

技術的負債に関連する 課題票抽出手法の構築

オープンソースソフトウェア工学研究室

学籍番号：60210032

氏名：木村 祐太

技術的負債：Technical Debt^[1]

- 短期的には有用，長期的には問題がある設計や実装
 - 例) レガシーコード : かなり前に実装されたコード
 - コードスメル : 深刻な問題を抱えているコードの特徴
- 保守コストの増加を引き起こす^{[2][3][4][5]}
 - 開発の約23～35%が余分な作業
 - 既存のTDによる新たなTDの導入 (TDの蓄積)

開発資源（人的，金銭的，時間的）は有限
TDの蓄積量を管理する必要がある

[1] Cunningham, W., 1992. The wycash portfolio management system. Proc. Obj. Oriented Programm. Syst. Lang. Appl. 4 (2), pp. 29-30.

[2] C. Seaman, G. Yuepu, N. Zazworka, F. Shull, C. Izurieta, C. Yuanfang, et al., "Using technical debt data in decision making: Potential decision approaches," in Managing Technical Debt (MTD), 2012 Third International Workshop on, 2012, pp. 45-48.

[3] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt," Journal of Systems and Software, vol. 86, no. 6, 2013, pp. 1498-1516.

[4] T. Besker, A. Martini and J. Bosch, "Technical Debt Cripples Software Developer Productivity: A Longitudinal Study on Developers' Daily Software Development Work," 2018 IEEE/ACM International Conference on Technical Debt (TechDebt), 2018, pp. 105-114.

[5] T. Besker, A. Martini, and J. Bosch, "The pricey Bill of Technical Debt: When and by whom will it be paid?," in IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 2017, pp. 13-23.

Self-Admitted TD

- 開発者によってTDが明示的に管理されるケース
 - SATD-Issue ^[6]
 - 課題票 (Issue) にTDラベルを割り当てて明示されるSATD
- ※ Issue : 不具合, 機能改善, 機能要求などの報告される課題

Cannot use IDialogService in some services due to custom dialog service dependencies
#109056

commented on 21 Oct 2020

Trying to use `IDialogService` in `ExtensionService` or in its dependencies like `ExtensionManagementService` is causing workbench to explode because dialog service is depending on `Layout` that depends on `ExtensionService`.

Due to this dependency, `DialogService` cannot be used in many services. We should allow dialog service to be used by all workbench services and not introduce such a restriction.

Added **important** **debt** labels on 21 Oct 2020

Assigned **bpasero** and **sbatten** on 21 Oct 2020

commented on 21 Oct 2020

This came in via [1d4c7fa](#) and is probably a consequence of having the `window.dialogStyle: custom` setting even in native.

It does not seem to be the `ILayoutService` dependency btw, it seems to be these:

- `IThemeService`

Assignees

sbatten

Labels

debt **dialogs** **important** **insiders-released**

Projects

None yet

Milestone

November 2020

Linked pull requests

Successfully merging a pull request may close this issue.

[Refactor DialogService to use model and c...](#)



“debt”というTDラベルが
割り当てられている

引用 : <https://github.com/microsoft/vscode/issues/109056>

[6] Laerte Xavier, Fabio Ferreira, Rodrigo Brito, and Marco Tulio Valente. Beyond the Code: Mining Self-Admitted Technical Debt in Issue Tracker Systems. In Proceedings of the 17th International Conference on Mining Software Repositories (MSR '20). 2020, pp. 137–146.

SATD-Issue研究の問題点

- 分析対象が少なく一般性が保障できない
 - SATD-Issueを扱っているプロジェクトが少ないためデータの絶対数が少ない



本研究の目的とアプローチ

目的

- SATD-Issueを自動抽出しデータを増やす

アプローチ

- SATD-Issueを抽出するモデルの構築
 1. モデル構築に利用する特徴量の選定
 2. モデルの評価実験
 - ・ ランダムフォレスト, ロジスティック回帰, サポートベクターマシン
 3. 他プロジェクトを利用したモデルの一般性の検証

データセット

リポジトリ	ドメイン	TD-Issue [件]	通常Issue [件]
microsoft/vscode	エディタ	1,612	102,292
influxdata/influxdb	データベース	111	9,903
saleor/saleor	プラットフォーム	106	2,623
nextcloud/server	インフラ	73	9,273

- 4つのOSSプロジェクトのデータを利用
 - GitHubにホスティングされている
 - SATD-Issueを慣習的に扱っている
- 課題内容が確定しているClosed Issue（解決済み）のみを対象

