修士論文

オープンソースソフトウェアプロジェクトの競合関係の分析

武井 優己

主指導教員 青木 利晃

北陸先端科学技術大学院大学
先端科学技術研究科
（情報科学）

令和７年３月

Abstract

ソフトウェアはこれまでの歴史の中で, 競争による台頭と淘汰を繰り返すことで進化してきた. Open Source Software(OSS) も例外に漏れず, そのサイクルは加速傾向となっている. 特に, Web 開発や深層学習などの進化の早い領域では顕著である. ただし, OSS プロジェクト同士の競合関係がその生存に与える影響は解明されておらず, 現在採用している OSS が競合との開発競争に敗れるリスクが現在も至る所に存在している. この, 強力な競合の出現により将来開発が停止することを, 本研究では REV(Rising Event) と呼称する. 本研究では, OSS プロジェクト間の競合関係を定量化する手法”Mutual Impact Analysis of OSS(MIAO)”と, 将来の REV 発生を予測する ”REV 予測モデル（REV Predictive Model, RPM）”の 2 つの手法を提案し, これらの有効性を検証した. MIAO は, マクロ経済学でよく用いられる, 構造的ベクトル自己回帰モデル (SVAR) とインパルス応答関数 (IRF) を用いて, 複数 OSS が相互に与え合う累積的な影響を算出する. これにより, 従来定性的にしか評価できなかった OSS の競合関係を定量化し, 特定の OSS が競合相手からどれほど負の影響を受けているかなどを把握できる. さらに，RPM は MIAO を応用し, 将来の REV 発生を機械的に分類する. これらの手法の有効性を検証するための実験において, 66 のグループに対する実験で, MIAO は 85%の精度で REV を特定することに成功した. RPM の予測精度は 73%と中程度の精度になったが, 将来の REV 発生リスクをスクリーニング的に評価するものとしては有用であることを示した. このことから, MIAO および RPM は,OSS エコシステムのダイナミクスを理解し, OSS プロジェクトの盛衰を予測する上で有用なツールとなる可能性がある.

目 次

図 目 次

表 目 次

第1章　はじめに

オープンソースソフトウェア (OSS) はデジタル上のインフラとして機能しており, 現代の情報システムの多くは, オペレーティングシステム, 言語, アプリケーションといった種々のレイヤーにおいて, 数多くの OSS コンポーネントで構成されている. Open UK のレポート [1] によると, 調査対象となった民間企業および公共部門組織の 97%が社内で OSS を使用していると報告している. 同様に, Synopsys の調査 [2] では, 17 の産業にわたる 1,067 の商用コードベースの 96%が OSS を採用しており, 全コードの 77%を占めていることが判明した.

現代の情報システムの多くが OSS へ依存していることを考えると, 生存能力の高い OSS を選ぶことがシステムを長期的に保守, 成長させるための効果的な戦略であるといえるが, OSS プロジェクトの持続可能性には疑問符があり, 多くの OSS は人気を博したものでさえも長続きしない [3]. OSS の失敗要因は多種多様であるが, Coelho ら [4] によれば, ”陳腐化”や”強力な競合の出現”が失敗の最も大きな要因であると述べられている. これまでのソフトウェアの歴史を鑑みれば, 新たに出現する OSS は既存のものの問題点を改善することで台頭し, 一方で新たな問題を生み出すというサイクルの中で進化してきており, この研究結果には正当性がある. つまり OSS には栄枯盛衰があり, 例え健全にコミュニティやプロジェクトが運営されていても, あるいはコード品質が高くても, 時代に抗えなくなる日が訪れるのである.

これまでの OSS の持続可能性に関する多くの先行研究は, 主にプロジェクトの内部要因に焦点を当ててきた. 例えば, Raja ら [5] は活力, 回復力, 組織という 3 つの次元に基づいて OSS プロジェクトの持続可能性を指数化した. Samoladas ら [6] は FLOSSMetrics database に対する生存分析により, OSS プロジェクトの生存期間に寄与する要因について調査し, プロジェクトに貢献者が 1 人増えるごとに, プロジェクトの生存率が 15.8%加するという分析結果を得た. Liao ら [7] はプログラミング言語, ファイル数, コア開発者の質の影響に基づいてプロジェクトの寿命を予測した. しかし, プロジェクトの持続可能性は内部要因のみならず, 前述の「陳腐化」や「強力な競合の出現」といった外部要因にも影響される. ただし, 陳腐化や強力な競合の出現は不確定なイベントであり, これらの要因が既存の OSS プロジェクトに与える影響を分析する研究は不足している. その理由として, OSS の競合関係を定量化することは難しいタスクであり, 対照実験や反実仮想といった複雑な因果推論が必要となる点が挙げられる.

一方で, 経済学では厳密な因果推論を想定せずとも, 変数間の影響を定量化するための仕組みが, VAR モデルやインパルス応答関数により確立されている. インパルス応答関数は, ある経済変数の変化がその他の経済変数に時間経過でどのような影響を与えるかを分析するための手法である. 具体的な事例としては, 原油価格の変

動が, 世界の経済活動や米国のマクロ経済指標に与える影響を分析するもの [8] や, 中央銀行の金融緩和に対する Consumer Price Index(CPI, 消費者物価指数) や為替レートといった各種経済変数の反応が, バブル経済時とバブル崩壊後とでどのように変化するかを分析するもの [9] などがある. この手法をソフトウェア工学に応用すると, ある OSS の活動量の変化が, その他の OSS の活動量に時間経過に応じてどのような影響を与えるかを考えられるようになる. つまり, インパルス応答から得られる情報を用いれば, 強力な競合の出現により負の影響を受けた OSS を特定することが可能となるかもしれない. 以降では, 強力な競合の出現により将来開発が停止することを REV(Rising Event) と呼称する.

これらの前提のもとで, 本研究では, OSS の競合関係を定量化する手法と, その定量化手法を用いて, REV により開発が停止する可能性のある OSS を予測するための手法の 2 つの手法を提案する. まず, OSS の競合関係を定量化する手法では, (Q1)IRF を用いて OSS の時間的競争関係を定量化する Mutual Impact Analysis of OSS(MIAO) を提案する. (Q2)REV グループと nonREV グループで構成されるデータセットに MIAO を適用する実験を実施する. (Q3) 実験結果に基づいて MIAO の有効性を検証する. REV により開発が停止する可能性のある OSS を予測するための手法では, (P1) 回帰モデル, 分類モデルといった複数のモデルを混合した REV 予測モデルを提案する. (P2) REV 予測モデルの評価用データセットを構築し, REV 予測モデルの性能評価を行う.

これらの提案により, 本研究は OSS の持続可能性の分野に貢献する. REV の概念は競争による開発停止の条件を定義し, MIAO は外部の競争ダイナミクスを考慮した OSS の持続可能性の定量的分析を可能にする. そして, REV 予測モデルにより将来の REV を特定することが可能にする. これらの方法論は, 従来定性的に評価されてきた OSS プロジェクト間の競争関係を定量的に評価し, その定量データを用いることで将来のリスクを予期するための新しいツールを提供する. 本研究の手法と, 内部要因や他の外部影響に関する先行研究の知見を組み合わせることで, OSS の持続可能性をより多角的に評価することが可能となる.

本論文は 7 つの章で構成される. 2 章では, OSS の生存性に関する関連研究および先行研究を 4 つのカテゴリ（コミュニティ, プロジェクト管理, 技術的要因, 外部要因）に分類し, それぞれのカテゴリに関連する代表的な研究のレビューを行う. そのレビューにより, これまでに得られた知見とまだ分かっていないことのギャップを明確にし, 本研究の立ち位置および新規性についてまとめる. 3 章では, MIAO と REV 予測モデルの理論の基礎となる, 時系列の特性や, ベクトル自己回帰 (VAR) モデル, 構造的ベクトル自己回帰 (SVAR) モデル, IRF などに関する理論的基盤を提供する. 4 章では, MIAO の理論的背景と詳細なアルゴリズムを詳解した後, 実験を通した MIAO の性能評価および解釈を行う. 5 章は, REV 予測モデルに関する理論的背景と, 実験を通した性能評価を行う. 6 章では, これまでの実験結果およびモデルの評価から得られる, 提案手法に関する考察を展開し, 先行研究との比較を交えて本研究手法の利点と限界点についてまとめる. 7 章では, これまでを総括し, 今後の課題についてまとめ, 本論文を締めくくる.

第2章　関連研究

OSS の急速な普及と重要性の高まりに伴い, その持続可能性に関する多くの研究がなされてきた. 本章では, OSS プロジェクトの生存性に影響を与える要因, 生存性を評価するための手法, そして OSS の持続可能な発展を促進するための戦略に関する既存の研究を概観する. これらの関連研究を概観することで, 本研究の位置づけや新規性を明確にし,OSS の生存性に関する理解を深める.

まず, OSS の生存性に関する研究は個々の目的やアプローチは異なるものの, それぞれの前提条件においてどのような要因が OSS の生存性に寄与するかを分析するものであるといえる. OSS プロジェクトは一般的に, 貢献者が自発的, 創発的に参加することにより運営されるが, 貢献者のほとんどは有志での参加である. したがって, プロジェクトを運用するコミュニティ内の相互作用や, 貢献者の動態といったコミュニティ内の構造が, OSS プロジェクトを長く続けるために重要な要素になる. そして, そのコミュニティが次第に拡大していくと, メンバーを計画的にコントロールするために, 意思決定プロセスやリーダーシップ, 組織のガバナンスなどが重要となってくるのは, 現実の組織と同様である. また, ソフトウェアは多種多様の技術ドメインがあり, それぞれの技術ドメインには固有のプロジェクト特性や技術的要因も存在する. これらの技術ドメインはそれぞれが異なるライフスパンや, 競争環境に晒されており, その特性に応じて生存性に差異があると考えられる. 直感的には, オペレーティングシステムやプログラミング言語は, 流行り廃りの激しい Web 開発ライブラリに比べ長寿命であり, ライフスパン的にも成熟していると考えられる. その他に, 商用利用に制限のあるライセンスでは多くの開発者に支持されることは難しいと考えられる. また, 健全に運用されていた OSS であっても, 全体のエコシステムにおける立ち位置や, 外部要因によりその生存性に大きく影響を受けることがあるだろう. このような観点で, OSS プロジェクトの生存性に影響を与える要因を大きく分類すると, 以下の 4 つが挙げられる.

- コミュニティ構造と貢献者の動態
- プロジェクト管理とガバナンス
- プロジェクト特性と技術的要因
- エコシステムと外部要因

以降では, これら 4 つのカテゴリに関する先行研究をまとめ, 最終的に本研究の立ち位置や新規性を明確にする.

2.1 コミュニティ構造と貢献者の動態

Raja ら [5] は, OSS プロジェクトの生存能力の指標を, 活力, 回復力, 組織の 3 つの次元から定義される VI(Viablity Index) として定式化した. 活力は, 単位時間あたりのバージョン数, 回復力はコミュニティの不具合等に対する応答速度, 組織はコミュニティのコアグループがタスクを監視する能力をそれぞれ定量化するものである. この研究では, VI が OSS の持続的な成長を高精度に説明可能な指標であることを示した. つまり, 生存能力の高いプロジェクトは, コミュニティ内にタスク監視を行うコアグループが存在し, 不具合や要望に対する応答速度が優れている. そのため, 単位時間あたりのリリース回数が多いという特徴があると考えられる.

Samoladas ら [6] は FLOSSMetrics database に対する生存分析により, OSS プロジェクトの生存期間に寄与する要因について調査した. 対象の変数のうち, 最も OSS の生存性に影響するのは貢献者数であり, プロジェクトに貢献者が 1 人増えるごとに, プロジェクトの生存率が 15.8%増加するという分析結果が得られている. これは, Linus の法則に準ずる結果である. 一方で, 貢献者毎にその生産性が異なることも事実である. Allaho らは [10], GitHub などの OSS コミュニティにおける貢献者のソーシャルネットワーク構造を統計的に分析し, 開発者の生産性への影響を調査した. 分析の結果, 開発者のソーシャルタイと生産性の間には正の相関関係が見られた. つまり, 多くのつながりを持つ開発者は, より多くのプロジェクトに参加し, より多くのコミットを行い, より長い期間にわたってプロジェクトに貢献する傾向があることが明らかになった. 開発者の外部ネットワークの質が, いずれの成長段階においても OSS プロジェクトの生存に正の影響を与えることを Wang[11] も言及している. したがって, 上位の開発者同士はソーシャルネットワーク上で繋がっており, 彼らが所属するコミュニティあるいはプロジェクトは成功する可能性が高いと考えられる.

ただし, Raja[5] ら, Valiev[12] ら, Liao ら [7] らが指摘するように, 成功する OSS プロジェクトにはコアグループの存在が必要不可欠であり, 彼らの質がプロジェクトの生存性に寄与する. したがって, 彼らの離脱がプロジェクトの生存能力を脅かす可能性がある. そのため, コアグループの開発者を引き継ぐ新規開発者のプロジェクトへの取り込みを継続的に行うことが重要となる. Ducheneaut[13] は, OSS プロジェクトで新規開発者がコミュニティに認められ正式な貢献者になるまでのプロセスを詳細に分析した. 分析の結果, OSS プロジェクトで活躍するための要素として, 開発技術よりもプロジェクト内で自身のアイデンティティを確立するほうが重要であることを発見した. したがって, プロジェクトを長期的に持続させたい場合, 新規開発者をスムーズに取り込み貢献させるための一連のプロセスに関する理解が重要であると述べられている.

2.2 プロジェクト管理とガバナンス

Noni ら [14] は OSS コミュニティのガバナンス構造と OSS コミュニティの活力の関係に関する詳細な分析を行った. Noni らによれば, OSS コミュニティのガバナンスはオープンソースベース, スポンサーベース, 相互主義ベース, 寛容な独裁者ベー

スの 4 つに定義できる．このうち, より中央集権的なリーダーシップと意思決定モデルを持つコミュニティ（スポンサー, 寛容な独裁者）は, 分散型モデル（オープンソース, 相互主義）を持つコミュニティよりも活力スコアが高い傾向にあった．ただし, メジャーリリースに限定した活力スコアを分析すると, 相互主義ベースと寛容な独裁者ベースのコミュニティは, その他 2 つのコミュニティよりも有意に高いスコアを示している．この結果は, 集中化されたリーダーシップは開発の迅速化につながる可能性がある一方で, メジャーリリースのような創造性を必要とする場面においては, 開発者への自由度が高い方が有利になる可能性を示唆している．

Li ら [15] は, Noni らとは異なる視点で OSS プロジェクトのリーダーシップの違いによる開発者の貢献意欲を評価するためのモデルを提案, 検証した．118 名の OSS 開発者を対象とした調査の結果, 変革型リーダーシップは開発者の内発的動機付けに正の影響を与え, 積極的な管理を行う交流型リーダーシップは外発的動機付けに正の影響を与えることが明らかになった．OSS プロジェクトの運営において, 内発的動機付けは, 開発者自身の楽しみや挑戦意欲, あるいは OSS の理念への共感に基づくタスクに対して, 特に効果を発揮する．一方, 外発的動機付けでは, 適切な管理により開発者の承認欲求を満たし, 貢献意欲を高めることに効果を発揮すると述べられている．この結果は, Noni らの研究と整合する部分もある．

その他の視点では, 財団や企業といった OSS コミュニティの上位組織がコミュニティに与える影響を調査するものがある．Izquierdo ら [16] は Apache, Linux, Mozilla などの有名な財団を含む 89 のソフトウェア財団を調査し, その役割を分析した．その結果, 財団は開発者コミュニティに法的および組織的な枠組みを提供する上で有用であるが, ソフトウェア開発の日常的な作業や意思決定プロセスへの関与や影響は限定的であることが明らかになった．Dahlander ら [17] は OSS 企業である MySQL, Cendio, Roxen, SOT の 4 社をケーススタディとし, これらの企業とコミュニティの関係性や経営課題を分析した．分析の結果, OSS 企業とコミュニティの関係性には, 共生, 片利共生, 寄生の 3 つのアプローチがあることが分かった．共生では, 企業は自社とコミュニティの双方にとって利益となるよう, 共同開発を試みる．MySQL はこのアプローチの好例であり, コミュニティと緊密に連携することで, 世界中で数百万件のインストール数を誇る主要なデータベースへと成長した．片利共生は企業がコミュニティに害を及ぼすことなく, コミュニティとの共存により利益を追求する．寄生では, 企業は自社の利益のみに焦点を当て, 自らの行動がコミュニティに害を及ぼす可能性を考慮しない．このアプローチに関する明らかなリスクは, 企業がコミュニティの基本的な規範, 価値観, 原則に違反する, あるいは単にフリーライダーとして認識されるなど, コミュニティから否定的な評価を受けることであると述べられている．

2.3 プロジェクト特性と技術的要因

Liao ら [7] は, GitHub にホストされる OSS プロジェクトから, 生存性に寄与する変数を発見し, それらを用いて OSS プロジェクトの寿命予測モデルを提案した．生

存性に寄与する変数にはプログラミング言語やプロジェクトに付与されたラベルなどが含まれる. 分析の結果, 生存性に寄与する変数のうち, プログラミング言語の選択が生存性に大きく寄与すること, また HTML や bootstrap といった特定のラベルが含まれる OSS プロジェクトの寿命は極端に低くなることを発見した.

Kuwata ら [18] は, OSS プロジェクトのライフサイクルステージに基づいたプロジェクト成長モデルを提案した. 成長モデルは, Born, Childhood, Adolescence, Adulthood, Obsoluted の過程を辿り, これらはコードサイズと活動量の時間的な推移を散布図にプロットすることでその図形的な特徴により推定される. この研究では, Apache HTTP Server や Hadoop, Docker などの基盤的な OSS や, その他の実験対象の多くはこの成長過程を辿ることが確認されている. 一方で, Childhood から Obsoluted にいきなり飛ぶなど, この成長モデルに該当しない OSS プロジェクトも存在し, その場合はプロジェクトの放棄などが考えられる. 成長モデルに該当しないものについて具体的な言及はないものの, 技術ドメインが関連している可能性があり, 先行研究で低寿命であるとされている技術ドメインではこのような傾向が見られるかもしれない.

多くの開発者はメンテナビリティの高いコードを好むが, OSS の生存性とコード品質に関連性があるかは興味深い問題である. Eluri ら [19] はコードメトリクスのうち, Cognitive Complexity および Cyclometric Complexity と OSS の生存性の関係について調べた. 具体的には, GitHub の対象レポジトリから得られるそれらのコードメトリクスとその他 3 つの変数を説明変数とした, OSS が現在生死のいずれの状況であるかを分類するモデルを構築した. 検証データセットを用いた予測結果では, 81% 以上の精度で OSS の生死を予測することに成功している. この結果は, コードの品質が OSS プロジェクトの生存性を説明する要因である可能性を示している. 一方で, コード品質は OSS 採用の意思決定にそれほど寄与しないことを発見した, Siyuan ら [20] の研究がある. これらの研究から, OSS の採用側は OSS のコード品質には無関心であるが, コミュニティ側はコード品質が低い場合に開発を停止する可能性があり, 両者のコード品質への態度にギャップがあることが推測できる.

OSS の持続性を考える上で, 多くのユーザーに採用されることも重要な側面である. そういった意味で, 多くの開発者が採用する OSS に関する特性の理解は欠かせない. Li ら [21] は, OSS の採用側が OSS を採用する際に考慮している要因や評価基準に関する調査を行った. その結果, 採用者側が最も重要視する要素は, ライセンス, コミュニティサポート, パフォーマンス, リスクであった. 特に, ライセンスの種類は訴訟に発展することなく OSS を組み込むために重要であると述べられている. 実際, 制限の弱いライセンスから制限の強いライセンスへの変更はコミュニティの分離を促進するケースも観測されている. Key Value Store(KVS) の代表的なソフトウェアである Redis[22] は, 2024 年に BSD3 ライセンスから RSALv2 と SSPLv1 のデュアルライセンスへの変更を行った. デュアルライセンスへの変更は Amazon Web Services(AWS) などのクラウドプロバイダが Redis を利用する際に制限を受けるもので, 彼らが実質的に Redis を無償で利用することができなくなることを意味する. そのため, AWS などは Redis のソースコードをフォークし, 新たに valkey というプロジェクトを立ち上げた [23]. クラウドプロバイダは Redis に変わり valkey をクラ

ウド上でホストすることになるため, Redis は潜在的なユーザーを減少させた可能性がある.

その他に, Li ら [21] は OSS 採用に関する重大な要素は, 時間の経過とともに変化してきており, セキュリティは OSS の品質を評価するための重要な要素になりつつあるとが述べている.

2.4 エコシステムと外部環境

エコシステムと外部環境は, 本研究が属する研究分野である. この分野では, OSS プロジェクトが属するエコシステムや, スポンサーシップの有無, 競合の台頭などの様々な外部要因が OSS プロジェクトの持続性に与える影響を調査する.

OSS はそれ単体で提供されることは稀であり, 別の OSS に依存し, その依存先もまた別の OSS に依存するという複雑なネットワーク関係をもつ. つまり, 何かしらの OSS を採用することは, その OSS が持つリスクのみならず, 依存先の OSS がもつリスクをも受容することを意味する. Valiev ら [12] は, PyPI エコシステムにおける持続可能性に影響を与えるエコシステムレベルの要因を調査するための混合手法研究を行った. 調査の結果, 上流の依存が多い場合, 開発初期段階は有利に働くが, 長期的には依存先のプロジェクトの開発停止に伴う互換性の維持により開発コストが増大し, 持続可能性に対するリスクが大きくなることが分かった. 反対に, 下流の依存関係が多い場合, 下流プロジェクトからのリソース流入が期待されるため, 長期的に見てプロジェクトの持続可能性にプラスになると述べられている.

多くの OSS 開発では金銭的な報酬は発生せず, 開発の原動力は開発者の意欲やボランティア精神である. 一方で, OSS 開発にはスポンサーシップやユーザーからの寄付といった金銭的報酬が与えられる場合があり, この報酬の有無が OSS 開発の持続性に与える影響も見過ごせない. Yuhang ら [24] は, OSS プロジェクトの活動量とスポンサーシップの間には正の相関関係があることを発見した. 同様に, GitHub のプロジェクト支援機能である, GitHub Sponsors プログラムの効果を測定した Jing ら [25] の調査では, プログラム開始後, スポンサー可能なユーザーは, そうでないユーザーに比べて, 毎月作成するリポジトリ数が 54%増加する調査結果を得た. これらの結果は, 金銭的報酬が開発者の活動に関するモチベーションを高め, 活動量の増加につながることを示唆する. 一方で, 金銭的な報酬はすべての開発者が受諾するわけではなく, 報酬の仕組みに大きく左右される複雑な問題であることが, Krishnamurthy ら [26] により指摘されている. また, 報酬を得られる者と得られない者の不均衡も存在し, この不公平性は開発者のモチベーションを低下させる恐れがあると Arzoo ら [27] が指摘している. これらの研究から, OSS 開発者の貢献意欲を損なわないような公平で透明性の高いインセンティブ設計が重要であると考えられる.

OSS の競合は, OSS だけではなく, PS(プロプライエタリソフトウェア) も存在する. Ravi[28] は, OSS-SS（サポートサービス付き OSS）と OSS（サポートサービスなし OSS）が PS の市場ポジションに悪影響を与える条件と, PS ベンダーが競争に打ち勝つための戦略について考察した. その結果, ネットワーク効果が弱く OSS-SS

の使いやすさが低い市場ではOSSが優勢になり, ネットワーク効果が強くOSS-SSの使いやすさが高い市場ではPSが優勢になることが示唆されている. ここで, ネットワーク効果が強い市場は, デスクトップオフィス生産性ソフトウェア, デスクトップオペレーティングシステムなどの市場である.

多くのソーシャルメディアでは, ポストに”いいね”が多く集まるほど人々の目に留まりやすくなるようなアルゴリズムが組まれている. GitHubにも, ”スター”と呼ばれるSNSの”いいね”に相当する機能が提供されており, スター数が多いプロジェクトほど検索の上位に来やすく, またメディアに取り上げられるなど開発者の目に留まる機会が多い傾向にある. 実際, Borgesら [29] が行った開発者へのインタビュー調査では, 回答者の4人に3人が,GitHubプロジェクト利用前にスター数を考慮することが確認された. また, 同研究内で, GitHub上のスター数上位5,000件のリポジトリを対象に定量的な調査を行ったところ, スター数とリポジトリの年齢間に相関関係は見られなかったものの, 貢献者およびフォークの数と中程度の相関関係があることが分かった. つまり, 開発者を引き付ける意味でスター数は重要な意味を持つ可能性がある.

2.5 本研究の新規性

これまでに挙げた4つのカテゴリのうち, コミュニティ構造と貢献者の動態, プロジェクト管理とガバナンス, プロジェクト特性と技術的要因の3つは, 主に内的要因とOSSの生存性の関係性を研究する分野である. これらの分野に属する研究では, 様々な内的要因がOSSの生存性に与える影響を明らかにし, 重要な知見を提供した. しかし, これらは内的要因に閉じたものであり, OSSプロジェクト間の競争関係といった外的要因を明示的に扱っているわけではない.

残りのエコシステムと外部要因についても, 依存関係ネットワークやスポンサーシップ, GitHubスターといった外部要因とOSSの生存性との関係性について重要な知見を提供した. 外的要因を扱うという意味において本研究と方向性は一致しているが, これらの研究も, 本研究が着目するOSSプロジェクト間の競争関係を明示的に扱っているわけではない. その他にも, Coelhoら [4] は, OSSの衰退要因で最も多いものとして”陳腐化”や”強力な競合の出現”に言及しているものの, 事後の定性的な分析にとどまっており, これを定量化あるいは予期するような手法を提案しているわけではない.

本研究では, OSSの競争関係に着目し, OSSプロジェクトどうしの時間的な影響を定量的に評価する手法と, この定量データを用いて将来REVが発生するかを予測するためのモデルを提案する. 本研究の新規性は, マクロ経済学で広く使われる手法を応用し, これまで定性的に扱われてきたOSSの競合関係を定量化することにある.

第3章　準備

第 2 章では，OSS の生存性に関する先行研究を概観するとともに，本研究が着目する「OSS 間の競合関係」について議論を行った. 本研究では，複数の OSS が時間的に互いへ与え合う影響を捉えるために，マクロ経済学の領域で広く用いられてきた VAR モデル（Vector Autoregression）や SVAR モデル（Structural VAR），そしてインパルス応答関数（IRF）を応用する. そこで本章では，MIAO や REV 予測モデルを理論的に支える「時系列解析」の基盤知識と，主要な数理モデルについて整理・説明することを目的とする.

具体的にはまず，時系列データがもつ定常性や自己共分散といった基礎概念を復習し，VAR モデルを構築・運用するために必要となる統計的性質を明確にする. 次に，通常の VAR モデルでは把握が困難な同時点における変数間の関係性を表現する SVAR モデルを導入し，識別制約やインパルス応答関数の計算過程について述べる. 最後に，OSS 間の累積的な影響を定量化するうえで重要となる「累積インパルス応答関数」や，本研究で用いる機械学習手法の概要にも触れることで，後続章における提案手法（MIAO と REV 予測モデル）の技術的背景を明らかにする.

3.1　時系列の性質

3.1.1　定常過程

本研究は OSS の時系列データと VAR モデルを主に用いるが, VAR モデルにおいては時系列が定常過程であることが要求される. そのため, 時系列が定常であるか, 非定常あるかを区別する必要がある. 特に, 本研究で定常過程というとそれは弱定常過程を意味する. 時系列 $T = (y_t)$ が定常過程であるとは次を満たすことである.

1. T の平均が時刻 t に依存せず一定となる. すなわち, $\mathrm{E}(t) = \mu$
2. T の分散が時刻 t に依存せず一定となる. すなわち, $\mathrm{Var}(y_t) = \sigma^2$
3. T の自己共分散が時刻 t に依存せず, ラグ k によってのみ定まる. すなわち, $\mathrm{Cov}(y_t, y_{t-k}) = Ck$

一方, 非定常過程はこれらを満たさず, 時刻 t によって統計的性質が変化する確率過程といえる. 例えば, トレンドを持つ時系列は非定常過程の代表的なものとなる. 時系列が定常であるか非定常であるかは, その時系列が単位根をもつかを確認する. 単位根の有無は, 単位根検定と呼ばれる種類の統計的仮説検定を用いるのが一般的

であり, Dickey ら [30] が提案した DF 検定（Dickey-Fuller Test）がよく用いられる. VAR モデルの場合はこれを多変量に拡張した, ADF 検定 (Augmented Dickey-Fuller Test) を用いることになる.

仮に時系列が単位根を持つ場合, その時系列は単位根過程と呼ばれ, 非定常な確率過程となる. 時系列が単位根を持つ場合でも, 階差を取ると定常となることがある. 階差とは, 1 時点前のデータとの差をとった時系列データのことで, 階差を取って得られる時系列を階差系列という. 階差は複数回取ることができる. つまり, 新たに得られた階差系列からさらに階差系列を作ることができるが, 元の時系列が持つ情報が次第に失われていく点には注意が必要である. Marcos[31] によれば, 整数による階差系列はどんな系列のデータに対しても情報を削除しすぎていることを指摘しており, 3/5 以下で差分を取ることで定常性が得られると主張している. このように, 小数の範囲で差分を取って得られる系列を分数差分系列と呼ぶ.

分数差分系列の導入は次である. $B^k X_t = X_{t-k}$ となるような作用素 B をバックシフトオペレータとすると, 1 階の整数差分系列は $(1-B)X_t = X_t - X_{t-1}$, 2 階の整数差分系列は $(1-B)^2 X_t = X_t - 2X_{t-1} + X_{t-1}^2$ となる. これを一般化すると次となる.

$$\begin{aligned}(1-B)^d &= \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k \\ &= 1 - dB + \frac{d(d-1)}{2!}B^2 + \frac{d(d-1)(d-2)}{3!}B^2 + \cdots\end{aligned}$$

この一般式を使って分数差分系列は次のような列で表せる.

$$\omega = \left\{1, -d, \frac{d(d-1)}{2!}B^2, \frac{d(d-1)(d-2)}{3!}B^2, \cdots, (-1)^k \Pi_{i=0}^{k-1} \frac{d-i}{k!}, \cdots\right\}$$

3.1.2 ホワイトノイズ

VAR モデルのような時系列予測モデルの誤差項は単なる誤差ではなく, モデルが説明できないランダム要素として重要な役割を担う. 特に VAR モデルや AR モデルにおける誤差項はホワイトノイズとして分布することが仮定されており, モデルの精度を議論するうえで誤差項がホワイトノイズに従うかの検証も必要になってくる. 時系列 $T = (y_t)$ がホワイトノイズであるとは次を満たすことである.

