

機械学習を用いた不吉な臭いの検出における 誤分類データの影響

梶田 利貴¹ 丸山 勝久^{1,a)}

概要：不格好な実装や設計の欠陥を指すコードの不吉な臭いは、ソースコードの理解を妨げ、さらに変更容易性の低下を招く。リファクタリングは、このようなコードの不吉な臭いを取り除く有力な手段である。リファクタリングとは、既存のプログラムの外部的振る舞いを維持しつつ、その内部構造を改善させる作業を指す。近年、リファクタリング対象の特定を支援するために、機械学習によるコードの不吉な臭いの自動検出の研究が行われている。しかしながら、検出の正確さは十分ではない。我々は、訓練データに含まれる誤分類データが検出の正確さを低下させている原因であると考えた。そこで、12 個の Java プロジェクトに含まれる 7 種類の不吉な臭いに対して、意図的に誤分類データを混入させたり、それらを機械的に除去したりすることで、不吉な臭いの検出の正確さの変化を調査した。その結果、誤分類データが含まれることで、正確さが有意に低下することが分かった。さらに、誤分類データを機械的に除去することが、必ずしも検出の正確さを向上させるとは限らないことも分かった。

キーワード：不吉な臭い, ソフトウェアメトリクス, 機械学習

1. はじめに

既存のソフトウェアに対する機能追加やバグ修正のためには、対象となるソースコードを深く理解することが必須である。ソースコード理解を支援する技術の一つにリファクタリングがある。リファクタリングとは、ソフトウェアの保守性や可読性を向上させることを目的として、既存のソフトウェアの外部的振る舞いを維持しつつ、その内部構造を変化させる活動である [1]。

リファクタリングを実施する際には、まずリファクタリング対象を特定することが必須である [2]。その際、コードの不吉な臭い（以下、不吉な臭いと呼ぶ）[1] が利用されることが多いが、その厳密な定義は存在しない。開発者は自身の知識や経験に基づき不吉な臭いを検出し、リファクタリング対象を特定しているのが現状である。さらに、不吉な臭いを検出するためには、注意深いソースコード解析が必須である。これは、開発者にとって面倒な作業である。

個々の開発者の能力に依存しないこと、さらに開発者の負担を軽減することを目的として、不吉な臭いを自動検出する手法が提案されている。その中でも、ソースコードメトリクスを利用する手法は、大きくヒューリスティックに

基づく手法と機械学習に基づく手法に分けられる [3]。

ヒューリスティックに基づく手法では、ソフトウェアメトリクスを組み合わせた検出ルール [4] をあらかじめ定義することによって、不吉な臭いを検出する [5–8]。代表的なツールとしては、DSL (Domain-Specific Language) を用いてソフトウェアメトリクスの組み合わせを柔軟に定義することが可能な DECOR [7] がある。他にも、ソフトウェアメトリクスに基づいた閾値と教師付きクラスタリングを組み合わせた JDeodorant [8] がある。

機械学習に基づく手法は、主に教師あり学習である。教師あり学習とは、説明変数の値と目的変数の値が既知である訓練データに対する学習を通して、説明変数と目的変数の対応関係を表現した予測モデルを構築する手法である。予測モデルを用いることで、説明変数の既知の値から目的変数の未知の値を予測することができる。ここでは、説明変数としてソフトウェアメトリクス、目的変数として不吉な臭いの有無を設定する。訓練データには、不吉な臭いを含む過去のプロジェクトのソースコードを利用する。

Kreimer は、分類器として決定木を採用することで、ソフトウェアメトリクスに基づき、Big Class と Long Method を検出するツールを提案した [9]。これに対して、Amorim らは、4 つの中規模のオープンソースプロジェクトで決定木の性能を評価した [10]。Khomh らは、Bayesian Belief

¹ 立命館大学
Ritsumeikan University
^{a)} maru@cs.ritsumei.ac.jp

Networks を用いて Blog アンチパターンを検出する手法を提案した [11]. Maiga らはサポートベクトルマシンを用いる手法を提案した [12]. Fontana らは、機械学習は不吉な臭いが有害であることを客観的に評価することができることを示し [13], 機械学習を用いることで不吉な臭いの強度を評価する手法を提案した [14].

さらに, Fontana らは, 4 種類の不吉な臭いの検出に対して 16 種類の分類アルゴリズムを比較した結果, 機械学習に基づく手法で 100%に近い F 値 (適合率と再現率の調和平均) が得られることを示した [15]. これに対して, Nucci らは, 実際の検出シナリオでは, 文献 [15] において報告された F 値の結果は一般化できないことを実証した.

そこで, Pecorelli らは, ヒューリスティックに基づく手法と機械学習に基づく手法で, どちらがより正確に不吉な臭いを検出できるのかを調査した [3]. 5 つの不吉な臭いに関して調査を行った結果, どちらも実用レベルの正確さを達成していなかったが, 多くの場面でヒューリスティックに基づく手法のほうが機械学習に基づく手法を用いるよりも優れていることが判明した. これに対して, Pecorelli らは, このような結果の原因が, 不吉な臭いの検出における訓練データがきわめて不均衡であるためと考え, データのバランシングを行うことによる正確さの変化を調査した [16,17]. 残念ながら, データの不均衡を取り除いても, 正確さに劇的な改善は見られないことが分かった.

