Kaggle カーネルを参照した機械学習アルゴリズムの選択/適用パターンの抽出と評価

晦日 慶太
日本電気株式会社
製造・装置業システム開発本部

大内 一哲
NEC ソリューションイノベータ
品質・プロセス統括本部

岡留 有哉
株式会社日立製作所
知能情報研究部

神崎 元
株式会社日立製作所
コネクティビティ研究部

土屋 俊雄
日本電気株式会社
第一官公ソリューション事業部

松岡 賢
東芝デジタルソリューションズ
生産技術センター

吉田 和樹
国立情報学研究所
GRACE センター

概要

近年、 IoT デバイスの普及により取得可能なデータが増大し、 データを利活用したビジネスが拡大している。 データを有効に活用するためには多くのノウハウが必要であり、そのようなノウハウを保有するエンジニアの需要が高まり、人材が不足する状況にある。 本研究では、 機械学習アルゴリズムの選択/適用パターンを抽出し、 スキルが十分でない人材でも適切なアルゴリズムの選択と適用を容易に実現できるようにすることを目指す。 実際、抽出したパターンを用いることでアルゴリズム適用に要する時間を約 77%削減できることが確認できた。 これにより、パターンを用意しておくことで、効率的なアルゴリズム適用が可能になる見通しを得た。

CCS コンセプト

• Software and its Engineering

キーワード

•Machine Learning •Kaggle

1. はじめに

近年、 IoTデバイスの普及により、 工場における機器稼働状況データから、 個人のバイタル情報まで多くのデータを取得・保存できるようになっている。そのため、 これらの多様なデータを活用することで、 データからの価値創造によるビジネスが拡大している。データを活用するためにはデータサイエンティストなどのデジタル人材も不可欠であり、 これらの人材の需要も急増している。しかしながら、 需要の拡大に対し人材の供給は追いついておらず、 人手不足が課題となっている[1]。

データを価値に結びつけるためには、 データの特性およびビジネスの目的に応じて適切にデータを扱う必要がある。そのような専門知識を獲得するためには、 実際にデータを扱うことで得られる知見や、 数理的な知識が不可欠である。しかしながら、 既存人材のスキルアップやスキルチェンジにより機械学習のスキルを持つエンジニアを増やすことにも多大な時間的コストがかかる。

機械学習の技術を習得するために時間的コストがかかる理由として、 数理的基礎知識の習得だけでなく、 アルゴリズムの数及び種類が多岐にわたることがある。そのため、業務を行う上でどのアルゴリズムを優先して勉強するか、といった優先順位をつけることが容易でない。また、 アルゴリズムを習得した場合でも、その利用に至るまでデータの分析テクニックや、スケーリングなどの前処理に関する知識まで網羅されているとは限らず、 適切にアルゴリズムを適用できないという問題もある。

そこで本稿では、機械学習のスキルが十分でない人材でも効率的なデータの活用を可能とするために、データサイエンスにおける代表的なコンペティションであるKaggle[2]から、アルゴリズムの適用だけでなく、 データの前処理を行う上で必要なノウハウも含めた機械学習アルゴリズムの適用パターン(以後、「適用パターン」と呼ぶ)の抽出を行った。また、それと併せて、 データの量や質的変数の数などの特性を見た上で、 どのアルゴリズムを適用すればよいかを示した機械学習アルゴリズムの選択パターン(以後、「選択パターン」と呼ぶ)の抽出も行なった。

これらにより、 新たなデータを得た際に、どのアルゴリズムを選択すればよいかが確認でき、 それが未知のア

ルゴリズムであれば、適用例とともに学習できるようになる。

本稿では、適用パターンがあることで、機械学習の実装時間がどれほど短縮できるかを実験により確認した結果を報告する。アルゴリズムの選択パターンについても、同様に、それがあることで、機械学習アルゴリズムの選択をどれほど正確かつ効率的に行えるかを確認する必要があるが、それには実験内容や実施方法の検討を重ねる必要があったため、今後の課題とする。