1. $\mathrm{E}(y_t) = 0$
2. $k = 0$ のとき, $\mathrm{Cov}(y_t, y_{t-k}) = \sigma^2$
3. $k \neq 0$ のとき, $\mathrm{Cov}(y_t, y_{t-k}) = 0$

この定義から, ホワイトノイズも自己相関を持たない定常過程であることが分かる. 時系列が自己相関を持たないことは, 一連のかばん検定により検証できる. かばん検

定では, Box-Pierce 検定 [32] や Ljung-Box 検定 [33] など複数の手法が用意されているが, それぞれ検出力などが異なる. 近年では Ljung-Box 検定の使用が推奨されている. Ljung-Box 検定は H_0 がラグ 1 からラグ n までの全てで自己相関が 0, H_1 がラグ 1 から m までの自己相関のうち, 少なくとも一つが 0 でない, である.

3.1.3 季節変動

季節変動とは, 時系列データにおいて規則的に繰り返される周期的な変動のことを指す. 例えば, 小売業の売上高データでは, クリスマスや年末シーズンに毎年上昇するといった変動パターンが見られることが多い. このような季節変動は, 時系列データの分析や予測において重要な要素となる. 季節変動を含む時系列データは, 一般的に次の 4 つの要素に分解できる.

1. トレンド成分: 長期的な増加または減少傾向
2. 季節成分: 一定期間ごとに繰り返される変動
3. 循環成分: 長期的な周期での変動（景気循環など）
4. 不規則成分: 上記以外のランダムな変動

季節変動の分析や除去には, 様々な手法が存在するが, 代表的なものに STL (Seasonal and Trend decomposition using Loess) 分解 [34] がある. STL 分解は, ローカル回帰（Loess）を用いて時系列データをトレンド (Trend), 季節 (Seasonal), 残差 (Residual) 成分に分解する手法である. すなわち, 原系列 T_t を次のように構成するものである.

$$T_t = \mathrm{Trend}_t + \mathrm{Seasonal}_t + \mathrm{Residual}_t \tag{3.1}$$

STL 分解は, 複雑な季節パターンや欠損値を含むデータにも適用可能であり, 多くの分野で使用されている. 季節性を取り除くことで時系列データの定常性を担保しやすくなるため, VAR のような定常性を前提とするモデルの信頼性を高めることができる.

3.2 VAR モデル

VAR(Vector Autoregression) モデルは, AR(Autoregression) モデルを多変量に拡張したモデルである. AR モデルは 1 次の時系列予測モデルで, 自身の過去の値を用いた回帰式により将来の値を予測する. VAR モデルもオリジナルが AR モデルであるので, 時系列の将来の値を自身の過去の値を用いて予測するという根本に相違はなく, 複数の時系列の関連を用いてより複雑に時系列を予測することが可能となっている. 標準的な VAR モデルは次式で表される.

$$y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + \epsilon_t \tag{3.2}$$

式 (3.2) は VAR モデルの誘導形と称され, 後述する構造形と区別される. 本式において, A_i は $n \times n$ 係数行列, c は $n \times 1$ 定数ベクトル, ϵ_t は撹乱項（誤差項）である. このとき, Σ を撹乱項 ϵ_t の分散共分散行列とすると, 確率変数としてみた ϵ_t は, $\epsilon_t \sim \mathrm{W.N}(\Sigma)$ となる必要がある. W.N はベクトルホワイトノイズである.

VAR は n 変量の関係をモデル化するものであるから, n 変量 VAR(p) モデルのように用いる. 具体的には, 2 つの時系列 y_1, y_2 および, ラグ次数 $p = 1$ とした際の 2 変量 VAR(1) モデルを次式で例示する.

$$\begin{cases} y_{1t} = c_1 + a_{11} y_{1,t-1} + a_{12} y_{2,t-1} + \epsilon_{1t} \\ y_{2t} = c_2 + a_{21} y_{1,t-1} + a_{22} y_{2,t-1} + \epsilon_{2t} \end{cases} \tag{3.3}$$

このように, 各変数ごとに方程式がそれぞれ現れるのが VAR モデルである. 次数やラグ数が増えると, それに伴い方程式の数と各回帰式の項数も増えていく. VAR モデルでは, ラグ次数 p の選択がモデルの性能に直接的に寄与する. ラグが大きいほど過去の情報を多く取り込むことになり, モデルが複雑化し過学習の懸念がある. 一方, ラグ次数が小さすぎれば時系列の持つ情報をモデルに取り込めきれずに, 精度の低いものとなる. ラグ次数の選択手法としては情報量基準を用いるのが一般的である. 情報量基準のうち, よく利用されるものに, AIC（Akaike information criterion）[35], SC(Schwarz Bayesian information criterion), HQ(Hannan–Quinn information criterion)[36] などがある. このうち, いずれを選択するかは利用者に委ねられるが, 村尾 [37] によれば, 一般的な傾向としては SC と HQ がラグ次数を漸近的に正しく選ぶ傾向にあり, AIC はラグ次数を漸近的に過剰に選ぶ傾向があると主張している. p(IC) を与えられた情報量基準に従いラグ次数を選ぶ関数とすると, 次の関係となることが知られている.

$$p(\mathrm{SC}) \leq p(\mathrm{HQ}) \leq p(\mathrm{AIC})$$

3.3 SVAR モデル

VAR モデルでは, 同日, 同月といった同時点の影響を明示的にモデル化できず, 同時点の影響が撹乱項に残存してしまうという問題がある. これにより, 各撹乱項が相互に相関する可能性があり, モデルが説明している各変数の関係が誤差を通じた間接的な関係として歪んでしまう可能性がある. SVAR(Structual Vector Autoregression) モデルは, 同時点の変数間の関係をモデル設計者が明示的に指定することができる VAR モデルである. これにより, 撹乱項から同時点の影響を取り除き, 撹乱項は互いに相関をもたないものとして直接的な関係性をモデル化できるようになる. 標準的な SVAR モデルは次式で表される.

$$B_0 y_t = c + B_1 y_{t-1} + B_2 y_{t-2} + \cdots + B_2 p_{t-p} + u_t, \quad u_t \sim \text{W.N}(\Sigma_\epsilon) \tag{3.4}$$

$$\Sigma_\epsilon = E(u_t, u_t') = \begin{pmatrix} \sigma_{u_1}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{u_2}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{u_p}^2 \end{pmatrix}$$

式 (3.4) で表される VAR モデルを VAR の構造形と呼ぶ. u_t は構造ショックと呼ばれる誤差項である. W.N はベクトルホワイトノイズを表し, 確率過程として見た u_t は自己相関を持たない弱定常過程となることは, 通常の VAR の ϵ_t と同様である. 通常の VAR との最大の差異は, 左辺に B_0 が乗じられていることである. これは, 同時点の変数間の関係を捉える役割を果たし, これにより u_t は互いに相関を持たない. つまり, u_t は特定の OSS プロジェクトに固有のイノベーションとして解釈でき, インパルス応答を計算する際に, ある OSS の純粋な影響を他の OSS の影響から分離可能にする.

3.4 VAR(p) の VMA 表現

VMA(Vector moving average representation) の基礎は, 無限次の移動平均 (Voving average) 過程を多次元に拡張したものである. VAR は過去のデータを用いて将来を自己回帰的に予測する枠組みであるが, このとき得られる式 (3.2) の右辺は, 同時点や過去時点における ϵ_t が将来の y_{t+k} にどれほど影響を与えるかを直接的に示すものではない. したがって, より直接的に誤差項の累積的な影響を計算可能にするのが, VMA 表現である. VMA 表現に変換する際に得られる係数行列がインパルスに相当することになる. ここでは, VAR を VMA 表現に変換するメカニズムを概観する. まず, VMA 表現への変形のため, ラグオペレータ L を次のように導入する.

$$L^k y_t = y_{t-k}$$

L を用いて VAR(p) をコンパクトに表現すると次のようになる.

$$A(L) y_t = c + \epsilon_t$$

ただし,

$$A(L) = I_k - A_1 L - A_2 L^2 - \cdots - A_p L^p$$

この表現は, 例えば VAR(1) において

$$\begin{aligned} A(L)y_t &= c + \epsilon_t \\ &= (I_k - A_1 L)y_t = c + \epsilon_t \\ &= I_k y_t - A_1 y_{t-1} = c + \epsilon_t \\ &= I_k y_t = A_1 y_{t-1} + c + \epsilon_t \\ &= y_t = A_1 y_{t-1} + c + \epsilon_t \end{aligned}$$

なるものである.

次に, VAR(p) を VAR(1) 形式へ変換し, 第 $t-k-1$ 期方程式の y_{t-k-1} を第 t 期方程式の y_{t-k} に挿入するということを無限回繰り返す. つまり

1 期目

$$y_t = A_1 y_{t-1} + c + \epsilon_t$$

2 期目

$$y_{t-1} = A_2 y_{t-2} + c + \epsilon_t$$

3 期目

$$y_{t-2} = A_3 y_{t-3} + c + \epsilon_t$$

とすれば, 1 期目に第 2 期目を挿入して

$$y_t = A_1(A_2 y_{t-2} + c + \epsilon_t) + c + \epsilon_t$$

第 3 期目を挿入して

$$y_t = A_1(A_2(A_3 y_{t-3} + c + \epsilon_t) + c + \epsilon_t) + c + \epsilon_t$$

とするものである. この結果, 次の表現を得ることができる.

$$\begin{aligned} y_t &= \mu + \epsilon_t + \Psi_1 \epsilon_{t-1} + \Psi_2 \epsilon_{t-2} + \cdots \\ &= \mu + \sum_{s=0}^{\infty} A_s \epsilon_{t-s} \\ &= \mu + A(L)\epsilon_t \end{aligned} \tag{3.5}$$

ただし,

$$
\begin{aligned}
A(L) &= A(L)^{-1} \\
&= [I_k - A_1 L - A_2 L^2 - \cdots - A_p L^p]^{-1} \\
&= I_k + A_1 L + A_2 L^2 + A_3 L^3 + \cdots \\
\\
\mu &= A(L)^{-1} c \\
&= A(1)^{-1} c \\
&= [Ik - A_1 - A_2 - \cdots - A_p]
\end{aligned}
$$

式 (3.5) はベクトル移動平均過程 (Vector moving average process) になっており, これを VMA(∞) と書く.

3.5 インパルス応答関数

インパルス応答関数（Impulse Response Function, IRF）は, n 変量 VAR(p) モデルにおいて, y_{jt} の撹乱項 ϵ_{jt} だけに 1 単位のショックを与えたときの k 期後の $y_{i,t+k}$ の値の変化を k の関数とみたものである. インパルス応答は, 式 (3.5) の VMA(∞) 表現で得られる y_t をその撹乱項 ϵ_t で偏微分することで得られる. すなわち,

$$
\frac{\partial y_t}{\partial \epsilon_t} = I_k = \Psi_0, \ \frac{\partial y_{t+1}}{\partial \epsilon_t} = \Psi_1, \ \frac{\partial y_{t+2}}{\partial \epsilon_t} = \Psi_2, \ \cdots, \ \frac{\partial y_{t+k}}{\partial \epsilon_t} = \Psi_k \tag{3.6}
$$

となる. これらはショック ϵ_t の各変数への影響が時間を通じてどのように変化するかを表しているといえる. ここで, Ψ_k の (i, j) 成分は,

$$
\frac{\partial y_{i,t+k}}{\partial \epsilon_{jt}}
$$

を表しており, j 番目の ϵ_{jt} から i 番目の $y_{i,t+k}$ への影響を表している. このように, 式 (5) の時系列を経過時間 k の関数とみたものをインパルス応答関数と呼び, 次のように表記する.

$$
\mathrm{IRF}_{ij}(k) = \frac{\partial y_{i,t+k}}{\partial \epsilon_{jt}} \tag{3.7}
$$

一方, SVAR のインパルス応答は通常の VAR に比べ複雑なプロセスを踏む. なぜならば, SVAR モデルは同時点の変数を説明変数として含む同時決定モデルであるからである. 同時決定モデルとは, 経済システムにおいて複数の変数が相互に影響を与え合い, 同時に決定される状況を表現するモデルのことである. よって, 誤差項と説明変数の間に相関が生じる可能性があり, そのまま OLS(Ordinary least squares) を行うことができない. なぜならば, VAR の文脈における OLS では, 誤差項と説明変数が無相関であるという重要な仮定があるためである. そのため, 一度構造形を同時点の変数を含まない誘導形に変換したうえで OLS 推計する. そして, その推定結果に識別制約を課すことで構造パラメータを復元する.

式 (3.4) を誘導形に変換するには, 単純に両辺に左から B_0^{-1} を乗じるのみである.

$$y_t = B_0^{-1}c + B_0^{-1}B_1 y_{t-1} + B_0^{-1}B_2 y_{t-2} + \cdots + B_0^{-1}B_p y_{t-p} + B_0^{-1}u_t \tag{3.8}$$

ここで, $\mu = B_0^{-1}c,\ A_k = B_0^{-1}B_k,\ \epsilon_t = B_0^{-1}u_t$ として, ラグオペレータ L を用いると次の形で表せる.

$$\begin{aligned} y_t &= \mu + A(L)y_t + \epsilon_t, \\ &\Longrightarrow E(\epsilon_t, \epsilon_t') = \Sigma_\epsilon = B_0^{-1}E(u_t, u_t')(B_0^{-1})' = B_0^{-1}\Sigma_u(B_0^{-1})' \end{aligned} \tag{3.9}$$

ここからは, 構造パラメータを復元する. ただし, 誘導形から構造形を常に復元できるわけではなく, そこには識別性という問題が存在する. なぜならば, 構造形は $n(n-1)+n(np+1)+n$ 個のパラメータを含むが, 誘導形は $n(np+1)+n(n+1)/2$ 個のパラメータしか含まないため, 構造形の復元に $n(n-1)/2$ 個パラメータが足りなくなってしまう. したがって, 構造形を識別するために $n(n-1)/2$ 個のパラメータを追加で用意する必要が出てくる. 識別問題の一般的な解決策は, B_0 に下三角行列を仮定するリカーシブ (recursive) な同時点制約を用いることである. 技術的には, コレスキー分解を用いて共分散行列 Σ_ϵ を分解し, B_0 を推定するというものである. 誘導形となった撹乱項のベクトルの分散共分散行列 Σ_ϵ は, $\Sigma_\epsilon = PP'$ を満たす一意的な下三角行列 P が存在することが知らている. P はコレスキー因子と呼ばれる. さらに, 分散共分散行列 Σ_ϵ を用いて, $\Sigma_\epsilon = PP' = B^{-1}\Sigma_u(B^{-1})'$ と分解することができる. したがって, $P = B^{-1}\sum_u^{\frac{1}{2}}$ である. B^{-1} は対角要素が 1 の下三角行列であるので, 式 (3.9) の B_0^{-1} をこの B^{-1} と捉えれば, B_0 の識別に必要な残りの $n(n-1)/2$ 個の制約が満たされることになる. ここまでから, (3.9) を誘導形ショック ϵ による VMA 表現として次に変形できる.

$$\begin{aligned} \{I - A(L)\}y_t &= \mu + \epsilon_t \\ y_t &= \frac{\mu}{\{I - A(1)\}} + \frac{1}{\{I - A(L)\}}\epsilon_t \\ &= \tilde{\mu} + \Psi_0\epsilon_t + \Psi_1\epsilon_{t-1} + \Psi_2\epsilon_{t-2} + \cdots \\ &= \tilde{\mu} + \Psi(L)\epsilon_t \end{aligned} \tag{3.10}$$

ただし, $\tilde{\mu} = \dfrac{\mu}{\{I - \Phi(1)\}} = \dfrac{\mu}{I - \Phi_1 - \Phi_2}$ は定数項ベクトルで, $\Psi_k(s = 0, 1, 2, \cdots)$ は誘導形ショック 1 単位に対する各変数の k 期後の反応を表す係数行列である. (3.10) のままでは複数の構造ショックが内在しているため, 本来の構造ショック u_t で偏微分できるようにさらに変形する. $\epsilon_t = B_0^{-1}u_t, \mu = B_0^{-1}c$ であったことを思い出し,

$$\begin{aligned} y_t &= \frac{B_0^{-1}c}{\{I - A(1)\}} + \frac{1}{\{I - A(L)\}}B_0^{-1}u_t \\ &= \tilde{c} + \Psi_0 B_0^{-1}u_t + \Psi_1 B_0^{-1}u_{t-1} + \Psi_2 B_0^{-1}u_{t-2} + \cdots \\ &= \tilde{c} + \Psi(L)B_0^{-1}u_t \end{aligned} \tag{3.11}$$

である. 最終的に u_t で偏微分可能となったインパルス応答 (3.12) を得る.

$$\mathrm{IRF}_{ij}(k) = \frac{\partial y_{i,t+k}}{\partial u_{jt}} = [\Psi_k B_0^{-1}]_{ij} \tag{3.12}$$

このように, SVAR モデルにリカーシブ制約を課したものを再帰的構造 VAR モデルと呼ぶ. 再帰的構造 VAR モデルでは, インパルス応答に上三角行列 B_0^{-1} がかかっていることから分かるように, 変数を $A, B, C, \cdots$ と並べたときに, $A \to B \to C, \cdots$ なる因果関係を仮定していることに注意する必要がある. つまり, 同時点において A は $B, C, \cdots$ に影響を与えるが, 一方で $B, C, \cdots$ は A に影響を与えない, あるいは B は $C, \cdots$ に影響を与えるが逆に $C, \cdots$ からの影響は受けず, A から影響を受けるといった構造がモデルに組み込まれているのである. したがって, 変数のオーダーによってインパルス応答が変化してしまう. こういった問題に対し, 経済学では外生性が高い順に変数を並べるといった慣例がある. 外生性が高いとは, ある変数が他の経済変数や経済システムから独立して決定されるという意味で, モデルの中で説明される他の要因からの影響をほとんど受けない変数のことを指す.

なお, SVAR の識別制約はリカーシブ制約以外にも, 経済理論に基づく様々なものを課すことができる. ただし, 与えられた制約によりモデルが非再帰的構造を取る場合, 先に挙げた識別問題が付きまとう. 仮に非再帰的構造を選択する場合は, モデルの識別に際してよく検討された経済理論に基づき実行する必要がある.

3.6 累積インパルス応答関数

累積インパルス応答関数は, 第 k 期における IRF を累積した関数である.

$$\mathrm{IRF\_CUM}_{ij}(k) = \sum_{k=1}^{\infty} \mathrm{IRF}_{ij}(k) \tag{3.13}$$

3.7 重回帰分析

重回帰分析は, 複数の説明変数と 1 つの目的変数との関係を分析する統計手法である. この手法は, 説明変数の線形結合によって目的変数を予測することを目的としており, 最小二乗法を用いて回帰係数を推定する. 重回帰分析のモデルは以下の式で表現される

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon \tag{3.14}$$

ここで, Y は目的変数, $X_1, X_2, \ldots, X_p$ は説明変数, $\beta_0, \beta_1, \ldots, \beta_p$ は回帰係数, ε は誤差項を表す.

最小二乗法は, 実際の観測値と予測値の差（残差）の二乗和を最小化することで最適な回帰係数を求める方法である. 目的関数は以下のように表される.

$$\min \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \tag{3.15}$$

ここで, Y_i は実際の観測値, $\hat{Y}_i$ は予測値である. 最小二乗法による解は, 以下の正規方程式を解くことで得られる.

$$(\mathbf{X}'\mathbf{X})\boldsymbol{\beta} = \mathbf{X}'\mathbf{Y} \tag{3.16}$$

ここで, $\mathbf{X}$ は説明変数の行列, $\mathbf{Y}$ は目的変数のベクトル, $\boldsymbol{\beta}$ は求めるべき回帰係数のベクトルである. 回帰係数の推定値は以下の式で与えられる.

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \tag{3.17}$$

重回帰分析において説明変数間に強い相関関係が存在する状況があり, これを多重共線性という. この問題は回帰係数の推定精度を低下させ, モデルの解釈を困難にする可能性がある [38]. 多重共線性が存在する場合, 以下のような問題が生じる.

- 回帰係数の推定値が不安定になる
- 標準誤差が大きくなり, 統計的有意性の検定が困難になる
- 個々の説明変数の効果を正確に分離することが困難になる

多重共線性の検出と評価には, 分散拡大要因（Variance Inflation Factor, VIF）が広く用いられる. VIF は各説明変数について計算され, 以下の式で定義される:

$$VIF_j = \frac{1}{1 - R_j^2} \tag{3.18}$$

ここで, R_j^2 は j 番目の説明変数を他のすべての説明変数で回帰した際の決定係数である. VIF の解釈と多重共線性の回避について, 以下のガイドラインが一般的に用いられる.

- VIF < 5: 多重共線性の可能性は低い
- $5 \geq$ VIF < 10: 多重共線性の可能性があり, 注意が必要
- VIF ≤ 10: 深刻な多重共線性の問題があり, 対処が必要

多重共線性を回避するためには, 相関の高い変数の一方を除外する, 主成分分析（PCA）を用いて変数を合成する, 説明変数の中心化や標準化を行う, などがある.

VIF を用いた多重共線性の検出と回避は, 重回帰分析の前処理として重要なステップである. これにより, モデルの安定性と解釈可能性を向上させ, より信頼性の高い分析結果を得ることができる.

3.8 LightGBM

LightGBM[39] は, Microsoft が提案した GBDT（Gradient Boosting Decision Tree, 勾配ブースティング決定木）アルゴリズムであり, 従来の GBDT アルゴリズムと比較して, 高速で効率的な学習が可能である. 主な特徴として, GOSS（Gradient-based One-Side Sampling）と EFB（Exclusive Feature Bundling）という 2 つの手法を導入している. GOSS は, 勾配に基づいてインスタンスをサンプリングする手法で, 学習データの中から重要なインスタンスを選択し, 計算コストを削減する. 一方, EFB は特徴量のバンドリング技術で, 互いに排他的な特徴を束ねることで特徴空間の次元を削減し, メモリ使用量を減らす. LightGBM の学習プロセスは以下の式で表現できる.

$$\hat{y}_i = \sum k = 1^K f_k(x_i), \quad f_k \in \mathcal{F} \tag{3.19}$$

ここで, $\hat{y}_i$ は予測値, x_i は入力特徴量, f_k は k 番目の決定木, $\mathcal{F}$ は可能な決定木の集合を表す. モデルの最適化は以下の目的関数を最小化することで行われる.

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}i) + \sum k = 1^K \Omega(f_k) \tag{3.20}$$

ここで, l は損失関数, Ω は正則化項である. LightGBM は, これらの技術により大規模なデータセットに対しても高速な学習が可能となっている.

第4章　Mutual Impact Analysis of OSS (MIAO)

Mutual Impact Analysis of OSS(MIAO) は, IRF を用いて, OSS の活動が時間経過を通じてその他の OSS に与える影響を定量化する手法である. MIAO は, 数理的には VAR および IRF を用いることから, OSS の生存性を表す変数は時系列データで表現される. この時系列データを本研究では, アクティビティ時系列データと称する. この言葉を用いて MIAO を言い換えれば, アクティビティ時系列データに対するショックがその他の OSS のアクティビティ時系列データにどのように伝搬するかをインパルス応答という形で数値化するものである. MIAO 自体は REV を判定するものではなく, あくまで複数の OSS 間の関係性を記述するためのものである.

アクティビティ時系列データはどのようなデータでも構わないが, OSS の生存性を表すことができる時系列である必要がある. Kalliamvakou[40] らによれば, OSS の活動日数とコミット数には正の相関 (0.61) が見られ, プロジェクトの活動日数が長いほど, コミット数も多くなる傾向がある. OSS の活動日数はまさに生存性を表す指標であるため, 本研究ではコミット数の推移を用いる. したがって, 単位期間あたりのコミット数の推移をアクティビティ時系列データと考えれば, その OSS のコミット数が上昇した際に, その他の OSS のコミット数の具体的な増減を時間経過を考慮して定量化することが MIAO の仕事である. MIAO のアルゴリズムは大別すると次の 2 フェーズに分かれる. フェーズ 1 では, 複数のアクティビティ時系列データを入力として, IRF を計算し, フェーズ 2 では, 得られた IRF から, OSS 間の相対的な影響の大きさを表す実数値である MIAO スコアを構成する.

本章では, MIAO のフェーズ 1, フェーズ 2 の理論的な説明を行い, MIAO を実データに適用した実験を行う. 最終的に, 実験結果の評価を通して MIAO の有効性を検証する.

4.1　MIAO フェーズ 1

フェーズ 1 は一般的な IRF の計算過程に基づくものである. 一般的な IRF の計算過程とは, 1: 時系列データの前処理（季節調整など）, 2: 時系列の定常性の確認, 3: 共和分の有無の確認（必要な場合のみ）, 4: VAR のための最適なラグ次数の推定, 5: VAR の推定, 6: VAR の評価, 7: VAR を VMA に変換, 8: VMA から IRF の計算, である.

ただし, MIAO では複数の分析期間を設けて, OSS 間の時間経過に伴う影響の変化を

Algorithm MIAO Phase1

Input: $\mathcal{A} = [A_1, A_2, \cdots, A_n]$, $\mathcal{T} = [T_1, T_2, \cdots, T_m]$
Output: $\mathcal{O} := m \times n \times n \times k$ の 4 次テンソル

$k :=$ IRF を計算する期間 $(k = 1, 2, \cdots)$
$max\_lag :=$ VAR の最大ラグ次数
$ic :=$AIC or BIC or HQIC
function PREPARE_VAR($\mathcal{D}$)
    $\mathcal{D}' \leftarrow []$
    for each $A_{T_m} \in \mathcal{D}$ **do**
        $A_{T_m} \leftarrow$ PREPROCESS(A_{T_m})
        if ADF_TEST(A_{T_m}) ≥ 0.05 **then**
            $n \leftarrow [0.1, 1]$ で, FRAC_DIFF(A_{T_m}, n) が定常となる最小数
            $A_{T_m} \leftarrow$ FRAC_DIFF(A_{T_m}, n)
        end if
        $\mathcal{D}' \leftarrow \mathcal{D}' + [A_{T_m}]$
    end for
    histories = []
    $lags \leftarrow$ ICS($\mathcal{D}', max\_lag, ic$)
    for $lag = 0$ to $|ics|$ **do**
        $\_, \_, \_, u \leftarrow$ SVAR_FIT($\mathcal{D}', lag$)
        $p \leftarrow$ WHITENESS_TEST(u)
        $histories \leftarrow histories + [(ics[lag], lag, p)]$
    end for
    $lag \leftarrow$ VAR_BEST_LAG($histories$)
    return $lag, \mathcal{D}'$
end function

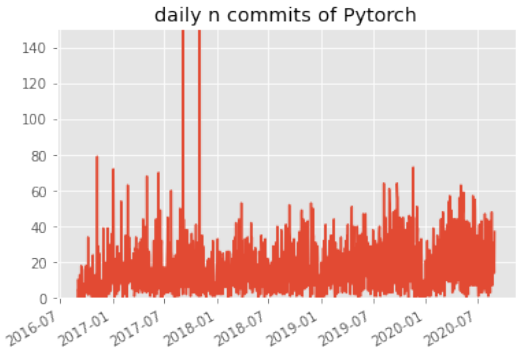
追跡するための改良がなされており, 上記の IRF の計算過程を分析期間の数だけ繰り返す. 復数の分析期間を設ける理由は, OSS のアクティビティと経済変数とで根本的な相違があるからである. 通常, CPI や為替レートといった経済変数は平均回帰という性質を持っている. これは短期的な変動があっても, データがある基準値に戻る傾向のことをいう. この平均回帰により, 経済研究では関係性の予測に短期的なインパルス応答分析を使用することができる. なぜならば, 現在のインパルス応答の傾向が中期的には大きく変わらないと考えられるからである. 一方で, OSS には成長モデルがある. 一般的には, 成長段階では活動量が増加するが, 成熟期に向かうにつれて活動量は減少し, 最終的にはゼロに近づく. 実際, Kuwata ら [18] の提唱するモデルでもそのような傾向が見られる. したがって, 経済変数と比較して, OSS では構造変化が起こりやすく, 平均回帰を前提とした分析では, 分析結果を見誤る可能性がある. このような課題に対して, MIAO では, 分析期間を複数に分割し, それぞれの期間における, 長期の累積的なインパルス応答をもとに OSS 同士の影響を分析するアプローチを取る.

function MIAO_P1($\mathcal{A}, T_m$)
    $\mathcal{D} \leftarrow []$
    for each $A \in \mathcal{A}$ **do**
        $A_{T_m} \leftarrow$ SUBSEQUENCE(A, T_m)
        $\mathcal{D} \leftarrow \mathcal{D} + [A_{T_m}]$
    end for
    $lag, \mathcal{D}' \leftarrow$ PREPARE_VAR($\mathcal{D}$)
    $B_0, B, c, u \leftarrow$ SVAR_FIT($\mathcal{D}', lag$)
    return IRF($B_0, B, c, u, k, \mathcal{D}'$)
end function
for each $T_m \in \mathcal{T}$ **do**
    $Y \leftarrow$ MIAO_P1($\mathcal{A}, T_m$)
    $\mathcal{O} \leftarrow \mathcal{O} + [Y]$
end for

表 4.1: MIAO Phase1 のアルゴリズム内で利用する関数の定義

関数名	説明
$\lvert \cdot \rvert$	リスト内の要素数を返す.
SUBSEQUENCE(A, T_m)	A 中の T_m に対応する期間の部分列を返す.
PREPROCESS(A)	A に前処理を施す. 前処理は時系列の特性によって変動する. 例えば, 季節調整などが該当する.
ADF_TEST(A)	A に ADF 検定を実施し p 値を得る. このとき, トレンド定常を許容する.
FRAC_DIFF(A, n)	分数 n に基づき, A に対して分数差分変換を行う.
ICS($\mathcal{A}, max\_lag, ic$)	引数で渡された ic（情報量基準）において, max_lag までのラグに対応する情報量基準量 の値を配列形式で返す.
SVAR_FIT($\mathcal{A}, lag$)	$\lvert\mathcal{A}\rvert$ 変量 SVAR(lag) を推定する. 出力は式 (3.2) を構成するための B_0, B, c, u のタプル
WHITENESS_TEST(u)	Ljung-Box 検定により撹乱項の系列相関の有無を確認し, p 値を返す
VAR_BEST_LAG($histories$)	histories は (ラグに対応する情報量基準値, ラグ次数, Ljung-Box 検定の p 値) のタプルが要素の配列. histories において, Ljung-Box 検定の p 値が 0.1 を超えるものの中で, 情報量基準値が最小の要素を返す. p 値が 0.1 を超えない場合は, 情報量基準値が最小の要素を返す.
PREPARE_VAR($\mathcal{D}$)	これまで定義した関数群を用いて, VAR の推定に必要な処理をまとめて行い, 最適なラグと入力変数を探索, 構成する.
IRF($B_0, B, c, u, \mathcal{A}, k$)	B_0, B, c, u に基づき, $\mathcal{A}$ 内すべての変数の組み合わせ に関する k 期の OIRF を計算. 出力は $n \times n \times k$ の 3 次テンソル

図 4.1: アクティビティ時系列データの例: 日次のコミット推移



この分析期間を $T_m(m = 1, 2, 3, \cdots)$ と表す. 例えば, 2016 年元旦から 2017 年末までの 2 年間を 2016 年の 1 年間の区間と, 2016 年から 2017 年の 2 年間の区間の日次データとして区切って分析する場合, $T_m = [T_1, T_2]$ となり, T_1, T_2 はそれぞれ次のようになる.

$$T_1 = \{2016/01/01, 2016/01/02 \cdots, 2016/12/31\}$$
$$T_2 = \{2016/01/01, 2016/01/02 \cdots, 2017/12/31\}$$

T_m の区間数やそれぞれの長さなどに関する詳細な条件は MIAO では定めていないため, 任意のものを構成してよい. この T_m が MIAO への入力変数の一つとなる.

