

GitHub 上のソフトウェア開発のためのフロー推薦 手法

プロジェクトマネジメントコース
ソフトウェア開発管理グループ
矢吹研究室
1242132
若月 純

謝辞

本研究を進めるにあたり，矢吹研究室矢吹太郎准教授には，多くの時間をご指導にさいて頂きました．また，先行研究を行った小野寺航己先輩をはじめ矢吹研究室の皆様には，多くの知識や示唆を頂きました．協力していただいた皆様に感謝の気持ちと御礼を申し上げます．

目次

第 1 章	序論	4
第 2 章	背景	5
2.1	バージョン管理システム	5
2.2	GitHub	8
2.3	GitHub 用語	9
2.4	GitHub を用いた開発フロー	11
2.5	GitHub を用いた開発フローの選択基準	19
第 3 章	目的	21
第 4 章	手法	22
4.1	調査準備	22
4.2	調査対象	23
4.3	調査方法	25
4.4	分析ツール	34
4.5	分析データ準備	36
4.6	階層的クラスター分析方法	37
4.7	決定木分析方法	38
4.8	決定木性能測定方法	38
第 5 章	結果	40
5.1	階層的クラスター分析結果	40
5.2	決定木分析結果	41
5.3	決定木性能測定結果	45
第 6 章	考察	60
6.1	すべてのデータで行った場合	60
6.2	データをランダムに並び替え，2 つに分けた場合	60
6.3	コントロールできる要因のみかつ，成功しているプロジェクトの場合	61
6.4	精度と再現率	61
第 7 章	結論	63

参考文献

64

第 1 章

序論

当研究は，GitHub を用いた開発フローの判別分析を行う．開発フローとは，開発の手順，ルールである．

GitHub を用いた開発フローは 13 種類ある．それぞれの開発フローには，メリット・デメリットがある．そのため，適切な開発フローを選択できる基準が必要である．

先行研究は，プロジェクトをリスクにより分類していた．そのため，選択基準が定性的になっていた．例を 1 つあげる．メンバのスキルが高く，大規模なプロジェクトの場合，フローが自動化されているイストフローが選択される [1]．

そこで当研究は，定量的な選択基準を提供することを目指す．

GitHub 上のプロジェクトの性質を調査し，決定木分析を行う．そうすることで，同一の開発フローを選択しているプロジェクトの共通項を見つけることができる．共通項を見つけることで，開発フローを選択する基準を割り出せると考える．

第 2 章

背景

ソフトウェア開発では、複数のメンバが同時に開発する場合がある。そのため、様々な問題が発生する。課題をチーム間で適切に共有できず、進捗が見えにくくなったり、複数の人たちが 1 つの製品のソースコードを編集するため、開発内容が競合したりすることもあります。さらに複数の人たちが関わることによって、コードの質を均一化することも難しくなりますし、製品のコードの全容を把握することも難しくなります [2]。

このような問題を解決するため、バージョン管理システムを用いる。バージョン管理システムとは、変更履歴を管理するシステムのことである。

バージョン管理システムの種類とトレンドを次に記述する。

2.1 バージョン管理システム

バージョン管理システムには、CVS、Apache Subversion、Git 等がある。Apache Subversion は、以下 subversion と記述する。

CVS とは、Concurrent Versions System の略称で、並行バージョン管理システムを意味します。1990 年に開発された比較的歴史の古いバージョン管理ツールです。こちらは集中型のバージョン管理システムになります [3]。

Apache Subversion とは、多くの開発現場で使われていた CVS の問題点を改善するために作られたツールです。2000 年に開発され、操作が CVS と似ていることから、CVS から Subversion に乗り換えるユーザが多くいました。CVS と同じく集中型バージョン管理システムです [3]。

Git とは、2005 年に開発された分散型のバージョン管理ツールです。Linux カーネルのソースコード管理のために開発されたもので、今では多くの開発者に使われている人気のバージョン管理ツールです [3]。

図 2.1 と図 2.2 は、GoogleTrend を用いて描かれたものである。それぞれの検索の割合を示している。

図 2.1 と図 2.2 は、2004 年 1 月 4 日から 2016 年 1 月 23 日まで、コンピュータ、電化製品、を対象としている。図 2.1 は、すべての世界を対象としている。図 2.2 は、日本のみを対象としている。

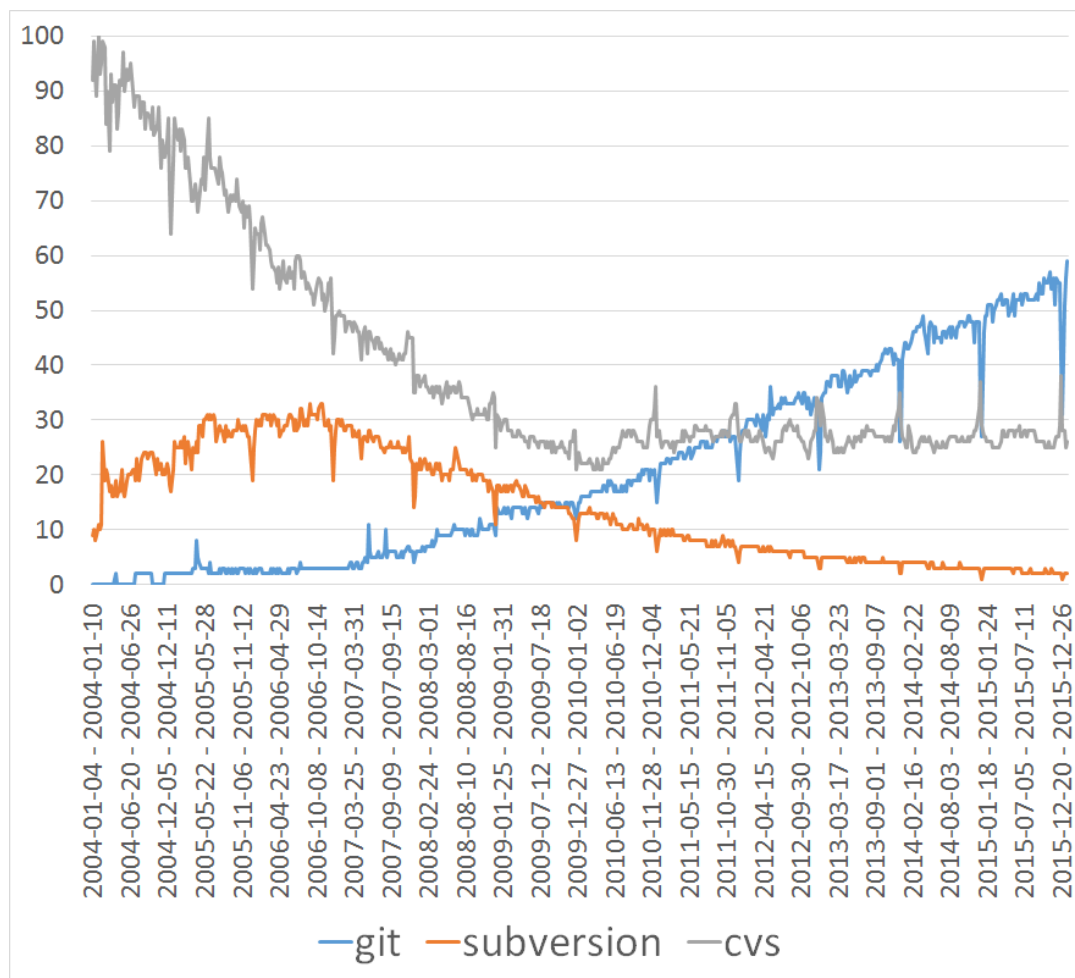


図 2.1 世界でのバージョン管理システムのトレンド

図 2.1 より，世界では，最も Git がトレンドになっていることがわかる．2004 年は CVS が最も人気が高かった．しかし，subversion の台頭により，2004 年以降人気が落ちていっていることがわかる．

Git が台頭してきたのは，2005 年ごろである．このころから，人気は上昇を続ける一方である．2009 年に，Git は，subversion の人気を超えた．2011 年に，Git は，CVS の人気を超えた．

こうして，2015 年は，最も Git が人気になっている．

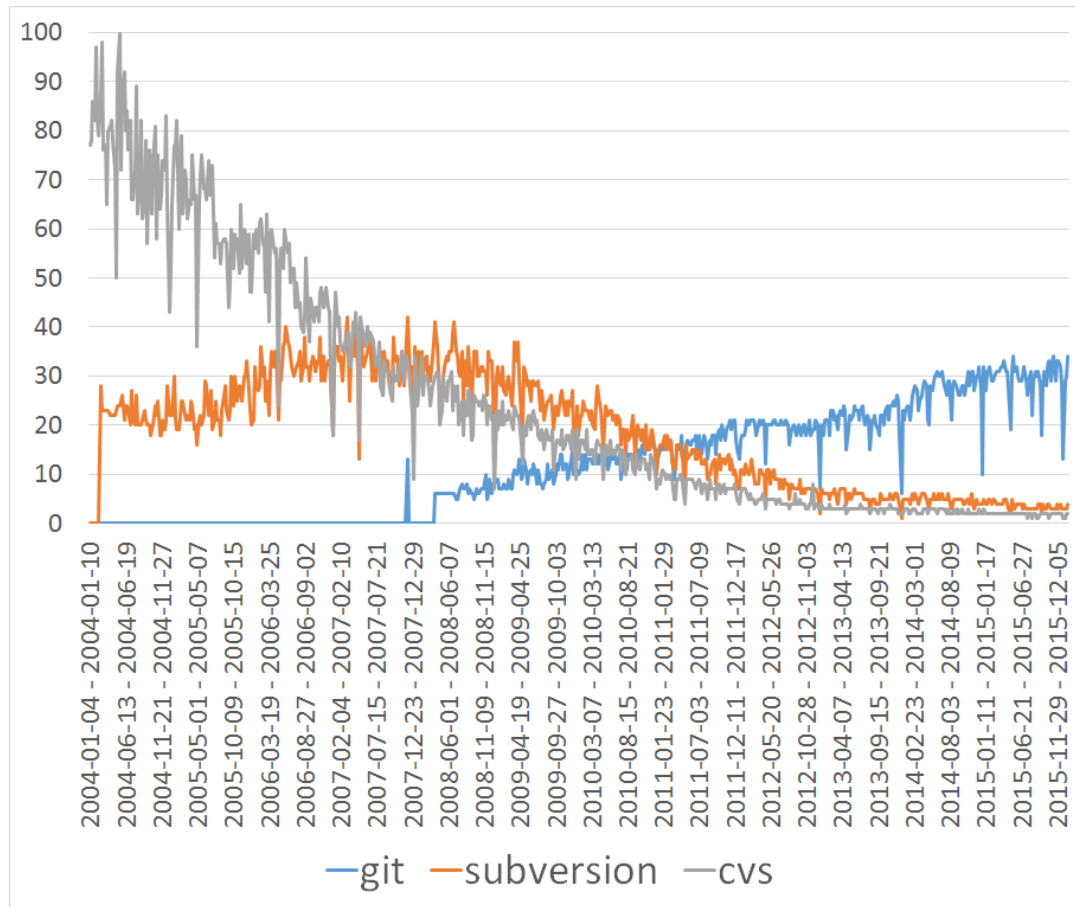


図 2.2 日本でのバージョン管理システムのトレンド

図 2.2 より，日本でも，最も Git がトレンドになっていることがわかる．2004 年は CVS が最も人気が高かった．しかし，subversion の台頭により，2004 年以降人気が落ちていっていることがわかる．

Git が台頭してきたのは，2008 年ごろである．このころから，人気は上昇を続ける一方である．2010 年に，Git は，CVS の人気を超えた．2011 年に，Git は，subversion の人気を超えた．

こうして，2015 年は，最も Git が人気になっている．

この世界と日本の状況を踏まえ，バージョン管理システムのトレンドについて考察する．まず，世界と日本の状況を比較する．

図 2.1 と図 2.2 の技術が伸びを見せる年式に注目する．世界では，Git が 2005 年に台頭してきた．しかし，日本では Git が台頭してきたのは 2008 年からである．

ここから，日本に新しい技術が入ってくるのは，世界よりも遅いことが言える．これは，日本が海外から新しい技術を輸入しているためだと考えられる．海外の技術は，その土地の言葉で書かれている．そのため，翻訳する必要があるので，技術を輸入し，定着するまでに 3 年かかっている．

また，日本独自の開発手順があることが言える．世界のトレンドでは，Git，CVS，subversion の順である．しかし，日本のトレンドでは，Git，subversion，CVS の順である．

以上の調査により，最も人気があるバージョン開発システムは，Git であることがわかった．

そのため，本研究では，バージョン管理システムの Git を提供するサービスの GitHub を用いる．GitHub についての詳しい説明を，次に記述する．

2.2 GitHub

GitHub とは，Git リポジトリのホスティング機能をもつ．また，スローガンに Social-Coding をかかげているように，複数人で共同開発を行いやすくするための機能を提供している．機能とは，トラッキングや管理を行うための Issue，ソースコードの差分を論議するための Pull Request 等がある．

GitHub は 2008 年のサービス開始以来，年率 100 % という成長率で登録ユーザー数やレポジトリ数を伸ばしてきて，オープンソースの世界ではデファクトのプラットフォームとなっている．2015 年 6 月現在，GitHub の登録ユーザー数は 970 万，レポジトリ数は 2330 万．最近では Microsoft や Oracle といったトラディショナルな IT 企業も GitHub にレポジトリを用意するようになってきているし，広く知られたメジャーなオープンソース製品の多くが GitHub をプロジェクトのホスト先に選ぶのがここ数年のトレンドだ [4]．

主な具体例として，アップルが Swift の OSS 化を GitHub で行った事例がある．Apple は，iOS や OS X の開発言語として提供してきたプログラミング言語の Swift をオープンソースとして公開した．今回オープンソース化されたのは，ソースコードを実行形式に変換するコンパイラと，プログラムの基本機能をまとめた標準ライブラリなどである．従来は，iOS/OS X 向けのソフトウェアしか開発できなかったが，Linux への移植版も公開している．そして，Apple が Swift を公開した場所も GitHub であった．Chacon 氏は，今回の Swift のオープンソース化の特徴として，コミュニティとの連携を強調した．Swift に対するプルリクエストがすでに社外から 500 件以上も寄せられており，そのうち 349 件が統合済みだという．中には，誤字の指摘なども含まれるが，Apple の開発ツール部門のシニアディレクターで，Swift の主要な開発者である Chris Lattner 氏が「小さな改善でも，多くの人が参加するきっかけになる」とツイートしていると紹介した [5]．

日本でも，セミナーや，行政活動に GitHub を用いる等，盛り上がりをみせている．具体例として，和歌山県が自治体として最初に GitHub 公式アカウントを取得した例をあげる．和歌山県は自治体としては初めて公共アカウントを取得し，ウェブサイトで公開した情報を GitHub でも公開するなど実際に活用を進めている．今回の訪問は和歌山県庁での活用に注目しているという GitHub 側が，現場の話を直接知り，今後の支援につなげたいとの要望から実現したもので，知事との対談に先駆けて，庁内のオープンデータ化を担当する職員や活動を支援する関係者らとも意見を交換した [6]．

また，初心者向けのイベントが豊富なことも，人気の一つである．初心者から参加できる GitHub 習得イベント「GitHub Patchwork」が神戸で開催された．GitHub Patchwork は，GitHub を基礎から学べるオンライン教材「Git-it」をみんなと一緒に学ぶというイベント．2 年前にサンフランシスコの GitHub 本社で実施されたのをきっかけに世界各地で開催されるようになり，現在 14 カ国で 44 件のイベントが実施されている．参加者からは「以前か

ら GitHub に興味があったが、なかなか勉強する機会がなかったため、今回の参加をきっかけに活用したい」という声が多く聞かれた [7]。

しかし、盛り上がりを見せている GitHub にも、問題点はある。GitHub でホスティングされたプロジェクトで作業する開発者が、自分たちが「無視され」、サポートが得られないとの苦情を公開書簡にて表明している。この公開書簡は GitHub の Web サイトの管理とサポートチャンネルへの不満を綴ったもので、1000 人以上の開発者が賛同している。書簡では、「もし GitHub そのものがオープンソースだったら、われわれはコミュニティとしてこれらの機能を実装する。われわれはこういうことを得意としているんだ!」と記している。GitHub の代表者は米 ZDNet に対し、「オープンソースは GitHub にとって極めて重要で、今回のフィードバックを真剣に受け止めている。議論されたイニシアティブの幾つかに取り組んでおり、オープンソースのメンテナーたちと積極的な関係を持ち、彼らのコミュニティにとって GitHub が素晴らしい体験となるようにしたい」と語った [8]。

ここまで、GitHub が注目を受けている背景について紹介した。様々なコミュニティが GitHub を使っている。

GitHub 特有の開発を補助する機能が多数あり、機能を一部抜粋して、説明を記述する。

2.3 GitHub 用語

2.3.1 リポジトリ

ファイルやディレクトリの状態を記録する場所。

2.3.2 リモートリポジトリ

手元に置いてあるローカルなリポジトリ以外の、ネット上に置かれたリポジトリのこと。

2.3.3 commit

ファイルやディレクトリの変更をリポジトリに記録する機能である。

2.3.4 clone

ネット上にあるリポジトリをローカルにコピーする機能である。

2.3.5 Origin

clone 元のリモートリポジトリのこと。

2.3.6 Push

リモートリポジトリに自分の変更履歴がアップロードされ、リモートリポジトリ内の変更履歴がローカルリポジトリの変更履歴と同じ状態にする機能である。

2.3.7 branch

履歴の流れを分岐して記録していくためのもの。分岐したブランチは、他のブランチの影響を受けないため、同じリポジトリ中で複数の変更を同時に進めていける機能である。

2.3.8 pull

リモートリポジトリから最新の変更履歴をダウンロードしてきて、自分のローカルリポジトリにその内容を取り込む機能である。

2.3.9 Pull Request

相手に対して自分の変更を pull してもらうように要求する機能である。

2.3.10 Revert

ステージングエリアに追加した変更を取り消す機能である。

2.3.11 タグ

コミットを参照しやすくするために、わかりやすい名前を付ける機能である。

2.3.12 Label

自由に作成でき、Issue をフィルタリングできる機能である。

2.3.13 Merge

当該ブランチに対して別のブランチの差分を取り込むことである。

2.3.14 Fork

GitHub のサービスで、相手のリポジトリを自分のリポジトリとしてコピー・保持できる機能ある。

2.3.15 Issue

ソフトウェア開発におけるバグや議論などをトラッキングして管理するために発行する。

2.3.16 デプロイ

ソフトウェアの分野で、開発したソフトウェアを利用できるように実際の運用環境に展開する。

2.3.17 リリース

プロセスを次の段階に進めることを認める機能である。

2.3.18 Watch

リポジトリに関する情報を Notifications に表示する機能である。

2.3.19 Star

リスト一覧からリポジトリを探すことが出来るようにする機能である。また、注目度を表す指標にもなる。

2.3.20 Fork

GitHub 側にある特定のリポジトリを自分のアカウント以下のリポジトリに複製する機能である。

2.3.21 人数

開発人数のことである。ここでは、Origin リポジトリにコミットした人数のことを示す。

2.3.22 MileStone

やるべきタスクの管理に Issue を用いることができるようにする機能である。

2.3.23 Wiki

簡単な記法によってドキュメントを作成、編集するための機能である。

2.4 GitHub を用いた開発フロー

GitHub を使用する手順を開発フローと呼ぶ。開発フローを網羅的に調査した先行研究がある。先行研究で調査され、本研究で用いた開発フローと、選択基準を以下に記述する。

2.4.1 GitHub フロー



図 2.3 GitHub フロー図 出典：[1]p17

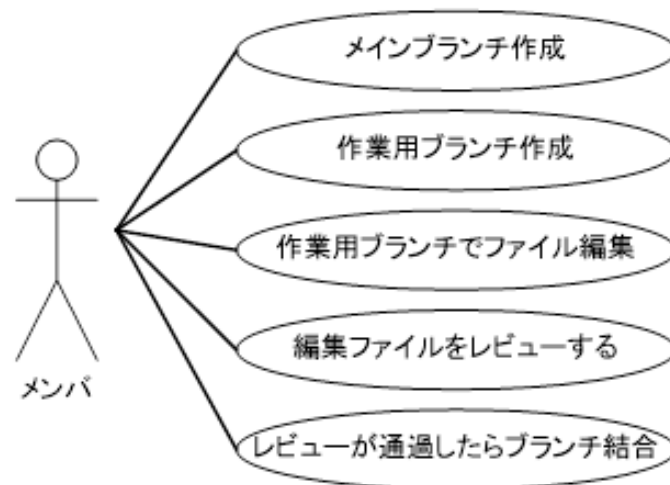


図 2.4 GitHub フローユースケース図 出典：[1]p18

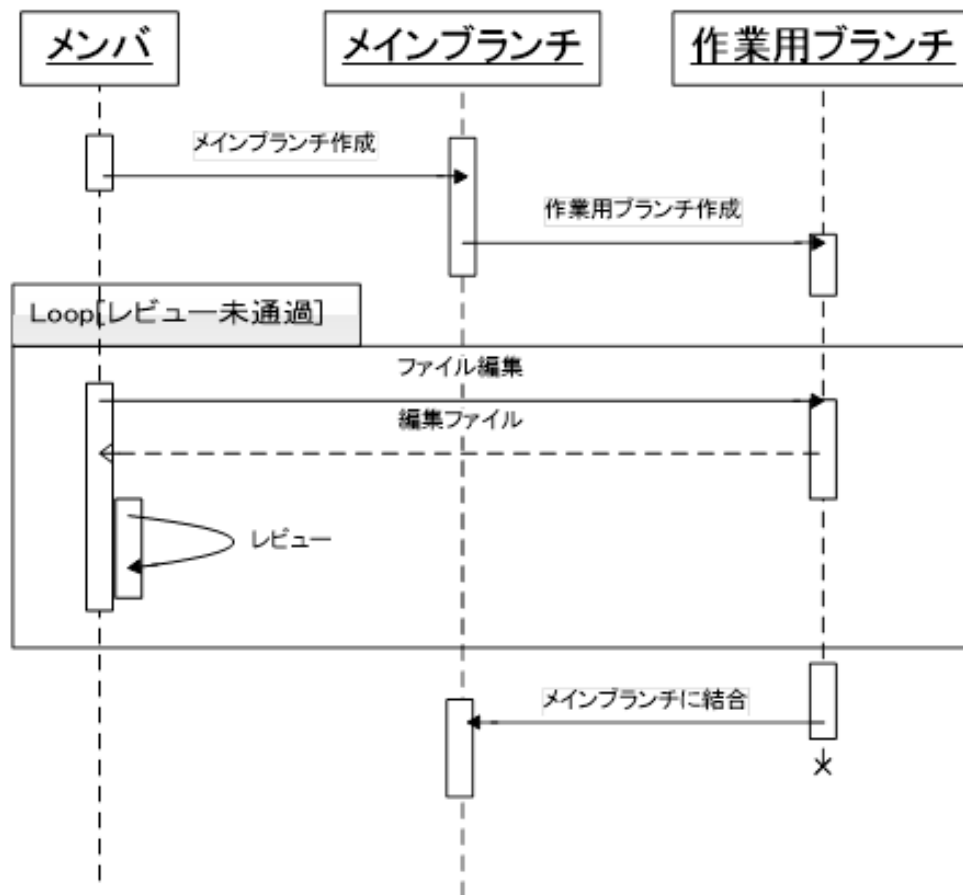


図 2.5 GitHub フローシーケンス図 出典:[1]p18

GitHub フローは GitHub 社が実践しているシンプルなワークフローである。基本的には特定の作業をするブランチを作成するだけなので、作業を始めてデプロイするまでの過程がとてもシンプルです。これはワークフローを実施するまでの学習コストを抑えられるという利点があります。さらに大きな利点として、シンプルであるからこと、多くの開発者がすばやく行うことを可能にします。そして、小さな変更などにも柔軟に対処できるようになります [9]。

