データ分析プログラムにおける ライブラリ関数利用方法の探索的分析

池上 綾乃[1,a)]　Ani Hovhannisyan[1,b)]　石尾 隆[1,c)]　竹之内 啓太[2,d)]　松本 健一[1,e)]

概要：ビジネスにおけるデータ分析のためのプログラミング言語として, 初心者が学習しやすく, データ分析に適したライブラリが豊富である Python が注目されている．データ分析における Python プログラムは, アプリケーション開発で作られるプログラムよりも比較的短く, 単純な構造で実装されていることが報告されており, その特徴を利用したプログラム作成の支援が可能であると考えられる．一方で, データ分析において実際にどのようなライブラリを用いて, どのような典型的な処理が実装されているのかは明らかとなっていない．そのため, たとえばデータ加工に用いられるライブラリである Pandas を用いた関数呼び出しの列を自動合成する手法が既存研究で提案されているが, それが実プログラムにおいてどの程度有用であるかは不明である．本研究では, Python プログラムの実装支援技術を開発することを目的として, Kaggle で公開されているデータ分析プログラムがどのようなライブラリ関数を使用しているのかを調査する．その結果, 利用頻度の高い 50 個の関数で約 240,000 件のプログラムのうち, 52,007 件 (約 20%) のプログラムを網羅できる一方で, 全体では 90,665 個の関数が使用されており, 関数呼び出しの検索や推薦が重要となることを確認した．

キーワード：実証的研究, データ分析プログラム, 探索的分析

1. はじめに

2010 年代頃から起こった機械学習ブームから，データサイエンスの重要性が高まっている．データ分析を行うためのプログラムの作成には，データサイエンスに関するライブラリが豊富であり，初心者でも学習しやすい言語である Python がよく用いられている [1]．Python でデータ分析プログラムを作成する際に用いられるツールの 1 つが Jupyter Notebook[2] である． これは，Notebook というプログラムや文章，画像などをまとめて保存できるファイルを操作できるツールで，パラメータチューニングやグラフの出力など，探索的に計算結果を確認する必要があるデータ分析に適している．

Jupyter Notebook が広く利用されていることから，その利用状況を調査する Notebook を用いて実装されたプログラムに関する分析 [3] が行われており，データ分析における Python プログラムはアプリケーション開発で作られるプログラムよりも比較的短く, 単純な構造で実装されていることが報告されている．しかし，データ分析の典型的な手順，たとえばどのライブラリのどの関数がよく使用されているのかは明らかになっていない．

本研究では，データ分析における Python プログラムでどのようなライブラリ関数が使用されているのかを明らかにする．これにより，データ分析に特化した Python プログラムの実装支援等に向けた有用な知見を得ることを目指す．本研究では，データ分析コンペティションプラットフォームである Kaggle[*1]からデータ分析プログラムを取得し, **(RQ1)** データ分析プログラムは利用頻度の高いライブラリだけで実現できるのか， **(RQ2)** 頻出するライブラリの利用パターンはあるのかについて，それぞれライブラリ単位，ライブラリに定義された関数単位での調査を行った．

2 節では本研究に関連する研究について述べる．3 節では使用したデータセットと前処理について説明し, 4 節ではその予備調査について報告する．5 節で調査課題の動機, 方法，結果について報告し，6 節で考察を行う．7 節で本研究の妥当性の脅威について述べ，最後に 8 節でまとめと

[1] 奈良先端科学技術大学院大学
Nara Institute Science and Technology, Ikoma, Nara 630-0101, Japan
[2] 株式会社 NTT データ
[a)] ikegami.ayano.hs9@is.naist.jp
[b)] hovhannisyan.ani.hb7@is.naist.jp
[c)] ishio@is.naist.jp
[d)] Keita.Takenouchi@nttdata.com
[e)] matsumoto@is.naist.jp

[*1] Kaggle `https://www.kaggle.com/` 2021/6/10 閲覧.

