Vol.2022-SE-210 No.18
2022/3/11

DVCリポジトリにおける MLパイプラインの進化に関する調査

中村 悠人[1,a)]　松尾 春紀[1,b)]　松田 雄河[1,c)]　近藤 将成[1,d)]　亀井 靖高[1,e)]　鵜林 尚靖[1,f)]

概要：機械学習の開発を効率的に進めるための手法の一つとして ML パイプラインが挙げられる．ML パイプラインとは，ステージと呼ばれる，機械学習開発で行われるデータ取得，特徴づけ，学習などの工程を 1 つにつなげ，順番に自動で実行する機能である．ML パイプラインを実現するアプリケーションとして，Data Version Control (DVC) が挙げられる．DVC は，ML パイプラインに加えて，データバージョン管理やリモートリポジトリ管理が可能で，有用な機能を多く持っているため GitHub 上のプロジェクトでよく利用される．しかし，実際にどのような ML パイプラインが作成されているかはあまり知られていない．これらの知見を得ることは，最適な ML パイプラインの選択や，有効でない ML パイプラインを除外する上で重要であると考える．そこで本研究では，DVC を使用しているプロジェクト内の 540 個の ML パイプラインを対象に，ML パイプラインファイル内のステージの変化に関する調査を行い，推奨される ML パイプライン作成の指針を提示する．調査の結果，プロジェクト存在期間にパイプラインファイルが変更された回数は平均 2.9 回，中央値 3.0 回であった．また，パイプラインファイルが編集されたコミット全てを対象とすると，モデル学習ステージの変更が最も多く，データ取得ステージでの変更が最も少なかった．さらに，ML パイプラインをコミット数により 3 つのパターンに分けたとき，最終的な ML パイプラインはどのパターンもデータ取得，モデル学習，モデル評価の構成であるものが最も多いということがわかった．

キーワード：Data Version Control, データバージョン管理, ML パイプライン, MLOps

1. はじめに

近年，機械学習はより一層関心を集めており，さまざまな製品へと実用化されている．機械学習モデルの作成にはデータ取得，モデル学習などさまざまな工程がある [1]．これらをつなげた一連の流れである機械学習ワークフローは，工程が多くなると複雑で管理が難しくなる．また，1 回の学習ではワークフロー内の工程を 1 つずつ手動で実行する．それに加え，1 つの工程には機械学習用データ，実行するコード，パラメータなどが含まれており，機械学習の精度向上のためにはこれらを変更しながら学習を繰り返し実行する必要がある．このため，開発効率が低下したり人為的なミスを誘発する可能性がある．以上より，ワークフローの管理は機械学習で重要な位置を占めるといえる．

これらの機械学習の課題点を支援する手法として，ML パイプラインがある．ML パイプラインは，機械学習における一つの工程をステージとして，それらをつなげた機械学習ワークフローを自動で実行できる．この手法を使用することで，複雑なワークフローの管理と，実験の再現性を確保できる．

ML パイプラインの機能を持つアプリケーションとして Data Version Control (DVC) がある [2]．Data Version Control (DVC) は，ML パイプラインを作成できるほか，大量のデータやモデルなどのファイルのバージョン管理を行うことができる．さらに，バージョン管理されているファイルをリモートリポジトリで管理できる．これらの理由により，今後の機械学習開発においてもさらなる導入が期待される．

しかし，DVC における ML パイプラインの構成は機械

[1] 九州大学
Kyushu University
[a)] n.nakamura@posl.ait.kyushu-u.ac.jp
[b)] matsuo@posl.ait.kyushu-u.ac.jp
[c)] matsuda@posl.ait.kyushu-u.ac.jp
[d)] kondo@ait.kyushu-u.ac.jp
[e)] kamei@ait.kyushu-u.ac.jp
[f)] ubayashi@ait.kyushu-u.ac.jp