2.4.2 Git フロー

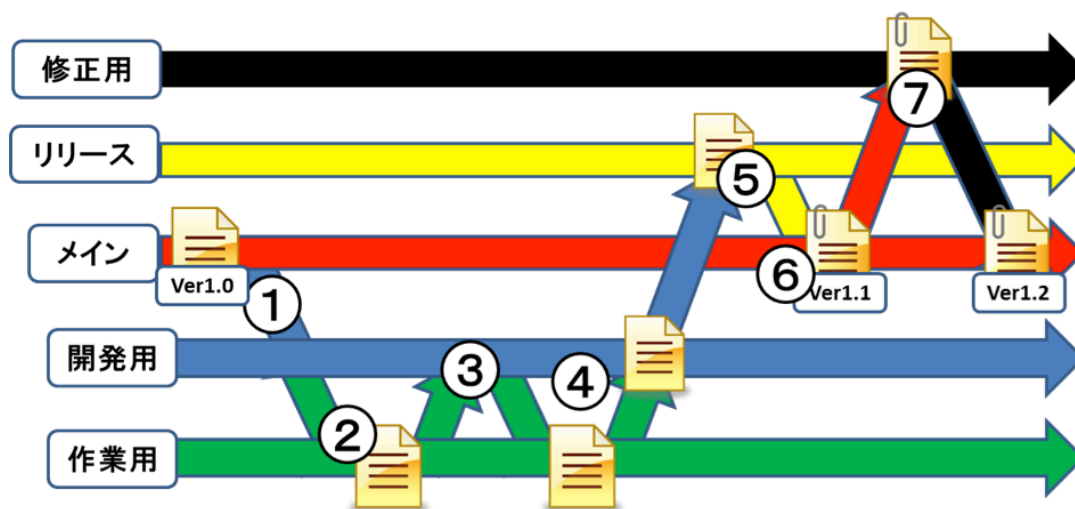


図 2.6 Git フロー図 出典：[1]p20

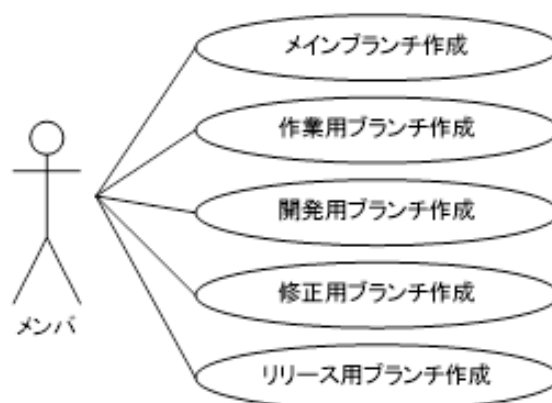


図 2.7 Git フローユースケース図 出典：[1]p20

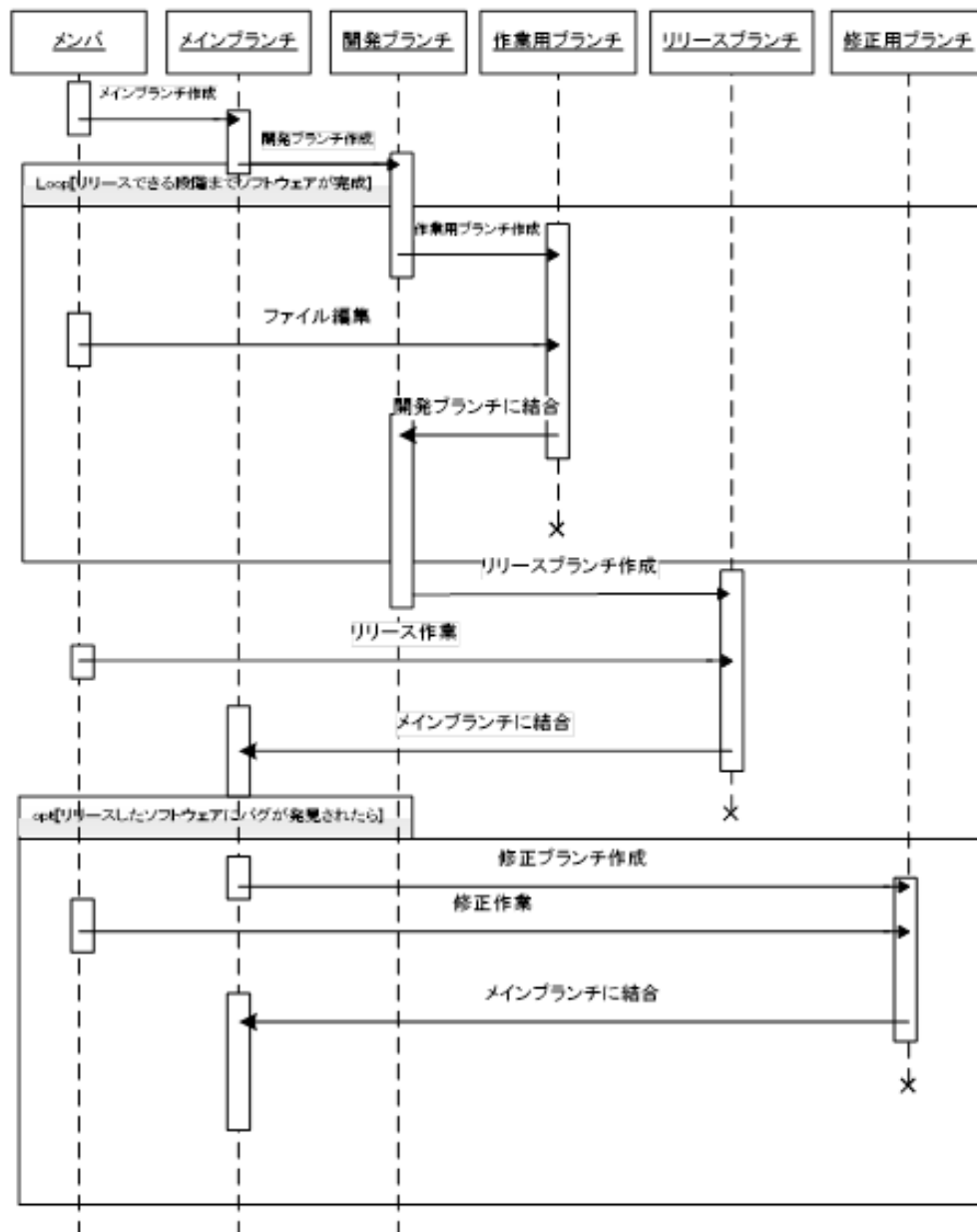


図 2.8 Git フローシーケンス図 出典：[1]p20

Git フローはリリース中心の開発スタイルである。この開発フローは、それぞれのブランチがコードの状態を表している。ソフトウェアのリリースを管理するリリースマネージャーなどが存在し、リリースを中心としたソフトウェア開発に向いている。しかし、覚えるブランチの状態が多く、開発フロー全体を事前に学習する必要がある。そのため、git フローなどのツールのサポートを受けることにより、強制的にフローからはずれない工夫が必要である [9]。

2.4.3 GitLab フロー

GitLab フローは、GitHub フローをベースにしている。

特徴は、branch に stable branch を採用している点である。GitLab branch とは、安定状態の branch を作ることで、バグが新たに入り込むリスクを減らす効果がある。GitLab branch は、できるだけ最新の master ブランチから分岐することで、バグを新たに入り込むリスクを減らせる [4]。

2.4.4 日本 CAW フロー

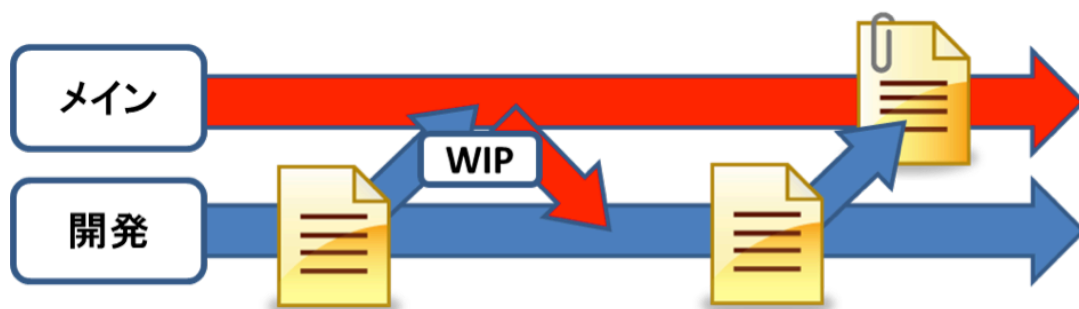


図 2.9 日本 CAW フロー図 出典：[1]p23

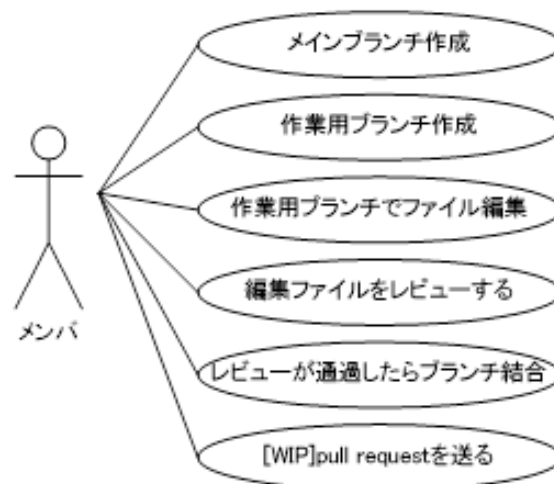


図 2.10 日本 CAW フローユースケース図 出典：[1]p24

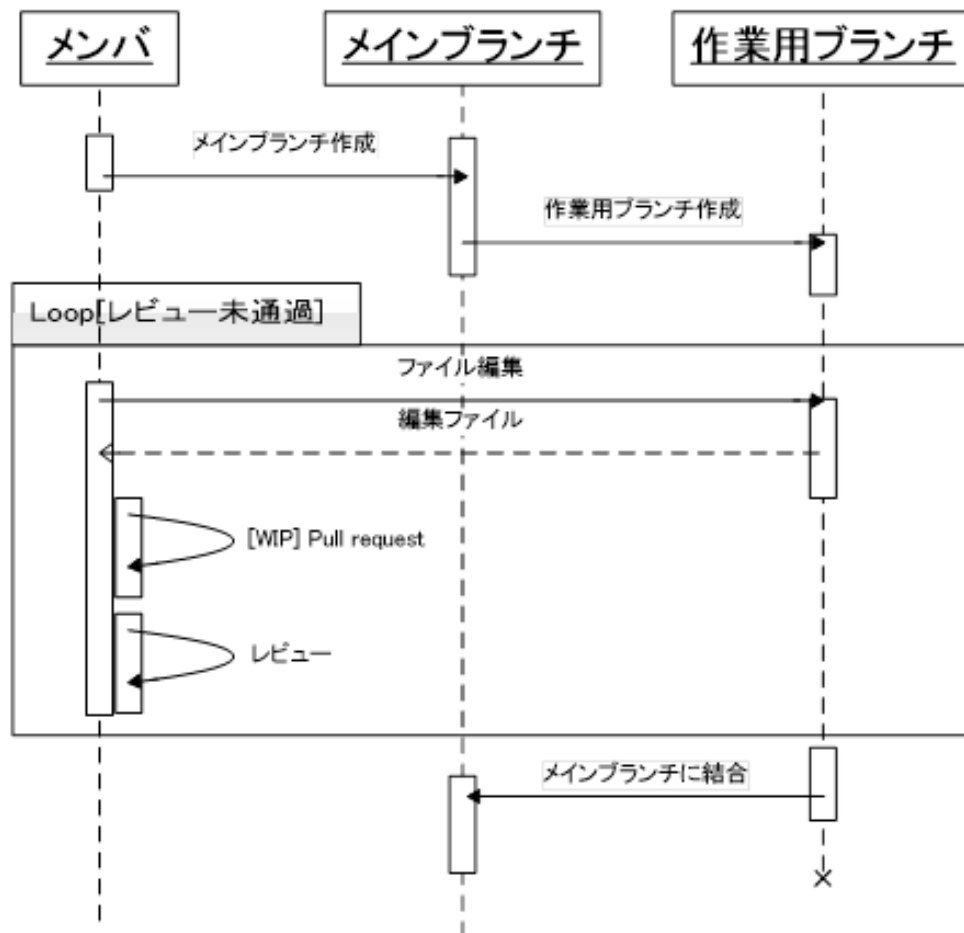


図 2.11 日本 CAW フローシーケンス図 出典：[1]p24

日本 CAW フローは、日本 CAW 株式会社が採用しているフローである。このフローは、GitHub フローをベースにして Pull Request を活用したスタイルを採用している。特徴は、WIP PR を採用している点である。WIP PR (Work In Progress Pull Request) とは、実装の仕方や、コードの設計など、コードに紐づく議論をしやすいように用いられる。Pull Request 作成時に頭に [WIP] を入れる。[WIP] がついた Pull Request はマージされず、議論や確認のために参照される。作成者は議論や確認が済んだら Close する、といったフローである [10]。

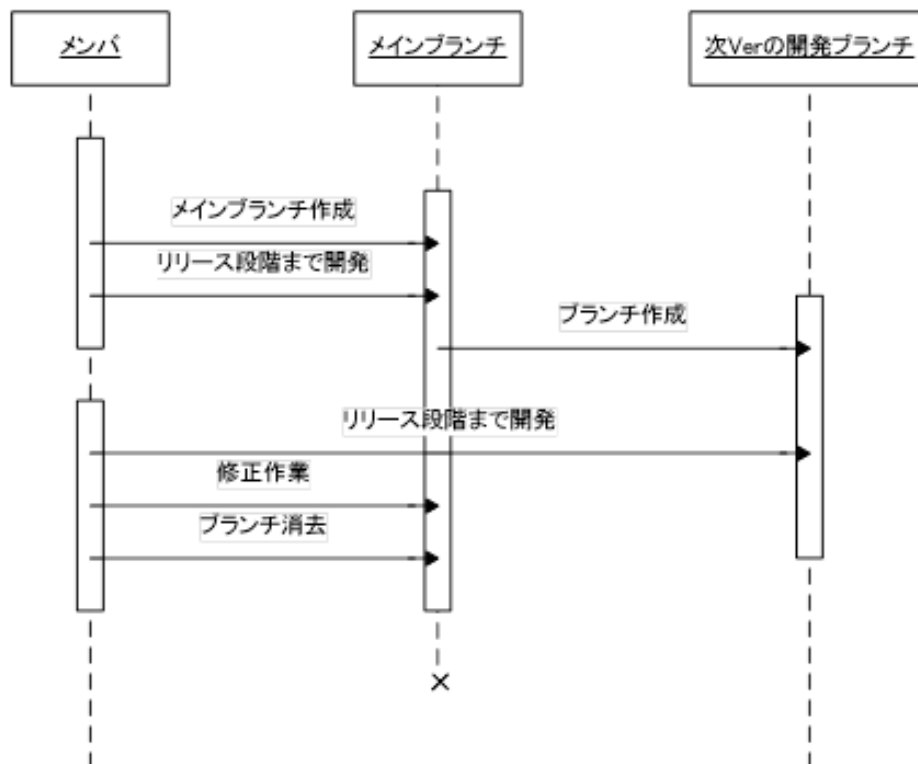


図 2.14 LINE フローシーケンス図 出典：[1]p32

LINE フローは Pull Request などを活用して行われる開発フローである．常に deploy 可能なバージョン毎の master を持っているという形で運用している．それぞれがリリース可能なブランチで，開発状況によっては下位のバージョンのコミットが上位のバージョンのコミットより新しい場合がある．LINE の IOS アプリではバージョン毎に提供する機能や修正を決めていて，QA 対象の切り分けがしやすいなどのメリットもあり，1 つの master に commit を繋げていく形では運用していない．現在開発中の最も下位のバージョンの修正がリリースされたら，そのブランチを上位のブランチにマージして下位のブランチを消していく流れで行われる [11]．

2.5 GitHub を用いた開発フローの選択基準

それぞれ異なったリスクがあるため，最適な開発フローを選択する基準をリスクの面から分類された．研究の結果が表 2.1 である．プロジェクトの特徴から最適な開発フローが選択できるようになった．

表 2.1 プロジェクトと開発フロー出典：[1]p62

プロジェクトの特徴	開発フローの特徴	該当する開発フロー
メンバのスキルが高い 大規模なプロジェクト	フローが自動化されている	フィヨルドフロー イストフロー
参加メンバが多い	複数のリポジトリを使用する	Aming フロー サイボウズフロー 矢吹研フロー 矢吹研フロー
アジャイル型のソフトウェア開発	アジャイル開発のような 流れが可能	Git フロー キャスレーフロー
GitHub を今まで導入した経験が無い プロジェクト	使用するブランチが少ない	GitHub フロー 日本 CAW フロー はてなブログフロー
ソフトウェアに何を盛り込むのかが明確	使用済みブランチを破棄	ラクスルフロー LINE フロー

しかし、メンバのスキルが高い、大規模なプロジェクト等、定性的な表現が多い。

そこで当研究では、選択する基準を定量的に分けられるようにすることを目指す。具体例を二つあげる。開発人数が 5 人の場合かつ、言語が Ruby の場合は、GitHub フローが最適である。開発人数が 15 人の場合かつ、言語が Java の場合は、Git フローが最適である、そうすることで、既存の基準より、適切な開発フローを選択できるようになると考える。

第 3 章

目的

GitHub を用いたソフトウェア開発プロジェクトの性質において，適切な開発フローを選択できるようにするための基準を提供する．

第 4 章

手法

調査準備，調査対象，調査方法，分析準備，分析方法について記述する．

環境は，2 種類用いた．一つ目は，OS X Yosemite バージョン 15.15.5 である．二つ目は，Ubuntu である．

4.1 調査準備

調査は，GitHubAPI を用いる．そのため，GitHubAPI の準備について，記述する．

ここで行う準備の環境は，Ubuntu を想定している．

curl のインストール

```
sudo apt-get install curl
```

ログイン情報入力を省略

```
echo ' ユーザ名:パスワード' > github.passwd  
chmod 600 github.passwd
```

Python と HTTP アクセスのための requests のインストール

```
sudo apt-get install python python-setuptools  
sudo easy_install requests
```

api.py のダウンロード

```
curl -s -u $(cat github.passwd)  
https://raw.githubusercontent.com/taroyabuki/yabukilab/master/library/github/api.py  
> api.py
```

api.py について

```
#!/usr/bin/python  
# coding: UTF-8
```

```
import sys, json, requests

#GitHub のログイン情報をファイルから取得する
#TODO: パスワードに「:」を使っているとダメ
tmp = open('github.passwd').readline().rstrip('\n').split(':');
username = tmp[0]
password = tmp[1]
#print >> sys.stderr, username, password

#API の URL はコマンドライン引数で与える
url = sys.argv[1]

count = 0
while (url is not None):
    print >> sys.stderr, url
    r = requests.get(url, auth=(username, password))
    print >> sys.stderr, r.headers['status'],
    items = r.json()['items'] if 'items' in r.json() else r.json()
    for item in items:
        count = count + 1
        print json.dumps(item)
    if (r.links.has_key('next')):
        url = r.links['next']['url']
    else:
        url = None
    print >> sys.stderr, count, 'items'
```

4.2 調査対象

調査対象プロジェクトと調査対象指標を記述する .

4.2.1 調査対象プロジェクト

調査するユーザ名とプロジェクト名を記載する .

表 4.1 調査対象プロジェクト

ユーザ名	リポジトリ名
zedapp	zed
LearnBoost	stylus
spine	spine
sirjuddington	SLADE
sass	sass
Polymer	polymer
play	play
ossec	ossec-hids
NTU-CCSP	ntu-ccsp.github.io
neovim	neovim
aol	moloch
rackerlabs	mimic
melonjs	melonJS
nasa	mct
waysome	libreset
knockout	knockout
Knplabs	FriendlyContexts
freifunkMUC	freifunkmuc.github.io
admc	flex-pilot-x
dfm	emcee
adamhjk	dynect_rest
ashesi-SE	datasaver
pyca	cryptography
karpathy	convnetjs
tlycken	Contour.jl
ellisonleao	clumsy-bird
mozbrick	brick
tyoshii	bms
sampsyo	beets
openstf	adbkit
CyberAgent	android-gpuimage

4.2.2 調査対象指標

調査する指標は ,Watch 数 ,Star 数 ,Fork 数 ,commit 数 ,branch 数 ,Release 数 ,人数 ,言語 ,行数 ,ファイル数 ,バイト数 ,OpenIssues 数 ,ClosedIssues 数 ,Issues 数 ,OpenPull Request 数 ,ClosedPull Request 数 ,Pull Request 数 ,Label 数 ,OpenMileStone 数 ,ClosedMileStone 数 ,MileStone 数 ,Wiki 数 ,言語 ,日数 ,開発フローである .