図 1 データの前処理手順

今後の展望について述べる．

2. 関連研究

本研究に関連する研究として，Jupyter Notebook に関する研究，そしてデータ分析プログラムの実装支援に関する研究が行われている．

Jupyter Notebook に関する研究．Pimentel ら [3] は，Jupyter Notebook に関する実証的研究が十分に行われていないという背景から，264,023 個の Github リポジトリに含まれる 1,159,166 個の Jupyter Notebook を分析した．この研究では Notebook に書かれたプログラムを実行しなおしても，Notebook に記載された実行結果が得られないことが多いという再現性の低さを指摘したほか，よく出現する Python の構文の分布から，Notebook で実装されるプログラムは従来の Python プログラムよりも比較的簡単な文法かつ短く記述されていると報告した．この研究では numpy や matplotlib，pandas，scikit-learn など，データ分析に特化したライブラリが高頻度で利用されていることも報告しているが，どのライブラリ関数がよく使用されているのかは分析されていない．

データ分析プログラムの実装支援に関する研究．Bavish ら [4] は，与えられた DataFrame 型の入出力例から，実現するプログラムを Pandas というライブラリを用いて合成する AutoPandas という手法を提案した．この研究では，Pandas が提供する全関数を使用し，そのうち最大 3 個を組み合わせたプログラムを合成することがで，Q & A サイト Stack Overflow 上の解答に挙がっているプログラムのいくつかを合成することができたと報告されている．一方で，その合成能力が，現実のプログラム作成にもたらす効果は明らかになっていない．

3. 分析対象データセット

本研究では，データ分析コンペティションプラットフォームの一種である Kaggle で公開された Notebook を収集したデータセットである KGTorrent[5] を利用する．GitHub 上でも多数の Jupyter Notebook ファイルが公開されているが，Notebook は推奨されるコーディング手法に従っていなかったり，使われていない変数や廃止された関数が含まれていたりするなど，プログラムの質が低いことが指摘されている [6]．Kaggle に公開されているソースコードは，GitHub で公開されているソースコードよりもデータ分析というドメインに特化している．また，コンペティションに利用された Notebook は，プログラムの質が比較的高い可能性があるほか，ソースコードにタグが付与されているため，メタデータを用いてプログラムの特性を比較することができるという利点がある．

KGTorrent は 2020 年 10 月 27 日時点で公開されている 248,761 個の Python で記述された Notebook ファイルと，コンペティションや入力データセット，それらに付与されたタグなどのメタデータを記録したデータベースから構成されている．タグは階層構造で記録されており，データセットやプログラムに対して付与することができる．特に，本研究では，分析手法に関するタグである “technique” タグに着目する．タグの一部には親子関係が定義されており，“technique” という名前のタグの子要素として，可視化，データクリーニング，特徴量選択など，プログラムが行っている処理や使用している技術の名前を表現するタグが定義されている．

前処理の手順を図 1 に示す．前処理は，Notebook ファイルからソースコードを抽出する処理，ソースコードからライブラリ名およびライブラリ関数名を抽出する処理，そして KGTorrent のメタデータと Notebook ファイルの特定する処理の 3 段階に分けて行った．

まず，Notebook ファイルからソースコードのみを抽出する処理を行った．Notebook ファイルには Python プログラムだけではなく，Markdown で記述されたテキストや，プログラムの出力結果なども保持されている．本研究ではソースコードに着目するため，Notebook ファイルからソースコードのみを抽出した．Notebook ファイルは JSON 形式で表現されており，Notebook ファイル内の各セルに “cell_type” という属性が存在する．この属性は，“code” や “markdown” といったセルの型を保持している．そこで，“cell_type” が “code” であるセルに含まれている “source” 属性からソースコードを順に取得し，それらを文字列結合して python プログラムを取得した．

次に，得られた Python プログラムから，構文解析によってライブラリ名およびライブラリ関数名を抽出した．パーサジェネレータ ANTLR (ANother Tool for Language Recognition)[7] で，Python3 の文法ファイル[*2]を用いて構文解析を行い，得られた構文木から，以下のルールのいず

[*2] https://github.com/antlr/grammars-v4/tree/master/python/python3-py 2021 年 7 月 29 日閲覧.