ⓒ 2022 Information Processing Society of Japan

学習モデルの品質向上において重要であるが，実際にどのように ML パイプラインが変更され，作成されているのか知られていない．実際に使用されている ML パイプラインを調査することで，最適な ML パイプラインの選択や，有効でない ML パイプラインの除外，および ML パイプライン変更のアンチパターンの知見の蓄積として有用であると考える．また，ML パイプラインは技術的負債と呼ばれるコードに存在するバグや解消すべき課題が存在している場合があるため [3]，この調査の知見は問題解決に向けた手助けになると考える．DVC の ML パイプラインの研究は，機械学習の精度向上のために重要であると考えられる．

本研究では，DVC における ML パイプラインの変更過程の実態を把握することによって，推奨すべき DVC パイプラインの作成方法を明らかにすることを目的とする．目的を達成するため，以下 3 つの研究課題（RQ）を設定し，GitHub 上のプロジェクトに含まれる 540 個の ML パイプラインを対象に調査を行った．

RQ1: ML パイプラインの変更はどの程度行われているか

プロジェクト存在期間内にパイプラインファイルが変更された回数は平均 2.9 回，中央値 3 回であった．また，1 回のコミットで変更するステージ数は 1 個であるものが最も多かった．

RQ2: ML パイプラインではどのようなステージが変更されているか

ステージの分類の 1 つである Data-Collection に分類されたステージのうち 8 割は 1 度も変更されなかったものだったが，Model-Training に分類されたものは 4 割が追加されたものであった．

RQ3: 現在使用されている ML パイプラインにはどのような特徴が存在するか

ML パイプラインの変更量によってその構造が安定しているか不安定かのパターンに分類し，調査を行った．コミット数の大小によって 3 つのパターンに分けた時，最終的な ML パイプラインはどのパターンもデータ取得，モデル学習，モデル評価で構成されているものが最も多かった．

以降，第 2 節では本研究に関連する研究，第 3 節では本研究の調査手法について説明する．第 4 節，第 5 節では調査結果についての説明とその考察を行う．第 6 節では，妥当性への脅威について述べ，第 7 節で結論と今後の課題について述べる．

2. 関連研究

2.1 ML パイプライン

ML パイプラインは，機械学習の工程をあらかじめ登録しておいた順番に自動で実行できる．機械学習では，ハイパーパラメータは変更を繰り返す．これは，最適なモデルを見つけるために．学習を繰り返すからである．従来の方法では，工程を最初から 1 つずつ手動で実行するためミスを起こす可能性があった．変更を繰り返してもワークフローを変えずに自動で実行できる ML パイプラインは機械学習において重要である．また，ML パイプラインを共有することで同じステップで学習を行えるため，チームで研究や開発をする場合に有用である．ML パイプラインは機械学習自動化の観点や，ビルドやテスト，リリースを自動化する CI/CD の観点からも研究が進められている [4]．

Olson ら [5] は，機械学習自動化ツールである TPOT を開発した．TPOT は，機械学習自動化に必要な ML パイプラインを作成する際に，前処理，特徴量抽出や選択，モデル構築の工程を枝分かれに組み合わせていく．実験では，さまざまなデータセットを TPOT，RandomForest，および異なる最適化手法を用いた TPOT に適用し，学習の精度を比較している．その結果，TPOT は RandomForest と同程度の性能の取得ができることを明らかにしている．また，TPOT により作成されたいくつかの ML パイプラインは，RandomForest よりも精度が飛躍的に向上する前処理とモデル構築の組み合わせを発見したことを示している．さらに，多目的最適化を組み込むことによって TPOT は分類精度を低下させずにコンパクトで解釈が簡単であるパイプラインを作成できることを示している．

また，Drori ら [6] は，機械学習自動化ツールである AlphaD3M を提案している．AlphaD3M は，AlphaZero から発想を得た，ML パイプライン合成を一人用ゲームとして当てはめ，ニューラルネットワークとモンテカルロ木探索で意思決定を行い，挿入，削除，置換の動作を繰り返し ML パイプラインを組み立てていく手法である．この手法の利点は，組み立てる際になぜその行動をするように判断したのかわかるため，完成した ML パイプラインの構造が説明可能であるということである．この AlphaD3M を，既存の機械学習自動化ツールと比較する実験を行っている．その結果，既存のツールと同等の精度を得ることができ，学習時間を短縮できることを