もうひとつの入力変数がアクティビティ時系列データである. ある時系列がアクティビティ時系列データとなるための条件は, 該当する時系列データが OSS の生存性を表し, 複数の OSS で共通して比較可能であることである. したがって, この条件を満たすものであればどんな時系列であっても構わない. ただし, MIAO に入力する全てのアクティビティ時系列データは, 同時に入力する T_m の最小時刻から最大時刻を含んでいる必要がある. ある OSS から得られるアクティビティ時系列データを A_i と表記する. 例えば, 3 つの OSS のアクティビティ時系列データを集める場合, $\mathcal{A} = [A_1, A_2, A_3]$ のように書く. 図 4.1 はアクティビティ時系列データの一例である.

これらの前提のもと, MIAO のフェーズ 1 におけるアルゴリズムの詳細を Algorithm MIAO Phase1 に示す. また, アルゴリズム内で利用する関数の定義を表 4.1 に示す.

次にアルゴリズムの概説を行う. まずは, PREPARE_VAR 関数の解説を行う. PREPARE_VAR は, IRF の計算過程 4 までを行うものであり, 最適なラグ次数 lag と, 諸々の変換工程を終えた VAR に入力するための時系列 $\mathcal{D}'$ を返す関数である. ただし, MIAO ではすべての時系列が定常であることを要求するため, 計算過程 3 の共和分の有無の確認は行わない. 行 6 から行 13 は, 必要であれば時系列 A_{T_m} に季節調整や定常過程への変換等の前処理を加え, 変換後の A_{T_m} を $\mathcal{D}'$ に格納する処理を行う. 定常過程への変換は, 行 11 の ADF 検定で時系列 A_{T_m} が単位根を持つかを判定することで始まる. 本アルゴリズムにおける ADF 検定では, トレンド定常を許容する. トレンド定常とは, その時系列からトレンドを取り除くことで, 定常過程となるものである. 確率過程において, トレンド定常は, 定常過程に分類される. 検定のた

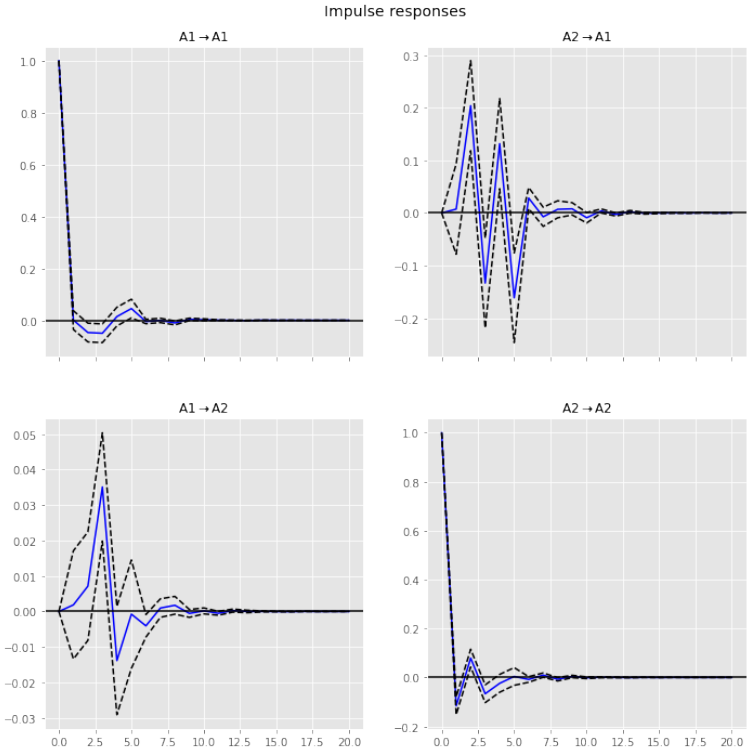
めの有意水準は5%に設定し, ADF 検定の結果から得られた p 値が 0.05 を超える場合, 時系列 A_{T_m} は単位根過程であると判断する. その場合, 行 10 において, 分数差分変換を適用し, A_{T_m} を定常過程に変換する. このとき, 分数差分 n は A_{T_m} が定常となりうる n のうちで最小のものとする. それ以外の時系列はレベルのまま用いる.

PREPARE_VAR 関数の行 6 以降は, VAR のための最適なラグ次数を推定する処理となる. 変数 $histories$ は, (ラグに対応する情報量基準値, ラグ次数, Ljung-Box 検定の p 値) のタプルを配列形式で保持する変数である. 行 15 では設定した ic に基づき, max_lag までのラグ次数に対応した情報量基準値の列を返す. このとき, 要素の並び順はラグ次数が小さいものから昇順となる. 行 16 から行 20 は各ラグ毎に SVAR の推定を行い, 撹乱項に対する Ljung-Box 検定を行うことで, 検定結果を $histories$ に格納する処理である. 最終的に, 行 21 の VAR_BEST_LAG 関数により, 情報量基準と撹乱項の系列相関を鑑みた最適なラグ次数を選択する. ここで選ばれたラグ次数と, 以前に作成した $\mathcal{D}'$ を返すことで PREPARE_VAR 関数は処理を終える.

次に, MIAO_P1 関数の解説を行う. MIAO_P1 は IRF の計算過程全てを担い, IRF の計算結果を返す関数である. 行 24 から 29 は入力されたアクティビティ時系列データ A の T_m 期間に該当する部分列を抽出し, $\mathcal{D}$ に格納していく処理である. $\mathcal{D}$ が完成したら, 前述の PREPARE_VAR により最適なラグ次数と諸々の変換を行った $\mathcal{D}'$ を得る. これらを用いて行 31 で SVAR のフィッティングを行い, 式 (3.4) で示したような推定結果のパラメータを得る. 推定結果の評価はアルゴリズムには含まれていないが, 満足いかない推定結果となれば, 再度パラメータ等を調整してアルゴリズムを最初から実行する. 最終的に, これらの SVAR_FIT で得られるパラメータを入力として, IRF を計算する. IRF 関数内では SVAR の構造形から式 (3.5) の VMA 形式に変換する一連のプロセスを実行し, 式 (3.12) によりインパルス応答を得る. インパルス応答は, A_i に対して, $A_i \to A_j$ のような単位で出力される. これは, A_i にショックを与えた時の A_j の反応を表しており, その反応値が k 個ある. 例えば, $|\mathcal{A}| = 3, k = 10$ である場合は, $(A_1 \to A_1, A_1 \to A_2, A_1 \to A_3), (A_2 \to A_1, A_2 \to A_2, A_2 \to A_3), (A_3 \to A_1, A_3 \to A_2, A_3 \to A_3)$ の計 9 つの組み合わせがあり, それぞれの $A_i \to A_j$ に対して 10 期分のインパルス応答が計算されることになる. これに加えて, 各 T_m 事に同様の処理が行われるので, $\mathcal{O}$ の形状が $m \times n \times n \times k$ の 4 次テンソルとなっている. 行 34 から行 37 が T_m 毎に MIAO_P1 を実行し, 出力を $\mathcal{O}$ に格納する処理である.

図 4.2 は MIAO Phase1 の出力である IRF を図示したものである. 図 4.2 では, $\mathcal{A} = [A_1, A_2], k = 10$ のときのインパルス応答を表す. 例えば, $A_1 \to A_1$ は A_1 から A_1 へのインパルス応答で, A_1 に 1 単位のショックを与えたときの A_1 の反応を表す. このときのインパルス応答は, $k = 0$ でピークを打ち, 以降で減衰して行くことが読み取れる. 他方で, $A_2 \to A_1, A_1 \to A_2$ では数期にわたりインパルス応答は振動している. それぞれの正方向のピークの値を見てみると, $A_2 \to A_1$ のピークは 0.2 付近であるのに対し, $A_1 \to A_2$ では 0.03 付近であることから, A_2 から A_1 に対する影響が A_2 から A_1 に対する影響よりも大きくなることが読み取れる. このように, インパルス応答では概形だけでなく, 影響の大きさに着目することが重要となる. なお, インパルス応答が 1.0 という値である場合は, 元のショックと同等の値であることを表し, 0.6 である場合は, 元のショックの 0.6 倍の反応であることを表す.

図 4.2: 2×2 のインパルス応答例



4.2 MIAO フェーズ 2

OSS の時系列データには成長モデルがあり, 構造変化が起こりやすいことは前述したとおりである. そのため, MIAO では短期的なインパルス応答の値は利用せずに, 式 (3.13) の累積インパルス応答と呼ばれる, 影響の累積的な影響度を表す値を用いる. 本研究では, k を無限にした時の累積インパルス応答の値を

$$\mathrm{SCE}_{ij} = \sum_{k=1}^{\infty} \mathrm{IRF}_{ij}(k) \tag{4.1}$$

と定義する. SCE は, Shock Cumulative Effect の略である. SCE の収束性は, インパルス応答の収束性を議論すれば良いことになる. インパルス応答の収束には, 定常な VAR が VMA 表現をもつこと, その VMA 表現の係数は時間とともに減衰することを示せば良い. まず, 定常な VAR が VMA 表現を持つことは, ウォルドの分解定理 [41](Chapter 4.8) により示される. 次に, VMA 表現の係数が時間とともに減衰することを示す. 簡単のため, VAR の次数を 1 にした, AR モデルを用いる. 最も単純な AR(1) モデルは $y_t = \phi y_{t-1} + \epsilon_t$ で表される. このとき, 単位ショックを $\epsilon_t = 1, t = 1$ として, ϵ_t で y_t を微分すると, ϵ_t の係数部のみが残り $y_1 = 1$ となる. これを一般の t に拡張していくと,

$$
\begin{aligned}
y_2 &= \Phi y_1 + 1 = \Phi \cdot 1 = \Phi \\
y_3 &= \Phi y_2 + 1 = \Phi^2 \\
y_4 &= \Phi y_3 + 1 = \Phi^3 \\
&\vdots \\
y_h &= \Phi^{h-1} \\
&\vdots
\end{aligned}
$$

となる. このとき, MA 表現の係数 Φ^h はインパルス応答である. AR モデルが定常である条件は, $|\Phi| < 1$ となることである [41](Formula [15.4.2]). この条件から, t が進む毎に, Φ^h の値は徐々に小さくなっていくので, $h \to \infty$ すると, Φ^h は無限遠で 0 に収束するのである. これを多変量に拡張したものが VAR となるので, この収束性は VAR でも同様に保証されることになる. ただし, 計算機では $h \to \infty$ とできないので, h をどこで止めるか, すなわち, 式 (3.7) の k をどこで止めて, SCE を近似すればよいのかという議論になる. 詳細な実験を行い, どの程度の k であれば計算効率を落とさずに精度を担保できるかを調べたところ, $k = 200$ 程度あれば, 計算効率と精度のバランスが取れることが分かった. 実験内容や実験結果は付録に掲載した.

SCE は T_m 毎に計算されるため, $A_i \to A_j$ 毎に全部で m 個の SCE が得られる. m 個目の SCE_{ij} を SCE_{ij}^m と表記し, MIAO スコアを次式で定義する.

$$
\mathrm{MS}_{ij} = (-1) \cdot \sum_{k=1}^{m} \mathrm{SCE}_{i,j}^k \tag{4.2}
$$

-1 を乗じて符号を反転しているのは, MIAO スコアは負の影響が与えられるほど大きくなるように構成するためである. つまり, T_m 毎に持続的に負の影響が与えられている場合, MS_{ij} は徐々に大きくなる. MS_{ij} が 0 に近いときは, 正負の影響が振動しており, 長期的には影響が打ち消されていることになる.

MIAO フェーズ 2 で求められる MS_{ij} の個数は, $n = |\mathcal{A}|$ としたときに, $n \times n$ である. ただし, 自身から自身への $\mathrm{MS}_{ij}(i = j)$ は不要であるので, $n \times n - n$ 個の MS_{ij} が最終的に得られ, 表 4.2 で示す, MIAO スコア表として出力される.

表 4.3 は, 右側の条件が成立した際の, MS_{ij} の意味についてまとめたものである. 表中の D は $\mathrm{D}(a, b) = |a - b|$ なる距離関数である.

表 4.3 だけでは, スコア表を読むための情報が不足しているので, いくつかの補足を行う. そのための準備として, スコア表に競合関係を導入する. 具体的には, $T = A_i$ を分析対象, $C_i = A_i (A_i \neq T)$ を T に対する i 番目の競合とし, MS_{C_iT} と書くとき, $C_i \to T$ の MIAO スコアを表すものとする.

まず, 真っ先に REV の発生を疑うケースは, $T = A_i, C_i = A_j$ のときに, M1 が成立し $\mathrm{D}(\mathrm{MS}_{C_iT}, \mathrm{MS}_{TC_i})$ が表中最大かつ, 相対的に $\mathrm{D}(\mathrm{MS}_{C_iC_j}, \mathrm{MS}_{C_jC_i})$ が小さくなるときである. 相互的に影響がないケースは, M1 から M4 のいずれにおいても, $\mathrm{D}(\mathrm{MS}_{ij}, \mathrm{MS}_{ji}) \fallingdotseq 0$ となることである. 一方で, M1 が成立しない M3, M4 のケースで

表 4.2: 一般的な MIAO スコア表

Direction of Influence	MIAO Score
$A_1 \to A_2$	MS_{12}
$\vdots$	$\vdots$
$A_1 \to A_n$	MS_{1n}
$A_2 \to A_1$	MS_{21}
$\vdots$	$\vdots$
$A_n \to A_{n-1}$	$\mathrm{MS}_{n,n-1}$

表 4.3: MIAO スコア表の意味

ID	意味	条件
M1	A_i は A_j に対して負の影響を与えており, $\mathrm{D}(\mathrm{MS}_{ij}, \mathrm{MS}_{ji})$ が影響の大きさを表す.	$\mathrm{MS}_{ij} > 0 \wedge \mathrm{MS}_{ji} \leq 0$
M2	A_j は A_i に対して負の影響を与えており, $\mathrm{D}(\mathrm{MS}_{ij}, \mathrm{MS}_{ji})$ が影響の大きさを表す.	$\mathrm{MS}_{ij} \leq 0 \wedge \mathrm{MS}_{ji} > 0$
M3	A_i, A_j は相互的に正の影響を与えており, $\mathrm{D}(\mathrm{MS}_{ij}, \mathrm{MS}_{ji})$ が影響の大きさを表す.	$\mathrm{MS}_{ij} \geq 0 \wedge \mathrm{MS}_{ji} \geq 0$
M4	A_i, A_j は相互的に負の影響を与えており, $\mathrm{D}(\mathrm{MS}_{ij}, \mathrm{MS}_{ji})$ が影響の大きさを表す.	$\mathrm{MS}_{ij} \leq 0 \wedge \mathrm{MS}_{ji} \leq 0$

あっても, $\mathrm{D}(\mathrm{MS}_{ij}, \mathrm{MS}_{ji})$ が表中で大きい場合, $\mathrm{MS}_{ij} \gg \mathrm{MS}_{ji}$ あるいは $\mathrm{MS}_{ji} \gg \mathrm{MS}_{ij}$ となっていることから, REV を疑うことがある. これは, T, C_i 間のみならず, C_i, C_j 間で REV が発生している可能性を意味する. $T = A_i, C_i = A_j$ のときに, M2 が成立する場合は, 逆に T から C_i に負の影響を与えているときで, $\mathrm{MS}_{TC_i} \gg \mathrm{MS}_{C_iT}$ となる場合では, T が C_i を REV に追いやっている可能性がある.

4.3 実験

本節では, 実データを用いて, MIAO の有効性を確認するための実験を行う. 実験結果の評価は次節で行う.

4.3.1 データセット

本実験では, REV 群, nonREV 群の 2 群を対象にデータセットを構築した. 各群は 3 つの競合 OSS からなるグループが集まって構成される. 端的に言えば, REV 群は過去に人気のあったものと現在でも人気のあるものから構成されており, nonREV 群については, 現在でも人気のあるもの, あるいはメンテナンスされているものから構成されている. データセットに含まれる技術ドメインは, 深層学習, Web フレームワーク (言語問わず), プログラミング言語, Web フロントエンド向けの UI 構築ライブラリ, NoSQL データベース, RDBMS, コンテナオーケストレーション/マネージ

表 4.4: 実験データセット

group	target	competitor1	competitor2	rev	deprecated	archived_date	start_date	end_date	n_split
1	chainer/chainer	tensorflow/tensorflow	pytorch/pytorch	1	1	–	2016-09-25	2020-02-25	1
2	Theano/Theano	tensorflow/tensorflow	chainer/chainer	1	1	–	2015-11-07	2018-02-07	1
3	docker-archive/classicswarm	kubernetes/kubernetes	apache/mesos	1	0	2021-02-02	2014-06-06	2016-09-06	1
4	bcit-ci/CodeIgniter	symfony/symfony	laravel/framework	1	0	–	2013-01-10	2019-01-10	2
5	zendframework/zendframework	cakephp/cakephp	symfony/symfony	1	0	2020-01-09	2010-01-04	2015-01-04	2
6	atom/atom	microsoft/vscode	jetbrains/intellij-community	1	0	2023-03-04	2015-11-13	2021-11-13	2
7	fuel/fuel	symfony/symfony	laravel/framework	1	0	–	2013-01-10	2014-04-10	1
8	rethinkdb/rethinkdb	apache/couchdb	mongodb/mongo	1	0	–	2009-10-03	2016-10-03	2
9	gulpjs/gulp	webpack/webpack	jakejs/jake	1	0	–	2013-07-04	2015-01-04	1
10	Semantic-Org/Semantic-UI	twbs/bootstrap	foundation/foundation-sites	1	0	–	2013-09-08	2017-09-08	1
11	jashkenas/coffeescript	microsoft/TypeScript	dart-lang/sdk	1	0	–	2014-07-08	2015-10-08	1
12	angular/angular.js	facebook/react	emberjs/ember.js	1	0	2024-04-13	2013-05-29	2018-05-29	2
13	knockout/knockout	angular/angular.js	facebook/react	1	0	–	2013-05-29	2014-08-29	1
14	yui/yui3	jquery/jquery	dojo/dojo	1	0	–	2007-10-10	2013-10-10	2
15	apache/cordova-android	facebook/react-native	flutter/flutter	1	0	–	2015-01-30	2018-04-30	1
16	apache/lenya	WordPress/WordPress	drupal/drupal	1	0	–	2003-04-01	2008-04-01	2
17	dresende/node-orm2	sequelize/sequelize	typeorm/typeorm	1	0	–	2016-02-21	2017-05-21	1
18	google/volley	apache/httpcomponents-client	square/okhttp	1	0	–	2012-11-07	2015-04-07	1
19	karma-runner/karma	mochajs/mocha	jestjs/jest	1	1	–	2014-05-14	2019-05-14	2
20	basho/riak_kv	redis/redis	memcached/memcached	1	0	–	2010-07-22	2015-07-22	2
21	Carthage/Carthage	CocoaPods/CocoaPods	apple/swift-package-manager	1	0	–	2015-10-30	2019-02-28	1
22	less/less	stylus/stylus	sass/node-sass	1	0	–	2012-06-30	2016-06-30	1
23	requirejs/requirejs	browserify/browserify	webpack/webpack	1	0	–	2012-03-10	2013-06-10	1
24	jashkenas/backbone	facebook/react	angular/angular.js	0	0	–	2013-05-29	2016-05-29	1
25	rails/rails	django/django	laravel/framework	0	0	–	2013-01-10	2023-01-10	3
26	mysql/mysql-server	postgres/postgres	FirebirdSQL/firebird	0	0	–	2001-05-23	2023-05-23	6
27	ruby/ruby	python/cpython	nodejs/node	0	0	–	2009-02-16	2023-02-16	4
28	nginx/nginx	apache/httpd	h2o/h2o	0	0	–	2014-09-04	2023-09-04	3
29	spring-projects/spring-boot	apache/struts	playframework/playframework	0	0	–	2012-10-21	2023-10-21	3
30	jquery/jquery	angular/angular.js	facebook/react	0	0	–	2013-05-29	2015-08-29	1
31	tidev/titanium-sdk	facebook/react-native	flutter/flutter	0	0	–	2015-01-30	2022-01-30	2
32	mongodb/mongo	apache/cassandra	apache/couchdb	0	0	–	2009-03-02	2023-03-02	4
33	symfony/symfony	bcit-ci/CodeIgniter	laravel/framework	0	0	–	2016-01-01	2023-01-01	2
34	kubernetes/kubernetes	apache/mesos	hashicorp/nomad	0	0	–	2015-06-01	2023-06-01	2
35	remix-run/remix	vercel/next.js	nuxt/nuxt	0	0	–	2020-07-09	2023-07-09	1

メント, Web サーバ, Android 向けのライブラリ, ビルドツール, テストフレームワーク, エディタ, CMS などと多岐にわたり, 多くのドメインをカバーする.

具体的なデータセットは, 表 4.4 で示すものである. 1 行が 1 グループに相当し, 各グループで REV が発生したかどうかのフラグ”rev”を持ち, rev=1 の場合に REV が発生したことを, rev=0 の場合に nonREV であることを表す. このうち, target は REV が発生したかを調べたい OSS を指定し, competitor1, competitor2 に競合 OSS2 つを指定する. 競合を 2 つ用意するのは, 公平性の観点のためである. これら 3 つの OSS からアクティビティ時系列データ $\mathcal{A} = [A_i]$ を取得する. 実験で利用するアクティビティ時系列データには日次のコミット数の推移を用い, コミット履歴は GitHub のリポジトリをクローンすることで取得する. コミット数の推移は多くの先行研究において OSS プロジェクトの生存性を表す指標として用いられており, 信頼性の高いデータである.

表中のその他の項目については, ”deprecated”は OSS の状態が deprecated であるかどうかの真偽値, ”archive_date”は GitHub 上でアーカイブされた日付である. ”start_date”, ”end_date”, ”n_split”は分岐期間に関する情報であるので, 4.3.2 にて詳細を述べる.

実験対象の選定にあたり課題となるのは, REV が発生したことを確定的に知ることが困難な点である. そのため, 一定のルールに従う OSS を REV 群に分類し, 手動でデータの収集を行った. 最も前提となるルールは, REV が発生した OSS は開発が停止, あるいはそれに近い状態にあることである. したがって, 次のルールの内いずれかを満たす必要がある.

1. Github 上で Archive[42] されている
2. GitHub 上で Deprecated であることが判別できる
3. リポジトリが休眠状態である

Github における Archive とは, リポジトリがリードオンリーになり, 編集できない状態である. OSS プロジェクトが Deprecated であるとは, これ以上当該プロジェクトのメンテナンスを行わないことを宣言していることである. この宣言は通常, Github リポジトリ上の説明文や README 等に明記されている. リポジトリが休眠状態であるとは, Archive でも Deprecated でもないが, 長期間にわたってプロジェクトの活動がない状態をいう. リポジトリの休眠状態とは, OSS の一旦の死を意味する. 休眠という言葉を用いるのは, OSS は一度死亡したように見えても, 再度復活する場合があるためである [43]. OSS の死の定義は明確に定まっておらず, 活動パターンをもとに生死を判定することが一般的である. Evangelopoulos ら, [43] は OSS の死を理解するプロセスの中で, 長期間活動がないことを OSS の死と定義した. 本研究では, 直近 12ヶ月間の平均コミット数が 1.5 以下となる場合に, リポジトリが休眠状態であると定義する. これは, [12] の休眠状態の定義である, 直近 12ヶ月間の平均コミット数が 1 以下となる場合に休眠とするものに, 多少の緩和を行ったものである.

一方で, 上記のルールを満たしていても, それが競合の台頭によるものでなければ REV 発生とは呼べない. そのため, 上記のルールに加え, "EX1: 競合の台頭により REV が発生したといえるデータソースがある", あるいは"EX2: 多くの開発者が REV により開発が停止されたと認識できる状態"であると考えられる場合に REV であると結論づける. EX1 の例は, Chainer の開発元が発表した Pytorch への移行記事 [44] や, Theano のメーリングリストで発表された深層学習周辺のエコシステムの発展に伴い開発を停止するという内容 [45] である. EX2 の例としては, PHP の Web フレームワークである CodeIgniter[46] が Laravel[47] にシェアを奪われたことなどがある. 無論, EX2 に関しては筆者自身の主観もあるため, REV が発生したと言える可能性は高いが必ずしもそうとはいえないという妥当性の脅威は捨てきれない. なお, OSS プロジェクトは一定以上の活動量や認知度がないと競合の台頭以前に自然と破棄されてしまうため, 次の条件のうち, いずれか 2 つを満たすもののみを選定した. 1. GitHub スター数が 3,500 以上, 2. 全体のコードサイズが 50MB 以上, 3. コミット数が 2,000 以上, 4. 40 人以上のコード改変を伴う貢献者がいること, である. 一般的に 3500 以上の GitHub スターを取得することは容易ではない. Borges ら [29] によれば, より多くの GitHub スターを持つプロジェクトは企業がホストするような規模の大きいものであること, スター数とコントリビューター数, フォーク数には, それぞれ中程度の相関関係がみられた. つまり, GitHub スター数は規模, 人気, シェアの一種のバロメータとなる. また, Kalliamvakou ら [40] によれば, OSS プロジェクトあたりのコミット数は非常に偏っており, サンプル内の中央値はわずか 6 コミット, 90%は 50 コミット未満であることが示された. つまり, 2000 コミットというのはこの中央値を大幅に上回り健全にメンテナンスされていたものであるといえる. したがって, これらの要件を満たす場合, 小規模が故に作者のモチベーションなどで開発が停止してしまうなどの可能性を低下させることができる.

表 4.5: End's Period Auto Detection のアルゴリズム内で利用する関数の定義

関数名	説明
START_DATE($\mathcal{D}$)	$\mathcal{D}$ に属するアクティビティ時系列データ が共通して持つ最小の日付を返す.
TO_MONTHLY(A)	A を月次データに変換する.
TO_ANUAL_MEAN(A)	A を年毎にグルーピングし, 年ごとに月次のアクティビティの平均を計算し, 年とその平均のリストをタプル形式で返す.
TAIL(A, n)	A の後ろから n 個の要素を取り出す.
SUBTRACT_MONTH($date, n$)	$date$ の n ヶ月前の日付を返す.

nonREV 群に関しては, 競合関係はあるものの, 全ての候補が実験実施期（2024 年 5 月）に生存していることが条件である. REV 群同様に認知度や活動量が多くメジャーなものを選出している.

4.3.2 分析期間の決定に関して

データセット中の各グループに属する OSS は, それぞれが異なる時間軸で開発されており, 分析期間を一様に決定できるわけではない. したがって, グループ毎に個別に分析期間の決定を行う必要がある. 本項では, 実験において分析期間をどのように決定するかについて述べる.

各グループ毎の分析期間の開始時刻は, グループに属するアクティビティ時系列データ全てが共通して持つ日次の内最小のもとする. 一方, 終了時刻に関しては, 時系列を解析し, アルゴリズムにより機械的に決定する. 終了時刻を決定するアルゴリズムを Algorithm End's Period Auto Detection に示す. アルゴリズムを簡潔にいえば, "アクティビティ時系列データの年毎の平均が, DT（休眠と判定する閾値）を下回る年を抽出し, さらにその年の月次のアクティビティ時系列データの推移の内で DT を下回る月を終了時刻とする"である. これに該当する時刻がなければ, 全てのアクティビティ時系列データが共通して持つ最後の時刻を終了時刻とする.