2. 事前準備

本章では、 適用パターンについて、抽出の対象とする機械学習アルゴリズム、パターンを構成する項目、パターンの抽出プロセスを説明する。

2. 1 パターンの抽出対象アルゴリズム

Kaggle ではコンペティションごとにデータセットとアルゴリズムを適用したプログラムが議論用および参考用にカーネルとして投稿されている。 さらに、これらのカーネルの中から特に参考になるものを選び出して、アルゴリズム毎にまとめたカーネルも投稿されている。

本研究では、このカーネル(Data Science Glossary on Kaggle![3])をもとにして、そこで挙げられたアルゴリズムの中から、主に、初学者にとっての重要度、難易度、活用場面の観点で絞り込みを行い、以下の 11 種類のアルゴリズム

■ 教師あり学習
1. 線形回帰
2. Ridge 回帰
3. Lasso 回帰
4. ロジスティック回帰
5. 決定木
6. ランダムフォレスト
7. 勾配ブースティング木(LightGBM 実装)
8. サポートベクターマシン (SVM)

■ 教師なし学習
9. K-means クラスタリング
10. 階層型クラスタリング
11. DB SCAN

を選定し、適用パターンの抽出を行なった。他にも、深層学習を含め多数のアルゴリズムが存在するが、それらからのパターン抽出は、引き続き行っていく予定である。

本稿では、 上記のアルゴリズムの一つである Lasso 回帰の適用パターンを示す。

2. 2 パターンを構成する項目

本研究では、適用パターンを抽出する上で、 文献 [4]を参考にしてパターンを構成する項目を決めた。具体的には、以下の項目をパターンテンプレートとして定め、パターンを記述した 。

- パターン名と分類
- 目的
- 課題例(Kaggle 上の問題/データセット)
- 適用条件
- 適用手順
- 実装上の注意点（python3）
- サンプルコード（jupyter notebook 形式）
- 適用結果
- 理論的背景
- 出展および参考文献
- 関連するパターン

実装上の注意点やサンプルコードについては、jupyter notebook 上で python3 を用いた内容を載せている。

2. 3 パターンの抽出プロセス

本稿は、 国立情報学研究所(以後、 NII と呼ぶ)の社会人向け教育プログラムであるトップエスイー[5]のソフトウェア開発実践演習での活動(2018/10～2019/3 に実施)をもとに執筆された。本演習のスケジュールを下の図 1 に示す。

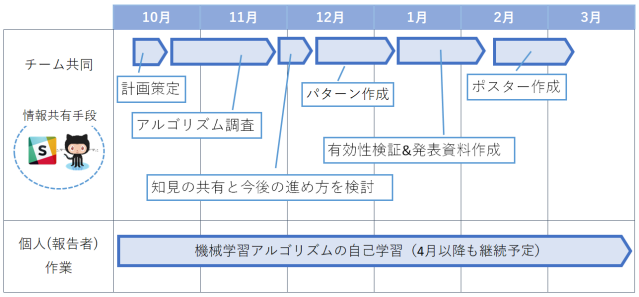

図 1:演習スケジュール

演習はチームで行われ、このチームは各企業から派遣された 6 名のメンバと 1 名の講師により構成された。さらに、6 名のメンバのうち、 1 名は機械学習の熟練者であり、残り 5 名は初学者であった。

以下に、パターンの抽出プロセスを時系列順に説明する。

2.3.1 計画策定（10/4～10 月中旬まで）

最初は計画策定フェーズとして、概ね隔週の頻度でメンバが NII に集合し、演習が進められ、演習目標、 課題設定、スケジュールのすり合わせを行った。

2.3.2 アルゴリズム調査（10 月中旬～11 月末）

先述の 11 種類のアルゴリズムを、各メンバが基本的に 2 種類ずつ調査を行い、 その結果を NII で他のメンバに説明し、調査を通して得られた知見を共有した。 なお、共有の際には、3 人 1 組で 2 グループに分かれ、各グループが交互に隔週で NII に集合することで、十分な調査期間の確保とメンバ全員に適度な負荷がかかるように、調整が図られた。

2.3.3 得られた知見の共有と今後の進め方の検討（11 月末～12 月上旬）

アルゴリズムの調査が完了し、グループ内で共有された知見を、さらにグループ間でも共有し合った。これにより、メンバは、アルゴリズム毎に、それがどのような性質のデータや問題に向くのかの理解に加え、アルゴリズム間の関係性(類似性や相反性など)や、Python での実装時の注意点などへも理解を深めることができた。