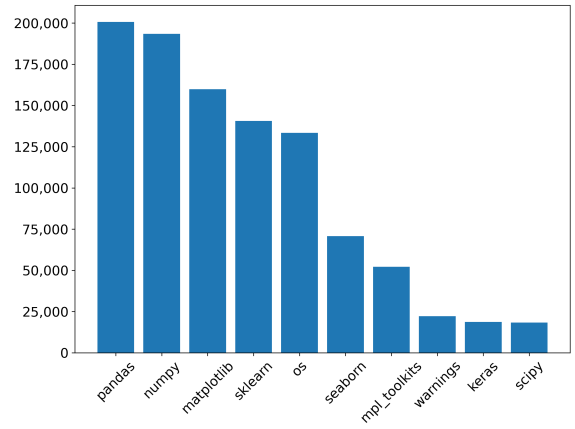

図 2 最も使用されている上位 10 件のライブラリ

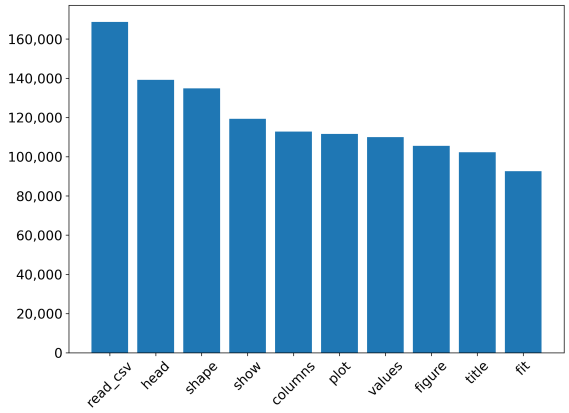

図 3 最も多くの Notebook に出現する上位 10 件のライブラリ関数

れかに該当する識別子をプログラムが利用するライブラリ名を抽出した．

- *dotted_as_names* に含まれる *dotted_name*
- 非終端記号 *import_as_name* に含まれる先頭の *Name*
- *import_from* に含まれる *dotted_name*

また，ライブラリ関数の呼び出しは，式を表す非終端記号 *atom_expr* に要素 *trailer* が含まれ，かつ *trailer* に *Name* がある場合のみ，*trailer* 内の *Name* をライブラリ関数名として抽出した．

コメントや空白行を除いたプログラムの行数 (以下, LOC) を計測するために *Stmt* を検出した回数を記録し，LOC やライブラリ関数の数が 0 の Notebook ファイルは分析の対象外として取り除いた．これらの工程を終え，241,655 件の Notebook ファイルのソースコードを用意した．

最後に，メタデータと Notebook ファイルの紐付けを行った．ドキュメントによると，Notebook ファイルの命名は KGTorrent の Users テーブルに含まれる UserName 属性と，Kernels テーブルに含まれる CurrentUrlSlug 属性を使用して，“UserName_CurrentUrlSlug” という形式で与えられている．この情報からメタデータを結合し，Notebook のファイル名を紐付けた．さらに，Tags テーブルと Kernels テーブルを結合し，Notebook ファイルとタグデータの紐付けをを行った．そして，分析方法を表すタグである “technique” に属するタグと紐づいたソースコードを取得した．

4. 予備調査

予備調査として，KGTorrent で公開されたプログラムのライブラリ，ライブラリ関数，LOC の分布，“technique” に属するタグについて調査した．

ライブラリの頻度分析：よく利用されるライブラリの分析は Pimentel ら [3] でも行われているが，Kaggle に公開されている Notebook でも同様の傾向が見られるのかどうか