スニペット 1　パイプラインファイルである dvc.yaml の例

```
stages:
  prepare:
    cmd: python src/prepare.py data/data.xml
    deps:
    - data/data.xml
    - src/prepare.py
    params:
    - prepare.seed
    - prepare.split
    outs:
    - data/prepared
```

明らかにした．

2.2 データバージョン管理

データバージョン管理とは，モデルやデータの変更履歴を管理するものである．現在では，ソースコードなど軽量なファイルのみであれば GitHub を使い，Git によりバージョン管理されているファイルを保存，共有することが一般的である．しかし，GitHub には大きなサイズのファイルを置くことができない．データバージョン管理ツールの登場により，機械学習で使用するような大量のデータの管理ができるようになった．

機械学習を用いた研究の正当性の証明や，信頼性の向上のために再現性が重要である．Baker の調査 [7] では，70%以上の研究者が他の研究者の再現実験に失敗しており，さらに 50%以上が自身の研究の再現に失敗したことがあるということが報告されている．機械学習研究の再現性の向上に向けた調査も行われており [8]，今後の課題であると考えられる．

再現性の確保のためには，使用されているコードやデータ，モデルを一緒に管理する必要がある．データバージョン管理ツールによって，学習に使用したファイルをまとめて管理できるため，再現性の問題も解決できる．よって，今後の研究者や開発者の使用が見込まれる．データバージョン管理ツールには DVC のほか，MLflow[9] や Pachyderm[10] などがある．

Data Version Control (DVC)[2] とは，2017 年 5 月 4 日に公開されたオープンソースソフトウェアであり，データやモデルのバージョン管理や，ML パイプラインの作成ができる．DVC では，ML パイプラインの構造は “dvc.yaml” ファイルで管理される．dvc.yaml の例をスニペット 1 に示す． この yaml ファイルにはステージの情報が含まれている．例では，“stages” 内に含まれている “prepare” が一つのステージであり，ステージ名は任意に設定できる．ステージの中には “cmd” という欄があり，ここでは Python ファイルの実行など，コマンドの実行ができる．“deps” では，依存しているファイルを記録しており，ステージが一度実行されていた場合，依存しているファイルに変更がなければ，そのステージは再び実行せずに済む．“params” 欄には，ハイパーパラメータを指定できる．

Barrak ら [11] は，DVC ファイルとその他のファイルとの間で，変更が与える影響に関する実態調査を行っている．まず，GitHub 上にある DVC を使用しているプロジェクトのうちコミット数が多く，プルリクエストが 10 回を超えた 10 件のプロジェクトを対象にしている．その結果，プルリクエストではソースコードが変更されたうちの 1/4，およびテストコードが変更されたうちの 1/2 で，DVC ファイルが変更される要因となっている．次に，DVC を使用しているプロジェクトのうちコミット数が多い 25 件に，ML パイプラインの複雑度が初期コミットから現在までどのような変化をしているかを調査している．その結果，78%のプロジェクトで ML パイプラインの複雑度が一定ではないことを明らかにした．

Zaharia ら [12] は，使用されるツールの多さ，実験の追跡や再現性，及び製品のデプロイの難しさといった機械学習を使用するソフトウェア開発における課題を提示している．また，これらの課題解決のためにデータバージョン管理ツールである MLflow を開発した．Python の関数のように使用できるため，ライブラリの使用制限がなく柔軟な機械学習ソフトウェアの開発工程を作成できることを示している．

3. 調査方法

本節では，実際に使用されている ML パイプラインの構造の変化を調査するための手法を説明する．

3.1 調査課題

本研究では GitHub 上における DVC の ML パイプラインの変更についての調査を行う．本研究は DVC における ML パイプラインの一般的な変更や，構成を明らかにする基礎となる研究であり，DVC をこれから導入する機械学習開発者にとって重要な知見となる．まず DVC パイプラインが過去から現在までどの程度変更されているかを調査する（RQ1）．さらに，パイプラインコミットごとのステージの追加，削除，変更の有無を調査し（RQ2），現在使用されている ML パイプライン構造の知見を得る（RQ3）．