このように, 機械学習に基づく手法は, 検出の正確さという点で, ヒューリスティックに基づく手法に劣る. その一方で, あらかじめ検出ルールを用意しなくてよいという利点を持つ. さまざまな不吉な臭いの検出に対して, ヒューリスティックに基づく手法を利用するためには, どのようなソフトウェアメトリクスを利用すればよいのか, ソフトウェアメトリクスの閾値をどのように設定するのかという難しい問題が残されている [18,19]. そこで, 筆者らは, 不吉な臭いの検出に対する機械学習の適用可能性の向上を目指して, 教師あり学習に基づく手法において, 検出の正確さを低下させている原因を明らかにすることを試みた.

ここで, 不吉な臭いを検出する予測モデルの構築において, 教師あり学習にそのまま利用可能な訓練データを準備することは一般的に困難である. 過去のソースコードに対して検出された不吉な臭いは, 開発者の能力や主観に大きく依存する. さらに, 開発者が間違っ検出したり, 検出漏れが発生したりすることもある. このような理由で, 訓練データには誤分類データが含まれる可能性がある. 筆者らは, このことが機械学習に基づく手法における検出の正確さを低下させていると考えた.

本論文では, 12 個のソフトウェアシステムに含まれる 7 種類の不吉な臭いに対して, 意図的に誤分類データを混入させたり, それらを機械的に除去したりすることで, 不

吉な臭いの検出の正確さの変化を調査した実験の結果を示す. 実験結果より, 誤分類データが存在することで, 正確さが有意に低下することが分かった. また, 誤分類データが含まれるかどうかに応じて, 予測モデルを構築するための適切な分類アルゴリズムが異なることが分かった. さらに, 誤分類データを機械的に除去することが, 必ずしも検出の正確さを向上させるとは限らないことも分かった.

以下, 2 章では, 誤分類データの影響に関する実験について述べる. 3 章では, 実験の結果を示す. 最後に 4 章で, まとめと今後の課題について述べる.

2. 実験

本章では, 誤分類データの影響を明らかにするための実験内容について説明する.

2.1 研究課題

本実験において明らかにする研究課題を以下に示す.

RQ1 訓練データに誤分類データが含まれると, 検出の正確さが低下するかどうか.

RQ2 訓練データに誤分類データが含まれている場合, 誤分類データを機械的に除去することで, 検出の正確さが向上するかどうか.

RQ3 誤分類データが含まれていない訓練データに対して, 誤分類データを機械的に除去する手法を適用した場合, 検出の正確さが変化するかどうか.

RQ1 と RQ2 に答えるための実験では, もとの訓練データに誤分類データを意図的に混入させ, 検出の正確さの変化を調査する. これに対して, RQ3 に答えるための実験では, 誤分類データを意図的に混入させずに, 誤分類データを除去する手法を適用する. たとえ RQ2 の実験結果において検出の正確さが向上していたとしても, RQ3 の実験結果において検出の正確さが低下していた場合, 誤分類データを除去する手法を適用すべきとは一概にはいえない.

2.2 データセット

本実験では, 文献 [21] で提供されている 30 個の Java プロジェクトから, 以下に示す 12 個を対象とした.

- Ant (ant): ビルドシステム
- ArgoUML (argouml): UML モデリングツール
- Derby (derby): 関係データベースシステム
- Eclipse Core (eclipse): 統合開発環境
- Elastic Search (elastic): 分散型検索・分析エンジン
- Hadoop (hadoop): 大規模分散処理フレームワーク
- HSQLDB (hsqldb): 関係データベースシステム
- Incubator (incub): コードベースサービス
- Nutch (nutch): Web 検索エンジン
- Qpid (qpid): メッセージングツール

表 1 検出する不吉な臭い

名前	説明	説明変数
Class Data Should Be Private (CDSBP)	公開されているフィールドを持つクラスが存在する	NOPA
Complex Class (CompClass)	高い循環的複雑度 [20] のメソッドを持つクラスが存在する	McCabe
God Class (GodClass)	異なる責任を実装したサイズの大きなクラスが存在する	ELOC, WMC, NOA, LCOM
Inappropriate Intimacy (InapIntim)	強く結びついている 2 つのクラスが存在する	FanIn, FanOut
Long Method (LongMethod)	行数の長いメソッドが存在する	LOC_METHOD, NP
Long Parameter List (LongParamList)	パラメータの多いメソッドが存在する	NP
Spaghetti Code (SpagCode)	複雑に絡み合っているメソッドを持つクラスが存在する	ELOC, NMNOPARAM

ELOC: 対象クラスにおいて、コメント、空行、括弧だけの行を除いた行数 (Effective Lines Of Code)
FanIn: 他のクラスから対象クラスへの参照の最大数
FanOut: 対象クラスから他のクラスへの参照の最大数
LCOM: 対象クラス内に存在するメソッドにおける結合性の欠如 (Lack of COhesion in Methods)
LOC_METHOD: 対象メソッドにおける行数 (Lines Of Code of METHOD)
McCabe: McCabe のサイクロマチック複雑度 (Cyclomatic Complexity)
NOA: 対象クラスにおける属性の数 (Number Of Attributes)
NOM: 対象クラスにおけるメソッドの数 (Number Of Methods)
NOPA: 対象クラスにおける公開属性の数 (Number Of Public Attributes)
NP: 対象メソッドにおける引数の数 (Number of Parameters)
NMNOPARAM: 対象クラスにおいて、引数を持たないメソッドの数 (Number of Methods with NO PARAMeters)
WMC: 対象クラスに含まれるメソッドを重みを付けて数えた値 (Weighted Methods Count)

