的に用いられている CART (Classification And Regression Trees) [15] を用いた．

3. サンプリング法

サンプリング法とは，判別モデルを構築する前に，フィットデータに含まれるケース（本章ではモジュールに該当する）を増加もしくは減少させることで，データセットの偏りを補正する前処理である．サンプリング法は，オーバーサンプリングとアンダーサンプリングに分類される．オーバーサンプリングは，少ない方の群のケース（以降，少数派ケース）を人工的に増やすことで多い方の群のケース（以降，多数派ケース）の数に近づけ，データセットの偏りを補正する．一方，アンダーサンプリングは，多数派ケースのいくつかを削除することで少数派ケースの数に近づけ，データセットの偏りを補正する．

一般に，判別モデルの構築において，多数派ケースを正確に判別することが判別誤差の最小化につながるため，少数派ケースの判別精度が犠牲となる [36]．サンプリング法を用いて，群間の偏りをなくすことで，結果として，少数派ケースの判別精度の向上が見込まれる．ただし，オーバーサンプリングでは，人工的に増やしたケースに過度に適合（オーバーフィッティング）するという可能性も否定できない．また，アンダーサンプリングでは，母集団の性質を表すのに必要なケースが削除される可能性がある．

以降では，本章の評価実験で用いたサンプリング手法について述べる．

3.1 オーバーサンプリング

本章では，オーバーサンプリングとしてランダムオーバーサンプリング（以降，ROS）と Synthetic Minority Over-sampling TEchnique（以降，SMOTE）を用いる．ROS は，オーバーサンプリングの中で最も単純なアルゴリズムであるが，少数派ケースを複製して追加するため，オーバーフィッティングする可能性が高い [4]．SMOTE では，少数派ケースを複製するのではなく，k-最近傍のケースを基に新たなケースを生成することで，オーバーフィッティングする可能性を下げている [6]．

ランダムオーバーサンプリング（ROS）

ランダムオーバーサンプリングは，少数派ケースをランダムに複製することで少数派ケース数を増加させる．ROS では追加するケースの数だけ以下の手順を繰り返す．

手順 1．ケースの選択　少数派ケースをランダムに 1 つ選択する．

手順 2．ケースの複製・追加　手順 1 で選択されたケースを複製し，新たなケースとしてデータセットに追加する．

Synthetic Minority Over-sampling TEchnique(SMOTE)

Chawla らは，ケースの複製によるオーバーフィッティングを緩和するために，選択したケースを複製するのではなく，k-最近傍のケースを基に新たなケースを生成する方法である SMOTE を提案している [6]．SMOTE の手順を以下に示す．

手順 1．ケースの選択　少数派ケース m_a を 1 つ選択する（図 2.2）．

手順 2．k-最近傍の特定　m_a の k-最近傍は，類似度計算により特定される．本章では，k の値として Chawla らが用いた $k = 5$ を採用した [6]．類似度は以下の方法で求められ，m_a の k-最近傍（図 2.3 の点線の×）が特定される．

手順 2-1．説明変数の値の正規化　SMOTE では k-最近傍のケースを求めるために，各ケースの説明変数の値の組に基づいてケース間の類似度を算出する．ただし，fault-prone モジュール判別においては，説明変数（ソースコードの複雑さメトリクスや変更履歴メトリクス）ごとに値域が異なるため，何らかの正規化が必要となる．そこで，本章では，Chawla らが提案した手法に説明変数の値を正規化する手順を追加し，各説明変数の値域を $[0, 1]$ に統一した．ケース m_i の説明変数 f_j の値 $v_{i,j}$ の正規化された値 $norm(v_{i,j})$ は式 (2.3) によって求められる．

$$norm(v_{i,j}) = \frac{v_{i,j} - min(f_j)}{max(f_j) - min(f_j)}. \tag{2.3}$$

ここで，$min(f_j)$ と $max(f_j)$ はそれぞれメトリクス f_j のデータセットにおける最小値と最大値である．

手順 2-2．説明変数の値の正規化　新たなケースを作成するために，各少数派ケース間の類似度を求める．本章では，ベクトル間の類似度指標として一般的に用いられているユークリッド距離を用いた．ケース m_a とケース m_i との類似度 $sim(m_a, m_i)$ は式 (2.4) で定義される．

$$sim(m_a, m_i) = \sqrt{\sum_{j=1}^{n} (v_{a,j} - v_{i,j})^2}. \tag{2.4}$$

ここで，$v_{i,j}$ はケース m_i の説明変数 f_j の値，n はデータセットの説明変数の数である．

手順 3．k-最近傍からケースの選択　k-最近傍からランダムにケース m_r を 1 つを求める（図 2.4）．

手順 4．ケースの追加　ケース m_a と m_r のベクトルの頂点を結ぶ直線上のランダムな場所に新たなケース（図 2.5 の太線の×）を追加する．

手順 5．ケース追加の繰り返し　手順 3 から手順 4 を（SMOTE によって追加する少数派ケース数/元の少数派ケース数）回繰り返す（図 2.6）．

SMOTE では，手順 1 から 5 をそれぞれの少数派ケースに対して実行する．

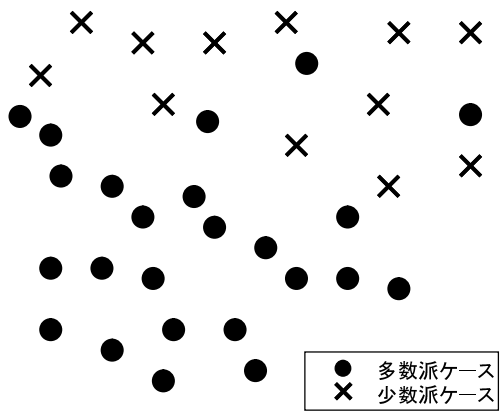


図 2.1　元のデータセット S

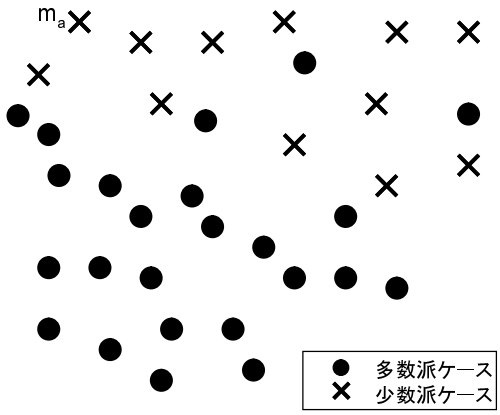


図 2.2 m_a の選択（手順 1 の後）

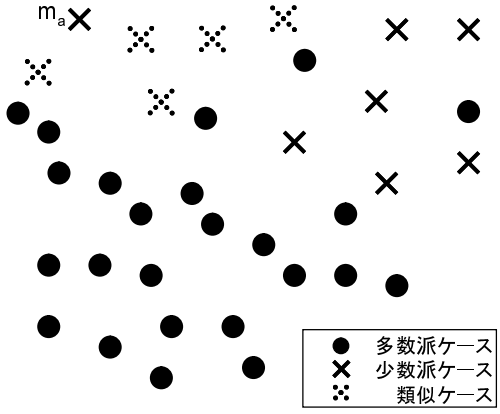


図 2.3 k-最近傍の特定（手順 2 の後）

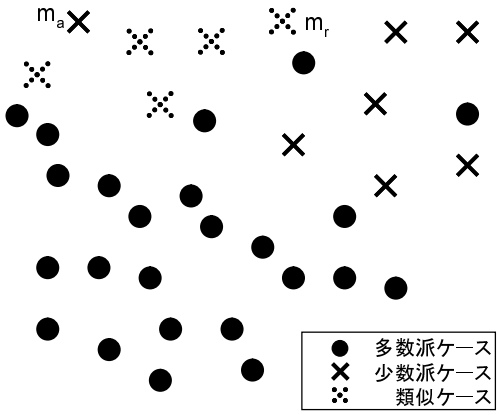


図 2.4 m_r の選択（手順 3 の後）

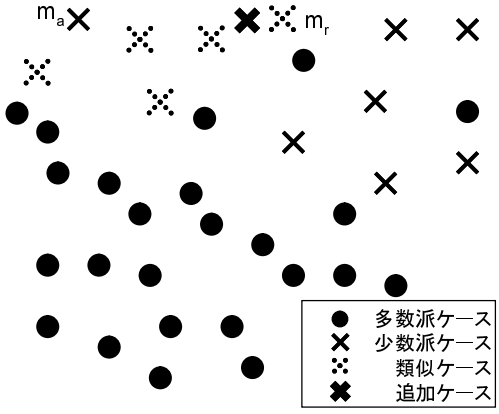


図 2.5 新たなケースの追加（手順 4 の後）

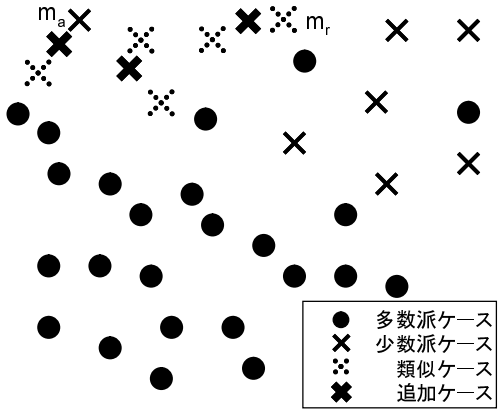


図 2.6 ケース追加の繰り返し後のデータセット（手順 5 の後）

3.2 アンダーサンプリング

本章では，アンダーサンプリングとしてランダムアンダーサンプリング（以降，RUS）と One-sided selection（以降，ONESS）を用いる．RUS は，アンダーサンプリングの中で最も単純なアルゴリズムであるが，多数派ケースをランダムに削除するため，母集団の性質を表すのに必要なケースが削除される可能性が高い [4]．ONESS では，多数派ケースをランダムに削除するのではなく，モデルを構築する上で冗長なケースもしくはノイズとなるケースを削除することで，母集団の性質を表すのに必要なケースが削除される可能性を下げている [6].

ランダムアンダーサンプリング（RUS）

RUS は，データセットから多数派ケースをランダムに選択し，削除することで多数派ケースの数を減少させてデータセットの偏りを解消する手法である．RUS では以下の手順を削除するケースの数だけ繰り返す.

手順 1．ケースの選択　多数派ケースをランダムに 1 つ選択する．

手順 2．ケースの削除　手順 1 で選択されたケースをデータセットから削除する．

One-sided selection(ONESS)

Kubat らは，母集団の性質を表すのに必要なケースを削除する可能性があるアンダーサンプリングの問題点を解決するために，Tomek links[38] を用いたアンダーサンプリング手法の ONESS を提案している [19]．Tomek links とは，$\delta(x,z) < \delta(x,y)$ もしくは $\delta(y,z) < \delta(y,x)$ を満たすケース z がデータセット中にないような，x と y の組である [38]．Kubat らは，$\delta(x,y)$ を少数派ケース x と多数派ケース y の距離と定義している．Tomek links である x と y はそれぞれ最も距離が近いケースの組であり，そのような組は多数派と少数派の境界付近，もしくは境界の少数派側にある少数派ケースと多数派ケースの組である可能性が高い．そのため ONESS では，境界付近，もしくはノイズであると考え，Tomek links に含まれる多数派ケースを削除する．さらに Kubat らは，多数派の類似したケース（図 2.7 の左下

にある黒丸）が判別モデルを構築する上で冗長なケースであると考え，1-最近傍に基づいて，その冗長な多数派ケースを削除する．ONESS の手順を以下に示す．

手順 1. ケースの選択　元のフィットデータ S（図 2.7）から，ランダムに 1 つ多数派ケース R を選択する．S に含まれる全ての少数派ケースと，ランダムに選択した多数派ケース R を合わせて，データセット C とする（図 2.8）．

手順 2. 冗長なケースの削除　S に含まれるそれぞれのケースを，データセット C に基づく 1-最近傍により fault を含むモジュールであるかどうか分類する．1-最近傍により分類された S のケースのうち，少数派ケースと分類された全ての多数派ケースを C に追加する．その結果，データセット C から冗長な多数派ケースが削除される（図 2.9）．

本章では，1-最近傍を求める際に SMOTE と同様，Kubat らが提案している手法に説明変数の正規化を行った．また，1-最近傍を求めるための類似度指標として，SMOTE と同様にユークリッド距離を用いた．

手順 3.Tomek links の削除　C に含まれる Tomek links（図 2.10 の破線で囲まれたケースの組）のうち，多数派のケースを削除する (図 2.10)．このデータセット C をモデル構築に用いるフィットデータ T とする（図 2.11）．