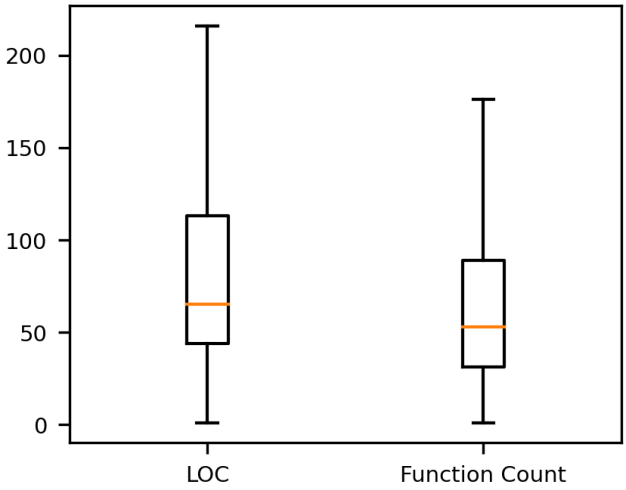

図 4 LOC およびライブラリ関数の個数（外れ値除く）

を確かめるためにこの調査を行った．集計したところ，全 Notebook ファイルで計 3,495 個のライブラリが使用されていることが分かった．図 2 によく使われているライブラリ上位 10 件を示す．なお，この予備調査ではライブラリのルートパッケージのみを出力している．先行研究では，numpy，matplotlib，pandas の順に使用される頻度が高いライブラリとして挙げられていたが，Kaggle の Notebook においても，ほぼ同様の傾向が見られた．一方で，先行研究の頻度分析では見られなかった keras, tensorflow などのデータサイエンスに特化したフレームワークが上位に入っている点が相違点として挙げられる．

ライブラリ関数の頻度分析：次に，ライブラリ関数の頻度分析について述べる．集計の結果，全 Notebook ファイルで計 90,655 個のライブラリ関数が使用されていることが分かった．図 3 は使用している Notebook ファイルの件数が最も多いライブラリ関数上位 10 件を表すグラフである．これらのライブラリ関数は，pandas や matplotlib など図 2 で示されたライブラリが提供しており，図中のライ

ブラリ関数はすべて主要なライブラリから利用されていることがわかった．また，ライブラリ関数は計 90,655 種類検出したが，10,000 件以上の Notebook ファイルで使用されているライブラリ関数はわずか 127 個であり，少数のライブラリ関数が多数の Notebook ファイルで使用されていることが明らかになった．

LOC の分布：Pimentel ら [3] によると，Notebook のプログラムは一般的な Python プログラムよりも単純な構造で LOC も少ないという報告がされている．しかし，Kaggle から取得した Notebook においても，同じ傾向が見られるのかは明らかになっていない．そこで，本稿では予備調査として，プログラムの規模がどの程度であるのかを調査した．図 4 は LOC およびライブラリ関数の個数のを示した箱ひげ図である．LOC は平均 96.3，標準偏差は 105.3，中央値が 65 であるが，最大値が 8,470 と外れ値が非常に大きいため，図 4 には外れ値を描画していない．外れ値はあるものの，平均 102.5，中央値 62 と報告されている Github 上の Notebook ファイルの LOC とほぼ変わらない [6] ため，Kaggle 上の Notebook の規模と GitHub 上の Notebook の規模はほぼ変わらないと考えられる．

ライブラリ関数の個数は平均 78.70，標準偏差は 109.3，中央値は 53 だが，LOC と同様に最大値が 16,911 と外れ値が非常に大きいことが分かった．ライブラリ関数の個数の中央値が 53 であることから， 最大 3 個の関数を合成できる AutoPandas は，現実のプログラムを合成するには課題があると考えられる．

“technique” タグの頻度分析: KGTorrent で “technique” タグが付与された Notebook ファイルの件数を集計した．表 1 は “technique” タグごとの KGTorrent で取得できる Notebook ファイルとメタデータの紐付けが成功した件数である．なお，“technique” タグを親にもつタグは全部で 49 個であったが，本分析では紐付けが成功した件数が多い 10 件のタグのみを分析対象とした．また，Kaggle では一つの Notebook ファイルに対して複数のタグを付与することが可能であるため，表 1 には一つの Notebook ファイルが重複して集計されている可能性がある．

表 1 “technique” タグが付与された Notebook ファイルの件数

	タグ名	ファイル件数
1	可視化	7,850
2	探索的分析	5,447
3	分類	3,619
4	データクリーニング	2,035
5	深層学習	2,124
6	特徴量加工	1,410
7	自然言語処理	1,180
8	CNN	853
9	ランダムフォレスト	813
10	線形回帰分析	797
	合計	26,758