そして、ここで理解したことを、適用パターンとは別に、選択パターンとして、何らかの形式でまとめておく必要があるという認識を持つに至った。

2.3.4 アルゴリズム適用パターンとアルゴリズム選択パターンの作成（12 月～1 月）

知見の共有の後、適用パターンと選択パターンを抽出した。

(1)適用パターンの抽出

適用パターンの抽出では、 各アルゴリズムについて 2.2 節で示した項目をまとめた。

(2)選択パターンの抽出

与えられたデータの特徴に対してどのアルゴリズムを選択すればよいかという観点で、チーム内で議論を重ね、4 章に示すフローチャート形式の選択パターンを抽出した。

2.3.5 適用パターンと選択パターンの有効性検証 （1 月）

両パターンの有効性に関して、次の 2 点の仮説を立てた。

(1) 適用パターンを活用することでソースコードの実装時間が短縮する。
(2) 選択パターンを活用することで、データの特徴（データ量、変数の数など）に合致した適切なアルゴリズムが選択できる。

仮説検証プロセスのフィジビリティーを行った結果、仮説(2)はチーム以外の被験者による AB テストが必須であり、残された演習の期間で実施するには厳しいことが分かった。そこで、仮説(2)は今後の課題とする代わりに、仮説(1)の有効性の検証にチームのリソースを投入した。

3. 適用パターン

本章では、選定した 11 種類のアルゴリズムの一つである Lasso 回帰について、適用パターンに記載した内容を簡潔に紹介する。

例：Lasso 回帰

Lasso 回帰は一次のペナルティ項(L1 ノルム)を付加した線形回帰のアルゴリズムである。 L1 ノルムを付加することで係数パラメータがゼロになることが多くある。 この特徴を利用して、 Lasso 回帰は単に線形回帰の問題を解くだけでなく、 どの特徴量が重要であるかを抽出することができる。

以下は 2 章で述べたパターン化項目を Lasso 回帰に適用し、 パターン化した例を示す。

- パターン名と分類
  パターン名：Lasso 回帰の適用パターン
  分類：「回帰/特徴量抽出」
- 目的
  ・既知の説明変数から未知の量的変数を予測する
  ・説明変数の中で重要なものを抽出する
- 課題例
  House Prices: Advanced Regression Techniques [6]
  アイオワ州の住宅価格を 79 の説明変数から予測する（回帰）
- 適用条件
  1. 目的変数のヒストグラムがひと山になるもの
  2. 特徴量が多く、 重要なものがわずかしかないと予想される場合
  3. (伝統的に)線形になることが予想されるもの
- 適用手順

