

kaggleのチュートリアル

はじめに

この本は、Kaggleを初めてやってみたい人のためのチュートリアルを目的としています。特に機械学習にあまり馴染みのない方が、Kaggleを通して機械学習に入門するための本として書いています。(言語はPythonを用います。)

機械学習について、Wikipediaで調べると

機械学習 (きかいがくしゅう、英: machine learning)とは、人工知能における研究課題の一つで、人間が自然に行っている学習能力と同様の機能をコンピュータで実現しようとする技術・手法のことである。

と書かれています。つまり、人間が自然に行なっている、既存データを学習して未知のデータに対して予測しようとするのを、コンピューターにさせることでしょう。

しかし、機械学習を勉強しようと思い「入門機械学習!」というようなタイトルの本を読むと、数学 (線形代数や確率統計など)を用いた説明があるため、全く機械学習をやったことがない状況だとイメージもわからず、なかなか理解できないことがよくあります。

一般に、理論から学び始めるのは難しく、まずは実践を行いイメージや概念がわかってから、その後に理論を学ぶと言う順序で学習をすると良いのだと思います。

そこで、機械学習の実践を行うにあたり、Kaggleというデータ分析のコンペがおすすめです。

Kaggleでは、常に複数のコンペが開かれ、データ分析の入門者からエキスパートまでが腕を競い、無料で参加できるばかりか、良い成績を残すと賞金までもがもらえます。

しかし、日本人がこのコンペに参加しようとする、と「機械学習等のデータ分析のスキルの壁」とコンペが英語で行われているという「言葉の壁」という2つの障壁があり、なかなか大変です。

そこでこの本では、Kaggleのチュートリアルコンペであるタイタニックコンペを例に丁寧に説明や翻訳を行い、初めてKaggleに参加してみることが、難なくできるようにしています。

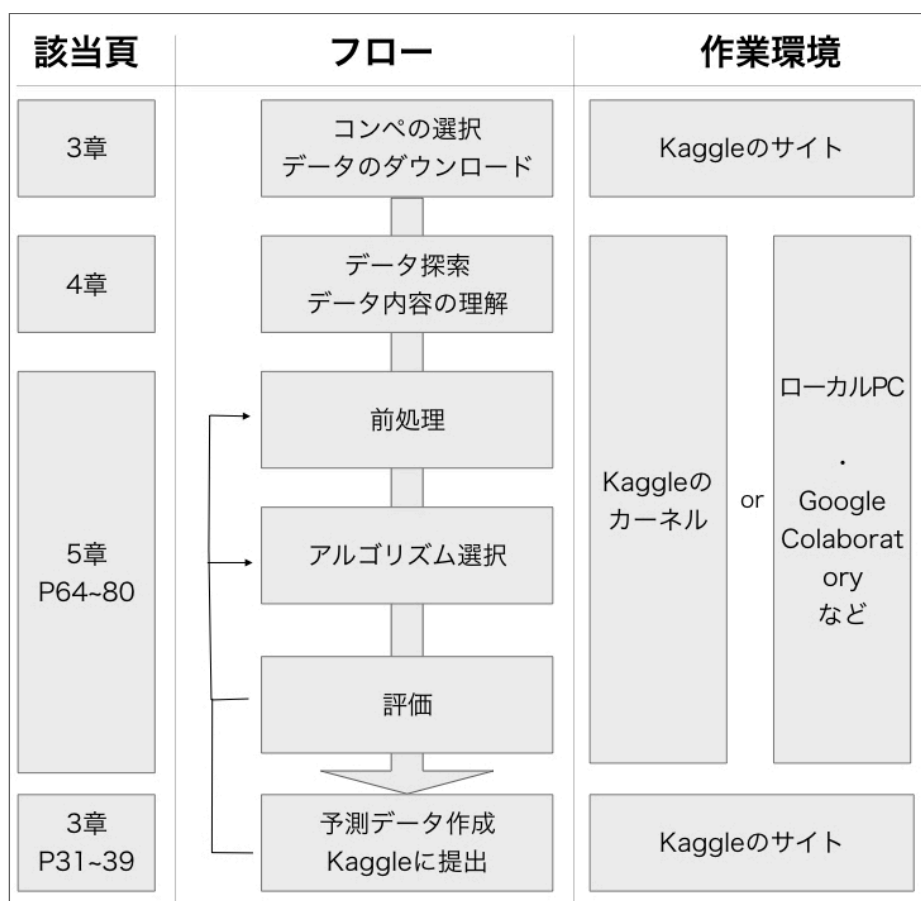
1度参加をしてみれば、壁は低くなっていますのであとは実践あるのみです。

本書の構成

第1部と第2部のKaggleのチュートリアルでは、タイタニック号の乗船客の生死を予想するコンペのデータを可視化し、機械学習のライブラリを使って分析を行いKaggleにデータを提出します。

この分析は、ブラウザから使えるKaggleのKernel (カーネル環境)を用いるので、PCとブラウザさえあれば、行うことができ、プログラミング言語のインストール等は必要ありません。

付録には、GoogleColaboratoryでKaggleをやる方法や、Kaggleの用語集、データ分析の勉強方法などをつけています。



Kaggleでのコンペ参加の流れと作業環境

本書の想定する読み方

第1部と第2部では、この本を順にコードを動かしながら読むことを想定しています。

【第1部】

まずは1章に概要、2章にタイタニックチュートリアルWebサイトの翻訳があるので、さらっと読んでみてください。

3章ではカーネルの使い方や、データの提出（サブミット）方法などを説明しますので、一通り動かしてみましょう。

【第2部】

第2部から実際にタイタニックのデータを扱います。サポートサイトにコードがありますので、手を動かし、可視化したりサブミットしながら読むとよいでしょう。

4章では、タイタニックのデータについて、どのような内容なのか探索します。

5章では、4章でわかったことをもとに、前処理を行ったりモデルの改良を行い、予測データを作成します。

【付録】

目次を参照いただき、気になった項目があれば読んでみてください。

目次

第1部

第1章 Kaggleについて	7
1.1 Kaggleとは	7
1.2 Kaggleへの登録とログイン	8
1.3 Kaggleトップページの説明	9
1.4 コンペ (Competitions)のページの概要	9
第2章 コンペのページの翻訳など	12
2.1 Overview (概要)	12
2.2 Data (データ)	21
2.3 Kernels (カーネル環境)	24
2.4 Discussion (ディスカッション)	27
2.5 Leaderboard (リーダーボード)	27
2.6 Rules (ルール)	28
2.7 Team (チーム)	29
2.8 My Submissions (自分のサブミット一覧)	29
2.9 Submit Predictions (予測のサブミット)	30
第3章 まずは、サブミットしてみる	31
3.1 データの作成・サブミット方法	31
3.2 カーネル環境のノートブックを用いる方法	31
3.3 スクリプトでサブミットする方法	36
3.4 ローカルPCで作成したデータをサブミットする方法	38

第2部

第4章 タイタニックデータの概要	40
4.1 ライブラリのインポートとデータの読み込み	40
4.2 データの概要を確認する	41
4.3 まとめ	63
第5章 識別器に学習させて予測する	64
5.1 前処理	64
5.2 識別器に学習させて予測	70
5.3 モデルの改良	72

5.4 まとめ	79
付録	
A GoogleColaboratoryの使い方	81
A.1 GoogleColaboratoryとは何か	81
A.2 Kaggleのカーネル環境との違い	81
A.3 Kaggleでの使用方法	82
A.4 matplotlibで日本語を使う方法	86
B 可視化の方法(matplotlibでの可視化)	87
B.1 pyplotでの可視化	87
B.2 pyplotのfigure関数を用いる方法	93
C Pythonのインストール	95
C.1 Anacondaのインストール	95
C.2 Jupyterノートブックについて	95
D Pythonの基本	95
D.1 四則演算	96
D.2 リスト、リスト内包表記	99
D.3 リストの要素を整数型にする	100
D.4 .copyが必要な理由	100
E Kaggleの称号と用語集	102
E.1 Kaggleの称号の説明	102
E.2 Kaggle用語集	103
F データ分析の勉強方法	104
F.1 Kaggleで初サブミット	104
F.2 Kaggleを楽しむ	104
F.3 知識をインプットする必要	105
F.4 実践	106
F.5 網羅的に理解する	106
F.6 その後	106

第1部

第1章 Kaggleについて

この章では、Kaggleについての説明、Kaggleでのアカウントの作り方、Kaggleのサイトがどのような構造になっているかなどKaggleを初めてやるべき時に知っておくとよいことを説明します。

1.1 Kaggleとは

Kaggleを一言で表現すると、「1年中色々なデータ分析の大会が開かれている、入門者からエキスパートまで楽しめるサイト」と言うことができると思います。

もう少し説明すると、Kaggleは世界中のデータサイエンティストがデータ分析の腕を競う、予測モデリング及び分析手法関連プラットフォームです。

具体的には、

- ・ 企業などが保有データの良い分析手法を求めてコンペ (大会)を開催し、データ及び賞金の提供を行い
- ・ コンペ参加者である世界中のデータサイエンティストは、データ分析の腕を競い、また分析手法等についての議論を行っている

ところです。

特に、各コンペのページにある「Kernels」や「Discussion」では、分析結果の共有や分析手法についての議論がされており、一流のデータサイエンティストが実践的な内容を公開しているので、読んでいだけでもとても勉強になります¹。

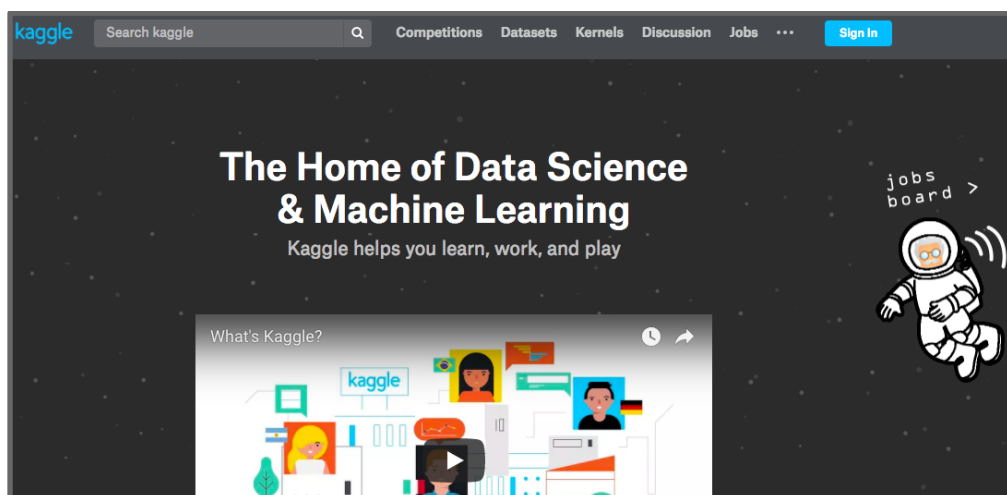
そのため、データ分析のガチ勢と言われる真剣にデータ分析をやっている強い方達にどのようにデータ分析の勉強をすれば良いか聞くと「Kaggleやれ」と返事をもらうことが多いです。

それではKaggleを始めてみましょう！

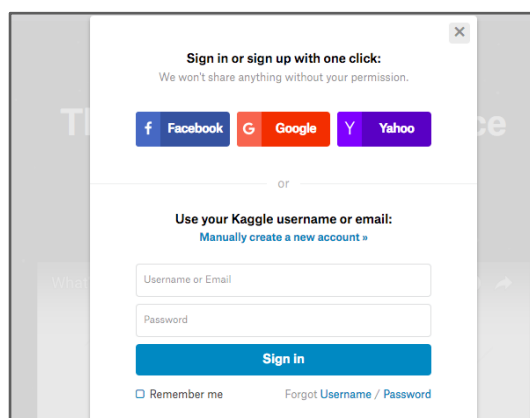
¹ ほぼ全てが英語で投稿されているので、英文を読む必要がありますが、コードと英文がセットなので、意外と英語が苦手でも読むことができます。

1.2 Kaggleへの登録とログイン

Kaggleのサイト (<https://www.kaggle.com/>¹⁾)にアクセスします。



右上にある「Sign In」をクリックすると、次のようにFacebook、Google、Yahoo!²⁾のいずれかのアカウントでログインするか、新しいアカウントを作ってログインするかを聞かれるので、お好みの方法でログインしましょう。

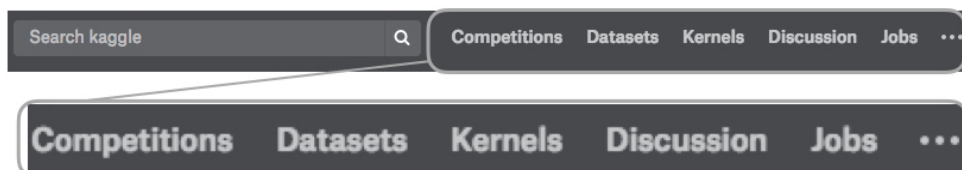


¹ このページに、What's Kaggle? (<https://www.youtube.com/watch?v=AoRSldLpFqU>)というKaggleの概要を説明したyoutubeのへのリンクが貼ってあります。アニメでわかりやすい説明になっているので、見てみるのも良いでしょう。

² Yahoo!JAPANのアカウントは使えません。

1.3 Kaggleトップページの説明

Kaggleにログインすると、上部に次のような検索ボックスとナビゲーションが表示されます。




ナビゲーションのうち、主な項目の内容は次のとおりです。

- Competitions：現在開催されているコンペと過去のコンペの一覧。
- Datasets：様々なデータセットの一覧。
- Kernels：Kernel (カーネル環境)は参加者がブラウザ上からPythonやRのコードを動かせる環境のこと。Kernelsは、参加者が公開しているカーネルの一覧。
- Discussion：ディスカッションの場。
- Blog：Kaggle公式のBlog。ニュースや「WINNERS' INTERVIEWS」というコンペ上位者へのインタビュー等のコンテンツがある。

1.4 コンペ (Competitions)のページの概要

上部のナビゲーションからCompetitionsに進むと、次のように自分が参加しているコンペ、現在開催されているコンペ、過去のコンペの順にカテゴライズされてコンペの一覧が表示されます。次の表示であれば、現在2つのコンペに参加しており、その他14のコンペが開催中ということです。