ⓒ 2022 Information Processing Society of Japan

表 1　単語パターンによるステージの分類

分類	単語パターン	分類に含まれるステージの種類
Data-Collection	load,unzip,get,etl,read,pull,make,fetch,create,prepare*data （load*model,create*model は除く）	25
Data-Cleaning	clean	1
Data-Labeling	label	0
Feature-Engineering	feat（ただし，Data-Collection に属している場合，and が含まれていること， train が含まれないこと）	8
Model-Training	train,log,split,prepare*model,load*model,create*model （ただし train が含まれており Data-Collection に属している場合，and が含まれていること train が含まており validate が含まれている場合，and が含まれていること）	47
Model-Evaluation	eval,valid,analyze*model	22

3.2　データセットの構築

3.2.1　対象プロジェクトの選定

本研究では，パイプラインファイルである dvc.yaml という名前のファイルの過去から現在まで，デフォルトのブランチにおける全コミットを調査の対象とする．GitHub 上のプロジェクトのうち，DVC と依存関係にあるプロジェクト 1,725 件の中から，クローンできた dvc.yaml ファイルが含まれるプロジェクト 989 件，ML パイプライン 1,037 件を取得した．

3.2.2　ステージの名前によるフィルタリング

3.2.1 節で取得した ML パイプラインを，本研究では一般的な機械学習開発で使用されている 9 個のステージ分類 [1] のうち，ML パイプラインに関係の深い以下の 6 個に分類する．

- Data-Collection： 学習に必要なデータを収集したり，データを統合し新しいデータセットを作成する．
- Data-Cleaning： ノイズを含むなど正確でないデータを取り除く．
- Data-Labeling： 教師あり学習で使われる正解ラベルを割り当てる．また，強化学習などで使用される，デモデータや報酬を調整する．
- Feature-Engineering : 機械学習モデルに有益である特徴を抽出，選択する．
- Model-Training : データセットと正解ラベルを元にモデルを学習させる．
- Model-Evaluation： 事前に定義された評価指標を使用し，モデルを評価する．

調査に使用される各ステージの分類は，クローンしたパイプラインファイルの作成から現在までに現れるすべてのステージを目視で確認し，その中から関係が深いと思われる単語パターン（表 1）を使用して分類した．各ステージは，表 1 に示す分類のうち，少なくとも 1 つが割り当てられており，2 つ以上のカテゴリに分類されているステージも存在する．全てのコミットから 93 種類のステージが確認された．

3.2.3　メタデータによるフィルタリング

Python は機械学習によく使われるプログラミング言語である．そこで，ML パイプラインのステージが機械学習として実行されているか確認するため，ML パイプライン内のステージの “cmd” 内に Python 実行コマンドが含まれる ML パイプラインを対象にした．さらに，本研究では ML パイプラインの変更過程の実態を把握することによって，推奨すべき DVC パイプラインの作成方法を明らかにすることを目的とするため，

- 作成された時点から現在に至るまで ML パイプラインファイル内が空だったもの
- 作成された時点から現在に至るまでステージが登録されなかったもの
- パイプラインファイルは yaml ファイルのため，yaml 形式に合っておらず，読み込みができなかったもの

を除いた．フィルタリングの結果，537 件のプロジェクトに含まれる 540 件の ML パイプラインを取得した．

4.　調査結果

本節では，3 章で取得した 540 件のパイプラインについて，3 つの RQ の調査結果を報告する．

4.1　RQ1: ML パイプラインの変更はどの程度行われているか

目的： 基本的な機械学習のワークフロー [1] はすでに提

表 2 コミット数ごとのパイプラインファイル件数

コミット数	ファイル数
1	147(27.2%)
2	61(11.3%)
3	143(26.5%)
4	128(23.7%)
5	38(7.0%)
6	7(1.3%)
7	8(1.5%)
8	4(0.7%)
9	3(0.6%)
10	1(0.2%)
合計	540