次にグループ毎の具体的な $T_1, T_2, \cdots, T_m$ をどのように定めるかについて述べる. 本実験では, $m = 1$ を起点として m が増える毎に 1 年分 T_m を構成する要素が増えるように T_m を作る. つまり, どの時点の m においても, $|T_m| \leq |T_{m+1}|$ の関係となる. 例えば, End's Period Auto Detection アルゴリズムにより得られた開始時刻が 2017 年 1 月 1 日で終了時刻が 2020 年 12 月 31 日までとするならば, T_m は式 (4.3) のように構成されることになる.

$$T_1 = \{2017/01/01, \cdots, 2017/12/31\},\ T_2 = \{2017/01/01, \cdots, 2018/12/31\} \quad (4.3)$$
$$T_3 = \{2017/01/01, \cdots, 2019/12/31\},\ T_4 = \{2017/01/01, \cdots, 2020/12/31\}$$

次にグループ毎の具体的な $T_1, T_2, \cdots, T_m$ をどのように定めるかについて述べる. 本実験では, $m = 1$ を起点として m が増える毎に 1 年分 T_m を構成する要素が増え

Algorithm End's Period Auto Detection

Input: $\mathcal{D}'$:= Algorithm MIAO Phase1 で PREPARE_VAR が返すもの

DT = 10 ▷ DormantThreadhold の略で, 休眠と判定する閾値
$start\_year, start\_month, start\_day \leftarrow$ START_DATE($\mathcal{D}'$)
for each $A_{T_m} \in \mathcal{D}'$ **do**
    $n\_commits\_per\_month \leftarrow$ TO_MONTHLY(A_{T_m})
    $years, anual\_means \leftarrow$ TO_ANUAL_MEAN($n_commits_per_month$)
    ASSERT_EQUAL($|anual\_means|, |years|$)
    $end\_year \leftarrow years[-1]$
    for $i = 0$ to $anual\_means$ **do**
        $mean \leftarrow anual\_means[i]$
        if $mean \leq$ DT **then**
            $end\_year \leftarrow years[i]$
            break
        end if
    end for
    if $start\_year \geq end\_year$ **then**
        $end\_year = start\_year + 1$
    end if
    if $(end\_year - start\_year) == 1$ **then**
        $end\_year = end\_year + 1$
    end if
    $end\_month, end\_day \leftarrow start\_month, start\_day$
    $end\_date \leftarrow (end\_year, end\_month, end\_day)$
    $n\_commits\_per\_month' \leftarrow n\_commits\_per\_month$ の end_date 以降の要素のみをスライス
    $take \leftarrow 10$
    $tail\_commits =$ TAIL($n\_commits\_per\_month', take$)
    for $i = 0$ to $tail\_commits$ **do**
        if $tail\_commits[i] \leq$ DT **then**
            break
        end if
    end for
    $subtract\_month = take - (i + 1)$
    $end\_date \leftarrow$ SUBTRACT_MONTH($end\_date, subtract\_month$)
end for

るように T_m を作る. つまり, どの時点の m においても, $|T_m| \leq |T_{m+1}|$ の関係となる. 例えば, 開始時刻が 2017 年 1 月 1 日で終了時刻が 2020 年 12 月 31 日までとするならば, T_m は, 次のように構成される.

$$T_1 = \{2017/01/01, \cdots, 2017/12/31\},$$

$$T_2 = \{2017/01/01, \cdots, 2018/12/31\},$$

$$T_3 = \{2017/01/01, \cdots, 2019/12/31\},$$

$$T_4 = \{2017/01/01, \cdots, 2020/12/31\}$$

なお, 分析期間 T_m は長くなるほど時系列がトレンドを持つ可能性が高くなる. 特に, $m > 4$ になると VAR の撹乱項に系列相関が出やすくなるため, 分析期間 T_m は最長でも 4 年とした. したがって, 分析期間がそれ以上長い場合は 4 年毎に分割して, それぞれを異なるグループとして扱う. その分割数を記述したものが, 表 4.4 中の n_split である. グループ 22 でいえば, n_split は 6 なので, 2000-07-31 から 2023-07-31 までを 6 分割し, それぞれが異なる 6 グループとして展開される. この分割を考慮しない状態では, データセット内のグループ件数は 29 件で, そのうち, REV 群が 19 件, nonREV 群が 11 件で 2 群間の標本数に不均衡が生じている. しかし, 分割を行うことでグループ件数は 59 件となり, REV 群が 30 件, nonREV 群が 29 件と標本数が拡張され不均衡が解消されるようになる.

4.3.3 時間シフト

インパルス応答は時系列の変動に対してセンシティブであり, 分析期間を多少前後させただけでも結果が大きく変わることがある. そのため, 実験結果の信頼性, 堅牢性を高める施策として, 期間の異なる MIAO モデルを複数個構成し, 各 MIAO モデルから得られるスコアの平均値を最終的なスコアとして利用する. これは, アンサンブル学習におけるバギングに近い概念である. 具体的には, 入力された $T_1, T_2, \cdots, T_m$ に関して, それぞれの開始時間を τ_n 日シフトして新たな分析期間を構成し, それらの期間をもって MIAO を複数回実行するというものである. シフトする時刻を $\tau = [\tau_1, \tau_2, \cdots, \tau_n]$ とし, 式 (4.2) で定義した MS_{ij} に関して, その開始時刻を τ_n 日シフトしたものを $\mathrm{MS}_{ij\to\tau_n}$ と表すと, 複数の MIAO モデルを統合して得られるスコアは次のようになる.

$$\mathrm{AMS}_{ij} = \frac{\sum_{k=1}^{|\tau|} \mathrm{MS}_{ij\to\tau_k}}{|\tau|} \tag{4.4}$$

次に, 開始時刻をシフトした際に T_m がどのように変化するかの例を示す. まず, 元の T_m が以下のように与えられているとする.

$$T_1 = \{2017/01/01, \cdots, 2017/12/31\}$$
$$T_2 = \{2017/01/01, \cdots, 2018/12/31\}$$
$$T_3 = \{2017/01/01, \cdots, 2019/12/31\}$$

これらに関して, 開始時間を1ヶ月, 2ヶ月シフトした期間をそれぞれ$T_{m\to 1M}, T_{m\to 2M}$のように表すならば次のようになる.

$$T_{1\to 1M} = \{2017/02/01, \cdots, 2018/01/31\},\ T_{1\to 2M} = \{2017/03/01, \cdots, 2018/02/28\}$$
$$T_{2\to 1M} = \{2017/02/01, \cdots, 2019/01/31\},\ T_{2\to 2M} = \{2017/03/01, \cdots, 2019/02/28\}$$
$$T_{3\to 1M} = \{2017/02/01, \cdots, 2020/01/31\},\ T_{3\to 2M} = \{2017/03/01, \cdots, 2020/02/29\}$$

同様の表記を用いて, MIAO の入力を構成すると次のようになる.

$$\begin{aligned}\mathcal{T}_{\to 0M} &= \mathcal{T} = [T_1, T_2, T_3] \\ \mathcal{T}_{\to 1M} &= [T_{1\to 1M}, T_{2\to 1M}, T_{3\to 1M}] \\ \mathcal{T}_{\to 2M} &= [T_{1\to 2M}, T_{2\to 2M}, T_{3\to 2M}]\end{aligned}$$

したがって, この例における MIAO スコアは次式で得られることになる.

$$\mathrm{AMS}_{ij} = \frac{\mathrm{MS}_{ij} + \mathrm{MS}_{ij\to 1M} + \mathrm{MS}_{ij\to 2M}}{3}$$

4.3.4 実験の設定

実験に関する設定は表 4.6 で示すものである. 最大T_m数である 4 は VAR の撹乱項が相関系列を持ちづらくなるための値である. 設定値にもあるように, 終了年-開始年が 4 より大きい場合は, それぞれのmが 4 になるように分析期間を n 個に分割する. $\mathcal{T} = [T_1, \cdots, T_m]$ の表記を用いれば, 1 つ目の分割は$\mathcal{T}_1$, 2 つ目の分割は$\mathcal{T}_2$, 3 つめの分割は... のようになる. ただし, n 分割したあとの最後のT_nは$m = 4$に満たない場合があるので, $\mathcal{T}_{n-1}$に属するT_mを用いて, $\mathcal{T}_n$内のT_mが連続する時系列として$m = 4$になるように補完する. 最終的に分割で得られた$\mathcal{T}_n$が MIAO フェーズ 1 への入力となる.

期間のシフトは1ヶ月毎に最大3ヶ月まで行い, オリジナルとシフトされた3期間の合計 4 つの期間における MIAO スコアの平均を最終的なスコアとする.

本実験のアクティビティ時系列データである日次のコミット数の推移は, 多くの場合に周期 7 日の季節変動を持つ. これは, OSS に関しても商用ソフトウェア開発と同様, 週末にアクティビティが低下することを示唆する. 具体的な季節変動の調整には式 (3.1) の STL 分解を用いる. つまり, MIAO フェーズ 1 のアルゴリズム中の PREPROCESS 関数内では, STL 分解でA_{T_m}を季節成分, トレンド成分, 残差成分に分解し, 季節成分を取り除いたトレンド+残差の時系列を返す処理が行われることになる.

SVAR のラグ次数を決定する情報量基準は AIC とした. AIC を選択する意図は, 経済分野の多くの研究で AIC が利用されていることと, 本実験ではT_mに応じて大小様々な標本があり, 最も安定的にラグ次数を選択できると判断したためである.

表 4.6: 実験のための設定項目表

設定項目	設定値
T_m の決定方法	m が増える毎に 1 年分 T_m を構成する要素が増える
最大 T_m 数	4
期間シフト	オリジナル, 1ヶ月シフト, 2ヶ月シフト, 3ヶ月シフトの 4 期間
季節調整アルゴリズム	STL 分解
季節調整の周期	7 日
ADF 検定の有意水準	5%
SVAR のラグ次数を決定するための情報量基準	AIC
最大ラグ次数	15
最小ラグ次数	1
SVAR の識別問題に対する制約	リカーシブ制約（3 × 3 の再帰的構造 VAR）
SVAR に入力する変数のオーダー	competitor1, competitor2, target
Ljung-Box 検定のための最大ラグ次数	$\begin{cases} 10 & (\text{BEST\_LAG} < 10) \\ \text{BEST\_LAG} \times 2 & otherwise \end{cases}$
Ljung-Box 検定のための有意水準	10%
IRF の収束判定を行うための N	200

3 章で挙げた SVAR の識別問題の対処として, リカーシブ制約を用いる. ただし, リカーシブ制約では変数の並び順がインパルス応答に影響を与えることは前述の通りである. 実験の目的を達成するために重要なことは, 競合から対象への影響を定量化することであるため, 変数の並び順として最後に対象 OSS が来るように設定する. つまり, 最も外生性の低い OSS が対象 OSS ということになる. 競合の並び順については, 外生性が高いものを先に並べるという指針に基づき, 当時のタイムスケールにおいてより確立されているもの, あるいはデファクトスタンダードと呼べるものを先に並べる. これは GitHub プロジェクトのスター数やプロジェクト規模などを鑑みた定性的な評価により行われる. 例えば, Group 1 の深層学習フレームワークでは, 当時最もシェアが高かったのは Tensorflow である. 次いで新興のプロダクトとして PyTorch が登場した. そして, Chainer はこの PyTorch に影響を受けている. したがって, 変数の並び順を Tensorflow, PyTorch, Chainer のようにした.

nonREV クラスの並び順は REV クラスほど単純ではない. なぜなら, 現実世界では競合関係にあるが, それぞれの OSS には固定のユーザーベースがあり, いずれかの OSS がいずれかの OSS を REV に追いやるほどの非対称な関係性ではないからである. つまり, どれを target に選べばよいのかを REV クラス以上に考える必要がある. そのため本研究では, 先に登場したものがよりデファクトスタンダードであるという仮定を設け, 変数を並べることにした. この並び順は表 4.4 に反映されており, competitor1, competitor2 のように並ぶ. つまり, 最終的には competitor1, competitor2, target のように入力変数が並ぶことになる.

Ljung-Box 検定では一般に, VAR で指定するラグよりも大きなラグ次数まで検定することを推奨しており, 本実験でもこのセオリーに従う. そのため, VAR のラグ次数が 10 未満の場合は 10 を指定し, 10 である場合は 2 倍の 20 を最大ラグ次数とする.

なお, VAR および, SVAR の推定は Python の statsmodels[1] パッケージが提供する,

[1] https://pypi.org/project/statsmodels/

statsmodels.tsa.vector_ar.var_model.VAR と statsmodels.tsa.vector_ar.svar_model.SVAR を用いる. インパルス応答の計算については, statsmodels.tsa.vector_ar.svar_model.SVARResults.irf を用いる.

4.3.5 実験結果

表 4.7: ADF 検定および定常過程への変換に係る各種統計

	検定統計量	p 値	分数差分 n
count	2964.00	2964.00	2964.00
mean	-7.77	0.01	0.01
std	4.48	0.05	0.06
min	-34.58	0.00	0.00
10%	-14.17	0.00	0.00
25%	-9.35	0.00	0.00
50%	-6.52	0.00	0.00
75%	-4.80	0.00	0.00
90%	-3.64	0.00	0.00
max	0.31	0.98	0.70

表 4.8: モデルの推定結果の各種統計

	VAR 推定結果					Ljung-Box 検定結果		
	標本数	ラグ	AIC	BIC	HQIC	最大ラグ数	検定統計量	p 値
count	988.00	988.00	988.00	988.00	988.00	988.00	988.00	988.00
mean	871.96	6.84	9.83	10.23	9.98	13.06	58.38	0.42
std	404.51	3.22	3.53	3.53	3.53	5.58	34.46	0.24
min	365.00	1.00	0.14	0.91	0.54	10.00	9.63	0.00
10%	365.00	3.00	5.15	5.61	5.41	10.00	21.65	0.13
25%	366.00	4.00	7.91	8.25	8.03	10.00	28.84	0.22
50%	731.00	7.00	9.74	10.17	9.97	10.00	52.80	0.39
75%	1096.00	9.00	12.13	12.53	12.26	10.00	81.96	0.59
90%	1461.00	11.00	14.38	14.82	14.55	22.00	105.41	0.78
max	1461.00	15.00	18.72	18.96	18.81	30.00	209.18	1.00

本実験では, 期間分割, 期間シフトなどにより合計 2964 件のアクティビティ時系列データを用いて 988 個の SVAR モデルが推定された. 最終的なグループ数は REV 群が 32 件, nonREV 群が 34 件の計 66 件となった.

表 4.8 は MIAO フェーズ 1 の実行に際して決定されたパラメータや, 各種検定統計量の分布などをまとめた表で, 時系列の定常化に関するもの, VAR の推定に関する

もの, Ljung-Box 検定による系列相関に関するものの 3 つのカテゴリに分けられている. 例えば, VAR 推定結果のラグに着目すると, ラグは全てで 988 個の標本 (count) があり, そのうち最小値 (min) が 1, 平均値 (mean) が 6.39, 最大値 (max) が 6.81 であることを示している. n%が示すものは, そのパーセンタイルに位置する値で, 25%が 4.0 というのは, データの 25%が 4.0 以下で,75%が 4.0 以上であることを表す. 50%が中央値にあたる.

時系列の定常化に関するものは, ADF 検定結果および分数差分が該当する. ADF 検定は, 全ての A_i に対して行う必要があるので, count が 2964 件となっている. ADF 検定統計量は, 通常は負の値をとり, 値が負の方向に大きいほど, 単位根の存在を棄却する証拠が強くなる. p 値は検定統計量に対応する p 値を表している. 実験の有意水準は 5%より, データの内 90%以上は p 値が 0.0 以下で単位根の存在を棄却し, レベルのまま定常過程として扱えることを意味している. 残り数%の単位根過程には分数差分変換が適用され, 定常過程に変換される. その際の分数差分を表すのが, 表中の N-Frac diff である. 標本の内, 大多数が定常過程であるため $n = 0$ の割合が多いが, 残り数%の単位根過程は, 最大で $n = 0.7$ の分数差分変換が適用されている.

VAR の推定結果にある標本数は, アクティビティ時系列データ長を示し, $m = 4$ の設定から最小 365 日, 最大 1461 日である. ラグは MIAO フェーズ 1 の VAR_BEST_LAG 関数から得られる, 各 VAR にとって最適なラグ次数である. 最大ラグ次数が 15, 最小ラグ次数が 1 で, 平均的には約 6.84 のラグ次数が用いられているている. AIC, BIC(Bayesian information criterion), HQIC(Hannan–Quinn information criterion) は VAR_BEST_LAG 関数から得られるラグに対応する各情報量基準である. 本実験では, AIC が最も小さいものが優先的に選択される. ただし, 3 者の分布に極端に差異があるわけではないので, 今回選ばれたラグは安定したものであるといえる.

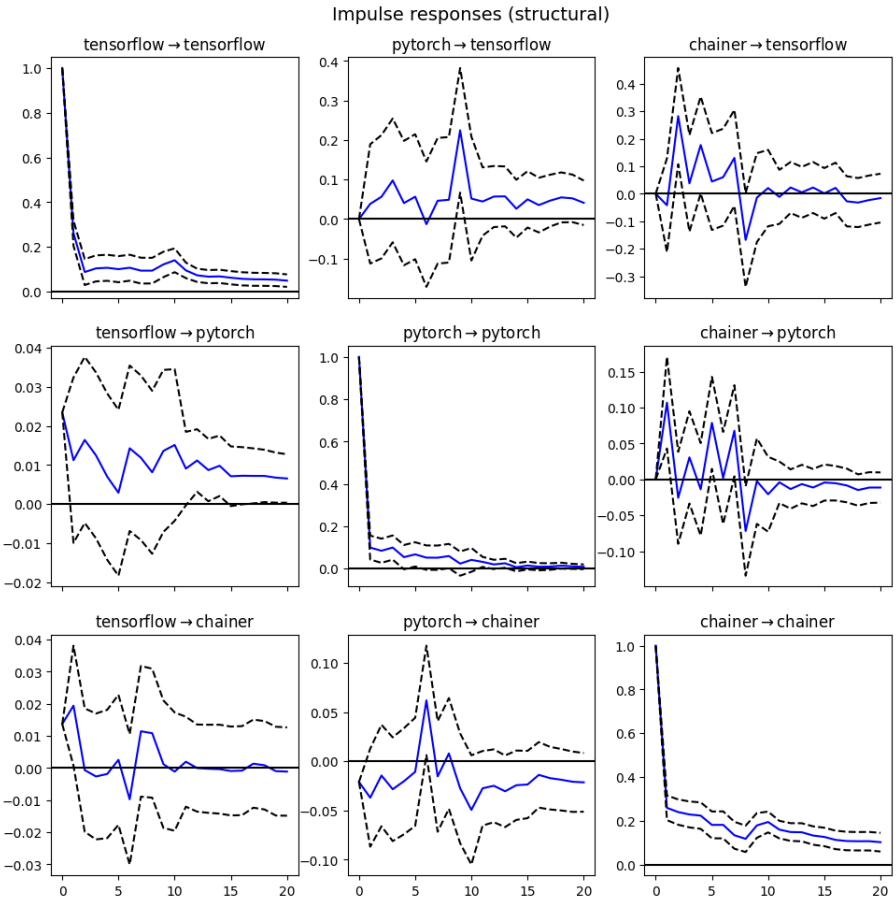
Ljung-Box 検定結果の最大ラグ次数は, 設定通り 10 か 30 いずれかで分布する. 検定統計量は大きいほど対立仮説を棄却する証拠が強くなる. 設定されている有意水準は 10%であるので, $p \geq 0.1$ となる場合に, VAR の誤差項に系列相関がないと判断できる. 表中では, 10%の時点で p 値が 0.13 と有意水準を超えているため, $90 + \alpha\%$ のデータでは VAR の推定精度の信頼性が高いことになる.

本実験で得られた全ての MIAO スコア表は, 付録中の表 A.2, A.3, A.4 に示した. Score は AMS_{ij} として出力されており, 4 つの MS_{ij} の平均である. Group は, データセット（表 4.4）中のグループに対応するもので, アンダースコア以降が分割 ID を表している. つまり, 1_0 である場合, グループ ID が 1 で分割 ID が 0, 1_1 の場合にグループ ID が 1 で分割 ID が 1 を表している.

4.3.6 実験で得られたインパルス応答の例

図 4.3, 4.4 は実験で得られた Group 1 におけるインパルス応答と累積インパルス応答を図示したものである. 前者のインパルス応答は, 3×3 における各 OSS 間のインパルス応答となっており, $k = 20$ までが出力されている. 対角要素は自身 A_i から自身 A_i へのインパルス応答で, それ以外が A_i に 1 単位のショックを与えたときの, 第 k 期における A_j の反応を示している.

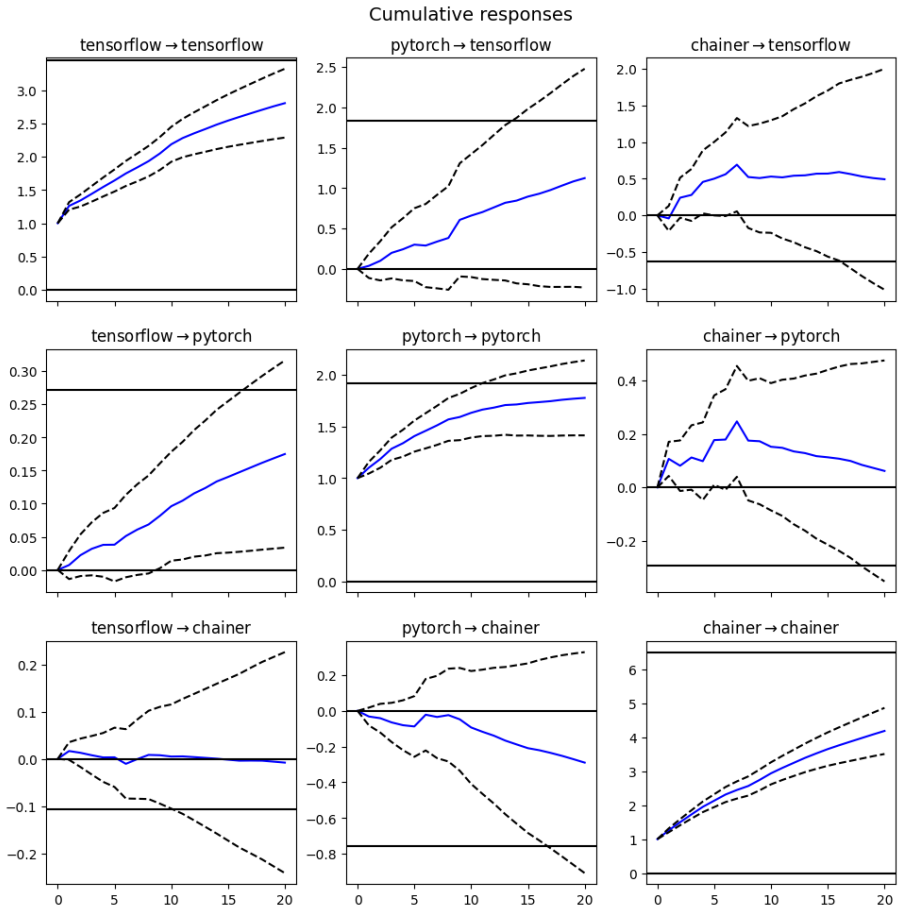
図 4.3: Group 1 におけるインパルス応答



例えば, chainer → pytorch では, $k = 1$ のときのインパルス応答が 0.11 でピークを打ち,$k < 8$ までは正方向で推移する. $k = 8$ でインパルス応答は一度マイナスに転じるが, $k > 8$ では再びプラスに転じ徐々に 0 に減衰していく. このことから, chainer のコミットが 1 単位上昇した場合, pytorch のコミットも 8 期まで上昇傾向であり, chainer のコミットが上昇した場合に, pytorch も同様にコミット数が上昇するといえる. 一方で, pytorch → chainer を見てみると, $k = 6, 8$ を除いて, すべてのインパルス応答はマイナスの値となっている. これは, pytorch のコミット数が上昇すると, chainer のコミットは下降傾向となることを表しており, chainer から pytorch と pytorch から chainer への影響にはが見られる.

このように, インパルス応答ではショックに対する各期の細かな変動を見ることができる. 一方で, 図 4.4 の累積インパルス応答では, より中長期的な影響を見ることができる. 先述した chainer → pytorch と pytorch → chiner の累積インパルス応答を見ると, chainer → pytorch では, $k = 20$ における累積インパルス応答はプラスの値である. 一方で, pytorch → chiner では, $k = 20$ の累積インパルス応答はマイナスの値となっている. これは影響が累積されたときに, その中長期的な影響がネガティブであるかポジティブであるかを確認する際に役立つものとなる. 図 4.4 では, $k = 20$ としているが, $k = \infty$ のときの値が, 式 4.1 で定義する SCE になる.

図 4.4: Group 1 における累積インパルス応答



4.3.7　実験で得られた MIAO スコア表の例

表 4.9 は, MIAO から得られたスコア表のうち特徴的な傾向を示すものである. GroupN_K として表現される場合, N はグループ番号を表し, K は 0 から始まる K 番目の区分を示す. グループ 1 と 8 は REV グループに属し, グループ 27 は nonREV グループに属する. グループ 1 の対象は chainer/chainer で, グループ 8 の対象は rethinkdb/rethinkdb である. これらのグループでは, $\mathrm{MS}C_2T > 0 \wedge \mathrm{MS}TC_2 < 0$ であり, $\mathrm{D}(\mathrm{MS}_{C_2T}, \mathrm{MS}_{TC_2})$ が大きな値を示していることから, C_2 から T への大きな負の影響が見られる. これは, 4.3 の M1 に該当しており, REV を疑うケースである. 一方, グループ 27 では, T, C_1, C_2 間の相互影響は概ね正であり, 何れかが上昇すると, いずれかも上昇するような関係が見られ, REV を引き起こすような明確な競合関係が見られないことが読み取れる.

4.4　評価

実験結果から, MIAO の有効性を判定するための評価を行う. MIAO の有効性の評価は, 決定木モデルによる REV の分類モデルを構築することで行う. 決定木モデル

表 4.9: 特徴的な MIAO スコア表の例

Direction	$\mathbf{AMS_{ij}}$
pytorch → tensorflow	-3.08
chainer → tensorflow	-3.58
tensorflow → pytorch	-0.12
chainer → pytorch	-0.99
tensorflow → chainer	-0.82
pytorch → chainer	2.11

Direction	$\mathbf{AMS_{ij}}$
mongo → couchdb	-1.92
rethinkdb → couchdb	-0.95
couchdb → mongo	-5.0
rethinkdb → mongo	-4.41
couchdb → rethinkdb	0.2
mongo → rethinkdb	5.12

Direction	$\mathbf{AMS_{ij}}$
cpython	-0.76
node → cpython	-1.02
cpython → ruby	-0.19
node → ruby	-1.26
cpython → node	0.03
ruby → node	-1.00

表 4.10: Classification Report

	Precision	Recall	F1-score	Support
nonREV	0.85	0.85	0.85	34
REV	0.78	0.91	0.84	32
Accuracy	–	–	0.85	66
Macro avg	0.85	0.85	0.85	66
Weighted avg	0.85	0.85	0.85	66

は, 閾値や重要な特徴量を視覚的に把握できるため, 比較的解釈性の高いモデルであり, 分類精度の検証以外に MIAO の解釈にも用いることができる. ここでは, MIAO スコアによる REV の分類性能の評価に焦点をあて, 解釈については 6 章で行う.

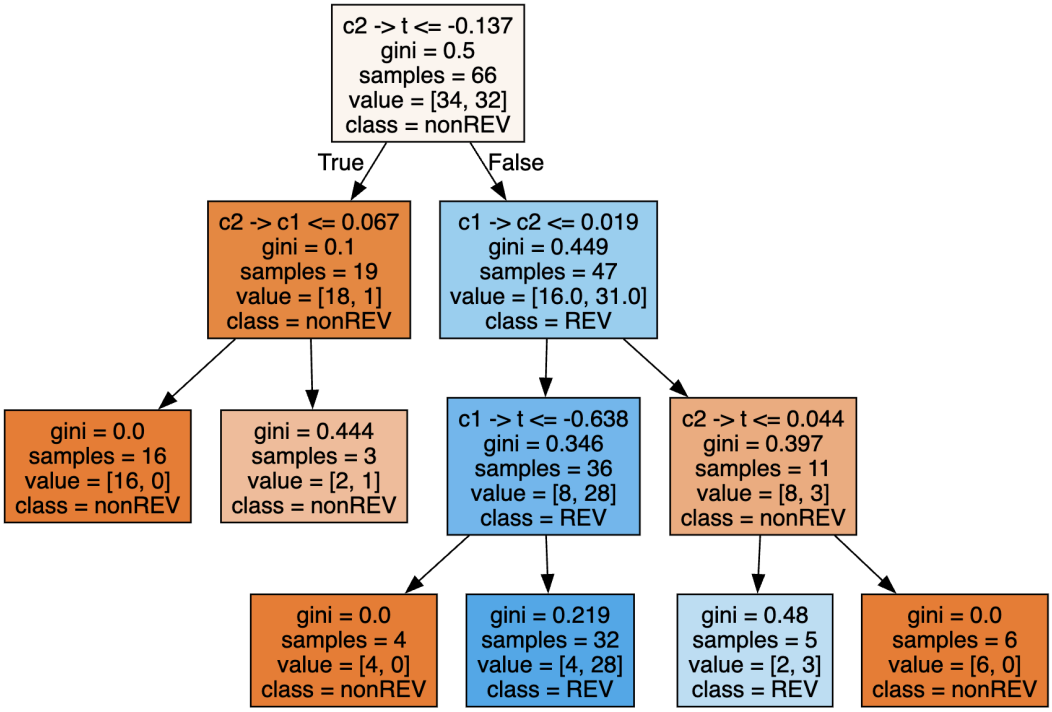
決定木モデルの目的変数は, 表 4.4 中の rev である. 説明変数は, 実験から得られた, 表 4.9 のような MIAO スコア表である. したがって, 59×6 個の MS_{ij} を用いて, 予測対象のグループが rev=1 か non-rev=0 かを予測するモデルということになる. ただし, スコア表毎に変数名が変わってしまうので, それぞれの変数名を $c1 \to t, c2 \to t, t \to c1, t \to c1, c1 \to c2, c2 \to c1$ のように統一する. t は target, $c1, c2$ はそれぞれ, competitor1, competitor2 である. 各グループは T_m の長さが異なるため, m が大きいほど, 大きな値, あるいは小さな値となる可能性が高まる. そのため, このバイアスを軽減するために, AMS_{ij}/m のように, AMS_{ij} を m で割って正規化した.

決定木モデルの学習および評価は, 汎化性能の向上のために, 交差検証を用いて行う. したがって, 本節で示す予測精度などの各種指標は, 各イテレーションで得られる指標の平均値である. ただし, 標本数が 66 件と十分ではないため, リーブワンアウ

表 4.11: 混同行列

		Predicted	
		Yes	No
Actual	Yes	27	5
	No	5	29

図 4.5: 決定木



ト交差検証 (LOOCV)[48] を用いる.

表 4.10 は, LOOCV によるモデルの平均的な性能を示すレポートで, 表 4.11 は, 各イテレーションにおける予測結果から得られた混同行列である. その他に, 図 4.5 は, LOOCV で用いたパラメータと同様のもので学習した決定木を可視化したものである. 決定木モデルの分類結果および真値との比較表は, 付録中の表 A.5 に示すものである. Actual が対照グループが REV であるか否かを示すもので, 1=REV, 0=nonREV である. Predicted が分類結果で, 1=REV, 0=nonREV である. なお, 決定木モデルには, sklearn.tree.DecisionTreeClassifier[49] を用いた.

結果を見ると, 決定木モデルの平均的な Accuracy は交差検証において 0.85 ± 0.35, 95%信頼区間は (0.76, 0.93) で, 一般的なモデルとしては中程度から良好な性能を示しているといえる. しかし, 図 4.5 に示す決定木は 4 層の深さを持ち, 過学習を避けながら適切な複雑さを保持し, 十分な予測力を持つ可能性を示唆している. 各クラスの分類においては, nonREV の F1 スコアが 0.85, REV が 0.84 であり, 両クラスに対して一貫した予測性能を示し, 安定した分類能力を持つことを示している. これらの結果から, 一般的な決定木モデルとしては中程度から良好な範囲にあると考え

られるが, 本研究が新しい手法であることを考慮すれば, 良好な結果とみなすことができる. 検証したグループ数は 64 と多くはないものの, ディープラーニングフレームワーク, プログラミング言語, Web アプリケーションフレームワーク, ビルドツール, データベースなど様々な領域を網羅しており, 特定のドメイン固有の結果というわけではない. したがって, MIAO は REV を判定するうえで有効な手法であるといえる.