- Wicket (wicket): Web アプリケーションフレームワーク
- Xerces (xerces): XML 解析器

検出する不吉な臭いは、データセット [21]*1を整理した 11 種類のデータセット [17]*2のうち、表 1 に示す 7 種類である。このデータセットにおける不吉な臭いは、複数人により手動で検証されたものである。よって、これを、誤分類データが含まれていない正解データとした。

表 1 における説明変数は、予測モデルを構築する際に利用したソフトウェアメトリクスを指す。これは、文献 [17] の設定と同じである。また、本実験における目的変数は、対象のソースファイルが表 1 に示す不吉な臭いを含むかどうかである。対象である 12 種のプロジェクトにおける不吉な臭いに関する統計量を表 2 に示す。数値は、各リリースに含まれる不吉な臭いを含むソースファイルの数を指す。

2.3 分類アルゴリズム

本実験において、予測モデルの構築の際に利用した分類アルゴリズムは、ナイーブベイズとロジスティック回帰である。ナイーブベイズを採用した理由は、文献 [3] の実験において、もっとも優秀な性能を示したからである。その利用においては、各ソフトウェアメトリクスは独立であると仮定する。対象のソースファイルに対して測定したソフトウェアメトリクスの値から、それが不吉な臭いを含む確率と含まない確率をベイズの定理に基づき推定し、確率の高い方を判定結果とする。

ロジスティック回帰を採用した理由は、単純かつ説明容

表 2 不吉な臭いに関する統計量

不吉な臭い	最小値	平均	中央値	最大値	合計
CDSBP	0	13	11	43	1,703
CompClass	0	9	4	53	1,212
GodClass	0	6	5	25	843
InapIntim	0	2	2	13	349
LongMethod	4	85	48	368	10,663
LongParamList	0	12	0	72	1,592
SpagCode	0	13	11	33	1,692

易性が高く、ベースラインとしてよく利用されるからである。その利用においては、測定したソフトウェアメトリクスの値をロジスティック関数に入力することで、不吉な臭いが含まれる確率を求める。求めた値が閾値を超えている場合に、対象のソースファイルに不吉な臭いが含まれると判定する。

予測モデルの構築には、scikit-learn*3を用いた。本実験では、パラメータチューニングは実施していない。

2.4 評価指標

不吉な臭いの検出に関する正確さは、以下に示すマシユーの相関係数 (Matthews Correlation Coefficient, 以下 MCC) で判断する。

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

TP, FP, TN, FN はそれぞれ、True Positive, False Positive, True Negative, False Negative を指す。

MCC は予測した二値分類結果 (対象のソースファイルが不吉な臭いを含むかどうか) と実測結果の相関係数である。この値は -1 と +1 となる。予測結果と実測結果が完

*1 <https://dibt.unimol.it/staff/fpalomba/reports/badSmell-analysis/>

*2 <https://figshare.com/s/5da162e21b8d54fbfce8>

*3 <https://scikit-learn.org/>

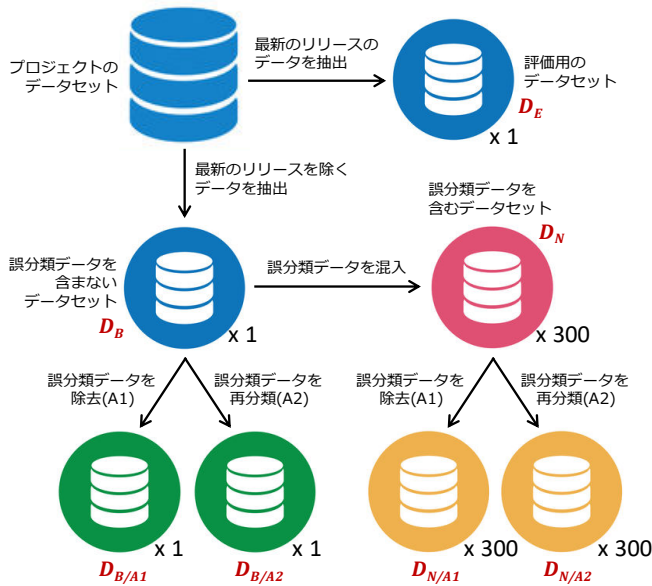


図 1 実験のために作成したデータセット

全に一致すると +1, 完全に不一致だと -1, ランダムな予測と同等だと 0 となる。MCC は, 正負の割合 (本実験では, 不吉な臭いを含むか含まないか) が不均衡な場合でも影響を受けにくいという特徴を持つ。

2.5 実験方法

本実験では, 12 個の対象プロジェクトのデータセットから, 図 1 に示す 7 種類のデータセットをそれぞれ作成する。ここで, もとのデータセットにおいて, 不吉な臭いを含むソースファイル数と含まないソースファイルの数は不均衡である。その一方で, Pecorelli ら調査 [16,17] により, データの不均衡を取り除いても正確性に劇的な改善は見られないことが報告されている。よって, 本実験では, データのバランシングを実施していない。

D_B は, 各プロジェクトのデータセットから, 最新のリリースを除いたすべてのリリースを抽出したものである。このデータセットは, 予測モデルを構築するための訓練データとして利用する。