表 3 コミットにおけるステージ変更数の分布

回数	件数
0	42(4.1%)
1	805(78.4%)
2	126(12.3%)
3	44(4.3%)
4	10(1.0%)
合計	1,027

示されている．そのため，ML パイプラインもワークフローの指針に従って作成されているならば，完成したパイプラインを 1 回のみコミットするはずである．しかし，著者らによって変更が複数回行われている ML パイプラインを確認している．実際に使用されている ML パイプラインはプロジェクトごとに変更頻度が違うと考えられるため調査を行う．具体的には，ML パイプライン 1 つあたりの変更された回数及びコミット 1 つあたりで変更されたステージの件数を調査する．

アプローチ：3 節で取得した，パイプラインファイル 540 件を対象に，その変更ログを追跡し，ML パイプラインが変更された全ての時点でのパイプラインファイルを取得した．このファイルの件数と内容をもとに，ファイル 1 つあたりの変更の回数を調査した．このとき，最初にコミットされたパイプラインファイルの中が空だった場合は考慮に入れず，テキストが追加された時点からの回数を調査した．また，yaml 形式に合っていないファイルがコミットされた場合は除外した．同じように取得したファイルから直前にコミットされたファイルと比較し，1 回のコミットでステージがいくつ変更されるか調べた．

結果： ML パイプラインの変更頻度の結果を表 2 に示す．ML パイプラインごとの現在までの変更回数は，平均 2.9 回，中央値 3.0 回であった． 最も多かったものが 1 回のみの変更で，147 件 (27.2%) だった．その次に多かったのが 3 回の変更で，143 件 (26.5%) だった．また，1 コミットにおけるステージの変更回数の結果を表 3 に示す．1 コミットごとのステージの変更回数は 1 回が最も多く，805 件 (78.4%) だった．

パイプラインファイル 540 件における現在までの変更回数は，平均 2.9 回，中央値 3.0 回であった．また，1,027 件のコミットのうち，1 コミットごとのステージの変更回数は 1 回が最も多く，805 件 (78.4%) であった．

4.2 RQ2: ML パイプラインではどのようなステージが変更されているか

目的：本 RQ では，DVC パイプラインの進化についての具体的な特徴を明らかにする．RQ1 では，変更頻度を調べるための ML パイプラインとそのステージを対象にした変更の有無のみの調査であった．よって本 RQ では，ステージ分類での変更内容を具体的に調査し，変更の特徴についての知見を得る．これは，今後開発者が ML パイプライン作成，変更する際に有益であると考える．

アプローチ： 540 件のパイプラインファイルについて，RQ1 で取得したファイルが変更された時点でのパイプラインファイルの内容を元に調査を行った．まず，初期から現在まで，パイプラインファイルのコミットが行われた時点と 1 つ前のコミットが行われた時点での内容を比較した．また，パイプラインファイルが追加され初期状態になった時点と現在の最新のコミットでの内容を比較した．初期状態とは，ファイルが空ではなく，さらに yaml ファイルとして読み込み可能な状態を指す．

比較した情報より，ステージの追加や削除，変更の有無を調査した．ここでの変更とは，ステージの追加と削除を除く，ステージが直前のコミットと現在のコミットで存在し，内容のみが変化したものを指す．また，追加および削除は，比較するファイルによるステージの名前の存在の有無のみにより分析した．最後に，対象のステージを 3 節で行った単語パターンに従い分類した．

結果： 結果を表 4 に示す． コミットが行われた時点のファイルと 1 つ前のコミットとの比較では，追加，変更，削除のうち，ステージの追加が最も多かった．ステージが追加されたもののうち，最も多かったステージ分類は Model-Training の 658 件（61.3%）であり，2 番目に多かった分類は Model-Evaluation の 324 件 (30.2%) であった．削除されたステージでも同様に Model-Training が 72 件 (61.5%) と最も多く，次いで Model-Evaluation が 32 件 (27.4%) と 2 番目に多かった．削除されたステージは全体で 117 件であり，追加や変更がなされたステージよりも少なかった．変更されたステージにおいても Model-Training が 257 件 (60.5%) と最も多く，