第5章　REV予測モデル

MIAO は OSS 間の影響を競合関係として定量化するものであり, 将来の REV を予測する能力は持たない. REV 予測モデルは将来の REV 発生を予測するものである. 本章では, REV 予測モデルの理論と, 予測モデルを用いた実験および実験結果の評価についてまとめる.

5.1　概要

REV 予測モデルは, OSS グループ単位で将来の MIAO スコア表を予測するためのモデルである. ただし, MIAO は実際に観測された時系列に対して MIAO スコアを計算するようなアルゴリズムのため, 将来の MIAO スコアを得るには, グループを構成する各 OSS における将来のアクティビティ時系列データの変動が必要となる. ただし, 時系列の予測は一般的に難しいタスクであり, 周期性に乏しい REV による急激な時系列の変動を, 長期にわたって正確に予測することは現実的ではない. そこで, 将来の SCE を点推定するというアプローチが, REV 予測モデルの根本的なアイデアである.

このような前提の元, REV 予測モデル (RPM, REV Predictive Model) は, 次のコンポーネントからなる分類器である.

1. SCE 予測モデル
2. MIAO フェーズ 2
3. REV 判定器

1 の SCE 予測モデルが REV 予測モデルの根幹であり, 将来の SCE を予測する回帰モデルである. 2 の MIAO フェーズ 2 は, 4 章の MIAO で定義したもので, SCE を入力に MIAO スコアを出力する層である. そして, 3 の REV 判定器は, REV か nonREV かを分類する 2 値分類モデルで, これも 4 章で利用したような決定木モデルなどにより実装されるものである.

ここまでをまとめる, SCE 予測モデルを用いて将来の SCE を予測し, その SCE を MIAO フェーズ 2 に入力することで MIAO スコア表を出力し, MIAO スコア表から REV 発生を機械的に判定するというのが RPM の動作原理である.

5.2 REV 予測モデルの構成要素

5.2.1 SCE 予測モデル

MIAO の実行単位である 1 グループに属する A_i の, $T_{m\to\tau}$ 番目の n 次の説明変数ベクトルを

$$f_i^{T_{m\to\tau}} = \begin{pmatrix} f_{i,1}^{T_{m\to\tau}} \\ f_{i,2}^{T_{m\to\tau}} \\ \vdots \\ f_{i,n}^{T_{m\to\tau}} \end{pmatrix} \tag{5.1}$$

とする. SPM を $f_i^{T_{m\to\tau}}$ と j 番目の OSS を表す番号を引数に取り, $O_i \to O_j$ に関する SCE_{ij} を予測する関数とする. すなわち,

$$\hat{\mathrm{SCE}}_{ij}^{T_{m\to\tau}} = \mathrm{SPM}(f_i^{T_{m\to\tau}}, j) \tag{5.2}$$

と定義する. SPM は SCE Predicitive Model の略である.

SPM の結果を用いると, 将来の MIAO スコア $\hat{\mathrm{MS}}_{ij}$ は, $\hat{\mathrm{SCE}}_{ij}^{T_{m\to\tau}}$ と MIAO_P2 を使って次のように表せる.

$$\begin{aligned} \hat{\mathrm{MS}}_{ij} &= \mathrm{MIAO\_P2}(\hat{\mathrm{SCE}}_{ij}) \\ \text{where} \quad \hat{\mathrm{SCE}}_{ij} &= [\hat{\mathrm{SCE}}_{ij}^{T_{1\to\tau}}, \hat{\mathrm{SCE}}_{ij}^{T_{2\to\tau}}, \cdots, \hat{\mathrm{SCE}}_{ij}^{T_{m\to\tau}}] \end{aligned} \tag{5.3}$$

SPM で予測できる SCE は T_{m+1} の時の値である. したがって, $\hat{\mathrm{SCE}}_{ij}^{T_m} = \mathrm{SCE}_{ij}^{T_{m+1}}$ なる等式が成り立つ. SPM の実態は実数値である SCE を予測するための回帰モデルである.

目的変数と説明変数

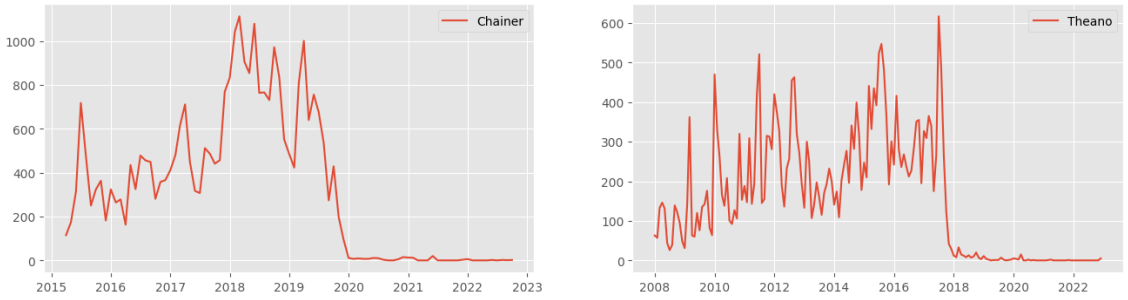
MIAO の実験において, データセットを構成する 1 グループの OSS の数は 3 つであった. したがって, 目的変数と説明変数については一般の OSS 数 n ではなく, $n = 3$ として説明を行う.

ここで, A_1, A_2, A_3 なる 3 つの OSS が与えられているとする. このとき, スコア表を構成するために SCE を予測したい OSS の組み合わせの数は, $(A_1 \to A_2, A_2 \to A_3), (A_2 \to A_1, A_2 \to A_3), (A_3 \to A_1, A_3 \to A_2)$ の計 6 つとなる. このとき, ある A_i を $A_i = T$ で固定し, 残りの A_j, A_k を $A_j = C_1, A_k = C_2$ としたときに C_1, C_2 から T に対する SCE は, $\mathrm{SCE}_{C_1,T}, \mathrm{SCE}_{C_2,T}$ の 2 つとなる. したがって, A_1, A_2, A_3 それぞれを T として固定したとき, 計 6 つの $\mathrm{SCE}_{C_1,T}, \mathrm{SCE}_{C_2,T}$ が得られ, これらをすべての T_m に関する SCE を計算すれば, スコア表を構成するための 6 つの MS_{ij} が得られるようになる.

表 5.1: SPM の説明変数 $f_i^{T_m \to \tau}$ の構成要素

変数名	説明
c1_to_t	$\mathrm{SCE}_{\mathrm{C_1,T}}^{T_m \to \tau}$
c2_to_t	$\mathrm{SCE}_{\mathrm{C_2,T}}^{T_m \to \tau}$
t_to_c1	$\mathrm{SCE}_{\mathrm{T,C_1}}^{T_m \to \tau}$
t_to_c2	$\mathrm{SCE}_{\mathrm{T,C_2}}^{T_m \to \tau}$
c1_to_c2	$\mathrm{SCE}_{\mathrm{C_1,C_2}}^{T_m \to \tau}$
c2_to_c1	$\mathrm{SCE}_{\mathrm{C_2,C_1}}^{T_m \to \tau}$
n_authors_cr	T_1 から T_m までのユニークな貢献者数の増加率
m	T_m における $m = (1, 2, \cdots)$

図 5.1: REV による急激なコミット数の減少例



つまり, 各 T_m に関して, T を固定したときの $\mathrm{SCE}_{C_1,T}, \mathrm{SCE}_{C_2,T}$ の 2 つが目的変数となれば, すべての i, j の組み合わせに関する $\hat{\mathrm{SCE}}_{C_1,T}^{T_m}$, $\hat{\mathrm{SCE}}_{C_2,T}^{T_m}$ を予測することができる. このことから, SPM は, 当期の $SCE_{C_1,T}^{T_m}, SCE_{C_2,T}^{T_m}, SCE_{T,C_1}^{T_m}, SCE_{T,C_2}^{T_m}, SCE_{C_1,C_2}^{T_m}, SCE_{C_2,C_1}^{T_m}$ を説明変数とし, 次期の $SCE_{C1,T}^{T_m+1}, SCE_{C2,T}^{T_m+1}$ を目的変数とする回帰モデルといえる. ただし, 回帰モデルの実装という観点では, 説明変数に対する予測値は一つの実数値となるため, $\hat{\mathrm{SCE}}_{C_1,T}^{T_m}$, $\hat{\mathrm{SCE}}_{C_2,T}^{T_m}$ それぞれを予測するための 2 つの SCE 予測モデルが必要となる. 以降, $\hat{\mathrm{SCE}}_{C_1,T}^{T_m}$ を予測するものを SPM1, $\hat{\mathrm{SCE}}_{C_2,T}^{T_m}$ を予測するものを SPM2 と命名する.

なお, 過去の SCE の推移だけでは将来の SCE を予測するための情報が不足している. なぜなら, SCE には時間情報が含まれていないため, 回帰モデルが SCE の時間的な推移を考慮できないためである. この時間的な情報を補完するための変数として, T_m における $m = 1, 2, \cdots$ を説明変数に加える.

将来の REV 予測に関してさらに考慮しなければならないことは, REV の発生は急激に起こる可能性があるという点である. 図 5.1 は, REV が発生した OSS である Chainer, Theano の月次のコミット数の推移である. 本図から, Chainer は 2019 年後半, Theano は 2018 年の初頭からコミット数が急激に低下していることが読み取れる. つまり, 過去の SCE の情報だけではこのような急激な変化を捉えることが難しく, 本来の時系列が持つ特徴が平滑化されてしまう懸念がある. そのため, この変化を捉えられるための情報が必要となる. そのための情報が, 貢献者数の増加率である.

Samoladas ら [6] の研究によれば, 貢献者数が増加する毎にプロジェクトの生存率が上昇する傾向にあった. 一方で, REV により生存率が低下している OSS プロジェクトは, コミット数そのもの減少により, 新規の貢献者の流入が鈍化していると考えられる. したがって, ネガティブな SCE の持続と新規貢献者の流入数の減少という特徴量が組み合わさることで, SCE 単体だけでは平滑化されてしまうコミット数の急激な減少という特徴を多少はモデルに組み込めるかもしれない.

貢献者数の増加率は, CAGR(Compound annual growth rate, 年平均成長率) を用いる. 全ての要素が正の実数となる $v = (v_i)$ を n 次のベクトルとすると, CAGR は次式で求められる.

$$\mathrm{CAGR}(v) = \left(\frac{v_n}{v_1}\right)^{1/n} - 1$$

例えば, 年間のユニークな貢献者の推移が $[10, 15, 45, 100]$ である場合, $\mathrm{CAGR}([10, 15, 45, 100]) \simeq 0.78$ となる. これらの説明変数を表にまとめたものが, 表 5.1 である. これらの変数名はコンピュータで扱うため, LaTeX ではなく, スネークケースで表されている.

5.2.2 REV 判定器 (RD)

MIAO フェーズ 2 により出力される MS_{ij} は OSS の競合関係を単に定量化したものであり, それ単体では REV 発生を分類することはできない. したがって, MS_{ij} を入力したときに, それが REV か nonREV であるかを判定する層が必要となる. その役割を担うのが REV 判定器 (REV Detector, RD) である. RD は, REV, nonREV 群の標本を集め, それぞれの MIAO スコアから REV の閾値を統計的あるいは経験的に決定し, その閾値を超えたら REV=1, そうでなければ nonREV=0 を出力する 2 値分類モデルである.

RD は上記の性質を持つ分類器であればどのようなものでも構わない. 例えば, ある閾値 t を統計的あるいは経験的に設け,

$$\mathrm{RD}([\mathrm{MS}_{ij}]) = \begin{cases} 1 & \sum \mathrm{MS}_{ij} > t \\ 0 & otherwise \end{cases} \tag{5.4}$$

のように, MS_{ij} の総和で REV か否かを決定することが考えられる. あるいは, より高度な機械学習モデルとして, 決定木モデルや勾配ブースティング, ロジスティック回帰モデルなどを用いることも考えられる. 例えば決定木モデルを用いる場合には,

$$\mathrm{RD}([\mathrm{MS}_{ij}]) = \mathrm{DecisionTree}([\mathrm{MS}_{ij}]) = \begin{cases} 1 & \text{決定木の分類結果で 1 となる} \\ 0 & \text{決定木の分類結果で 0 となる} \end{cases} \tag{5.5}$$

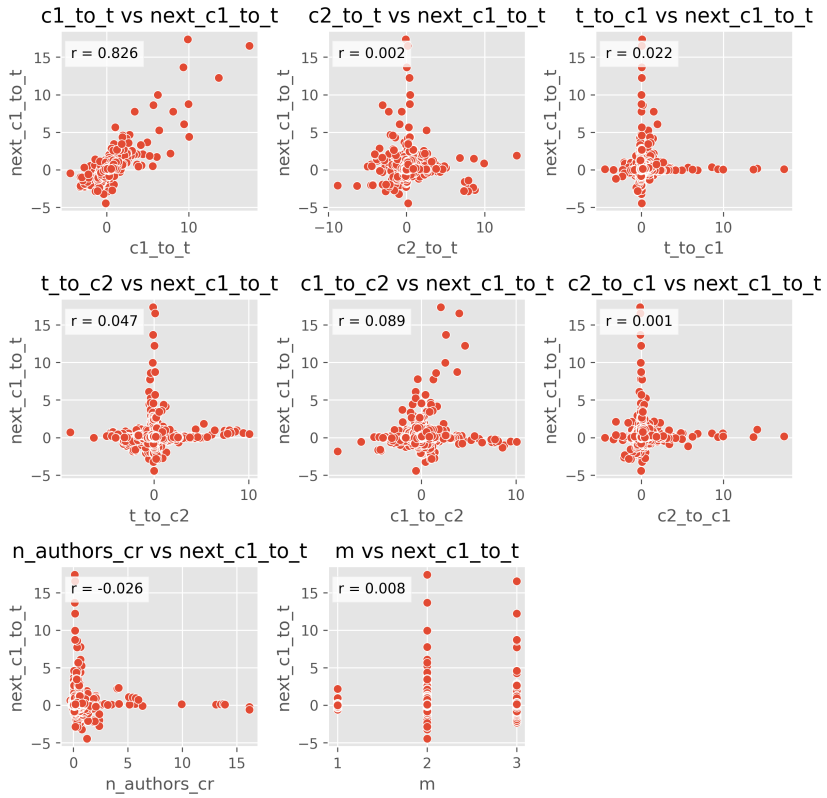
となり, RD は決定木モデルそのものに置き換わる. その他の機械学習モデルを用いた場合にも, 同様に RD はそのモデルに置き換わる.

5.3 実験

本説では, 具体的な SPM および, RD を実装し, 評価データセットによる性能評価のための実験を行う. SPM では, 評価データセットを k 分割交差検証により k 個のサブセットに分割し, サブセット毎の R^2 スコアや MSE などの評価指標を用いた性能評価のための実験を行う. RD では, SPM が予測した SCE を用いて MIAO スコアテーブルを構成し, それを説明変数とすることで, 4 章で行ったような LOOCV による 2 値分類モデルとしての性能評価のための実験を行う.

5.3.1 具体的な SPM および RD の決定

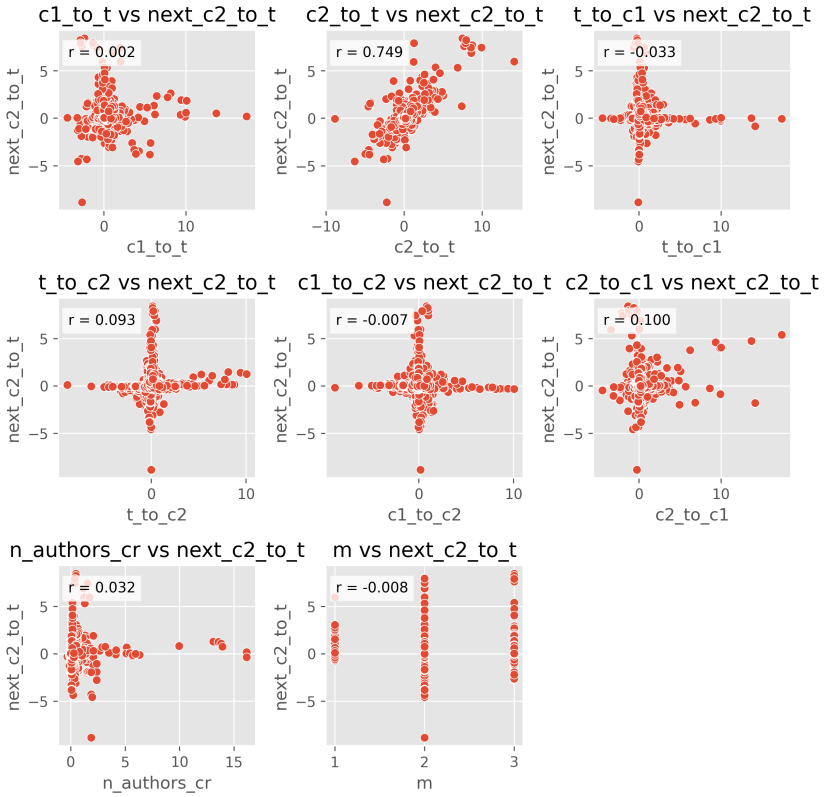
図 5.2: 次期 $C_1 \to T$ と説明変数との散布図



SPM の具体的な実装は回帰モデルとなるが, 回帰モデルにはいくつかの種類があり, 代表的なものに線形回帰モデルと非線形回帰モデルがある. SCE の予測に際してどちらが有効であるかを確認するために, 目的変数と説明変数の分布の散布図をまとめたものが, 図 5.2 と, 図 5.3 である. 図中の 8 つの説明変数は, 表 5.1 に対応するもので, next_c1_to_t と next_c2_to_t がそれぞれ目的変数である, $\mathrm{SCE}_{C_2,T}, \mathrm{SCE}_{C_1,T}$ に対応している. また, 図中の r はピアソンの相関係数を表している.

まず, 目的変数 next_c1_to_t とその他の変数との関連に着目すると, c1_to_t との相関係数が 0.826 と強い正の相関がみられるものの, その他の変数との相関はほぼ

図 5.3: 次期 $C_2 \to T$ と説明変数との散布図



皆無であることが読み取れる．他方の next_c2_to_t においても同様の傾向が見られ，c2_to_t との相関係数が 0.747 と強い正の相関があるが，その他の変数との相関はほぼみられない．つまり，次期 $\mathrm{SCE}_{ij}^{T_m}$ は前期の $\mathrm{SCE}_{ij}^{T_m-1}$ との相関が強く，それ以外の変数とは相関していないことが分かる．このことから，線形回帰モデルにより SCE 予測モデルを構築する場合，c1_to_t と c2_to_t だけで，next_c1_to_t と next_c2_to_t を説明できる可能性がある．一方で，単なる相関関係で捉えられない変数間の関係性もあると考えられることから，コンピュータが非線形な変数間の特徴を捕捉してくれる高度な機械学習手法を用いるのも手である．したがって，本実験では線形回帰モデルと非線形回帰モデルの双方で SPM を構築し，それぞれの性能を評価する．線形回帰モデルの具体的な実装には，3.7 の最小二乗法による重回帰モデルを選定し，非線形回帰モデルには 3.8 の LightGBM を選定する．LightGBM は非線形関係を扱える勾配ブースティングモデルの中でも軽量なものであり，ハイパーパラメータを調整するための試行を多く行えるのが特徴である．線形，非線形のうちで性能が良好な方を SPM 層とし，この後の RD の性能評価に用いる．

具体的な RD の実装については，4 章で行った実験との比較が容易になるという意味で，4 章のものと同様の決定木モデルを採用する．ただし，学習データは SPM が予測した SCE を含む MIAO スコアテーブルとなる．

表 5.2: 実験データセット例

c1_to_t	c2_to_t	t_to_c1	t_to_c2	c1_to_c2	c2_to_c1	next_c1_to_t	next_c2_to_t	n_authors_cr	m	shift_no	label	group
-0.334	0.300	-0.672	-0.279	-0.238	0.886	0.102	3.178	0.868	1	0	tensorflow/tensorflow	1_0
0.102	3.178	-0.272	0.518	-0.315	0.590	0.632	1.664	0.882	2	0	tensorflow/tensorflow	1_0
0.632	1.664	0.213	0.351	-0.042	0.372	1.741	-0.236	0.607	3	0	tensorflow/tensorflow	1_0
-0.045	-0.112	-0.185	-0.047	-0.138	0.704	0.322	2.535	0.953	1	1	tensorflow/tensorflow	1_0
0.322	2.535	-0.083	0.859	-0.083	0.367	0.888	1.295	0.883	2	1	tensorflow/tensorflow	1_0
0.888	1.295	0.232	0.296	-0.218	0.134	1.853	-0.629	0.608	3	1	tensorflow/tensorflow	1_0
0.552	0.882	-0.457	-0.275	-0.403	1.015	0.278	3.592	1.111	1	2	tensorflow/tensorflow	1_0
0.278	3.592	0.021	1.063	-0.293	0.369	0.863	0.895	0.883	2	2	tensorflow/tensorflow	1_0
0.863	0.895	0.262	0.261	-0.453	-0.149	1.901	-1.317	0.617	3	2	tensorflow/tensorflow	1_0
0.359	0.711	-0.369	-0.298	-0.612	0.898	0.481	2.377	1.633	1	3	tensorflow/tensorflow	1_0
0.481	2.377	0.133	0.905	-0.330	0.297	0.636	0.437	0.808	2	3	tensorflow/tensorflow	1_0
0.636	0.437	0.248	0.165	-0.608	-0.090	2.076	-1.257	0.633	3	3	tensorflow/tensorflow	1_0
-0.672	0.886	-0.334	-0.238	-0.279	0.300	-0.272	0.590	1.508	1	0	pytorch/pytorch	1_0
-0.272	0.590	0.102	-0.315	0.518	3.178	0.213	0.372	1.054	2	0	pytorch/pytorch	1_0
0.213	0.372	0.632	-0.042	0.351	1.664	0.280	0.125	0.865	3	0	pytorch/pytorch	1_0
-0.185	0.704	-0.045	-0.138	-0.047	-0.112	-0.083	0.367	1.512	1	1	pytorch/pytorch	1_0
-0.083	0.367	0.322	-0.083	0.859	2.535	0.232	0.134	1.109	2	1	pytorch/pytorch	1_0
0.232	0.134	0.888	-0.218	0.296	1.295	0.313	-0.291	0.883	3	1	pytorch/pytorch	1_0
-0.457	1.015	0.552	-0.403	-0.275	0.882	0.021	0.369	1.615	1	2	pytorch/pytorch	1_0
0.021	0.369	0.278	-0.293	1.063	3.592	0.262	-0.149	1.153	2	2	pytorch/pytorch	1_0
0.262	-0.149	0.863	-0.453	0.261	0.895	0.339	-0.651	0.896	3	2	pytorch/pytorch	1_0
-0.369	0.898	0.359	-0.612	-0.298	0.711	0.133	0.297	3.112	1	3	pytorch/pytorch	1_0
0.133	0.297	0.481	-0.330	0.905	2.377	0.248	-0.090	1.712	2	3	pytorch/pytorch	1_0
0.248	-0.090	0.636	-0.608	0.165	0.437	0.470	-0.641	1.190	3	3	pytorch/pytorch	1_0
-0.279	-0.238	0.300	0.886	-0.672	-0.334	0.518	-0.315	0.477	1	0	chainer/chainer	1_0
0.518	-0.315	3.178	0.590	-0.272	0.102	0.351	-0.042	0.564	2	0	chainer/chainer	1_0
0.351	-0.042	1.664	0.372	0.213	0.632	0.057	-0.434	0.375	3	0	chainer/chainer	1_0
-0.047	-0.138	-0.112	0.704	-0.185	-0.045	0.859	-0.083	0.554	1	1	chainer/chainer	1_0
0.859	-0.083	2.535	0.367	-0.083	0.322	0.296	-0.218	0.564	2	1	chainer/chainer	1_0
0.296	-0.218	1.295	0.134	0.232	0.888	-0.035	-0.892	0.387	3	1	chainer/chainer	1_0
-0.275	-0.403	0.882	1.015	-0.457	0.552	1.063	-0.293	0.700	1	2	chainer/chainer	1_0
1.063	-0.293	3.592	0.369	0.021	0.278	0.261	-0.453	0.564	2	2	chainer/chainer	1_0
0.261	-0.453	0.895	-0.149	0.262	0.863	-0.071	-1.171	0.387	3	2	chainer/chainer	1_0
-0.298	-0.612	0.711	0.898	-0.369	0.359	0.905	-0.330	0.854	1	3	chainer/chainer	1_0
0.905	-0.330	2.377	0.297	0.133	0.481	0.165	-0.608	0.564	2	3	chainer/chainer	1_0
0.165	-0.608	0.437	-0.090	0.248	0.636	-0.209	-2.243	0.390	3	3	chainer/chainer	1_0

5.3.2 実験データ

SPM におけるデータセットは, 表 4.4 内の OSS に関する, 表 5.1 の情報と目的変数, next_c1_to_t, next_c2_to_t により構成される. ただし, そのままでは標本数が 522 とそれほど多くないので, 標本数を拡張するために, 1ヶ月, 2ヶ月, 3ヶ月の期間シフトを加えた計 2088 件の標本でデータセットを構成する. 表 5.2 は, 本実験で用いる実際のデータセットの内, グループ 1 のみを抜粋したものである. 表 5.2 のうち, shift_no, label, group は各行がどの OSS かを識別のための情報であり, モデルの学習に用いるものではない. 本表を見ると, 目的変数の next_c1_to_t および, next_c2_to_t は来期の c1_to_t, c2_to_t となっている. 例えば, tensorflow における m=1, shift_no=0 の next_c1_to_t は 0.102 で, 来期 (m=2) の c1_to_t の 0.102 に一致しているのが分かる. したがって目的変数として用意できる next_c1_to_t, next_c2_to_t は当該グループの最終 T_m までの SCE となる. そのため, 各グループにおける最大の m を m^* とすると, $1 \leq m < m^*$ までが説明変数として, $1 < m \leq m^*$ までの SCE が目的変数としてデータセット内に存在することになる. 例として, 本表のグループ 1 は最大 $m^* = 4$ であるので, $1 \leq m \leq 3$ までの SCE_{ij} が説明変数に, $1 < m \leq 4$ までの SCE_{ij} が目的変数に設定される.

回帰モデルは外れ値の影響を受けやすくなるため, SCE の分布を確認する. 図 5.4 は各 m 毎の SCE_{ij} の分布を散布図として図示したものである. 本図に従えば, $m = 4$ に 47 を超える大きな外れ値がある. そのため, この値を説明変数から取り除くこと

図 5.4: m 毎の SCE_{ij} の分布

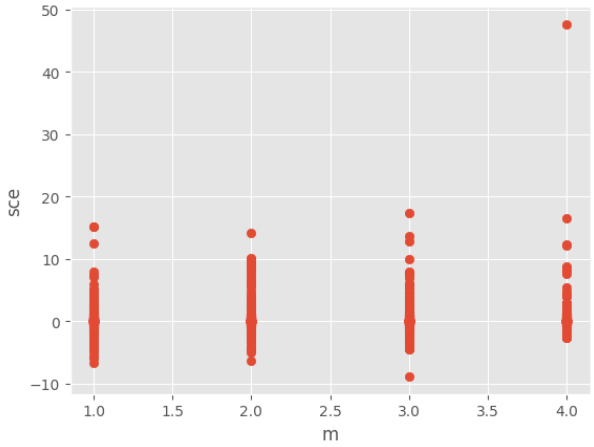


表 5.3: 説明変数間の VIF

変数名	VIF
c1_to_t	1.040
c2_to_t	1.031
t_to_c1	1.068
t_to_c2	1.040
c1_to_c2	1.033
c2_to_c1	1.059
n_authors_cr	1.103
m	1.178

にした. 表 5.3 は実験データの VIF を計算した表である. 本表に従えば, VIF は最大でも 1.178 以下であるため, 多重共線性の可能性は低く, 独立性が高い変数を選べているといえる.

RD のデータセットに関しては, 4 章の決定木モデルに入力するものと同様の形式を用いる. すなわち, 6 つの AMS_{ij} を説明変数として, REV=1, nonREV=0 を目的変数とするものである.

5.3.3 実験方法

まず SPM に関する実験では, 5 分割交差検証を用いて各分割毎に各種モデルの学習及び予測を行い, R^2, MSE, RMSE, MAE といった各種評価指標を得る. 各種モデルとは, 重回帰モデルおよび LightGBM の 2 種類の回帰モデルと, それぞれのモデルに関して $\mathrm{SCE}_{C1,T}$ を目的変数とするもの, $\mathrm{SCE}_{C_2,T}$ を目的変数とするものの計 4 つのことをいう. 以降, $\mathrm{SCE}_{C1,T}$ に関するものを SPM1, $\mathrm{SCE}_{C_2,T}$ に関するものを SPM2 と呼ぶことにする. なお, LightGBM は過学習しやすい特性と多くのハイパーパラメータを持つため, 選択されたハイパーパラメータ次第では予測性能の低下や過学習が起こり得る. そのため, これらの改善のために, ハイパーパラメータの調整を行う必要がある. 本実験では, Optuna[1]を用いて LightGBM の各種ハイパーパラメータの自動最適化を行う. Oputna はベイズ最適化によりハイパーパラメータの最適化を自動化するためのソフトウェアである. 具体的には, optuna.integration.LightGBMTunerCV を用いて, 全てのフォールドで平均的に最も良い性能を示したパラメータセットを特定する.