4.3 調査方法

4.3.1 Watch 数 , Star 数 , Fork 数 , commit 数 , branch 数 , Release 数 , 人数の調査

調査は , 手動で行う .

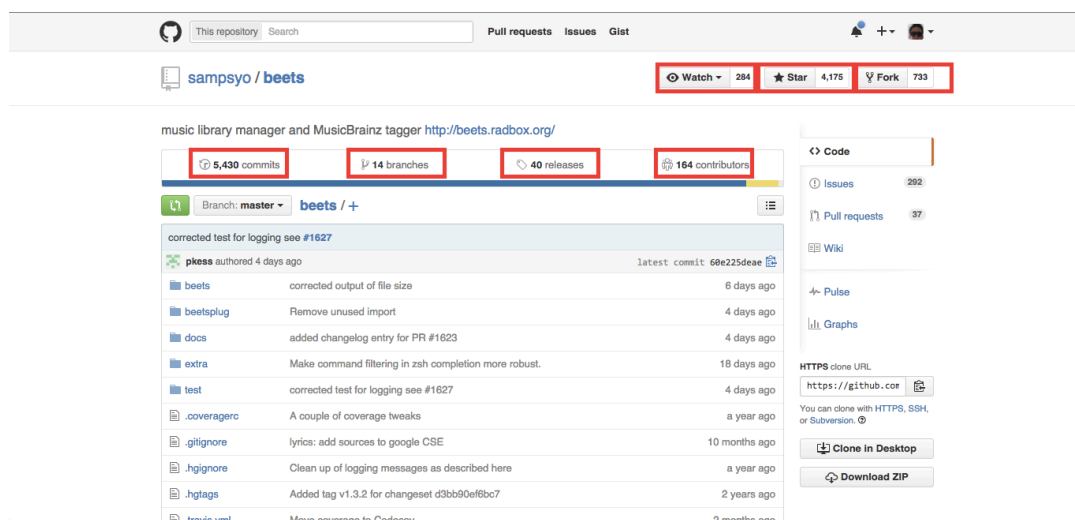


図 4.1 Watch 数 , Star 数 , Fork 数 , commit 数 , branch 数 , Release 数 , 人数の調査

図 4.1 は Watch 数 , Star 数 , Fork 数 , commit 数 , branch 数 , Release 数 , 人数の調査画面である . commits が commit 数 , branches が branch 数 , releases が Release 数 , contributors が人数である . 赤枠に囲まれている数値を取得する . 所得した数値は , "データ一覧.csv" に保存する .

4.3.2 言語の調査

調査は , 手動で行う .

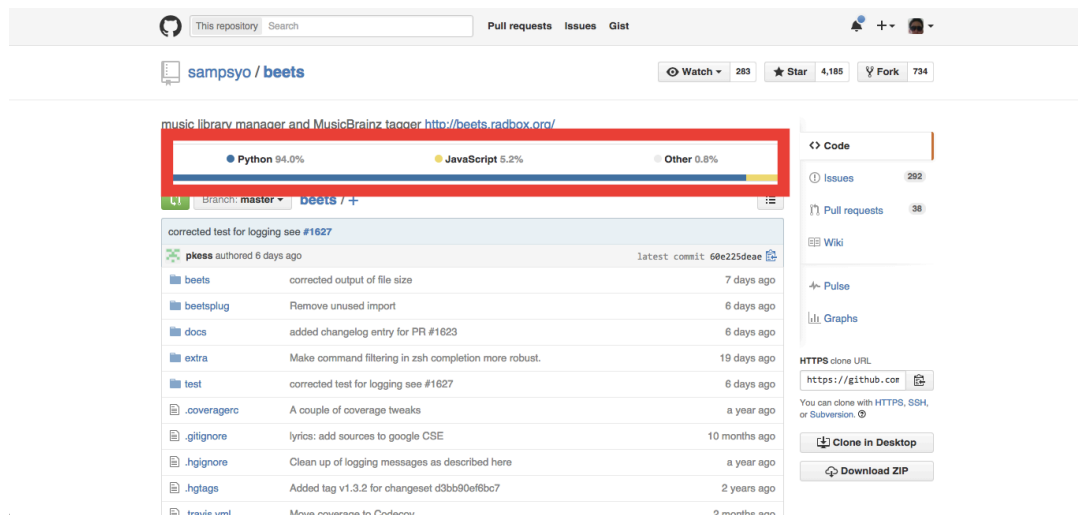


図 4.2 言語の調査

図 4.2 は言語の調査画面である．赤枠に囲まれている数値を取得する．所得した数値は，"データ一覧.csv"に保存する．

4.3.3 OpenIssues 数，ClosedIssues 数，Issue 数の調査

調査は，手動で行う．

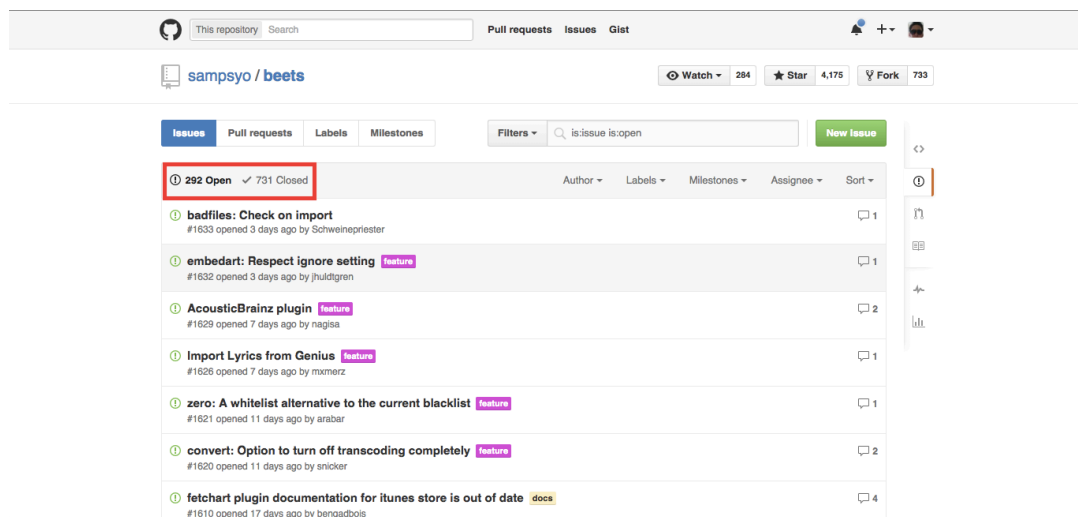


図 4.3 Issues 数の調査

図 4.3 は OpenIssues 数，ClosedIssues 数，Issue 数の調査画面である．Open が OpenIssues 数，Closed が ClosedIssues 数である．Open と Closed の合計が Issue 数である．赤枠に囲まれている数値を取得する．所得した数値は，"データ一覧.csv"に保存する．

4.3.4 OpenPull Request 数 , ClosedPull Request 数

調査は , 手動で行う .

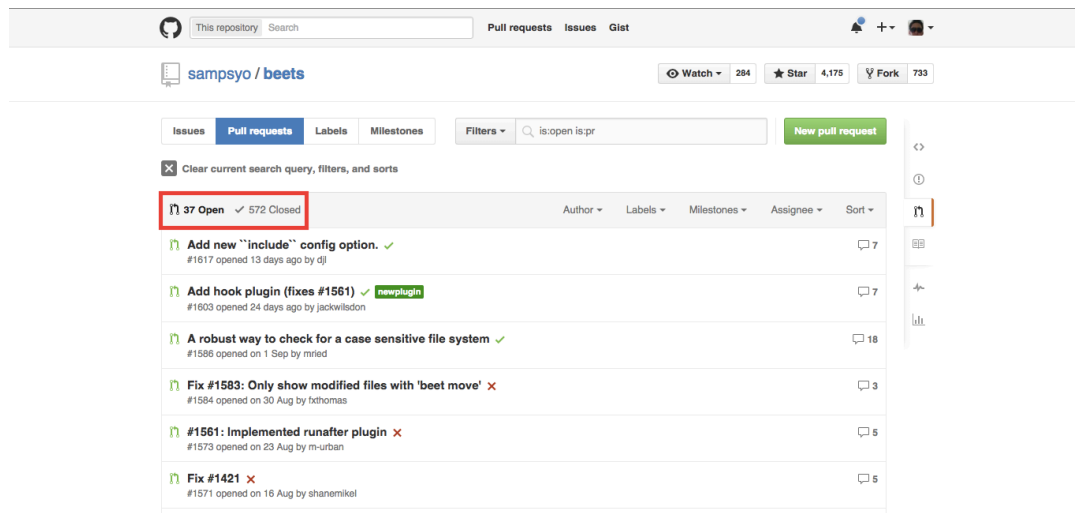


図 4.4 Pull Request 数の調査

図 4.4 は OpenPull Request 数 , ClosedPull Request 数の調査画面である . Open が OpenPull Request 数 , Closed が ClosedPull Request 数である . Open と Closed の合計が Pull Request 数である . 赤枠に囲まれている数値を取得する . 所得した数値は , "データ一覧.csv"に保存する .

4.3.5 Label 数

調査は , 手動で行う .

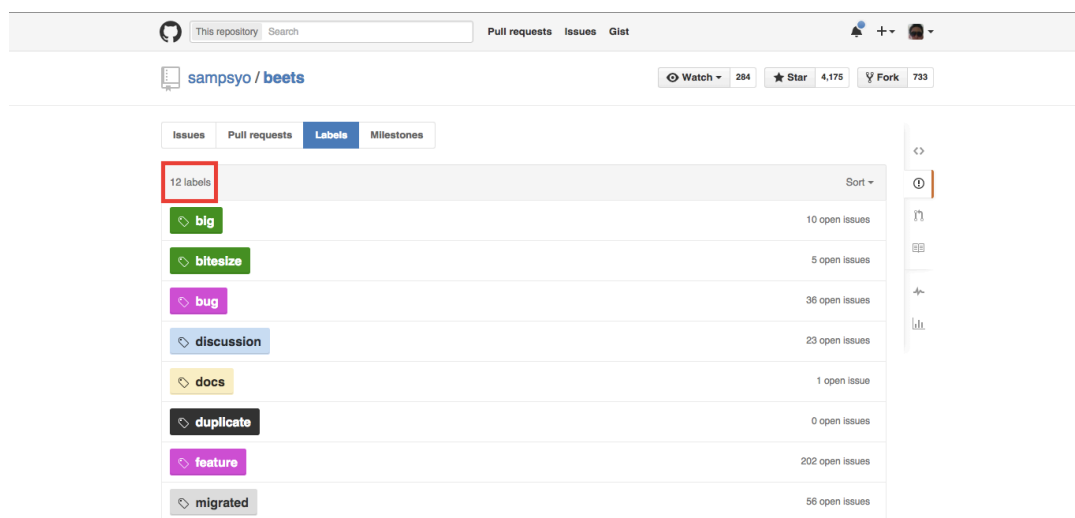


図 4.5 Label 数の調査

図 4.5 は Label 数の調査画面である。labels が Label 数である。赤枠に囲まれている数値を取得する。所得した数値は、"データ一覧.csv"に保存する。

4.3.6 OpenMilestone 数, ClosedMilestone 数, Milestone 数

調査は、手動で行う。

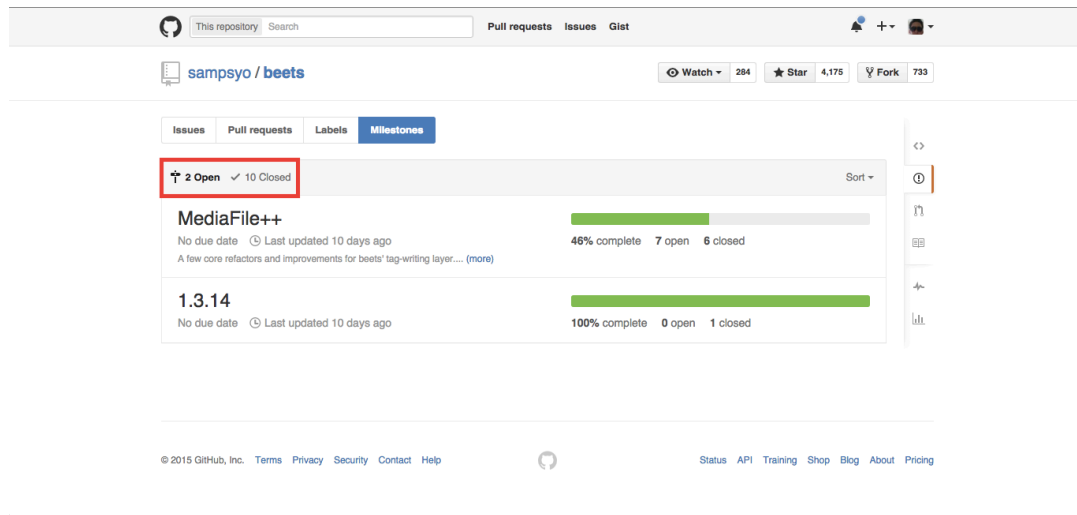


図 4.6 Milestone 数の調査

図 4.6 は OpenMilestone 数, ClosedMilestone 数, Milestone 数の調査画面である。Open が OpenMilestone 数, Closed が ClosedMilestone 数である。Open と Closed の合計が Milestone 数である。赤枠に囲まれている数値を取得する。所得した数値は、"データ一覧.csv"に保存する。

4.3.7 Wiki 数

調査は、手動で行う。

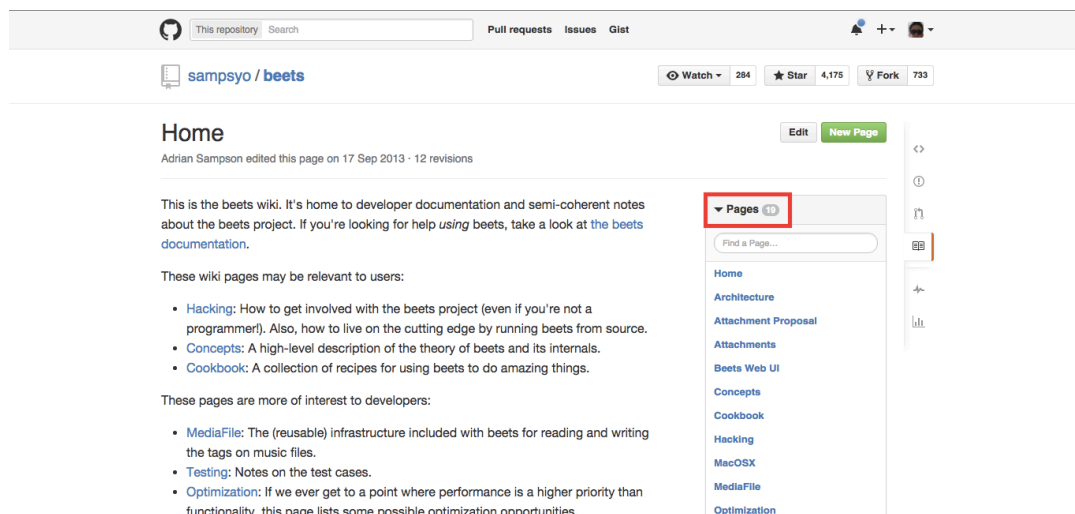


図 4.7 Wiki 数の調査

図 4.7 は Wiki 数の調査画面である。Pages の隣の数値が Wiki 数である。赤枠に囲まれている数値を取得する。所得した数値は、「データー一覧.csv」に保存する。

4.3.8 ファイル数, バイト数

指定のリポジトリを clone する。

```
git clone git@github.com:アカウント名/リポジトリ名
```

clone したリポジトリの情報を確認する。右クリックし、「情報をみる」を選択する。選択すると、バイト数とファイル数が表示される。表示された数値を、「データー一覧.csv」に保存する。

4.3.9 行数

指定のリポジトリを clone する。

```
git clone git@github.com:アカウント名/リポジトリ名
```

clone したリポジトリをターミナルで開く。ターミナルで、以下のコマンドをうつ。

```
grep -rI '' リポジトリ名 | wc -l
```

行数が出力される。

出力された行数を、「データー一覧.csv」に保存する。

4.3.10 日数

GitHub が提供している API を用いる。

最初の commit した日をプロジェクト開始日とする。最後の commit した日をプロジェクト終了日とする。これは、すでに終了しているプロジェクトや、commit が長期間されて

いないプロジェクトを考慮しているためである。

まず、API を用いて、すべての commit を取得する。

```
python api.py https://api.github.com/repos/ユーザ名/リポジトリ名/commits?per_page=100 > ユーザ名.リポジトリ名.-commits.txt
```

すべての commit から最初と最後の commit を特定し、日付を取得する。取得した数値を、"データ一覧.csv"に保存する。

4.3.11 一日あたりの行数

一日あたりの行数は、行数をプロジェクト経過日数で割る。

4.3.12 1 人日あたりの行数

1 人日あたりの行数は、行数をプロジェクト経過日数と人数で割る。

4.3.13 一日あたりの commit 数

一日あたりの commit 数は、commit 数をプロジェクト経過日数で割る。

4.3.14 1 人日あたりの commit 数

1 人日あたりの commit 数は、commit 数をプロジェクト経過日数と人数で割る。

4.3.15 開発フロー

開発フローの特徴とプロジェクトの特徴を照らし合わせる。

GitHub フロー

master branch から記述的な名前の branch がある場合は、GitHub フローである。

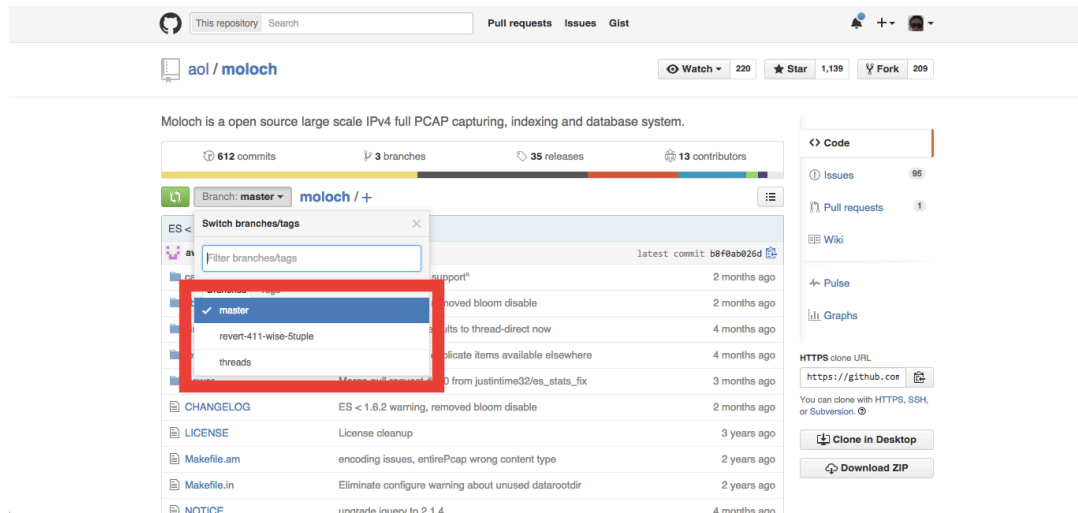


図 4.8 GitHub フローの調査

Git フロー

develop branch と release branch がある場合は , Git フローである .