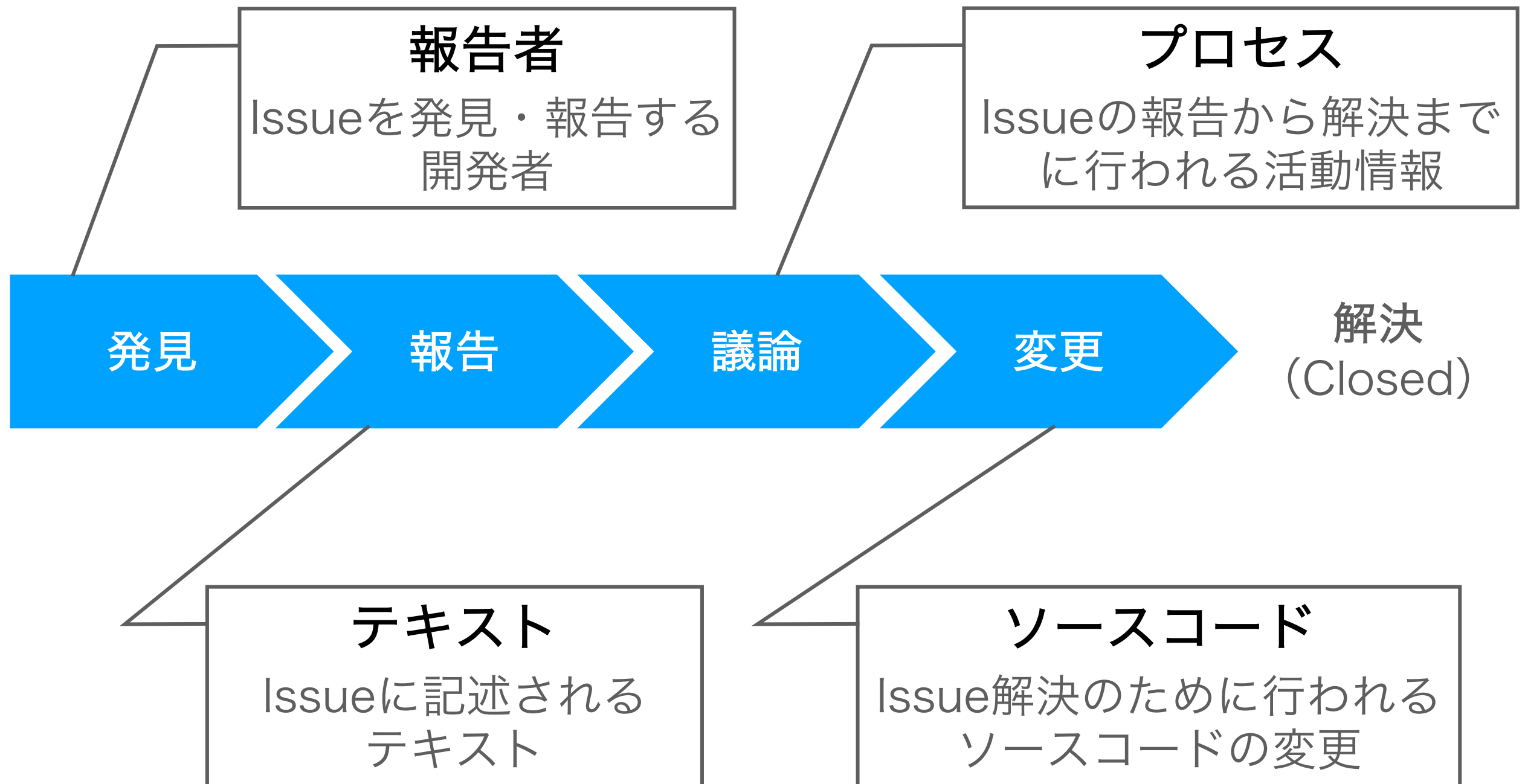
特徴量の検討

- Closed Issueから取得できる特徴量を抽出
 - SATD-Issue分類に関する研究が存在しないため、有効な特徴量が不明
- Issue発見から解決までの工程における構成要素に着目



Issue発見から解決までの工程

特徴量の抽出



使用するテキストと前処理

- 使用するテキスト
 - Issueのタイトルと本文
- 前処理
 - 1-gram（単語）に分割
 - ストップワードや記号などノイズとなる単語の除去
 - 特定の単語を置換
 - URL, コード片, 固有名詞, など
 - 単語の正規化
 - 小文字統一, ステミング
 - BoW（Bag of Words）の用いたベクトル化