RD に関する実験では, 4 つのモデルのうち, 良好な性能を示した SPM1, SPM2 を用いて, 各 OSS グループの最終 T_m 期の SCE を予測する. 例えば, グループ 1 の例で言えば, $m = 1, 2, 3$ のデータを用いて, $m = 4$ における SCE を予測することになる. そして, 予測された SCE を MIAO フェーズ 2 に入力し, MIAO スコアテーブル

[1]https://optuna.org/

を出力する．ここで得られた MIAO スコアテーブルから 4 章で行った実験のように LOOCV で決定木モデルを学習し, 分類精度の評価を行うというのが RD の評価のための実験フローである．

5.3.4 実験結果

SPM に関する実験結果

表 5.4: 重回帰モデル実装の推定結果

Model	Index	1	2	3	4	5	Mean
SPM1	R2	0.66 ± 0.08	0.68 ± 0.08	0.63 ± 0.09	0.64 ± 0.14	0.68 ± 0.11	0.66
	ADJ-R2	0.65 ± 0.08	0.67 ± 0.08	0.62 ± 0.09	0.63 ± 0.14	0.67 ± 0.11	0.65
	MSE	0.47 ± 0.11	0.49 ± 0.17	0.47 ± 0.15	0.52 ± 0.11	0.46 ± 0.18	0.48
	RMSE	0.68 ± 0.08	0.69 ± 0.11	0.68 ± 0.11	0.72 ± 0.08	0.67 ± 0.13	0.69
	MAE	0.34 ± 0.03	0.34 ± 0.03	0.34 ± 0.03	0.35 ± 0.03	0.34 ± 0.03	0.34
SPM2	R2	0.54 ± 0.10	0.56 ± 0.09	0.49 ± 0.09	0.54 ± 0.10	0.55 ± 0.11	0.54
	ADJ-R2	0.53 ± 0.10	0.54 ± 0.10	0.48 ± 0.09	0.53 ± 0.10	0.54 ± 0.11	0.52
	MSE	0.47 ± 0.14	0.55 ± 0.16	0.55 ± 0.16	0.61 ± 0.19	0.48 ± 0.17	0.53
	RMSE	0.67 ± 0.11	0.73 ± 0.10	0.73 ± 0.11	0.77 ± 0.12	0.68 ± 0.12	0.72
	MAE	0.34 ± 0.03	0.37 ± 0.03	0.37 ± 0.04	0.37 ± 0.04	0.35 ± 0.03	0.36

表 5.5: LightGBM 実装の推定結果

Model	Index	1	2	3	4	5	Mean
SPM1	R2	0.76 ± 0.05	0.76 ± 0.05	0.72 ± 0.09	0.71 ± 0.12	0.73 ± 0.07	0.74
	ADJ-R2	0.76 ± 0.05	0.76 ± 0.05	0.72 ± 0.09	0.71 ± 0.12	0.73 ± 0.07	0.74
	MSE	0.33 ± 0.11	0.39 ± 0.23	0.35 ± 0.14	0.41 ± 0.10	0.38 ± 0.14	0.37
	RMSE	0.57 ± 0.10	0.60 ± 0.16	0.58 ± 0.12	0.64 ± 0.07	0.61 ± 0.10	0.60
	MAE	0.30 ± 0.03	0.30 ± 0.03	0.31 ± 0.03	0.32 ± 0.03	0.31 ± 0.02	0.31
SPM2	R2	0.66 ± 0.06	0.62 ± 0.11	0.60 ± 0.09	0.66 ± 0.06	0.69 ± 0.06	0.65
	ADJ-R2	0.65 ± 0.06	0.61 ± 0.11	0.59 ± 0.09	0.66 ± 0.07	0.68 ± 0.06	0.64
	MSE	0.34 ± 0.09	0.46 ± 0.15	0.42 ± 0.11	0.45 ± 0.11	0.34 ± 0.12	0.40
	RMSE	0.58 ± 0.08	0.67 ± 0.10	0.64 ± 0.08	0.66 ± 0.09	0.57 ± 0.10	0.62
	MAE	0.32 ± 0.02	0.36 ± 0.02	0.34 ± 0.02	0.34 ± 0.03	0.33 ± 0.03	0.34

表 5.4, 5.5 はそれぞれ, 線形回帰モデルおよび, LightGBM 実装における SPM の予測結果に関する指標をまとめたものである．本表に記載の各種評価指標は, 交差検証における random_state として 42 から 51 までの 10 個の整数値を使用し, 異なるデータセットの分割パターンから得られた結果の平均値が $\mu \pm \sigma$ の形式で記載される．σ は, random_state の違いによる標準偏差で, 平均の周りでどれほど当該指標がばらついているかを表す．したがって, この値が小さいほど, モデルの性能は安定していると言える．

本表において, SPM1 は, next_c1_to_t を予測するもの, SPM2 は next_c2_to_t を予測するものである. 表中の Index は, モデルの評価指標で本実験では, 決定係数 R^2, MSE(Mean Squared Error), RMSE(Root Mean Squared Error), MAE(Mean Absolute Error) を用いる. 1 から 5 の数値は k 分割交差検証の第 k 番目のフォールドであることを表し, そのフォールドに対応した各種評価指標が表記される. 決定係数 R^2 は, $1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}i)^2}{\sum i = 1^n (y_i - \bar{y})^2}$ で定義されるもので, モデルの当てはまりの良さを 0 から 1 の実数値で表現するものであり, 1 に近いほど良いとされる. この指標は, 説明変数が目的変数の変動をどの程度説明できているかを示すものである. 例えば, R^2 が 0.7 の場合, モデルが従属変数の変動の 70%を説明できていることを意味する. 言い換えれば, データの全体的なばらつきのうち, 70%がモデルによって捉えられていると解釈できることを意味する. 残りの 30%は, モデルで説明されない変動となる. なお, モデルの予測値が実測値から大きく外れている場合には R^2 が負の値となることがある. ただし, R^2 は説明変数を増やすほど大きくなるという欠点があるため, これを改善したものが自由度調整済み決定係数である. 本表では ADJ-R^2 がこれに該当し, 基本的には通常の R^2 よりも小さい値となる. MSE は $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ で定義されるもので, 予測値と実測値の差である誤差の二乗の平均である. 誤差なので, 0 に近いほど予測精度が高いことを示す. ただし, 誤差の大きさを二乗で強調するため, 外れ値に敏感となる. RMSE は, $\sqrt{\text{MSE}}$ で定義されるもので, MSE の平方根を取ったものである. MSE よりは外れ値に敏感でなくなり, 元のデータのスケールで結果を解釈できる. MAE は $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$ で定義されるもので, 予測値と実測値の差の絶対値の平均である. MAE も 0 に近いほど予測精度が高いことを示す. また, 外れ値に対して MSE や RMSE ほど敏感ではない. これも, 元のデータのスケールを反映している.

なお, 実験の結果, R^2 スコアなどの各種指標は, 学習データに $m = 1$ のものを含めないほうが性能が向上することが分かった. そのため, 本表はデータセットから $m = 1$ のものを除いた上での結果となっている. このような結果となった要因の一つとして, 最終 T_m における $\text{SCE}_{C1,T}, \text{SCE}_{C2,T}$ は $m \geq 2$ 以降の $\text{SCE}_{C1,T}, \text{SCE}_{C2,T}$ に強い相関があり, $m = 1$ では弱い相関となるとなることが挙げられる. 具体的な数値としては, next_c1_to_t と c1_to_t の相関係数は, $m = 1 : 0.264, m = 2 : 0.792, m = 3 : 0.868$, next_c2_to_t と c2_to_t では, $m = 1 : 0.398, m = 2 : 0.782, m = 3 : 0.710$ となった. いずれの目的変数においても, $m = 1$ の相関係数が最も低いものとなり, これが最終 T_m の予測性能に悪影響を与えている可能性がある.

表 5.7 は, LightGBM 実装における SPM1, SPM2 の学習曲線の統計を要約したものである. count は, 評価に用いたモデルの個数を表し, 10 個の random_state 毎に 5 分割交差検証があるので, 全部で 50 個のモデルを評価することになる. initial_train_rmse と initial_valid_rmse は, 一回目のイテレーションにおける訓練, 検証データにおける RMSE を表す. 25_p から 75_p までの train および valid_rmse は, イテレーションの 25%, 50%, 75%点における RMSE を表す. そして, final_train_rmse と final_valid_rmse

は最終イテレーションにおける訓練, 検証データにおける RMSE を表す. 最大イテレーション回数は 100 であるが, 過学習の抑制のために early_stopping=10 を設定しているため, 早期に学習が終了しているモデルもである. early_stopped が early_stopping されたモデルの割合を表し, best_iteration が 0 から 99 のイテレーションの内で RMSE が最も低いイテレーションとなる. 通常, モデルの予測性能が低い場合に, 学習データセットと検証データセットによる学習曲線は大きく乖離する. つまり, 訓練データセットでは予測精度が高くなるが, 検証データセットでは予測制度が低くなるといった問題が起こる. また, 典型的に過学習を示すケースでは, イテレーションを増す毎に訓練データセットでの予測精度は上がるにも関わらず, 検証データセットでは予測精度が低下し, RMSE が上向きになっていくような事象が起こる. つまり, 本表では, final_train_rmse と final_valid_rmse に大きな乖離がなく, initial から 25_p, 50_p, 75_p, final までにかけて平均的に RMSE の値が低下していればよいことになる.

SPM1 では訓練データの方の学習曲線の平均値は, initial: 1.15, 25_p: 0.68, 50_p: 0.56, 75_p: 0.49, final: 0.45 という推移で, イテレーション毎に低下していることがわかる. 検証データでの学習曲線の平均値は, initial: 1.14, 25_p: 0.72, 50_p: 0.56, 75_p: 0.62, final: 0.61 と推移しており, RMSE の改善が一定値に収束する傾向にある. これは引き続き学習を続けると過学習してしまう兆候となるが, n_iterations の平均が 80.72, early_stopped が 0.42 と 4 割強のモデルは早期に学習が打ち切られており, final_valid_rmse と best_valid_rmse とでは 0.01 ポイントしか差がないことから, 過学習が抑制されていることが示唆される. また, 訓練と検証データセットの final 時点の差は 0.16 ポイントであり, 深刻な乖離は見られない.

SPM2 の訓練データの学習曲線の平均値は, initial: 1.05, 25_p: 0.56, 50_p: 0.45, 75_p: 0.39, final: 0.35 という推移で, 検証データの方は, initial: 1.05, 25_p: 0.70, 50_p: 0.65, 75_p: 0.63, final: 0.63 と, SPM1 同様に訓練データの RMSE の低下が顕著で, 検証データは 0.63 に徐々に収束している. n_iterations の平均が 88.64, early_stopped が 0.34 と 3 割強のモデルは早期に学習が打ち切られており, final_valid_rmse と best_valid_rmse とでは差がないことから, 過学習が抑制されていることが示唆される. 訓練と検証データセットの final 時点の差は 0.28 ポイントであり, SPM1 よりは乖離が大きい結果となった.

これまでの結果に記載されている全てのモデルは Optuna で得られた同一のハイパーパラメータを用いて計算された. 実験に用いたハイパーパラメータは表 5.8, 5.9 に示すものである. この内, 重要度の高いパラメータは num_leaves と min_child_samples である. num_leaves のデフォルト値は 31 で値が大きくなるなるほど, モデルの複雑度が増し過学習しやすい傾向となる. min_child_samples は一つの葉のデータの最小個数を表す. デフォルト値は 20 で, これが小さい場合に過学習しやすくなる. 本実験のものでは, num_leaves は 17, 26 とデフォルト以下の値が選択されており, モデルの複雑度はそれほど高くない. min_child_samples は 10, 10 とデフォルト値以下であるが, 各交差検証における訓練データセットのサンプルは 1100 程度なので, 許容範囲であると言える.

SPM の予測結果から, 各グループに関して AMS_{ij} を計算した表を, 付録中の表 B.1, B.2, B.3 に示す. これは 4 章で示した結果表 A.2 らと同様のフォーマットである.

RD に関する実験結果

表 5.4, 5.5 に関する詳細なモデルの評価は次節で行うが, 簡易的に各種評価指標を比較すると, SPM1, SPM2 共に LightGBM 実装のほうが全体的な性能が良好となった. したがって, RD に関する実験では, LightGBM 実装の SPM を用いる. MIAO フェーズ 2 に入力する SCE は, 5.3.4 の SPM の実験で得られたものをそのまま利用する. SPM における実験では, 10 個の異なる random_state 毎に 5 分割交差検証が実施されるというものであった. このとき, 最終 T_m における予測 SCE は, 10 個の random_state から得られる予測 SCE を平均したものとなる. すなわち, あるグループにおける第 r 番目の random_state に対応する $\hat{\mathrm{SCE}}_{ij}^{T_m}$ を $\hat{\mathrm{SCE}}_{i,j,r}^{T_m}$ と表記すると, 各グループの最終 T_m における予測 SCE は $\hat{\mathrm{SCE}}_{ij}^{T_m} = \dfrac{1}{10}\sum_{r=1}^{10} \hat{\mathrm{SCE}}_{i,j,r}^{T_m}$ で算出される.

この $\hat{\mathrm{SCE}}_{ij}^{T_m}$ を全てのグループにおいて計算し, MIAO フェーズ 2 に入力することで, 予測 MIAO スコア表を得る. その結果は, 付録 B.1 に示すものである. したがって, RD のための説明変数が付録 B.1 の MIAO スコア表となる. これを学習データとして, LOOCV により得られた分類結果のレポートが表 5.10 で, 混同行列が表 5.11 である. また, RD の分類結果と真値の比較表を付録中の表 B.4 に示す. これらは全て 4 章のものと同様のフォーマットである.

5.4 評価

SPM に関する実験の結果, SPM1, SPM2 ともに LightGBM 実装のほうが各種指標のスコアが高性能を示すことが分かった. 具体的には, 重回帰モデル実装の SPM1 では, 平均的な R^2 スコアが 0.66, RMSE が 0.69, SPM2 では, 平均的な R^2 スコアが 0.54, RMSE が 0.72 である. LightGBM 実装の SPM1 では, 平均的な R^2 スコアが 0.74, RMSE が 0.60, SPM2 では, 平均的な R^2 スコアが 0.65, RMSE が 0.62 となった. これが意味することは, LightGBM 実装の方が, モデルの当てはまりが良く, また予測値と真値がより近いことを表す. 重回帰モデル, LightGBM 双方において, SPM2 よりも SPM1 の性能が良好であることから, next_c1_to_t は next_c2_to_t よりも予測性が高く, シンプルな構造であることが読み取れる. 変数の並び順から, c1 はデファクトスタンダードであるため, 最も外生性が高いと仮定されているものからの対象への影響は, 新興のものに比べて, 予測性能が優れていると言える. SPM2 に関しては前述のように, 重回帰, LightGBM 共に SPM1 に比べて予測性能が低下する結果となった. 特に, 重回帰版の SPM2 の R^2 は 0.54 であり, next_c2_to_t を線形モデルで説明することが困難であることが示唆される. つまり, 新興の OSS から対象への影響は, デファクトスタンダードからの影響に比べ複雑であり, 非線形関係を考慮できるモデルの仕様が推奨される. ただし, LightGBM 版の R^2 も 0.65 と SPM1 のものに比べて 0.9 ポイントも低いものであるため, LightGBM でも複雑な変数の関係性を十分にモデルに取り込めているとはいえない.

その他として, 複数の random_state の使用により, 各フォールド毎に極端な性能差が見られず, データセットの分割基準でモデルの性能が大きく変わることは抑制

されている. 極端な性能差とは, 例えばフォールド 1 の R^2 は 0.2 だが, フォールド 2 の R^2 は 0.7 となるようなものである.

これまでを総括すると, SPM1, SPM2 双方ともに非線形モデルのほうが明確に性能が高い結果となった. ただし, LightGBM を用いても, 過去の SCE と貢献者の変動率を用いた将来の SCE 予測精度は中程度であり, SCE の予測が簡単なタスクではないことが分かる. また, 今回のデータにおける next_c2_to_t の変動は next_c1_to_t に比べ複雑であることも分かった. ただし, これはデータの並び順に依存する結果であるとも言えるため, より信頼性の高い結果を得るには, より大規模なデータセットを用いて, 感度分析等を行う必要がある.

RD の分類性能 (表 5.10) については, Accuracy は 0.73 で, 4 章の結果（表 4.10）と比べると, 0.12 ポイントも低下した. 特に, nonREV クラスにおいては F1 スコアが 0.85 から 0.7 ポイントまで低下し, 多くのケースを REV と分類してしまう結果となった. そのため, 真陽性の数が 27 と 4 章の結果と変化がないにも関わらず, 偽陽性の増加により REV クラスの F1 スコアも 0.84 から 0.75 に低下する結果となった. このような結果が招かれたのは RPM の予測性能が十分でなく, AMS_{ij} の分布が変化してしまったことによると考えられる. 後の考察において, この分布の変化が予測性能に与える影響などの詳細な分析を行う.

表 5.6: SMP1 における学習曲線の統計的要約

	count	mean	std	min	25%	50%	75%	max
n_iterations	50.00	80.72	27.83	16.00	53.75	100.00	100.00	100.00
initial_train_rmse	50.00	1.15	0.06	0.93	1.13	1.16	1.19	1.23
initial_valid_rmse	50.00	1.14	0.23	0.76	0.97	1.13	1.27	1.81
25p_train_rmse	50.00	0.68	0.11	0.54	0.60	0.64	0.76	0.98
25p_vaild_rmse	50.00	0.72	0.15	0.49	0.61	0.69	0.79	1.22
50p_train_rmse	50.00	0.56	0.10	0.46	0.49	0.52	0.62	0.85
50p_vaild_rmse	50.00	0.65	0.14	0.46	0.55	0.62	0.72	1.11
75p_train_rmse	50.00	0.49	0.09	0.40	0.44	0.46	0.55	0.78
75p_vaild_rmse	50.00	0.62	0.13	0.43	0.53	0.59	0.68	1.03
final_train_rmse	50.00	0.45	0.09	0.35	0.39	0.41	0.50	0.73
final_valid_rmse	50.00	0.61	0.12	0.44	0.52	0.57	0.66	0.96
best_valid_rmse	50.00	0.60	0.12	0.43	0.52	0.56	0.66	0.96
best_iteration	50.00	74.52	31.37	5.00	42.75	94.50	99.00	99.00
early_stopped	50.00	0.42	0.50	0.00	0.00	0.00	1.00	1.00

表 5.7: SMP2 における学習曲線の統計的要約

	count	mean	std	min	25%	50%	75%	max
n_iterations	50.00	88.64	19.68	33.00	77.75	100.00	100.00	100.00
initial_train_rmse	50.00	1.05	0.04	0.95	1.02	1.05	1.07	1.11
initial_valid_rmse	50.00	1.05	0.16	0.70	0.93	1.05	1.17	1.41
25p_train_rmse	50.00	0.56	0.06	0.50	0.53	0.54	0.57	0.75
25p_vaild_rmse	50.00	0.70	0.11	0.45	0.61	0.68	0.76	0.98
50p_train_rmse	50.00	0.45	0.05	0.40	0.42	0.44	0.47	0.61
50p_vaild_rmse	50.00	0.65	0.10	0.43	0.58	0.64	0.71	0.96
75p_train_rmse	50.00	0.39	0.05	0.34	0.35	0.37	0.40	0.54
75p_vaild_rmse	50.00	0.63	0.10	0.42	0.56	0.62	0.69	0.94
final_train_rmse	50.00	0.35	0.05	0.30	0.32	0.34	0.36	0.49
final_valid_rmse	50.00	0.63	0.10	0.42	0.56	0.62	0.68	0.93
best_valid_rmse	50.00	0.63	0.10	0.41	0.56	0.61	0.68	0.93
best_iteration	50.00	83.26	23.21	22.00	66.75	98.00	99.00	99.00
early_stopped	50.00	0.34	0.48	0.00	0.00	0.00	1.00	1.00

表 5.8: SPM1 のハイパーパラメータ

Hyper Parameter	Value
objective	regression
boosting_type	gbdt
metric	rmse
min_child_samples	10
feature_pre_filter	False
lambda_l1	0.01
lambda_l2	0.00
num_leaves	17
feature_fraction	1.00
bagging_fraction	0.67
bagging_freq	1

表 5.9: SPM2 のハイパーパラメータ

Hyper Parameter	Value
objective	regression
boosting_type	gbdt
metric	rmse
min_child_samples	10
feature_pre_filter	False
lambda_l1	0.10
lambda_l2	0.10
num_leaves	26
feature_fraction	0.50
bagging_fraction	1.00
bagging_freq	0

表 5.10: 分類結果のレポート

	Precision	Recall	F1-score	Support
nonREV	0.81	0.62	0.70	34
REV	0.68	0.84	0.75	32
Accuracy	–	–	0.73	66
Macro avg	0.74	0.73	0.72	66
Weighted avg	0.74	0.73	0.72	66

表 5.11: 混同行列

		Predicted	
		Yes	No
Actual	Yes	27	5
	No	13	21

第6章　考察

6.1　MIAO の結果の解釈および考察

4 章の実験では, REV クラスと nonREV クラスの合計 66 グループに対して, 日次コミット数の推移をもとに, MIAO を適用した. 実験結果から, アクティビティ時系列データの 90% 以上は, 分数差分変換を用いることなく, 季節調整後のみで定常過程と判定されることが分かった. SVAR モデルのラグ次数は AIC を指標として自動選択し, さらに撹乱項の系列相関を Ljung-Box 検定にかけることで，モデルの妥当性を担保した. 検定の結果, 90% 以上のケースで p 値 0.1 以上を達成した. このことから, 多くの OSS の日次のコミット数の推移は, 過剰な前処理を行わずとも定常性, あるいはトレンド定常性を持ち, VAR のような線形モデルで十分に予測可能であると考えられる. また, フェーズ 2 から得られる MIAO スコアは, 決定木モデルを用いた評価 (表 4.10) において, Accuracy が 0.85 ± 0.35 という良好な性能で REV か nonREV かを分類できることを示した. 一方で, なぜこのような結果になったかの議論はまだ行えていない.

本節では, このような結果が導かれた要因として, 主に図 4.5 の決定木を用いて解釈することを試みる. まず, 特筆すべきは, 決定木の最上位ノードが $c2 \to t \leq -0.137$ という条件で分岐していることであり, これは $c2$ から t への影響が REV を決定する最も重要な要因であることを示唆している. この条件が False, つまり $c2$ から t への正の影響が限定的で, 負の影響が大きくなる可能性がある場合, REV クラスのうち 31/32 が右側のノードに分岐する. 右側のノードは REV クラス, 左側のノードは nonREV クラスと判定される傾向にある. つまり, $c2 \to t$ の大小で, 大多数の REV か nonREV かが決定してしまうといっても良い. 変数の並び順に基づけば, c2 は, c1 よりも目新しい OSS となっている. そのため, target が REV となる場合に, より新興の OSS からの影響が大きい可能性がある. 右側のノードの次の分岐は $c_1 \to c2 \leq 0.019$ で, これが True となる場合は $c1$ から $c2$ への正の影響が強くなることを表す. つまり, REV クラスにおいては, 相対的に競合同士の影響が正になりやすいということである. この分岐をさらに辿ると, 次の条件は $c1 \to t \leq -0.638$ である. これが False となる場合, つまり $c1$ から t への正の影響が限定的, あるいは負の影響が強くなる場合に, 28 サンプルが REV に分類される.

逆に, ルートノードの条件が True の場合, すべてのサンプルは nonREV に分類される. $c2 \to c1 \leq 0.067$ という分岐はあるものの, この条件の分類における寄与はそれほど大きくない. これらから, REV が起こる場合には, $c2, c1$ 双方から t に対する限定的な正の影響, あるいは負の影響があることが分かった. 特に, より新興の競合 OSS からの強い累積的な負の影響が，対象 OSS を衰退・停止に追い込む一因」であ

る可能性を, MIAO の定量結果が示唆している．これは，Coelho らの先行研究で言及されていた ”強力な競合の出現が OSS を衰退させる” 事象の一つの実例を, 時系列データを用いることで定量化できたことを意味する．したがって, 複数期間における累積的なインパルス応答は, REV を特徴づける値として有用なものであるといえる．

ターゲットから競合への影響は決定木の分類条件に寄与していないものの, 表 4.3 の M1 は 27 件の真陽性ケースのうち 15 件で成立し, そのうち 9 件は表内で $\mathrm{D}(\mathrm{AMS}_{C_1T}, \mathrm{AMS}_{TC_1})$ または $\mathrm{D}(\mathrm{AMS}_{C_2T}, \mathrm{AMS}_{TC_2})$ の最大値を示している．つまり, 真陽性の内, 1/3 のサンプルは MIAO スコアの解釈において与えられた REV を疑うケースに一致するものが含まれることになる．

次に誤分類の要因を分析する．このため, 決定木において REV 分類に重要な変数として特定された $c1 \to t$ および $c2 \to t$ の関係に注目した．混同行列の各グループについて, すべての期間シフトにおける $\mathrm{MS}^* = \mathrm{MS}C_1T + \mathrm{MS}C_2T$ を計算し, 各ケースの平均を取ったところ, 真陽性 (TP): 2.50, 真陰性 (TN): -2.34, 偽陽性 (FP): 0.68, 偽陰性 (FN): -3.01, の結果が得られる．

これらの平均スコアを比較すると, TP と FN, および TN と FP 間に顕著な差異が見られる．まず, TP と FN の差は 5.51 である．これは TP 事例では MS^* が高く, 競合から対象への強い負の影響を示す一方, FN 事例では MS^* が負の値を示すことを意味する．この差異は Mann-Whitney U 検定 [50] においても 5%有意水準で有意であった (検定統計量: 1897.0, p 値: 0.00)．この結果から, REV クラス内においても $c1 \to t, c2 \to t$ がネガティブとなるサンプルがあることを示している．

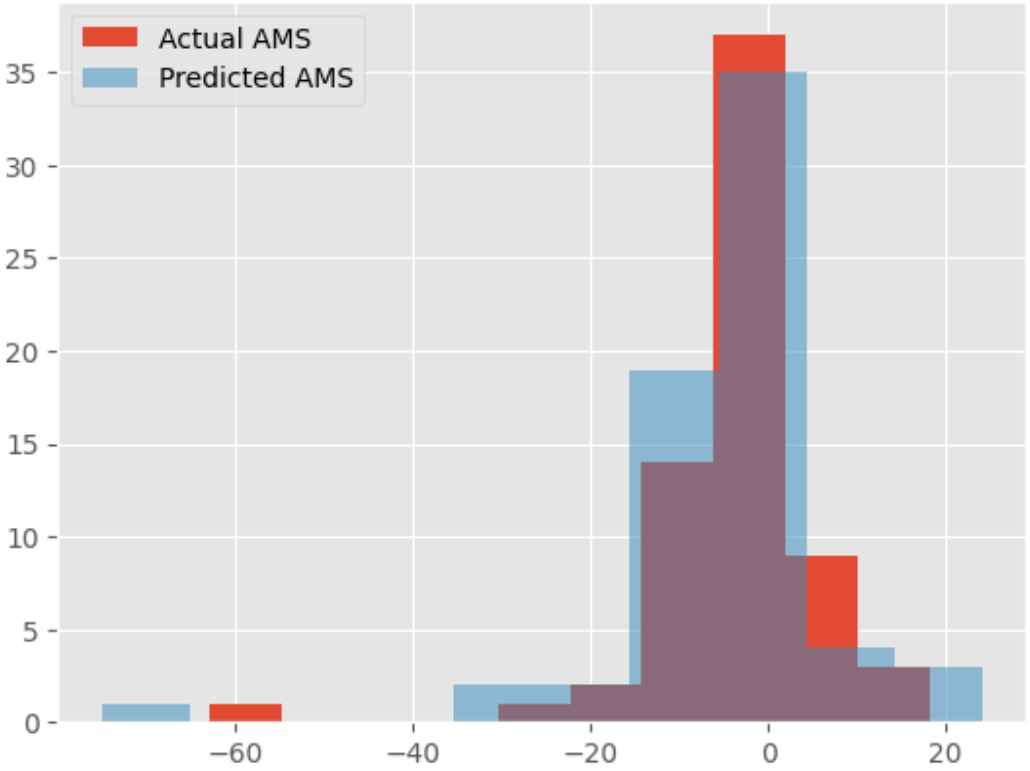
TN と FP については, その差は 3.02 で TP と FN の差を下回ったものの, この差異も Mann-Whitney U 検定で有意であった (検定統計量: 610.0, p 値: 0.00)．したがって, REV が発生していない場合でも競合からターゲットへの相対的に強い負の影響が観測され, これが誤分類につながっている．nonREV クラスはより多くの期間区分を持つことを考慮すると, これが偽陽性の主要な要因となっている可能性がある．そのため, データセットの整理や OSS の成長モデルをより考慮することが誤分類を減らすため有効な対策となるかもしれない．

その他に, 誤分類の根本的な原因として, データセット構築の段階で, REV クラスに本来 REV ではないものが含まれている可能性がある．現状, OSS 開発の停止理由が REV によって引き起こされたかどうかを確定的に知ることは困難であり, 定性的なルールによって判断されている．その結果, データセットには REV ではないケースが含まれている可能性がある．したがって, 当該 OSS が REV で衰退したかどうかをより確度を高めて知ることは, 本研究の信頼性を高めるうえで重要なものとなる．

6.2 RPM の結果の考察

RPM では, 4 章の事後分析と異なり, 分類結果 (表 5.10) の Accuracy が 0.73 と低下した．予測 SCE を用いるというアプローチであるため, 予測 SCE の精度に, 分類結果も依存することは自明である．まず, 分類性能が 4 章の事後分析よりも下回った

図 6.1: 真の AMS と予測 AMS の分布の比較



という点は, 事後分析で得られた実際の SCE が, REV か nonREV かを定量化するための特徴量としてより適切であることを示す. つまり, SPM の予測精度を向上させ, 実際の SCE により近い値を予測できれば, RPM の予測性能の向上につながるという結論が得られる.

次に, RPM の分類性能の低下の要因を分析する. 図 6.1 は真の AMS と, 予測 AMS のヒストグラムを取り, それぞれを比較するために重ねたものである. 本図を見ると, 予測 AMS はヒストグラムの形状が元のものよりも広がり, やや左側に偏って分布するような形状となっている. 分布の広がりは, 決定木において曖昧さあるいは複雑さを増す要因となり, 左側への偏りは, SCE が 0 以上と予測される割合が多くなっていることを表している. 言い換えると, 各グループにおける最終 T_m の SCE は真値と比較して正の影響として予測されてしまっているといえる. これが, 4 章の事後分析 (表 4.11) に比べ, 偽陽性が増加 (表 5.11) する要因となっていると考えられる.

実際, REV の確度が高いグループ 1 において, Pytorch から Chainer への真の SCE は [-0.43, -0.90, -1.80, -2.24] として推移するにも関わらず, 予測 SCE は [0.44, 0.50, 0.28, 0.20] と推移している. SCE の時点ではネガティブな値であるほど, 競合から対象への負の影響が強いことを表すので, 本来は負の影響であるものを, 正の影響として予測してしまっていることになる. これは, 過去の SCE と貢献者の CAGR だけでは, REV による急激な活動量の減少を依然として平滑化してしまっていることを示唆する. したがって, より急激な変動を捉えられる変数や, ダウントレンドを形成しやすい変数などを選択し, SCE の過剰な見積もりを抑制する必要がある.