D_E は, 各プロジェクトのデータセットから最新のリリースを抽出したものである。このデータセットは, 予測モデルを評価するためのデータとして利用する。

D_N は, D_B に対して, 任意の数の目的変数の値をランダムに変化させることによって, 誤分類データを混入させたデータセットである。本実験では, 文献 [22] で提供されている cleanlab^{*4}を利用して, 平均 2% の誤分類データを混入させた。生成したデータセットは各プロジェクトに対して 300 個である。

データセットに含まれる誤分類データを除去する手法として, 以下の 2 つを用意した。

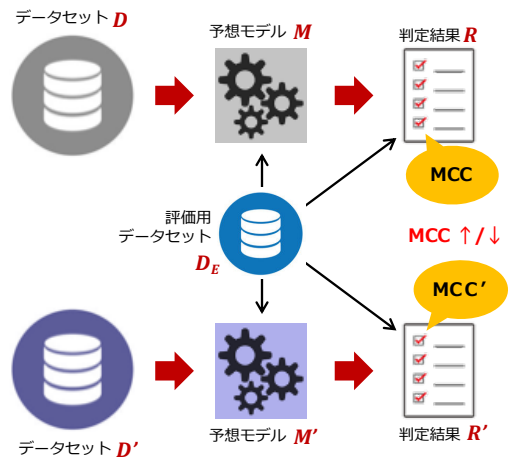


図 2 実験の手順

A1: cleanlab を用いて, 誤分類データを推測し, それらを訓練データから削除する。

A2: cleanlab を用いて, 誤分類データを推測し, それらに疑似ラベルを付与して訓練データに戻す。

$D_{N/A1}$ と $D_{N/A2}$ は, D_N から手法 A1 および手法 A2 を利用することで, 誤分類データを除去した 300 個のデータセットである。 $D_{B/A1}$ と $D_{B/A2}$ は, D_B に手法 A1 および手法 A2 を適用したデータセットである。

2.1 節に述べた 3 つの研究課題に答えるために, これら 7 種類のデータセットを利用した 5 つの実験を行った。5 つの実験は, 対象とするデータセットが異なるだけで, その手順は同一である。

図 2 に実験の手順を示す。2 つのデータセット D および D' を訓練データとして, ナイーブベイズあるいはロジスティック回帰による機械学習を行うことで, 予測モデル M と M' をそれぞれ構築する。その後, 評価用データセット D_E を予測モデル M および M' に適用することで, 不吉な臭いを検出する。最後に, 予測値 (検出結果 R および R') と D_E に含まれる実測値から, 2.4 節で説明した MCC を計算し, それらの値を集計する。

以下, それぞれの実験について説明する。

2.5.1 実験 1

RQ1 に答えるために, 訓練データに誤分類データが含まれている場合と誤分類データが含まれていない場合で, 検出の正確さを比較する。この実験では, 図 2 において, $D \leftarrow D_B$ および $D' \leftarrow D_N$ となる。

検出結果 R の MCC に比べて検出結果 R' の MCC が減少しているとき, 誤分類データの存在が不吉な臭いの検出における正確さを低下させているといえる。 D_B の標本は 1 個であるので, MCC の増減に有意な差があるかどうかを確認するために 1 標本 t 検定を実施する。

2.5.2 実験 2

RQ2 に答えるために, 訓練データに含まれる誤分類デー

^{*4} <https://github.com/cgnorthcutt/cleanlab>

タを除去しない場合と除去する場合で、検出の正確さを比較する。この実験では、手法 A1 と手法 A2 を利用して、それぞれ実施する。実験 2-A1 では、図 2 において、 $D \leftarrow D_N$ および $D' \leftarrow D_{N/A1}$ となる。実験 2-A2 では、 $D \leftarrow D_N$ および $D' \leftarrow D_{N/A2}$ となる。

検出結果 R の MCC に比べて検出結果 R' の MCC が増加しているとき、誤分類データの除去が不吉な臭いの検出における正確さを上昇させているといえる。 D_N , $D_{N/A1}$, $D_{N/A2}$ の MCC が正規分布でないため、MCC の増減に有意な差があるかどうかを確認するために、Mann-Whitney の U 検定を実施する。

2.5.3 実験 3

RQ3 に答えるために、誤分類データを含まない訓練データに対して手法 A1 と手法 A2 を適用した場合に、検出の正確さが変化するかどうかを確認する。実験 3-A1 では、図 2 において、 $D \leftarrow D_B$ および $D' \leftarrow D_{B/A1}$ となる。実験 3-A2 では、 $D \leftarrow D_B$ および $D' \leftarrow D_{B/A2}$ となる。

一般的に、手法 A1 と手法 A2 を適用することで誤分類データが除去できたとしても、訓練データの数が減ることや疑似ラベルを付与することが、検出の正確さを低下させる可能性がある。検出結果 R の MCC に比べて検出結果 R' の MCC が変化しない（あるいは、増加している）とき、誤分類データがもとの訓練データに含まれているかどうかを気にせず、それらの除去を試みるのがよいといえる。反対に、MCC が減少しているとき、誤分類データの除去は適用しない方がよい。 D_B , $D_{B/A1}$, $D_{B/A2}$ の標本はどれも 1 個であるので検定を利用せず、MCC の増減率で正確さの変化を確認する。

3. 結果

D_B を訓練データとしたときの評価における MCC の値を表 3 に示す。また、 D_I を訓練データとしたときの評価における MCC の平均値を表 4 に示す。これらの表において、0.5 以上の値を緑色で示す。