2 Entered Competitions



Toxic Comment Classification Challenge
Identify and classify toxic online comments
Featured · 22 days to go · arguments, text data


コンペのタイトル

コンペの締め切り




\$35,000
3,186 teams

コンペの賞金と参加チーム数




Titanic: Machine Learning from Disaster
Start here! Predict survival on the Titanic and get familiar with ML basics
Getting Started · 2 years to go · tutorial, tabular data, binary classification



Knowledge
5397/10020

14 Active Competitions



2018 Data Science Bowl
Find the nuclei in divergent images to advance medical discovery
Featured · 2 months to go · biology

\$100,000
1,776 teams

また、コンペのタイトルや、締め切り、賞金、参加チーム数などが表示されており、どのようなコンペがあるか確認することができます。

この本で挑戦するタイタニックのコンペに進んでみましょう。
(「Titanic:Machine Learning from Disaster」と記載があるのがタイタニックのコンペであり、初めての方の場合はActive Competitionsに表示されます。)


次のタブが表示されるので、「Join Competitions」をクリックし、コンペのルールに同意するか聞かれるので、ルールを確認した上で、「I Understand and Accept」をクリックします。

Overview Data Kernels Discussion Leaderboard Rules


Join Competition

次のように「Overview」、「Data」、「Kernels」、「Discussion」、「Leaderboard」、「Rules」、「Team」、「My Submissions¹」、「Submit Predictions」というタブが表示されます。

Getting Started Prediction Competition



Titanic: Machine Learning from Disaster
Start here! Predict survival on the Titanic and get familiar with ML basics



Kaggle · 9,573 teams · 2 years to go

Overview Data Kernels Discussion Leaderboard Rules Team My Submissions Submit Predictions

¹ 正確にはMy submissionsは、初サブミット（データの提出）後に表示されます。

10

コンペのページにどのようなことが記載されるのか、タイタニックのコンペを例に説明します。

- Overview¹：概要が記載されます。Overviewには、更に次のようなメニューがあります。
 - Description：コンペ概要の説明
 - Evaluation：どのようなデータをサブミット (提出) する必要があるか、どのように点数がつけられるか
 - Frequently Asked Questions：よくある質問と回答
 - Tutorials：タイタニックチュートリアルを進め方
- Data：提供されるデータ (トレーニングデータ、テストデータ、提出データのサンプル等) についての説明。
- Kernels：参加者が公開しているスクリプトやノートブックの一覧。
- Discussion：ディスカッションの場。
- Leaderboard：順位表。
- Rules：チームやサブミット回数等のルールについて。
- Team：チームを作るときに使う。
- My Submissions：自分がサブミットしたデータの一覧。
- Submit Predictions：予測をサブミットするページ。

このようにそれぞれのコンペのページでは、コンペの概要、評価方法、スケジュール、どのようなデータが提供されるのか、ルールなどを確認することができます。

コンペのページにはどのようなデータを扱うかや評価方法などが含まれるため、コンペに参加する際に確認することはとても大切です。

次章ではタイタニックチュートリアルのコンペのページに、具体的にどのような記載がされているかを説明します。

¹ Overviewにある項目はコンペによって異なりますが、通常、「Prizes：賞金に関する記載」、「Timeline：データの提出期限に関する情報」もあることが多いです。

第2章 コンペのページの翻訳など

この章では、タイタニックチュートリアルコンペのページに、具体的にどのようなことが書いてあるかを確認します。少し長いですが、知っておくと良いことが多いので全て訳しています。長いので、まずは2.1.2Evaluation (評価)と2.2 (Data)を読んで3章に進んでいただき、必要に応じて残りを読むと良いでしょう。

なお、この章では、Kaggleのページを翻訳した箇所は、四角の枠で囲むことにします。

2.1 Overview (概要)

タイタニックチュートリアルOverviewは、Description、Evaluation、Frequently Asked Questions、Tutorialsの4つに分かれています。

2.1.1 Description (説明)

ここから始めよう

もし、データサイエンスや機械学習に慣れていない、あるいは、Kaggleのコンペの簡単な説明を探しているのであればこのコンペから始めよう。

コンペの説明

タイタニック号の沈没は、歴史上最も悪名高い船の事故の1つです。タイタニック号は初めての航海中の1912年4月15日に冰山に衝突して沈み、2224人の乗客と乗組員のうち1502人が死亡しました。この驚異的な悲劇は国際社会に衝撃を与え、船舶の安全規制に影響を与えました。

この事故がこのような大きな事故となった原因の1つは、乗客と乗組員に対し十分な救命艇がなかったことです。けれども女性、子供、高い客室クラスの人などは、他の人よりも生存率が高かったという事実があります。

このコンペのチャレンジでは、どのような人々が生き残る可能性が高いのか分析してください。特に、どの乗客がこの悲劇から生き残ったかを予測するために、機械学習を用いましょう。

練習するスキル

- ・ 2 値分類
- ・ Python、Rの基礎

2.1.2 Evaluation (評価)

やること

コンペ参加者のすべきことは、乗客が、タイタニックの沈没で生存したか、死亡したか予測することです。

テストデータのそれぞれの乗客に対し、「Survived」変数を0または1の値で予測する必要があります。

スコア

正しく生存を予測した乗客の割合がスコアになります。これは単に「accuracy」(精度)と表記されます。

提出するファイルのフォーマット

ヘッダ (1 行) と 4 1 8 行で構成される csv ファイルをサブミットする必要があります。もし、余分な列または行がある場合には、サブミット時にエラーが表示されます。

サブミットするファイルは次の 2 列である必要があります：

- ・ 「passengerId」 (並び順は任意)
- ・ 「Survived」 (1 (生存) 又は 0 (死亡) のどちらかからなる予想)

(サブミットするデータの例)

PassengerId,Survived

892,0

893,1

894,0

Etc.

「Data page¹」でサブミットファイルのサンプル
(gender_submission.csv)をダウンロードできます。

2.1.3 Frequently Asked Questions (よくある質問)

「Getting Started competition」って？

Getting Started competitionは、機械学習のバックグラウンドがない人のためにKaggleのデータサイエンティストによって作成されました。このコンペは、データサイエンスを初めて勉強した、あるいは、MOOC²を終えてKaggleを始めたい、という方に最適です。

Getting Started competitionでは、Kaggleの仕組みを知り、機械学習の概念を学び、コミュニティの人たちと出会います。なお、賞金はなく、またrolling timeline³を採用しています。

¹ <https://www.kaggle.com/c/titanic/data>

² Massive Open Online Course またはMassive Open Online Courses の略。インターネット上で誰もが無料で受講できる大規模な開かれた講義のこと

³ 17ページの「私のチームは何故leaderboardから消えたのですか」参照

「Private Leaderboard (非公開スコア)」と「Public Leaderboard (公開スコア)」の違いは何ですか？

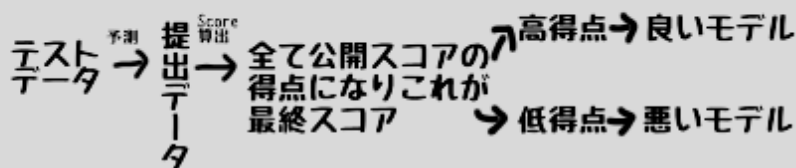
Kaggleのスコアには、参加者が特定のデータに「overfitting (過学習)」するのを防ぐため、非公開スコアと公開スコアがあります。もしモデルが過学習している場合、モデルを訓練したデータ以外のデータには適合しません。これは、類似の別のサンプルでは、モデルの精度が低くなることを意味します。

Public Leaderboard (公開スコア)：どの参加者も、テストデータの予測(サブミットしたデータ)の50%が公開スコアに割り当てられます。参加者が、公開スコアでみるスコアは、この50%のスコアになります。

Private Leaderboard (非公開スコア)：残り50%の予測は非公開スコアに割り当てられます。参加者は非公開スコアをコンペが終わるまで、見ることはできません。コンペが終わると、非公開スコアが公開され、スコアが表示されます。非公開スコアの得点は、コンペの勝者を決定するために使用されます。なお、Getting started competitionsでは、非公開スコアは公開されません。

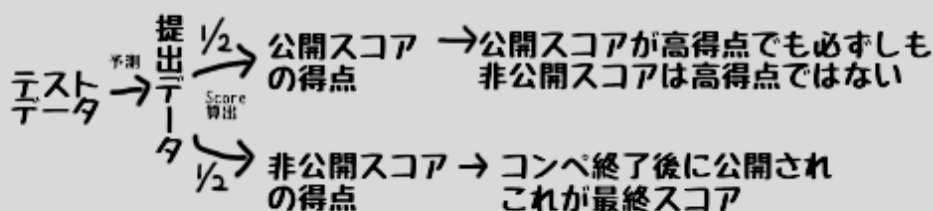
Public LeaderboardとPrivate Leaderboardの2つにLeaderboardを分ける理由は、初めての方には分かりづらいので次の図で説明します。

①PublicLeaderboard(公開スコア)のみ



→ 公開スコアを頼りにサブミットを繰り返しこれに最適化！

②PublicLeaderboard(公開スコア)とPrivateLeaderboard(非公開スコア)



→ 非公開スコアに最適化する必要がある

→ 新規データに最適化したモデルに！

①のようにPublicLeaderboardのみだと、公開スコアを頼りに、モデルを修正しサブミットを繰り返すことにより、公開スコアに最適化してしまうため、新規のデータに適合しないモデルがコンペの上位になってしまいます。

そこで、②のように1/2を公開スコア、1/2を非公開スコアにして、コンペ終了後に公開する非公開スコアを最終スコアとします。こうすることで、公開スコアを目安とするものの、過剰な公開スコアへの適合は避けた、新規データに最適化したモデルが非公開スコアの上位になることになります。

どのようにチームを作成すればよいですか？

「competition rules」に同意すると、(自分ひとりの)チームが作成されます。チームに他のメンバーを招待したり、他のチームとのマージ(合併)を承認したり、チーム名などの基本情報を更新したりするには、Team page¹に移動します。

チームを組むことが新しいスキルを習得し、Kaggleを楽しむための最良の方法であると、多くの「Kagglers」から聞いています。もしチームメイトがいない場合、ディスカッションフォーラムでチームを組むことを望んでいる人がいないか尋ねてみましょう。

Kernelsとは何ですか？

カーネル環境は、再現性のある共同分析を可能にするクラウドコンピューティング環境です。カーネル環境は、RとPythonのスクリプトとJupyterノートブック及びRmarkdownをサポートしています。Kernelsのタブに移動すると、このコンペで公開されている全てのコードを見ることができます。カーネル環境を使用してデータサイエンスを学ぶ方法の詳細については、チュートリアルタブを参照ください。

私のチームは何故leaderboardから消えたのですか

Getting Started competitionsの精神を維持するために、サブミットから2ヶ月で入れ替える仕組みにしています。つまり、サブミットから2ヶ月経過すると無効になり、リーダーボードにカウントされなくなります。チームに過去2か月間サブミットがない場合、チームはリーダーボードから落ちます。

こうすることにより、リーダーボードを管理しやすいサイズに保ち、リーダーボードのスコアが多すぎることで、新しく始める人が迷子になることを防ぎます。

「私は高いスコアを得るために、頑張ったの元に戻して」という方は、「rolling leaderboard」に関するディカッション²を読んでください。

¹ <https://www.kaggle.com/c/titanic/team>

² <https://www.kaggle.com/c/titanic/discussion>

サポートに連絡するにはどうすればよいですか。

Kaggleには専用のサポートチームがないため、適切なフォーラムで質問することで、より迅速に回答を受け取ることができます。（このコンペでは、タイタニックのディスカッション¹を利用すると良いでしょう）。

サポートは、すべての参加者に起こるであろう問題のみが対象です。サポートに連絡する前に、ディスカッションにその問題の答えがないか確認するようにしてください。見当たらない場合は、フォーラムに問題を投稿してください。フォーラムには、データ、評価方法、さまざまなアプローチに関する有用な情報が満載であり、フォーラムを使用することをお勧めします。フォーラムで知識を共有すれば、他の方は更に多くのことを共有するでしょう。

問題が解決しない場合、またはすべての参加者に対して有益であると思われる場合は、コンタクトページ²から連絡ください。

2.1.4 Tutorials (チュートリアル)

Kernels環境で学ぶ

Kaggleのカーネル環境では、ほとんどのコンペのデータセットをブラウザ内で扱え、多くのデータサイエンス用のパッケージとライブラリが使えるようになっています。

また、カーネル環境は、RとPythonのスクリプトとJupyterノートブック及びRmarkdownをサポートしており、提出ファイルを作成することもできるし、コンペのデータを探索することもできます。

カーネル環境を使うには、次のいずれかの方法があります：

1. 「Kernels」タブから新しい「script」又は「notebook」を作成する方法
2. 既存のカーネルを「Fork」（コピー）することにより、編集して動かせるコピーを作成する方法

¹ <https://www.kaggle.com/c/titanic/discussion>

² <https://www.kaggle.com/contact>

コンペを始めるのに役立つ最良のカーネルを選びました。 次のカーネルを使用して、サブミットファイルを作成したり、データを探索することができます。スクリプト又はノートブックを開いて「Fork」をクリックすると編集可能なコピーが作成されます。

Pythonで始める方へ

人気のPythonモジュールを使用するこの簡単なアプローチから始めてみましょう。

[Titanic Data Science Solutions Python Notebook](https://www.kaggle.com/startupsci/titanic/titanic-data-science-solutions)¹

- ・ pandasを使用してデータを操作します。
- ・ matplotlibとseabornを使用してデータを可視化します。
- ・ scikit-learnを使用して予測モデルの構築について学びます。

次のカーネルは、より高度なテクニックと複雑なアプローチが含まれます。

[An Interactive Data Science Tutorial](https://www.kaggle.com/helgejo/titanic/an-interactive-data-science-tutorial)²

- ・ Jupyterノートブックの使い方に慣れる。
- ・ 機械学習における特徴量選択の重要性を学ぶ。

[Machine Learning from Start to Finish with Scikit-Learn](https://www.kaggle.com/jeffd23/titanic/scikit-learn-ml-from-start-to-finish)³