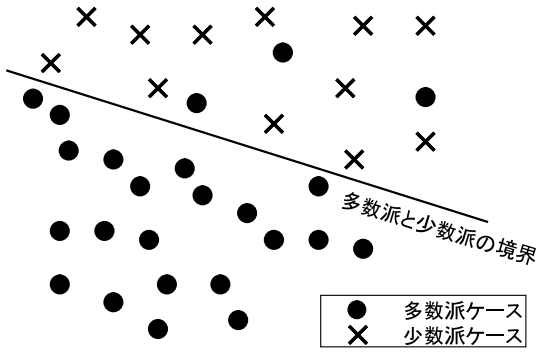


図 2.7　元のデータセット S

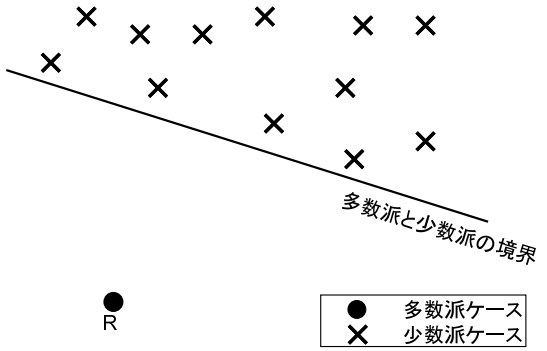


図 2.8 R と全ての少数派ケースを含むデータセット C（手順 1 の後）

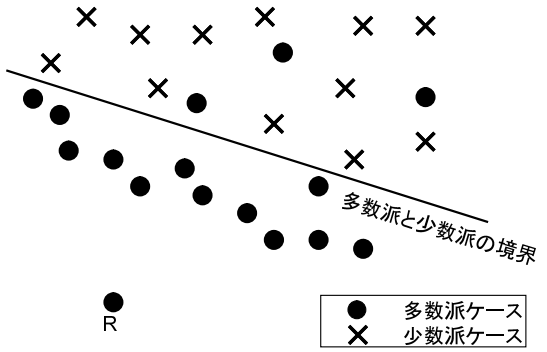


図 2.9 冗長な多数派ケースを削除したデータセット C（手順 2 の後）

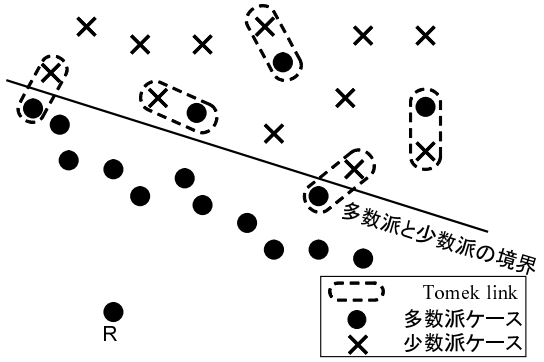


図 2.10 C に含まれる Tomek links

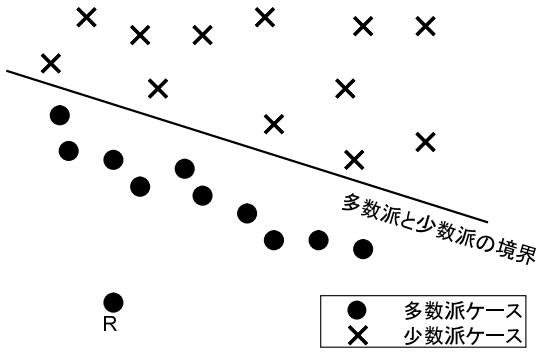


図 2.11 モデル構築に用いるフィットデータ T（手順 4 の後）

4. 適用実験

4.1 概要

実験では，fault を含むモジュールの個数が（fault を含まないモジュールと比べて）著しく少ないデータセットを対象とし，サンプリング法適用の効果を実験的に評価する．実験では，4 種類のサンプリング法（ROS，SMOTE，RUS，ONESS），4 種類のモデル（線形判別分析，ロジスティック回帰分析，ニューラルネット，分類木）の全ての組み合わせについて，それぞれ判別精度を評価する．

なお，ROS，SMOTE，RUS の 3 種類のサンプリング法については，サンプリングを行う度合いを調整することが可能である．例えば，ROS では，追加するケースの数を変化させることで，データセットの fault モジュール含有率 P_{fp} を自由に調整可能である．P_{fp} は，群間の偏りをなくす観点からは 50% に設定すべきであるが，2 章 3 節で述べたように，ケースを追加／削除しすぎるとオーバーフィッティングや必要なケースが削除されるという問題がある．そのため，データセットの P_{fp} を 50% にすることが必ずしも最適とは限らない．

そこで本実験では，まず，フィットデータだけを用いて，最適な fault モジュール含有率を決定するための事前実験を行う．次に，事前実験で決定した fault モジュール含有率に基づいて，フィットデータに対してサンプリング法を適用し，判別モデルを構築する．各サンプリング法はランダム要素を含むため，各手法で 5 回ずつサンプリング法を適用し，それぞれ判別モデルを構築する．最後に，テストデータを用いて判別モデルの判別性能（5 回の平均値）を，サンプリング法を適用しない場合と比較することで評価する．

本実験では，4 種類の fault-prone モジュール判別モデルの構築に，SPSS 社のデータマイニングソフトである Clementine を用いた．線形判別分析とロジスティック回帰分析のモデル構築では，変数選択（変数減少法）を行った．ニューラルネットのモデル構築では，学習回数をそれぞれ 10000，20000，30000，50000，100000 回に設定し，仮実験を行った．仮実験の結果，学習回数として最も精度の高かった 10000 回に設定し，モデルを構築した．

4.2 データセット

本章の実験で対象としたデータセットは，日本のあるソフトウェア開発企業で開発された大規模レガシーソフトウェアの，保守期間中に収集された各モジュールのメトリクス値を記録したものである．開発されたソフトウェアは，Capers Jones の分類 [14] における MIS(Management Information System) に該当する，企業，官公庁，銀行等における業務の基盤となるアプリケーションソフトウェアであり，主に 2 種類の開発言語（ともに手続き型言語）で記述されている．本ソフトウェアは，初期リリースから約 20 年間保守され続け，ある期間（数年ごと）で新たな要求による機能追加や変更，および多くの fault が修正されリリースされている．

本実験では，一方の言語で実装されたモジュールの集合をデータセット A，もう一方の言語のモジュールの集合をデータセット B と呼ぶ．また，A，B それぞれのデータセットにおけるフィットデータを A_{fit}，B_{fit}，テストデータを A_{test}，B_{test} と記す．各データセットの P_{fp} を表 2.1 に示す．

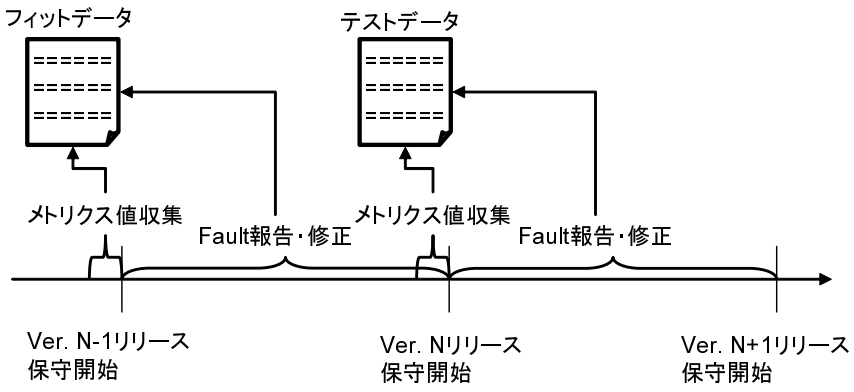


図 2.12 保守期間におけるフィットデータとテストデータの収集時期

本実験では，保守におけるあるリリース前の 3 年間に発見された fault の記録をフィットデータ（A_{fit} の P_{fp}:5.37%，および B_{fit} の P_{fp}:11.63%）として用い，リリース後 3 年間に検出された fault をテストデータ（A_{test} の P_{fp}:2.11%，および

表 2.1 各データセットの統計量

	A_{fit}	A_{test}	B_{fit}	B_{test}
fault を含むモジュールの数（個）	103	42	210	61
fault を含まないモジュールの数（個）	1815	1950	1596	1815
fault モジュール含有率（%）	5.37	2.11	11.63	3.25

B_{test} の P_{fp}:3.25%）として用いた（図 2.12）．モデルの説明変数として，ソースコードの複雑さメトリクス 16 個と変更履歴メトリクス 3 個を各モジュールについて計測したものを用いた．

フィットデータとテストデータでモジュール数が異なる理由は，リリースにおいてモジュールが新たに追加されたためである．それぞれのデータセットの 20 個のメトリクスの名称を表 2.2 に示す．語尾に「/LOC」がついているものは，行数で正規化されたメトリクスである．表 2.2 に示すメトリクスのうち，モジュールに含まれる fault の有無を目的変数とし，それ以外のメトリクスを説明変数の候補とした．

4.3 評価基準

fault モジュール判別の評価基準として，再現率，適合率，および F1 値 [11] を用いた．再現率は，fault を含むすべてのモジュールのうち，fault を含むモジュールと判別した割合を表し，表 2.3 で示す記号を用いて式 (2.5) で定義される．

$$\text{再現率} = \frac{n_{22}}{n_{21} + n_{22}} \tag{2.5}$$

また，適合率は，fault を含むモジュールと判別したモジュールのうち，実際に fault を含むモジュールである割合を表し，表 2.3 で示す記号を用いて式 (2.6) で定義される．

$$\text{適合率} = \frac{n_{22}}{n_{12} + n_{22}} \tag{2.6}$$

表 2.2 適用実験に用いたデータセットのメトリクスの名称

	メトリクス
m1	fault の有無
m2	プログラムの総行数（Lines of Code）
m3	コメント行数/LOC
m4	モジュール内手続き数/LOC
m5	オペレータの種類
m6	オペランドの種類
m7	オペレータ総出現数/LOC
m8	オペレンド総出現数/LOC
m9	Halstead 尺度の Volume
m10	Halstead 尺度の Difficulty
m11	最大ネストレベル
m12	サイクロマティック数/LOC
m13	ネストレベル/LOC
m14	ジャンプ文数（GOTO 文）/LOC
m15	外部変数参照数/LOC
m16	内部手続き呼び出し数/LOC
m17	外部モジュール呼び出し数/LOC
m18	改訂数
m19	モジュールが生成された日からの日数
m20	モジュールが生成された日から最終改訂日までの日数

表 2.3 判別結果の分類

	fault を含まないと判別	fault を含むと判別
実際に fault を含まない	n_{11}	n_{12}
実際に fault を含む	n_{21}	n_{22}

Fault-prone モジュール判別モデルを評価するためには，再現率と適合率のバランスが重要である．再現率が高くても，適合率が低ければ精度が高いとは言えず，また適合率が高くても，再現率が低ければ同じく精度が高いとは言えない．そこで本章では，適合率と再現率のバランスを考慮した指標である F1 値を評価基準として用いた．F1 値は，式 (2.7) として定義され，値域は $[0, 1]$ となる．F1 値が 1 のとき判別精度が最も高く，0 のとき最も低い．

$$F = \frac{2 \times \text{再現率} \times \text{適合率}}{\text{再現率} + \text{適合率}} \tag{2.7}$$

4.4 実験の手順

最適な fault モジュール含有率の決定（事前実験）

サンプリング法適用時の最適な fault モジュール含有率を決定するための事前実験を，フィットデータだけを用いて（交差検証法により）行った．最適な fault モジュール含有率は，ONESS を除く 3 種類のサンプリング法（ROS，SMOTE，RUS）と 4 種類のモデル（線形判別分析，ロジスティック回帰分析，ニューラルネット，分類木）の組み合わせ計 12 通りについて求めた．具体的には，12 通りの組み合わせそれぞれについて下記の手順を実施した[1] ．

手順 1．フィットデータの分割　フィットデータを 2 等分し，一方のデータをデータセット α，もう一方のデータをデータセット β とした．

[1] ONESS については，適用した場合としない場合のデータセットを用いて，各判別モデルの精度を比較した．

手順 2. サンプリング法の適用　データセット α に対して，fault モジュール含有率 P_{fp} が 20%，33%，50%，60%，67% となるようにサンプリングを行い，つまり（fault を含むモジュールの数:fault を含まないモジュールの数）の比が (0.25 : 1.00)，(0.50 : 1.00)，(1.00 : 1.00)，(1.50 : 1.00)，(2.00 : 1.00) となるようにサンプリングを行い，データセット $\alpha_1, \cdots, \alpha_5$ を作成した．

手順 3. 判別モデルの構築　データセット $\alpha_1, \cdots, \alpha_5$，および，サンプリング法を適用しない場合のデータセット α_0 それぞれを用いて，判別モデルを 6 つ構築する．

手順 4. 判別モデルの精度の評価　データセット β を用いて，各判別モデルの判別精度を評価する．

手順 5. fault モジュール含有率の決定　最も精度の良い（F1 値の大きい）判別モデルが得られた P_{fp} を採用する．

サンプリング法の効果の評価

2 章 4.4 項の事前実験で決定した fault モジュール含有率 P_{fp} に基づき，次の手順によりサンプリング法の効果を評価した．

手順 1. サンプリング法の適用　事前実験で決定した P_{fp} になるように，フィットデータに対してサンプリング法を適用した．

手順 2. 判別モデルの構築　サンプリング法を適用したフィットデータを用いて，4 種類の判別モデルを構築する．また，サンプリング法を適用しない場合の判別モデルを構築し，テストデータに対して精度評価を行った．

手順 3. モデルの精度の評価　予測したモジュールの fault の有無と，実際のモジュールの fault の有無から，2 章 4.3 項で説明した各評価基準で各 fault モジュール判別モデルの精度を求めた．

表 2.4 事前実験により決定された fault モジュール含有率（データセット A）

	LDA(%)	LRA(%)	NN(%)	CT(%)
ROS	67	20	33	67
SMOTE	67	20	20	50
RUS	60	20	20	未適用
ONESS	未適用	適用	適用	適用

表 2.5 事前実験により決定された fault モジュール含有率（データセット B）

	LDA(%)	LRA(%)	NN(%)	CT(%)
ROS	67	20	20	67
SMOTE	67	33	未適用	67
RUS	67	33	20	33
ONESS	適用	適用	適用	未適用

5. 結果と考察

5.1 Fault モジュール含有率に関する結果と考察

Fault モジュール含有率 P_{fp} を変化させたときの，fault モジュール判別モデルの F1 値を図 2.13 に示す．グラフの縦軸は F1 値を，横軸は fault モジュール含有率を示す．グラフの左端の値は，サンプリング法を適用しなかった場合の F1 値を示す．また，事前実験において決定された最適な fault モジュール含有率を，各サンプリング手法，および各判別モデルについてまとめたものを，データセット A については表 2.4 に，データセット B については表 2.5 に示す．表 2.4 と表 2.5 中の「LDA」は線形判別分析を,「LRA」はロジスティック回帰分析を,「NN」はニューラルネットを,「CT」は分類木を示し,「未適用」はサンプリング法を適用しなかった場合を示す．図 2.13 と表 2.4，表 2.5 より，fault モジュール含有率が