それぞれの実験において、MCC 値の増減に対して検定した結果を表 5 に示す。実験 1 と実験 2 では、 $p < 0.01$ で検定した際に有意差が認められる場合、MCC の値が増加あるいは減少と判定した。実験 3 では、増減率を 10% に設定し、それを超える場合を増加あるいは減少と判定した。

「U:D:E」の U は MCC の値が増加したと判定されたプロジェクトの数を指す。また、D は MCC の値が減少したと判定されたプロジェクトの数を指す。E は、増加あるいは減少と判定されなかったプロジェクトの数を指す。括弧付きの記号 (U), (D), (E) は、単独でもっともプロジェクト数が多い変化を指している。

本実験では、訓練データに、検出対象の不吉な臭いが含まれていないプロジェクトや、実験 2 において MCC の値

のユニークな数が 3 個より少ないプロジェクトは評価対象としていない。このため、評価対象としたプロジェクトの総数 ($U + D + E$) は 12 個とは限らず、それ以下となる。

それぞれの実験で得られた MCC 値と、MCC 値の増減に対する検定結果を Web サイト^{*5} で公開する。

3.1 実験 1

表 3 に示す実験 1 の結果から、誤分類データが含まれていない場合において、MCC の値は不吉な臭いの種類と対象プロジェクトに応じて異なることが分かる。たとえば、GodClass に対しては、ナীবベイズとロジスティック回帰のどちらを利用しても、MCC 値の高いプロジェクトが多い。それに対して、SpagCode の MCC 値はどのプロジェクトに対しても低い。また、ナীবベイズを用いた場合、xerces に対する MCC の値は高い。それに対して、ロジスティック回帰を用いた場合、hadoop に対する MCC の値は高い。

表 3 と表 4 を見ると、MCC の値あるいは MCC の平均値が 0.5 以上（緑色）のプロジェクトの数が 13 個から 7 個（ナীবベイズ）、15 個から 8 個（ロジスティック回帰）に減少している。このことより、全体的に MCC の値が減少していると考えられる。ただし、表 4 に示す値は MCC の平均値であるため、表 3 と表 4 の数値を単純に比較するだけで、検出の正確さが増加あるいは減少していると判断できないことに注意する必要がある。

そこで、実験 1 では、1 標本 t 検定を実施することで、検出の正確さが有意に増加あるいは減少しているかどうかを確認した。検定結果をまとめた表 5 を見ると、実験 1 において、ナীবベイズを用いた場合、合計 14 個のプロジェクトで MCC の値が有意に増加している。それに対して、合計 48 個のプロジェクトで MCC の値が有意に減少している。減少したプロジェクトの数は全体の約 74% となっており、その割合は高い。さらに、どの不吉な臭いの検出においても、MCC の値が減少したプロジェクトの数が、MCC の値が増加したプロジェクトの数を上回っている。ロジスティック回帰を用いた場合、合計 19 個のプロジェクトで MCC の値が有意に増加、合計 24 個のプロジェクトで MCC の値が有意に減少している。減少したプロジェクトの割合は 50% に抑えられている。

RQ1 に対する回答

訓練データに誤分類データが含まれると、検出の正確さが低下する傾向にある。また、ロジスティック回帰に比べて、ナীবベイズは誤分類データの混入により悪い影響を受けやすい。

^{*5} <https://www.fse.cs.ritsumei.ac.jp/sigse208Appendix>

表 3 D_B を訓練データとしたときの評価における MCC の値

分類アルゴリズム	不吉な臭い	ant	argouml	derby	eclipse	elastic	hadoop	hsqldb	incub	nutch	qpid	wicket	xerces
ナイーブベイズ	CDSBP	0.27	0.47	0.13	0.16	0.43	0.58	0	0.44	0.51	0.48	N/A	0.73
	CompClass	0.32	N/A	0.32	0.29	N/A	0.41	0.49	0.49	N/A	0.51	0.15	0.53
	GodClass	0.39	N/A	0.49	0.37	0.71	0.31	0.64	0.54	N/A	0.41	0.38	0.56
	InapIntim	0.15	0	N/A	0.16	0.41	0.28	0.23	0.49	0.44	0.11	0.25	-0.01
	LongMethod	0.27	0.07	0.29	0.20	0.14	0.36	0.35	0.34	0.20	0.21	0.08	0.44
	LongParamList	0.32	N/A	N/A	N/A	N/A	0.50	0.44	0.59	0.79	0.43	N/A	0.74
	SpagCode	0.41	0.04	0.14	0.17	0.16	0.35	0.28	0.31	N/A	0.15	0.17	0.19
ロジスティック回帰	CDSBP	0.27	0.35	0.10	0.17	0.29	0.58	0	0.14	0.32	0	N/A	0.66
	CompClass	0	N/A	0.30	0.04	N/A	1.00	-0.01	0	N/A	0.19	0.71	0.47
	GodClass	0.77	N/A	0.59	0.51	0	1.00	0.45	0.47	N/A	0.29	0.35	0.61
	InapIntim	0	0	N/A	0	0	-0.01	0	0.52	0	0	0.71	0
	LongMethod	0.32	0	0.25	0.15	0	0.50	0.43	0.21	0.19	0.25	0	0.44
	LongParamList	0.61	N/A	N/A	N/A	N/A	0.30	0.55	0.34	0.40	0.68	N/A	0.23
	SpagCode	0.34	0	0.14	0.09	0.41	-0.01	0.08	0	N/A	0	0	-0.01