特徴量選択と削減

- 報告者・プロセス・ソースコードに関連する特徴量を対象に以下の方法を実施
 - 仮説検定と効果量測定
 - 量的データ
仮説検定：マンホイットニーのU検定, 効果量測定：Cliff's d
 - 質的データ
仮説検定：カイ二乗検定, 効果量測定：Cohen's w
 - VIF統計量による特徴量削減
 - 多重共線性の高い特徴量を使用しない

有意差 ($p < 0.01$) があり効果量が小程度以上かつ
VIF統計量が3未満の特徴量を選択

特徴量一覧

灰色：外れ値除去とモデルの学習に利用している特徴量
有意差：- ($0.05 < p$) , * ($p < 0.05$) , ** ($p < 0.01$)
効果量：Negligible (効果なし) , Small (小) , Medium (中) , Large (大)

CodeChurnAdd (白色)

- p値1%水準で有意差があり，効果が中程度ある

CodeChurnDel (灰色)

- p値1%水準で有意差があり，効果量が大幅度ある

プロセス (9次元)	CommentNum	課題票のコメントの数	改善方法を決めるために多くの議論がされる	**	Small
	SelfAssign	報告者自身が課題票の担当になっているか	間に合わせて実装したコードを後から自分で変更する	**	Small
	CodeChurnAdd	変更時に追加したコードの総行数		**	Medium
	CodeChurnDel	変更時に削除したコードの総行数		**	Large
	CountDeclFunctionAdd	変更によって増加した関数の数		**	Small
	CountDeclFunctionDel	変更によって減少した関数の数		**	Small
	CountLineCodeAdd	変更によって増加したコードの行数規模	リファクタリングによりファイル規模 (LOC) が減少する	**	Small
	CountLineCodeDel	変更によって減少したコードの行数規模		**	Medium
	CodeChurnAdd	変更時に追加したコードの総行数	変更により労力がかかる	**	Medium
ソースコード (20次元)	CodeChurnDel	変更時に削除したコードの総行数		**	Large
	CountLineCommentAdd	変更によって増加したコメントの行数規模	SATDコメントが書かれていれば、返済後にコメント行数が減少する	**	Small
	CountLineCommentDel	変更によって減少したコメントの行数規模		**	Medium
	CyclomaticAdd	変更によって増加したCyclomatic複雑度		**	Negligible
	CyclomaticDel	変更によって減少したCyclomatic複雑度		**	Small
	CyclomaticModifiedAdd	変更によって増加したCyclomaticModified複雑度		**	Negligible
	MaxNestingDel	変更によって減少した最大ネスト数		**	Small

※ 有意差があり効果量が小程度以上で 灰色 に塗られていない特徴量は， VIF統計量で削減された特徴量

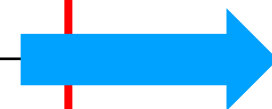
外れ値の検出と除去

- 各特徴量が正規分布を仮定できないため、箱ひげ図（四分位範囲と四分位数）を用いて外れ値を検出
 - 外れ値 $< Q1 - 1.5 \times IQR$, $Q3 + 1.5 \times IQR < \text{外れ値}$

$Q1$: 第1四分位数, $Q3$: 第3四分位数, IQR : 四分位範囲

- 外れ値を含むデータをデータセットから除外

	特徴量A	特徴量B	特徴量C
Issue1	外れ値	正常値	正常値
Issue2	正常値	外れ値	外れ値
Issue3	正常値	正常値	正常値



データを除外

評価実験：概要

- データセット（外れ値除去後）

SATD-Issue（正例）	通常のIssue（負例）
945 件	52,693 件

- モデル
 - テキスト + （報告者・プロセス・ソースコード）から1つ
 - テキスト + （報告者・プロセス・ソースコード）から2つ
 - テキスト + 報告者 + プロセス + ソースコード
- 評価方法
 - 10分割交差検証を10回試行
 - 平均Precision, Recallで精度評価

評価実験：結果

- ランダムフォレスト，ロジスティック回帰，線形SVMのアルゴリズムを利用

モデル	ランダムフォレスト		ロジスティック回帰		線形SVM	
	Precision	Recall	Precision	Recall	Precision	Recall
T+R	0.990	0.658	0.979	0.867	0.916	0.875
T+P	0.947	0.526	0.907	0.771	0.805	0.790
T+C	0.985	0.468	0.900	0.614	0.720	0.683
T+R+P	0.996	0.736	0.991	0.932	0.963	0.935
T+R+C	0.997	0.729	0.988	0.899	0.950	0.909
T+P+C	0.962	0.604	0.905	0.770	0.803	0.788
T+R+P+C	0.998	0.758	0.992	0.936	0.965	0.938