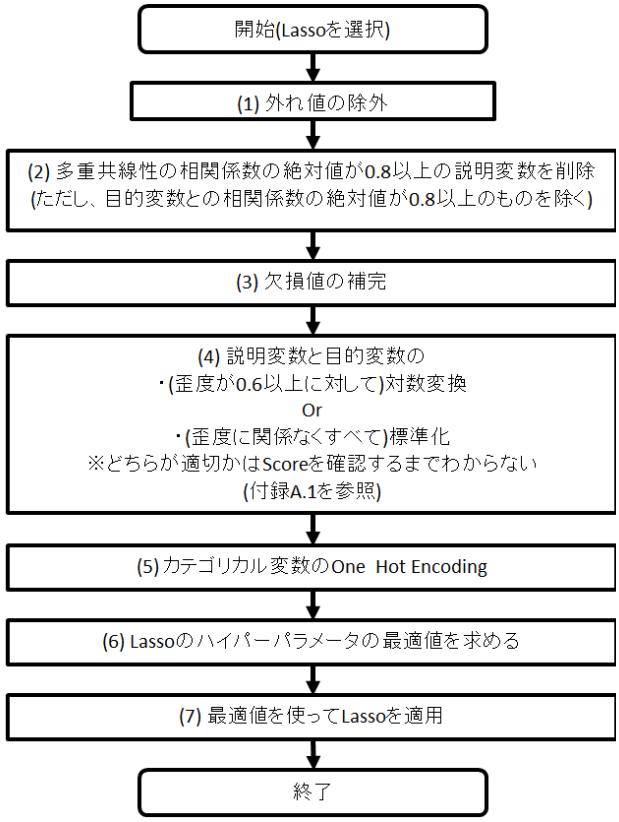

図 2: Lasso 回帰(回帰問題)の適用手順

図 2 は Lasso 回帰の回帰問題を解く適用手順である。Lasso 回帰で特徴量抽出を行う場合、 図 1 の「最適値を使って Lasso を適用」の後に回帰係数がゼロのものがあれば、 その説明変数を削除する。 その後、 Lasso のハイパーパラメータの最適値を求め、 Lasso を適用する。 回帰係数がゼロになるものがなくなるまでこれを繰り返し行うことで特徴量抽出ができる。

適用手順の 4 番目の内容(対数変換、 標準化)は対象データにより異なる。 現時点では、 対象データの特徴からどちらを選択すべきかを完全には特定できていないため、 今後の課題とする(付録 A.1 を参照)。

- 実装上の注意点（python）
  対数変換した場合は Lasso の予測値の exp をとる。
- サンプルコード（jupyter notebook 形式）
  データ読みから特徴量抽出までを行うサンプルを用意(付録 A.2 を参照)。
- 適用結果
  サンプルコードの実行結果について記載(付録 A.3 を参照)。
- 理論的背景
  L1 正則化することでスパースな解を得られやすい。
- 出展および参考文献
  省略。
- 関連するパターン
  Rigde 回帰：Lasso が L1 正則化に対して Ridge は L2 正則化
  XGBoost：Kaggle ではよく Lasso を使ってアンサンブルをとる例が見られる。

4. アルゴリズム選択パターン

適用パターンを抽出する中で各アルゴリズムには特性があり、 データによって適用すべきアルゴリズムが異なり適切に選択する必要があることが分かった。 従来は、 データに合わせた適切なアルゴリズムを機械学習スキルの高い人材によるノウハウにより選択していた。 そのため、 スキルが十分でない人材が適切なアルゴリズムを選択することは難しく、 十分なデータ分析を行うことができなかった。 適用パターンを抽出した際に得た知見をメンバ全員で共有する中で、 スキルが十分でない人材であっても容易に適切なアルゴリズムが選択可能となる選択パターン(図 3)を抽出した。

まず、データ自体が教師ありか教師なしかで分類する。

教師なし学習の場合、クラスタリング問題となり、クラスタ数が予め決まっているかどうかで分岐する。クラスタ数が既知の場合、クラスタ数が入力として必要となる Kmeans を使用する。クラスタ数が未知で説明変数が少ない場合に限り、DBScan を使用する。これは、DBScan が、入力としてクラスタ数が不要であり、説明変数が多くなると計算量も多くなるためである。一方、説明変数が多い場合は Hierarchical Clustering を使用する。

教師あり学習では、回帰または分類問題となる。

予測結果の過程を可視化する必要がある場合、説明変数の値から予測結果へのツリー分岐が可視化可能な Decision Tree を使用する。

可視化が不要で、データ数が十分に多い場合は、多量データに対して短時間で分析が可能な LightGBM を使用する。一方、LightGBM はデータ数が少ないと容易に過学習をおこしてしまい十分な性能を達成できないことから、データ数が少ない場合は他のアルゴリズムを使用する必要がある。

データ数が少ない回帰問題では、線形性があるかどうかを確認する。アルゴリズム選択パターンにおける線形性があるとは、 目的変数と説明変数の間でピアソンの相関係数を求め、 0.8 以上であるものである。 相関分析等で線形性があるかどうかを確認し、ラベルとの相関があり、線形性があると想定される場合、線形回帰アルゴリズムを使用する。説明変数が多い場合は、次元削減効果のある Lasso 回帰を使用する。一方、説明変数が多くない場合は、多くの