表 4 D_N を訓練データとしたときの評価における MCC の平均値

分類アルゴリズム	不吉な臭い	ant	argouml	derby	eclipse	elastic	hadoop	hsqldb	incub	nutch	qpid	wicket	xerces
ナイーブベイズ	CDSBP	0.24	0.53	0.13	0.16	0.26	0.37	0	0.43	0.51	0.48	N/A	0.71
	CompClass	0.31	N/A	0.32	0.32	N/A	0.34	0.48	0.45	N/A	0.44	0.20	0.52
	GodClass	0.30	N/A	0.44	0.38	0.21	0.33	0.59	0.44	N/A	0.28	0.26	0.44
	InapIntim	0.13	0	N/A	0.04	0.10	0.24	0.19	0.46	0.09	0.09	0.20	-0.01
	LongMethod	0.26	0.10	0.28	0.19	0.16	0.34	0.33	0.34	0.22	0.18	0.13	0.44
	LongParamList	0.32	N/A	N/A	N/A	N/A	0.41	0.27	0.52	0.55	0.34	N/A	0.48
	SpagCode	0.40	0.04	0.12	0.16	0.17	0.32	0.30	0.29	N/A	0.14	0.16	0.24
ロジスティック回帰	CDSBP	0.35	0.32	0.01	0.17	0.43	0.57	0	0.17	0.19	0	N/A	0.66
	CompClass	0	N/A	0.35	0.04	N/A	0.25	0	0	N/A	0.18	0.62	0.29
	GodClass	0.52	N/A	0.58	0.45	0	0.98	0.14	0.48	N/A	0.28	0.36	0.66
	InapIntim	0	0	N/A	0	0	0	0	0.53	0	0	0.24	0
	LongMethod	0.14	0	0.19	0.11	0	0.25	0.42	0.24	0.03	0.12	0	0.44
	LongParamList	0.25	N/A	N/A	N/A	N/A	0	0.06	0.23	0.04	0.17	N/A	0.03
	SpagCode	0.35	0	0.04	0.10	0.03	-0.01	0.08	0	N/A	0	0	-0.01

3.1.1 実験 2

表 5 に示す実験 2-A1 と実験 2-A2 の結果を見ると、ナイーブベイズを用いた場合とロジスティック回帰を用いた場合に大きな違いが観測された。このことより、訓練データに誤分類データが含まれている場合、誤分類データを除去することで、検出の正確さが向上するかどうかは分類アルゴリズムに依存するといえる。その一方で、誤分類データを除去する手法 A1 と手法 A2 のどちらを適用しても、それらの効果に大きな違いは見られない。

ナイーブベイズを用いた場合、手法 A1 および手法 A2 により誤分類データを除去しても、すべての不吉な臭いに対して、MCC の値が減少したプロジェクトの数が、MCC の値が増加したプロジェクトの数を上回っている。これに対して、ロジスティック回帰を用いた場合、手法 A1 および手法 A2 のどちらにおいても、5 種類の不吉な匂いに対して、MCC の値が増加したプロジェクトの数が、MCC の値が減少したプロジェクトの数を上回っている。

ここで、ナイーブベイズを用いた場合の MCC の値とロ

ジスティック回帰を用いた場合の MCC の値を個々に見ると、必ずしもナイーブベイズを利用しない方がよいという結論にならないことに注意する必要がある。不吉な臭いの種類やプロジェクトによって、誤分類データが含まれる D_N に対するナイーブベイズを用いた場合の MCC の値の方が、誤分類を除去した $D_{N/A1}$ や $D_{N/A1}$ に対するロジスティック回帰を用いた場合の MCC の値よりも高い場合が存在する。このことは、たとえ誤分類データが含まれていても、ナイーブベイズを用いる方が不吉な臭いの検出に有利である場面があることを指している。

RQ2 に対する回答

ナイーブベイズを用いた場合、誤分類データが含まれている訓練データに対して、誤分類データの除去は検出の正確さを低下させる傾向にある。これに対して、ロジスティック回帰を用いた場合、誤分類データを除去は検出の正確さを向上させる可能性が高い。

表 5 MCC 値の増減に対する検定結果

分類アルゴリズム	不吉な臭い	実験 1	実験 2-A1	実験 2-A2	実験 3-A1	実験 3-A2
		U:D:E	U:D:E	U:D:E	U:D:E	U:D:E
ナイーブベイズ	CDSBP	2:7:0 (D)	5:6:0 (D)	5:6:0 (D)	4:2:4	4:3:3 (U)
	CompClass	3:6:0 (D)	1:10:0 (D)	1:10:0 (D)	3:1:5 (E)	2:3:4 (E)
	GodClass	1:8:0 (D)	0:11:0 (D)	0:11:0 (D)	0:8:2 (D)	0:8:1 (D)
	InapIntim	0:7:2 (D)	2:10:0 (D)	2:10:0 (D)	2:7:1 (D)	1:7:1 (D)
	LongMethod	5:7:0 (D)	3:8:1 (D)	3:8:1 (D)	1:3:8 (E)	2:3:7 (E)
	LongParamList	0:6:0 (D)	0:12:0 (D)	0:12:0 (D)	0:2:5 (E)	0:4:3 (D)
	SpagCode	3:7:1 (D)	4:8:0 (D)	5:7:0 (D)	1:5:5	2:5:4 (D)
	合計	14:48:3	15:54:1	16:53:1	11:28:30	11:33:23
ロジスティック回帰	CDSBP	3:5:1 (D)	6:5:0 (U)	5:6:0 (D)	4:5:1 (D)	3:4:3 (D)
	CompClass	3:1:2 (U)	6:4:1 (U)	6:5:0 (U)	2:4:3 (D)	3:4:2 (D)
	GodClass	3:4:1 (D)	7:3:0 (U)	7:3:0 (U)	6:0:3 (U)	5:0:3 (U)
	InapIntim	2:0:0 (U)	4:4:0	3:5:0 (D)	0:2:2	0:2:1 (D)
	LongMethod	3:7:0 (D)	11:1:0 (U)	11:0:1 (U)	8:2:2 (U)	8:2:2 (U)
	LongParamList	0:6:0 (D)	6:6:0	7:5:0 (U)	5:0:2 (U)	5:0:2 (U)
	SpagCode	5:1:1 (U)	9:3:0 (U)	7:5:0 (U)	4:7:0 (D)	5:5:1
	合計	19:24:5	49:13:1	46:16:1	29:20:13	29:17:14