6.3 本研究の利点と限界点

これまでの実験結果や考察から, MIAO および RPM を含めた本研究の利点と限界点をまとめる. まず, 利点は以下の 4 つである.

1 つめは, MIAO により, OSS プロジェクト間の競合関係を良好な精度で定量化できる点が挙げられる. これにより, これまで定性的な評価にとどまっていた OSS の競合関係を定量的に評価できるようになる. 先行研究が主に内部的な持続可能性要因に注目していたのに対し, MIAO は OSS プロジェクトの生存性に影響を与える競合のダイナミクスを理解する新たな手法を提供する.

2 つめは, MIAO スコアの解釈が容易な点である. 単一の MIAO スコアは, ある OSS の活動量が変化した際のある OSS への影響を, 累積的なインパルス応答の総和として単純計算したものである. また, それぞれのインパルス応答は直交化されており, 理論的には, ある OSS からある OSS への固有の影響を数値化している. したがって, 同グループ内で MIAO スコア同士を単純比較可能であり, 値の大小に応じて競合関係を直接的に把握することができる. また, 各 MIAO スコアの独立性から, 決定木モデルなどを用いて, どの変数からどの変数への影響が REV の分類に寄与するか, その時の閾値はどのくらいか, を視覚的に把握することが可能となる. 具体的には, 例えば, 新興からの影響が強いのか, レガシーなものからの影響が強いのかを, 変数の選び方とオーダーを規定することで把握できる. この解釈可能性により, 研究者や開発者は競合関係にある OSS の生存または衰退に寄与する要因や閾値を理解することができる. 本研究では決定木モデルを用いたが, その他にも説明性の高いモデルを選択して分類モデルを構築することで, 決定木と同様になぜそのような結果となったかを解釈することも可能である.

3 つめは, RPM により REV の発生そのものを予期できる点である. MIAO は REV の事後的な分析に焦点を当てているため, 将来の REV 発生を予期する能力は持たない. 一方で, RPM を構成する SPM では, MIAO で得られた SCE を説明変数とすることで, T_{m+1} 時点の SCE を予測することができる. これにより, T_{m+1} 時点の予測 MIAO スコアを算出し, 将来の競合間の関係性をも定量化可能となる. このとき, 当期 T_m と比較し, T_{m+1} において競合と対象 OSS とで明確な競合関係が出現する場合, REV 発生に備えることができる.

4 つめは, 最小限のデータ要件で実用的に利用可能な点である. 先行研究ではコミュニティ構造や開発者のダイナミクスの分析, および大規模なネットワーク分析が必要とされ, データ収集が困難であった. MIAO では, 実験で示したように対象の OSS と競合する OSS の時系列データのみを必要とし, OSS 採用者が容易に MIAO を実行して採用した OSS の位置づけと将来性を評価することができる. RPM においても, 過去の SCE と貢献者数の推移のみをデータとして得られればよく, これらは単一の Git リポジトリを clone することで容易に得られるものである. この分析の容易性は, 組織が OSS ポートフォリオのリスクを評価・管理する方法に大きな影響を与える可能性がある. さらに, 必要なデータの単純さにより, MIAO を継続的インテグレーション (Contenius integration, CI) に統合し, OSS の競争力の健全性を自動的に継続評価することが可能となる. CI では, 事後分析に MIAO を, REV 予測に RPM

をそれぞれ用いることで, 現時点におけるリスクと将来におけるリスクの双方を分析可能となる.

本手法は REV を判定/予測するための有効な手法ではあるが, REV の定義に主観性が交じること, また厳密な因果分析の欠如から, 妥当性への脅威が存在する. 1 つ目の懸念は, 考察でも挙げた通り, REV を確定的に知ることが困難であり, REV クラスに本来 REV ではないケースが含まれている可能性がある点である. これを防止するために, 手作業でデータセットを取捨選択する必要があり, データセット構築のハードルが上がることから, サンプルサイズの拡大に対する制限も生まれている. 今後の研究において, より確立された REV の定義を導入できれば, 本研究の信頼性を向上させることができる. 2 つ目の懸念は, MIAO が厳密な因果推論に基づいておらず, コミットの頻度のみで REV を特定している点である. 本来のコミット頻度は複雑なものであり, 競争関係以外の外部要因や内部要因の影響も受ける. そのため, MIAO があるケースを REV と特定しても, そのケースでは他の要因によって開発が停止した可能性がある. 3 つ目の懸念は, RPM の精度に関する懸念である. 現状の RPM では陽性と偽陽性と判定する割合が多く, 多くの場合に REV と判定される可能性がある. つまり, REV のリスクを過剰に評価してしまう可能性が高く, 仮に REV と判定されても, RPM の出力結果に関する詳細な分析が必要となる.

6.4 本研究と関連研究の比較

ここでは, 本研究と関連研究に関して, カバー領域とパフォーマンスの観点で比較を行う. これにより, 本研究と関連研究の違いや得意/不得意を明確にし, 分析の目的に応じた手法を選ぶための指標を与える.

6.4.1 カバー領域の比較

	評価項目	本研究	Liao ら (2019)	Eluri ら (2019)	Raja ら (2010)	Samoladas ら (2012)	Valiev ら (2018)
	分析手法	MIAO, RPM	LP(p)[*1]	MLP	VI[*2]	生存分析	混合分析 [*3]
外的要因	競合関係の考慮	◎	×	×	×	×	×
	エコシステムの考慮	△	×	×	×	×	◎
内的要因	技術的要因の考慮	×	◎	○	×	○	×
	コミュニティ要因の考慮	×	△	△	◎	○	○
	PM 要因の考慮	×	×	×	○	×	×
その他	時間経過の考慮	◎	○	×	○	◎	○

表 6.1: 各研究がカバーする領域を比較した表

[*1] $LP(p) = \alpha \cdot log_2[n(p)] \cdot l(p) \cdot log_2[m(p)] + \beta \cdot lab(p)$

[*2] Viability Index (VI): Raja らが提案した独自の指標で, OSS プロジェクトの生存可能性を評価する複合指標. 活力, 組織力, 回復力の 3 つの側面を統合している.

[*3] ネットワーク分析, 生存分析, 構造化インタビュー

評価基準: ◎: 最も対応している, ○: 対応している, △: 部分的に対応, ×: 未対応

各研究がカバーする領域を比較した表が, 表 6.1 である. この表では, 6 つの領域が比較対象となる.

6 つの領域とその評価の尺度について述べる. 6 つの領域は, 外的要因, 内的要因, その他のカテゴリに分類される. まず外的要因から述べると, 1 つ目は, 競合関係の考慮である. これは文字通り, 外的要因の内, 競合関係を考慮可能であるかを示すものである. これは特に時系列分析など, 分析手法を限定したものではなく, どのような分析手法を用いても競合関係を考慮している場合に◎を与える. 2 つ目は, エコシステムの考慮である. エコシステムは, 特定のプラットフォームやテクノロジーを中心として形成される, 相互に依存し合う要素の集合体を指す, 具体的には Android, iOS などのモバイルプラットフォームや, Python のソフトウェアエコシステムである PyPI , JavaScript のソフトウェアエコシステムである NPM などが該当し, これらプラットフォームの上で多くのソフトウェアが開発される. このようなエコシステムの考慮の規模により評価を行う. 例えば, プラットフォーム全体を含むような研究では◎を与え, 部分的な場合は△を与える.

内的要因は 3 つの領域からなる. 内的要因は外的要因に比べると, 定量化が比較的行いやすい領域であるため, どれくらいの量を扱っているかで評価する. つまり, モデルや分析手法の中で当該領域を最も多く扱っていれば◎, 次に扱っていれば◯のような評価となる. 内的要因の 1 つ目は, 技術的要因の考慮である. 技術的な要因とは, 特定の技術に関する情報やコードの保守性や複雑性などを指す. 例えば, 各 OSS がどのような技術カテゴリであるか, どのようなプログラミング言語を用いているか, コードの品質はどうか, などが技術的要因に含まれる. 4 つ目は, コミュニティ要因の考慮である. コミュニティ要因とは, コミュニティの規模や所属する開発者の人数や質, 運営の健全性などを指す. 例えば, 大企業がホストしているか, 個人がホストしているかなどの規模に関するものや, ひとつの issue や PR に対する返答率や速度, 開発者の相互作用, 新規貢献者の定着率などが挙げられる. 5 つ目は, プロジェクトマネジメント (PM) 要因の考慮である. コミュニティ要因と PM 要因は共通部分も見受けられるが, PM では, 主に OSS の運営を如何にコントロールできているかを見る. これは例えば, 新機能やバグフィクスのためのリリース頻度や, タスク割当の効率性などが挙げられる.

最後は, 時間経過を考慮しているかである. 通常, OSS の分析はある時点におけるスナップショットのデータが用いられる. しかし, 多くの OSS は一定の成長モデル [18] に従うため, どのステージで分析したかによって, 分析結果が大きく変わる可能性がある. したがって, 単一の時点を分析しただけでは OSS の生存性を見誤る可能性があり, 複数期間にわたって分析, 評価することがフェアである. この領域では, 時間経過がモデルに組み込まれている度合いで評価される. 例えば, 時系列分析や生存分析では時間経過の評価は◎となる. 一方で, 回帰モデルの変数として時間経過を考慮したスカラー値が含まれるような場合は, ◯や△を与える.

表 6.1 の結果をまとめると, 本研究がカバーする領域は, 外的要因の 2 つと, 時間経過ということになった. したがって, 本研究は内的要因は一切考慮しない, 外的要因の分析に特化した手法となる. 内的要因, 外的要因の両方を扱う最もバランスが良いものが Valiev らである. その他の研究は, 内的要因に特化した手法で, 特に, Liao

らは技術要因が強く, Raja らはコミュニティ要因が強い研究となった.

6.4.2 パフォーマンス比較

評価項目	本研究	Liao ら (2019)	Eluri ら (2019)	Raja ら (2010)	Samoladas ら (2012)	Valiev ら (2018)
予測モデルの提供	◎	◎	◎	×	×	×
実験のサンプルサイズ	66 Group(N=198)	843,763	449	232	1,147	46,547
モデルの精度	Acc: 0.85 F1: 0.84	36%が誤差 10% 53%が誤差 30%	Acc: 0.81 F1: 0.82	Acc: 0.93 F1: 0.93	-	-
一般性	△	◎	△	○	○	○
説明性	◎	◎	△	◎	○	◎
実務可能性	◎	◎	○	△	◎	×

表 6.2: 各研究のパフォーマンス比較表

評価基準: ◎: 非常に優れている, ○: 優れている, △: やや劣る, ×: 劣る

表 6.2 は各研究が提供するモデルの結果のパフォーマンスを比較した表である. それぞれが異なるモデルや評価基準であるため, 直接的な比較は難しいが, 6 つの項目で評価を行う.

6 つの項目とその評価の尺度について述べる. 1 つ目は, 予測モデルを提供しているかである. 予測モデルを提供している場合は◎, 直接的な予測モデルではないが, 実装次第で予測モデルとできる場合に○や△を与える. 2 つ目は, 実験の標本数である. これは, 研究や実験の文脈により最適なものが異なるが, 大きいものがより一般性の高い実験結果を示していると言える. つまり, この観点では, Liao ら > Valiev ら > Samoladas ら > Rluri ら > 本研究 > Raja らとなる. 3 つ目は, モデルの精度である. ただし, Valiev ら, Samoladas らは各要因や変数が OSS の生存性とどのような関連があるかを説明するための研究であるため, モデルの精度は明言されていない. したがって, 残りの 4 つの研究のみ比較可能である. なお, それぞれの研究が事後解析であることを踏まえると, 本研究では MIAO の結果を用いるのがフェアである. モデルの精度評価に同じ指標を用いている本研究, Eluri ら, Raja らのみが直接比較できる. 4 つ目は, 一般性である. この項目では, それぞれの研究分野における結果の一般性を評価する. 結果の一般性は, データセットの標本数や, サンプリング方法, 特定のドメインへの偏りがないかなどで判断する. したがって, データセットの標本数が大きく, 偏りが最も少ないものを◎と評価する. 5 つ目は, 説明性である. これは, モデルや分析手法の説明性, 解釈性の高さを評価する. つまり, なぜそのような結果となったかを説明できる度合いと言える. 最後は実務可能性である. これは, 実務での利用可能性が高いかを評価する. 主な評価基準は, モデルの複雑性や, データセット構築の難易度が低いかである. つまり, 精度が高くとも, モデルの実装や, データセットの構築が大規模で困難である場合に, 評価は下がる.

表 6.2 の結果をまとめると, 本研究は中程度の予測精度を持ち, 説明性と実務可能性が優れているが, データセットの制限から一般性に疑問が残る結果となった. 最もバランスの良いものは Liao らのもので, シンプルなモデルでありながら, データセットも大きく, 一般性の高い結果となっている. モデル精度の観点で, 最も優秀なもの

はRajaらのものである．ただし, Rajaらのモデルは事後分析を前提に設計されているため, 予測モデルを提供しているわけではない．予測モデルを提供しているという観点では, 本研究, Liaoら, Eluriらに軍配が上がる．これらの研究ではデータセットを準備できれば, 論文記載のモデルを実装することで, 将来のOSSの生存性を予測することができる．特に, 本研究とLiaoらはデータセットの入手が比較的容易な点が評価できる．一方で, 予測モデルを提供していない研究では, 研究の説明性やモデルの精度が高いものの, データセットの取得が容易ではないというトレードオフがある．特に, Valievらはネットワーク分析のために大規模なデータセットが必要で, Rajaらは Source Forge中の開発者間のコミュニケーションやタスクの割当の記録などを取得する必要がある．

第7章　おわりに

先行研究では, 持続性の高い OSS プロジェクトが持つ特徴を主に内的要因を用いて明らかにした. また, 外的要因を扱う研究では, スポンサーシップの有無や GitHub スター, 依存関係ネットワークを用いて, それらと OSS の生存性がどのように関係するかを分析した. 本研究ではこれまでの視野を拡張し, OSS の競合関係という外的要因が OSS プロジェクトの生存性に与える影響を含むようにした.

具体的には, MIAO により複数の OSS 同士の経時的な競合関係を定量化することに成功し, 特定のクラスにおいて REV か nonREV であるかを高精度かつ偏りなく分類できることが分かった. これは, OSS 間の累積インパルス応答という特徴量が, OSS の競合関係を定量化するためのデータとして十分な説明性を持っていることを示唆する. RPM の精度は中程度にとどまり改善の余地が残るが, REV をスクリーニング的に予測するという意味では十分に利用可能な精度である.

今後の課題として残される議題は, MIAO については, データセットの拡張やコミット数以外のアクティビティ時系列データと REV の関連性のほか, 厳密な因果推論手法を導入するなどが考えられる. RPM については, SPM の予測精度を向上するための変数の発見や, SPM ではなく, 直接的に将来のコミット数の推移を予測する時系列予測モデルの導入などがある.

これらの課題に関して, アルゴリズムなどを変えずに, すぐに着手できるものに標本数の拡張がある. ただし, 本研究の難しい点は, REV を確定的に知ることが困難な点である. したがって, 慎重に事実確認を行い, 手動でデータセットを構築せざるを得ないため, 大規模な標本からランダムサンプリングするといったことができない. そのため, より効率的なデータセットの構築方法が見つかれば, 本研究の信頼性が格段に向上することになる.

コミット数以外のアクティビティ時系列データと REV の関連性を分析する際には, コミット数の推移をその他のアクティビティ時系列データに変更すれば良い. 例えば, 貢献者数の推移や, GitHub スターの推移, GitHub 上の PullRequest(PR) 数や Issue 数などを用いることが考えられる. これらを用いて, REV の分類精度が向上する場合は, REV の分類という文脈においてコミット数よりもその他の変数のほうが優れているといえる. この他にも, 直接的な REV の分類とはならないが, 各変数間の関係性を MIAO で定量化することもできる. 例えば, ひとつの OSS に関する貢献者数の推移や, GitHub スターの推移, GitHub 上の PR 数や Issue 数などをそれぞれ A_i として MIAO に入力すると, Issue 数が上昇したときの PR 数の変動などを分析することができる. この分析により, REV と nonREV の OSS とで, 各アクティビティの関係性に異なる特徴が現れるかなどを分析できる.

より厳密な因果推論手法の導入は, 当該 OSS が内的要因で衰退したのか, 市場要

因で衰退したのか, あるいは競合関係で衰退したのかといった, どのような原因で衰退したかを推測するためのものである. 現状, MIAO では REV により衰退したと考えられる OSS が用いられているが, 前述した通り REV により衰退したかどうかは定性的な判断に基づいている. このため, VAR-LiNGAM などの因果探索手法や, 合成コントロール法 (SCM) , CausalImpact などを用いた反実仮想シナリオの分析などを行い, OSS の衰退要因を厳密に推論する. これにより MIAO を適用可能な OSS を定量的に特定でき, MIAO の結果の信頼性を向上させることができる.

RPM では, SCE の予測を向上させるための変数の発見が重要となるが, SCE の点推定ではなく, 将来のアクティビティ時系列データを直接的に予測するというアプローチも考えられる. このような複雑な時系列の予測には, VAR のような線形モデルでなく, LSTM(Long short term memory) や状態空間モデルのような非線形モデルが役立つ可能性がある. このとき, 先に上げた因果推論の結果を用いることで, 市場が楽観的に成長するだろう, あるいは悲観的な見通しである, といったトレンドを予測モデルに組みこむことで, 翌年のコミット数の推移を様々なシナリオでシミュレーション可能となる. 各シナリオで MIAO スコアを計算するという行為を定期的に行い, 採用 OSS と競合 OSS の MIAO スコアに大きな非対称性が見られる場合に, REV の発生に備え当該 OSS のリプレイスの準備をするなどの意思決定が可能となる.

謝辞

本研究を進めるにあたって, 多くの方々に, ご支援, ご協力をいただきました. この場を借りて心から謝意を表します.

主指導教員である, 北陸先端科学技術大学院大学先端科学技術専攻科 青木利晃教授には, 本学で研究を行うにあたり, 研究の進め方, 論文の書き方, 学会への論文提出, 方法論そのものに関するアドバイスなど, 研究に関する包括的な知識を丁寧にご指導いただきました. 学術的な研究を行うのは本研究が初めてであり, 先生のご指導が私の研究スタイルの基礎を固めるものとなりました. 深く感謝いたします.

副指導教官である, 北陸先端科学技術大学院大学先端科学技術専攻科 冨田尭准教授には, 主に方法論的な部分で多数のご助言をいただきました. また, 国内学会発表にご同行くださり, 発表内容に関するご助言をいただけたことで, 本研究を客観的に見つめ直し, 改善する機会をいただけました. 心より感謝申し上げます.

副テーマ指導教員である, 北陸先端科学技術大学院大学先端科学技術専攻科 情報社会基盤研究センター 本郷研太准教授には, 副テーマ研究を遂行するにあたり, テーマの決定から関連研究や関連ソフトウェアの提供などを丁寧にご指導, ご助言いただきました. 副テーマ研究の題材である量子コンピューティング領域は, 主テーマ研究とは大きく離れた分野であるにも関わらず, 最後まで遂行できたことは, 先生のご助言の賜物でした. 心より感謝申し上げます.

同研究室の東京社会人コースの学生, 石川の足立氏には, 研究に関するご助言や発表練習にお付き合いいただくなど, 多大なるご支援とご協力をいただきました. 社会人コースではそれぞれの学生が着手する研究テーマが多様であるため, 活発な議論のものと非常に充実した 3 年間を過ごすことができました. 毎月 1 回のゼミがいつも楽しみでした. 心より感謝申し上げます.

付 録A　4章の実験結果

A.1　k=200の根拠となる実験

表 A.1: k 毎の SCE の近似精度の統計

k	mean	median	std	min	max
10	0.906	0.333	2.316	0.000	28.879
100	0.029	0.000	0.123	0.000	1.282
200	0.002	0.000	0.014	0.000	0.196
1000	0.000	0.000	0.000	0.000	0.000

$k = 200$ の根拠となる実験内容は, $k = 10000$ をベンチマークとし, $k = 10, k = 100, k = 200, k = 1000$ と, $k = 10000$ のときの AMS_{ij} の差の絶対値がどれほど異なるかを比較したものである. 標本数は各グループ毎に 6 つの AMS_{ij} があるので, $66 \times 6 = 396$ である.

この結果を見ると, k=10 と k=10000 のときの AMS_{ij} の差の平均は 0.906 もあり, 標準偏差も 2.316 と大きくばらつくことが分かる. これは k が小さい場合に, 平均的に AMS_{ij} が約 1 ポイント前後することを意味する. k=200 基準では AMS_{ij} の平均値は-0.626 であるため, 1.0 ポイントの変動はかなりのインパクトがあるといえる. k=100 としても平均的に 0.029 の変動があり, 最大 1.282 ポイント変動する. これは無視できない誤差である. 一方で, k=200, 1000 と k を大きくするにつれて k=10000 との差の平均は縮小し, 標準偏差も小さくなる. これを見ると, k=200 のときの各種統計は k=1000 と比べても十分に許容できる誤差であり, MIAO の結果に大きく影響しないと考えられる. したがって, 計算効率的と近似精度のバランスが良く取れている k=200 を選択した.

A.2 MIAO スコア表

表 A.2: MIAO スコア表 - 1/3

	Direction	AMS_{ij}
1_0	pytorch → tensorflow	-3.08
	chainer → tensorflow	-3.58
	tensorflow → pytorch	-0.12
	chainer → pytorch	-0.98
	tensorflow → chainer	-0.82
	pytorch → chainer	2.12
2_0	chainer → tensorflow	-4.31
	Theano → tensorflow	-0.79
	tensorflow → chainer	-1.41
	Theano → chainer	1.02
	tensorflow → Theano	0.29
	chainer → Theano	1.84
3_0	mesos → kubernetes	-2.65
	classicswarm → kubernetes	-20.12
	kubernetes → mesos	-1.61
	classicswarm → mesos	1.29
	kubernetes → classicswarm	-0.76
	mesos → classicswarm	1.65
4_0	framework → symfony	-0.67
	CodeIgniter → symfony	0.75
	symfony → framework	-1.18
	CodeIgniter → framework	1.19
	symfony → CodeIgniter	0.05
	framework → CodeIgniter	0.51
4_1	framework → symfony	0.70
	CodeIgniter → symfony	-1.12
	symfony → framework	-0.00
	CodeIgniter → framework	-0.01
	symfony → CodeIgniter	0.13
	framework → CodeIgniter	-0.24
5_0	symfony → cakephp	0.76
	zendframework → cakephp	0.17
	cakephp → symfony	-0.89
	zendframework → symfony	1.02
	cakephp → zendframework	2.18
	symfony → zendframework	2.19
5_1	symfony → cakephp	1.64
	zendframework → cakephp	0.72
	cakephp → symfony	-0.77
	zendframework → symfony	1.15
	cakephp → zendframework	5.69
	symfony → zendframework	4.37
6_0	intellij-community → vscode	-0.58
	atom → vscode	0.36
	vscode → intellij-community	-5.05
	atom → intellij-community	0.28
	vscode → atom	1.58
	intellij-community → atom	0.03
6_1	intellij-community → vscode	-0.43
	atom → vscode	1.76
	vscode → intellij-community	-4.32
	atom → intellij-community	2.75
	vscode → atom	0.91
	intellij-community → atom	0.07
7_0	framework → symfony	-0.01
	fuel → symfony	-2.73
	symfony → framework	-0.29
	fuel → framework	-1.76
	symfony → fuel	-0.00
	framework → fuel	0.01
8_0	mongo → couchdb	-1.92
	rethinkdb → couchdb	-0.95
	couchdb → mongo	-4.99
	rethinkdb → mongo	-4.40
	couchdb → rethinkdb	0.20
	mongo → rethinkdb	5.13
8_1	mongo → couchdb	-0.67
	rethinkdb → couchdb	-1.25
	couchdb → mongo	-0.13
	rethinkdb → mongo	1.11
	couchdb → rethinkdb	-1.96
	mongo → rethinkdb	3.93
9_0	jake → webpack	-0.20
	gulp → webpack	-0.37
	webpack → jake	0.01
	gulp → jake	-0.09
	webpack → gulp	0.35
	jake → gulp	0.50
10_0	foundation-sites → bootstrap	-1.35
	Semantic-UI → bootstrap	-2.08
	bootstrap → foundation-sites	-0.27
	Semantic-UI → foundation-sites	3.44
	bootstrap → Semantic-UI	0.79
	foundation-sites → Semantic-UI	1.92
11_0	sdk → TypeScript	-1.31
	coffeescript → TypeScript	-3.19
	TypeScript → sdk	-0.77
	coffeescript → sdk	-7.19
	TypeScript → coffeescript	0.07
	sdk → coffeescript	-0.09
12_0	ember.js → react	0.24
	angular.js → react	0.12
	react → ember.js	-0.64
	angular.js → ember.js	0.16
	react → angular.js	-0.00
	ember.js → angular.js	0.64
12_1	ember.js → react	0.37
	angular.js → react	-0.63
	react → ember.js	-1.50
	angular.js → ember.js	-3.37
	react → angular.js	0.51
	ember.js → angular.js	0.31
13_0	react → angular.js	-0.44
	knockout → angular.js	-0.64
	angular.js → react	0.17
	knockout → react	-0.89
	angular.js → knockout	0.10
	react → knockout	0.08
14_0	dojo → jquery	3.02
	yui3 → jquery	0.30
	jquery → dojo	-0.84
	yui3 → dojo	0.02
	jquery → yui3	7.82
	dojo → yui3	7.89
14_1	dojo → jquery	3.90
	yui3 → jquery	0.87
	jquery → dojo	-2.71
	yui3 → dojo	-0.68
	jquery → yui3	6.50
	dojo → yui3	-20.35
15_0	flutter → react-native	-4.68
	cordova-android → react-native	1.47
	react-native → flutter	-0.65
	cordova-android → flutter	-3.41
	react-native → cordova-android	-0.06
	flutter → cordova-android	0.34
16_0	drupal → WordPress	-2.19
	lenya → WordPress	1.58
	WordPress → drupal	-0.66
	lenya → drupal	0.89
	WordPress → lenya	-0.36
	drupal → lenya	3.87
16_1	drupal → WordPress	0.50
	lenya → WordPress	0.09
	WordPress → drupal	-0.65
	lenya → drupal	0.02
	WordPress → lenya	1.44
	drupal → lenya	0.08
17_0	typeorm → sequelize	-0.05
	node-orm2 → sequelize	0.45
	sequelize → typeorm	0.67
	node-orm2 → typeorm	-1.68
	sequelize → node-orm2	0.06
	typeorm → node-orm2	-0.11

表 A.3: MIAO スコア表 - 2/3

	Direction	AMS_{ij}
18_0	okhttp → httpcomponents-client	-0.42
	volley → httpcomponents-client	0.01
	httpcomponents-client → okhttp	0.00
	volley → okhttp	-0.04
	httpcomponents-client → volley	-1.09
	okhttp → volley	6.94
19_0	jest → mocha	0.06
	karma → mocha	-0.04
	mocha → jest	0.29
	karma → jest	-0.17
	mocha → karma	0.69
	jest → karma	-0.30
19_1	jest → mocha	0.08
	karma → mocha	-0.25
	mocha → jest	-0.36
	karma → jest	1.07
	mocha → karma	0.39
	jest → karma	0.72
20_0	memcached → redis	2.01
	riak_kv → redis	-0.37
	redis → memcached	-0.15
	riak_kv → memcached	-0.00
	redis → riak_kv	-0.01
	memcached → riak_kv	2.60
20_1	memcached → redis	-0.03
	riak_kv → redis	-0.61
	redis → memcached	-0.28
	riak_kv → memcached	0.03
	redis → riak_kv	1.57
	memcached → riak_kv	1.19
21_0	swift-package-manager → CocoaPods	-0.88
	Carthage → CocoaPods	0.14
	CocoaPods → swift-package-manager	-0.59
	Carthage → swift-package-manager	-0.29
	CocoaPods → Carthage	0.79
	swift-package-manager → Carthage	-0.34
22_0	node-sass → stylus	0.48
	less → stylus	-0.49
	stylus → node-sass	-0.98
	less → node-sass	-0.84
	stylus → less	-2.50
	node-sass → less	0.77
23_0	webpack → browserify	-2.16
	requirejs → browserify	-1.11
	browserify → webpack	0.00
	requirejs → webpack	-0.66
	browserify → requirejs	-0.03
	webpack → requirejs	0.11

	Direction	AMS_{ij}
24_0	react → angular.js	0.24
	backbone → angular.js	-1.26
	angular.js → react	0.72
	backbone → react	0.53
	angular.js → backbone	-0.68
	react → backbone	-0.94
25_0	rails → django	-0.03
	framework → django	0.27
	django → rails	-1.37
	framework → rails	0.36
	django → framework	1.30
	rails → framework	-0.26
25_1	rails → django	-0.06
	framework → django	-0.57
	django → rails	1.69
	framework → rails	-1.99
	django → framework	-2.73
	rails → framework	-1.24
25_2	rails → django	0.10
	framework → django	-0.34
	django → rails	1.07
	framework → rails	-1.70
	django → framework	-0.60
	rails → framework	0.50
26_0	postgres → mysql-server	-1.72
	firebird → mysql-server	-3.41
	mysql-server → postgres	1.46
	firebird → postgres	0.96
	mysql-server → firebird	-16.44
	postgres → firebird	-3.93
26_1	postgres → mysql-server	-11.06
	firebird → mysql-server	4.16
	mysql-server → postgres	-0.18
	firebird → postgres	-0.02
	mysql-server → firebird	0.95
	postgres → firebird	-1.96
26_2	postgres → mysql-server	-8.90
	firebird → mysql-server	-0.12
	mysql-server → postgres	0.32
	firebird → postgres	0.09
	mysql-server → firebird	0.19
	postgres → firebird	3.24
26_3	postgres → mysql-server	-5.72
	firebird → mysql-server	4.35
	mysql-server → postgres	-0.01
	firebird → postgres	-4.41
	mysql-server → firebird	-0.23
	postgres → firebird	-0.76