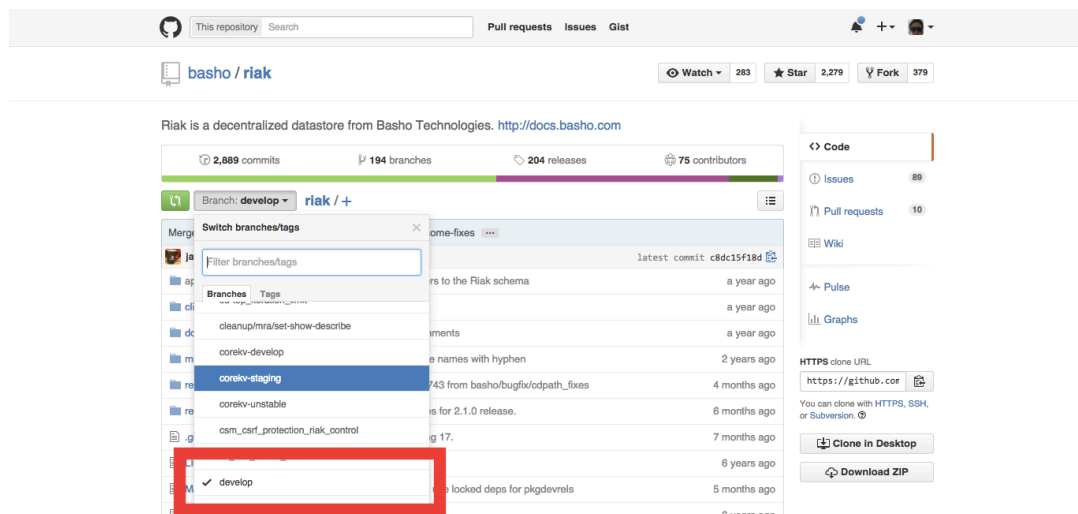


図 4.9 Git フローの調査

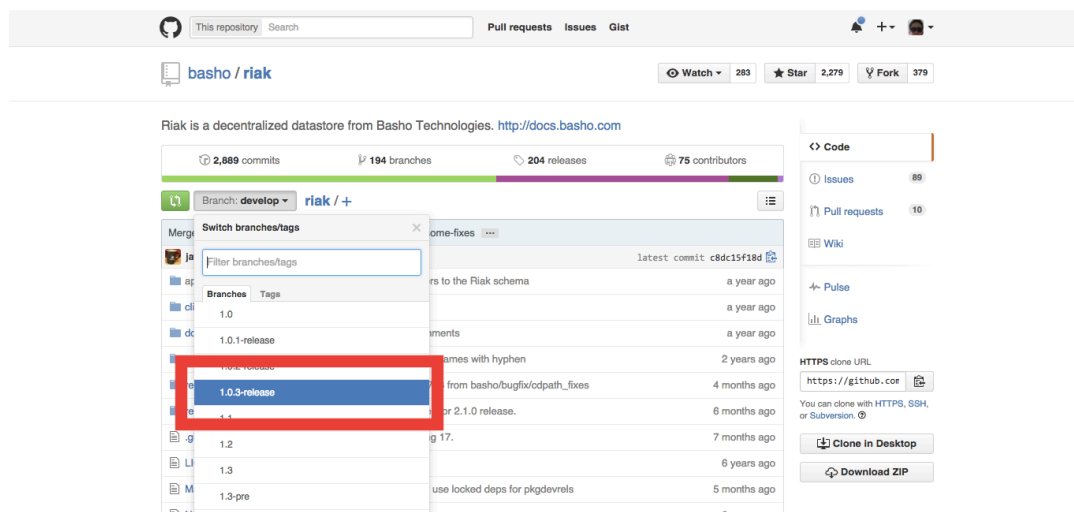


図 4.10 Git フローの調査

LINE フロー

バージョンごとに branch が作られている場合は、LINE フローである。

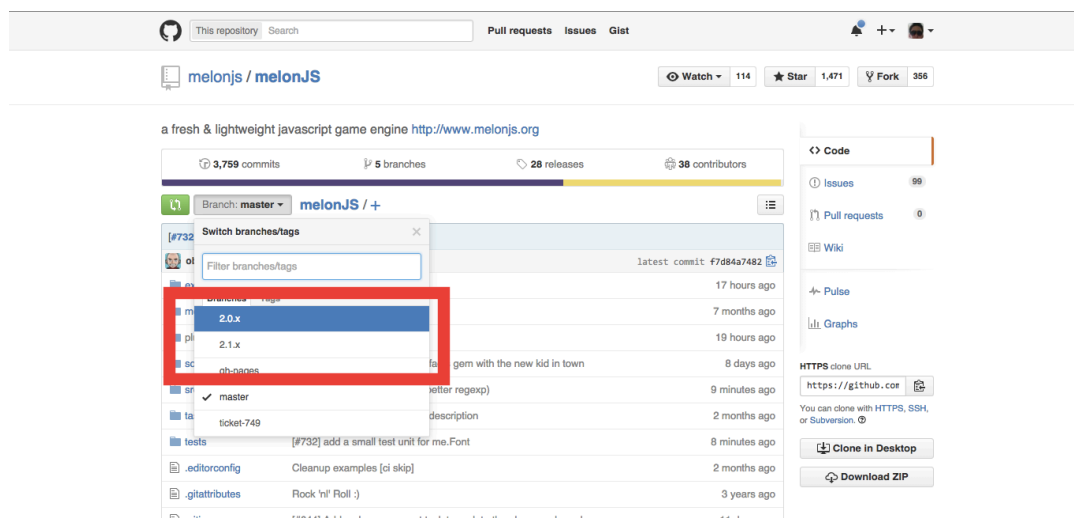


図 4.11 LINE フローの調査

日本 CAW フロー

Pull Request に [WIP] がある場合は、日本 CAW フローである。

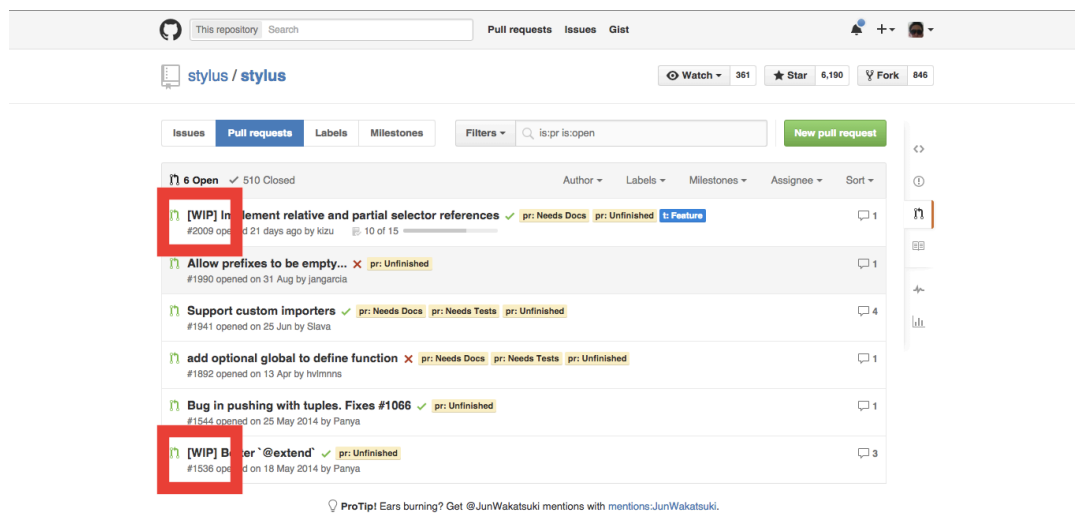


図 4.12 日本 CAW フローの調査

GitLab フロー

branch に stable がある場合は、GitLab フローである。

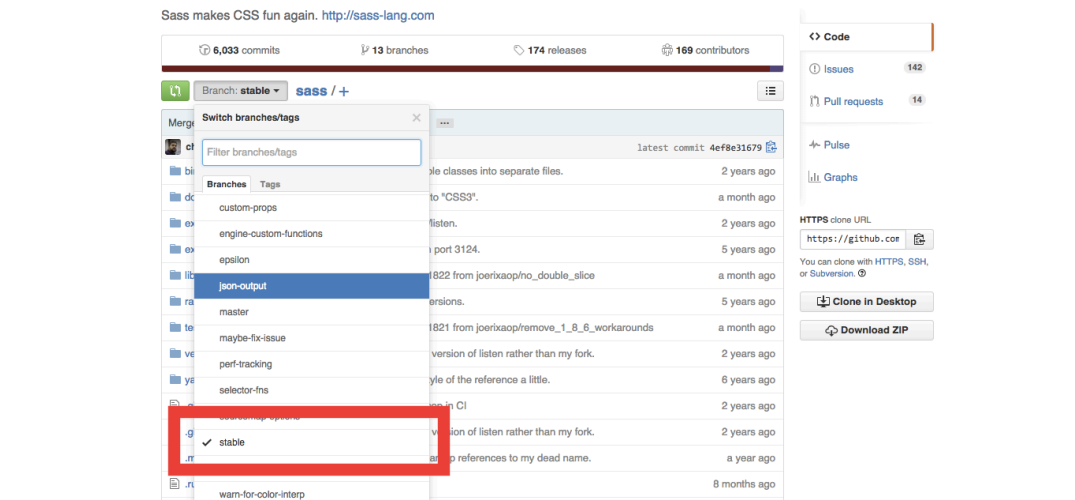


図 4.13 GitLab フローの調査

4.3.16 コマンドの説明

調査に用いたコマンドの説明を、以下に記述する。

requests

人が使いやすいように設計されていて、Python で書かれている Apache2 Licensed ベースの HTTP ライブラリである。

sudo

UNIX および, UNIX 系オペレーティングシステムのプログラムの 1 つで, ユーザが別のユーザの権限レベルでプログラムを実行するためのコマンドである.

echo

文字列を出力するコマンド

apt-get

パッケージを取得してインストール/アップデートする Debian プロジェクトが開発したパッケージ管理システムである.

chmod

UNIX および, UNIX 系オペレーティングシステムにおけるシェルコマンドの一種である. ファイルやディレクトリのファイルモードを変更するのに使われる.

grep

ファイルの中の文字列を検索する.

-r

各ディレクトリ下のすべてのファイルを再帰的に読み取る.

-l

バイナリファイルを無視する.

|

コマンドの標準出力を次のコマンドに渡す処理を行う.

wc

テキスト・ファイルの行数, 単語数, バイト数を表示する.

-l

行数のみ集計し表示する.

4.4 分析ツール

分析に使うツールは, Excel と R である. Excel は, 集めたデータを分析用のデータに分別したり, 保存しておくために使う. R は, 集めたデータを分析するために使う.

R とは, 統計計算をしたり, その結果をグラフにまとめるための言語・環境で, ボランティアによって作られた, S-PLUS の主要な関数をほぼ備えた GNU 版 (コピーフリー版)

です。1991年、ニュージーランド、オークランド大学統計学科の2人の講師、Ross Ihaka と Robert Gentleman によって開発が始まり、現在も開発が進められています [12]。

4.4.1 R パッケージのインストール

決定木分析を、R で行う方法を記述する。
3 種類のパッケージをインストールする

```
options(repos=c(CRAN="http://cran.ism.ac.jp"))
```

"rattle"をインストールする。"rattle"とは、

```
install.packages("rattle")
```

成功すると以下の画面になる。

```
> install.packages("rattle")
Installing package into 'C:/Users/██████/Documents/R/win-library/3.1'
(as 'lib' is unspecified)
URL 'http://cran.ism.ac.jp/bin/windows/contrib/3.1/rattle_4.0.5.zip' を試しています
Content type 'application/zip' length unknown
開かれた URL
downloaded 3.6 Mb

パッケージ 'rattle' は無事に展開され、MD5 サムもチェックされました
```

図 4.14 rattle インストール成功画面

"RColorBrewer"をインストールする。"RColorBrewer"とは、様々な美しいカラーパレットが含まれているパッケージである。

```
install.packages("RColorBrewer")
```

成功すると以下の画面になる。

```
> install.packages("RColorBrewer")
Installing package into 'C:/Users/██████/Documents/R/win-library/3.1'
(as 'lib' is unspecified)
URL 'http://cran.ism.ac.jp/bin/windows/contrib/3.1/RColorBrewer_1.1-2.zip' を試しています
Content type 'application/zip' length unknown
開かれた URL
downloaded 26 Kb

パッケージ 'RColorBrewer' は無事に展開され、MD5 サムもチェックされました
```

図 4.15 RColorBrewer インストール成功画面

"rpart.plot"をインストールする。"rpart.plot"とは、視覚的に表示するためのパッケージである。

```
install.packages("rpart.plot")
```

成功すると以下の画面になる。

```
> install.packages("rpart.plot")
Installing package into 'C:/Users/████████/Documents/R/win-library/3.1'
(as 'lib' is unspecified)
URL 'http://cran.ism.ac.jp/bin/windows/contrib/3.1/rpart.plot_1.5.3.zip' を試しています
Content type 'application/zip' length unknown
開かれた URL
downloaded 510 Kb
```

パッケージ 'rpart.plot' は無事に展開され、MD5 サムもチェックされました

図 4.16 rpartPlot インストール成功画面

4.5 分析データ準備

4.5.1 全てのデータをまとめる

Excel の csv ファイルに調査した全てのデータを保存する．実際に保存したデータが図 4.17 である．

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000
Year	Month	Day	Hour	Minute	Second	Millisecond	Microsecond	Nanosecond	Picosecond	Femtosecond	Attosecond	Zeptosecond	Yoctosecond	R Planck	R Boltzmann	R Gas	R Avogadro	R Faraday	R Coulomb	R Ampere	R Volt	R Ohm	R Watt	R Joule	R Newton	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal	R Pascal																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						

図 4.17 すべてのデータ一覧

Y 列からは，言語になっている．その言語があった場合は 1，ない場合は 0 を入れている．

4.5.2 全ての2つにわたる

データをランダムに並び替え，2 つにわけるとランダムに分ける方法は，Excel の RAND 関数を使用する．わけられたデータで，それぞれ決定木分析を行う．

分けられたデータが図 4.18 と図 4.19 である。

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119	1120	1121	1122	1123	1124
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

図 4.18 半分のデータ-1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM	BN	BO	BP	BQ	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA	CB	CC	CD	CE	CF	CG	CH	CI	CJ	CK	CL	CM	CN	CO	CP	CQ	CR	CS	CT	CU	CV	CW	CX	CY	CZ	DA	DB	DC	DD	DE	DF	DG	DH	DI	DJ	DK	DL	DM	DN	DO	DP	DQ	DR	DS	DT	DU	DV	DW	DX	DY	DZ	EA	EB	EC	ED	EE	EF	EG	EH	EI	EJ	EK	EL	EM	EN	EO	EP	EQ	ER	ES	ET	EU	EV	EW	EX	EY	EZ	FA	FB	FC	FD	FE	FF	FG	FH	FI	FJ	FK	FL	FM	FN	FO	FP	FQ	FR	FS	FT	FU	FV	FW	FX	FY	FZ	GA	GB	GC	GD	GE	GF	GG	GH	GI	GJ	GK	GL	GM	GN	GO	GP	GQ	GR	GS	GT	GU	GV	GW	GX	GY	GZ	HA	HB	HC	HD	HE	HF	HG	HH	HI	HJ	HK	HL	HM	HN	HO	HP	HQ	HR	HS	HT	HU	HV	HW	HX	HY	HZ	IA	IB	IC	ID	IE	IF	IG	IH	II	IJ	IK	IL	IM	IN	IO	IP	IQ	IR	IS	IT	IU	IV	IW	IX	IY	IZ	JA	JB	JC	JD	JE	JF	JG	JH	JI	IJ	JK	JL	JM	JN	JO	JP	JQ	JR	JS	JT	JU	JV	JW	JX	JY	JZ	KA	KB	KC	KD	KE	KF	KG	KH	KI	KJ	KK	KL	KM	KN	KO	KP	KQ	KR	KS	KT	KU	KV	KW	KX	KY	KZ	LA	LB	LC	LD	LE	LF	LG	LH	LI	LJ	LK	LM	LN	LO	LP	LQ	LR	LS	LT	LU	LV	LW	LX	LY	LZ	MA	MB	MC	MD	ME	MF	MG	MH	MI	MJ	MK	ML	MM	MN	MO	MP	MQ	MR	MS	MT	MU	MV	MW	MX	MY	MZ	NA	NB	NC	ND	NE	NF	NG	NH	NI	NJ	NK	NL	NM	NO	NP	NQ	NR	NS	NT	NU	NV	NW	NX	NY	NZ	OA	OB	OC	OD	OE	OF	OG	OH	OI	OJ	OK	OL	OM	ON	OO	OP	OQ	OR	OS	OT	OU	OV	OW	OX	OY	OZ	PA	PB	PC	PD	PE	PF	PG	PH	PI	PJ	PK	PL	PM	PN	PO	PP	PQ	PR	PS	PT	PU	PV	PW	PX	PY	PZ	QA	QB	QC	QD	QE	QF	QG	QH	QI	QJ	QK	QL	QM	QN	QO	QP	QQ	QR	QS	QT	QU	QV	QW	QX	QY	QZ	RA	RB	RC	RD	RE	RF	RG	RH	RI	RJ	RK	RL	RM	RN	RO	RP	RQ	RR	RS	RT	RU	RV	RW	RX	RY	RZ	SA	SB	SC	SD	SE	SF	SG	SH	SI	SJ	SK	SL	SM	SN	SO	SP	SQ	SR	SS	ST	SU	SV	SW	SX	SY	SZ	TA	TB	TC	TD	TE	TF	TG	TH	TI	TJ	TK	TL	TM	TN	TO	TP	TQ	TR	TS	TU	TV	TW	TX	TY	TZ	UA	UB	UC	UD	UE	UF	UG	UH	UI	UJ	UK	UL	UM	UN	UO	UP	UQ	UR	US	UT	UU	UV	UW	UX	UY	UZ	VA	VB	VC	VD	VE	VF	VG	VH	VI	VJ	VK	VL	VM	VN	VO	VP	VQ	VR	VS	VT	VU	VV	VW	VX	VY	VZ	WA	WB	WC	WD	WE	WF	WG	WH	WI	WJ	WK	WL	WM	WN	WO	WP	WQ	WR	WS	WT	WU	WV	WW	WX	WY	WZ	XA	XB	XC	XD	XE	XF	XG	XH	XI	XJ	XK	XL	XM	XN	XO	XP	XQ	XR	XS	XT	XU	XV	XW	XX	XY	XZ	YA	YB	YC	YD	YE	YF	YG	YH	YI	YJ	YK	YL	YM	YN	YO	YP	YQ	YR	YS	YT	YU	YV	YW	YX	YZ	ZA	ZB	ZC	ZD	ZE	ZF	ZG	ZH	ZI	ZJ	ZK	ZL	ZM	ZN	ZO	ZP	ZQ	ZR	ZS	ZT	ZU	ZV	ZW	ZX	ZY	ZZ
g1-Row	gating	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																																																															

図 4.19 半分のデータ-2

4.5.3 コントロールできる要因かつ，成功プロジェクトの場合

プロジェクトが適切な開発フローを選択していることを実証する。プロジェクトが成功している場合、適切な開発フローを選択していることにする。プロジェクトが成功している場合の定義は、リリース数が1以上の時である。

また、選択基準を、プロジェクトが始められる時に使えるように、コントロールできる要因のみにする．コントロールできる要因とは、行数，プロジェクト経過日数，一日あたりの行数，一人日あたりの行数人数，言語である．

プロジェクトが成功している場合のデータをまとめたデータが図 4.20 である。

[illegible]

図 4.20 コントロールできる要因かつ、成功プロジェクト

4.6 階層的クラスタ分析方法

"データ一覧.csv"のデータを読み込み、データセット"myData"に保存する。1行目は、行ラベルということにして、データを読み込む。

```
myData <- read.csv("データ一覧.csv", row.names=1)
```

標準化してから距離を求める．標準化とは，各列の平均を 0，分散を 1 にすることである．

```
(myDistanceMatrix <- dist(scale(myData)))
```

階層的クラスタ分析を実行する.

```
myClusters <- hclust(myDistanceMatrix)
```

分析結果を表示する

```
plot(myClusters, hang = -1)
```

4.7 決定木分析方法

"データ一覧.csv"のデータを読み込み，データセット"myData"に保存する．

```
myData <- read.csv("データ一覧.csv")
```

データセット"myData"を読み込む．

```
head(myData)
```

library の"mypart"と"maptools"を読み込む

```
library(mypart)
```

```
library(maptools)
```

"myData"のデータをデータセット"myTree"に保存する

```
myTree <- rpart(work フロー ~ ., data = myData)
```

```
myTree
```

決定木を描画する．

```
fancyRpartPlot(myTree)
```

成功すると以下の画面になる．

```
> myData <- read.csv("決定木1.csv")
> head(myData)
```

	workflow	行数	プロジェクト経過日数	ファイル数	バイト数	Fork数	Commit数	Watch数	Star数	X1人あたりのCommit数	X1日あたりのCommit数	
1	GitHub フロー	1170	1781	18	128651	40	100	3	44		0.0026	0.0562
2	GitHub フロー	11359	597	67	2632298	633	86	359	3439		0.0096	0.1400
3	GitHub フロー	101062	622	459	13359619	213	614	225	1153		0.0710	0.9870
4	GitHub フロー	29697	410	142	64966290	13	292	19	6		0.0400	0.7120
5	Git フロー	8276	792	134	1285551	24	469	28	160		0.1000	0.5920
6	Git フロー	11740	1099	167	1551457	637	241	169	1952		0.0120	0.2190

```
> X1人あたりの行数 X1日あたりの行数 branch数 release数 人数 OpenIssues ClosedIssues Issues OpenPullRequest数 ClosedPullRequest数 PullRequest数
```

	X1人あたりの行数	X1日あたりの行数	branch数	release数	人数	OpenIssues	ClosedIssues	Issues	OpenPullRequest数	ClosedPullRequest数	PullRequest数
1	0.03	0.66	1	5	22	10	5	15	4	26	30
2	1.30	19.00	1	1	15	18	14	32	3	17	20
3	11.00	160.00	3	35	14	97	294	391	1	32	33
4	4.00	72.00	2	0	18	23	64	87	0	29	29
5	2.00	10.00	2	71	5	12	16	28	1	6	7
6	0.58	11.00	2	6	19	123	37	160	12	43	55

```
> Label数 OpenMilestone数 ClosedMilestone数 Milestone数 Wiki数 HTML JavaScript CSS Makefile C PHP Java Python Ruby TeX CoffeeScript Perl Shell
```

	Label数	OpenMilestone数	ClosedMilestone数	Milestone数	Wiki数	HTML	JavaScript	CSS	Makefile	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell
1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
2	6	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
3	9	0	0	0	0	15	1	1	0	1	1	0	0	0	0	0	1	0
4	11	1	1	2	44	1	1	1	0	0	0	0	0	0	0	0	0	0
5	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
6	6	1	0	1	0	0	0	0	0	1	0	1	1	0	0	0	0	0

```
> VimL Julia C.. Erlang LiveScript Objective.C EmacsLisp Lua Cmake Inno.Setup IDL NSIS Other
```

	VimL	Julia	C..	Erlang	LiveScript	Objective.C	EmacsLisp	Lua	Cmake	Inno.Setup	IDL	NSIS	Other
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0

```
>
> library(rpart)
> myTree <- rpart(workflow ~ ., data = myData, control = rpart.control(minsplit = 5))
>
> #木の描画
> library(rattle)
> library(rpart.plot)
> fancyRpartPlot(myTree)
```