- ・ クロスバリデーションを使用して、モデルが新しいデータに汎化していることを確認する。(すなわち、オーバーフィットしていないということ)
- ・ パラメータチューニングとグリッドサーチを使用して、いくつかの分類アルゴリズムの中から最適なモデルを選択する。

¹ <https://www.kaggle.com/startupsci/titanic/titanic-data-science-solutions>

² <https://www.kaggle.com/helgejo/titanic/an-interactive-data-science-tutorial>

³ <https://www.kaggle.com/jeffd23/titanic/scikit-learn-ml-from-start-to-finish>

XGBoost Example¹

- ・ 極めて人気のある「XGBoost」 アルゴリズムを学ぶ。
- ・ リンク先に移動し、「show more」 をクリックするとコードを見ることができます。

An Introduction to Ensembling/Stacking in Python²

- ・ いくつかのモデルの予測を組み合わせる「アンサンブル」の基本的なスキルを使う

Rで始める方

Exploring Survival on the Titanic³

- ・ 特徴量選択と可視化の基礎。
- ・ データの欠損値をどのように扱うか。
- ・ ランダムフォレストを用いて予測を行う方法。
- ・ もしRmarkdownに慣れていない場合、「Code」タブをクリックしコードを見することもできます。

Families are Not Good For Survival⁴

- ・ モデルの予測を理解するために必要なこと。
- ・ 異なるモデルを比較するために、決定木の可視化をする。
- ・ 特徴量が予測精度にどのように影響するか見極める。

¹ <https://www.kaggle.com/datacanary/titanic/xgboost-example-python>

² <https://www.kaggle.com/arthurtok/titanic/introduction-to-ensembling-stacking-in-python>

³ <https://www.kaggle.com/mrisdal/titanic/exploring-survival-on-the-titanic>

⁴ <https://www.kaggle.com/jasonm/titanic/large-families-not-good-for-survival>

Kaggle外のチュートリアル

- ・ R & Python (interactive): Titanicのコンペで、最初の投稿ファイルを作成するための無料の対話式のチュートリアルが、DataCamp (R¹ / Python²) とDataquest.io (Python³)にあります。これらのチュートリアルは、機械学習及びRまたはPythonにはじめて触れる方を対象にしています。
- ・ R (local):あなたのローカルPCにRをインストールし、最初のサブミットファイルを作成する方法についての「Kaggler」作成のチュートリアル⁴。
- ・ Excel: 使い慣れたエクセルによる基本的な機械学習のコンセプトに関するチュートリアル⁵

2.2 Data (データ)

Dataのページにはいくつかのデータが並んでいます。データのいずれかを選択した上で、右上にあるDownloadをクリックすると、ダウンロードすることができます。その他、データの概要と定義が記載されています。

なお、2018年4月15日時点で、ベータ版ですが例えば、タイタニックコンペであれば、`kaggle competitions download -c titanic`とコマンドを打つことで、データをダウンロードできる公式のapiがリリースされました。apiの詳細について興味がある方は、私のブログ⁶に解説記事をかきましたので、参考にしてください。

¹ <https://www.datacamp.com/community/open-courses/kaggle-r-tutorial-on-machine-learning#gs.OAowlPw>

² <https://www.datacamp.com/community/open-courses/kaggle-python-tutorial-on-machine-learning#gs.=IXCXgA>

³ <https://www.dataquest.io/mission/74/getting-started-with-kaggle>

⁴ <http://trevorstephens.com/kaggle-titanic-tutorial/getting-started-with-r/>

⁵ <https://www.kaggle.com/c/titanic/discussion/28323>

⁶ <http://www.currypurin.com/entry/2018/kaggle-api>

概要

データは次の2つのグループに分かれています。

- ・ training set(train.csv) ・・・ トレーニングデータ
- ・ test set (test.csv) ・・・ テストデータ

トレーニングデータは、機械学習モデルを構築するために使用する必要があります。トレーニングデータは、乗客それぞれの生存に関する結果 (正解データ)を与えます。機械学習のモデルは、乗客の性別やクラスのような特徴量に基づいています。新しい特徴量を作成して使うこともできます。

テストデータは、モデルが新たなデータに対してどの程度うまくいくかを確認するために使用する必要があります。テストデータは、各乗客の生存について正解を与えていません。これを予測するのがあなたの役目です。トレーニングデータで作成したモデルを使用して、テストデータの各乗客がタイタニックの沈没時に生存したか、亡くなったかを予測します。

また、提出ファイルのサンプルである、gender_submission.csv (女性乗客のみが生き残っていると予測したデータ)が与えられています。

トレーニングデータとテストデータについては、分かりにくいので補足します。4章と5章で扱う時に見返してください。

・トレーニングデータ（学習用で、正解データ付きのデータ）は、分析を行い予測モデルを作成するために使います。

・テストデータ（本番予測用で、正解データなしのデータ）は、作成した予測モデルを使って、正解データの予測を行うために使います。

データの定義

変数	定義	Key
survival	生存	0=No,1=Yes
pclass	チケットのクラス	1=1st(Upper) 2=2nd(Middle) 3=3rd(Lower)
sex	性別	
Age	年齢	
sibsp	乗船している兄弟・配偶者の数	
parch	乗船している両親・子供の数	
ticket	チケットナンバー	
fare	運賃	
cabin	キャビン番号	
embarked	乗船した港	C = Cherbourg, Q = Queenstown, S = Southampton

変数について

pclass:社会的経済地位のかわり

1st = 上位

2nd = 中間

3rd = 下位

age:1歳未満の場合は小数点以下の端数を含む。年齢が推測されている場合は、「xx.5」としている。

sibsp:

Sibling = brother, sister, stepbrother, stepsister (兄弟、姉妹、異母 (異父)兄弟姉妹)

Spouse = husband, wife (mistresses and fiancés were ignored) (夫と妻 (愛人と婚約者は無視されている))

parch:

Parent = mother, father (母、父)

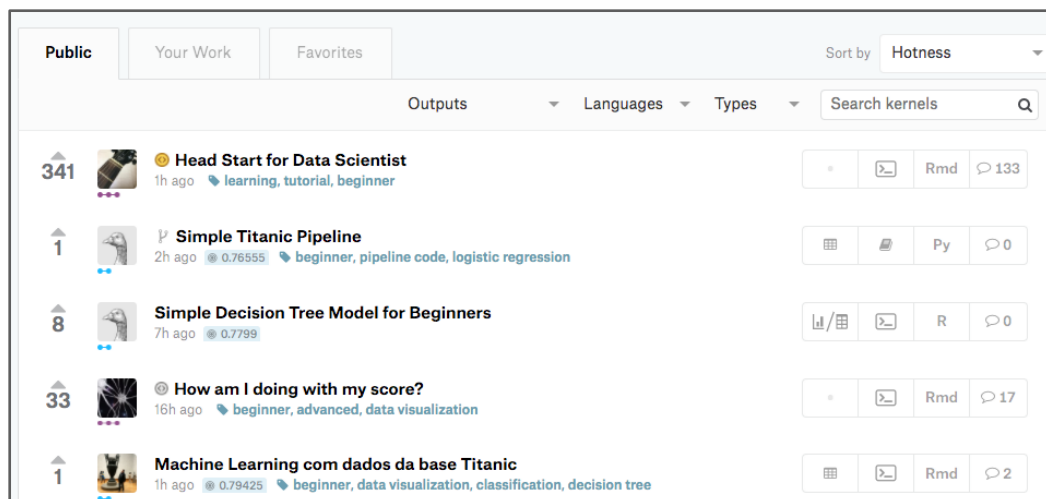
Child = daughter, son, stepdaughter, stepson (娘、父、継娘、継子)

何人かの子供は、乳母と旅行しているため、parchは0となっている。

2.3 Kernels (カーネル環境¹)

Kernelsのページを開くと、次のようにカーネルが並んでいます。

¹ 通常は単にカーネルと表現されますが、この本ではスクリプトとノートブックを合わせた環境のことを「カーネル環境」、個別のスクリプトまたはノートブックを「カーネル」と表現することとします。また、カーネルの使い方は3.2で説明します。







以下では、Kernelsに並んでいる項目について説明します。

Kernelsの左側の例

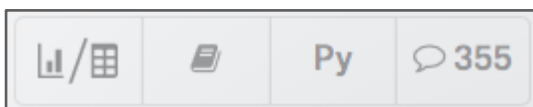


Kernelsの左側の項目は、次のとおりです。





 856	カーネルは数字の上の三角形の箇所をクリックすると、upvote (賛成票を投じること) することができます。数字はupvoteされた回数です。
 A Journey through Titanic	カーネルのタイトルとメダル ¹ 。
 0.74162	このカーネルのoutputデータをサブミットした時のスコア。
 beginner, eda, random forest	タグ。どのようなカーネルかわかる。

¹ カーネルのメダルについては私のブログ (<http://www.currypurin.com/entry/2018/02/21/011316>)を参考にしてください。

Kernelsの右側の例



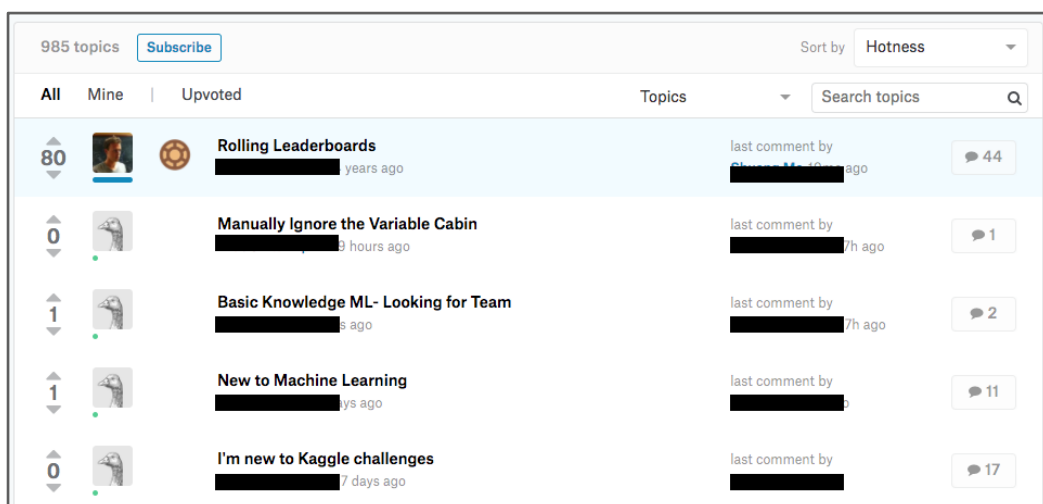
Kernelsの右側の項目は、次のとおりです。

 <div style="display: flex; justify-content: space-around; margin-top: 5px;"> <div style="text-align: center;">↑ ①</div> <div style="text-align: center;">↑ ②</div> <div style="text-align: center;">↑ ③</div> </div>	<p>①の棒グラフが表示されているカーネルは、可視化のコードが含まれカーネルということを表しています。</p> <p>②のエクセルの表のような絵が表示されているカーネルは、データのアウトプットがあることを表しています。</p> <p>③の①と②が両方表示されているカーネルは、可視化のコードとDataのアウトプットの両方があることを表しています。</p> <p>また、表示されている図の上にカーソルを乗せることで、いくつかの可視化のコードが含まれるか、データのアウトプットがあるかが表示されます。</p>
	<p>左の図はノートブック形式であることを表しています。右の図はスクリプト形式であることを表しています。</p>
	<p>Pyは、カーネルがPythonで書かれていることを表しています。RはR、Rmdは、RMarkdownで書かれていることを表します。</p>
	<p>カーネルへのコメント数です。</p>

2.4 Discussion (ディスカッション)

ディスカッションは、参加者がそのコンペでの分析手法等について議論をする場所です。

カーネルと異なりDiscussionではdownvote (反対票を投じること)ができます。そのため、左端の数字は「upvote数 – downvote数」が表示されています。また右端にはコメント数が表示されています。



2.5 Leaderboard (リーダーボード)

Leaderboardは、コンペの順位表です。上部には、自分の直近のサブミットの情報 (サブミット時につけた名前やスコア等)が表示されます。

また、その下には、Public LeaderboardとPrivate Leaderboardタブが表示されます。Public Leaderboardにおいて、その時点の順位表を確認でき、コンペ終了後にPrivate Leaderboardでそのコンペでの最終順位を確認できます。

「Kernel」の欄に表示があるものは、カーネルが登録されているものです。コンペ中に他の参加者が公開しているカーネルは当然参考になりますし、コンペ終了後にPrivate Leaderboardの上位で公開されているカーネルは、まさに知見のかたまりでもの凄く参考になります。

Your most recent submission				
Name	Submitted	Wait time	Execution time	Score
svm20180216.csv	12 days ago	0 seconds	0 seconds	0.77511
Complete				
Jump to your position on the leaderboard				

Public Leaderboard

Private Leaderboard

This leaderboard is calculated with approximately 50% of the test data.

The final results will be based on the other 50%, so the final standings may be different.