表 4　分類ごとのステージの変更

分類	1つ前のコミットとの比較				初期と最終のコミットとの比較			
	追加	削除	変更	編集なし	追加	削除	変更	編集なし
Data-Collection	87	9	62	804	81	3	38	256
Data-Cleaning	1	0	2	8	1	0	1	0
Data-Labeling	0	0	0	0	0	0	0	0
Feature-Engineering	3	0	3	7	3	0	0	2
Model-Training	658	72	257	501	593	7	50	84
Model-Evaluation	324	36	104	57	292	4	21	27

次いで Model-Evaluation が 104 件 (24.3%) と 2 番目に多かった．また，編集されなかったステージの中では Data-Collection が最も多く，804 件 (58.4%) であった．

初期状態と現在のコミットとの比較調査でも，追加，変更，削除のうち，ステージの追加が最も多く，変更されなかったステージも含めた全 2,995 件のうち，970 件 (32.4%) を占めていた．追加されたステージのうち，Model-Training に分類されるステージが最も多く 593 件 (61.1%) であり，次いで Model-Evaluation の 292 件 (30.1%) であった．削除されたステージは 14 件と少なく，最も多い Model-Training で 7 件 (50.0%) であった．変更されたステージも Model-Training が最も多く 50 件 (45.5%) であり，次いで Data-Collection が 38 件 (34.5%) と 2 番目に多かった．

ステージの編集は，Model-Training の工程が多くを占めた．対照的に，Data-Collection に属するステージは編集が少なかった．

4.3 RQ3: 現在使用されている ML パイプラインにはどのような特徴が存在するか

目的：本 RQ では実務においてどういった ML パイプラインが使われているのかの初期調査を行う．また，分析の際に，「変更が少ないパイプラインは，実務に沿った形で収束したパイプラインである」という仮説を立て検証した．具体的には，パイプラインを変更量によって分類し，変更量によって使われているパイプラインに差があるのか調査した．この調査をすることで，他の実務者がパイプラインを導入する際に，どういったパイプラインを導入すれば良いかの知見となる．そういった知見は，導入コストやパイプラインの変更コストの削減にも繋がると期待できる．

アプローチ：この RQ では，RQ1 のパイプラインファイルのコミット数の結果を基に第 1 四分位数と第 3 四分位数により，コミットが 1 回のみと安定しているパターン 1，2〜4 回と標準的なコミット数であるパターン 2，変更が 5 回以上と不安定であるパターン 3 の 3 つのパターンに分類し以下の調査を行った．なおこの分類は初期分析のための簡易的なものであり，今後より厳密な分類においての調査も行う予定である．

表 5　収束した ML パイプラインのステージ分布

分類	パターン 1	パターン 2	パターン 3
Data-Collection	133(31.0%)	319(26.3%)	56(23.7%)
Data-Cleaning	0	0	2(0.8%)
Data-Labeling	0	0	0
Feature-Engineering	9(2.1%)	3(0.2%)	2(0.8%)
Model-Training	195(45.5%)	606(50.0%)	121(51.3%)
Model-Evaluation	92(21.4%)	285(23.5%)	55(23.3%)
合計	429	1,213	236

RQ1，RQ2 で使用したパイプラインファイル 540 件から，最終的にコミットされている ML パイプラインを取得した．これらの ML パイプラインを対象に，パターンごとに使用されている各ステージの分類された個数と組み合わせを調査した．

結果：構成されているステージの分類結果を表 5 に示す．パターン 1 では Data-Collection の割合がパターン 2，3 よりも多かった．また，パターン 3 にのみ Data-Cleaning のコミットが存在しており，Feature-Engineering のステージは，パターン 1 で占める割合がパターン 2，3 より大きかった．