説明変数の情報を使用して予測することが可能な Ridge 回帰を使用する。

また、分類問題で線形性がある場合、計算量の少ないロジスティック回帰を用いる。

回帰問題、分類問題において線形性がないと推定される場合は、非線形問題に対応可能なアルゴリズムを選択する。

説明変数の数が多い場合は、Random Forest を用いる。

説明変数の数が少なく、カテゴリカルデータがある場合は、カテゴリカルデータをそのまま処理可能な Decision Tree を使用する。

説明変数が少なく、データ量も少ない場合は、多様な非線形データに対応可能な SVM (Support Vector Machine)を使用する。

しかし、SVM はデータ量が多いと計算量が大幅に増加するため、データ量が多い場合は kNN (k-Nearest Neighbor)を使用する。

データ分析の目的が、回帰でも分類でもなく、データに影響する主要な説明変数を抽出することである場合、次元削減効果のある Lasso 回帰を利用して抽出を行う。

5. 評価

5. 1　実験

適用パターンを利用することにより、 初めて見るデータセットに対して、機械学習の実装時間が短縮できることを、実験を通して検証する。 ここで、 データセットには、その特徴に応じて、適用するアルゴリズムと達成すべき予測精度(平均二乗誤差、正解率)の目標値を予め定めておく。そして、 グループのメンバを被験者として、データセットを割り振り、被験者が機械学習の実装に要した時間を計測し、また、実装したモデルの予測制度が目標値に到達するか否かを確認した。実装時間については、適用パターンの抽出前に、調査の段階で実装に要した時間と比較することで、 適用パターンの有無による実装時間の差異とした。なお、 アルゴリズムにより計算時間が異なるため、 プログラム実行時間は実装時間に含まないこととする。

実験の進め方に関する具体的なプロセスを以下に示す。

- チームメンバは、各自がすでに調査したアルゴリズムを対象に、問題と模範解答（ソースコード）を作成し、講師に提出する。
- 講師は、チームメンバに問題を割り振る。その際、チームメンバには、調査を担当したアルゴリズムから、出来るだけ遠い関係にあるアルゴリズムを割り振るように留意した。
- チームメンバは、講師が指定した問題を解き、ソースコードの実装時間と実装したモデルの予測精度とを併せて、解答(ソースコード)を講師に提出する。
- 講師は、解答をチームメンバに公開する。
- 模範解答を作成したチームメンバは、解答を評価し、解答者にフィードバックする。

5. 2　結果

図 4 は被験者ごとの適用パターンを参照していない状態(以後、「Before」と呼ぶ)と参照した状態(以後、「After」と呼ぶ)での実装時間を示したものである。 適用パターンを参照することで最大 95%、 平均 77%の実装時間が削減できることがわかる。 一方で最も改善効果が低い被験者 E は機械学習の熟練者であり、 その他の被験者は初学者である。 初学者は改善効果が大きいが熟練者は小さいことが分かった。 また、 Before と After で各被験者が使用した