[Raw Data](#)
[Refresh](#)

Public Leaderboard

Private Leaderboard

Score

Kernel

#	△1w	Team Name	Kernel	Team Members	Score	Entries	Last
1	—				1.00000	2	2mo

Leaderboardをスクロールすると、自分のベストスコアがLeaderboardに表示され、その下には、直近のスコアが表示されます。

5271	▼ 610	currypurin	直近のサブミットのスコア	0.77511
Your Best Entry ↑				ベストスコア
Your submission scored 0.76555, which is not an improvement of your best score. Keep				

2.6 Rules (ルール)

どのコンペでも次のようにルールの記載があり、あらかじめ読んでおく必要があります。

参加者ごとに1つのアカウント

Kaggleに複数のアカウントを作ることはいけません。従って複数のアカウントからサブミットすることもできません。

チーム外で、情報を私的に共有してはいけません

チーム外でコードやデータを個人的に共有すること (Privately sharing) はできません。フォーラムで全ての参加者が利用できるのであれば、コードを共有可能です。

チーム管理

チームマージ (チームを合体すること)はチームリーダーが行うことができます。チームマージするには、マージ後のチームの合計のサブミット回数が、マージ日の最大許容回数以下である必要があります。最大許容回数は、「1日あたりのサブミット回数×マージ日までのコンペ開催日数」です。

チームサイズの上限

チームサイズの上限はありません。

サブミットの制限

1日あたり最大10回のサブミットができます。

ジャッジのために、提出するサブミットを5つまで選択することができます。

コンペのタイムライン

- ・ スタート:2012/9/28 9:13 PM UTC
- ・ チームマージの期限:なし
- ・ 参加期限: なし
- ・ 終了:2020/1/7 12:00 AM UTC

このコンペは、機械学習を始めるための楽しいコンペです。タイタニックのデータはインターネット上で公開されていますが、それを見ると、機械学習を用いて予測する楽しみをなくしてしまいます。なので、答えを見ずにやるべきだと非常に思います。

特に、「Privately sharing」は禁止されているため、チームメンバー以外の人との情報交換は、カーネルやディスカッションにおいてする必要があるため注意が必要です。

2.7 Team (チーム)

Teamでは、チームの名前の設定や、チームメンバーの閲覧、チームメンバーの招待、招待中のリクエストの閲覧、招待されているリクエストの閲覧をすることができます。

2.8 My Submissions (自分のサブミット一覧)

My Submissionsでは、自分のサブミットの一覧が表示されます。ここで「Use for Final Score」にチェックをつけることで、最終順位の算出 (Private Leaderboardのスコア)に使うサブミットを選びます。

2.9 Submit Predictions (予測のサブミット)

ローカルで作ったデータをサブミットする場所です。サブミットの仕方については、3.4 の「ローカルPCで提出データを作成してサブミット」で説明しています。

第3章 まずは、サブミットしてみる

この章では、タイタニックチュートリアルで予め用意されているサブミットサンプル (gender_submission.csv)や適当に作った提出データをサブミットし、どのような手順で提出データのサブミットを行なうかということを学びます。併せて、提出ファイルのフォーマットについても学びます。

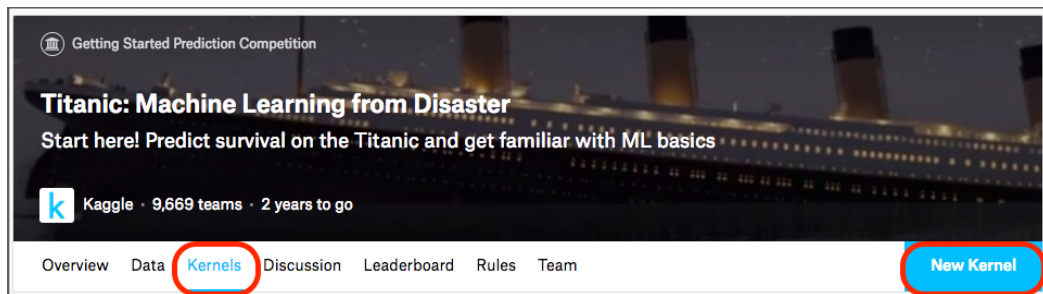
3.1 データの作成・サブミット方法

Kaggleにサブミットするデータの作成をKaggleのカーネル環境で行う方法と、ローカルPC等のカーネル環境以外で行う方法があります。また前者のカーネル環境で行う方法は、ノートブックで行う方法と、スクリプトで行う方法に分かれます。それぞれを順に説明します。

3.2 カーネル環境のノートブックを用いる方法

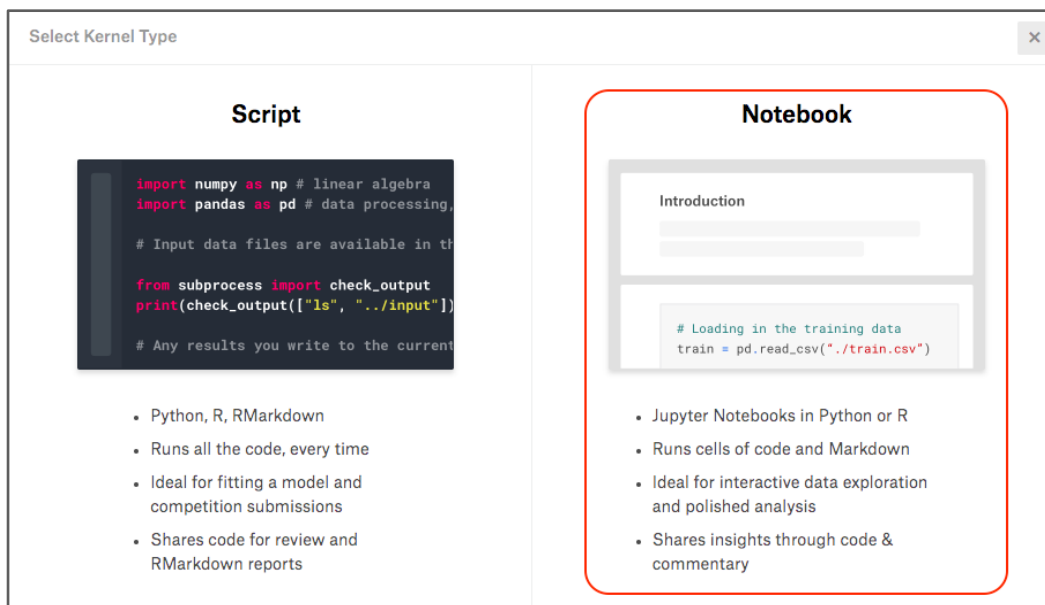
3.2.1 New Kernelの作成

KaggleのタイタニックのKernelsのタブ¹を開き、「New Kernel」をクリックします。



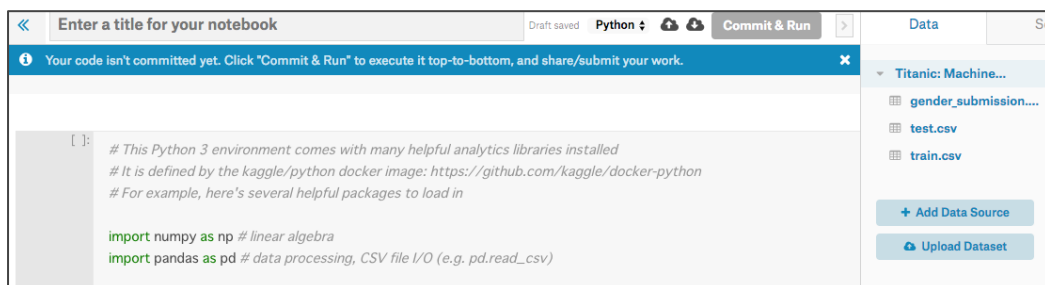
すると、次のようにスクリプトとノートブックのどちらを選ぶか聞かれるので、Notebookをクリックして選択します。

¹ <https://www.kaggle.com/c/titanic/kernels>



次のように、ノートブックのヘッダーが表示されるので、「Enter a title for notebook」と表示されているセルに適当なノートブック名を入力します。とりあえず「Firstnote」とでも名前をつけてみましょう。

これからは、このノートブック上で、分析や提出データの作成を行なっていきます。



ノートブックの使い方



ノートブックでは、白いセルにコードを入力し、▶をクリックして実行すると、下の黒いセルに実行結果が表示されます¹。

この本では、これ以降、次のように2つの四角形を並べ、上の色付きのセルにコードを記載し、下のセルに実行結果を記載することにします。

例えば、上の図のコードは次のように記載するということです。

<code>print('hello world')</code>
hello world

ここからは、この本に記載のあるコードを写経 (手入力しながら学ぶこと) するか、もしくはサポートサイトにあるコードをコピーして、ノートブックで動かしながら読みましょう。

3.2.2 ノートブックでサブミットする方法

¹ ノートブックはJupyterノートブックとほぼ同様に使えるので、Jupyterノートブックを使用したことがある方なら、簡単に使えるでしょう。

ノートブックでサブミットする手順

1. 提出データを作成するコードを書く
2. 「commit & Run」をクリックするとコードが実行され、outputタブで確認可
3. 「Submit to Competition」をクリックし提出

上の図は、カーネル環境のノートブックでサブミットする手順です。順に説明します。

1. 提出データを作成するコードを書く

予測したデータを、指定された提出フォーマットにするコードを書きます。

今回は、既にgender_submission.csvがサンプルデータとして提供されているので、pandasを使い読み込んだ上でそのまま書き出します。

```
import pandas as pd # pandasのインポート

# csvデータの読み込み
df_gender_submission = pd.read_csv('../input/gender_submission.csv')
```

```
# df_gender_submissionを、gender_submission.csvに書き出し
df_gender_submission.to_csv('gender_submission.csv', index=False)
```

2. commit & Runをクリックしコードを実行

csvファイルを書き出すコードが出来上がったので、右上にある「Commit & Run」をクリックします。

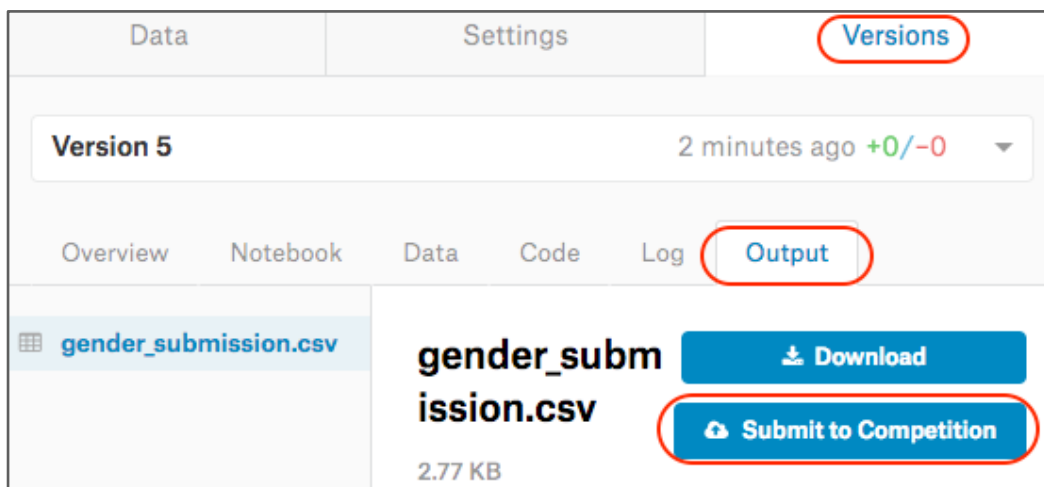


コードが実行され、「Versions」タブ内の「output」タブに

「gender_submission.csv」が表示されます。

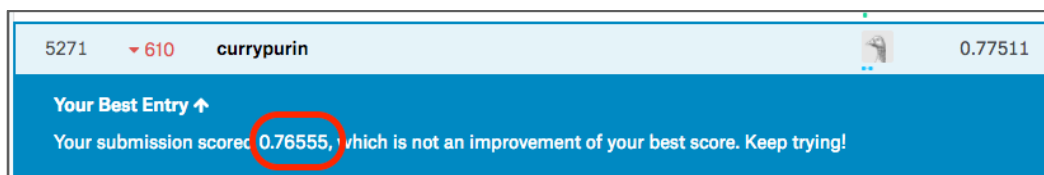
3. Submit to Competitionをクリックしサブミット

サブミットするcsvファイルが出来上がったので、「Submit to Competition」をクリックし、コンペにデータをサブミットします。



データをサブミットすると、次のように、スコアが表示されます。今回のスコアは0.76555であることがわかります。

タイタニックコンペの場合は、2.1.2で確認したようにaccuracy (正しく生存を予測した乗客の割合)がスコアになるので、男性を死亡と予測し、女性を生存と予測する単純なモデルで76.6%もの正解率ということです。次章以降ではこのスコアを超える予測モデルを作成することが目標になります。



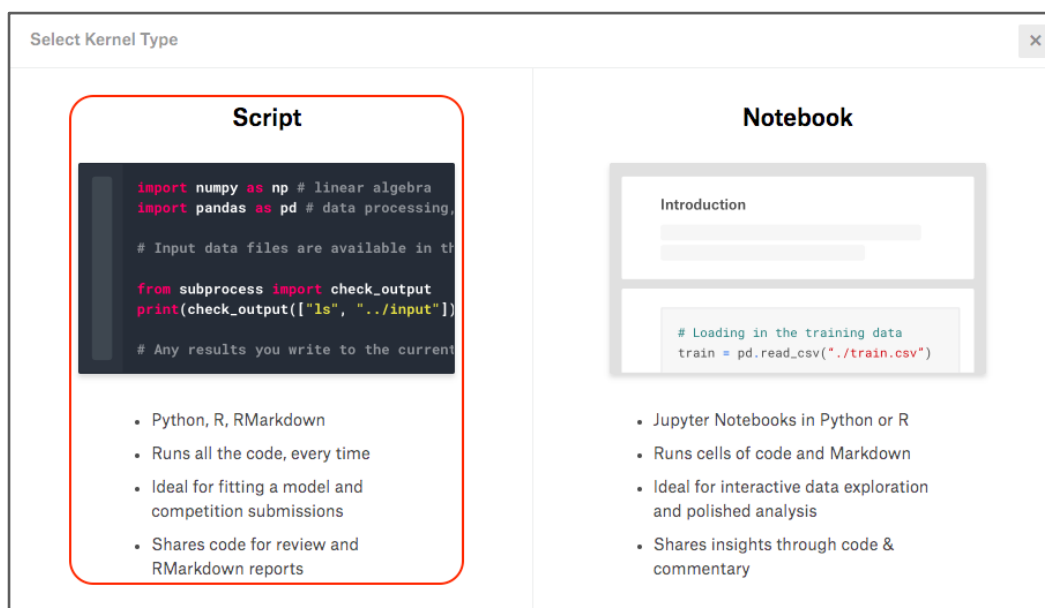
なお、上部に表示されているスコアが0.7751で5271位というのは、自分のベストスコアであり、今回サブミットした点数ではありません。

サブミットしたデータがベストスコアでない場合は、「Your submission scored 0.76555, which is no an improvement of your best score. Keep trying」と表示されているように、「今回のスコアはベストスコアじゃないので頑張ろう」という表示がされます。