また，各パターンで確認できた ML パイプラインのステージ構成の分布を表 6 に示す．パターンごとの ML パイプラインの構成を見てみると，3 パターンとも Data-Collection と Model-Training と Model-Evaluation で構成されている ML パイプラインが半数以上を占めた．パターン 1 では Data-Collection のみの ML パイプラインが他のパターンよりも多かった．

ⓒ 2022 Information Processing Society of Japan

表 6　パターン別ステージ構成

ML パイプライン構成	パターン 1	パターン 2	パターン 3
Data-Collection	42(28.6%)	8(2.4%)	2(3.3%)
Data-Collection, Data-Cleaning,Model-Training, Model-Evaluation	0	0	1(1.6%)
Data-Collection, Feature-Engineering	1(0.7%)	0	0
Data-Collection, Feature-Engineering, Model-training, Model-Evaluation	5(3.4%)	1(0.3%)	1(1.6%)
Data-Collection, Model-Training	4(2.7%)	27(8.1%)	1(1.6%)
Data-Collection, Model-Training, Model-Evaluation	80(54.4%)	279(84.0%)	50(82.0%)
Data-Cleaning, Model-Training, Model-Evaluation	0	0	1(1.6%)
Feature-Engineering	1(0.7%)	0	0
Feature-Engineering, Model-Training	1(0.7%)	0	0
Feature-Engineering, Model-Training, Model-Evaluation	1(0.7%)	1(0.3%)	1(1.6%)
Model-Training	8(5.4%)	10(3.0%)	3(4.9%)
Model-Training, Model-Evaluation	4(2.7%)	3(0.9%)	1(1.6%)
その他	0	3(0.9%)	0
合計	147	332	61

パイプラインファイルのコミット数に関係なく最も多い ML パイプラインは, Data-Collection と Model-Training と Model-Evaluation で構成されているものであった.

5. 考察

本稿では，DVC における ML パイプラインについて調査した．本節では，DVC における ML パイプラインの推奨すべき作成方法を探るため，調査結果について考察する．

RQ3 の結果より，Data-Collection，Model-Training，Model-Evaluation の 3 つのステージからなる ML パイプラインが最も多かった．このことから，実務においては表 1 に示している 6 つの工程全てを ML パイプラインとして自動化しているわけではないことが伺える．

さらに RQ2 では，ステージ追加のためのコミットが多いという結果が得られた．RQ1 よりコミット数は中央値で 3.0 回であり，各コミットにおいて変更されるステージは 1 個である場合が多かった．このことから，実務において，ML パイプラインを作成する際には，いきなり全ての工程を自動化するわけではなく，1 つずつステージをパイプラインに追加し，自動化している場合が多い．

これらを考慮すると，Data-Collection，Model-Training，Model-Evaluation の 3 つのステージをパイプラインに組み込み自動化すると良いと考えられる．また，その際にも，3 つのステージを一度にパイプラインに組み込むのではなく，1 つずつ組み込んでいくと良いと考えられる．

全工程をパイプラインとして一度に自動化するのではなく，自動化しやすい工程を順次ステージとしてパイプラインに組み込み自動化すると良い．また，全ての工程を自動化しなくとも良い.

6. 妥当性への脅威

6.1 内的妥当性

本研究では，ステージ内の “cmd” で Python を実行している DVC パイプラインを対象にした．しかし，“cmd” 内にはシェルスクリプトも実行できるため，シェルスクリプト内で Python を実行している可能性がある．そのため，対象にすべきファイルを取得できていない可能性がある．

また，ステージの分類は名前のみにより決定しているため，誤った分類が存在している可能性がある．

さらに，RQ3 では，最終的にコミットされたパイプラインファイルを収束した ML パイプラインとして取得したが，実際には ML パイプラインが完成しているとは限らない．実際，RQ3 では Data-Collection ステージのみの ML パイプラインが存在することから，これから変更されることが予想される．