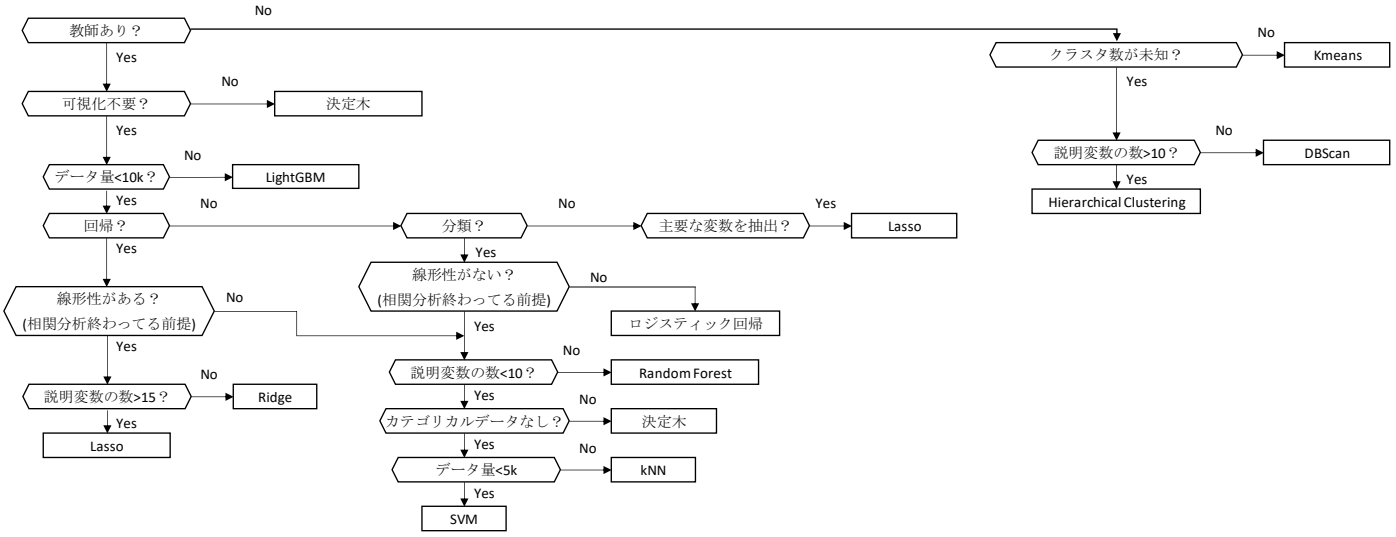

図 3: 選択パターン

アルゴリズムが異なるため、 図 5 にアルゴリズムごとに集計した実装時間で比較した結果を示す。 SVM 以外のアルゴリズムについては、 適用パターンを参照することにより実装時間の短縮効果があった。 一方、 Before の SVM の実装時間は熟練者が要した時間であり、 After の実装時間は初学者が要した時間である。 これにより、 初学者に適用パターンを参照させることで、 熟練者を超えることはできないが同等程度まで実装時間を短縮できる見通しを得た。

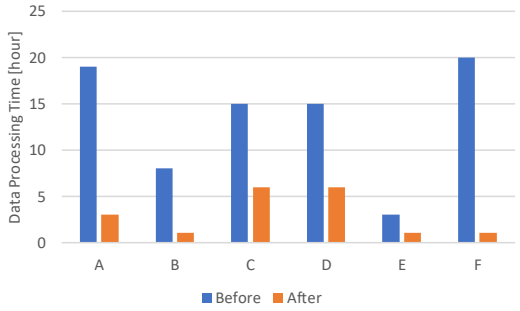

図 4: 被験者ごとの実装時間の比較

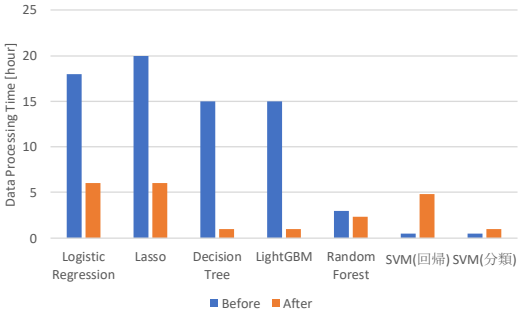

図 5: アルゴリズムごとの実装時間の比較

表 1 は、いくつかのアルゴリズムについて、予測精度に関する目標値と解答値を並べて示したものである。適用パターンを参照することで、精度においても、目標値を超える結果が得られていることがわかる。

アルゴリズム	予測精度(※)	
	目標値	解答値
ロジスティック回帰	0.75	データ無
ロジスティック回帰（テキスト分析）	0.996	0.97
SVM（回帰）	指定なし	6.88
SVM（分類）	0.98以上	1
ランダムフォレスト回帰	1.0未満	0.996

(※) 回帰の場合は平均二乗誤差, 分類の場合は正解率

表 1: アルゴリズムごとの予測精度の比較(抜粋)

6. 考察

6. 1 適用パターンによる効果に関する考察

機械学習の実装時間は、データの分析も含めた前処理とアルゴリズム適用に分けることができ、一般に 8 割は前処理であるという経験則がある[7]。適用パターンには、データ前処理も含めたノウハウを記載しているため、前処理においても時間短縮効果があり、結果として大きな時間短縮が可能になったと考える。