3.3 スクリプトでサブミットする方法

前節では、ノートブックを用いてデータのサブミットをしましたが、この節では、スクリプトを用いる方法を説明します。

スクリプトを用いる場合は、新しいカーネルを作成する際に、Notebookではなく、Scriptを選択します。



次のような、コード入力画面が表示されるので、Scriptではここにコードを入力します。

なお、スクリプトでは、普段は拡張子が「.py」のファイルに記載しているPythonのコードを、このコード入力欄に入力して実行します。

```
1 # This Python 3 environment comes with many helpful analytics libraries installed
2 # It is defined by the kaggle/python docker image: https://github.com/kaggle/docker
3 # For example, here's several helpful packages to load in
4
5 import numpy as np # linear algebra
6 import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
7
8 # Input data files are available in the "../input/" directory.
9 # For example, running this (by clicking run or pressing Shift+Enter) will list the files in the input directory
10
11 import os
12 print(os.listdir("../input"))
13
14 # Any results you write to the current directory are saved as output.
```

スクリプトも手順は先ほどのノートブックと同様で、「提出データを作成するコードを書く」、「commit & Runをクリックしコードを実行」、「Submit to Competitionをクリックしサブミット」の3段階となります。

今回は、乗客の生存を全て0 (全員が亡くなった)と予測するサブミットファイルを作ってみます。Pythonのコードでは次のように書くことができます。

```
# pandasとnumpyをインポート
import pandas as pd
import numpy as np

# テストデータの読み込み
df_test = pd.read_csv('../input/test.csv')

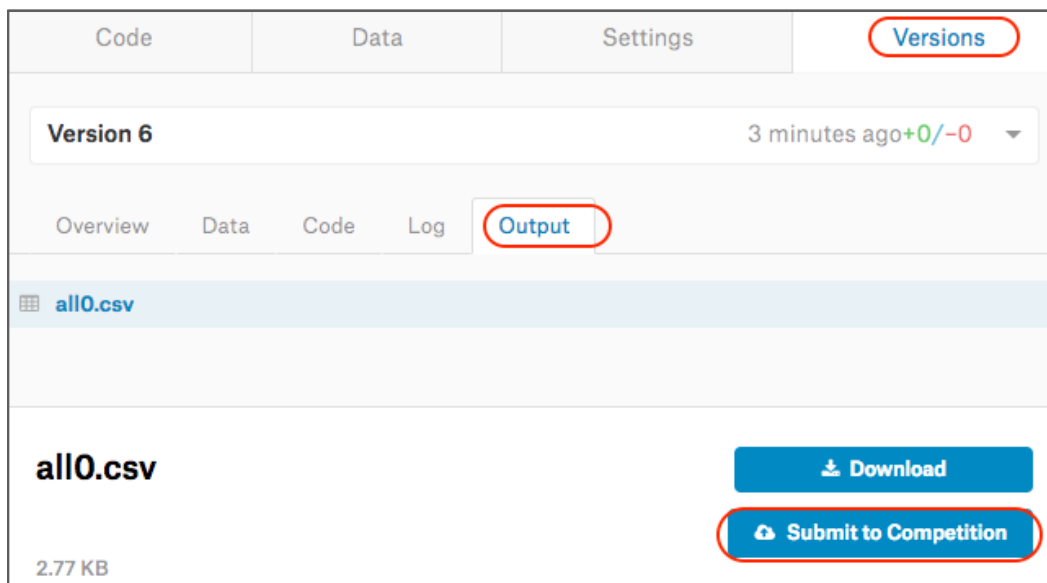
# テストデータのPassengerIdをnumpyの配列として取得
PassengerId = np.array(df_test['PassengerId']).astype(int)

# 予測データを作成 (0が418個並ぶnumpyの配列)
all_zeros = np.zeros(418, dtype = int)

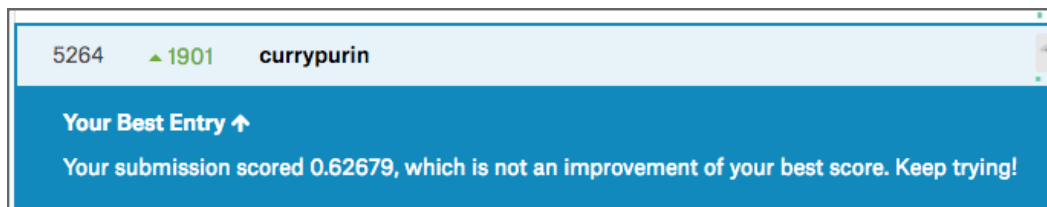
# 予測データ(all_zeros)とPassengerIdからデータフレームを作成
my_solution = pd.DataFrame(all_zeros, PassengerId, columns = ['Survived'])

# all0.csvとして書き出し
my_solution.to_csv('all0.csv', index_label = ['PassengerId'])
```

コードを入力したら、ノートブックと同じように、右上の「Commit & Run」をクリックします。そうするとコードが実行され、次のようにall0.csvが「Versions」タブの「Output」タブに表示されます。「Submit to Competition」 ボタンを押してサブミットしましょう。



全てを0 (死亡)という予測を提出するとスコアが62.679%だということがわかりました。



3.4 ローカルPCで作成したデータをサブミットする方法

最後は、これまでとは違い、ローカルPCで提出データを作成する方法です。カーネル環境は便利ですが、自分が使いたい言語・ライブラリが使えない、ログを残しておきたい等の理由により、ローカルPCで提出データを作りたい時があります。提出データができあがった後、どのようにサブミットするか説明します。

タイタニックのページの1番右のタブの「Submit Predictions¹」を開きます。Step1の右側にUpload Submission Fileという箇所があるので、そこにサブミットするデータをドラッグ&ドロップします。

¹ <https://www.kaggle.com/c/titanic/submit>

OverviewDataKernelsDiscussionLeaderboardRulesTeamMy SubmissionsSubmit Predictions

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
all0.csv	26 minutes ago	1 seconds	0 seconds	0.62679


Complete

Jump to your position on the leaderboard ▾

Make a submission for [currypurin](#)








You have 7 submissions remaining today. This resets 10 hours from now (00: 00 UTC).

Step 1
Upload submission file


Upload Submission File

Step2と表示された右側にある「briefly describe your submission」と表示されている箇所にこのサブミットの説明を入力し、「Make Submission」をクリックします。

Step 2
Describe submission

B *I* % “ < >     H    Styling with Markdown supported

Briefly describe your submission.

Make Submission

これまでと同じように、結果が表示されます。

この章では、3種類のサブミットの方法があることがわかりました。
いよいよ次章から実際のデータを使って分析をしていきます。

第2部

第4章 タイタニックデータの概要

2章では、Kaggleのページに英語で記載されている説明 (の翻訳)を読んで、コンペの概要をつかむことができました。

この章では、可視化ライブラリのmatplotlibとseabornを使うことにより、タイタニックデータの詳細を把握することを目指します。

4.1 ライブラリのインポートとデータの読み込み

3.2と同じように、カーネル環境で、新しいノートブックを作成します。

ノートブックを作成すると、1番上のセル (四角形の枠内)に、次のようなコードが表示されています。このコードを実行してみましょう。

```
# This Python 3 environment comes with many helpful analytics libraries
installed
# It is defined by the kaggle/python docker image: https://github.com/
kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list
the files in the input directory

from subprocess import check_output
print(check_output(["ls", "../input"]).decode("utf8"))

# Any results you write to the current directory are saved as output.

gender_submission.csv
test.csv
train.csv
```

このコードは、機械学習でよく使われる数値計算ライブラリであるnumpyと

データ解析ライブラリpandasをインポートし (使える状況にし)、inputフォルダに格納されているファイルを表示する命令をするコードです。

実行すると2.2のDataで確認した3ファイルが表示されました。

次にこの3つのcsvファイルを、pandasのread_csvメソッドを使い「データフレーム¹」として読み込みます。

```
# df_train、df_testとしてそれぞれ読み込み
df_train = pd.read_csv('../input/train.csv')
df_test = pd.read_csv('../input/test.csv')
df_gender_submission = pd.read_csv('../input/gender_submission.csv')
```

matplotlib、pyplot、seabornも以降の可視化で使用するため、インポートしておきます。

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
```

4.2 データの概要を確認する

データの分析をする時は、初めは細部に入り込まずに、どのようなデータかという大枠を把握しながら、徐々に細部の確認をしていくことが重要になります。

4.2.1 データフレームについて

データフレームとは、2次元 (行×列)のデータであり、イメージとしてはExcelシートのデータのようなものです。

データフレームを、列方向²にみると、同じ特徴量のデータが並び (例えば、左から1列目にはAge列があり、年齢のデータが縦方向に並んでいます)、行方向にみるとある人のデータが並びます。

データフレームの中身を見てみましょう。データフレームは、`df.head(n)`で、nに好きな数字をいれると、最初のn行が表示されます。

¹ 4.2.1 参照

² 列方向は縦方向、行方向は横方向です。

df_train.head(5)

	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	SibSp	Survived	Ticket
0	22.0	NaN	S	7.2500	Braund, Mr. Owen Harris	0	1	3	male	1	0.0	A/5 21171
1	38.0	C85	C	71.2833	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	2	1	female	1	1.0	PC 17599
2	26.0	NaN	S	7.9250	Heikkinen, Miss. Laina							
3	35.0	C123			Heath y Peel)	0	4	1	female	1	1.0	113803
4	35.0	NaN			Henry	0	5	3	male	0	0.0	373450

1行目 (ある人のデータ)

1列目 (Age列)

df_trainの初めの5行が表示されました。

このデータフレームは、5人のデータが並び、例えば1番上の人であれば、Ageが22.0、CabinがNaN、EmbarkedがSなどということを表しています。これがデータフレームです。

4.2.2 データフレームの行数と列数を確認

df.shapeにより、データフレームの行数と列数を確認することができます。

```
print(df_train.shape) # 学習用データ
print(df_test.shape) # 本番予測用データ
print(df_gender_submission.shape) # 提出データのサンプル
```

(891, 12)

(418, 11)

(418, 2)

それぞれ次の行数と列数を持つデータ構造だということがわかりました。

- df_train : 891行×12列
- df_test : 418行×11列
- df_gender_submission : 418行×2列

本番予測用データであるdf_testは、学習用データであるdf_trainよりも正解(生存)の列が1列少なくなっています。

学習用データにより学習したモデルで、本番予測用データの生存予測をするためです。

4.2.3 列の名前の確認

df.columnsにより、列の名前を確認します。

```
print(df_train.columns) # トレーニングデータの列名
print('-*10) # 区切りを挿入
print(df_test.columns) # テストデータの列名

Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
      'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
-----
Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare',
      'Cabin', 'Embarked'],
      dtype='object')
```

df_trainには、「PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked」の12列が含まれることがわかります。

また、df_testは、予測を行う「Survived」の列がないものの、他はdf_trainと同じことがわかります。

4.2.4 df.info()で概要の確認

df_train.info()により、各列の欠損値 (NaN)¹の数とデータの型がわかります。

¹ 欠損値については、4.2.6で説明します。

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64   ← 乗客ID
Survived       891 non-null int64   ← 生存
Pclass         891 non-null int64   ← チケットクラス
Name           891 non-null object  ← 名前
Sex            891 non-null object  ← 性別
Age           714 non-null float64 ← 年齢
SibSp          891 non-null int64   ← 同乗した兄弟または配偶者の人数
Parch          891 non-null int64   ← 同乗した親または子供の人数
Ticket         891 non-null object  ← チケット番号
Fare           891 non-null float64 ← 運賃
Cabin          204 non-null object  ← キャビン番号
Embarked       889 non-null object  ← 乗船港
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

各行末に記載されているint64、float64、objectなどとの記載でデータフレームの各列のデータ型 (dtype)がint型 (整数)かfloat型 (小数)かbool型 (True or False)、object型 (string型など)等がわかります。

dtypes: float64(2), int64(5), object(5)と表示されたことから、float型が2つ、int型が5つ、object型が5つであることがわかります。

同様にdf_test.info()により、df_testの概要を確認します。

```
df_test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age           332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

df_testには含まれないSurvived列を除き、df_trainと同じ型のデータが含まれることがわかります。

4.2.5 df.head()で概要の確認

再度、df.head()を使い、はじめの5行を確認します。生のデータを見ることで、df.info()では分からなかったことがわかります。

df_train.head()													
	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	SibSp	Survived	Ticket	
0	22.0	NaN	S	7.2500	Braund, Mr. Owen Harris	0	1	3	male	1	0.0	A/5 21171	
1	38.0	C85	C	71.2833	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	2	1	female	1	1.0	PC 17599	
2	26.0	NaN	S	7.9250	Heikkinen, Miss. Laina	0	3	3	female	0	1.0	STON/O2. 3101282	
3	35.0	C123	S	53.1000	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	4	1	female	1	1.0	113803	
4	35.0	NaN	S	8.0500	Allen, Mr. William Henry	0	5	3	male	0	0.0	373450	

例えば

- ・Embarkedには1文字の大文字のアルファベットが入っていること、
- ・Nameには、Mr、Mrs、Missという敬称が含まれること、
- ・Ticketには、アルファベットや数字が含まれそうなこと

など多くのことがわかります。

4.2.6 欠損値がいくつあるか確認

欠損値¹とは、データ項目の一部が欠けていることで、機械学習で予測をするためには、何らかの処理をする必要があります。なお、欠損値の処理は、5.1.1で説明します。

`df_test.isnull().sum()`により欠損値の数をカウントします。

```
df_train.isnull().sum()
# isnull()は、欠損値に対しTrueを返し、欠損値以外にはFalseを返す
# sum()は、Trueを1、Falseを0として合計する
# よってdf.isnull().sum()で欠損値を算出することができる
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age          177
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin         687
Embarked       2
dtype: int64
```

欠損値の数は、多い順にCabinが687、Ageが177、Embarkedが2、であることがわかります。

同様に`df_test`についても欠損値の数を確認します。

¹ Pandasのデータフレームでの欠損値はNaN (Not a Number)と表示されます。

df_test.isnull().sum()	
PassengerId	0
Pclass	0
Name	0
Sex	0
Age	86
SibSp	0
Parch	0
Ticket	0
Fare	1
Cabin	327
Embarked	0
dtype: int64	

df_testの欠損値は、Cabinが327、Ageが86、Fareが1であることがわかります。

4.2.7 要約統計量の表示

df_train.describe()により、数値データのための、データ数、平均値、標準偏差、最小値、最大値、四分位数 (データを大きさ順に並べた時に25%、50%、75%の位置にくる値)が表示されます。