図 4.21 分析成功画面

4.8 決定木性能測定方法

決定木性能測定方法を記述する．

4.8.1 訓練データとテストデータの準備

データをランダムに2種類に分ける。一方は、16件の訓練データ、もう一方は、8件のテストデータに分ける。

ランダムに分ける方法は、ExcelのRAND関数を使用する。

16件の訓練データで、決定木を作る。8件のテストデータで、精度と再現率を求める。

これを10回行い、10通りのデータを用意する。

4.8.2 平均と信頼区間を求める

平均は、ExcelのAVERAGE関数を用いる。標準偏差は、STDEVP関数を用いる。信頼区間は、CONFIDENCE関数を用いる。

第 5 章

結果

5.1 階層的クラスター分析結果

2 種類のデータを用いて，R で決定決定木分析を行った．すべてのデータで行った場合，コントロールできる要因のみかつ，成功しているプロジェクトの場合である．

次に，それぞれの結果を記述する．

5.1.1 すべてのデータで行った場合

図 5.1 は，すべてのデータで分析を行った結果である．

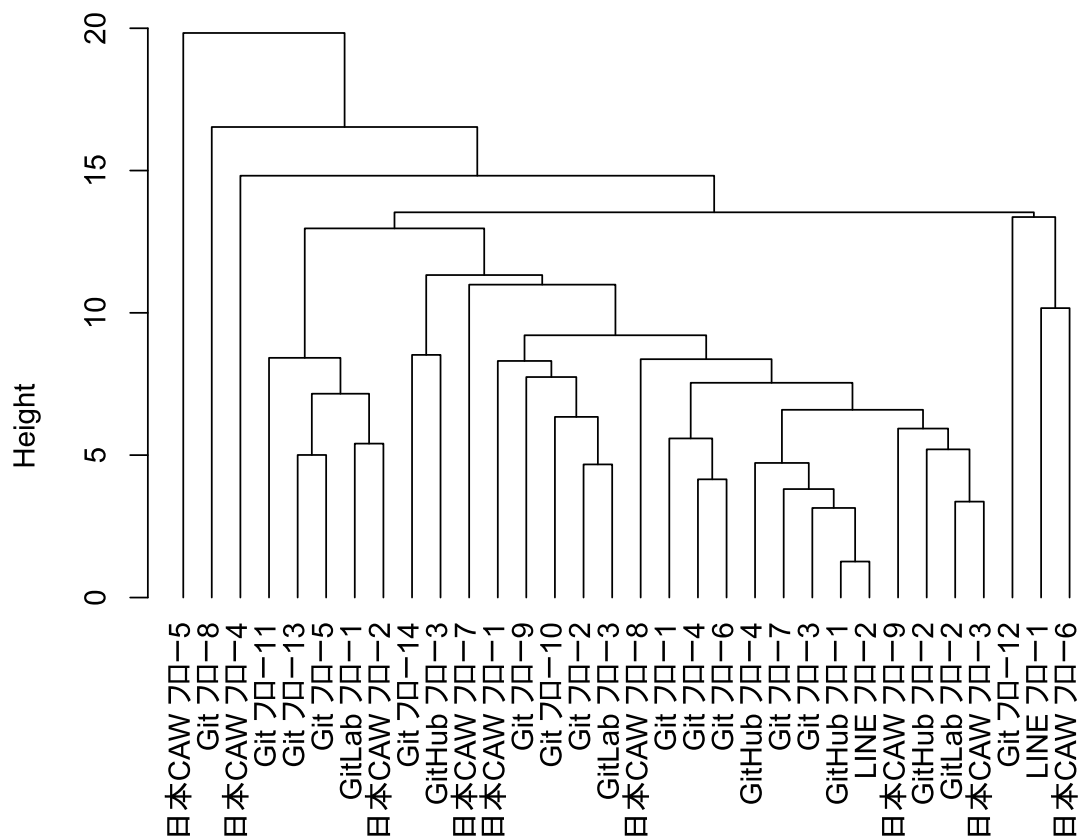


図 5.1 全てのデータで作った場合

5.1.2 コントロールできる要因のみかつ，成功しているプロジェクトの場合

図 5.2 は図 4.20 を用いて分析を行った結果である．

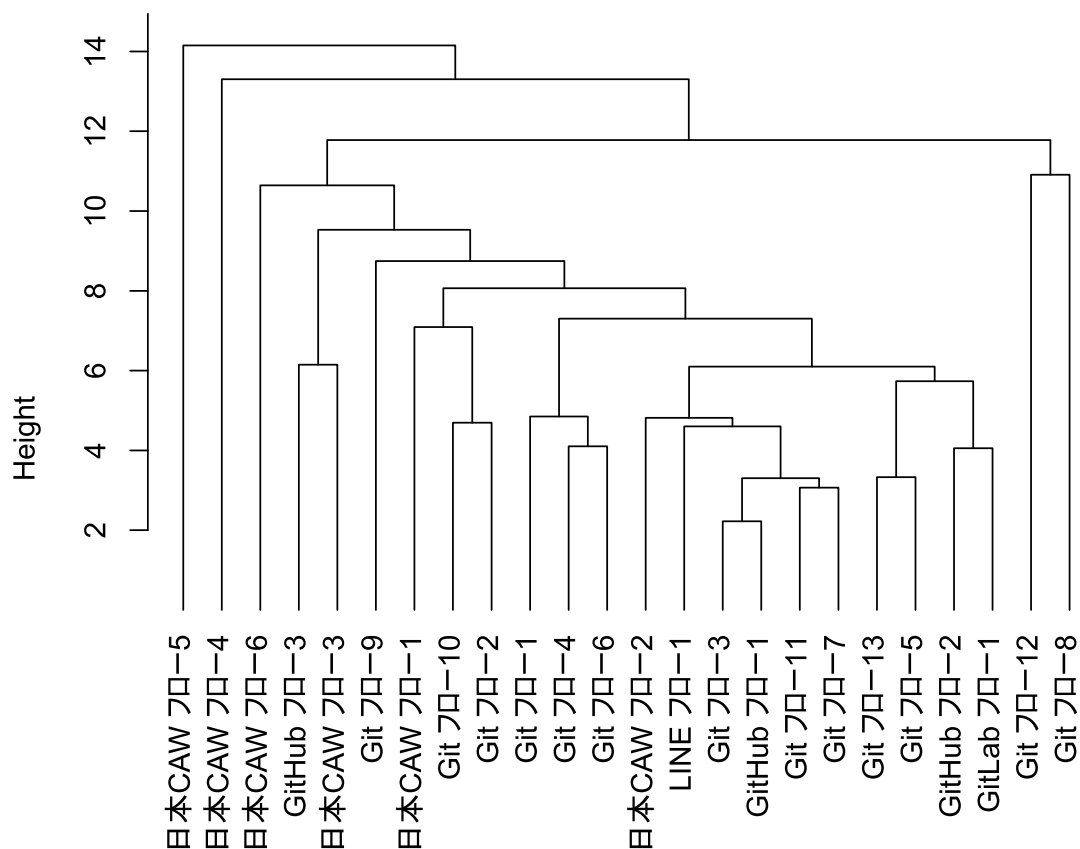


図 5.2 コントロールできる要因かつ，成功しているプロジェクトの場合

5.2 決定木分析結果

3 種類のデータを用いて，R で決定決定木分析を行った．すべてのデータで行った場合，データをランダムに並び替え，2 つに分けた場合，コントロールできる要因のみかつ，成功しているプロジェクトの場合である．

次に，それぞれの結果を記述する．

5.2.1 すべてのデータで行った場合

図 5.3 は，すべてのデータで分析を行った結果である．

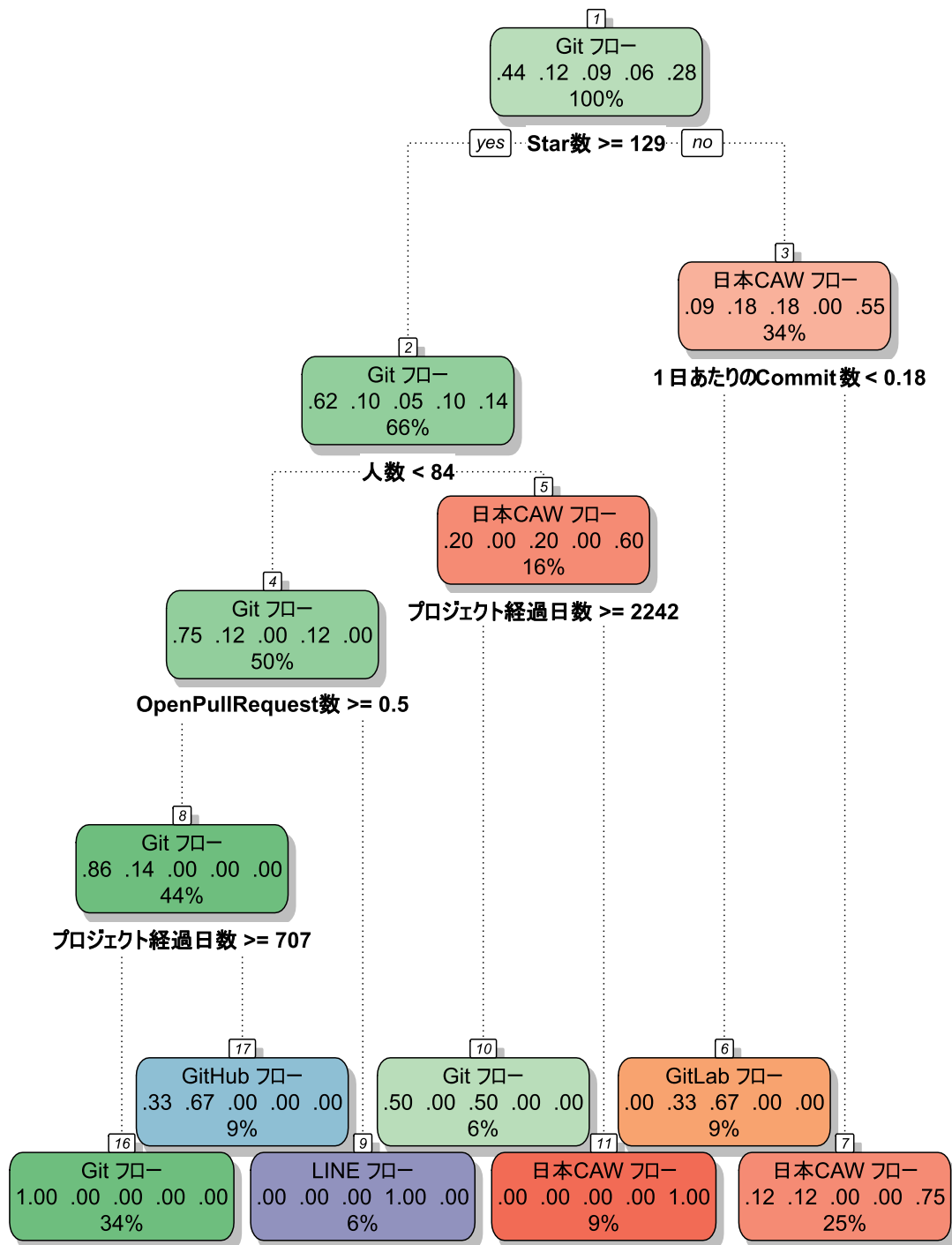


図 5.3 全てのデータで作った決定木

すべてのデータで分析した場合，Star 数，人数，Open Pull Request 数，Watch 数，ファイル数，branch 数，JavaScript で別れた．

5.2.2 データをランダムに並び替え，2 つに分けた場合

図 5.4 は，図 4.18 を用いて分析を行った結果である．図 5.5 は，図 4.19 を用いて分析を行った結果である．

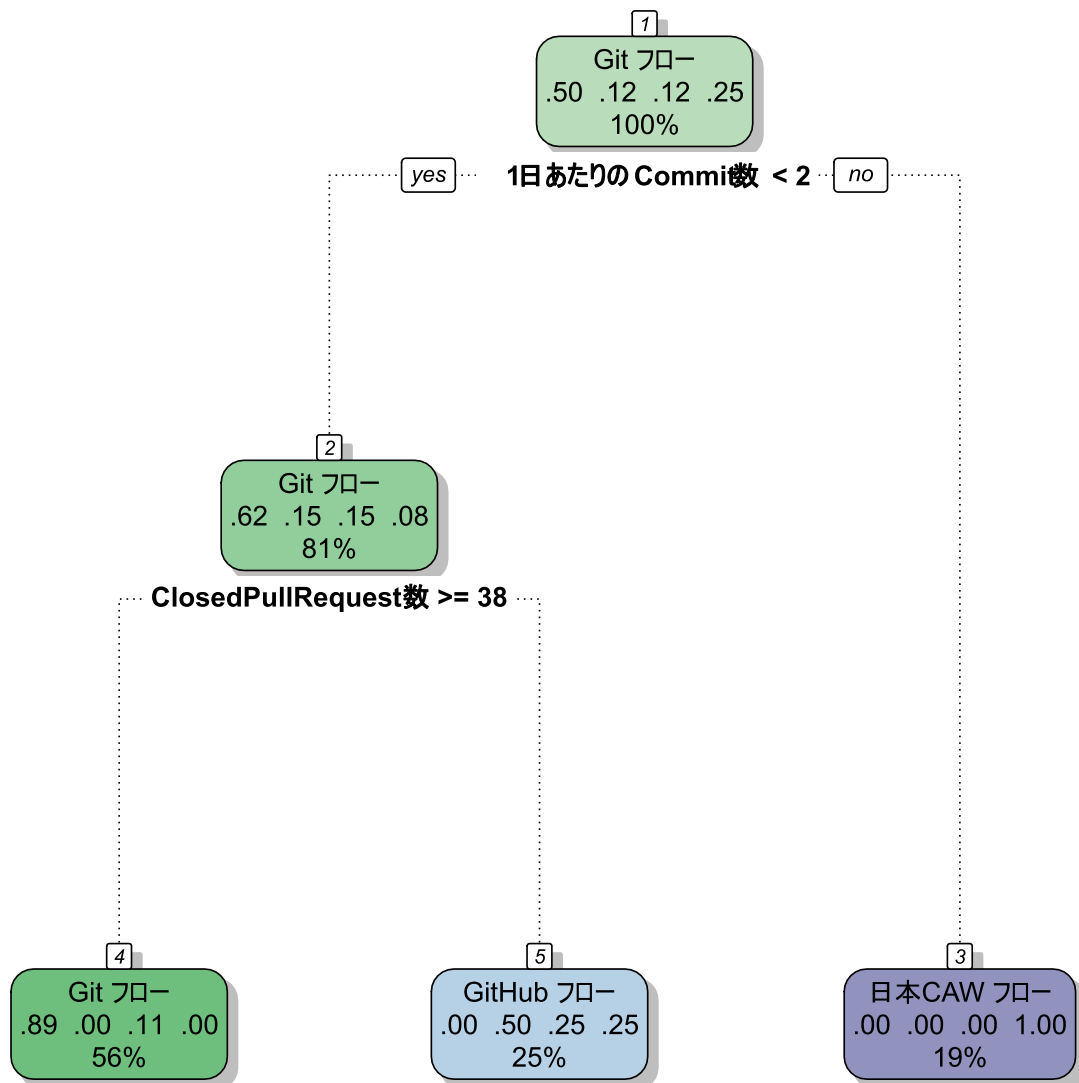


図 5.4 決定木-1

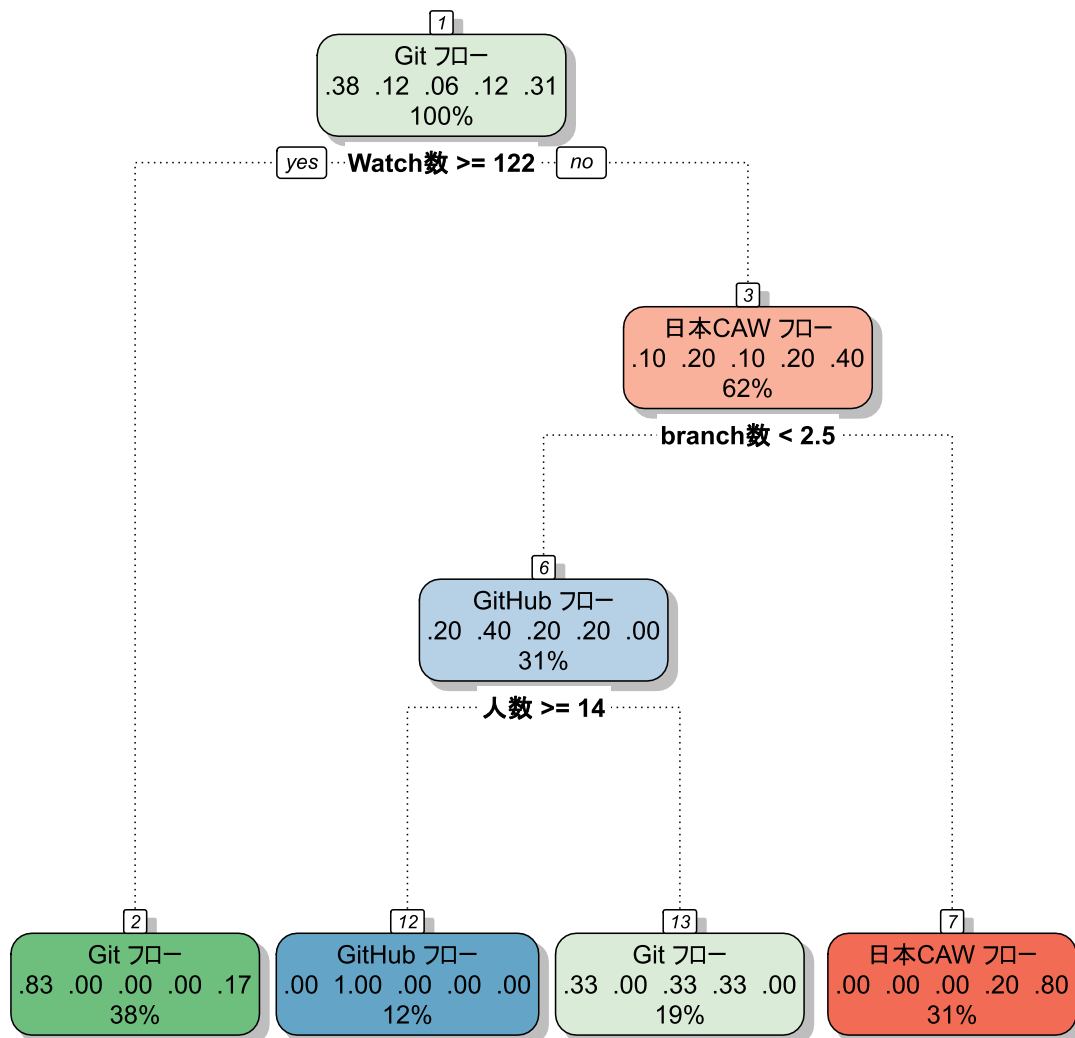


図 5.5 決定木-2

データをランダムに2つに分けた場合、異なった結果がでた。図 5.4 は、Issue 数、ファイル数で別れた。図 5.5 は、Ruby、行数、バイト数で別れた。

最も重要なファクターが全ての決定木で異なることがわかる。図 5.4 では、Issue 数である。図 5.5 では、Ruby である。また、別の決定木では、これらファクターは出てきていない。

5.2.3 コントロールできる要因のみかつ、成功しているプロジェクトの場合

図 5.6 は図 4.20 を用いて分析を行った結果である。

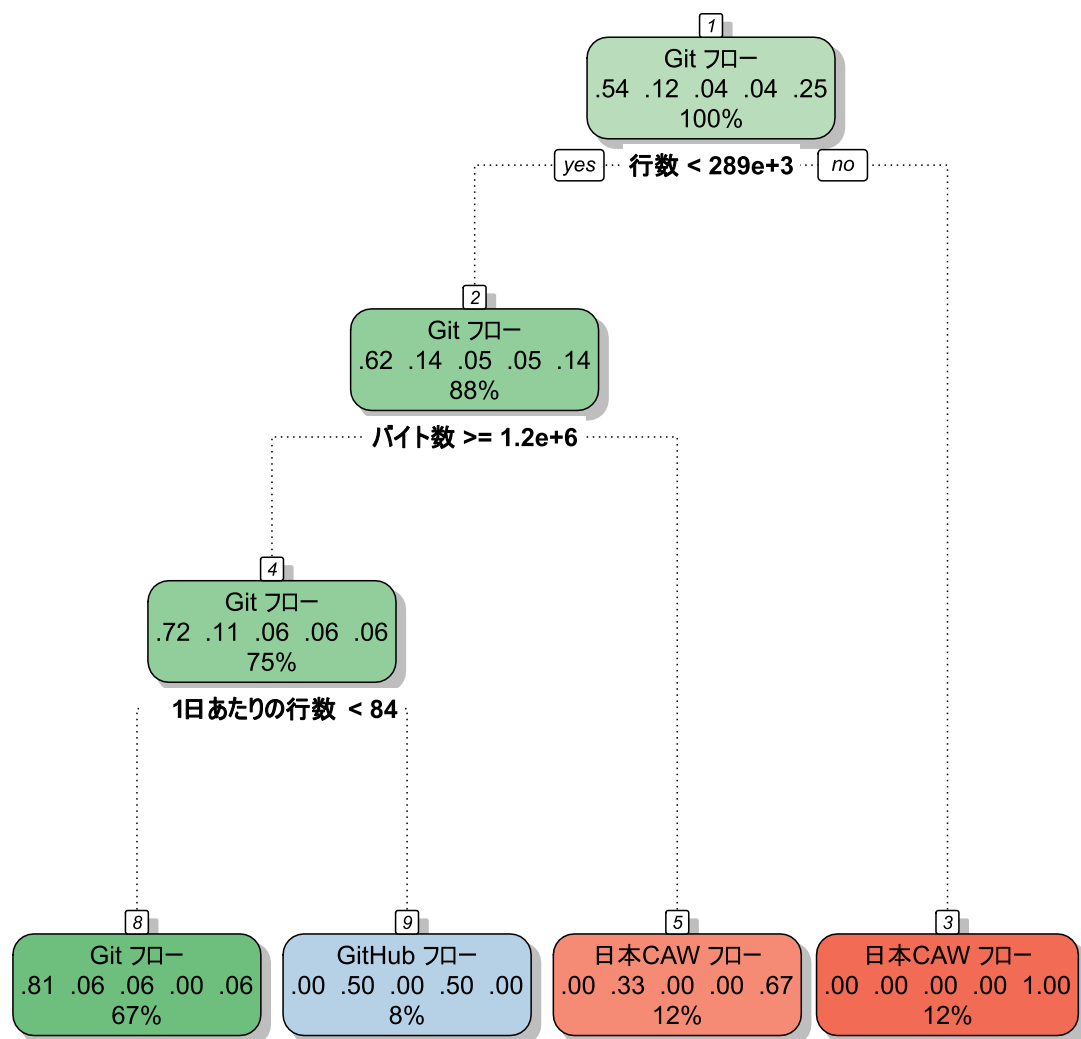


図 5.6 コントロールできる要因かつ、成功しているプロジェクトの場合

5.3 決定木性能測定結果

図 5.6 の性能を測定する．性能を測定した 10 回分の結果を，次に記す．

5.3.1 1 回目

1 回目に，決定木分析に使ったデータは，図 5.7 である．図 5.7 の決定木分析結果は，図 5.9 である．図 5.9 の性能を測定するのに使ったテストデータは，図 5.8 である．

1 回目の性能測定の結果，精度は，37.5 % だった．再現率は，50.0 % だった．

setflow	行数	プロジェクト経過日数	ファイル数	1日あたり	1人あたりの行数	人数	HTML	JavaScript	CSS	Markdown	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Brainf	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Yacc	Setup	SQL	NISS	Other	
GitCAW フロー	9137	814	214	1123424	0.41	11	27	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	11359	597	67	2032206	1.3	19	15	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	15297	2210	80	14745696	0.091	69	76	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
Git フロー	19256	1662	45	2903916	0.17	12	71	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	154218	3395	323	25964026	0.095	16	169	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	101062	622	459	13359619	11	160	14	1	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	250803	3959	1610	29134100	1.2	69	57	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	1
GitCAW フロー	911769	629	1965	66246519	0.799	146	193	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
Git フロー	3566	1982	76	1704651	0.19	51	28	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	24372	1933	150	6879431	0.21	13	61	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	8857	802	512	14245757	1.9	71	40	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GitCAW フロー	963309	896	2454	66657721	1.2	1100	81	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	6398	1519	152	11027830	0.13	42	32	1	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
GitCAW フロー	1157	472	9	1133524	0.5	2	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
GitCAW フロー	45673	1771	1077	1048551	0.184	25.8	140	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	9276	792	134	1385591	2	10	5	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