一方、今回の実験では、Before と After の間で、2.3 節の抽出プロセスに沿った流れで、個人で試行錯誤を繰り返しながらプログラミングのスキルを高め、また、チーム内での情報共有から機械学習の知識を習得することも行われていたため、時間短縮には適用パターン以外の効果も含まれていると考えられる。表 2 に、実装時間に影響を与える要因と、パターン適用による効果、及び、抽出プロセスを通じて得られた効果の関係を定性的に示す。適用パターンには、プログラミングスキルに関するものや、環境構築・設定等のノウハウは含まれていない。これらについては、抽出プロセスを通して得られたと考える。

本実験では、適用パターンを抽出したメンバ同士が被験者になり、クロスチェックにより効果を確認した。 しかし、適用パターンの効果を正確に評価するためには、抽出プロセスに関わらなかった被験者を集めて、 パターンを参照するグループと参照しないグループに分け、双方の実装時間の差を評価する必要があると考える。

適用パターンの効果の正確な評価については、今後の課題である。

	パターンの効果	抽出プロセスの効果
心理的障壁	△	○
機械学習の知識	○	○
Python スキル	△	○
ライブラリ知識	△	△
プログラミング環境設定	×	○
前処理	○	○
アルゴリズム適用	○	○
性能評価	○	○

表 2: アルゴリズムごとの実装時間の比較

6. 2　機械学習の習熟時間に関する考察

チームのメンバのうち 5 名は python およびデータ分析の初学者であったにもかかわらず、 初めて見るデータセットに対して短い時間で機械学習アルゴリズムを適用して結果を得られるようになった。 多くの参加者は他のプログラミング言語には習熟していたが python には殆ど触れたことがない状況であった。 その中で、 各自が 2 種類のアルゴリズムについて適用パターンを抽出し、その後、チーム内の議論を通して選択パターンを抽出することで、実験では実装時間を短縮することができた。ある程度アルゴリズムを定めた上で、 機械学習の実装に習熟しようとした場合、 表 3 に示したように、抽出プロセスの各アクティビティに費やした時間から、全体でおよそ 50 時間を必要とすることがわかった。

アクティビティ	作業時間(H)
アルゴリズム調査	30
得られた知見の共有と今後の進め方の検討	4
適用パターンと選択パターンの抽出	8
適用パターンの有効性検証	8
合計	50

表 3 抽出プロセスの各アクティビティに費やした時間

今後は、 基礎的な数学的原理の理解を含めた習熟時間の推計を行うことで、 企業の人材育成に活かすことができると期待される。

7. まとめと今後の課題

本研究では、機械学習のスキルが十分でない人材が、容易にデータを活用できるようにすることを可能にし、機械学習に熟練した人材の不足に対処していくことを目的に、次の 2 種類のパターン、すなわち、データの特徴に合致した適切な機械学習アルゴリズムを選択できるようにする選択パターンと、 機械学習のアルゴリズム毎に、それを適用するために必要なノウハウをまとめた適用パターンを抽出した。また、実験を通して、適用パターンを利用することで、機械学習の実装時間を平均で 77%削減することができ、適用パターンによりスキルが十分でない人材でも効果的に機械学習アルゴリズムを適用することが可能となる見通しを得た。

ただし、この実装時間の短縮効果が、適用パターンの効果によるものか、 あるいは、パターンの抽出プロセスを通して獲得したプログラミングのスキルや、チーム内での情報共有で習得した機械学習の知識によるものか現時点では判断することができない。

そこで、 今後は、適用パターンの効果と選択パターンの効果を併せて検証するために、抽出プロセスにかかわりがない第三者の被験者に対して、 パターンを参照するグループと参照しないグループに分けて、効果を判定するための実験を企画・実施していく予定である。

参考文献

[1] 前 一平(2018)　「ＡＩ時代を担う人材の育成」（立法と調査 2018. 10 No. 425 48-49p）
[2] Kaggle、 https://www. kaggle. com/
[3] Kaggle: Data Science Glossary on Kaggle、 https://www. kaggle. com/shivamb/data-science-glossary-on-kaggle。
[4] エリック ガンマ 他（1999）「オブジェクト指向言語における再利用のためのデザインパターン」ソフトバンククリエイティブ.
[5] トップエスイー、 https://www. topse. jp/ja/
[6] House Prices: Advanced Regression Techniques https://www. kaggle. com/c/house-prices-advanced-regression-techniques
[7] 有賀 康顕 他(2018)「仕事ではじめる機械学習」オライリージャパン.