ここでは、df_trainとdf_testをpd.concat()により縦に連結し、df_fullを作成してから、要約統計量を表示してみます。

```
# df_trainとdf_Testを縦に連結
df_full = pd.concat([df_train, df_test], axis = 0, ignore_index=True)

print(df_full.shape) # df_fullの行数と列数を確認

df_full.describe() # df_fullの要約統計量
```

(1309, 12)							
	Age	Fare	Parch	PassengerId	Pclass	SibSp	Survived
count	1046.000000	1308.000000	1309.000000	1309.000000	1309.000000	1309.000000	891.000000
mean	29.881138	33.295479	0.385027	655.000000	2.294882	0.498854	0.383838
std	14.413493	51.758668	0.865560	378.020061	0.837836	1.041658	0.486592
min	0.170000	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000
25%	21.000000	7.895800	0.000000	328.000000	2.000000	0.000000	0.000000
50%	28.000000	14.454200	0.000000	655.000000	3.000000	0.000000	0.000000
75%	39.000000	31.275000	0.000000	982.000000	3.000000	1.000000	1.000000
max	80.000000	512.329200	9.000000	1309.000000	3.000000	8.000000	1.000000

これにより以下のことがわかります。

- Survived列から、生存した人は4割弱であること。
- Age列から、乗客の平均年齢は約29.9歳、最大は80歳、最小は0.17歳であること。

なお、数値データ以外も表示したいときは、`df.describe(include = 'all')`と書くことで全てのデータについての要約統計量を表示することができます。

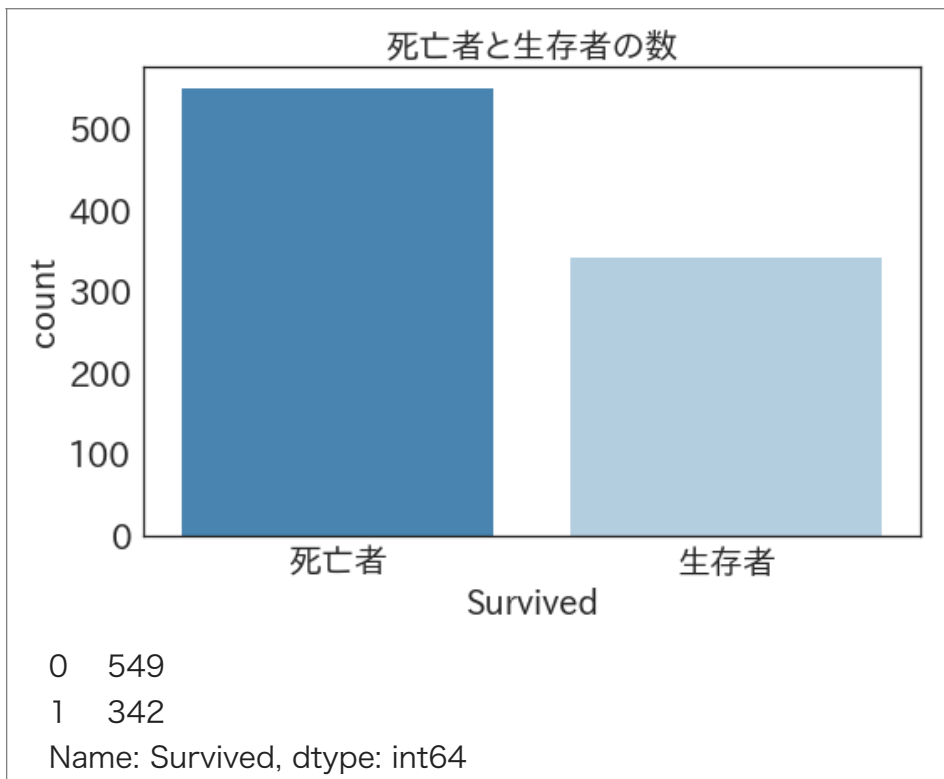
df_full.describe(include = 'all')												
	Age	Cabin	Embarked	Fare	Name	Parch	PassengerId	Pclass	Sex	SibSp	Survived	Ticket
count	1046.000000	295	1307	1308.000000	1309	1309.000000	1309.000000	1309.000000	1309	1309.000000	891.000000	1309
unique	NaN	NaN	186	3	NaN	1307	NaN	NaN	NaN	2	NaN	929
top	NaN	C23 C25 C27	S	NaN	Connolly, Miss. Kate	NaN	NaN	NaN	male	NaN	NaN	CA. 2343
freq	NaN	6	914	NaN	2	NaN	NaN	NaN	843	NaN	NaN	11
mean	29.881138	NaN	NaN	33.295479	NaN	0.385027	655.000000	2.294882	NaN	0.498854	0.383838	NaN
std	14.413493	NaN	NaN	51.758668	NaN	0.865560	378.020061	0.837836	NaN	1.041658	0.486592	NaN
min	0.170000	NaN	NaN	0.000000	NaN	0.000000	1.000000	1.000000	NaN	0.000000	0.000000	NaN
25%	21.000000	NaN	NaN	7.895800	NaN	0.000000	328.000000	2.000000	NaN	0.000000	0.000000	NaN
50%	28.000000	NaN	NaN	14.454200	NaN	0.000000	655.000000	3.000000	NaN	0.000000	0.000000	NaN
75%	39.000000	NaN	NaN	31.275000	NaN	0.000000	982.000000	3.000000	NaN	1.000000	1.000000	NaN
max	80.000000	NaN	NaN	512.329200	NaN	9.000000	1309.000000	3.000000	NaN	8.000000	1.000000	NaN

4.2.8 死亡者と生存者の可視化

seabornのcountplot関数を使用して、df_trainの死亡者と生存者別に集計し可視化します。

```
sns.countplot(x='Survived', data=df_train)
plt.title('死亡者と生存者の数')
plt.xticks([0,1],['死亡者', '生存者'])

# Survived列の値を集計
df_train['Survived'].value_counts()
```

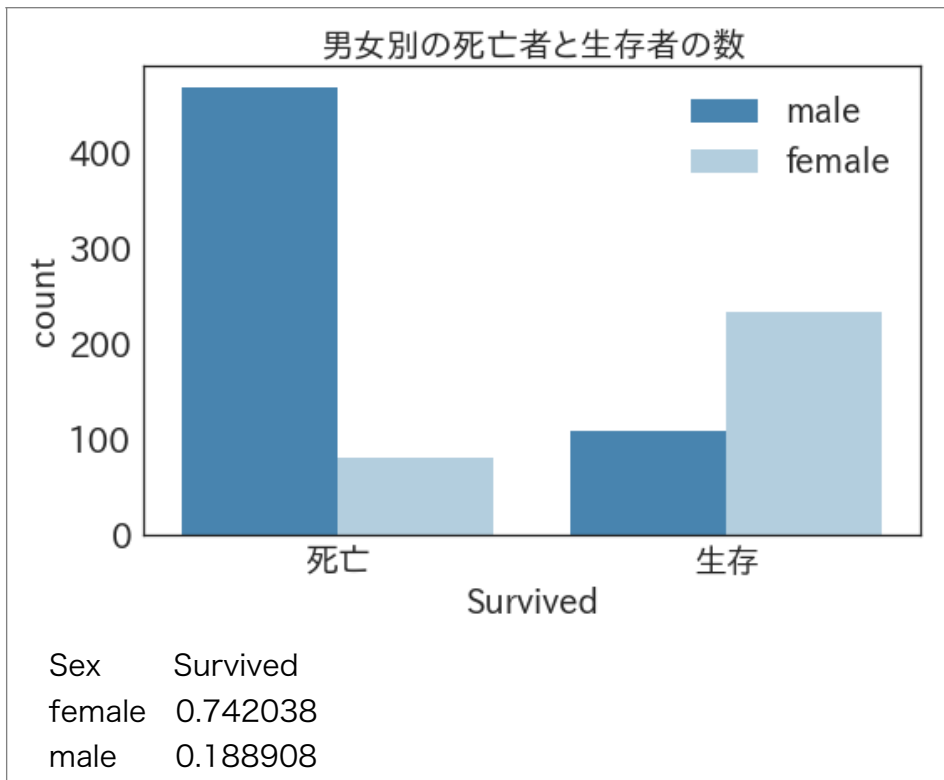



グラフにすることで、死亡者と生存者の割合を見ることができます。またSurvived列を集計し死亡者数と生存者数を確認します。

次に男女別の死亡者と生存者数を確認します。

```
# 男女別の生存者数を可視化
sns.countplot(x='Survived', hue='Sex', data=df_train)
plt.xticks([0.0, 1.0], ['死亡', '生存'])
plt.title('男女別の死亡者と生存者の数')

# 男女別の生存割合を表示する
df_train[['Sex', 'Survived']].groupby(['Sex']).mean()
```



男性は、死亡者が生存者の約5倍ですが、女性は反対に生存者が死亡者の2倍以上だということがわかります。

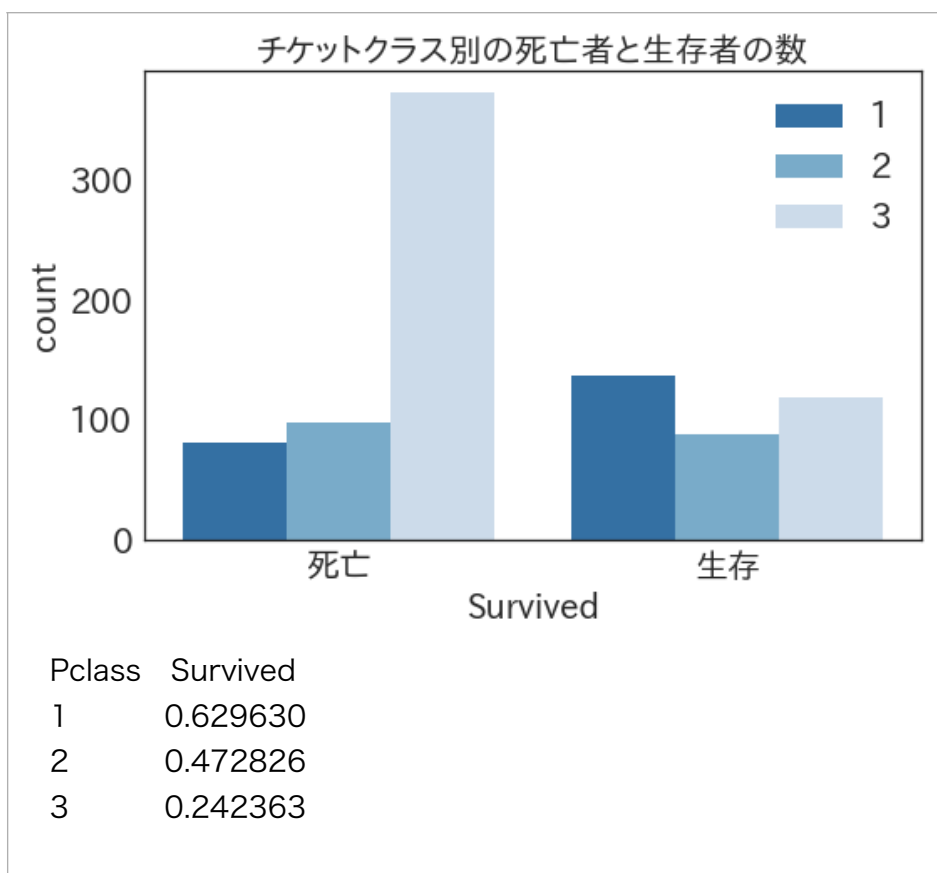
映画を見たことがある方ならわかると思いますが、タイタニック号の事故の際、女性や子供から先に、船から脱出が行われました。映画の知識と同じく、女性の方が生存者の割合が多いことがわかりました。

4.2.9 チケットクラス

次にチケットクラス別の死亡者と生存者数を確認します。

```
# チケットクラス別の生存者数を可視化
sns.countplot(x='Survived', hue='Pclass', data=df_train)
plt.xticks([0.0, 1.0], ['死亡', '生存'])

# チケットクラス別の生存割合を表示する
df_train[['Pclass', 'Survived']].groupby(['Pclass']).mean()
```



1stクラスの乗客は生存者が死亡者の2倍弱いますが、3rdクラスの乗客は、1/3程度とチケットクラスにより生存割合に大きく差があることがわかります。

4.2.10 年齢の分布

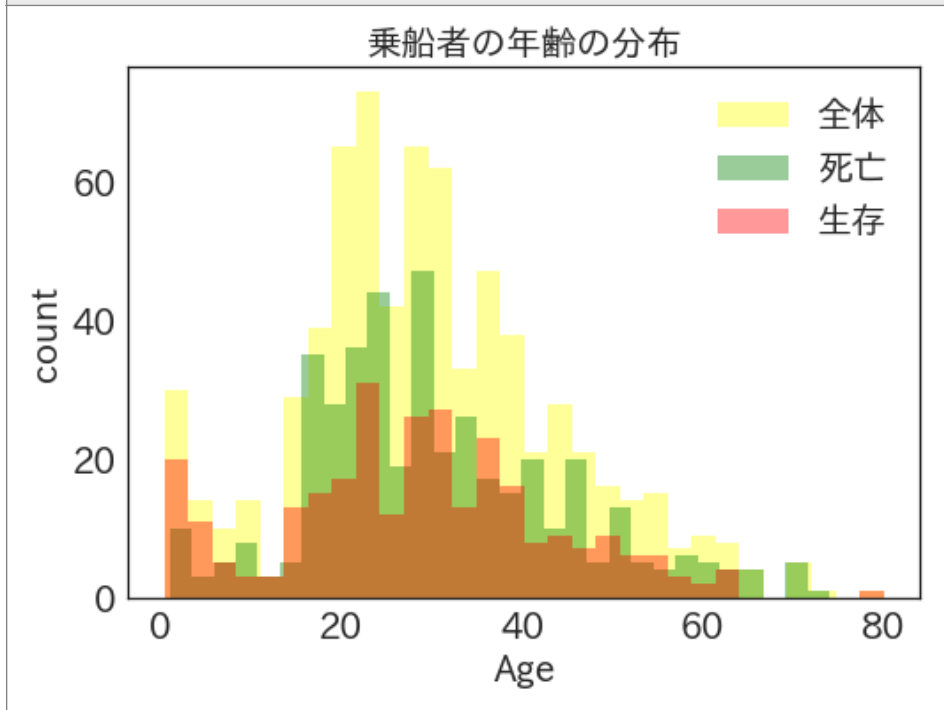
乗船者の年齢の分布を確認するために、年齢のヒストグラムを描きます。

```
# 全体のヒストグラム
sns.distplot(df_train['Age'].dropna(), kde=False, bins = 30 ,label='全体')

# 死亡者のヒストグラム
sns.distplot(df_train[df_train['Survived'] == 0].Age.dropna(), kde = False,\
bins=30, label='死亡')

# 生存者のヒストグラム
sns.distplot(df_train[df_train['Survived'] == 1].Age.dropna(), kde = False,\
bins=30, label='生存')

plt.title('乗船者の年齢の分布') # タイトル
plt.legend() # 凡例を表示
```