ⓒ 2022 Information Processing Society of Japan

6.2 外的妥当性

本研究では，GitHub の DVC を使用しているプロジェクトより，ステージ名と実行コマンドによってフィルタリングした ML パイプライン 540 件を対象にしているが，調査結果の一般化のためには，より多くの DVC パイプラインを含むプロジェクトを対象に，同様の調査を行う必要がある．

7. おわりに

本研究では，DVC における ML パイプラインの進化についての実態調査として，実際に使われている ML パイプラインの工程ごとのステージ変更や，コミット数による構成の違いについて分析した．その結果，パイプラインファイルは初期状態としてコミットされてから現在まで変更されなかったものが最も多く，次に 3 回編集されたものが多かった．また 1 回のコミットで 1 つのステージのみ編集されているものが多数を占め，1 回のコミットで複数のステージを変更したものは少数だった．ステージの変更については，ステージ追加のコミットが最も多く，追加されたステージのうち Model-Training が最も多かった．編集されなかったステージでは Data-Collection が最も多かった．さらに，ML パイプラインは，Data-Collection，Model-Training，および Model-Evaluation で構成されているものが最も多いが，コミット回数が 1 回のみの ML パイプラインではこの構成の割合が低かった．本研究結果より，推奨すべき DVC パイプラインの作成方法を明らかにした．

今後の展望として，3.2.3 節で行った分類では，Python ファイルが含まれていることを条件としているが，コードファイルの中身を分析対象としていない．そのため，ML パイプライン内で使用されているコードを分析し，ML パイプラインの変更とどのように関わりあっているかを調査する必要がある．RQ2 で行った ML パイプラインの変更の調査では，ML パイプラインステージ内の変更の有無についてのみ着目している．より詳細な ML パイプライン進化は，最適な ML パイプラインの選択において重要であると考えられるため，ML パイプライン内に記述されているコマンドやパラメータなどの詳細な変更内容について調査する．

謝辞

本研究の一部は，JSPS 科研費 JP18H04097, JP21H04877，および，JSPS・スイスとの国際共同研究事業（JPJSJRP20191502）の助成を受けた．

参考文献

[1] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann. Software engineering for machine learning: A case study. In *Proceeding of the IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 291–300, 2019.

[2] iterative.ai. "Open-source Version Control System for Machine Learning Projects". http://dvc.org/（閲覧日：2022 年 2 月 8 日）.

[3] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, Vol. 28, , 2015.

[4] I. Karamitsos, S. Albarhami, and C. Apostolopoulos. Applying devops practices of continuous automation for machine learning. *Information*, p. 363, 2020.

[5] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the genetic and evolutionary computation conference 2016*, pp. 485–492, 2016.

[6] I. Drori, Y. Krishnamurthy, R. Rampin, R. de Paula Lourenco, J. P. Ono, K. Cho, C. Silva, and J. Freire. Alphad3m: Machine learning pipeline synthesis. *arXiv preprint arXiv:2111.02508*, 2021.

[7] M. Baker. 1,500 scientists lift the lid on reproducibility. *Nature News*, p. 452, 2016.

[8] J. Pineau, P. Vincent-Lamarre K. Sinha, V. Larivière, A. Beygelzimer, F.d' Alché-Buc, E. Fox, H. Larochelle. Improving reproducibility in machine learning research: a report from the neurips 2019 reproducibility program. *Journal of Machine Learning Research*, 2021.

[9] LF Projects, LLC. "An open source platform for the machine learning lifecycle". https://mlflow.org/（閲覧日：2022 年 2 月 8 日）.

[10] Pachyderm Inc. "The Leader in Data Versioning and Pipelines for MLOps". https://www.pachyderm.com/（閲覧日：2022 年 2 月 8 日）.

[11] A. Barrak, E. E. Eghan, and B. Adams. On the co-evolution of ml pipelines and source code-empirical study of dvc projects. In *Proceeding of the 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 422–433, 2021.

[12] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, F. Xie, and C. Zumar. Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, pp. 39–45, 2018.

ⓒ 2022 Information Processing Society of Japan