必ずしも最適とならない（むしろ最適でないことの方が多い）ことがわかった．また，fault モジュール含有率と F1 値の関係に着目すると，データセットに関わらず，判別モデルごとに次の特徴が見られた．

- **線形判別分析**　サンプリング未適用または P_{fp} が 20% のときに最も精度が低く，P_{fp} が高い（$60\% - 67\%$）ときに最も精度が高くなっている．
- **ロジスティック回帰分析**　P_{fp} が $20\% - 33\%$ のときに最も精度が高くなり，P_{fp} をさらに高くすると精度が下がっていく傾向にある．ロジスティック回帰分析には，サンプリング法をわずかに適用するべきであり，これは線形判別分析と逆の傾向にあるといえる．
- **ニューラルネット**　P_{fp} が 33% 付近のときに最も精度が高くなり，P_{fp} をさらに高くすると精度が下がっていく傾向にある．ニューラルネットには，サンプリング法をわずかに適用するべきであり，これはロジスティック回帰分析と似た傾向にあるといえる．
- **分類木**　サンプリング法の適用によって判別精度が低下する場合がある．また，精度が向上する場合でも，その効果は小さい．

以上のことから，モデルによって最適な fault モジュール含有率は異なることがわかった．また，いずれのデータセット，サンプリング法を用いた場合においても，平均的な傾向として，(1) 線形判別分析では P_{fp} が $60\% - 67\%$ 付近のときに精度が高い，(2) ロジスティック回帰分析およびニューラルネットでは P_{fp} が $20\% - 33\%$ のときに精度が高い，(3) 分類木ではサンプリング法の効果はほとんど見られない，ことがわかった．

(1)(2) の結果から，線形判別分析はモジュール比が 1.0 を超えるほどの強いオーバー／アンダーサンプリングに対してもロバスト（精度の低下を招きにくい）であり，逆にロジスティック回帰分析とニューラルネットワークは強すぎるオーバー／アンダーサンプリングに対してロバストでないことがうかがえる．一つの解釈として，線形判別分析は（他の 2 つのモデルよりも）モデル自身が単純であるために，追加／削除されたケースに過度に適合する（オーバーフィッティング）ことが起こらず，結果としてロバストであったことが考えられる．

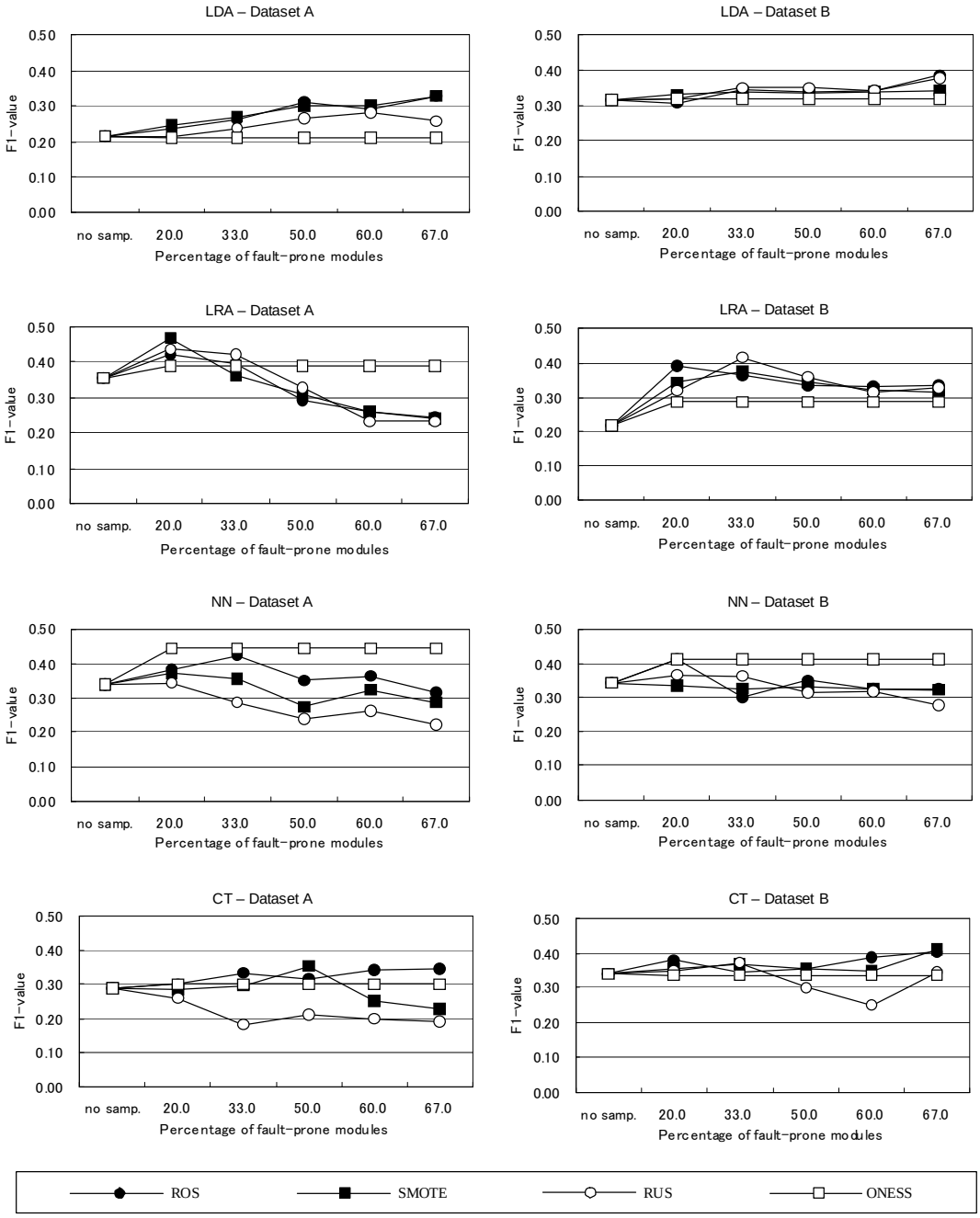


図 2.13 構築する判別モデルごとのサンプリング法適用の効果

5.2 サンプリング法適用の結果と考察

事前実験において決定された最適な fault モジュール含有率に従って，サンプリング法の適用，判別モデルの構築，および判別精度の評価を行った結果を，データセット A については表 2.6 に，データセット B については表 2.7 に示す．表 2.6，表 2.7 中の「LDA」は線形判別分析を,「LRA」はロジスティック回帰分析を,「NN」はニューラルネットを,「CT」は分類木を示し,「-」はサンプリング法を適用しなかった場合を示す.

- **判別モデル間の比較**　実験の結果，線形判別分析とロジスティック回帰分析の判別モデルにおいて，4 種類のサンプリング法のいずれを適用した場合にも判別精度が向上した.

  一方，ニューラルネットと分類木の判別モデルでは，判別精度が向上しなかった．むしろ，サンプリング法を適用しない判別モデルのほうが，判別精度が高い場合もあった．そのため，ニューラルネットと分類木では，サンプリング法を適用するべきでないといえる.

  なお，最も精度（F1 値）の高い判別モデルは，データセット A ではロジスティック回帰分析，データセット B では線形判別分析となった．この結果は，最適なモデルはデータセットによって異なるという Gray らの結果 [8] に反しない.

- **サンプリング手法間の比較**　最も効果の高いサンプリング手法は，判別モデルごと，および，データセットごとに異なっていた．例えば，データセット A では，SMOTE とロジスティック回帰分析の組み合わせにおいて最も F1 値が高くなった（0.382）．データセット B では，RUS と線形判別分析の組み合わせが最も高くなった（0.324）.

  線形判別分析とロジスティック回帰分析の結果に着目すると，それぞれのサンプリング法（ROS，SMOTE，RUS，ONESS）を適用した場合，ONESS はいずれのデータセットにおいても効果が小さく（平均 F1 値は 0.224），残りの 3 つ（ROS, SMOTE, RUS）は互いに優劣が付けられなかった（それ

ぞれ平均 F1 値は 0.276，0.287，0.293 であった）．この傾向は，F1 値そのものではなく,（サンプリング法適用による）F1 値の上昇幅に着目した場合においても，同様であった．線形判別分析とロジスティック回帰分析の判別モデルに，ONESS 以外のサンプリング法を適用した場合，F1 値の向上は最小 0.078，最大 0.224，平均 0.125 であった．

ONESS によって削除されたモジュール数について調べると，データセット A_{fit} では 1,815 個の fault を含まないモジュールのうち，145 個のモジュールしか削除されておらず，fault モジュール含有率は 5.37%から 5.81%にしか変化していなかった．また，データセット B_{fit} では 1,596 個の fault を含まないモジュールのうち，147 個のモジュールしか削除されておらず，fault モジュール含有率は 11.63%から 12.66%にしか変化していなかった．すなわち，ONESS を適用してもデータセットの偏りが解消されていない．そのため，判別精度が向上しなかったと考えられる．ONESS では削除するモジュール数を指定することができないため，今回の結果のようにデータセットの偏りが解消されない可能性がある．今後の改善策として，フィットデータに ONESS を繰り返し適用することで，モジュール比を制御することが考えられる．

ただし，本章の結果は，日本のソフトウェア開発企業から収集した 2 つのデータセットから得られたものであり，結果の信頼性を向上させるためには他のデータセットを用いて実験を追加する必要がある．また，判別モデルにおいて一般的に利用されているメトリクス（Cyclomatic 複雑度や Halstead 複雑度）を用いているので,（特殊なメトリクスを用いるよりは）一般性のある結果であると考えるが，一般性をより向上させるためには今回用いた以外のメトリクスによる実験を追加する必要がある．

結果を一般化する上での他の制約として，本章の結果が手続き型言語で実装されたソフトウェアに基づくものであることがあげられる．結果の一般性を高めるために，今後オブジェクト指向言語でも同様の実験をすることが望まれる．

今回のソフトウェアが，長年保守され続けたものであるということも結果を一般化する上での制約の一つである．保守開始直後のソフトウェアを用いた場合，

表 2.6 判別モデルとサンプリング法の各組み合わせの判別精度（データセット *A*）

		LDA	LRA	NN	CT
再現率	未適用	0.857	0.238	0.333	0.762
	ROS	0.752	0.629	0.419	0.648
	SMOTE	0.791	0.595	0.443	0.733
	RUS	0.833	0.624	0.538	0.729
	ONESS	-	0.286	0.414	0.729
適合率	未適用	0.057	0.357	0.233	0.062
	ROS	0.146	0.266	0.142	0.094
	SMOTE	0.137	0.282	0.168	0.080
	RUS	0.117	0.257	0.136	0.088
	ONESS	-	0.375	0.228	0.088
F1 値	未適用	0.106	0.286	0.275	0.115
	ROS	0.244	0.374	0.212	0.162
	SMOTE	0.233	0.382	0.243	0.143
	RUS	0.205	0.364	0.217	-
	ONESS	-	0.324	0.294	0.156

表 2.7 判別モデルとサンプリング法の各組み合わせの判別精度（データセット B）

		LDA	LRA	NN	CT
再現率	未適用	0.590	0.033	0.098	0.393
	ROS	0.420	0.138	0.069	0.325
	SMOTE	0.443	0.220	-	0.220
	RUS	0.482	0.328	0.105	0.377
	ONESS	0.669	0.145	0.118	-
適合率	未適用	0.118	0.200	0.200	0.068
	ROS	0.234	0.299	0.099	0.061
	SMOTE	0.249	0.210	-	0.043
	RUS	0.244	0.246	0.098	0.052
	ONESS	0.129	0.142	0.159	-
F1 値	未適用	0.196	0.056	0.132	0.116
	ROS	0.300	0.187	0.081	0.101
	SMOTE	0.319	0.215	-	0.072
	RUS	0.324	0.280	0.100	0.090
	ONESS	0.217	0.131	0.135	-

判別に寄与するメトリクス，および，サンプリングを行う度合いの最適値が異なることが考えられる．保守開始直後のソフトウェアでは，複雑さメトリクスの大きいモジュールほど欠陥を含む傾向があるが，保守され続けることで，そのようなモジュールから欠陥が除去され，変更の多いモジュールほど欠陥を含む傾向が強まる．また，保守され続けることで，潜在的な欠陥の数が少なくなるため，データセットにサンプリングを行う度合いの最適値が異なる可能性が高い．

6. 関連研究

従来，判別モデルの判別精度の向上を目的として，ソフトウェア工学分野以外のデータセットに対してサンプリング法を適用し，その効果を評価した事例がいくつか報告されている．例えば，Batista ら [4] は判別モデルとして k-最近傍モデルを用い，10 種類のサンプリング法の効果を比較している．実験の結果，データのクリーニングと SMOTE を組み合わせた手法の精度が高くなりがちであることがわかった．本章では，fault-prone モジュールの判別において一般的に用いられる 4 つのモデル（線形判別分析，ロジスティック回帰分析，ニューラルネット，分類木）に対する効果を評価した点が異なる．これら 4 つのモデルに対し，データのクリーニングと SMOTE の組み合わせ手法の効果を評価することは，残された課題の一つである．また，Japkowicz[13] は判別モデルとしてニューラルネットを用い，6 種類のサンプリング法の効果を比較している．Japkowicz の実験では，シミュレーションによりデータセットの大きさ，複雑さ，少数派ケースの含有率を任意に決定した複数のデータセットを用いて，サンプリング法の効果を比較している．実験の結果，複雑であり，かつ，偏りの大きいデータセットを用いた場合に，サンプリング法の効果が大きいことがわかった．この研究では，1 つの説明変数を利用した場合のサンプリング法の効果を評価しているのに対して，本章では，ソースコードから収集可能な複数の説明変数を用いた場合におけるサンプリング法の効果を評価した点が異なる．