全体のヒストグラムから、乳幼児も数十人が乗船していたこと、10代後半から30代にかけての乗客が多かったことがわかります。

また、死亡者と生存者のヒストグラムから、10歳未満の乗客は死亡者よりも生存者の方が多いが、10代後半から30代にかけては生存者より死亡者の方が多いことがわかります。

また、pandasのcutメソッドを用いて、年齢を等間隔に8等分し、生存率を

表示してみます。

```
# 年齢を8等分し、CategoricalAgeという変数を作成
df_train['CategoricalAge'] = pd.cut(df_train['Age'], 8)

# CategoricalAgeでグルーピングして、Survivedを平均
df_train[['CategoricalAge', 'Survived']].groupby(['CategoricalAge'],
as_index=False).mean()
```

CategoricalAge	Survived
(0.34, 10.368]	0.593750
(10.368, 20.315]	0.382609
(20.315, 30.263]	0.365217
(30.263, 40.21]	0.445161
(40.21, 50.158]	0.383721
(50.158, 60.105]	0.404762
(60.105, 70.052]	0.235294
(70.052, 80.0]	0.200000

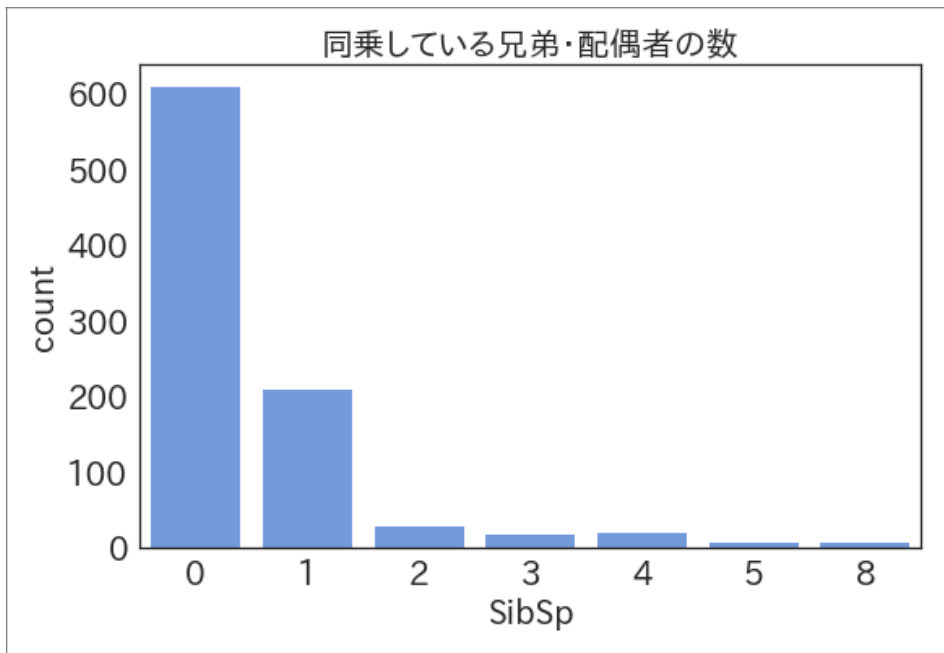
例えば1番上の(0.34,10.368]の行は、0.34歳超から10.368歳までのグループを集計すると、生存者と死亡者の割合が約0.594という意味になります。

1番若いカテゴリは生存者の割合が高いことなどがわかりました。

4.2.11 タイタニック号に乗っている兄弟・配偶者の数

次に、乗船している兄弟・配偶者の数についてのヒストグラムを描いてみます。

```
sns.countplot(x='SibSp', data = df_train)
```



兄弟または配偶者1人と乗船していた者は約200人おり、2人以上と同乗している者は少ないことがわかります。

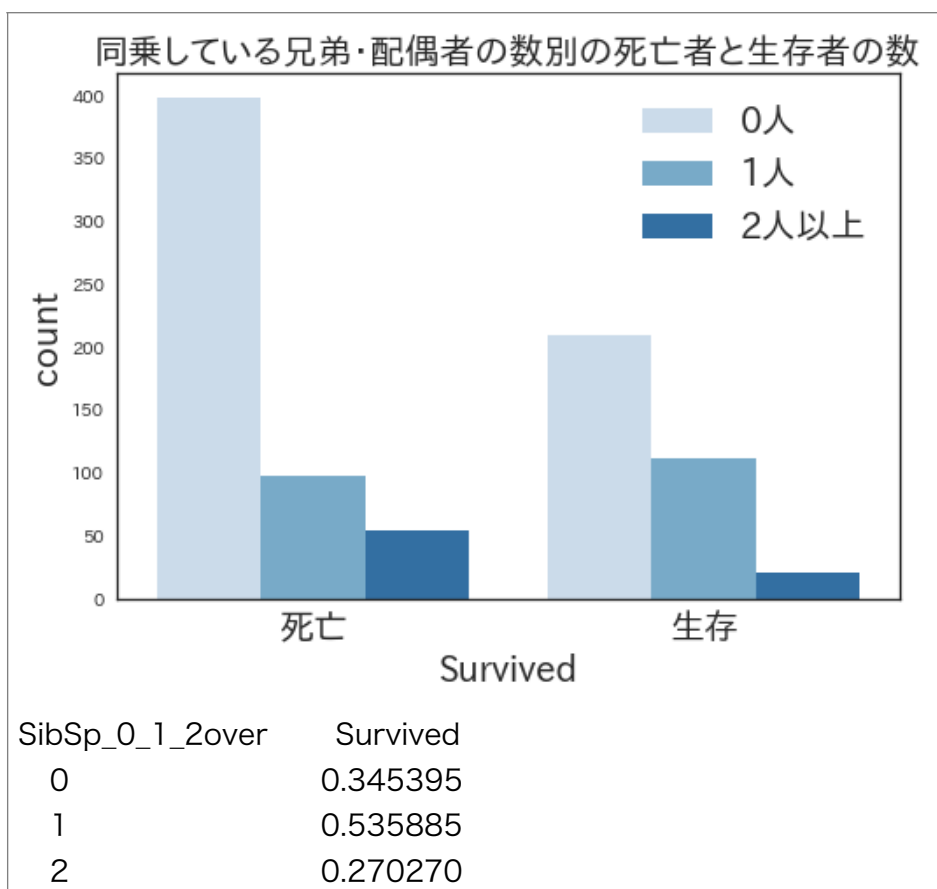
同乗している兄弟・配偶者の数と、生存・死亡との関係を表示してみます。なお、同乗している兄弟・配偶者の数が2人以上の人は数が少ないため、2人以上として集計しています。

```
# SibSpが0か1であればそのまま、2以上であれば2である特徴量SibSp_0_1_2over
# を作成
df_train['SibSp_0_1_2over'] = [i if i <=1 else 2 for i in df_train['SibSp']]

# SibSp_0_1_2overごとに集計し、可視化
sns.countplot(x='Survived', hue='SibSp_0_1_2over', data=df_train)

plt.legend(['0人','1人','2人以上'],loc = 'upper right')
plt.title('同乗している兄弟・配偶者の数別の死亡者と生存者の数')

df_train[['SibSp_0_1_2over','Survived']].groupby(['SibSp_0_1_2over']).mean()
```

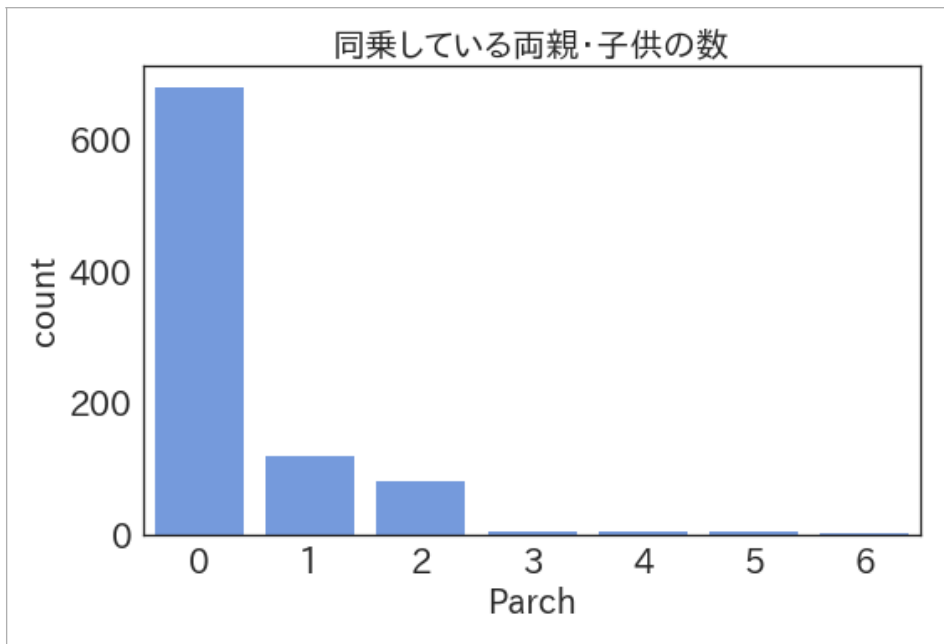


同乗している兄弟・配偶者が1人の方が生存者の割合が高く、その次に高いのが0人、2人以上は生存者の割合が低いということがわかりました。

4.2.12 タイタニック号に乗っている両親・子供の数

兄弟・配偶者の数と同じように、両親・子供の数についても、可視化してみます。

```
sns.countplot(x='Parch', data = df_train)
plt.title('同乗している両親・子供の数')
```



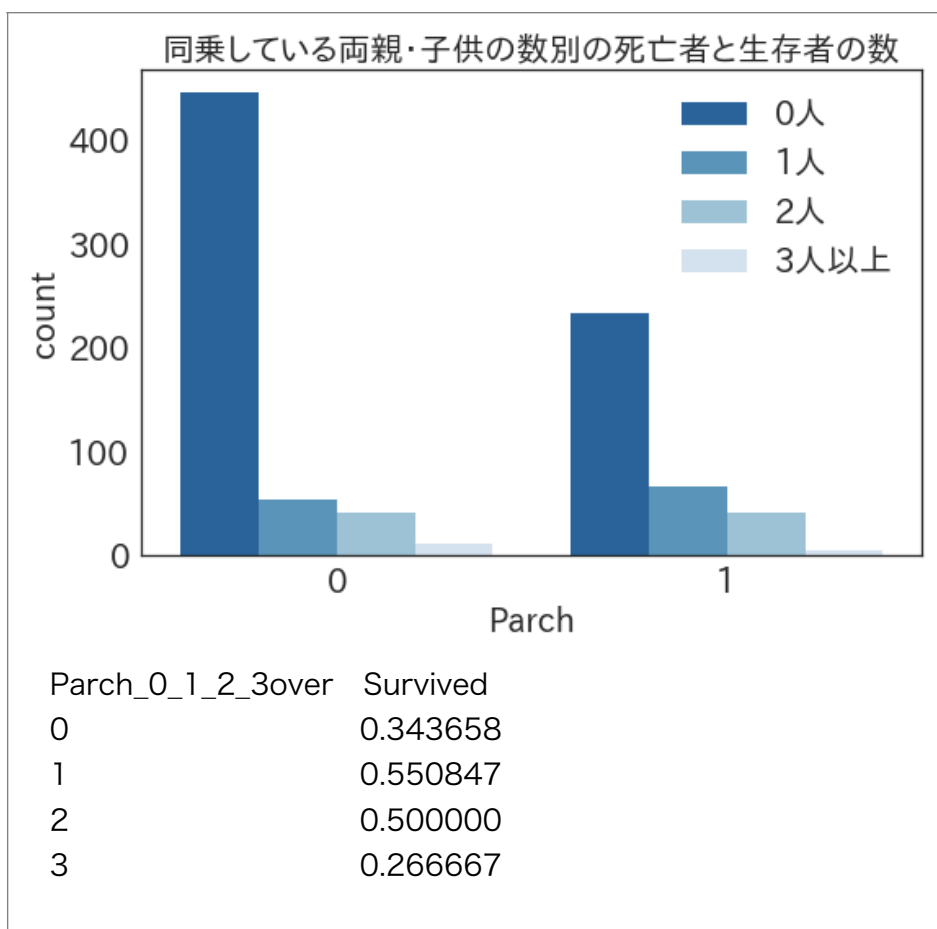
3人以上の両親又は子供と乗っている人はほとんどいません。
同乗している両親・子供の数と生存者・死亡者のとの関係を表示してみます。

```
# 2以下であればそのままの数、3以上は3という変換を行う
df_train['Parch_0_1_2_3over'] = [i if i <=2 else 3 for i in\
df_train['Parch']]

# Parch_0_1_2_3overごとに集計し可視化
sns.countplot(x='Survived', hue='Parch_0_1_2_3over', data =\
df_train)

plt.title('同乗している両親・子供の数別の死亡者と生存者の数')
plt.legend(['0人', '1人', '2人', '3人以上'])

df_train[['Parch_0_1_2_3over', 'Survived']].groupby(['Parch_0_1_2_\
3over']).mean()
```