5. ライブラリ関数利用方法の調査

本研究では，下記二つの調査課題を設定する．

- **RQ1:** データ分析プログラムは利用頻度の高いライブラリだけで実現できるのか？
- **RQ2:** 頻出するライブラリの利用パターンはあるのか？

5.1 RQ1:データ分析プログラムは利用頻度の高いライブラリだけで実現できるのか？

RQ1 では，利用頻度の高いライブラリおよび関数呼び出しを n 個使用し，網羅できるプログラムの件数の割合を調査する．

5.1.1 ライブラリ単位の分析

調査動機: 先行研究では，Maven を用いてパッケージ管理を行っているクライアントのプログラムを分析し，わずか 13 %のライブラリで 83 %のクライアントは動作するということを明らかにしている [8]．一方で，Python にもライブラリやライブラリ関数が数多く提供されているが，実プログラムでどれほどのライブラリ関数が実際に使用されているのかは関連研究で明らかになっていない．そこで，本研究も同様に Python で記述されたデータ分析プログラムを分析し，利用頻度の高い上位 n 個のライブラリで網羅できるのかを検証する．ライブラリとプログラムの網羅率の関係を調査することによって，データ分析プログラムの実装を補助するツールを開発する際に，どのライブラリをサポートすると実プログラムに有用であるかを知る上で有用な知見になりうる．

調査方法：3 節でメタデータを付与する前に取得した 241,655 件の Notebook ファイルを対象に，4 節で調査したライブラリの頻度集計結果を用いて上位 n 個のライブラリを使うと，何件のプログラムを網羅できるのかを調査した．なお，本分析では上位 n 個のライブラリがプログラム中に出現するすべてのライブラリが含まれるプログラムの件数の割合を網羅率と定義する．

調査結果：図 5 に出現頻度の高いライブラリ 500 個で網羅できるプログラムの割合を示す．x 軸が利用頻度の高いライブラリ件数，y 軸が網羅率を表す．4 節で述べた通り，検出したライブラリは計 3,495 個のライブラリが使用されているが，50 件のライブラリを使用する段階で，78.8%のプログラムを，500 件のライブラリを使用すると，97.5%のプログラムを網羅できることが分かった．

この結果から，3,495 個あるライブラリのごく少数しか分析に使用されていないことが分かったが，それぞれのライブラリには多数のライブラリ関数が提供されているため，同様にライブラリ関数単位で網羅率を計算し，より詳細な単位で分析を行った．

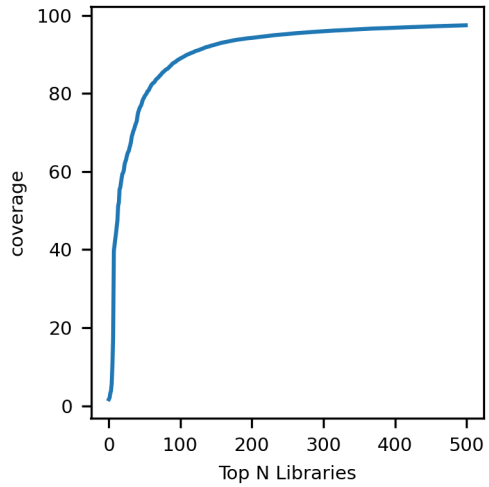

図 5 上位 n 個のライブラリとプログラムの網羅率

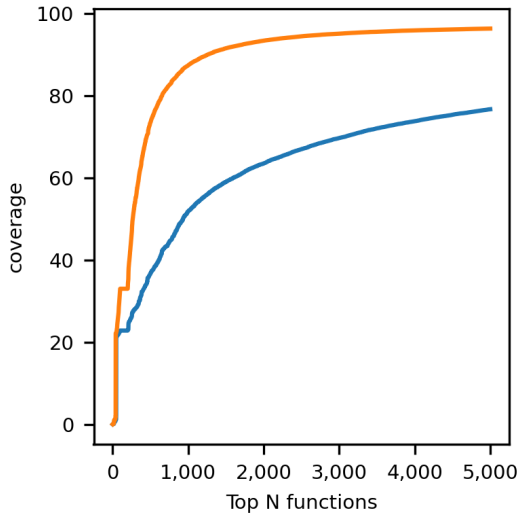

図 6 上位 n 個のライブラリ関数とプログラムの網羅率

5.1.2 ライブラリ関数単位の分析

調査動機：利用頻度の高いライブラリ関数は何か，実プログラムを記述するのに必要なライブラリ関数の数はいくつかをを知ることは，プログラム作成の支援手法を考える上で有用な知見になると考えられる．