図 5.7 訓練データ-1

setflow	行数	プロジェクト経過日数	ファイル数	1日あたり	1人あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Brainf	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Yacc	Setup	SQL	NISS	Other
GitCAW フロー	9137	814	214	1123424	0.41	11	27	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	11359	597	67	2032206	1.3	19	15	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	15297	2210	80	14745696	0.091	69	76	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0
Git フロー	19256	1662	45	2903916	0.17	12	71	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	154218	3395	323	25964026	0.095	16	169	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	101062	622	459	13359619	11	160	14	1	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	250803	3959	1610	29134100	1.2	69	57	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1	1
GitCAW フロー	911769	629	1965	66246519	0.799	146	193	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	1
Git フロー	3566	1982	76	1704651	0.19	51	28	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	24372	1933	150	6879431	0.21	13	61	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	8857	802	512	14245757	1.9	71	40	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GitCAW フロー	963309	896	2454	66657721	1.2	1100	81	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	6398	1519	152	11027830	0.13	42	32	1	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
GitCAW フロー	1157	472	9	1133524	0.5	2	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
GitCAW フロー	45673	1771	1077	1048551	0.184	25.8	140	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	9276	792	134	1385591	2	10	5	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

図 5.8 テストデータ-1

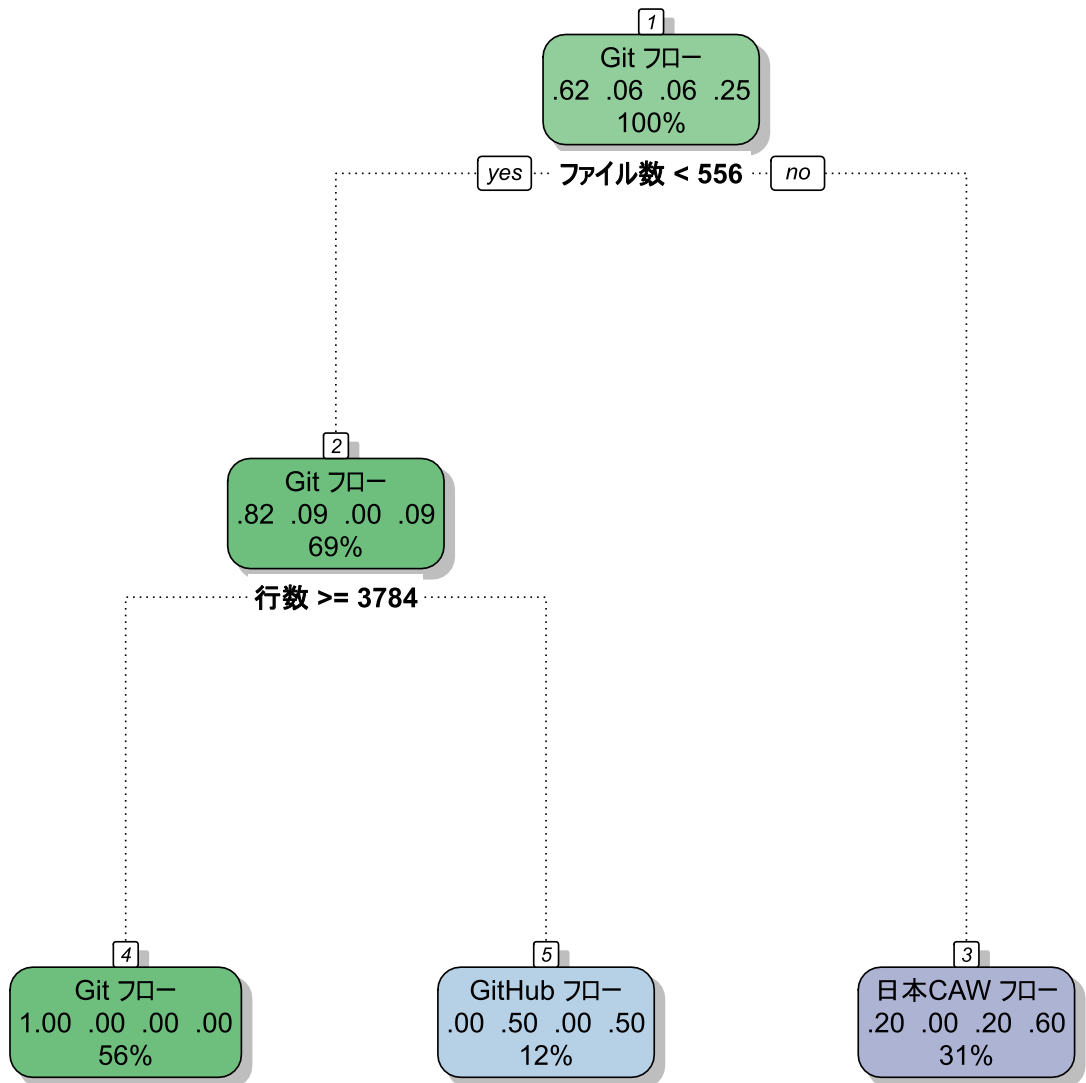


図 5.9 性能測定用決定木-1

5.3.2 2 回目

2 回目に，決定木分析に使ったデータは，図 5.10 である．図 5.10 の決定木分析結果は，図 5.12 である．図 5.12 の性能を測定するのに使ったテストデータは，図 5.11 である．

2 回目の性能測定の結果，精度は，50.0 % だった．再現率は，57.1 % だった．

workflow	行数	プロジェクト経過日数	ファイル数	バイト数	1人あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Makefile	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Inno Setup	IDL	NEBS	Other
GA プロ	8276	782	134	1385551	2	10	5	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
GA プロ	23618	485	238	2191879	2.6	49	19	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LINE プロ	161332	1845	600	73855444	2.5	86	39	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GA プロ	355803	3659	1610	39135100	1.2	69	27	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	1
GA プロ	15287	2210	80	14745696	0.091	6.9	76	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0
日教AW プロ	45673	1771	1077	1046591	0.184	29.6	140	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日教AW プロ	911769	629	1965	66246918	0.759	146	183	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	1
GA プロ	9568	1882	76	7326465	0.19	51	28	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA プロ	11740	1099	167	1551457	0.58	11	19	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日教AW プロ	1157	472	9	1132536	0.3	2	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
GitHub プロ	105062	622	459	13359616	11	160	14	1	1	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
日教AW プロ	9137	814	214	1123424	0.41	11	27	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA プロ	24372	1933	150	8878438	0.21	13	61	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GA プロ	71855	2714	319	21424103	0.199	29.5	170	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GA プロ	56687	302	912	14246757	1.6	71	40	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GitHub プロ	11359	597	67	2632286	1.3	19	15	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

図 5.10 訓練データ-2

workflow	行数	プロジェクト経過日数	ファイル数	バイト数	1人あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Makefile	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Inno Setup	IDL	NEBS	Other
GitHub プロ	1170	1781	18	128851	0.03	0.95	22	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA プロ	6388	1516	152	11027830	0.13	42	32	1	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
GA プロ	19258	1662	45	2903916	0.17	12	71	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub プロ	54219	3395	323	29564309	0.095	16	169	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA プロ	217687	1220	240	19870364	1.3	18	14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA プロ	42206	309	231	19784887	0.7	46	68	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日教AW プロ	326788	1113	1379	24326378	21	290	14	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0
日教AW プロ	863309	806	2424	89852721	12	1100	81	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

図 5.11 テストデータ-2

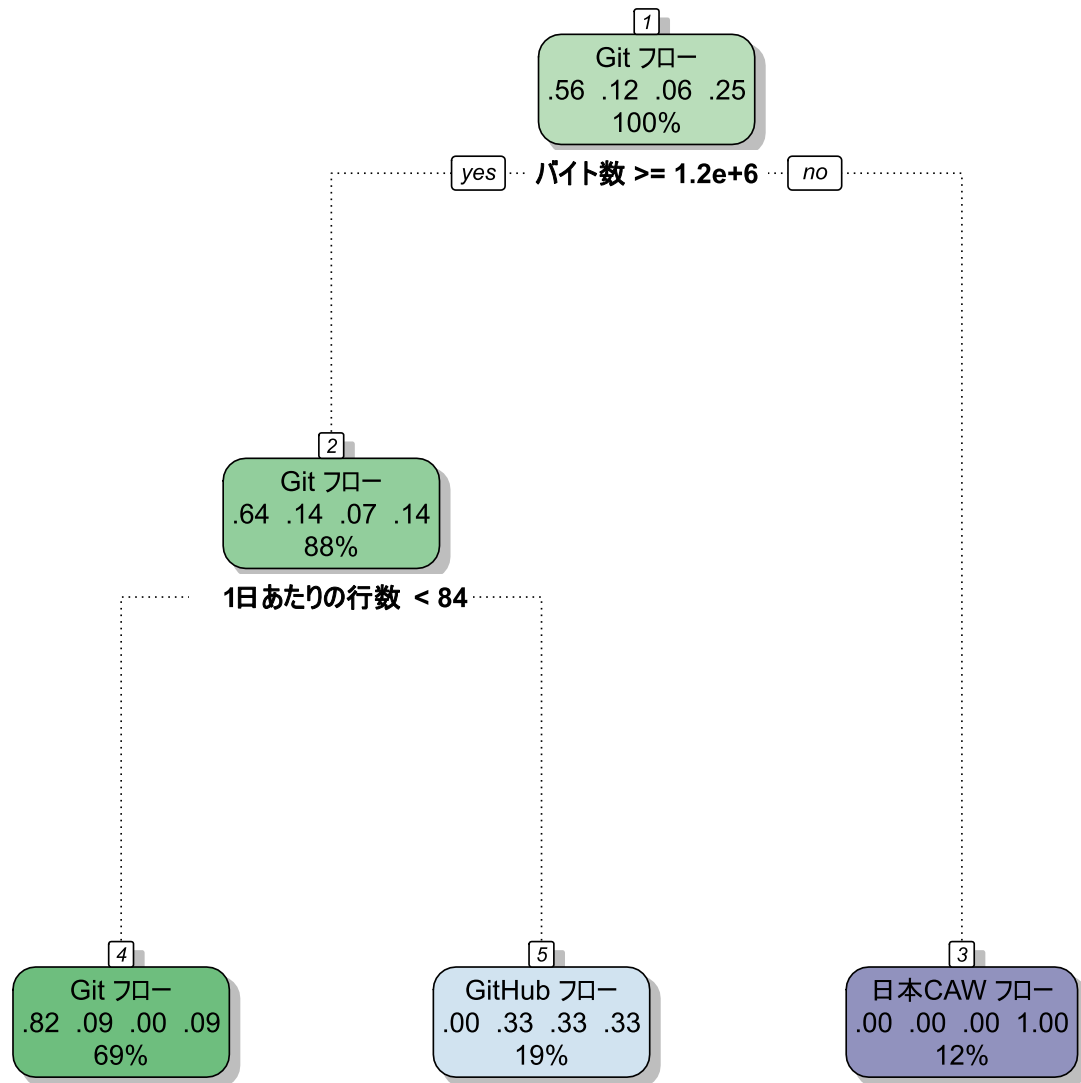


図 5.12 性能測定用決定木-2

5.3.3 3 回目

3 回目に、決定木分析に使ったデータは、図 5.13 である。図 5.13 の決定木分析結果は、図 5.15 である。図 5.15 の性能を測定するのに使ったテストデータは、図 5.14 である。

3 回目の性能測定の結果、精度は、37.5 % だった。再現率は、50.0 % だった。

workflow	行数	プロジェクト経過日数	ファイル数	バイト数	1人あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Material	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	YAML	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Inno Setup	SQL	NEOS	Other	
日本CAW フロー	81139	429	1885	56248518	0.295	146	193	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
GitHub フロー	1170	1781	18	108951	0.093	0.66	22	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	49619	1771	1077	10485917	0.084	25.6	140	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	54218	3395	323	35664028	0.095	16	169	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	250803	3659	1610	20134100	1.2	89	57	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1
GA フロー	24872	1935	150	8879458	0.21	13	61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GA フロー	11740	1989	167	1551457	0.58	11	19	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	8276	762	134	1335551	2	10	5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
GA フロー	19256	1922	45	2903916	0.17	12	71	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	15287	2210	80	1474556	0.891	6.9	76	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	326788	1113	1370	24289378	21	290	14	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	0
GA フロー	217887	1229	2421	1861266	1.3	19	14	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	15818	485	208	2161879	0.6	49	19	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	66567	802	512	14245757	1.8	71	40	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
日本CAW フロー	9137	814	214	1123424	0.41	11	27	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	101062	622	459	13359618	11	160	14	1	1	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1

図 5.13 訓練データ-3

バージョン	行数	プロジェクト経過日数	ファイル数	バイン	1人あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Matlab	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	Yaml	Julia	C++	Brain	UseScript	Objective-C	Emacs Lisp	Lua	Creole	Imo	Setup	IDL	NEOS	Other
Git フロー	6398	1519	152	11027830	0.13	42	32	1	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	71859	2714	319	21424103	0.196	285	170	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	9568	1882	76	17264891	0.18	51	28	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	1157	472	9	1132534	0.3	2	6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	42256	309	231	19784687	0.7	46	66	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	11359	597	67	3852396	1.3	18	15	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LINE フロー	141332	1440	600	73853444	2.5	96	39	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
日本CAW フロー	863306	806	2424	98852721	12	1100	81	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

図 5.14 テストデータ-3

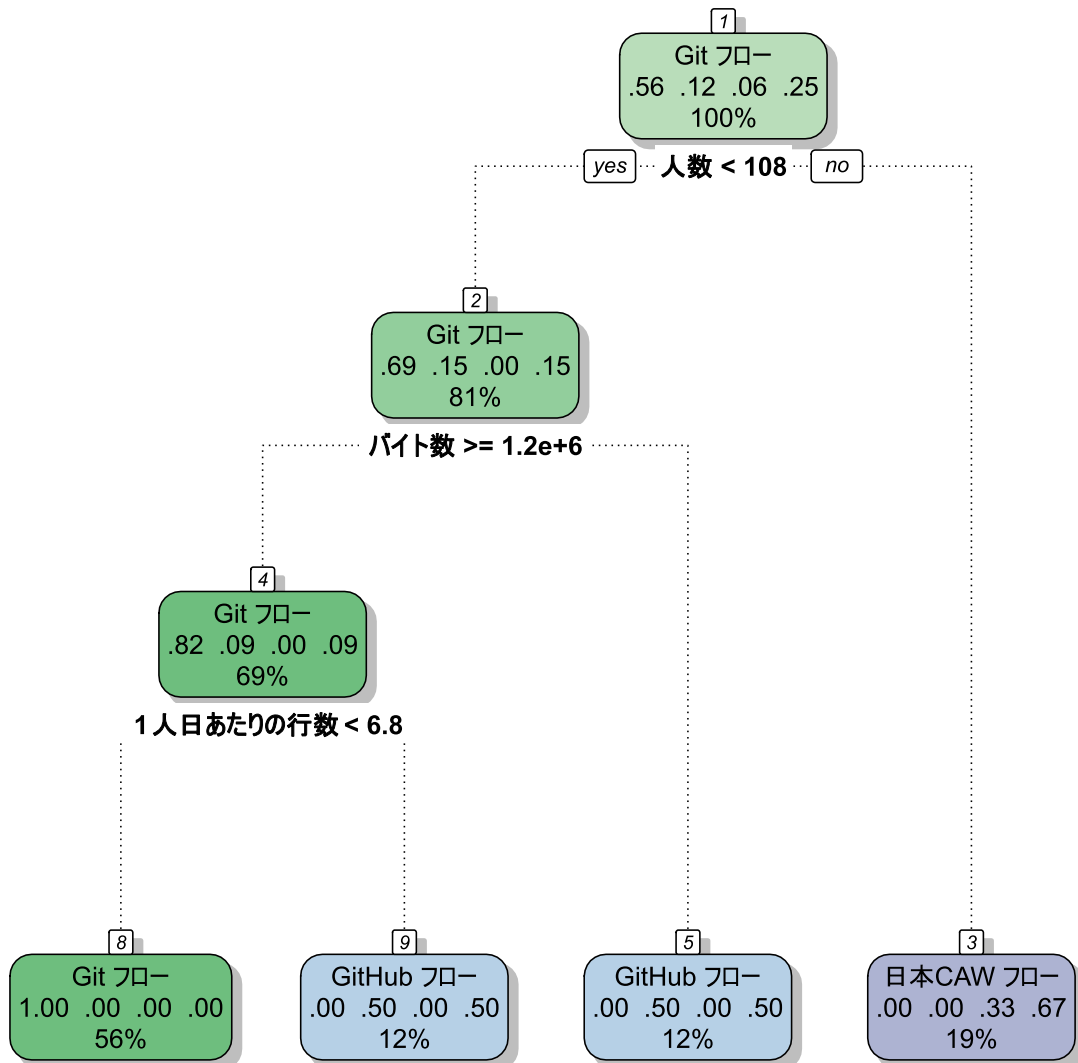


図 5.15 性能測定用決定木-3

5.3.4 4 回目

4 回目に、決定木分析に使ったデータは、図 5.16 である。図 5.16 の決定木分析結果は、図 5.18 である。図 5.18 の性能を測定するのに使ったテストデータは、図 5.17 である。

4 回目の性能測定の結果、精度は、37.5 %だった。再現率は、37.5 %だった。

workflow	行数	プロジェクト経過日数	ファイル数	1行あたり行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	Visual	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Inno Setup	SQL	NISS	Other
GitHub フロー	1170	1781	18	129851	0.03	0.66	22	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	6388	1913	152	1102783	0.19	4.2	32	1	1	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	24372	1933	150	8878438	0.21	13	61	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GitHub フロー	54218	3395	323	25664028	0.095	16	168	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	45673	1771	1077	14865917	0.184	25.8	140	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	101862	622	459	13359619	11	160	14	1	1	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	111340	1089	167	1551487	0.98	11	18	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LINE フロー	161332	1640	600	73850444	2.5	88	39	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	42036	309	231	19794697	0.7	46	66	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	1157	472	9	1135354	0.3	2	6	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Git フロー	8586	1882	76	7284861	0.19	5.1	28	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	363788	1113	1370	24385378	21	290	14	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0
Git フロー	58867	830	512	14245787	1.8	71	40	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	18266	1682	45	2803816	0.17	12	71	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	863308	806	2424	28852721	12	1100	81	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	250802	3859	1910	33134100	1.2	69	57	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1

図 5.16 訓練データ-4

workflow	行数	プロジェクト経過日数	ファイル数	1行あたり行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	Visual	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Inno Setup	SQL	NISS	Other	
Git フロー	11237	2210	80	1474560	0.091	9.9	75	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
Git フロー	71855	2714	319	21424103	0.199	29.5	170	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
日本CAW フロー	9137	814	214	1123424	0.41	11	27	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Git フロー	91769	629	1985	66246519	0.796	146	193	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	217887	1223	2491	19873594	1.3	19	14	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	11359	597	67	2623298	1.3	19	15	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	8278	792	134	1288551	2	10	5	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	23816	485	238	2161978	2.9	49	19	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

図 5.17 テストデータ-4

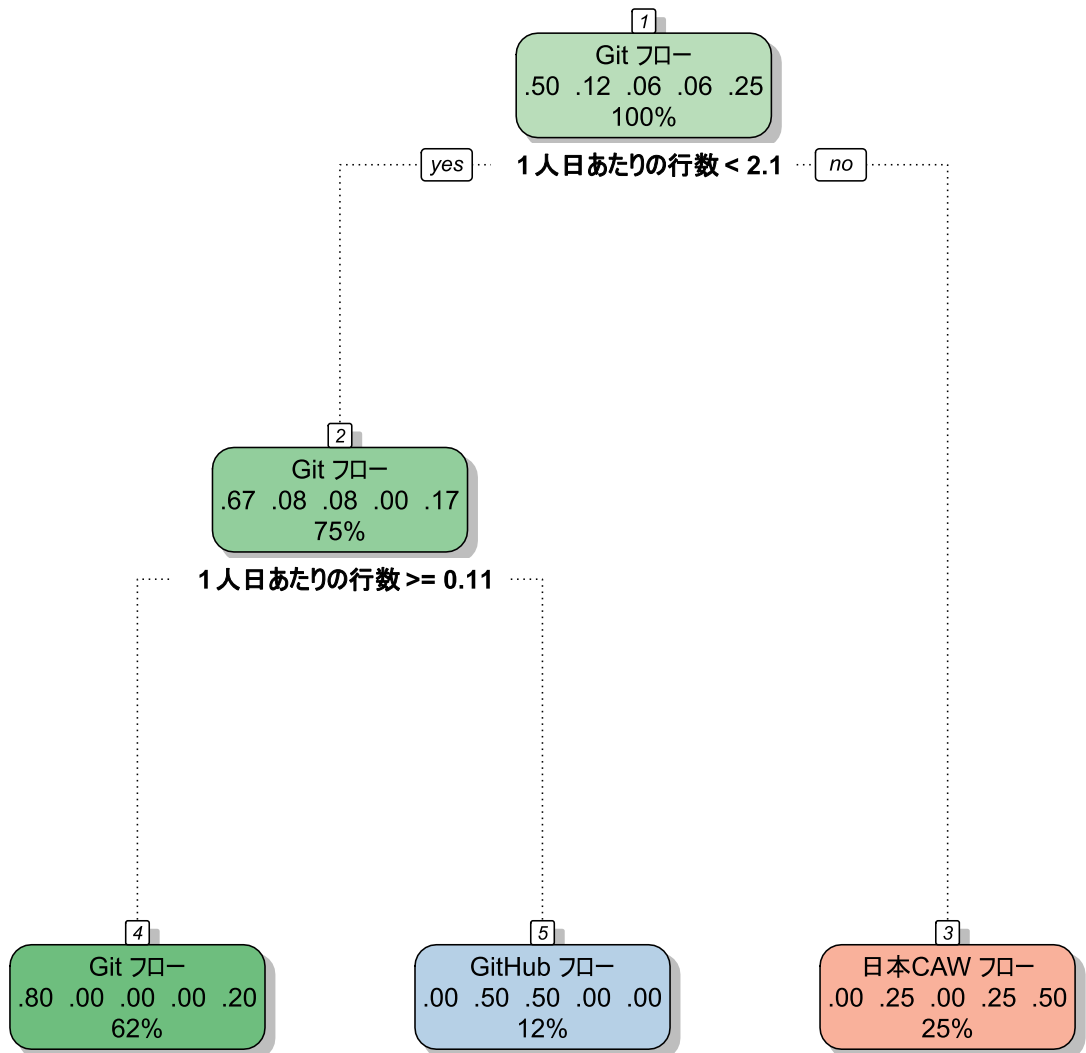


図 5.18 性能測定用決定木-4

5.3.5 5 回目

5 回目に、決定木分析に使ったデータは、図 5.19 である。図 5.19 の決定木分析結果は、図 5.21 である。図 5.21 の性能を測定するのに使ったテストデータは、図 5.20 である。