また，ソフトウェア工学分野では，サンプリング法の適用事例は報告されていないが，判別モデル自身を拡張することにより，fault モジュール含有率の低い

データセットを用いた場合の判別精度の改善を行うアプローチがいくつか報告されている．例えば，Khoshgoftaar ら [18] はフィットデータセットの fault モジュール含有率に基づき，モジュールに fault が含まれているかどうかの判別基準を変更する分類木を提案している．実験では，fault モジュール含有率約 6%のフィットデータを用いて fault モジュール含有率約 5%のテストデータの判別を行い，従来の分類木よりも再現率が向上したことが示されている．また，Khoshgoftaar ら [17] はデータセットの fault モジュール含有率に基づき，モジュールに fault が含まれているかどうかの判別基準を変更する Analogy-based 法を提案している．実験では，あるシステムのリリース時に収集したデータセット（4 回分のリリース）を用いた．リリース時に収集したデータセットの fault モジュール含有率は，それぞれ 6.3%，4.7%，1.3%，2.3% である．実験の結果，Khoshgoftaar らが提案している分類木より精度が向上したことが示されている．一方，本章では，判別モデル自身を拡張するのではなく，モデル構築の前処理としてサンプリングを適用するため，判別モデルの種類に関わらず実施可能である．ただし，Khoshgoftaar らのようにモデルを拡張するアプローチと，本章のようにサンプリングを行うアプローチのどちらの効果が大きいかを比較することは，残された課題の一つである．

ソフトウェア工学分野において適用事例は報告されていないが，データの偏りを解消するためではなく，多変量の確率分布からサンプルを得るためのマルコフ連鎖モンテカルロ法という手法がある [12]．少数派群の各メトリクスの確率分布を数式化することができれば，マルコフ連鎖モンテカルロ法を用いることで，その分布に基づいたモジュールを得ることができる．しかしながら，各メトリクスの正しい確率分布を求めることは一般的に難しい．一方で，SMOTE などのオーバーサンプリング法は各メトリクスの確率分布を求めずに，手元にあるデータからモジュールを生成することができる．

7. まとめ

本章では，fault-prone モジュール判別モデルの構築において，2 群のケース数の偏りを解消するためにサンプリング法を用いた場合の効果を実験的に評価し

た．実験では，日本のあるソフトウェア開発企業で開発された大規模レガシーソフトウェアを対象とし，4 つのモデル（線形判別分析，ロジスティック回帰分析，ニューラルネット，分類木）に 4 種類のサンプリング法（ROS，SMOTE，RUS，ONESS）を適用した場合の効果を確かめた．

実験により得られた主な結果および知見は，下記の通りである．

- 線形判別分析とロジスティック回帰分析に判別モデルにおいて，4 種類のサンプリング法のいずれを適用した場合にも判別精度が向上した．一方，ニューラルネットと分類木ではサンプリング法の効果はほとんど見られなかった．

- モデルによって最も高い判別精度が得られる fault モジュール含有率に違いがあった．線形判別分析では P_{fp} が $60\% - 67\%$ 付近のときに精度が高く，ロジスティック回帰分析では P_{fp} が $20\% - 33\%$ のときに精度が高かった．

- 線形判別分析とロジスティック回帰分析の結果に着目すると，それぞれのサンプリング法（ROS，SMOTE，RUS，ONESS）を適用した場合，ONESS は他の 3 つのサンプリング法よりも効果が小さいことがわかった（それぞれのモデルの判別精度の平均値は 0.276，0.287，0.293，0.224）．線形判別分析とロジスティック回帰分析の判別モデルに，ONESS 以外のサンプリング法を適用した場合，F1 値の向上は最小 0.078，最大 0.224，平均 0.125 であった．

第3章　ルールベースとモデルベースの組み合わせによるfault-proneモジュール判別方法

1. はじめに

ソフトウェアテストおよび保守工程において，fault-proneモジュールを特定しテスト工数を重点的に割り当てるために，モジュールから計測されたメトリクス（サイクロマティック数，変更行数など）を説明変数とし，モジュールのfaultの有無を目的変数とするfault-proneモジュール判別モデルが多数提案されている[3][5][15][26][30][35]．ロジスティック回帰分析，線形判別分析，分類木などのモデル化手法が提案されており，ロジスティック回帰分析が特に広く用いられている[3][5][26].

ロジスティック回帰分析では，与えられた説明変数の値の組（モジュールから計測されるメトリクス値の組）に対する，ある現象の発生する条件付き確率（モジュールがfaultを含む確率）をロジスティック回帰式としてモデル化する．一般に，faultの混入は確率的に生じるため，線形判別分析のようにfaultの有無を二者択一で判定するよりも，確率事象として捉える方がより妥当であるとの考えから，ロジスティック回帰分析は広く用いられてきた[3][5][26].

一方，非モデルベースの方法として，相関ルール分析を用いたfault-proneモジュール判別方法が近年提案されている[35]．相関ルール分析は，与えられたデータセットから「ある事象Xが発生した場合に高確率で別の事象Y（今回の場合はfaultの含有）が発生する」という法則を相関ルールとして抽出する手法である．例えば，相関ルールは「$(4 \leq \text{cyclomatic number} < 5)$ and $(3 \leq \text{fan-in} < 5) \Rightarrow$

fault」のように表す．このルールでは，モジュールのサイクロマティック数が 4 以上で 5 より小さく，かつ，fan-in が 3 以上で 5 より小さいモジュールの fault の有無を判別することができる．

相関ルール分析の特長の一つは，fault の混入という事象を（一つのモデル式ではなく）多数のルールの集合として捉えるため，多様な fault 混入の要因を抽出し，判別に用いるのに適していることである．もう一つの特長としては，各ルールの重要性を表す指標値（支持度，信頼度など）を利用することで，判別精度の向上に寄与するであろうルールのみを選択することができる（3 章 2 節）．ただし，選定されたルール集合は，あらゆるモジュールを判定できるとは限らない．モジュールによっては条件の一致するルールが存在せず，fault の有無を判別できない場合がある．

本章は，ルールベースとモデルベースの判別手法を組み合わせた fault-prone モジュール判別手法の 1 つとして，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 モジュール判別手法を提案する．ロジスティック回帰分析をモデルベースの手法として用いる理由は，fault-prone モジュール判別における標準的なモデルであると言われている [3] ため，および，一般に fault の混入は確率的に生じることから，線形判別分析や分類木のように fault の有無を二者択一で判定するよりも，確率事象として捉える方がより自然であると考えたためである．提案手法では，与えられたモジュールに対し，重要なルール（支持度，信頼度，または，リフト値の大きなルール）が存在する場合は相関ルール分析によって判別し，そうでない場合は，ロジスティック回帰分析によって判別する．適用可能な複数のルールが存在する場合には，判別結果の多数決を行う．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 fault-prone モジュール判別手法は存在せず，また，いずれの指標値が本手法に最も適しているかも明らかではない．さらに，各指標の閾値をどの程度の大きさに設定し，どの程度ルールの絞り込みを行うべきかも明らかではない．

そこで本章では，いずれの指標値の閾値をどの程度の大きさに設定するべきかを明らかにするために，各指標値の閾値を変化させて，それぞれの判別精度を比較する実験を行う．また，その判別精度を，3 つの代表的な fault-prone 判別モデ

ル（ロジスティック回帰分析，線形判別分析，分類木）[1] と比較対照することで評価する．評価実験では，NASA/WVU が公開しているデータセット（モジュールのメトリクス値と fault の有無が記されたもの）と，Eclipse プロジェクトから収集したデータセットを対象として，交差検証法による評価と，複数バージョン間における評価を行う．

交差検証法は，1 つのデータセットをモデル構築用のフィットデータとモデル評価用のテストデータに分割しモデルの判別精度を評価する方法であり，fault-prone モジュール判別モデルの精度評価に広く行われている [22][33]．本実験では，NASA(アメリカ航空宇宙局) が公開しているデータセットのうち，評価実験に広く用いられている [22][33]KC1 プロジェクトのデータセットを用いた（3 章 4 節）．

一方，複数バージョン間における評価では，fault-prone モジュールを利用する状況により近い条件での実験を行う．実験では，過去のバージョンで報告された fault を記録したフィットデータを用い，次期バージョンに含まれていた fault の予測を試みた．ただし，NASA が公開しているデータセットには fault が報告された時期が示されていない．そこで本章では，Eclipse プロジェクトからモジュールのメトリクス値と fault の報告の履歴を収集し，Gyimothy らの示す方法 [9] によって 2 バージョン分のデータセットを作成し，実験に用いた（3 章 5 節）．

以降，3 章 2 節で相関ルール分析について説明する．3 章 3 節で相関ルール分析とロジスティック回帰分析を組み合わた fault-prone モジュール判別手法を提案する．3 章 4 節では交差検証法を用いた評価実験，3 章 5 節では複数バージョンを用いた評価実験について述べる．3 章 6 節で関連研究について述べ，最後に 3 章 7 節で本章のまとめを述べる．

2. 相関ルール分析

相関ルール分析は，事象間の強い関係をデータセットから相関ルールとして抽出する手法である．Agrawal らは頻出する組合せ（相関ルール）の抽出方法を文献 [1] で次のように定義している．販売履歴を対象とした相関ルール抽出の場

[1] 各モデルの説明は，2 章 2 節を参照のこと

合，販売履歴を D，個々の購買をトランザクション T_i，T_i に含まれる 1 つの商品をアイテム I_k として，与えられた頻度 s よりも大きく D に現れる商品の組合せを $X \Rightarrow Y$ という形式で抽出する．具体的には，$T_i \in D(1 \leq i \leq n)$，$T_i \subset I$，$I = I_1, ..., I_k, ..., I_m(m$ はユニークなアイテムの数) としたときに，s よりも多い数の T_i を満たす $X \Rightarrow Y$ を求める ($X \subset I$, $Y \subset I$，$X \wedge Y = \phi$)．ここで，$X \subset T_i \wedge Y \subset T_i$ のとき，T_i は $X \Rightarrow Y$ を満たす．また，X を前提部と呼び，Y を結論部と呼ぶ.

各ルールの重要性を表す指標値として次がある．

支持度: 支持度は対象データにおける相関ルールの出現頻度であり，$support(X \Rightarrow Y)$ と表記され，$support(X \Rightarrow Y) = s/n$ である．ただし，$s = |\{T \in D | X \subset T \cap Y \subset T\}|, n = |\{T \in D\}|$.

信頼度: 信頼度は前提部 X が満たされたときに同時に結論部 Y も満たされる割合であり，$confidence(X \Rightarrow Y)$ と表記され，$confidence(X \Rightarrow Y) = s/y$ である．ただし，$y = |\{T \in D | X \subset T\}|$.

リフト値: リフト値は前提部 X によりどのくらい結論部 Y が満たされやすくなっているかを示している．リフト値は 0 より大きい値を取り，リフト値が 1.0 より大きければ大きいほど，前提部 X が含まれることで結論部 Y が満たされやすくなることを表し，逆に，リフト値が 1.0 より小さければ小さいほど，前提部 X が含まれることで結論部 Y が満たされにくくなることを表す．また，1.0 のときは X が Y に寄与しないことを表す．リフト値は $lift(X \Rightarrow Y)$, と表記され，$lift(X \Rightarrow Y) = \frac{confidence(X \Rightarrow Y)}{z/n}$ である．ただし，$z = |\{T \in D | Y \subset T\}|$.

例えば，モジュール数 n を 20 とし，X を満たすモジュール数を 10，Y を満たすモジュール数を 8，X と Y を満たすモジュール数を 6 とする．その際，相関ルール $X \Rightarrow Y$ の支持度は 0.3（6/20），信頼度は 0.6（6/10），リフト値は 1.5（$\frac{0.6}{8/20}$）となる．

本章では，各モジュールのソースコードメトリクスを前提部，fault の有無を結論部として用いる．ただし，ソースコードメトリクスは量的変数（間隔尺度や比率尺度）である一方，相関ルールで扱える尺度は質的変数（名義尺度や順序尺

度）であるため，相関ルール分析を適用する前に各メトリクスを量的変数から質的変数に変換する．サイクロマティック数の場合，例えば low $[0, 10)$，medium $[10, 30]$，high $[30, 1/0)$ のように 3 段階の順序尺度に変換する．すべてのメトリクスが順序尺度に変換された後の相関ルールは「medium($10 \leq$ cyclomatic number < 30) and medium($10 \leq$ fan-in < 25) $\Rightarrow$ fault」のように表される．このルールでは，モジュールのサイクロマティック数が 10 以上で 30 より小さく，かつ，fan-in が 10 以上で 25 より小さいモジュールの fault の有無を判別することができる．

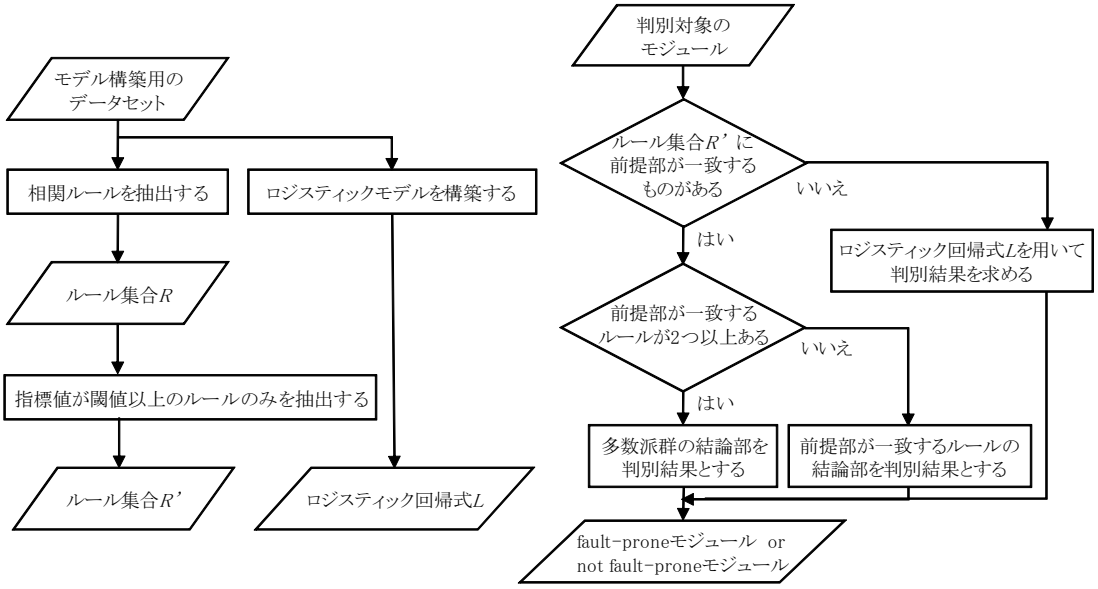
3. 提案手法

3.1 概要

本章は，fault-prone モジュール判別の精度向上を目的として，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 モジュール判別手法を提案する．提案手法において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る上で， 各ルールの重要性を表す指標値（支持度，信頼度，リフト値）を利用する．