調査方法：3 節で取得した 241,655 件の Notebook ファイルを対象に，4 節で調査したライブラリ関数の頻度集計結果を用いて上位 n 個の関数を使うと，何件のプログラムを網羅できるのかを調査した．なお，この分析では，網羅率として上位 n 個のライブラリ関数がプログラム中に出現するすべてのライブラリ関数が含まれるプログラムの件数の割合と，プログラム中に出現するライブラリ関数の 8 割が含まれるプログラムの件数の割合を提示する．

調査結果：図 6 にプログラムの網羅率を示す．x 軸が利用頻度の高いライブラリ関数の個数，y 軸が網羅率を表す．橙の線が 8 割網羅した際の網羅率，青線がすべてのライブラリ関数を網羅したときの網羅率を示している．上位のライブラリ関数が 50 個未満の場合は，すべてのライブラリ関数を網羅率が 20%未満だが，50 個を境に 20%のプログラムを完全に網羅できるようになっていることがわかる．また，5,000 個のライブラリ関数ではすべてのライブラリ関数を網羅できる割合は 70%程度までであるが，8 割まで条件を緩めると，1,000 個のライブラリ関数で 90%近くのプログラムを網羅できる．

この結果から，網羅率が急激に増加した箇所である上位 50 個のライブラリ関数で網羅できたプログラムの特徴は何かを調査した．

上位 50 個のライブラリ関数で網羅できたプログラム 52,007 件のファイルを調査したところ，52,007 件中 48,561 件が *kerneler*[*3] という特定のユーザによる Notebook ファイルだった．このユーザは Kaggle が提供している Bot の一種で，様々なデータセットの探索的分析を行うプログラムを自動で作成し，公開している．また，Notebook 中に “If you’re inspired to dig deeper, click the blue ”Fork Notebook” button at the top of this kernel to begin editing.” とあることから，Kaggle の初心者ユーザ向けに分析を行う取っ掛かりとして公開していると考えられる．

5.2 RQ2: 頻出するライブラリの利用パターンはあるのか？

RQ2 では，“technique” タグで表現される分析手法の違いで，よく使われるライブラリや関数呼び出しの組み合わせが異なるかどうかを調査する．

5.2.1 ライブラリ単位の分析

調査動機：4 節から，前処理によく使われる pandas ライブラリはどのようなプログラムでも使用されると考えられるが，グラフを描画する場合は matplotlib，機械学習モデルを構築する際には scikit-learn が利用されると予想できる．この仮説を検証するために，RQ2 ではタグとライブラリの組み合わせについて調査する．

調査方法：ライブラリと分析手法の関連を調査するため，ライブラリと “technique” タグを用いてアソシエーション分析を行い，どのような組み合わせがよく利用されるのかを明らかにする．アソシエーション分析とは，データの要素に関する相関を，ルールとして抽出し，データに潜在する項目館の関連を把握できる手法である．本分析では，PyPI で *mlxtend* ライブラリとして提供されている Apriori

[*3] `https://www.kaggle.com/kerneler`

表 2 ライブラリおよびライブラリ関数に関するルールおよびパターン

タグ	ライブラリに関するアソシエーションルール			支持度	信頼度	リフト値
分類	*sklearn.neighbors*	⇒	*pandas, KNeighborsClassifier*	0.06	0.98	10.86
可視化	*numpy, plotly.graph_objs*	⇒	*plotly.offline*	0.07	0.88	8.70
探索的分析	*sklearn.model_selection, matplotlib.pyplot*	⇒	可視化 ,*seaborn, numpy*	0.06	0.54	3.80
特徴量加工		⇒	*pandas, sklearn.model_selection, numpy*	0.06	0.63	1.56
データクリーニング	*matplotlib.pyplot, pandas*	⇒	可視化, *searborn, numpy*	0.07	0.63	1.57
深層学習	*matplotlib.pyplot, pandas*	⇒	*numpy, os*	0.07	0.73	1.40

タグ	ライブラリ関数に関するシーケンシャルパターン	出現頻度
Notebook ファイル全体	(title, show, title, show)	70,375
Notebook ファイル全体	(figure, title, show, figure, title, show)	59,818
分類	(read_csv, head, fit, predict, fit, predict)	2,299
可視化	(read_csv, show, show, show, show, show, show, show)	1,878
自然言語処理	(read_csv, head, drop, fit)	6,327
線形回帰分析	(read_csv, head, figure, show)	7,852
CNN	(read_csv, head, fit, predict)	8,865
ランダムフォレスト	(read_csv, head, fit, fit, predict)	6,062