3.1.2 実験 3

表 5 に示す実験 3-A1 と実験 3-A2 の結果を見ると、どちらの分類アルゴリズムを用いた場合でも、MCC の値が変化しないプロジェクトが数多く存在することが分かった。これらのプロジェクトでは、誤分類データの除去を適用したことに影響を受けていないとみなせる。

残念ながら、ナイーブベイズを用いた場合には、MCC の値が減少したプロジェクトの数が、MCC の値が増加したプロジェクトの数を上回っている。このことは、ナイーブベイズを用いる場合には、誤分類データが含まれていない訓練データに対して悪い影響を与えることを指している。実験 2 の結果と合わせて考えると、ナイーブベイズを用いる場合、訓練データに誤分類データが含まれていてもいなくても、それらの除去を適用しない方がよいといえる。

これに対して、ロジスティック回帰を用いた場合、MCC の値が増加したプロジェクトの数が、MCC の値が減少したプロジェクトの数を上回っている。実験 2 の結果と合わせて考えると、誤分類データの除去を適用することで、MCC の値が増加する傾向にあるといえる。MCC の値が減少するプロジェクトが存在することを無視することはできないが、誤分類データの除去を適用することが奨励される。

RQ3 に対する回答

ナイーブベイズを用いた場合、誤分類データが含まれていない訓練データに対して、誤分類データを除去する手法を適用することで、検出の正確さが低下する傾向にある。これに対して、ロジスティック回帰を用いた場合、誤分類データを除去する手法を適用しても、検出の正確さが低下することへの影響は少ない。

3.2 妥当性への脅威

本実験で利用したデータセットにおける分類は人手で検証されているものの、ヒューマンエラーにより誤分類データが含まれる可能性は否定できない。誤分類データがはじめから含まれていた場合、実験結果に悪影響を与える可能性がある。

また、表 1 に示すソフトウェアメトリクスを利用して、予測モデルを構築する。これは文献 [17] と同じであるが、これらのメトリクスが不吉な臭いの検出に適しているとは限らない。実際、表 3 を見ると、Spaghetti Code に対する検出の正確さは、誤分類データの有無にかかわらず低い。このことは、利用するメトリクスが妥当でない可能性を指している。利用するソフトウェアメトリクスによって、誤分類データを含まない場合だけでなく、誤分類データを含む場合の検出の正確さも変化する可能性がある。

本実験では、cleanlab を利用して、平均 2% の誤分類データを混入させた。データセット中の目的変数の値をランダムに変更させるという戦略で作られた誤分類データが、実際のデータセットに含まれる誤分類データと同じような傾向（数や分布）を示すとは限らない。当然であるが、混入させる誤分類データの割合や利用するツールを変えることで、実験対象のデータセットが変わる。これにより、実験結果が変わる可能性がある。また、誤分類データの推測にも cleanlab を利用している。利用するツールを変えることで、実験対象のデータセットが変わり、実験結果も追従して変わる可能性がある。

検出の正確さをマシユーの相関係数 (MCC) で計測している。別の指標を用いて評価することで、検出の正確さの増減に関する主張が変わる可能性が高い。

12 個のソフトウェアシステムに含まれる 7 種類の不吉

な臭いを対象とした。用意したソフトウェアシステムの数はまだまだ少なく、実際に不吉な臭いを含むソフトウェアシステムを代表している保証はない。よって、他のソフトウェアシステムを利用した場合に、実験結果が変わる可能性がある。さらに、別の不吉な臭いを対象とした場合にも、実験結果が変わる可能性がある。

4. おわりに

不吉な臭いの検出に対する機械学習の適用可能性の向上を目指して、意図的に誤分類データを混入させた場合の検出の正確さの変化を調査した。さらに、誤分類データを機械的に除去する2つの手法を適用し、検出の正確さの変化を調査した。

調査の結果、ナイーブベイズとロジスティック回帰のどちらを用いた場合でも、誤分類データの混入により検出の正確さは減少する傾向にあることが分かった。このような状況において、ナイーブベイズを用いた場合、誤分類データを除去する手法を適用しても、検出の正確さは向上しないことが多いことが確認できた。これに対して、ロジスティック回帰を用いた場合、誤分類データの除去は検出の正確さをそれほど低下させないことが確認できた。ただし、不吉な臭いの種類やプロジェクトによっては、誤分類データを除去する手法を適用せずに、そのままナイーブベイズを用いる方が不吉な臭いの検出において有利である場面があることも判明した。