	Direction	AMS_{ij}
26_4	postgres → mysql-server	-6.97
	firebird → mysql-server	-7.36
	mysql-server → postgres	0.16
	firebird → postgres	-0.58
	mysql-server → firebird	0.13
	postgres → firebird	-0.05
26_5	postgres → mysql-server	-2.73
	firebird → mysql-server	1.33
	mysql-server → postgres	0.17
	firebird → postgres	0.35
	mysql-server → firebird	0.00
	postgres → firebird	-0.81
27_0	ruby → cpython	-0.76
	node → cpython	-1.02
	cpython → ruby	-0.19
	node → ruby	-1.26
	cpython → node	0.03
	ruby → node	-1.00
27_1	ruby → cpython	0.79
	node → cpython	-1.85
	cpython → ruby	0.15
	node → ruby	-1.30
	cpython → node	0.54
	ruby → node	0.82
27_2	ruby → cpython	-0.85
	node → cpython	-0.39
	cpython → ruby	-2.62
	node → ruby	-1.83
	cpython → node	0.31
	ruby → node	0.26
27_3	ruby → cpython	-0.24
	node → cpython	0.10
	cpython → ruby	0.14
	node → ruby	1.79
	cpython → node	0.30
	ruby → node	0.19
28_0	httpd → nginx	-0.38
	h2o → nginx	0.04
	nginx → httpd	-0.98
	h2o → httpd	0.30
	nginx → h2o	3.28
	httpd → h2o	0.52
28_1	httpd → nginx	-1.44
	h2o → nginx	-0.25
	nginx → httpd	0.30
	h2o → httpd	0.60
	nginx → h2o	-2.13
	httpd → h2o	0.27

表 A.4: MIAO スコア表 - 3/3

	Direction	AMS_{ij}
28_2	httpd → nginx	-1.22
	h2o → nginx	0.07
	nginx → httpd	-0.46
	h2o → httpd	0.04
	nginx → h2o	-2.60
	httpd → h2o	1.24
29_0	struts → playframework	-1.64
	spring-boot → playframework	0.62
	playframework → struts	-0.36
	spring-boot → struts	0.05
	playframework → spring-boot	-0.53
	struts → spring-boot	-1.86
29_1	struts → playframework	0.24
	spring-boot → playframework	0.15
	playframework → struts	0.44
	spring-boot → struts	0.06
	playframework → spring-boot	6.95
	struts → spring-boot	4.52
29_2	struts → playframework	-0.97
	spring-boot → playframework	0.03
	playframework → struts	-1.68
	spring-boot → struts	0.05
	playframework → spring-boot	-4.45
	struts → spring-boot	3.09
30_0	angular.js → jquery	-0.11
	react → jquery	-0.25
	jquery → angular.js	1.76
	react → angular.js	0.34
	jquery → react	2.30
	angular.js → react	0.12
30_1	angular.js → jquery	-0.81
	react → jquery	0.02
	jquery → angular.js	-1.31
	react → angular.js	0.39
	jquery → react	-6.54
	angular.js → react	11.63
30_2	angular.js → jquery	0.13
	react → jquery	-0.28
	jquery → angular.js	-0.01
	react → angular.js	0.32
	jquery → react	-13.20
	angular.js → react	10.39
31_0	flutter → titanium-sdk	-4.04
	react-native → titanium-sdk	0.49
	titanium-sdk → flutter	-3.02
	react-native → flutter	0.88
	titanium-sdk → react-native	-0.03
	flutter → react-native	-5.13

	Direction	AMS_{ij}
31_1	flutter → titanium-sdk	0.09
	react-native → titanium-sdk	0.00
	titanium-sdk → flutter	-0.21
	react-native → flutter	-0.14
	titanium-sdk → react-native	1.15
	flutter → react-native	-1.59
32_0	cassandra → couchdb	0.54
	mongo → couchdb	-1.18
	couchdb → cassandra	-0.44
	mongo → cassandra	-0.13
	couchdb → mongo	-3.51
	cassandra → mongo	-3.57
32_1	cassandra → couchdb	-0.78
	mongo → couchdb	-1.15
	couchdb → cassandra	-2.54
	mongo → cassandra	-1.31
	couchdb → mongo	-0.91
	cassandra → mongo	-0.87
32_2	cassandra → couchdb	-0.17
	mongo → couchdb	0.43
	couchdb → cassandra	-1.65
	mongo → cassandra	-0.17
	couchdb → mongo	-1.39
	cassandra → mongo	-1.45
32_3	cassandra → couchdb	0.22
	mongo → couchdb	0.91
	couchdb → cassandra	-0.47
	mongo → cassandra	-0.91
	couchdb → mongo	-1.24
	cassandra → mongo	-10.49
33_0	CodeIgniter → symfony	-0.44
	framework → symfony	0.77
	symfony → CodeIgniter	0.14
	framework → CodeIgniter	-0.60
	symfony → framework	1.05
	CodeIgniter → framework	-7.44
33_1	CodeIgniter → symfony	-1.16
	framework → symfony	-1.27
	symfony → CodeIgniter	0.06
	framework → CodeIgniter	0.21
	symfony → framework	-0.32
	CodeIgniter → framework	-1.10
34_0	mesos → kubernetes	-5.27
	nomad → kubernetes	-2.68
	kubernetes → mesos	-0.44
	nomad → mesos	-0.55
	kubernetes → nomad	-1.05
	mesos → nomad	0.44

	Direction	AMS_{ij}
34_1	mesos → kubernetes	-46.88
	nomad → kubernetes	-1.52
	kubernetes → mesos	-0.24
	nomad → mesos	0.20
	kubernetes → nomad	0.33
	mesos → nomad	-14.63
35_0	nuxt → next.js	-6.46
	remix → next.js	1.08
	next.js → nuxt	-2.80
	remix → nuxt	-0.26
	next.js → remix	-0.91
	nuxt → remix	-2.06

A.3　決定木モデルの予測結果

表 A.5: 決定木モデルの分類結果と真値の比較表

Group	Actual	Predicted	Group	Actual	Predicted
1_0	1	1	25_0	0	1
2_0	1	1	25_1	0	0
3_0	1	1	25_2	0	0
4_0	1	1	26_0	0	0
4_1	1	1	26_1	0	0
5_0	1	1	26_2	0	0
5_1	1	1	26_3	0	0
6_0	1	1	26_4	0	0
6_1	1	1	26_5	0	0
7_0	1	1	27_0	0	0
8_0	1	1	27_1	0	0
8_1	1	1	27_2	0	1
9_0	1	1	27_3	0	0
10_0	1	1	28_0	0	1
11_0	1	1	28_1	0	0
12_0	1	1	28_2	0	1
12_1	1	1	29_0	0	0
13_0	1	0	29_1	0	0
14_0	1	1	29_2	0	0
14_1	1	0	30_0	0	0
15_0	1	1	30_1	0	0
16_0	1	1	30_2	0	0
16_1	1	1	31_0	0	0
17_0	1	0	31_1	0	0
18_0	1	1	32_0	0	0
19_0	1	0	32_1	0	0
19_1	1	1	32_2	0	0
20_0	1	1	32_3	0	0
20_1	1	1	33_0	0	0
21_0	1	1	33_1	0	0
22_0	1	0	34_0	0	1
23_0	1	1	34_1	0	0
24_0	0	0	35_0	0	0

付 録B　5章の実験結果

B.1　MIAOスコア表

表 B.1: MIAO スコア表 - 1/3

	Direction	AMS_{ij}
1_0	pytorch → tensorflow	-3.67
	chainer → tensorflow	-4.17
	tensorflow → pytorch	-0.54
	chainer → pytorch	-1.40
	tensorflow → chainer	-1.17
	pytorch → chainer	1.76
2_0	chainer → tensorflow	-4.74
	Theano → tensorflow	-1.23
	tensorflow → chainer	-1.72
	Theano → chainer	0.71
	tensorflow → Theano	-0.02
	chainer → Theano	1.52
3_0	mesos → kubernetes	-3.45
	classicswarm → kubernetes	-20.91
	kubernetes → mesos	-2.14
	classicswarm → mesos	0.76
	kubernetes → classicswarm	-1.22
	mesos → classicswarm	1.20
4_0	framework → symfony	-0.85
	CodeIgniter → symfony	0.58
	symfony → framework	-1.39
	CodeIgniter → framework	0.98
	symfony → CodeIgniter	-0.12
	framework → CodeIgniter	0.34
4_1	framework → symfony	0.82
	CodeIgniter → symfony	-1.00
	symfony → framework	0.11
	CodeIgniter → framework	0.10
	symfony → CodeIgniter	0.22
	framework → CodeIgniter	-0.15
5_0	symfony → cakephp	0.86
	zendframework → cakephp	0.27
	cakephp → symfony	-0.76
	zendframework → symfony	1.15
	cakephp → zendframework	2.30
	symfony → zendframework	2.31
5_1	symfony → cakephp	2.47
	zendframework → cakephp	1.55
	cakephp → symfony	-0.04
	zendframework → symfony	1.88
	cakephp → zendframework	6.49
	symfony → zendframework	5.17
6_0	intellij-community → vscode	-1.06
	atom → vscode	-0.12
	vscode → intellij-community	-5.79
	atom → intellij-community	-0.47
	vscode → atom	1.16
	intellij-community → atom	-0.38
6_1	intellij-community → vscode	-0.79
	atom → vscode	1.40
	vscode → intellij-community	-4.82
	atom → intellij-community	2.25
	vscode → atom	0.43
	intellij-community → atom	-0.41
7_0	framework → symfony	-0.09
	fuel → symfony	-2.81
	symfony → framework	-0.39
	fuel → framework	-1.86
	symfony → fuel	-0.09
	framework → fuel	-0.08
8_0	mongo → couchdb	-2.00
	rethinkdb → couchdb	-1.03
	couchdb → mongo	-4.81
	rethinkdb → mongo	-4.22
	couchdb → rethinkdb	0.36
	mongo → rethinkdb	5.29
8_1	mongo → couchdb	-0.95
	rethinkdb → couchdb	-1.52
	couchdb → mongo	-0.30
	rethinkdb → mongo	0.94
	couchdb → rethinkdb	-2.31
	mongo → rethinkdb	3.58
9_0	jake → webpack	-0.17
	gulp → webpack	-0.34
	webpack → jake	0.07
	gulp → jake	-0.03
	webpack → gulp	0.43
	jake → gulp	0.58
10_0	foundation-sites → bootstrap	-1.66
	Semantic-UI → bootstrap	-2.38
	bootstrap → foundation-sites	-0.55
	Semantic-UI → foundation-sites	3.16
	bootstrap → Semantic-UI	0.67
	foundation-sites → Semantic-UI	1.80
11_0	sdk → TypeScript	-1.59
	coffeescript → TypeScript	-3.47
	TypeScript → sdk	-1.19
	coffeescript → sdk	-7.62
	TypeScript → coffeescript	-0.21
	sdk → coffeescript	-0.38
12_0	ember.js → react	0.19
	angular.js → react	0.07
	react → ember.js	-0.71
	angular.js → ember.js	0.09
	react → angular.js	-0.03
	ember.js → angular.js	0.61
12_1	ember.js → react	0.30
	angular.js → react	-0.70
	react → ember.js	-1.68
	angular.js → ember.js	-3.55
	react → angular.js	0.31
	ember.js → angular.js	0.12
13_0	react → angular.js	-0.57
	knockout → angular.js	-0.76
	angular.js → react	0.12
	knockout → react	-0.94
	angular.js → knockout	0.11
	react → knockout	0.09
14_0	dojo → jquery	4.08
	yui3 → jquery	1.35
	jquery → dojo	-0.00
	yui3 → dojo	0.86
	jquery → yui3	8.88
	dojo → yui3	8.95
14_1	dojo → jquery	4.44
	yui3 → jquery	1.41
	jquery → dojo	-2.50
	yui3 → dojo	-0.47
	jquery → yui3	7.07
	dojo → yui3	-19.78
15_0	flutter → react-native	-5.55
	cordova-android → react-native	0.61
	react-native → flutter	-1.15
	cordova-android → flutter	-3.91
	react-native → cordova-android	-0.52
	flutter → cordova-android	-0.12
16_0	drupal → WordPress	-2.39
	lenya → WordPress	1.38
	WordPress → drupal	-0.90
	lenya → drupal	0.65
	WordPress → lenya	-0.50
	drupal → lenya	3.73
16_1	drupal → WordPress	0.60
	lenya → WordPress	0.19
	WordPress → drupal	-0.69
	lenya → drupal	-0.02
	WordPress → lenya	1.43
	drupal → lenya	0.07
17_0	typeorm → sequelize	-0.02
	node-orm2 → sequelize	0.48
	sequelize → typeorm	0.80
	node-orm2 → typeorm	-1.55
	sequelize → node-orm2	0.11
	typeorm → node-orm2	-0.05

表 B.2: MIAO スコア表 - 2/3

	Direction	AMS_{ij}
18_0	okhttp → httpcomponents-client	-0.39
	volley → httpcomponents-client	0.04
	httpcomponents-client → okhttp	0.12
	volley → okhttp	0.08
	httpcomponents-client → volley	-1.30
	okhttp → volley	6.73
19_0	jest → mocha	0.08
	karma → mocha	-0.02
	mocha → jest	0.32
	karma → jest	-0.14
	mocha → karma	0.73
	jest → karma	-0.26
19_1	jest → mocha	0.09
	karma → mocha	-0.24
	mocha → jest	-0.35
	karma → jest	1.09
	mocha → karma	0.40
	jest → karma	0.74
20_0	memcached → redis	2.18
	riak_kv → redis	-0.19
	redis → memcached	0.03
	riak_kv → memcached	0.19
	redis → riak_kv	0.23
	memcached → riak_kv	2.85
20_1	memcached → redis	0.10
	riak_kv → redis	-0.47
	redis → memcached	-0.15
	riak_kv → memcached	0.15
	redis → riak_kv	1.65
	memcached → riak_kv	1.28
21_0	swift-package-manager → CocoaPods	-1.04
	Carthage → CocoaPods	-0.03
	CocoaPods → swift-package-manager	-0.70
	Carthage → swift-package-manager	-0.40
	CocoaPods → Carthage	0.72
	swift-package-manager → Carthage	-0.40
22_0	node-sass → stylus	0.41
	less → stylus	-0.56
	stylus → node-sass	-1.10
	less → node-sass	-0.96
	stylus → less	-2.77
	node-sass → less	0.49
23_0	webpack → browserify	-2.68
	requirejs → browserify	-1.63
	browserify → webpack	-0.35
	requirejs → webpack	-1.02
	browserify → requirejs	-0.36
	webpack → requirejs	-0.22

	Direction	AMS_{ij}
24_0	react → angular.js	0.38
	backbone → angular.js	-1.12
	angular.js → react	0.96
	backbone → react	0.77
	angular.js → backbone	-0.58
	react → backbone	-0.83
25_0	rails → django	-0.03
	framework → django	0.27
	django → rails	-1.38
	framework → rails	0.35
	django → framework	1.37
	rails → framework	-0.19
25_1	rails → django	0.03
	framework → django	-0.48
	django → rails	1.57
	framework → rails	-2.11
	django → framework	-2.93
	rails → framework	-1.43
25_2	rails → django	-0.02
	framework → django	-0.45
	django → rails	0.90
	framework → rails	-1.87
	django → framework	-0.88
	rails → framework	0.23
26_0	postgres → mysql-server	-3.43
	firebird → mysql-server	-5.13
	mysql-server → postgres	0.30
	firebird → postgres	-0.19
	mysql-server → firebird	-18.44
	postgres → firebird	-5.93
26_1	postgres → mysql-server	-12.78
	firebird → mysql-server	2.44
	mysql-server → postgres	-1.48
	firebird → postgres	-1.32
	mysql-server → firebird	0.35
	postgres → firebird	-2.56
26_2	postgres → mysql-server	-9.79
	firebird → mysql-server	-1.00
	mysql-server → postgres	-0.37
	firebird → postgres	-0.61
	mysql-server → firebird	-0.59
	postgres → firebird	2.46
26_3	postgres → mysql-server	-6.20
	firebird → mysql-server	3.86
	mysql-server → postgres	-0.30
	firebird → postgres	-4.70
	mysql-server → firebird	-0.46
	postgres → firebird	-0.99

	Direction	AMS_{ij}
26_4	postgres → mysql-server	-7.66
	firebird → mysql-server	-8.05
	mysql-server → postgres	-0.58
	firebird → postgres	-1.32
	mysql-server → firebird	-0.25
	postgres → firebird	-0.44
26_5	postgres → mysql-server	-2.92
	firebird → mysql-server	1.14
	mysql-server → postgres	0.08
	firebird → postgres	0.27
	mysql-server → firebird	-0.19
	postgres → firebird	-1.00
27_0	ruby → cpython	-0.85
	node → cpython	-1.11
	cpython → ruby	-0.36
	node → ruby	-1.43
	cpython → node	-0.09
	ruby → node	-1.12
27_1	ruby → cpython	0.85
	node → cpython	-1.79
	cpython → ruby	0.23
	node → ruby	-1.23
	cpython → node	0.59
	ruby → node	0.87
27_2	ruby → cpython	-1.09
	node → cpython	-0.63
	cpython → ruby	-2.86
	node → ruby	-2.07
	cpython → node	0.11
	ruby → node	0.06
27_3	ruby → cpython	-0.45
	node → cpython	-0.11
	cpython → ruby	-0.14
	node → ruby	1.50
	cpython → node	0.09
	ruby → node	-0.01
28_0	httpd → nginx	-0.21
	h2o → nginx	0.21
	nginx → httpd	-0.85
	h2o → httpd	0.43
	nginx → h2o	3.24
	httpd → h2o	0.48
28_1	httpd → nginx	-1.79
	h2o → nginx	-0.59
	nginx → httpd	-0.09
	h2o → httpd	0.21
	nginx → h2o	-2.58
	httpd → h2o	-0.17

表 B.3: MIAO スコア表 - 3/3

	Direction	AMS_{ij}
28_2	httpd → nginx	-1.59
	h2o → nginx	-0.30
	nginx → httpd	-0.81
	h2o → httpd	-0.31
	nginx → h2o	-3.01
	httpd → h2o	0.83
29_0	struts → playframework	-1.82
	spring-boot → playframework	0.43
	playframework → struts	-0.53
	spring-boot → struts	-0.12
	playframework → spring-boot	-0.69
	struts → spring-boot	-2.02
29_1	struts → playframework	0.75
	spring-boot → playframework	0.66
	playframework → struts	0.94
	spring-boot → struts	0.57
	playframework → spring-boot	7.58
	struts → spring-boot	5.16
29_2	struts → playframework	-1.39
	spring-boot → playframework	-0.39
	playframework → struts	-1.92
	spring-boot → struts	-0.19
	playframework → spring-boot	-4.75
	struts → spring-boot	2.80
30_0	angular.js → jquery	0.10
	react → jquery	-0.04
	jquery → angular.js	1.98
	react → angular.js	0.56
	jquery → react	2.53
	angular.js → react	0.35
30_1	angular.js → jquery	-1.11
	react → jquery	-0.29
	jquery → angular.js	-1.60
	react → angular.js	0.09
	jquery → react	-7.10
	angular.js → react	11.06
30_2	angular.js → jquery	-0.69
	react → jquery	-1.11
	jquery → angular.js	-0.82
	react → angular.js	-0.50
	jquery → react	-14.10
	angular.js → react	9.50
31_0	flutter → titanium-sdk	-4.57
	react-native → titanium-sdk	-0.05
	titanium-sdk → flutter	-3.26
	react-native → flutter	0.64
	titanium-sdk → react-native	-0.49
	flutter → react-native	-5.58

	Direction	AMS_{ij}
31_1	flutter → titanium-sdk	-0.02
	react-native → titanium-sdk	-0.11
	titanium-sdk → flutter	-0.40
	react-native → flutter	-0.33
	titanium-sdk → react-native	0.97
	flutter → react-native	-1.76
32_0	cassandra → couchdb	0.38
	mongo → couchdb	-1.34
	couchdb → cassandra	-0.63
	mongo → cassandra	-0.32
	couchdb → mongo	-3.85
	cassandra → mongo	-3.90
32_1	cassandra → couchdb	-1.04
	mongo → couchdb	-1.41
	couchdb → cassandra	-2.93
	mongo → cassandra	-1.70
	couchdb → mongo	-1.24
	cassandra → mongo	-1.20
32_2	cassandra → couchdb	-0.22
	mongo → couchdb	0.38
	couchdb → cassandra	-1.90
	mongo → cassandra	-0.42
	couchdb → mongo	-1.52
	cassandra → mongo	-1.58
32_3	cassandra → couchdb	-0.08
	mongo → couchdb	0.61
	couchdb → cassandra	-0.75
	mongo → cassandra	-1.20
	couchdb → mongo	-1.46
	cassandra → mongo	-10.71
33_0	CodeIgniter → symfony	-0.54
	framework → symfony	0.67
	symfony → CodeIgniter	0.14
	framework → CodeIgniter	-0.60
	symfony → framework	1.04
	CodeIgniter → framework	-7.45
33_1	CodeIgniter → symfony	-1.25
	framework → symfony	-1.36
	symfony → CodeIgniter	0.07
	framework → CodeIgniter	0.22
	symfony → framework	-0.58
	CodeIgniter → framework	-1.35
34_0	mesos → kubernetes	-6.14
	nomad → kubernetes	-3.55
	kubernetes → mesos	-1.03
	nomad → mesos	-1.13
	kubernetes → nomad	-1.62
	mesos → nomad	-0.13

	Direction	AMS_{ij}
34_1	mesos → kubernetes	-43.50
	nomad → kubernetes	-5.99
	kubernetes → mesos	-4.53
	nomad → mesos	-4.09
	kubernetes → nomad	-1.50
	mesos → nomad	-15.33
35_0	nuxt → next.js	-7.93
	remix → next.js	-0.39
	next.js → nuxt	-4.34
	remix → nuxt	-1.81
	next.js → remix	-2.11
	nuxt → remix	-3.26

B.2 RDの予測結果

表 B.4: RD の分類結果と真値の比較表

Group	Actual	Predicted	Group	Actual	Predicted
1_0	1	1	25_0	0	1
2_0	1	0	25_1	0	0
3_0	1	1	25_2	0	1
4_0	1	1	26_0	0	0
4_1	1	1	26_1	0	0
5_0	1	1	26_2	0	1
5_1	1	1	26_3	0	0
6_0	1	1	26_4	0	1
6_1	1	1	26_5	0	0
7_0	1	1	27_0	0	0
8_0	1	1	27_1	0	1
8_1	1	1	27_2	0	1
9_0	1	1	27_3	0	1
10_0	1	1	28_0	0	1
11_0	1	0	28_1	0	1
12_0	1	1	28_2	0	1
12_1	1	1	29_0	0	0
13_0	1	1	29_1	0	1
14_0	1	1	29_2	0	0
14_1	1	0	30_0	0	1
15_0	1	1	30_1	0	0
16_0	1	1	30_2	0	0
16_1	1	1	31_0	0	0
17_0	1	0	31_1	0	0
18_0	1	1	32_0	0	0
19_0	1	1	32_1	0	0
19_1	1	1	32_2	0	0
20_0	1	1	32_3	0	0
20_1	1	1	33_0	0	0
21_0	1	1	33_1	0	0
22_0	1	0	34_0	0	1
23_0	1	1	34_1	0	0
24_0	0	0	35_0	0	0

参考文献

[1] Open UK. State of open phase two, july 2021. [Accessed: 25-Sep-2024].

[2] Synopsis. Open source security and risk analysis report, 2024. [Accessed: 25-Sep-2024].

[3] Jymit Khondhu, Andrea Capiluppi, and Klaas-Jan Stol. Is it all lost? a study of inactive open source projects. 06 2013.

[4] Jailton Coelho and Marco Tulio Valente. Why modern open source projects fail. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, aug 2017.

[5] Uzma Raja and Marietta J. Tretter. Defining and evaluating a measure of open source project survivability. *IEEE Transactions on Software Engineering*, 38(1):163–174, 2012.

[6] Ioannis Samoladas, Lefteris Angelis, and Ioannis Stamelos. Survival analysis on the duration of open source projects. *Information and Software Technology*, 52(9):902–922, 2010.

[7] Zhifang Liao, Benhong Zhao, Shengzong Liu, Haozhi Jin, Dayu He, Liu Yang, Yan Zhang, and Jinsong Wu. A prediction model of the project life-span in open source software ecosystem. *Mob. Netw. Appl.*, 24(4):1382–1391, aug 2019.

[8] Lutz Kilian. Not all oil price shocks are alike: Disentangling demand and supply shocks in the crude oil market. *American Economic Review*, 99(3):1053–69, June 2009.

[9] Koichiro Kamada and Tomohiro Sugo. Extracting the effects of monetary policy under the zero lower bound constraint [seisakukinri zero seiyakuka ni okeru kinyukouka no tyuusyutsu], July 2006. [Accessed: 25-Sep-2024].

[10] Mohammad Y. Allaho and Wang-Chien Lee. Analyzing the social ties and structure of contributors in open source software community. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 56–60, 2013.

[11] Jing Wang. Survival factors for free open source software projects: A multi-stage perspective. *European Management Journal*, 30(4):352–371, 2012.

[12] Marat Valiev, Bogdan Vasilescu, and James Herbsleb. Ecosystem-level determinants of sustained activity in open-source projects: a case study of the pypi ecosystem. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2018, page 644–655, New York, NY, USA, 2018. Association for Computing Machinery.

[13] Nicolas Ducheneaut. Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work (CSCW)*, 14:323–368, 2005.

[14] Noni De, A Ganzaroli, and L Orsi. The governance of open source software communities: An exploratory analysis. *Journal of Business Systems Governance & Ethics*, 6(1), apr 2011.

[15] Yan Li, Chuan-Hoo Tan, and Hock-Hai Teo. Leadership characteristics and developers ' motivation in open source software development. *Information & Management*, 49(5):257–267, 2012.

[16] Javier Luis Cánovas Izquierdo and Jordi Cabot. The role of foundations in open source projects. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, pages 3–12, 2018.

[17] Linus Dahlander and Mats G. Magnusson. Relationships between open source software companies and communities: Observations from nordic firms. *Research Policy*, 34(4):481–493, 2005.

[18] Yoshitaka Kuwata and Hiroshi Miura. A study on growth model of oss projects to estimate the stage of lifecycle. *Procedia Computer Science*, 60:1004–1013, 2015. Knowledge-Based and Intelligent Information & Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings.

[19] Vijaya Kumar Eluri, Shahram Sarkani, and Thomas Mazzuchi. Open source software survivability prediction using multi layer perceptron. In Frederick Harris, Sergiu Dascalu, Sharad Sharma, and Rui Wu, editors, *Proceedings of 28th International Conference on Software Engineering and Data Engineering*, volume 64 of *EPiC Series in Computing*, pages 148–157. EasyChair, 2019.

[20] Siyuan Jin, Ziyuan Li, Bichao Chen, Bing Zhu, and Yong Xia. Software code quality measurement: Implications from metric distributions. In *2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS)*, pages 488–496, 2023.

[21] Xiaozhou Li, Sergio Moreschini, Zheying Zhang, and Davide Taibi. Exploring factors and metrics to select open source software components for integration: An empirical study. *Journal of Systems and Software*, 188:111255, 2022.

[22] Redis. Redis.

[23] AWS Open Source Blog. Why aws supports valkey.

[24] Yuhang Zhao, Ruigang Liang, Xiang Chen, and Jing Zou. Evaluation indicators for open-source software: a review. *Cybersecurity*, 4:20, 06 2021.

[25] Jing Chen, Christian Peukert, and Imke Reimers. Incentivizing innovation in open source: Evidence from the github sponsors program. Working Paper 31668, National Bureau of Economic Research, Aug 2023.

[26] Sandeep Krishnamurthy, Shaosong Ou, and Arvind K. Tripathi. Acceptance of monetary rewards in open source software development. *Research Policy*, 43(4):632–644, 2014.

[27] Arzoo Atiq and Arvind Tripathi. Impact of financial benefits on open source software sustainability. 12 2016.

[28] Ravi Sen. A strategic analysis of competition between open source and proprietary software. *J. of Management Information Systems*, 24:233–257, 07 2007.

[29] Hudson Borges and Marco Tulio Valente. What ' s in a github star? understanding repository starring practices in a social coding platform. *Journal of Systems and Software*, 146:112–129, 2018.

[30] David A. Dickey and Wayne A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366):427–431, 1979.

[31] Marcos López De Prado. *Advances in Financial Machine Learning*. Wiley, Feb 2021.

[32] G. E. P. Box and David A. Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, 65(332):1509–1526, 1970.

[33] G. M. Ljung and G. E. P. Box. On a measure of lack of fit in time series models. *Biometrika*, 65(2):297–303, 1978.

[34] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, mar 1990. ProQuest document ID: 1266805989.

[35] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[36] E. J. Hannan and B. G. Quinn. The determination of the order of an autoregression. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):190–195, 1979.

[37] Hiroshi Murao. *Learning VAR Empirical Analysis with R: From Fundamentals of Time Series Analysis to Forecasting [R de Manabu VAR Zissyoubunseki: Zikeiretsubunseki no Kiso kara Yosoku made].* Ohmsha, Dec 2019. (in Japanese).

[38] Donald E. Farrar and Robert R. Glauber. Multicollinearity in regression analysis: The problem revisited. *The Review of Economics and Statistics*, 49(1):92–107, 1967.

[39] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[40] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. An in-depth study of the promises and perils of mining github. *Empirical Softw. Engg.*, 21(5):2035–2071, October 2016.

[41] James D. Hamilton. *Time Series Analysis*. Princeton University Press, Jan 1994.

[42] GitHub Docs. Archiving repositories. `https://docs.github.com/en/repositories/archiving-a-github-repository/archiving-repositories`. Accessed: 2025-01-15.

[43] Stergios Fotopoulos Nicholas Evangelopoulos, Anna Sidorova and Indushobha Chengalur-Smith. Determining process death based on censored activity data. *Communications in Statistics - Simulation and Computation*, 37(8):1647–1662, 2008.

[44] Preferred Networks. Preferred networks migrates its deep learning research platform to pytorch, Dec 2019. [Accessed: 25-Sep-2024].

[45] Pascal Lamblin. Mila and the future of theano, 2017. Original URL not available. Archived version accessed on 2024-09-25.

[46] CodeIgniter. Welcome to codeigniter. [Accessed: 25-Sep-2024].

[47] Laravel. Laravel - the php framework for web artisans. [Accessed: 25-Sep-2024].

[48] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, page 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[49] Scikit learn developers. sklearn.tree.decisiontreeclassifier. `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`. Accessed: 2025-01-15.

[50] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50 – 60, 1947.

[51] Tatsuyoshi Okimoto. *Econometric Time Series Analysis of Economic and Financial Data [Keizai Fainansu deta no Keiryouzikeiretubunseki]*. Asakura Shoten, Feb 2023. (in Japanese).

[52] Kenshiro Ninomiya and Masaaki Tokuda. *The Economics of Financial Structural Changes and Instability: Theory and Empirical Evidence[Kinyuukouzou no Henka to Fuanteisei no keizaigaku Riron to zissyou]*. Kinyuuzaiseijijyou Kenkyuukai, 2024. (in Japanese).