アルゴリズム [9] を使用して，リフト値が 1 より大きく，かつ条件分にタグ名が含まれるルールを抽出する．なお，本分析ではライブラリのルートパッケージだけでなく，サブモジュール単位でアソシエーションを抽出した．

調査結果：表 2 にタグが含まれるルールを一部提示する．可視化タグが条件部のルールには，plotly[*4]が条件部，結論部に含まれるルールが見られた．plotly は matplotlib と同様にグラフを描画する機能を提供するライブラリだが，Web ブラウザで操作可能なグラフを作成できるという特徴がある．4 節でライブラリの出現頻度を調査した際に，matplotlib の出現頻度は 159,855 件だったが，plotly の出現頻度は 11,118 件だった．このアソシエーションルールから，可視化を主目的とした分析ではそれ以外の分析目的と比べて plotly が使用されると考えられる．

5.2.2 ライブラリ関数単位の分析

続いて，ライブラリ関数の組み合わせを調査する．

調査動機：ライブラリ関数の組み合わせを調査することで，プログラム作成の支援をするときに，作成途中のプログラムをヒントに，どのライブラリが使われそうかを推定することができる．また，そのようなパターンが存在する場合，プログラムのテンプレートを提供することで，作成支援につながる可能性がある．この調査により，データ分析プログラムにおいて，よく使われるライブラリ関数の組み合わせがあるのかどうかを明らかにする．

調査方法：本研究では，プログラムの上から順に抽出したライブラリ関数を時系列的なデータとして捉え，シーケンシャルパターンマイニングのアルゴリズムの一種である BIDE (BI-Directional Extension)[10] を適用し，頻出するライブラリ関数の組み合わせを抽出する．シーケンシャルパターンマイニングには，間に不必要な要素が入っていても抽出されるパターンに影響を受けないという特性があると言われており，複数のプログラムで共通して見られるライブラリ関数のパターンを抽出する．ただし，より有用なパターンを抽出するために各タグごとにライブラリ関数を含むファイル件数を算出し，最も利用頻度の高い上位 100 件のライブラリ関数のみを対象とした．また，最小文字列数を 4 に設定し，各タグごとに出現頻度が大きいパターン上位 100 件を抽出した．

調査結果：表 2 の下部に，一部のライブラリ関数に関するシーケンシャルパターンを示す．ライブラリ関数列に対して，シーケンシャルパターンマイニングを行ったが，すべてのタグにおいて，グラフの描画に関するライブラリ関数のパターンが見られた．さらに，グラフを表示するライブラリ関数である show が 6 回にもわたってパターンとして出現する場合があり，Kaggle におけるデータ分析プログラムは一つの Notebook ファイル内でグラフを複数枚描画する処理が多く行われていると考えられる．

6. 考察

RQ1 で得られた結果から，データ分析プログラムを作成する際には，あまり利用頻度の高くない様々なライブラリ関数を調べなければならない可能性があり，学習コストが高いと考えられる．既存研究である AutoPandas [4] のアプローチは，Pandas の多数の関数から状況に合ったものを自動で探してくれるという点で学習コストを低減させる効果があると考えられるが，1 つのライブラリしか扱えないことからデータ分析の支援としての効果は限定的であることが分かった．今後，データ分析プログラムの作成を支援するためには，適切なライブラリ関数を検索，推薦する技術が重要である．

また，RQ2 から Kaggle におけるデータ分析プログラムでは，グラフを描画する頻度がかなり大きいことが伺えた．シーケンシャルパターンマイニングは，間に要素が入って

[*4] https://plotly.com/python/ 2021 年 7 月 31 日閲覧.