本手法では，重要なルール（支持度，信頼度，または，リフト値の大きなルール）が存在する場合はロジスティック回帰分析よりも優先して相関ルール分析によって判別する．存在しない場合は，任意のモジュールを判別できるロジスティック回帰分析によって判別する．適用可能な複数のルールが存在する場合には，判別結果の多数決を行う．

ただし，いずれの指標値（支持度，信頼度，リフト値）が本手法に最も適しているのか，さらに，各指標の閾値をどの程度の大きさに設定し，どの程度ルールの絞り込みを行うべきかは明らかではない．上記の点は評価実験（3 章 4 節，3 章 5 節）により明らかにする．



(a) 提案手法によるモデルの構築　　(b) 提案手法による判別手順

図 3.1　モデルの構築と判別手順

3.2 モデルの構築と判別の手順

提案手法による，モデルの構築からそのモデルによる判別の流れを図 3.1 に示す．まず，図 3.1(a) に示すようにフィットデータを用いて，ロジスティック回帰分析によりロジスティック回帰式 L と，相関ルール分析により相関ルールの集合 R とを抽出する．ここで，相関ルール分析をフィットデータに適用するために，あらかじめ量的変数（間隔尺度や順序尺度）を質的変数（順序尺度）に変換し，均等な k 区間に分割する（本章では，k の値は 5 とした）．例えば，サイクロマティック数の場合，その最小値が 3 で，最大値が 38 であるとすると，[3, 10), [10, 17), [17, 24), [24, 31), [31, 38] のように，5 つに分割される．

次に，相関ルールの集合 R に対して，指標値があらかじめ与えられた閾値以上のルール（支持度が閾値 $\theta_{support}$ より大きいルール，信頼度が閾値 $\theta_{confidence}$ より大きいルール，もしくは，リフト値が閾値 θ_{lift} より大きいルール）のみを選択する（相関ルール集合 R'）．

そして，図 3.1(b) に示すように，判別対象のモジュールのメトリクス値が R' に含まれる相関ルールの前提部に一致するかどうかを判別する．モジュールの判別に相関ルールを用いる場合，前提部が一致するルールの数が 1 つ，もしくは複数個が一致することが考えられる．前提部が一致する相関ルールが 1 つの場合は，その相関ルールの結論部を判別結果とする．複数個ある場合は，多数派群（相関ルール集合 R' に含まれるルールのうち，結論部 Y が多いほうの群）の結論部を判別結果とする．R' の中に前提部が一致する相関ルールが存在しない場合は，任意のモジュールを判別可能なロジスティック回帰式 L により判別する．

4. 交差検証法による実験

4.1 概要

実験の目的は，各指標（支持度，信頼度，リフト値）の閾値を変化させた場合における，提案手法の判別精度を評価することである．さらに，その判別精度を，3 つの代表的な fault-prone 判別モデル（ロジスティック回帰分析，線形判別分析，分類木）と比較対照することで評価する．本実験では，単一のデータセットをランダムに二分割し，一方のデータセット（フィットデータ）を用いて判別モデルを構築し，もう一方のデータセット（テストデータ）を用いてその判別精度を求める（交差検証法）．

支持度，もしくは信頼度を指標値として用いる場合には，各指標の閾値 $\theta_{support}$，もしくは閾値 $\theta_{confidence}$ を 0.1，0.2，0.3,... と 0.1 ずつ変化させた．また，リフト値を指標値として用いる場合には，閾値 θ_{lift} を 1.5 から 2.0，3.0，4.0,... と変化させた．相関ルールの抽出には Morisaki らが開発したプロトタイプツール [25] を用いて，最小支持度は 0.01，相関ルールに含まれる前提部の長さは最大で 4 までとした．

提案手法の比較対象である 3 種類の fault-prone モジュール判別モデル（ロジスティック回帰分析，線形判別分析，分類木）の構築には統計解析用のソフトウェアである R[37] を用いた．

表 3.1 KC1 で収集されたデータセットの概要

fault あり モジュール数（個）	fault なし モジュール数（個）	fault モジュール 含有率（%）
325	1,782	15.4

4.2 データセット

実験には，NASA/WVU IV&V facility Metrics Data Program（MDP） [28] が公開しているデータセットのうち，評価実験によく用いられる [22][33]KC1 のデータセットを用いた．KC1 では，C++で開発されたソフトウェアのモジュールデータが収集されており，ソースコードの規模は 43k ステップである．KC1 で収集されたデータセットの概要を表 3.1 に示す．

実験では，fault の有無を目的変数，21 種類のソースコードメトリクスを説明変数として用いた．ソースコードメトリクスの一覧を表 3.2 に示す．

4.3 実験手順

各指標（支持度，信頼度，リフト値）に対して，以下の手順を行う．

1. データセットをランダムに二等分し，一方をフィットデータ，もう一方をテストデータとする．
2. 手順 (1) で作成したデータセットのペアを用いて，提案手法の判別精度を求める．判別の際には，指標値が閾値以上の相関ルールを用いた．
3. 閾値を変化させて，手順 (2) を繰り返す．
4. 手順 (1) のデータセットのペアを用いて，比較対象である 3 つの判別モデル（ロジスティック回帰分析，線形判別分析，分類木）の判別精度を求める．

表 3.2 KC1 のソースコードメトリクス

番号	ソースコードメトリクス
m_1	空白の行数
m_2	分岐の数
m_3	コメントを含むコード行数
m_4	コメント行数
m_5	Cyclomatic Complexity
m_6	Design Complexity
m_7	Essential Complexity
m_8	コードの実行数（空行とコメント行以外）
m_9	Halstead 尺度の Content
m_{10}	Halstead 尺度の Difficulty
m_{11}	Halstead 尺度の Effort
m_{12}	Halstead 尺度の Error Estimate
m_{13}	Halstead 尺度の Length
m_{14}	Halstead 尺度の Abstract Level
m_{15}	Halstead 尺度のプログラミング時間
m_{16}	Halstead 尺度の Volume
m_{17}	オペランドの数
m_{18}	オペレータの数
m_{19}	ユニークなオペランドの数
m_{20}	ユニークなオペレータの数
m_{21}	コードの総行数

4.4 評価指標

Fault-prone モジュール判別の評価基準として，再現率，適合率，および F1 値 [11] を用いた．再現率，適合率，F1 値それぞれは，値が大きいほど，判別精度が高いことを表す（2 章 4.3 項を参照のこと）．

4.5 結果

各指標の閾値を変化させた場合の提案手法の判別精度（F1 値）と，比較対象のモデルの判別精度を図 3.2 に示す．図 3.2 の (a)，(b)，(c) に，指標値それぞれの結果を示し，グラフの横軸は各指標の閾値，左側の縦軸は F1 値，右の縦軸は相関ルールが適用できたモジュール（前提部の一致する相関ルールが存在したモジュール）の割合を示す．

図 3.2(c) に示すように，提案手法を用いることで，従来手法の中で最も F1 値が大きかった線形判別分析より最大 0.103 向上した．提案手法で用いた指標ごとに以下の傾向が見られた．

- **支持度**　図 3.2(a) に示すとおり，閾値 $\theta_{support}$ の変化に関わらず提案手法の F1 値は，すべての判別手法の中で最も小さい（常に F1 値は 0）．また，提案手法ではすべてのモジュールが相関ルールによって判別されている，つまり，ロジスティック回帰分析によって一切判別されていない．

- **信頼度**　図 3.2(b) に示すとおり，閾値 $\theta_{confidence}$ の変化に関わらず提案手法の F1 値は，支持度を用いた場合と同様，線形判別分析と比較して小さい．閾値 $\theta_{confidence}$ の値が小さい場合，すべてのモジュールが相関ルールによって判別されており，閾値 $\theta_{confidence}$ を 1.0 に設定した場合でも，60%以上のモジュールが相関ルールによって判別されている．

- **リフト値**　図 3.2(c) に示すとおり，閾値 θ_{lift} が 3.0 までは，提案手法の F1 値が大きくなり，3.0 からは F1 値が小さくなった．閾値 θ_{lift} の大きさが 1.5 から 4.0 では，提案手法の F1 値は，今回用いた従来手法すべてと比較して大きい．特に，閾値 θ_{lift} が 3.0 のとき（相関ルールによって判別されるモ

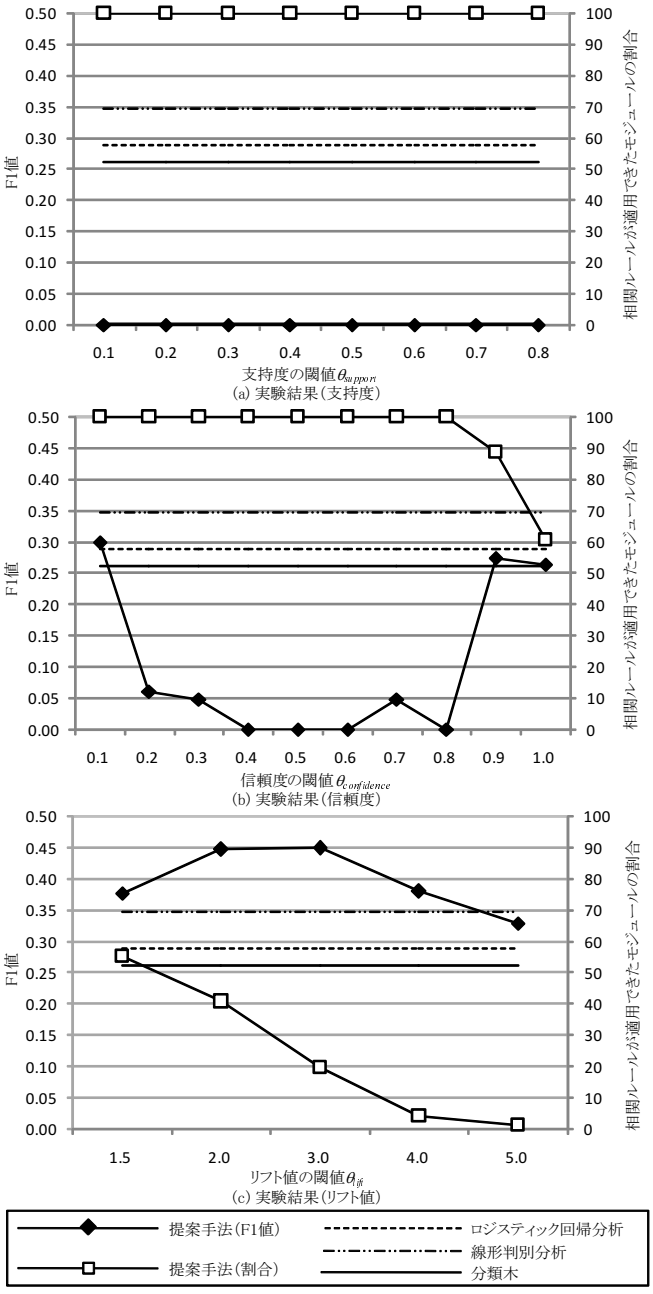


図 3.2 相関ルール抽出の指標値による判別可能なモジュールの割合と判別精度の変化（KC1）

ジュールの割合が約 20%のとき），提案手法の F1 値は 0.449，線形判別分析の F1 値は 0.346，ロジスティック回帰分析の F1 値は 0.288，分類木の F1 値は 0.261 であった．一方，閾値 θ_{lift} が 4.0 以上のとき（相関ルールによって判別されるモジュールの割合が 20%より小さい場合），提案手法の性能が低下している．閾値 θ_{lift} が 5.0 のとき（相関ルールによって判別されるモジュールの割合が 5%以下のとき），提案手法の F1 値は線形判別分析と比較して小さい．

結果として，3 つの指標（支持度，信頼度，リフト値）の中で，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る際に用いるルール選択には，リフト値が最も適していることがわかった．特に，リフト値 3.0 から 4.0 のとき，提案手法の F1 値は最も大きかった．提案手法の性能向上に寄与していたリフト値 3.0 から 4.0 の相関ルールの一部を表 3.3 に示す．表 3.3 の各行は 1 つの相関ルールを示し，「FP」は結論部が fault-prone モジュールであることを示す．今回の実験では，表 3.3 に示す相関ルールのように，複雑さ，及び，規模を表すメトリクスが大きいという条件が前提部に含まれる場合に，（含まれない場合と比べて高い確率で）fault-prone モジュールであることを示す結果が得られた．また，R_1，R_2，R_3 の前提部に Halstead 尺度の Abstract Level（m_4）が小さい（値が小さいほど，ソースコードが複雑であることを表す）という条件が含まれた．この結果は Halstead 尺度の Abstract Level が fault モジュールの判別に最も寄与するメトリクスであるという報告 [29] と一致する．

一方，提案手法の性能向上に寄与しなかった支持度，信頼度が大きい相関ルールの多くは，多数のモジュールに該当する網羅性の高いルールであった．例えば，「$1.00 \leq m_2$(分岐の数) $\leq 3.00 \Rightarrow$ not-fault-prone モジュール（支持度:0.52，信頼度:0.92，リフト値:1.10）」などである．