T：テキスト， R：報告者， P：プロセス， C：ソースコード

評価実験：結果

- ランダムフォレスト，ロジスティック回帰，線形SVMのアルゴリズムを利用

モデル	ランダムフォレスト		ロジスティック回帰		線形SVM	
ソースコードの特徴量を加えていなくても同等の精度 → ソースコードの特徴量を使わなくてもよい可能性						
T+C	0.985	0.468	0.900	0.614	0.720	0.683
T+R+P	0.996	0.736	0.991	0.932	0.963	0.935
T+R+C	0.997	0.729	0.988	0.899	0.950	0.909
T+P+C	0.962	0.604	0.905	0.770	0.803	0.788
T+R+P+C	0.998	0.758	0.992	0.936	0.965	0.938

T：テキスト， R：報告者， P：プロセス， C：ソースコード

補足資料

特徴量一覧

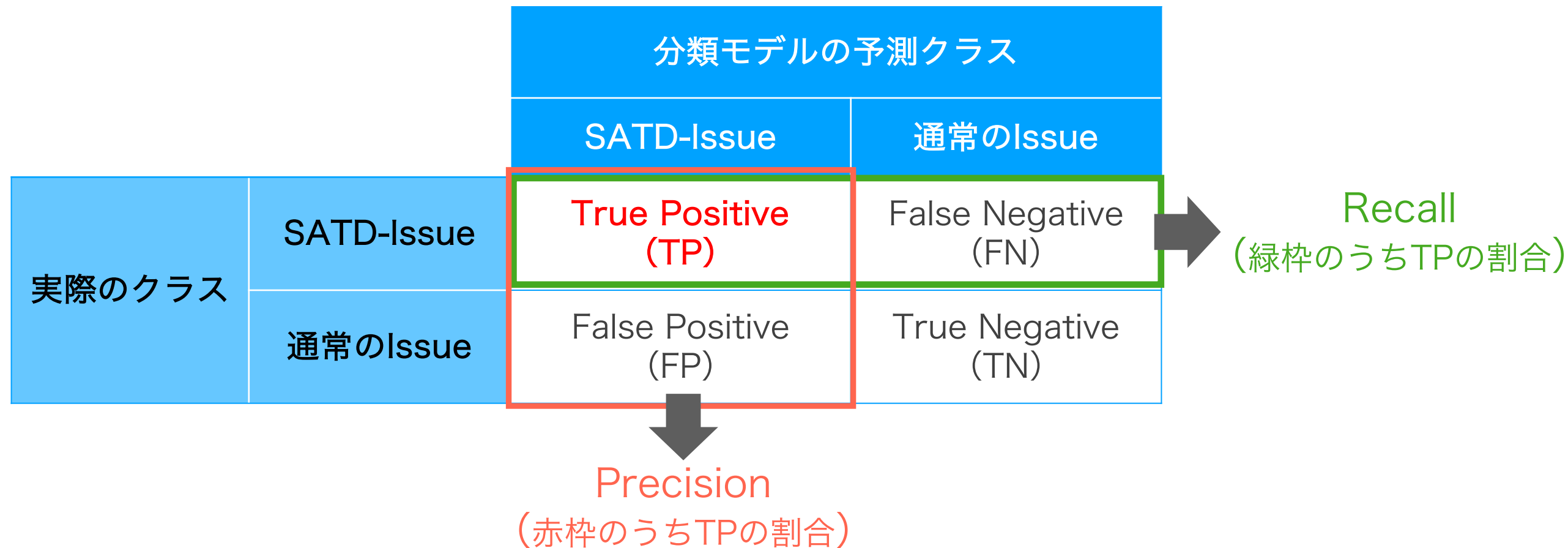
灰色：外れ値除去とモデルの学習に利用している特徴量

有意差：- (0.05 < p), * (p < 0.05), ** (p < 0.01)

効果量：Negligible (効果なし), Small (小), Medium (中), Large (大)