いても抽出されるパターンに影響がないという利点があるが，本研究の場合はグラフ描画に関する処理が多用されていたため，前処理のようなグラフ描画以外の処理に関するパターンを抽出することができなかった．この原因として，Notebook ファイルでは関数などの処理の区切りがなく，様々な処理が順番に並べられていることが挙げられる．グラフ描画以外のデータ処理に関するパターンを抽出するには，データフロー解析を用いて変数の状態を考慮して追跡するなどの何らかの方法で，処理の区切りを自動で認識するなどの解析が必要であると考えられる．

7. 妥当性の脅威

7.1 内的妥当性

本研究では，ANTLR を用いてライブラリ関数とライブラリを収集した．しかし，Python は動的言語であるため，静的解析だけではライブラリ関数のみを適切に取得できていない可能性がある．例えば，Notebook 内で独自のクラスを定義し，その中でメソッドを定義していた場合，ライブラリ関数と独自クラスのメソッドの区別がついていない可能性がある．ただし，Pimentel ら [3] によれば多くの Notebook は単純であり，該当するケースは少ないと考えられる．また，本研究ではライブラリ関数の出現頻度ではなく，そのライブラリ関数が出現した Notebook ファイルの件数を集計しているため，このようなケースで検出されたライブラリ関数が出現頻度に与える影響は小さい．同様に，ライブラリの場合も import 文から名前を取得したため，独自のライブラリであるかを区別する処理は行っていない．しかし，独自のライブラリの場合は出現頻度が低くなると考えられるため，本分析での影響は小さい．

7.2 外的妥当性

本研究では Kaggle 上で公開されている Notebook ファイルを分析しているため，Kaggle 上の Python で記述されたプログラムには本研究の分析結果を一般化することができると考えられるが，Kaggle 外のデータ分析プログラムには本研究の調査結果を汎化することはできない．本研究の調査を一般化するためには，GitHub 上の Notebook に対しても同様の分析を行う必要がある．

8. おわりに

本研究では，Python におけるデータ分析プログラムの利用方法の分析として，利用頻度の高いライブラリおよびライブラリ関数のみを用いてプログラムを作成できるのか，頻出するライブラリの利用パターンはあるのかを調査した．その結果，50 個程度のライブラリ関数で網羅できるプログラムもある一方で，大半のプログラムは利用頻度の低いライブラリ関数を使用しなければならないということが明らかになった．今後の展望として，プログラムの制御構造，類似したデータ処理の繰り返しの存在を考慮した上での分析や，入出力データセットを加味した分析などが挙げられる．

謝辞 本研究は JSPS 科研費 Grant Number JP20H05706 の支援を受けたものです．

参考文献

[1] Loukides, M.: Where Programming, Ops, AI, and the Cloud are Headed in 2021, `https://www.oreilly.com/radar/where-programming-ops-ai-and-the-cloud-are-headed-in-2021/` (2021). (Accessed on 2021/6/10).

[2] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S. et al.: Jupyter Notebooks - a publishing format for reproducible computational workflows., Vol. 2016 (2016).

[3] Felipe, P. J., Murta, L., Braganholo, V. and Freire, J.: A large-scale study about quality and reproducibility of jupyter notebooks, IEEE International Working Conference on Mining Software Repositories, Vol. 2019-May, IEEE, pp. 507–517 (2019).

[4] Bavishi, R., Lemieux, C., Fox, R., Sen, K. and Stoica, I.: AutoPandas: neural-backed generators for program synthesis, Proceedings of the ACM on Programming Languages, Vol. 3, No. OOPSLA, pp. 1–27 (2019).

[5] Quaranta, L., Calefato, F. and Lanubile, F.: KGTorrent: A Dataset of Python Jupyter Notebooks from Kaggle, (online), available from ⟨http://arxiv.org/abs/2103.10558⟩ (2021).

[6] Wang, J., Li, L. and Zeller, A.: Better Code, Better Sharing: On the Need of Analyzing Jupyter Notebooks, Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results, ICSE-NIER '20, New York, NY, USA, Association for Computing Machinery, p. 53–56 (online), DOI: 10.1145/3377816.3381724 (2020).

[7] Parr, T.: The Definitive ANTLR Reference, The Pragmatic Bookshelf (2007).

[8] Harrand, N., Benelallam, A., Soto-Valero, C., Barais, O. and Baudry, B.: Analyzing 2.3 Million Maven Dependencies to Reveal an Essential Core in APIs, pp. 1–15 (2019).

[9] Agrawal, R. and Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases, Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., p. 487–499 (1994).

[10] Wang, J. and Han, J.: BIDE: Efficient Mining of Frequent Closed Sequences, Proceedings of the 20th International Conference on Data Engineering, ICDE '04, USA, IEEE Computer Society, p. 79 (2004).