4.6 考察

支持度，もしくは信頼度を指標として用いた場合，提案手法の判別精度は従来手法よりも低かった．Song ら [35] は，同時に発生しやすい fault の種類を予測す

表 3.3 KC1 から抽出した相関ルールの一部

番号	前提	結論	支持度	信頼度	リフト値
R_1	$(4.00 \leq m_2) \wedge (m_4 < 0.06) \wedge (17.00 \leq m_5) \wedge (13.00 \leq m_6)$	FP	0.05	0.56	3.47
R_2	$(7.00 \leq m_1) \wedge (m_4 < 0.06) \wedge (17.00 \leq m_5) \wedge (13.00 \leq m_6)$	FP	0.05	0.56	3.47
R_3	$(7.00 \leq m_1) \wedge (4.00 \leq m_2) \wedge (24.00 \leq m_3) \wedge (m_4 < 0.06)$	FP	0.05	0.53	3.26

m_1:分岐の数　m_2:サイクロマティック数　m_3:コードの実行行数（空行とコメント行以外）

m_4:Halstead 尺度の Abstract Level　m_5:ユニークなオペランドの数　m_6:ユニークなオペレータの数

るために，支持度を用いてルールの順位付けを行っているが，本章で提案した手法においては判別性能の向上に寄与しなかった．これは，支持度はルールの出現頻度を表す指標であり，前提部が満たされたときに結論部が満たされる確率を表すための指標ではないためであると考えられる．一方，信頼度は，前提部が満たされたときに結論部が満たされる確率を表す指標であるが，fault ありモジュールがフィットデータに含まれる割合が考慮されていないため，信頼度も提案手法の判別性能の向上には寄与しなかったと考えられる．一般的に，モジュールデータは 2 群のケース数に偏りがあることが多い（データセット中の fault ありモジュールと fault なしモジュールの数に大きな差がある）．本データセットの fault ありモジュールの割合も 15.4%であり，2 群のケース数に偏りがあった．そのため，例えば，”xxx ⇒ fault-prone”というルールにとって信頼度 0.7 は重要であるが，”xxx ⇒ not fault-prone”というルールにとっては重要ではない．その結果，信頼度もまた性能向上に寄与しなかったと考えられる．

一方，リフト値を指標とした場合には，図 3.2(c) に示すとおり，提案手法の判別精度は従来手法よりも高かった．また，閾値 θ_{lift} に関わらず，提案手法の判別精度はロジスティック回帰分析よりも高かった．これは，リフト値によって選択

された相関ルールがロジスティック回帰モデルの性能向上に寄与していることを示唆している．ただし，閾値 θ_{lift} を 5.0 に設定した場合のみ，従来手法よりも低くなった．相関ルールによって判別されたモジュールの割合に着目すると，閾値 θ_{lift} を 5.0 に設定した場合，その割合は 1.23%であった．このことから，ほとんどのモジュールが相関ルールによって判別されず，ロジスティック回帰分析によって判別されており，モデルを組み合わせることによる利点が活かされていないことが窺える．

5. 複数バージョン間における実験

5.1 概要

複数バージョン間における実験では，fault-prone モジュールを利用する状況により近い条件での実験を行う．そこで，あるバージョンで発見された fault の履歴を基に，次のバージョンをリリースする際に fault が含まれるであろうモジュールを判別する．本章では Gyimothy ら [9] の方法を用いて Eclipse プロジェクトからモジュールのメトリクス値と fault の履歴を取得した．

本実験において比較する fault-prone モジュール判別手法と評価指標は，3 章 4 節と同様である．また，相関ルール分析の条件も 3 章 4 節と同様である．ただし，相関ルール分析を行ったところ，最もリフト値の大きい相関ルールであってもその値は 3.0 未満であった．そのため，リフト値を指標として用いる場合，閾値 θ_{lift} の大きさを 1.5，2.0，以降，2.5，2.7，2.9 と変化させて，提案手法の判別精度を求めた．

5.2 データセット

対象プロジェクト

本実験で対象としたデータセットは，オープンソースの統合ソフトウェア開発環境の 1 つである Eclipse の開発中に収集された各メトリクス値を記録したもの

である．Eclipse のバージョン 3.0（フィットデータ）と 3.1（テストデータ）を実験対象とした．

Eclipse プロジェクトでは，3 章 4 節で対象とした KC1 プロジェクトのような整形済みのデータセットが公開されていない．本章では，以降に示す方法で，各モジュールのメトリクス値と fault の有無を求めた．ここで，モジュールとは，1 つの Java ファイル（拡張子が.java のもの）とする．

メトリクス値の計測

各バージョンのソースコードを取得し，Eclipse Metrics plugin[7] を用いて，各モジュールのメトリクス値を計測した．実験では，fault の有無を目的変数，Eclipse Metrics plugin で収集可能な 14 種類のメトリクスに，コードの実行行数を加えた計 15 種類のソースコードメトリクスを説明変数として用いた（表 3.4）．

fault の発見／修正履歴の収集

メトリクス値を計測した各バージョンのモジュールが fault を含むかどうかを調べるために，Eclipse プロジェクトが運用している障害管理ツール Bugzilla[2] から，表 3.5 に示す条件に基づいて，fault の報告と修正の履歴を収集した．

Fault の報告と修正がどのモジュールに対して行われたものなのかを特定するために，履歴から，モジュール名を取得した．また，どの時期に fault が含まれていたのかを特定するために，fault が報告された日と，fault が修正された日を取得した．

モジュールと fault の有無の関連付け

本章では，Gyimothy ら [9] が行った方法で，各バージョンの各モジュールに fault が含まれているかどうかを求めた．

まず，どのモジュールに fault が含まれているかを特定した．Bugzilla において，fault の報告と修正の履歴には，fault が修正されたソースコード（修正パッチ）が

[2] `https://bugs.eclipse.org/bugs/`

表 3.4 Eclipse のソースコードメトリクス

番号	ソースコードメトリクス
m_1	コードの実行数
m_2	メソッドの総行数
m_3	最大ネスト数
m_4	メソッドのパラメータ数
m_5	Cyclomatic Complexity
m_6	フィールド数
m_7	メソッド数
m_8	オーバーライドしたメソッド数
m_9	サブクラスの数
m_{10}	静的フィールドの数
m_{11}	静的メソッドの数
m_{12}	SIX （Specialization Index）
m_{13}	DIT (Depth of Inheritance Tree)
m_{14}	LCOM (Lack of Cohesion of Methods)
m_{15}	WMC (Weighted Methods per Class)

表 3.5 Fault の発見／修正履歴の収集条件

Classification	Eclipse
Product	Platform
Status of faults	Resolved, Verified, Closed
Resolution of faults	Fixed
Severity	Enhancement 以外
Priority	all

表 3.6 Eclipse で収集されたデータセットの概要

バージョン	fault あり モジュール数（個）	fault なし モジュール数（個）	fault モジュール 含有率（%）
Eclipse 3.0	793	3,659	17.8
Eclipse 3.1	859	4,415	16.3

添付されており，そこにモジュール名が記述されている．そこで，修正パッチに書かれているモジュール名を取得し，どのモジュールに fault が含まれていたのかを特定した．

次に，fault が含まれている時期（バージョン）を特定した．fault が報告された日からその fault が修正されるまでの期間に修正されたモジュールには，fault が含まれているものとみなした．図 3.3 に示すように，例えば，バージョン 3.0 から 3.1 の間に fault が報告され，3.2 から 3.3 の間に修正されたモジュールの場合，バージョン 3.0 から 3.2 の間までの期間，fault を含んでいるとみなす．

上記の手順で Eclipse プロジェクトから収集されたデータセットの概要を表 3.6 に示す．表 3.6 に示すとおり，バージョン 3.0 には 4,452 個のモジュールが，バージョン 3.1 には 5,274 個のモジュールが含まれていた．それぞれのデータセットに fault ありモジュールが含まれていた割合は，17.8%と 16.3%であった．

5.3 実験手順

各指標（支持度，信頼度，リフト値）に対して，以下の手順を行う．

1. Eclipse のバージョン 3.0 と 3.1 から収集されたデータセットを用いて，提案手法の判別精度を求める．判別の際には，ある指標が閾値以上の相関ルールを用いた．

2. 閾値を変化させて，手順 (1) を繰り返す．

3. 手順 (1) と同様のデータセットを用いて，比較対象である 3 つの判別モデル（ロジスティック回帰分析，線形判別分析，分類木）の判別精度を求める．

5.4 結果

各指標の閾値を変化させた場合の提案手法の判別精度（F1 値）の変化と，比較対象のモデルの判別精度を図 3.4 に示す．図 3.4(c) に示すように，提案手法を用いることで，従来手法の中で最も F1 値が大きかった線形判別分析より F1 値が最大 0.163 向上した．提案手法で用いた指標ごとに以下の特徴が見られた．

- **支持度**　図 3.4(a) に示すとおり，3 章 4 節と同様の結果で，閾値 $\theta_{support}$ の変化に関わらず提案手法の F1 値は従来手法の中で最も F1 値が大きかった線形判別分析と比較して小さい（常に F1 値は 0）．また，提案手法では，すべてのモジュールが相関ルールによって判別されている．

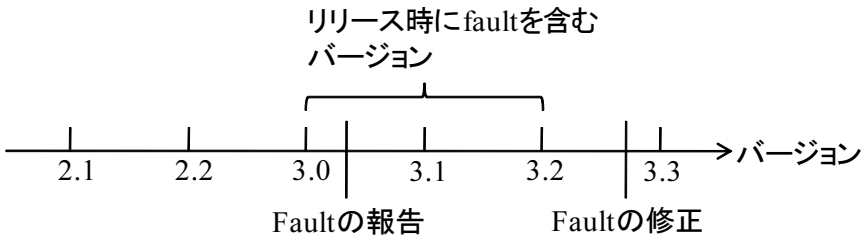


図 3.3　fault の修正／報告とモジュールが関連する Eclipse のバージョン

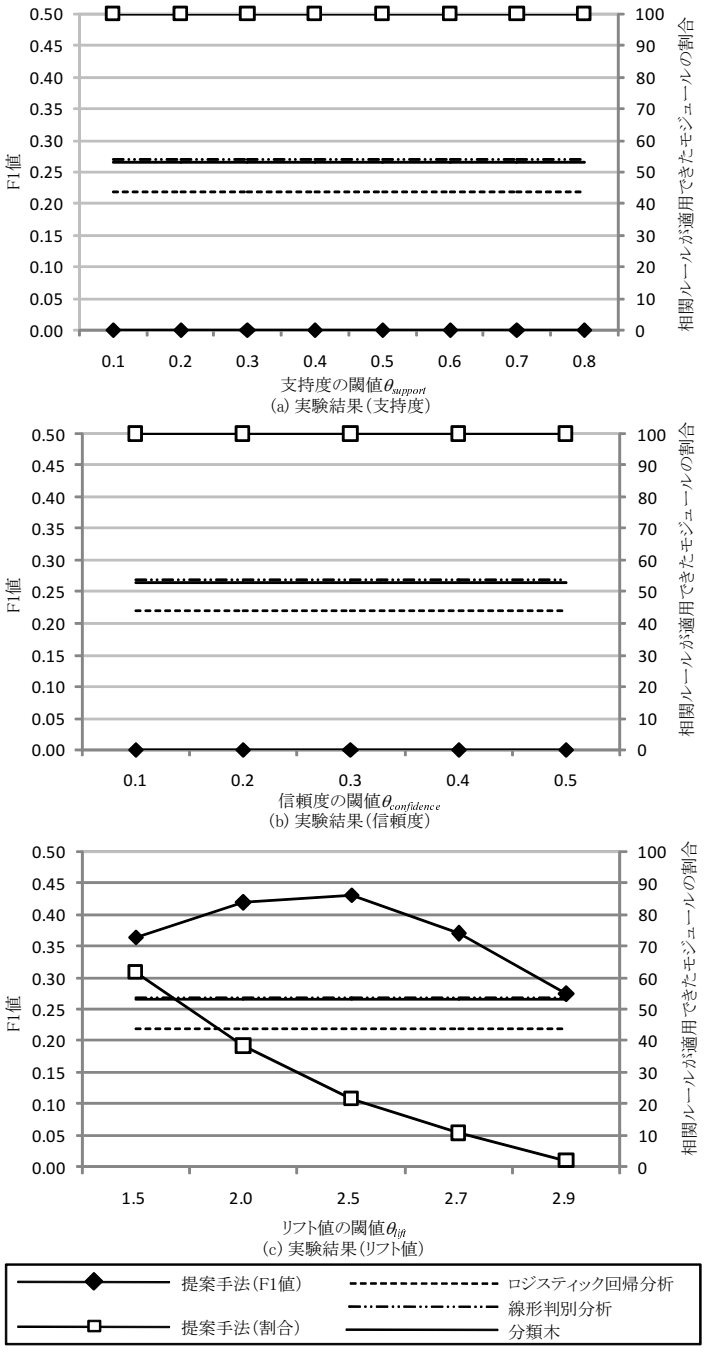


図 3.4 相関ルール抽出の指標値による判別可能モジュール割合と判別精度の変化（Eclipse）

- **信頼度**　図 3.4(b) に示すとおり，3 章 4 節の結果と異なり，閾値 $\theta_{confidence}$ の変化に関わらず提案手法の F1 値は線形判別分析と比較して小さい（常に F1 値は 0）．また，提案手法では，すべてのモジュールが相関ルールによって判別されている．

- **リフト値**　図 3.4(c) に示すとおり，3 章 4 節と同様の結果で，閾値 θ_{lift} の変化に関わらず提案手法の F1 値は，3 つの従来手法と比較して大きい（閾値 θ_{lift} が 2.5 のとき，提案手法の F1 値は 0.432，従来手法の中で最も精度がよかった線形判別分析の F1 値は 0.269 であった）．