5 回目の性能測定の結果、精度は、37.5 %だった。再現率は、42.9 %だった。

workflow	行数	プロジェクト経過日数	ファイル数	バイト数	1人あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	Visual	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Inno Setup	IDL	NEOS	Other	
日#CAMP プロ-	8137	814	214	1123424	0.81	11	27	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
日#CAMP プロ-	45473	1771	1077	10465517	0.184	25.8	140	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
GA プロ-	23618	485	228	2161879	2.6	48	19	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
GitHub プロ-	14318	3395	323	25964025	0.096	16	169	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
GA プロ-	6388	1518	152	11027830	0.13	42	32	1	1	1	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
GA プロ-	8276	792	134	1285551	2	10	5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	
GA プロ-	24372	1933	150	8878439	0.21	13	61	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
GA プロ-	18256	1692	45	2903916	0.17	12	71	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
GA プロ-	42205	909	231	19794687	0.7	46	66	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
日#CAMP プロ-	811769	625	1885	26246519	0.756	146	193	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	0	1
GitHub プロ-	115062	622	499	13559619	11	160	14	1	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GA プロ-	11740	1389	167	1551457	0.98	11	19	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA プロ-	8566	1882	76	17264651	0.18	51	28	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub プロ-	11359	597	67	2632298	1.3	19	15	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA プロ-	253030	3653	1610	29124100	1.2	69	57	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	1	1
日#CAMP プロ-	863309	806	2424	88852721	12	1100	91	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

図 5.19 訓練データ-5

workflow	行数	プロジェクト経過日数	ファイル数	バイト数	1人あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	Visual	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Inno Setup	IDL	NEOS	Other	
日#CAMP プロ-	1157	472	9	1132534	0.3	2	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
GA プロ-	217887	1220	249	18373394	1.3	13	14	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日#CAMP プロ-	325788	1113	1370	24293378	21	290	14	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0
GitHub プロ-	1170	1781	18	128951	0.03	0.88	22	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LINE プロ-	161332	1640	600	73952444	25	99	39	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GA プロ-	99567	802	512	14245757	1.8	71	40	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GA プロ-	15337	2310	80	14746696	0.091	6.9	76	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0
GA プロ-	71955	2714	319	21424103	0.156	25.5	170	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

図 5.20 テストデータ-5

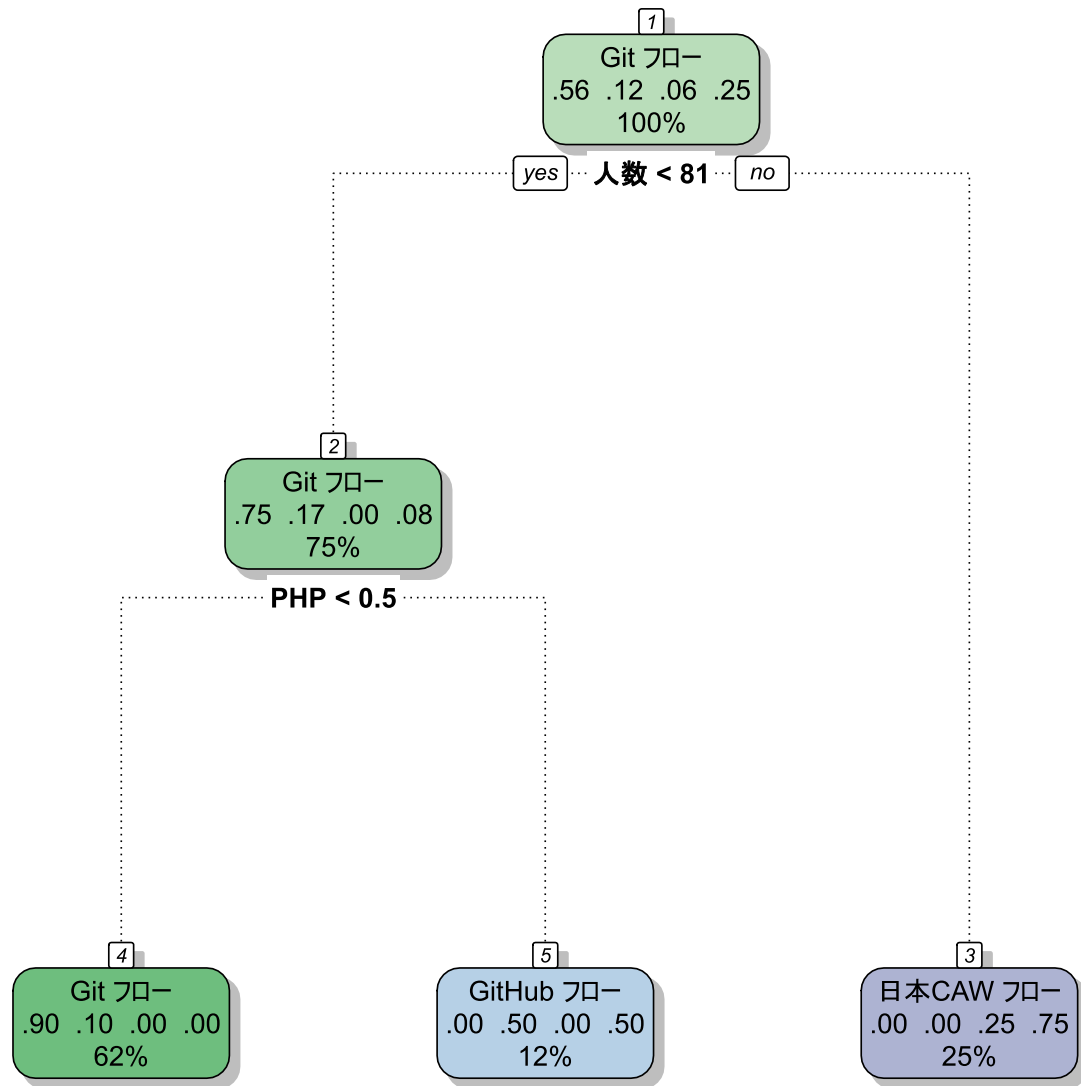


図 5.21 性能測定用決定木-5

5.3.6 6 回目

6 回目に、決定木分析に使ったデータは、図 5.22 である。図 5.22 の決定木分析結果は、図 5.24 である。図 5.24 の性能を測定するのに使ったテストデータは、図 5.23 である。

6 回目の性能測定の結果、精精度は、37.5 % だった。再現率は、42.9 % だった。

workflow	行数	プロジェクト経過日数	ファイル数	バース数	1人あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Material	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	Visual	Julia	C++	Erlang	LiveScript	Objective-C	Rascal	Lisp	Go	Make	Inno Setup	IDL	NEOS	Other
Git フロー	15397	2213	80	1476666	0.091	93	76	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	9197	814	214	1120424	0.41	11	27	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	95987	802	912	1406767	1.9	71	40	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	6598	1519	152	1102780	0.13	42	32	1	1	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	326788	1113	1370	14385378	21	290	14	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0
日本CAW フロー	883309	806	2424	16851721	12	1100	91	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	911769	626	1985	16246618	0.756	146	193	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1
Git フロー	250803	3059	1610	29134100	1.2	69	57	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1
GitHub フロー	105062	622	459	13596918	11	160	14	1	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	19256	1662	45	2902916	0.17	12	71	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	42205	909	231	15794687	0.7	46	66	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	54218	2395	323	25604035	0.095	16	169	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	217987	1223	2401	18870964	1.3	18	14	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	11740	1099	167	1551457	0.59	11	19	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	23918	495	238	2161679	2.6	49	19	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	1170	1781	18	128651	0.03	0.66	22	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

図 5.22 訓練データ-6

Workflow	行数	プロジェクト経過日数	ファイル数	バイト数	1人当たりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VerL	Julia	C++	Fortran	LiveScript	Objective-C	EmacsLisp	Lua	Go	Java Setup	IDL	REBOL	Other
日本CAWフロー	1157	472	9	1132534	0.3	2	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Gitフロー	8276	792	134	1285551	2	15	5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
GitHubフロー	11359	587	67	2532236	1.3	19	15	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gitフロー	24972	1935	150	8879429	0.21	13	81	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
日本CAWフロー	45973	1771	1077	10465517	0.184	25.9	142	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gitフロー	1866	78	1254951	0.18	51	28	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Gitフロー	7165	214	319	24549103	0.156	28.5	170	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
LINEフロー	161332	1640	600	73850444	2.5	89	39	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

図 5.23 テストデータ-6

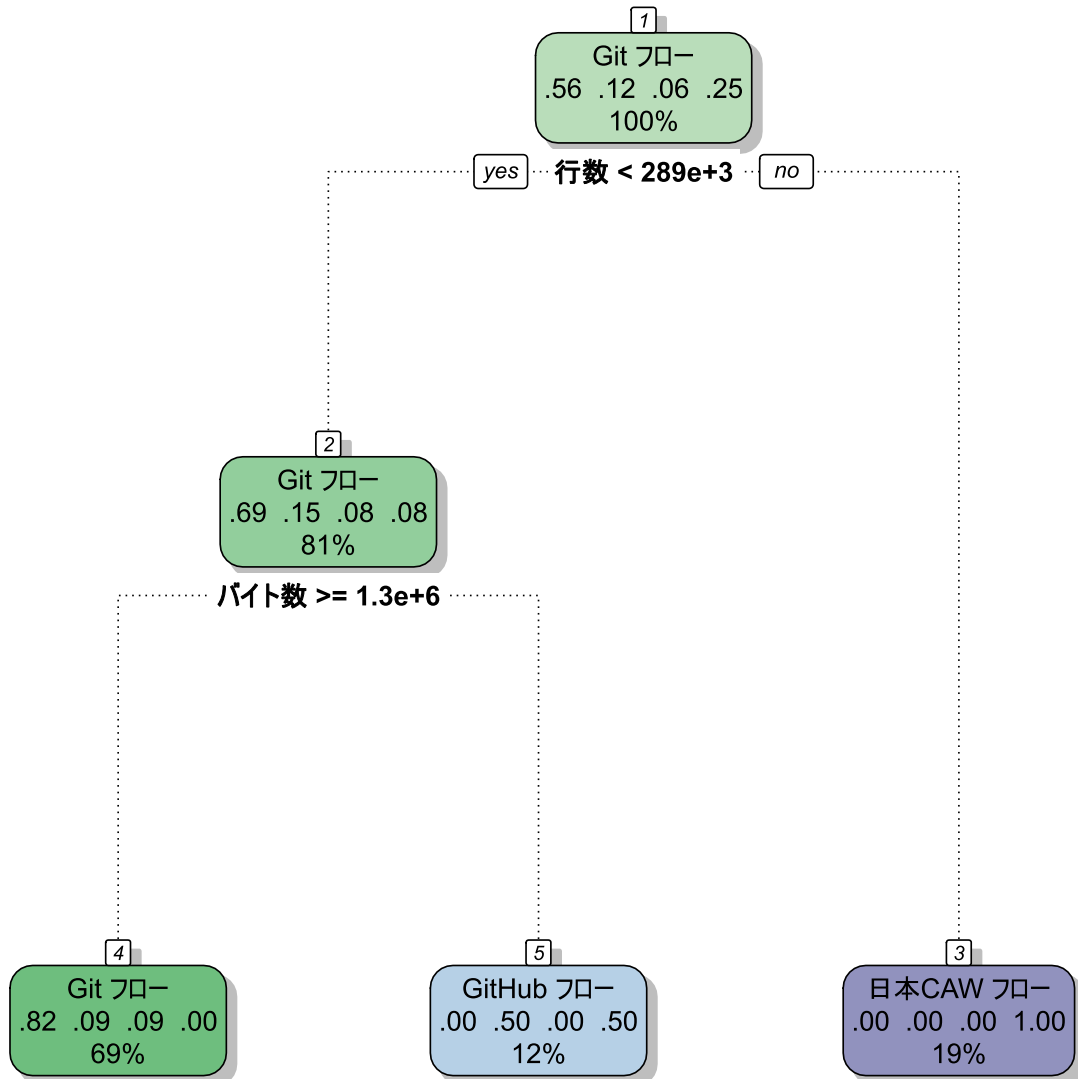


図 5.24 性能測定用決定木-6

5.3.7 7 回目

7 回目に、決定木分析に使ったデータは、図 5.25 である。図 5.25 の決定木分析結果は、図 5.27 である。図 5.27 の性能を測定するのに使ったテストデータは、図 5.26 である。

7 回目の性能測定の結果、精精度は、25.0 % だった。再現率は、50.0 % だった。

workflow	行数	プロジェクト経過日数	ファイル数	バイト数	1人日あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Brainf	LuaScript	Objective-C	EmacsLisp	Lua	Cmake	Inno Setup	SQL	NEOS	Other
GitHub フロー	109062	622	459	13359619	11	160	14	1	1	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	15297	2212	80	1445595	0.091	5.9	76	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
Git フロー	6398	1519	152	1102780	0.13	42	32	1	1	1	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
Git フロー	42206	809	231	19794697	0.7	46	66	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	9568	1882	76	17594951	0.18	5.1	28	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	56567	802	512	14248751	1.8	71	40	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	326788	1113	1202	54265378	21	290	14	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0
Git フロー	11359	597	67	2632098	1.3	19	15	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	1170	1781	16	129651	0.03	0.66	22	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	11740	1099	167	1551457	0.98	11	19	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	1157	472	9	1132324	0.3	2	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Git フロー	9137	814	214	1125424	0.41	11	27	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	54218	3395	323	25964035	0.095	16	169	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	8276	792	134	1395553	2	10	5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Git フロー	217887	1220	2401	18870384	1.3	18	14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	18256	1692	45	2003916	0.17	12	71	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

図 5.25 訓練データ-7

workflow	行数	プロジェクト経過日数	ファイル数	バイト数	1人日あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Brainf	LuaScript	Objective-C	EmacsLisp	Lua	Cmake	Inno Setup	SQL	NEOS	Other		
Git フロー	23918	485	201	2161879	2.6	89	19	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Git フロー	24372	1933	150	8879431	0.21	13	61	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
Git フロー	45673	1771	1077	14865217	0.184	29	140	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Git フロー	71955	2714	319	21424103	0.196	26	170	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
Git フロー	250603	3959	1619	29134100	1.2	69	57	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	
Git フロー	911769	629	1985	96249519	0.759	146	193	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
Git フロー	161332	1840	600	73952444	2.5	96	39	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	863309	806	2424	39957521	1.2	1100	81	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

図 5.26 テストデータ-7

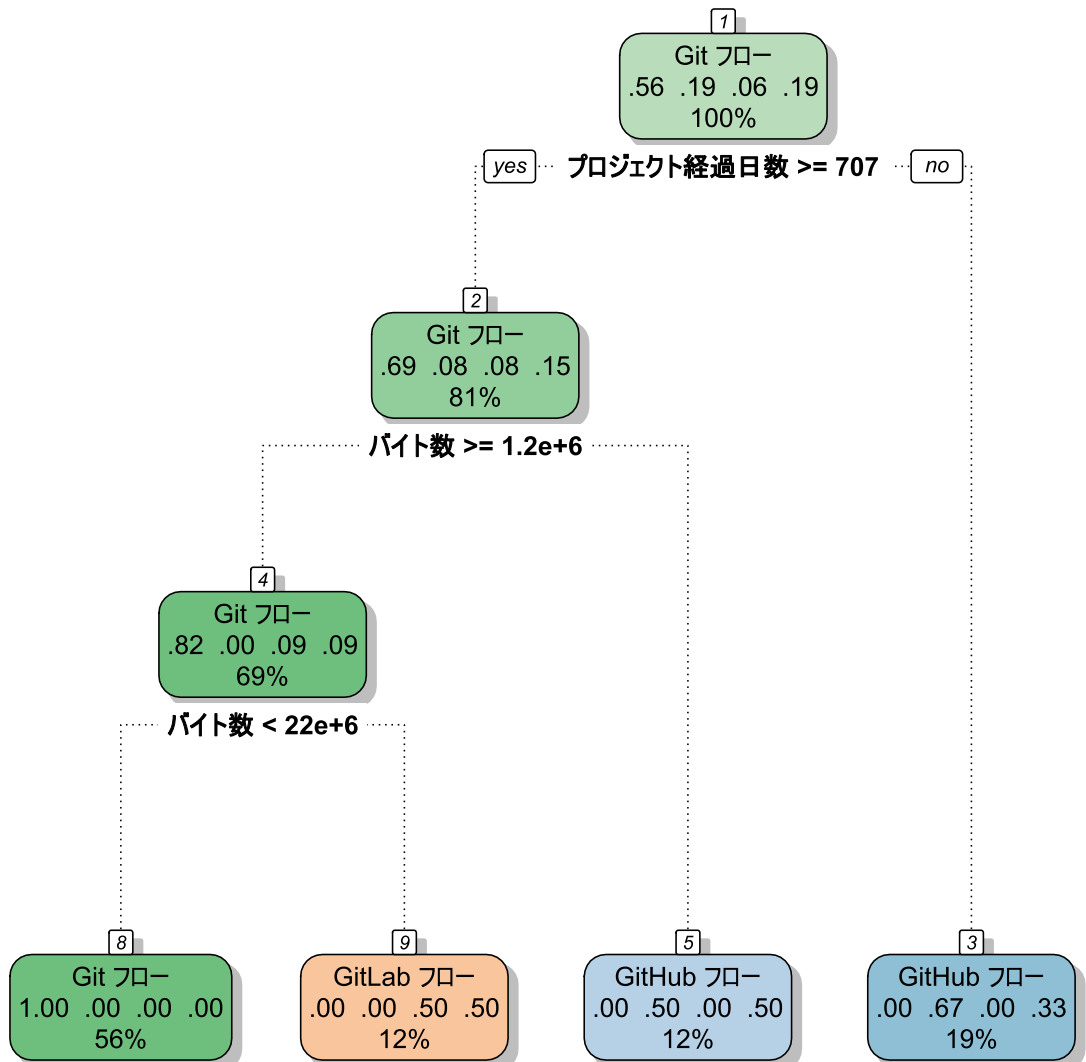


図 5.27 性能測定用決定木-7

5.3.8 8 回目

8 回目に、決定木分析に使ったデータは、図 5.28 である。図 5.28 の決定木分析結果は、図 5.30 である。図 5.30 の性能を測定するのに使ったテストデータは、図 5.29 である。

8 回目の性能測定の結果、精精度は、37.5 % だった。再現率は、42.9 % だった。

workflow	行数	プロジェクト経過日数	ファイル数	バイト数	1人日あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Ynn Setup	IDL	NEBS	Other
GR プロ	6398	1919	152	1102790	0.19	4.2	32	1	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
日教AW プロ	326788	1119	1570	24205376	21	290	14	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0
GA Lab プロ	54518	3395	323	25664038	0.095	16	169	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日教AW プロ	9137	814	214	1122424	0.41	11	27	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GR プロ	8276	792	134	1295551	2	10	5	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
GR プロ	56967	802	912	14243763	1.6	71	40	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GR プロ	42205	809	231	19794667	0.7	46	66	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA Lab プロ	11269	597	67	2622296	1.3	19	15	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GR プロ	15297	2210	80	14745686	0.091	6.9	76	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
日教AW プロ	863309	806	2424	6882721	12	1100	81	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GR プロ	24572	1933	150	8879469	0.21	13	61	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GR プロ	11740	1099	167	1551457	0.58	11	19	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日教AW プロ	811369	626	1965	56246519	0.796	146	193	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1
GR プロ	9566	1882	76	17264851	0.18	51	28	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA Lab プロ	11170	1791	18	129651	0.03	0.66	22	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日教AW プロ	1157	472	9	1132534	0.3	2	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

図 5.28 訓練データ-8

workflow	行数	プロジェクト経過日数	ファイル数	バイト数	1人日あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Ynn Setup	IDL	NEBS	Other
GR プロ	19256	1662	45	2903916	0.17	12	71	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GR プロ	22918	465	238	2142470	2.6	49	19	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GR プロ	71965	2714	319	21424703	0.156	253	170	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GA Lab プロ	101362	622	459	13295919	11	150	14	1	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
LINE プロ	101332	1940	600	73950444	2.5	89	39	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
日教AW プロ	45673	1771	1077	10485517	0.184	25.8	140	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GR プロ	250903	2659	1910	29134100	1.2	69	57	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	1
GR プロ	217667	1220	2401	18970364	1.3	19	14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