Category	Metrics	Description	Hypothesis	Significant	Effect Size
テキスト (22,678次元)	Title	課題票のタイトル (4,544次元)	記述される内容が異なる		
	Description	課題票の本文 (17,134次元)			
報告者 (4次元)	Experience	報告者がアカウントを登録してから課題票Iを報告するまでの時間	コードの品質の良し悪しを判断するには技術的な経験を必要とする	**	Medium
	rCommitNum	報告者が課題票Iを報告するまでにおこなったコミットの数		**	Large
	OpenIssueNum	報告者が課題票Iを報告するまでにおこなった課題報告の数		**	Large
	Type	報告者の種別 (1: プロジェクトメンバー, 2: コントリビュータ, 0: その他)	プロジェクト外部の開発者に比べて, 内部の開発者の方がコードの品質を重視する	**	Small
プロセス (9次元)	TitleLen	課題票のタイトルに含まれる単語の数	問題が抽象的に報告されるため, 単語の数が少なくなる	**	Negligible
	DescriptionLen	課題票の本文に含まれる単語の数		**	Medium
	AssigneeNum	課題票の担当に割り当てられている開発者の数	品質管理はコア開発者がおこなうため, 関われる人数が少ない	-	Negligible
	ParticipantNum	課題票にコメントしている開発者の数 (担当の開発者は除く)		**	Medium
	CommentNum	課題票のコメントの数	改善方法を決めるために多くの議論がされる	**	Small
	SelfAssign	報告者自身が課題票の担当になっているか	間に合わせて実装したコードを後から自分で変更する	**	Small
	pCommitNum	課題票解決のためにおこなわれたコミットの数		**	Large
	ChangeFileNum	課題票解決のために変更したファイルの数	変更が複雑になる, 変更に労力がかかる	**	Medium
	ResolutionTime	課題票解決にかかった時間		**	Medium
ソースコード (20次元)	CountDeclClassAdd	変更によって増加したクラスの数	リファクタリングにより抽象化される	**	Small
	CountDeclClassDel	変更によって減少したクラスの数		**	Small
	CountDeclFunctionAdd	変更によって増加した関数の数		**	Small
	CountDeclFunctionDel	変更によって減少した関数の数		**	Small
	CountLineCodeAdd	変更によって増加したコードの行数規模	リファクタリングによりファイル規模 (LOC) が減少する	**	Small
	CountLineCodeDel	変更によって減少したコードの行数規模		**	Medium
	CodeChurnAdd	変更時に追加したコードの総行数	変更により労力がかかる	**	Medium
	CodeChurnDel	変更時に削除したコードの総行数		**	Large
	CountLineCommentAdd	変更によって増加したコメントの行数規模	SATDコメントが書かれていれば, 返済後にコメント行数が減少する	**	Small
	CountLineCommentDel	変更によって減少したコメントの行数規模		**	Medium
	CyclomaticAdd	変更によって増加したCyclomatic複雑度	リファクタリングにより複雑度が下がる	**	Negligible
	CyclomaticDel	変更によって減少したCyclomatic複雑度		**	Small
	CyclomaticModifiedAdd	変更によって増加したCyclomaticModified複雑度		**	Negligible
	CyclomaticModifiedDel	変更によって減少したCyclomaticModified複雑度		**	Small
	CyclomaticStrictAdd	変更によって増加したCyclomaticStrict複雑度		**	Negligible
	CyclomaticStrictDel	変更によって減少したCyclomaticStrict複雑度		**	Small
	EssentialAdd	変更によって増加したEssential複雑度		**	Small
	EssentialDel	変更によって減少したEssential複雑度		**	Small
	MaxNestingAdd	変更によって増加した最大ネスト数		**	Small
	MaxNestingDel	変更によって減少した最大ネスト数		**	Small

評価指標



$Precision = \frac{TP}{TP+FP}$... SATD-Issueと分類して、実際にSATD-Issueであった割合

$Recall = \frac{TP}{TP+FN}$... 実際のSATD-Issueのうち、分類でSATD-Issueを特定した割合

microsoft/vscode

model	RF_precision	RF_recall	RF_f1	LR_precision	LR_recall	LR_f1	SVM_precision	SVM_recall	SVM_f1
T+P+C	0.963	0.594	0.734	0.918	0.777	0.841	0.834	0.805	0.819
T+C	0.985	0.431	0.598	0.857	0.57	0.683	0.698	0.648	0.67
T+P	0.954	0.531	0.681	0.918	0.779	0.842	0.828	0.807	0.816
T+R	0.995	0.566	0.720	0.981	0.871	0.922	0.932	0.881	0.905
T+R+C	0.999	0.658	0.792	0.989	0.897	0.940	0.958	0.906	0.931
T+R+P	0.995	0.678	0.806	0.993	0.934	0.963	0.975	0.940	0.957
T+R+P+C	0.998	0.708	0.828	0.993	0.934	0.962	0.974	0.939	0.956
T	0.896	0.218	0.349	0.741	0.476	0.578	0.569	0.554	0.561

特徴量を加えると精度向上が見られる
テキストのみの精度もそこそこ高い