  閾値 θ_{lift} が 2.5 までは，提案手法の F1 値が大きくなり，2.5 からは F1 値が小さくなった．相関ルールによって判別されるモジュールの割合が 20% より小さい場合，提案手法の性能が低下している．

提案手法の性能向上に寄与していたリフト値 2.5 から 2.7 の相関ルールの一部を表 3.7 に示す．3 章 4 節の実験で得られた結果と同様，複雑さ，及び，規模を表すメトリクスが大きい場合に，(そうでない場合と比べて高い確率で）fault-prone モジュールであることを示す結果であった．また，支持度，信頼度が大きい相関ルールの多くも，前節の実験で得られた結果と同様，多数のモジュールに該当する網羅性の高いルールであった．例えば，「m_{10}(静的フィールドの数) $\leq 2.00 \Rightarrow$ not-fault-prone モジュール（支持度:0.65，信頼度:0.86，リフト値:1.04)」などである．

5.5 考察

3 章 4 節と同様に，支持度，もしくは信頼度を指標として用いた場合，提案手法の判別精度は従来手法よりも低かった．閾値の大きさに関わらず，提案手法の F1 値は常に 0 であった．

一方，リフト値を指標とした場合には，提案手法の判別精度は従来手法よりも高かった．ただし，閾値 θ_{lift} を 2.9 に設定した場合のみ，従来手法とほぼ同じ F1 値の大きさとなった．相関ルールによって判別されたモジュールの割合に着目す

表 3.7 Eclipse から抽出した相関ルールの一部

番号	前提	結論	支持度	信頼度	リフト値
R_1	$(40.00 \leq m_5) \wedge (3.00 \leq m_{13}) \wedge (5.00 \leq m_6) \wedge (1.00 \leq m_8)$	FP	0.01	0.48	2.69
R_2	$(40.00 \leq m_5) \wedge (5.00 \leq m_6) \wedge (2.00 \leq m_{10}) \wedge (181.00 \leq m_1)$	FP	0.03	0.46	2.60
R_3	$(4.00 \leq m_3) \wedge (5.00 \leq m_6) \wedge (2.00 \leq m_{10}) \wedge (181.00 \leq m_1)$	FP	0.02	0.46	2.59

m_1:コードの実行行数　m_3:最大ネスト数　m_5:サイクロマティック数　m_6:フィールド数

m_8:オーバーライドしたメソッド数　m_{10}:静的フィールドの数　m_{13}:DIT (Depth of Inheritance Tree)

ると，閾値 θ_{lift} を 2.9 に設定した場合，その割合は 2.03%であった．このことから，ほとんどのモジュールが相関ルールによって判別されず，ロジスティック回帰分析によって判別されており，組み合わせの恩恵を受けられていないことが窺える．現時点では，3 章 4 節と本節の結果からは，相関ルールによって判別されるモジュールの割合が 20%程度の高さを保つように閾値 θ_{lift} の大きさを設定することで，組み合わせの恩恵を受けられると示唆される．また，リフト値 2.0 を閾値としてルールを採用した場合は 3 章 4 節，5 節ともに従来手法より F1 値が大きいことから，リフト値 2.0 以上のルールを採用することが fault-prone モジュール判別におけるひとつの目安となると考える．

ただし，3 章 4 節と本節の結果は，2 章と同様，2 つのデータセットから得られたものであり，結果の信頼性を向上させるためには他のデータセットを用いて実験を追加する必要がある．また，3 章 4 節と本節では，判別モデルにおいて一般的に利用されているメトリクス（3 章 4 節：主に Cyclomatic 複雑度や Halstead 複雑度，本節：主に CK メトリクス）を用いているので，(特殊なメトリクスを用いるよりは）一般性のある結果であると考えるが，一般性をより向上させるためには今回用いた以外のメトリクスによる実験を追加することが望ましい．

6. 関連研究

ソフトウェアモジュールから計測したソースコードメトリクス（サイクロマティック数，変更行数など）に基づいて，モジュールの fault の有無を推定する fault-prone モジュール判別モデルが多数提案されており [3][5][15][26][30][35]，ロジスティック回帰分析，線形判別分析，分類木などのモデル化手法が用いられる．例えば，ロジスティック回帰分析は，与えられた説明変数の値の組に対する，ある現象の発生する条件付き確率（モジュールが fault を含む確率）をロジスティック回帰式としてモデル化する手法である．一般に，fault の混入は確率的に生じるため，確率事象として捉えることが妥当であるとの考えから，ロジスティック回帰分析は広く用いられてきた [3][5][26]．

一方で，従来から広く用いられているソースコードメトリクスではなく，ネットワークメトリクス（モジュール間の関係を表すメトリクス）を判別モデルのパラメータとして利用することが Zimmermann らによって提案されている [40]．重回帰分析，ロジスティック回帰分析を判別モデルとして，Windows Server 2003 を対象とした実験の結果，ネットワークメトリクスをソースコードメトリクスと併用して用いることはソースコードメトリクスを単体で用いるよりも，fault-prone モジュール判別に効果があることを示した．ネットワークメトリクスを本章で提案する手法にて用いることで，更なる精度向上が期待できる．

また，ソースコードからメトリクスを抽出することなく，ソースコードのテキスト自身を入力として，fault-prone モジュールを判別する手法が Mizuno らによって提案されている [24]．Mizuno らは，メールのスパムフィルタなどに利用されているベイジアンフィルタを用いることで，ソースコードを単なるテキストと見なして fault-prone モジュールの判別を試みている．オープンソースプロジェクトから収集したデータセットを対象とした実験では，fault を含むモジュールの全体のうち 78%のモジュールを検出することができた．本章で提案する相関ルール分析との組み合わせを行った場合，ソースコードメトリクスを抽出する必要がないという Mizuno らの手法の利点が失われることにはなるが，判別精度が向上する可能性がある．

ソフトウェア工学以外の分野では，相関ルール分析は，小売販売店での販売履

歴を対象とした併せ買いの傾向の抽出 [1]，ウェブサイトのアクセスログを対象とした個々のウェブページのキャッシング，先読みの決定 [39]，外膜タンパク質の予測 [34] をはじめとして，様々な分野でその効果が認められている．例えば，販売履歴からの併せ買いの分析で「商品 A を購入する ∧ 商品 B を購入する⇒ 商品 C を購入する」というルールが得られた場合，商品 A，B，C を近くに配置し，顧客がそれらの商品の存在を認識しやすくするなどの利用方法が考えられる．

ソフトウェア工学分野における利用例も少ないながらいくつか報告されている．Song ら [35] は，開発中の fault の報告履歴（fault 混入の原因，修正工数など）から相関ルールを抽出し，同時に起こりやすい fault の種類の発見や，fault の修正工数（”1 人時未満”，”1 人時以上，1 人日未満”，”1 人日以上，3 人日未満”，”3 人日以上”）の予測に用いた．浜野ら [10] は，ソフトウェア開発プロジェクトを混乱（開発予算，もしくは開発期間が基準値を超えている状態）に陥れる要因を明らかにするために，開発プロジェクトのリスク要因に関するメトリクス値を集め，相関ルール分析を行った．Michail[23] は，アプリケーション内でのクラスライブラリの再利用パターンを相関ルール分析を用いて発見し，そのパターンをライブラリ構築時に利用することを試みた．これらの研究は相関ルール分析を単独で利用している一方，本章では fault-prone モジュールの判別精度向上のために相関ルール分析をモデルベースの判別手法と組み合わせている．また，これらの研究はルールの重要性を表す指標値として支持度，信頼度のみを用いている一方，本章ではそれらに加えてリフト値も用いて，その効果を確認している．

7. まとめ

本章では，fault-prone モジュール判別の精度向上を目的として，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 モジュール判別手法を提案し，その性能を実験的に評価した．実験により得られた主な結果および知見は以下の通りである．

- 提案手法を用いることで，3 つの従来手法（ロジスティック回帰分析，線形判別分析，分類木）と比較して F1 値が最大 0.163 向上した．

- 今回用いたデーセットでは，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る際に用いる相関ルールの選択には，リフト値が適していることがわかった．また，モデル利用者は，判別されるモジュールの割合が 20%程度の高さを保つように閾値 θ_{lift} の大きさを設定することで，組み合わせの恩恵を受けられると示唆された．一方，支持度と信頼度は適していないことがわかった．

- リフト値を提案手法に用いた場合には，サイクロマティック数や分岐の数などの複雑さを表すメトリクス値が一定の値を超えると，fault を含む確率が高いことを示す相関ルールが含まれていた．

なお，参考までに線形判別分析や分類木を相関ルールの組み合わせに用いた場合の結果を次に示す．3 章 4 節，5 節の実験のうち，本実験で最も F1 値が大きくなったリフト値を閾値として用いた（$\theta_{lift} = 3.0$，$\theta_{lift} = 2.5$）．線形判別分析，分類木それぞれと組み合わせた場合の F1 値は，3 章 4 節のデータセットを用いた場合では 0.454，0.441 であり，3 章 5 節のデータセットでは 0.433，0.432 であった．いずれの場合においても，相関ルールとの組み合わせによって精度の向上が得られた．

第4章　おわりに

本論文では，モジュールのメトリクス値から fault の有無を判別する fault-prone モジュール判別モデルの性能を向上することを目的とし，(1) フィットデータの偏りを解消するサンプリング法の適用，および (2) モデル自体の表現能力の限界を補うためのルールベース判別との組み合わせ方法の提案を行った．

まず，2 群のケース数の偏りを解消するためにサンプリング法を用いた場合の効果を実験的に評価した．実験では，ソフトウェア開発企業で開発された大規模レガシーソフトウェアを題材とし，4 つの fault-prone モジュール判別モデル（線形判別分析，ロジスティック回帰分析，ニューラルネット，分類木）に対して，4 種類のサンプリング法（ROS，SMOTE，RUS，ONESS）を適用した．実験の結果，線形判別分析とロジスティック回帰分析に対しては効果がみられたが，ニューラルネットと分類木に対しては効果がみられなかった．また，ONESS は，他の 3 つのサンプリング法と比べると，効果が小さいことがわかった．線形判別分析とロジスティック回帰分析に対して ONESS 以外のサンプリング法を適用した場合の結果に着目すると，F1 値の向上は最小 0.078，最大 0.224，平均 0.125 であった．

次に，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 モジュール判別手法を提案し，その性能を実験的に評価した．実験では，NASA/WVU の公開しているデータセットと，Eclipse プロジェクトから収集したデータセットを用いて，提案手法の判別精度を従来手法（線形判別分析，ロジスティック回帰分析，分類木）と比較対照した．その結果，重要とみなすルールの選択には，リフト値が適していることがわかり，一方，支持度と信頼度は適していないことがわかった．具体的には，リフト値に閾値を設けてルールを選定することで，提案手法の F1 値は従来手法と比較して 0.163 向上した．また，モデル利用者は，判別されるモジュールの割合が 20%程度の高さを保つように閾値 θ_{lift} の大きさを設

定することで，組み合わせの恩恵を受けられることが示唆された．

これら本論文の成果は，多くのテスト現場において，モデル構築に適したフィットデータの準備を容易するともに，モデルの判別精度（fault の有無）を向上することが見込まれる．これによって，テスト計画者は fault を含むモジュールをより正確に特定し，それらに重点的にテスト工数を割り当てことが可能となり，より効果的なテストを実施できると考える．

謝辞

本研究を進めるにあたり，多くの方に御指導，御協力を頂きました．ここに感謝の意を表させて頂きたいと思います．本当にありがとうございました．

奈良先端科学技術大学院大学 情報科学研究科 松本 健一 教授には，研究に対する直接的な指導に限らず，研究に対する姿勢から，原稿の書き方に至るまで，あらゆる面で熱心な御指導を賜りました．先生が整えてくださった研究環境は非常に充実したものであり，おかげで研究活動に専念することができました．先生から機会を頂いた NICTA での研究生活で得た知識と人脈は，今でも私の大きな財産です．心より厚く御礼申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 関 浩之 教授には，本研究の課題や方針などで的確な御助言，御指摘を賜りました．先生の御指導により，本論文がより洗練されたものとなりました．本研究に対する直接的な御指導のみならず，他の研究発表に対する常に適切かつ建設的な質問やコメントは，大いに参考にさせていただきました．心より厚く御礼申し上げます．

大阪大学大学院 情報科学研究科 楠本 真二 教授には，本論文に関して適切な御指導，御助言を賜りました．また，国内外の会議やワークショップでお会いした際には，非常に有意義な時間を共有させていただきました．心より厚く御礼申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 門田 暁人 准教授には，私の研究生活の全過程において熱心な御支援を賜りました．研究の進め方から，論文執筆，プレゼンテーション技術，全てに渡り，熱意ある御指導を賜りました．特に，原稿投稿の締め切り前には，昼夜を問わず付きっ切りで御指導を賜りました．私の研究は，先生の御指導がなければ成しえませんでした．また，先生の温かいお人柄により，学生生活も豊かなものとなりました．心より厚く御礼申し上げます．

Kasetsart 大学 Pattara Leelaprute 博士には，本研究を進めるにあたり，貴重な御助言，御指摘を賜りました．研究活動を離れても楽しい時間を共有させていただきました．心より厚く御礼申し上げます．

奈良先端科学技術大学院大学 情報科学研究科 飯田 元 教授には，本研究に対する直接的な御指導のみならず，研究者としての考え方も御指導を賜りました．心

より厚く御礼申し上げます.

奈良先端科学技術大学院大学 情報科学研究科 大平 雅雄 助教には，研究者としてあるべき姿について数多くの御指導，御助言を賜りました．研究の楽しさ，厳しさ，研究に対する姿勢など数多くのことを学ぶことができました．研究以外でも大変親しく接していただき，私の学生生活はより充実したものとなりました．心より厚く御礼申し上げます.