図 5.29 テストデータ-8

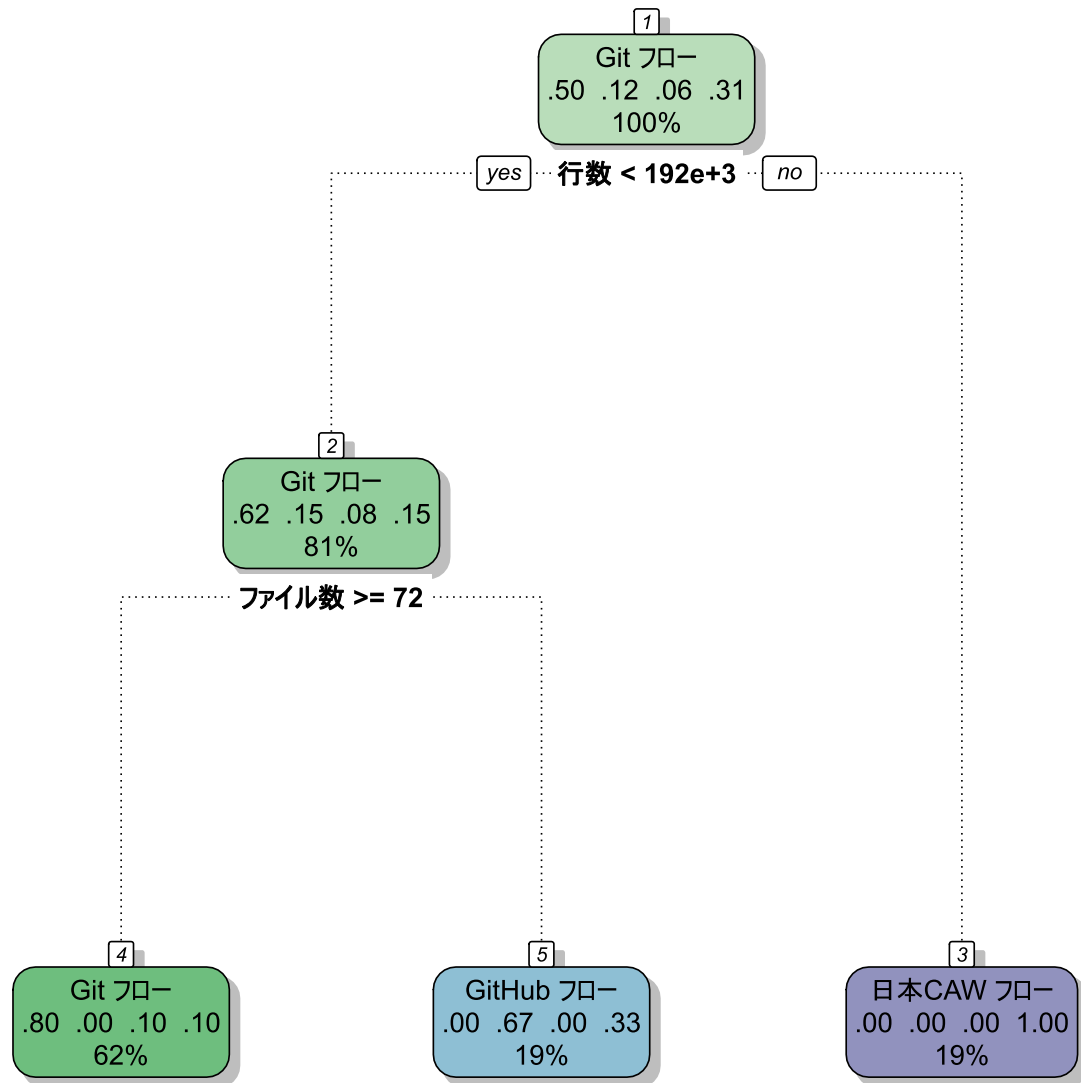


図 5.30 性能測定用決定木-8

5.3.9 9 回目

9 回目に、決定木分析に使ったデータは、図 5.31 である。図 5.31 の決定木分析結果は、図 5.33 である。図 5.33 の性能を測定するのに使ったテストデータは、図 5.32 である。

9 回目の性能測定の結果、精精度は、50.0 % だった。再現率は、57.1 % だった。

workflow	行数	プロジェクト経過日数	ファイル数	バッチ数	1人あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Material	C#	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	Visual	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Inno Setup	SQL	NEOS	Other
日本CAW フロー	9137	894	214	1123634	0.41	11	27	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	217687	1220	2401	18870364	1.3	18	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	8076	782	134	1205551	2	10	5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	56567	802	512	1445757	1.8	71	40	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	24372	1933	150	8279431	0.21	13	51	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	23616	495	226	2161576	2.6	49	19	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	1157	472	9	1132534	0.3	2	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	45672	1771	1077	1066531	0.84	25	140	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	11740	1589	167	1551457	0.98	11	19	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	11130	1781	18	125953	0.93	0	65	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	911769	623	1985	95649519	0.796	146	193	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	8968	1882	76	7259481	0.19	51	28	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	326788	1113	1370	24325376	21	290	14	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GA フロー	42006	930	231	19794697	0.7	46	66	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LINE フロー	161352	1640	600	73953444	2.5	86	39	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	11359	597	67	3532398	1.3	19	15	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

図 5.31 訓練データ-9

Workflow	行数	プロジェクト経過日数	ファイル数	バイト数	1人当たりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	Visual	Julia	C++	Fortran	LiveScript	Objective-C	Emacs Lisp	Lua	Creole	Java Setup	IDL	REBOL	Other
Git フロー	19256	1662	45	2903916	617	12	71	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	6396	1516	152	11027930	613	42	32	1	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	101062	622	459	13299619	11	160	14	1	1	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	15297	2210	80	14145986	0381	69	76	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
Git フロー	71985	2714	319	21424103	0156	255	170	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GitHub フロー	54318	3386	323	25964020	0386	19	169	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	250003	3659	1610	220334100	12	69	57	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	1
日本CAW フロー	865308	806	2424	88952721	12	1100	91	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

図 5.32 テストデータ-9

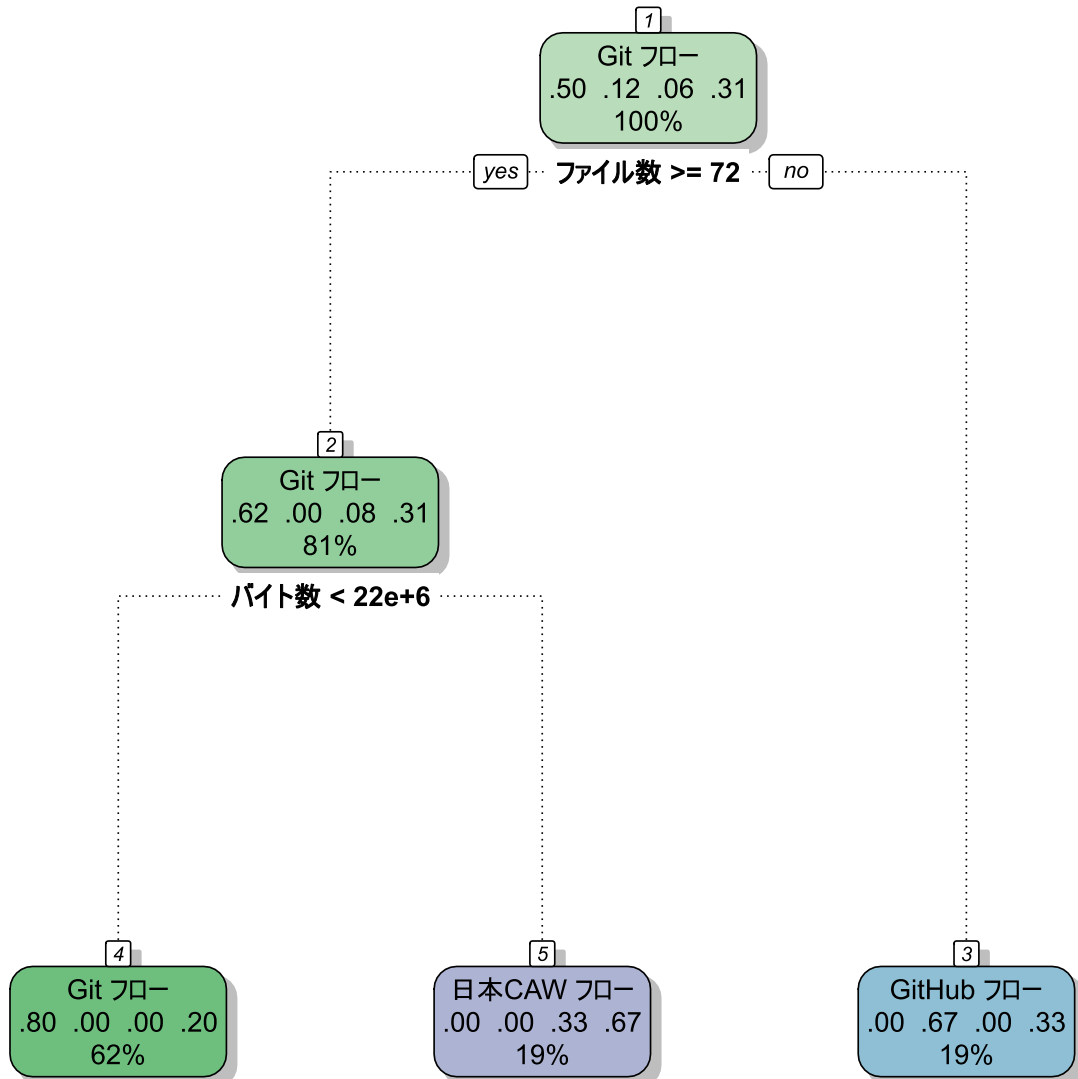


図 5.33 性能測定用決定木-9

5.3.10 10 回目

10 回目に、決定木分析に使ったデータは、図 5.34 である。図 5.34 の決定木分析結果は、図 5.36 である。図 5.36 の性能を測定するのに使ったテストデータは、図 5.35 である。

10 回目の性能測定の結果、精精度は、25.0 % だった。再現率は、28.6 % だった。

Workflow	行数	プロジェクト経過日数	ファイル数	バイン	1人日あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Yoro Setup	IDL	NEBS	Other
Git フロー	6398	1519	152	11027830	0.13	4.2	32	1	1	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	19254	1692	45	2903916	0.17	12	71	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	15297	2210	80	14745696	0.091	6.9	76	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0
Git フロー	24372	1933	150	8879431	0.21	13	61	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	42036	309	231	17795697	0.7	46	66	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	911769	626	1985	96246619	0.756	146	193	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1	1
GitHub フロー	110062	622	459	13359619	11	160	14	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	217687	1220	2401	18870364	1.3	18	14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	863369	808	2434	98857721	12	1100	91	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
GitHub フロー	45673	1771	1077	10495917	0.184	25.8	140	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	34218	3395	323	25664026	0.095	16	169	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	11359	597	67	2653286	1.3	19	15	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	8568	1892	76	17294851	0.18	51	28	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	326788	1113	1370	14365370	21	290	14	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0
Git フロー	71855	2714	319	17424108	0.156	29.5	170	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	8276	792	134	1285591	2	10	5	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

図 5.34 訓練データ-10

Workflow	行数	プロジェクト経過日数	ファイル数	バイン	1人日あたりの行数	1日あたりの行数	人数	HTML	JavaScript	CSS	Markdown	C	PHP	Java	Python	Ruby	TeX	CoffeeScript	Perl	Shell	VimL	Julia	C++	Erlang	LiveScript	Objective-C	EmacsLisp	Lua	Cmake	Yoro Setup	IDL	NEBS	Other
日本CAW フロー	1157	472	9	1133234	0.3	2	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GitHub フロー	1170	1791	18	129951	0.03	0.66	22	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	11740	1086	47	1501451	0.06	11	19	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
日本CAW フロー	9137	814	214	1123424	0.41	11	27	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	23916	485	238	2191979	2.6	49	19	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Git フロー	56567	802	512	14245757	1.8	71	40	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
LINE フロー	181332	1640	600	73852444	2.5	89	29	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Git フロー	250603	3659	1810	29134700	1.2	69	37	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1	1

図 5.35 テストデータ-10

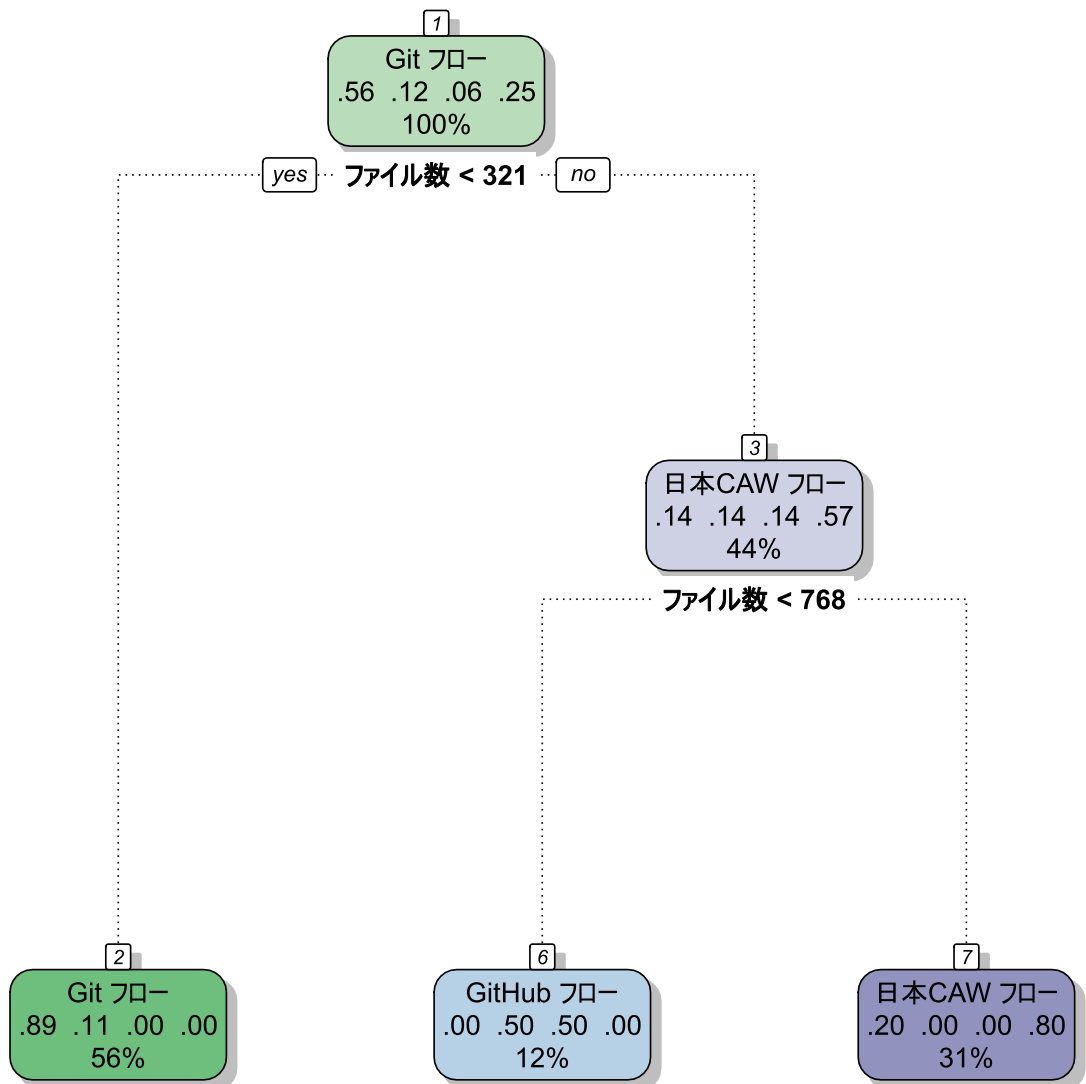


図 5.36 性能測定用決定木-10

5.3.11 平均と信頼区間

精度の平均は、37.5% であった。精度の信頼区間は、32.6 42.4% であった。再現率の平均は、45.9% であった。再現率の信頼区間は、40.7 51.1% であった。

第 6 章

考察

6.1 すべてのデータで行った場合

図 5.3 を考察する．図 5.3 は，図 4.17 を用いて分析を行った結果である．

図 5.3 より，Star 数，人数，OpenPullRequest 数，Watch 数，ファイル数，branch 数，JavaScript で開発フローを再現できることがわかった．また，リーフは，Git フローと GitHub フローと LINE フローと日本 CAW フローと GitLab フローである．

それぞれの分岐要因を考察していく．

Star 数により，選択するフローが変わっている．ここから，開発に直接かかわる人だけでフローが決まらないことが言える．これは，開発に関心がある人，注目している人も含めて，フローを選択しなければならないことが，想定される．

人数により，選択するフローが変わっている．変わっているのは，Git フローと日本 CAW フローである．Git フローと日本 CAW フローの大きな違いは，複雑度である．人数が 84 人以上の場合，複雑すぎると開発しづらいことから，より複雑度が低い日本 CAW フローを選択していることが，想定される．

1 日あたりの commit 数により，選択するフローが変わっている．ここからプロジェクトの生産性により，適正な開発フローが異なることが言える．3 番目に重要な指標となっていることから，ほかの生産性の指標を用いることで，より正確なフローを選択できるようになるだろう．

6.2 データをランダムに並び替え，2 つに分けた場合

図 5.9 と図 5.12 を比較して，考察する．図 5.9 は，図 4.18 を用いて分析を行った結果である．図 5.12 は，図 4.19 を用いて分析を行った結果である．

図 5.9 は，一日あたりの commit 数と ClosedPullRequest 数で，分岐している．また，リーフは，Git フローと GitHub フローと日本 CAW フローである．

図 5.12 は，Watch 数と branch 数と人数で，分岐している．また，リーフは，Git フローと GitHub フローと日本 CAW フローである．

2 つの決定木を見比べると，分岐が大きく異なっていることがわかる．

大きく異なっていることから，16 件に分けられたデータのばらつきが大きいことがわか

る．ここから，精度と再現率が低い可能性があることがわかる．

そのため，決定木の性能測定を行う必要がある．

以上までの考察を含めて，以下の改善を行う．

1 つ目は，指標を絞る．ばらつきが大きいことがわかったため，指標を絞ることにする．指標を絞ることで，ばらつきを減らし，精度と再現率を上げられると想定する．

指標は，プロジェクト開始時にわかっているものにした．プロジェクト開始時にわかっているものは，行数と人数と言語と生産性と規模である．

2 つ目は，成功しているプロジェクトに絞る．成功していないプロジェクトを含めると，最適なフローを選択できていない可能性がある．そのため，データに矛盾が生じている可能性が考えられる．データに矛盾があると，決定木がうまく再現できない．

成功しているプロジェクトの基準は，リリースを 1 以上していることにする．

6.3 コントロールできる要因のみかつ，成功しているプロジェクトの場合

図 5.6 を考察する．図 5.6 は，図 4.20 を用いて分析を行った結果である．

図 5.6 は，行数とバイト数と 1 日あたりの行数で，分岐している．行数とバイト数は，プロジェクトの規模に関わる．1 日あたりの行数は，プロジェクトの生産性に関わる．

分岐より，プロジェクトの規模と生産性が，重要な指標であることがわかった．

プロジェクトが大きくなるにつれて，管理が難しくなる．その管理を助けるためにフローを変える必要があるのだろう．

プロジェクトの生産性は，作業の多さに関わる．テストが多かったり，難しい開発であれば，生産性は大きく変わる．プロジェクトが進むスピードに比べて，複雑すぎるフローだと，開発を送らせてしまう．簡単すぎるフローだと，開発を進ませすぎてしまい，バグが多く入ってしまう可能性が考えられる．

6.4 精度と再現率

精度と再現率は，あまりよくない結果になった．

よくない結果になった要因は，人手で選んだ開発フローが間違えている，プロジェクトの重要な指標を調査できていない等があげられる．

人手で選んだ開発フローが間違えていることの対処は，第三者に判定してもらう方法がある．一人が選択したより，二人が選択したほうが精度は高くなるだろう．

プロジェクトの重要な指標を調査できていないことの対処は，プロジェクトの生産性に注目する．現在では，日数で割った指標しか用いていない．ここまでの分析で，プロジェクトの生産性が，要因になることがわかった．そのため，人数の増減傾向や，特定の曜日に開発が進む等，より詳細なプロジェクトの生産性を表していない．人数の増減傾向は，人がどのタイミングで増減したのかわかれば，指標として用いることが可能だろう．また，ソフトウェア開発プロジェクトは，開発が進むにつれて人数の変化がある．開発段階は人数が増加する．納入段階に近づくにつれて，減少が起こる．特定な曜日に開発が進み，特

定の曜日にテストを行うプロジェクトもある。

より詳細な指標を用いれば、新しい要因を作ることが出来、精度と再現率を上げることができるだろう。

第 7 章

結論

本研究により，開発フローを再現手法を提案した．GitHub 上のプロジェクトから性質と開発フローを調査し，決定木分析を行うことで，再現した．

この再現手法は，精度と再現率が高いとはいえない．精度は，37.5% だった．再現率は，45.9% だった．

しかし，精度と再現率を高めることができれば，開発経験の浅いプロジェクトでも，最適なフローを選択できるようになるだろう．

参考文献

- [1] 小野寺航己. バージョン管理システムを活用するソフトウェア開発の開発フロー. 卒業論文, 千葉工業大学, 2015.
- [2] 池田尚史, 藤倉和明, 井上史彰. チーム開発実践入門～共同作業を円滑に行うツール・メソッド. 技術評論社, 2014.
- [3] 松田航. システム開発の質を高めるバージョン管理ツールとは? <http://www.linuxacademy.ne.jp/lablog/programmer/319/> (2016.01.22 閲覧).
- [4] Ken Nishimura. Gitlab flow から学ぶワークフローの実践. <http://postd.cc/gitlab-flow/> (2016.01.22 閲覧).
- [5] 可知豊. アップルが swift の oss 化で狙うもの-github で企業とコミュニティの連携が進む. <http://japan.zdnet.com/article/35075308/> (2016.01.22 閲覧).
- [6] 野々下裕子. Github 共同創業者が和歌山県庁を表敬訪問し知事と対談. <http://japan.cnet.com/news/business/35075525/> (2016.01.22 閲覧).
- [7] 野々下裕子. 初心者から参加できる github 習得イベント「github patchwork」が神戸で開催. <http://japan.cnet.com/news/business/35075265/> (2016.01.22 閲覧).
- [8] Charlie Osborne. Github のサポート体制にユーザーが苦情-「完全に無視されている」. <http://japan.zdnet.com/article/35076469/> (2016.01.22 閲覧).
- [9] 大塚弘記. GitHub 実践入門 Pull Request による開発の変革. 技術評論社, 2014.
- [10] harada4atsushi. Github flow で pull request ベースな開発フローの進め方. <http://qiita.com/harada4atsushi/items/527d5f98320d993b3072> (2015.10.03 閲覧).
- [11] hayaishi. Line ios アプリ開発についてのご紹介. <http://developers.linecorp.com/blog/?p=2921> (2015.10.03 閲覧).
- [12] 竹内俊彦. はじめての S-PLUS/R 言語プログラミング 例題で学ぶ S-PLUS/R 言語の基本. オーム社, 2005.