付録

A.1

Lasso 回帰の適用手順 (4)の、対数変換すべきか、 それとも、標準化すべきかは、データにより異なることがわかったが、その条件を明らかにできていない。

一般に回帰分析における説明変数の標準化は、 数値データについて単位を持っていないため他の変数と比較可能にするために必要な処理である。 一方で歪度が高い(0.6 以上)変数を対数変換する理由は、 歪度が高いと分布の山が下に偏っているのを対数変換することで分布を正規分布に近づくように矯正するためである。 回帰分析では変数が正規分布することが前提になっているため、 分布を矯正する対数変換は必要な処理だと考えている。 従って、 標準化と対数変換のどちらも必要な処理だと考えるが、 実際のデータに適用するとどちらかのみを適用した方が精度が良くなった。

この理由については、 次の 2 つのデータ特性が関係しているのではないかと考えている。

(1) 欠損値が多く含まれるデータ

欠損値が含まれる場合、 標準化前に欠損値の穴埋めをする必要がある。この穴埋めに通常、 平均や中央値、 最頻値といった値で一律に穴埋めする。欠損値が多く含まれるデータの場合、 穴埋めをしたデータが多く本来の分布と乖離して標準化がうまくできないのはないかと思われる。従って、 欠損値が多く含まれるデータについては、 標準化はしないほうが良いと考えている。

(2) 目的変数がゼロ付近の値を多く持つデータ

ゼロ付近の値に対数をとる(log1p)と僅かな差が、 対数変換後の大きな値の差になる。この差がモデル構築に影響し、性能が得られなくなってしまうのではないかと考えている。従って、 目的変数がゼロ付近の値を持つデータについては、対数変換しない方が、性能が出るのではないかと考えている。

A.2

3 章の Lasso 回帰のサンプルコードの抜粋を以下の図 6 に示す。

実際のサンプルコードではデータの読み込みや外れ値の除外などの前処理も含む。

Lassoのハイパーパラメータの最適値を求める

```
In [29]: #学習データ、テストデータに分割
         # X: df_allの1行目からdf_trainの行数目まで(トレーニングデータセット部分)
         X = df_all[:df_train.shape[0]]
         # X_for_test: df_trainの行数目+1から最終行まで(テストデータセット部分)
         X_for_test = df_all[df_train.shape[0]:]
         y = df_train.SalePrice
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1111)

In [30]: best_rmse = 1
         for num in range (1, 100):
             alpha = num*0.00001
             reg = Lasso(alpha=alpha, max_iter=3000)
             reg.fit(X_train, y_train)
             y_pred = reg.predict(X_test)
             rmse = np.sqrt(mean_squared_error(y_pred, y_test))
             if best_rmse > rmse :
                 best_rmse = rmse
                 best_alpha = alpha

         print("alpha:", best_alpha,", ラッソ回帰でのRMSE:", best_rmse)
         alpha: 0.00055 , ラッソ回帰でのRMSE: 0.114252193916
```

図 6 : サンプルコード例

A.3

3 章の Lasso 回帰のアルゴリズム適用パターンの中で、Lasso 回帰を繰り返し適用することで特徴量抽出ができることを紹介した。 実際に、 3 章の問題で Lasso 回帰を繰り返し適用した結果が以下の表 4 である。 表 4 の説明変数の数はカテゴリカル変数を One Hot 表現後の数である。 この結果を見ると、 説明変数の数は 1 回目の適用で 190 もの説明変数の回帰係数がゼロになった。 2 回目の適用では 1 つしか回帰係数がゼロにならなかった。 このことから Lasso 回帰を 1 回適用すれば有効な説明変数をほぼ特定できると考えている。 また、 RMSE の値は各回で大きな差はなかった。

適用回数	説明変数	RMSE	Kaggle 順位
1	283	0.11703	614 位
2	93	0.11708	622 位
3	92	0.11703	614 位

表 4: Lasso 回帰適用時の説明変数の数と RMSE