奈良先端科学技術大学院大学 情報科学研究科　森崎 修司 助教には，ソフトウェアレビューの研究を通じて，物事を進める上での段取りの重要さや，締め切りを守ることへの心構えなど，数多くの御指導を賜りました．また，学生生活において，しばしばお気遣いの言葉を賜り，大変元気付けられました．心より厚く御礼申し上げます.

奈良先端科学技術大学院大学 情報科学研究科 川口 真司 助教，名倉 正剛 特任助教（現 株式会社 日立製作所）には，研究活動や学生生活に関する貴重な御助言を賜りました．心より厚く御礼申し上げます.

奈良先端科学技術大学院大学 情報科学研究科 角田 雅照 特任助教には，本研究の核となる事柄について多くの御指摘，御助言を賜りました．また，統計学全般についても丁寧かつ熱心な御指導を賜りました．心より深く感謝いたします.

奈良先端科学技術大学院大学 情報科学研究科 松村 知子 特任助教には，研究生活に関する御助言を賜りました．深く感謝いたします.

New South Wales 大学 Ross Jeffery 教授には，御多忙の中, 先生の御好意により，NICTA での研究活動の受け入れを快諾していただきました．NICTA での研究活動は非常に刺激的であり，新たな発見に満ち溢れた日々でした．心より厚く御礼申し上げます.

関西大学 総合情報学部 田中研究室の皆様，特に田中 成典 教授には，本の執筆活動を通して，文章を書く楽しさ，難しさについて御指導を賜りました．また，どんな困難な課題にもめげずに立ち向かっていくための精神を叩き込んでいただきました．心より厚く御礼申し上げます.

神戸大学大学院 工学研究科 中村 匡秀 准教授には，研究論文の書き方や研究会などでの発表の作法など，研究者に必要な素養の習得について御指導を賜りま

した．また，学生生活において数多くの御助言を賜りました．心より厚く御礼申し上げます．

神戸大学大学院 工学研究科 井垣 宏 特命助教，京都産業大学 コンピュータ理工学部 玉田 春昭 助教，ソニー株式会社 岡本 圭司 氏には，研究生活においても研究以外の場においても様々な相談に乗っていただき，多くのアドバイスをいただきました．心より厚く御礼申し上げます．

株式会社 NTT データ 大杉 直樹 氏，大阪大学大学院 情報科学研究科 柿元 健 特任助教には，本研究を進めるにあたり，研究発表や論文執筆の面で御指導を賜りました．心より厚く御礼申し上げます．

NICTA (National ICT Australia) Jacky Keung 氏には，NICTA での研究生活において，つたない英語であっても真剣に議論に付き合ってくださいました．生活面においても，自身の部屋の一室を貸してくださるなど，多大な御支援を賜りました．心より厚く御礼申し上げます．

株式会社富士通研究所 小林 健一 氏には，NICTA でお会いした折に，本研究に関して様々な議論をしていただきました．また，氏が自身の研究成果を現場にどのように広めているかについて伺うことができたのは，貴重な経験となりました．心より厚く御礼申し上げます．

株式会社 NTT データ MSE 松井 恭 氏，佐々木 健介 氏，赤松 敬 氏には，ソフトウェアレビューについて，実務の立場から示唆に富む数多くの御意見を賜りました．心より厚く御礼申し上げます．

三菱電機マイコン機器ソフトウエア株式会社 佐方 陽一 氏，富本 達明 氏には，共同研究を通じて，組み込みソフトウェア開発の実情を学ぶことができました．心より厚く御礼申し上げます．

日本電気株式会社 石井 健一 氏，パナソニック株式会社 江川 学 氏には，大学院進学にあたり，親身になって相談に乗っていただきました．深く感謝いたします．

坂岡学習塾 坂岡 士朗 先生には，学ぶ楽しさを教えていただきました．中学生の頃に先生との出会いがなければ，勉強に取り組むこともなく，大学院に進学し研究を行うことはなかったと思います．深く感謝いたします．

奈良先端科学技術大学院大学 ソフトウェア工学講座 乾 純子 氏には，研究の遂行に必要な事務処理など多岐に渡り手助けいただきました．深く感謝いたします．

奈良先端科学技術大学院大学 ソフトウェア工学講座 山内 寛己 氏には，本研究を進めるにあたり，研究環境の整備の面で御助力を賜りました．深く感謝いたします．

奈良先端科学技術大学院大学 ソフトウェア工学講座 上野 秀剛 氏（現 奈良工業高等専門学校 助教），戸田 航史 氏には，研究生活に関する御助言を賜りました．大学院入学以来の先輩として，多くの時間を共有していただき励ましてくださいました．深く感謝いたします．

奈良先端科学技術大学院大学 ソフトウェア工学講座 柗本 真佑 氏には，本研究を進めるにあたり，実験方法，論文執筆，システムの実装の面において大変な御助力を賜りました．氏の何事に対しても妥協を許さず，真剣に取り組む姿勢には，ただ尊敬の念を抱くばかりです．常日頃より曖昧なアイデアであっても議論に付き合ってくださり，研究を行う上で非常に心強い存在でした．心より深く感謝いたします．

奈良先端科学技術大学院大学 ソフトウェア工学講座 伊原 彰紀 氏，左藤 裕紀 氏，田村 晃一 氏には，論文の体裁チェックや研究発表の練習に数多く付き合っていただきました．深く感謝いたします．

奈良先端科学技術大学院大学 ソフトウェア工学講座ならびにソフトウェア設計学講座の皆様には，研究や日々の生活で様々な御支援を賜りました．おかげさまで非常に充実した研究生活を過ごさせて頂きました．特に，同期生である，瀧 進也 氏（現 株式会社 日立製作所），田中 章弘 氏（現 株式会社 日立製作所），三井 康平 氏（現 株式会社インターネットイニシアティブ），大蔵 君治 氏（現 同大学院 博士後期課程），後藤 慶多 氏（現 日本電子計算株式会社），伏田 享平 氏（現 同大学院 博士後期課程）には，常日頃より楽しい時間を共有させていただきました．心より厚く御礼申し上げます．

最後に，日頃より私を励まし，支えてくれた父と母と妹に心より感謝します．

参考文献

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. of 1993 ACM SIGMOD Int'l Conf. on Management of Data*, pp. 207–216, 1993.

[2] 麻生英樹. ニューラルネットワーク情報処理–コネクショニズム入門、あるいは柔らかな記号に向けて. 産業図書, 1988.

[3] Victor R. Basili, Lionel C. Briand, and Walcelio L. Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Trans. Software Engineering*, Vol. 22, No. 10, pp. 751–761, 1996.

[4] Gustavo E. Batista, Ronaldo C. Prati, and Maria C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations*, Vol. 6, No. 1, pp. 20–29, 2004.

[5] Lionel C. Briand, Victor R. Basili, and Christopher J. Hetmanski. Developing interpretable models with optimized set reduction for identifying high-risk software components. *IEEE Trans. Software Engineering*, Vol. 19, No. 11, pp. 1028–1044, 1993.

[6] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, Vol. 16, pp. 321–357, 2002.

[7] Frank Sauer. Eclipse Metrics plugin. `http://sourceforge.net/projects/metrics`.

[8] Andrew R. Gray and Stephen G. MacDonell. Software metrics data analysis – exploring the relative performance of some commonly used modeling techniques. *Empirical Software Engineering*, Vol. 4, No. 4, pp. 297–316, 1999.

[9] Tibor Gyimothy, Rudolf Ferenc, and Istvan Siket. Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Trans. Software Engineering*, Vol. 31, No. 10, pp. 897–910, 2005.

[10] 浜野康裕, 天嵜聡介, 水野修, 菊野亨. 相関ルールマイニングによるソフトウェア開発プロジェクト中のリスク要因の分析. コンピュータソフトウェア, Vol. 24, No. 2, pp. 79–87, 2007.

[11] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Information Systems*, Vol. 22, No. 1, pp. 5–53, 2004.

[12] 伊庭幸人, 種村正美, 大森裕浩, 和合肇, 佐藤整尚, 高橋明彦. 計算統計 II – マルコフ連鎖モンテカルロ法とその周辺. 岩波書店, 2005.

[13] Nathalie Japkowicz. The class imbalance problem: Singnificance and strategies. In *Proc. Int'l Conf. on Artificial Intelligence (ICAI'00)*, pp. 111–117, 2000.

[14] Capers Jones. *Applied Software Measurement, Second Edition.* McGraw-Hill, New York, 1996.

[15] Taghi M. Khoshgoftaar and Edward B. Allen. Modeling software quality with classification trees. In *Recent Advances in Reliability and Quality Engineering*, pp. 247–270, Singapore, 1999. World Scientific.

[16] Taghi M. Khoshgoftaar, Kehan Gao, and Robert M. Szabo. An application of zero-inflated poisson regression for software fault prediction. In *Proc. 12th Int'l Symposium on Software Reliability Engineering (ISSRE'01)*, pp. 66–73, Hong Kong, China, 2001.

[17] Taghi M. Khoshgoftaar and Naeem Seliya. Analogy-based practical classification rules for software quality estimation. *Empirical Software Engineering*, Vol. 8, No. 4, pp. 325–350, 2003.

[18] Taghi M. Khoshgoftaar, Xiaojing Yuan, and Edward B. Allen. Balancing misclassification rates in classification-tree models of software quality. *Empirical Software Engineering*, Vol. 5, No. 4, pp. 313–330, 2000.

[19] Miroslav Kubat and Stan Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *Proc. 14th Int'l Conf. on Machine Learning (ICML'97)*, pp. 179–186, Nashville, USA, 1997.

[20] 栗田多喜夫. 情報量基準による 3 層ニューラルネットの隠れ層のユニット数の決定法. 電子情報通信学会論文誌, Vol. J73-D-II, No. 11, pp. 1872–1878, 1990.

[21] Paul L. Li, James Herbsleb, Mary Shaw, and Brian Robinson. Experiences and results from initiating field defect prediction and product test prioritization efforts at ABB inc. In *Proc. 28th Int'l Conf. on Software Engineering (ICSE'06)*, pp. 413–422, 2006.

[22] Shinsuke Matsumoto, Yasutaka Kamei, Akito Monden, and K Matsumoto. Comparison of outlier detection methods in fault-proneness models. In *Proc. 1st Int'l Symposium on Empirical Software Engineering and Measurement (ESEM'07)*, pp. 461–463, Sep 2007.

[23] Amir Michail. Data mining library reuse patterns using generalized association rules. In *Proc. 22nd Int'l Conf. on Software Engineering (ICSE'00)*, pp. 167–176, 2000.

[24] Osamu Mizuno, Shiro Ikami, Shuya Nakaichi, and Tohru Kikuno. Fault-prone filtering: Detection of fault-prone modules using spam filtering technique. In *Proc. 1st Int'l Symposium on Empirical Software Engineering and Measurement (ESEM'07)*, pp. 374–383, Sep 2007.

[25] Shuji Morisaki, Akito Monden, Haruaki Tamada, Tomoko Matsumura, and Ken-ichi Matsumoto. Mining quantitative rules in a software project data set. *IPSJ Journal*, Vol. 48, No. 8, pp. 2725–2734, 2007.

[26] John C. Munson and Taghi M. Khoshgoftaar. The detection of fault-prone programs. *IEEE Trans. Software Engineering*, Vol. 18, No. 5, pp. 423–433, 1992.

[27] Glenford J. Myers. *The Art of Software Testing.* John Wiley and Sons, New York, 1st edition, 1979.

[28] NASA/WVU IV&V Facility. Metrics Data Program. `http://mdp.ivv.nasa.gov/`.

[29] NASA/WVU IV&V Facility. Metrics Data Program -Halstead Metrics-. `http://mdp.ivv.nasa.gov/halstead_metrics.html#L`.

[30] Niclas Ohlsson and Hans Alberg. Predicting fault-prone software modules in telephone switches. *IEEE Trans. Software Engineering*, Vol. 22, No. 12, pp. 886–894, 1996.

[31] Maurizio Pighin and Roberto Zamolo. A predictive metric based on discriminant statistical analysis. In *Proc. 19th Int'l Conf. on Software Engineering (ICSE'97)*, pp. 262–270, 1997.

[32] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, Vol. 323, pp. 533–536, 1986.

[33] Naeem Seliya and Taghi M. Khoshgoftaar. Software quality estimation with limited fault data: a semi-supervised learning perspective. *Software Quality Journal*, Vol. 15, No. 3, pp. 327–344, 2007.

[34] Rong She, Fei Chen, Ke Wang, Martin Ester, Jennifer L. Gardy, and Fiona S. L. Brinkman. Frequent-subsequence-based prediction of outer membrane proteins. In *Proc. of ACM SIGKDD Int'l Conf. of Knowledge Discovery and Data Mining*, pp. 436–445, 2003.

[35] Qinbao Song, Martin Shepperd, Michelle Cartwright, and Carolyn Mair. Software defect association mining and defect correction effort prediction. *IEEE Trans. Software Engineering*, Vol. 32, No. 2, pp. 69–82, 2006.

[36] 鈴木英之進. 正確な学習よりも得する学習–学習コストを考慮する分類学習–(1) 評価偏. 情報処理, Vol. 45, No. 4, pp. 395–401, 2004.

[37] The R Foundation. R. `http://www.r-project.org/`.

[38] Ivan Tomek. Two modifications of cnn. *IEEE Trans. Systems, Man and Cybernetics*, Vol. SMC-6, pp. 769–772, 1976.

[39] Qiang Yang, Haining H. Zhangand, and Tianyi Li. Mining web logs for prediction models in www caching and prefetching. In *Proc. of ACM SIGKDD Int'l Conf. of Knowledge Discovery and Data Mining*, pp. 473–478, 2001.

[40] Thomas Zimmermann and Nachiappan Nagappan. Predicting defects using network analysis on dependency graphs. In *Proc. 30th Int'l Conf. on Software Engineering (ICSE'08)*, pp. 531–540, 2008.