以上より、教師あり学習に基づく手法を用いて不吉な臭いを検出する際、誤分類データが含まれるかどうかに応じて、予測モデルを構築するために適切な分類アルゴリズムを選択することが重要である。このような選択を実現するためには、訓練データにどの程度の誤分類データが含まれているのかを推測する手法が必要である。また、訓練データに含まれる誤分類データの程度が検出の正確さに与える影響を一般化するためには、プロジェクトの数や不吉の臭いの種類をさらに増やした実験が必須である。

参考文献

- [1] Fowler, M.: *Refactoring: Improving the Design of Existing Code*, Addison-Wesley (1999).
- [2] Mens, T. and Tourwé, T.: A Survey of Software Refactoring, *IEEE TSE*, Vol. 30, No. 2, pp. 126–139 (2004).
- [3] Pecorelli, F., Palomba, F., Nucci, D. D. and Lucia, A. D.: Comparing Heuristic and Machine Learning Approaches for Metric-Based Code Smell Detection, *Proc. ICPC '19*, pp. 93–104 (2019).
- [4] Lanza, M. and Marinescu, R.: *Object-Oriented Metrics in Practice*, Springer-Verlag (2010).
- [5] Marinescu, R.: Detection Strategies: Metrics-Based Rules for Detecting Design Flaws, *Proc. ICSM '04*, pp. 350–359 (2004).
- [6] Munro, M. J.: Product Metrics for Automatic Identification of “Bad Smell” Design Problems in Java Source-

- Code, *Proc. METRICS '05*, pp. 1–9 (2005).
- [7] Moha, N., Guéhéneuc, Y.-G., Duchien, L. and Meur, A.-F. L.: DECOR: A Method for the Specification and Detection of Code and Design Smells, *IEEE TSE*, Vol. 36, No. 1, pp. 20–36 (2010).
- [8] Tsantalis, N. and Chatzigeorgiou, A.: Identification of Move Method Refactoring Opportunities, *IEEE TSE*, Vol. 35, No. 03, pp. 347–367 (2009).
- [9] Kreimer, J.: Adaptive Detection of Design Flaws, *Electronic Notes in Theoretical Computer Science*, Vol. 141, No. 4, pp. 117–136 (2005).
- [10] Amorim, L., Costa, E., Antunes, N., Fonseca, B. and Ribeiro, M.: Experience Report: Evaluating the Effectiveness of Decision Trees for Detecting Code Smells, *Proc. ISSRE '15*, pp. 261–269 (2015).
- [11] Khomh, F., Vaucher, S., Guéhéneuc, Y.-G. and Sahraoui, H.: A Bayesian Approach for the Detection of Code and Design Smells, *Proc. QSIC '09*, pp. 305–314 (2009).
- [12] Maiga, A., Ali, N., Bhattacharya, N., Sabané, A., Guéhéneuc, Y.-G., Antoniol, G. and Aïmeur, E.: Support Vector Machines for Anti-Pattern Detection, *Proc. ASE '12*, pp. 278–281 (2012).
- [13] Fontana, F. A., Zanoni, M., Marino, A. and Mäntylä, M. V.: Code Smell Detection: Towards a Machine Learning-Based Approach, *Proc. ICSM '13*, pp. 396–399 (2013).
- [14] Fontana, F. A. and Zanoni, M.: Code Smell Severity Classification Using Machine Learning Techniques, *Knowledge-Based Systems*, Vol. 128, pp. 43–58 (2017).
- [15] Fontana, F. A., Mäntylä, M. V., Zanoni, M. and Marino, A.: Comparing and Experimenting Machine Learning Techniques for Code Smell Detection, *EMSE*, Vol. 21, No. 3, pp. 1143–1191 (2016).
- [16] Pecorelli, F., Nucci, D. D., Roover, C. D. and Lucia, A. D.: On the Role of Data Balancing for Machine Learning-Based Code Smell Detection, *Proc. MaL-TeSQuE '19*, pp. 19–24 (2019).
- [17] Pecorelli, F., Nucci, D. D., Roover, C. D. and Lucia, A. D.: A Large Empirical Assessment of the Role of Data Balancing in Machine-Learning-Based Code Smell Detection, *JSS*, Vol. 169, p. 110693 (2020).
- [18] Zhang, M., Hall, T. and Baddoo, N.: Code Bad Smells: A Review of Current Knowledge, *JSME*, Vol. 23, No. 3, pp. 179–202 (2011).
- [19] Fernandes, E., Oliveira, J., Vale, G., Paiva, T. and Figueiredo, E.: A Review-Based Comparative Study of Bad Smell Detection Tools, *Proc. EASE '16* (2016).
- [20] McCabe, T. J.: A Complexity Measure, *IEEE TSE*, Vol. 2, No. 4, pp. 308–320 (1976).
- [21] Palomba, F., Bavota, G., Penta, M. D., Fasano, F., Oliveto, R. and Lucia, A. D.: On the Diffuseness and the Impact on Maintainability of Code Smells: A Large Scale Empirical Investigation, *EMSE*, Vol. 23, pp. 1188–1221 (2018).
- [22] Northcutt, C., Jiang, L. and Chuang, I.: Confident Learning: Estimating Uncertainty in Dataset Labels, *Journal of Artificial Intelligence Research*, Vol. 70 (2020).