同乗している両親・子供が1人または2人の場合は、0人の場合よりも生存者の割合が高く、3人以上は1番低いということがわかります。

4.2.13 1人で乗船しているか2人以上で乗船しているか

4.2.11と4.2.12から家族と乗船せずに1人で乗船している者が多数おり、その生存率が低いことがわかりました。

そこで、SibSpとParchを合計した上で、1人で乗船しているという特徴量を作り、可視化してみます。

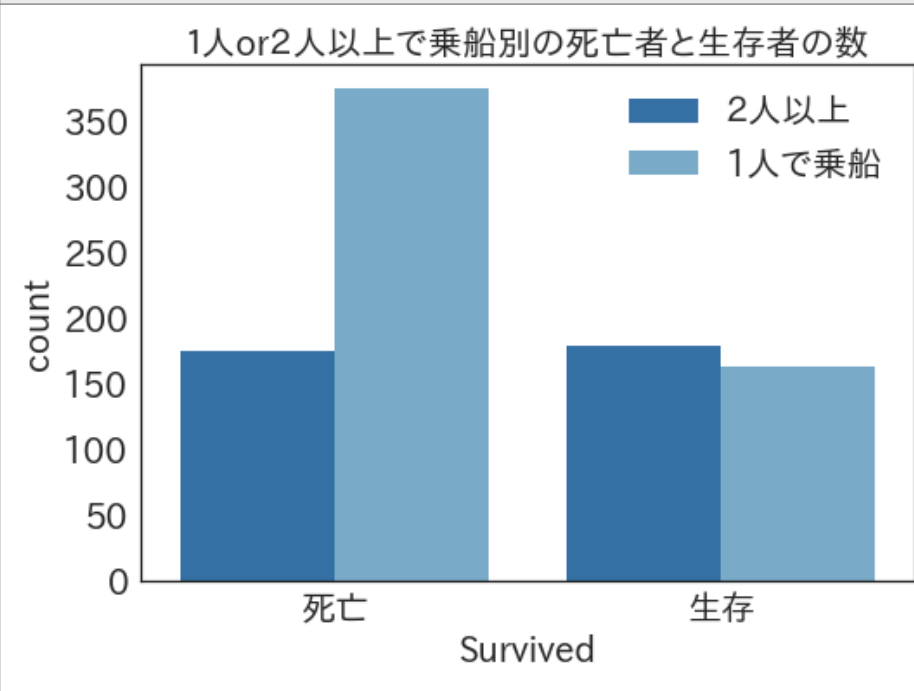
```
#SibSpとParchが同乗している家族の数。1を足すと家族の人数となる
df_train['FamilySize']=df_train['SibSp']+ df_train['Parch']+ 1

# IsAloneを0とし、2行目でFamilySizeが1であれば1にしている
df_train['IsAlone'] = 0
df_train.loc[df_train['FamilySize'] == 1, 'IsAlone'] = 1

# IsAloneごとに可視化
sns.countplot(x='Survived',hue='IsAlone',data=df_train)

plt.legend(['2人以上','1人で乗船'])
plt.xticks([0,1],['死亡','生存'])

plt.title('1人or2人以上で乗船別の死亡者と生存者の数')
```



1人で乗船している者は2人以上と比べて、死亡者の割合が高いことがわかります。

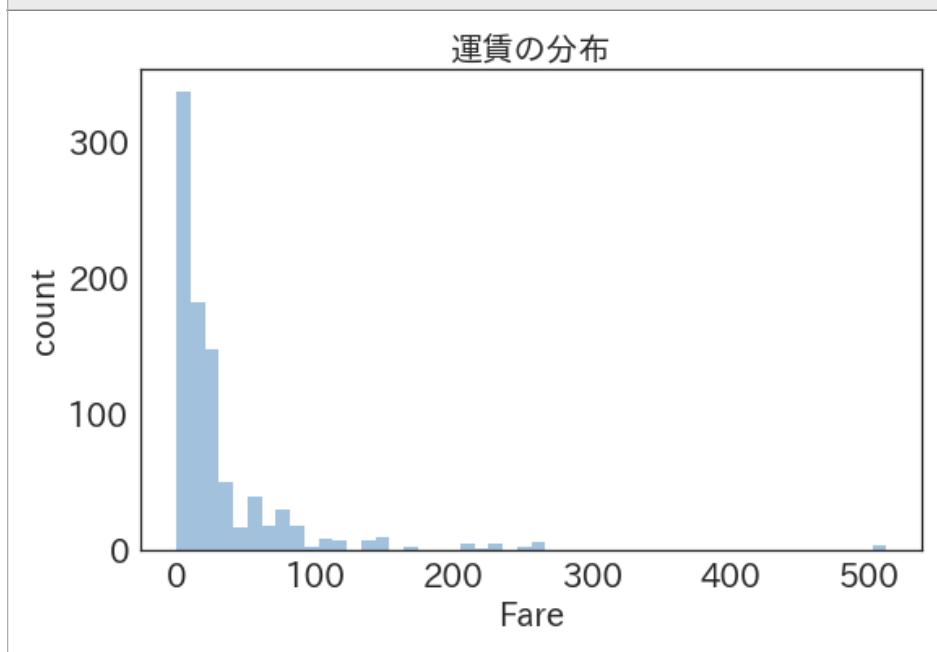
このように既存の複数の特徴量から新しい特徴量を作成し、その特徴量を使用したモデルを作成することで、良いモデルになることがあります。¹

¹ この特徴量は、5.3.4で使用します。

4.2.14 運賃の分布

乗船者の運賃の分布を描いてみます。

```
sns.distplot(df_train['Fare'].dropna(), kde=False, hist=True)
```



多くの人の運賃は0～100に固まっており、300を超える運賃は、ほとんどないということがわかりました。

pandasのqcutメソッドを用いて、運賃を各カテゴリの人数が等しくなるように分割し生存者の割合を確認します。

```
df_train['CategoricalFare'] = pd.qcut(df_train['Fare'], 4)
df_train[['CategoricalFare', 'Survived']].groupby(['CategoricalFare'],\
as_index=False).mean()
```

CategoricalFare Survived

(-0.001, 7.91]	0.197309	← 運賃が-0.001から7.91のカテゴリ
(7.91, 14.454]	0.303571	← 運賃が7.91から14.454のカテゴリ
(14.454, 31.0]	0.454955	← 運賃が14.454から31.0のカテゴリ
(31.0, 512.329]	0.581081	← 運賃が31.0から512.329のカテゴリ

運賃が高くなるにつれ、生存者の割合が高くなってきていることがわかります。

qcut¹メソッドは、2つ目の引数の値により、いくつかのカテゴリに分割するかを指定します。今回は4分割しましたが、数値を変更し試してみて、自分が表現したいカテゴリになるよう最適な分割数を探します。

4.2.15 名前

Name列は乗船者の名前のデータであり、機械学習で特徴量として使うためには、何らかの変換をしてやる必要があります。

そのために、初めの5人の名前を確認してみます。

```
df_test['Name'][0:5]
Kelly, Mr. James
Wilkes, Mrs. James (Ellen Needs)
Myles, Mr. Thomas Francis
Wirz, Mr. Albert
Hirvonen, Mrs. Alexander (Helga E Lindqvist)
Name: Name, dtype: object
```

真ん中に、Mr.やMrs.という敬称が含まれていることがわかります。

敬称は、「大文字のアルファベット+小文字のアルファベット+. (ドット)」で構成されるので、正規表現を使用して次のように抽出することができます。

```
# 敬称を抽出し、重複を省く
set(df_train.Name.str.extract('([A-Za-z]+)\.', expand=False))
```

¹ 類似のメソッドに4.2.10で使用したcutメソッドがある。cutメソッドは各カテゴリが等間隔になるように分割するメソッド。

```
{'Capt',  
'Col',  
'Countess',  
'Don',  
'Dr',  
'Jonkheer',  
'Lady',  
'Major',  
'Master',  
'Miss',  
'Mlle',  
'Mme',  
'Mr',  
'Mrs',  
'Ms',  
'Rev',  
'Sir'}
```

setを使い、重複を除くことで、17種類の敬称が含まれることがわかります。

それぞれの敬称がいくつ含まれるか数え上げます。

```
# collections.Counterを使用して、数え上げる  
import collections  
collections.Counter(df_train.Name.str.extract('([A-Za-z]+\.)\.', expand=False))
```

```
Counter({'Capt': 1,  
        'Col': 2,  
        'Countess': 1,  
        'Don': 1,  
        'Dr': 7,  
        'Jonkheer': 1,  
        'Lady': 1,  
        'Major': 2,  
        'Master': 40,  
        'Miss': 182,  
        'Mlle': 2,  
        'Mme': 1,  
        'Mr': 517,  
        'Mrs': 125,  
        'Ms': 1,  
        'Rev': 6,  
        'Sir': 1})
```

Miss、Mr、Mrsというお馴染みの敬称は数が多いことがわかります。また、4番目に多い継承はMasterだということがわかります。

抽出した敬称をTitle列に入れ、年齢の平均値を算出してみます。

```
# df_trainにTitle列を作成、Title列の値は敬称
df_train['Title'] = df_train.Name.str.extract('([A-Za-z]+)\.', expand=False)

# df_testにTitle列を作成、Title列の値は敬称
df_test['Title'] = df_test.Name.str.extract('([A-Za-z]+)\.', expand=False)

# df_trainのTitle列の値ごとに平均値を算出
df_train.groupby('Title').mean()['Age']
```

Title	Age
Capt	70.000000
Col	58.000000
Countess	33.000000
Don	40.000000
Dr	42.000000
Jonkheer	38.000000
Lady	48.000000
Major	48.500000
Master	4.574167
Miss	21.773973
Mlle	24.000000
Mme	24.000000
Mr	32.368090
Mrs	35.898148
Ms	28.000000
Rev	43.166667
Sir	49.000000

Name: Age, dtype: float64

← 年齢が1桁であり子供の敬称と推測

Masterは平均年齢が約4.6才であることから、子供の敬称だということがわかりました。4.2.10で見たように、年齢により生存率に差があったため、有効な特徴量になりそうです。

例えば次のように、Masterを1、Missを2、Mrを3、Mrsを4に変換し、この4つ以外の敬称を5に変換すると、新しい特徴量¹として使えるようになります。

¹ この特徴量を用いた予測はこの本では扱いませんが、次章の改善点に例として記載しています。是非試してみてください。

ます。

```
# 変換するための関数を作成
def title_to_num(title):
    if title == 'Master':
        return 1
    elif title == 'Miss':
        return 2
    elif title == 'Mr':
        return 3
    elif title == 'Mrs':
        return 4
    else:
        return 5

# リスト内包表記1を用いて変換
df_train['Title_num'] = [title_to_num(i) for i in df_train['Title']]
df_test['Title_num'] = [title_to_num(i) for i in
df_test['Title']]
```

4.3 まとめ

この章では、データの可視化などを行い、各特徴量の生存割合との関係を中心に確認しました。具体的には各特徴量と生存割合の間に相関関係があったり、Nameのように名前から敬称を抽出することにより、相関関係があったりしました。

次章では本章で確認したことを基に、生存予測を行います。

¹ リスト内包表記については、D.2で説明しています。

第5章 識別器に学習させて予測する

この章では、前章でわかったことなどを参考に、タイタニックデータの「前処理」を行い、機械学習でよく使われるアルゴリズムである「ランダムフォレスト」を用いて、トレーニングデータの学習をした後、テストデータの生存予測を行います。

5.1 前処理

前処理 (Preprocessing) とは、与えられたデータについてアルゴリズムを用いて予測ができる形にするまでの処理をいいます。具体的には、欠損値への対応、外れ値の検出・処理、ダミー変数の作成、連続データの離散化、特徴量選択などを行います。

適用するアルゴリズムにより、異なる前処理をしなければならないことがあるので、何度も前処理をやり直す必要があるなど、データ分析を行うにあたりかなり時間がかかる作業です。

5.1.1 欠損値の処理

欠損値があると、scikit-learnを使用しての予測ができないため、欠損値の処理を行います。欠損値の処理方法として、その行を削除する方法と、何らかの値で補完する方法がありますが、今回は補完する方法でやってみます。

なお、欠損値をどう処理するかにより予測精度が上がったり、下がったりするため、欠損値をどのように処理するかは重要です。

5.1.1.1以降のコードは、ノートブックを新しく作成し、4.1のコード（データの読み込み、ライブラリのインポート）を実行したのちに実行してください。

5.1.1.1 年齢 (Age) の補完

年齢の補完を行います。

トレーニングデータの年齢の平均値は次のように、約30歳であることがわかります。

```
df_train['Age'].mean() # 年齢の平均値を算出
```

```
29.69911764705882
```


そこで、年齢が欠損になっている箇所に30を補完してやります。ただし、このように欠損値に単純に平均値を補完する方法は推奨される方法ではありません。5.3.1では、別の方法で補完を行いスコアを改善しています。

```
# 'Age'の欠損値に30を代入する。
df_train['Age'] = df_train['Age'].fillna(30)
df_test['Age'] = df_test['Age'].fillna(30)
```

—

5.1.1.2 乗船した港 (Embarked)の補完

次にトレーニングデータの乗船した港の補完を行います。
乗船した港が欠損となっているデータを見てみると、次のように、Ticketが113572と同じ数値になっていることがわかります。(その他に、38歳、62歳の女性であり、チケットのクラスは1st class、運賃は80.0、キャビン番号はB28であることなどが分かります。)

```
# df_trainでEmbarkedが欠損のデータを表示
df_train[df_train['Embarked'].isnull()]
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	
61	62	1	1	lcard, Miss. Amelle	female	38.0	0	0	113572	80.0	B28	NaN
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0	B28	NaN

そこで、df_trainにおいてTicketが113572の者がいるか抽出してみますが、この2人だけですし、df_testにおいても同じ番号の者はいません。

```
df_train[df_train['Ticket'] == '113572']
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	
61	62	1	1	lcard, Miss. Amelle	female	38.0	0	0	113572	80.0	B28	NaN
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0	B28	NaN

```
df_test[df_test['Ticket'] == '113572']
```

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
-------------	--------	------	-----	-----	-------	-------	--------	------	-------	----------

他に手がかりとなる情報もなく推測が難しいため、この2人のCabinは、1番

乗船者が多い乗船港である'C'で埋めることにします。

```
# 欠損値を'C'で埋め、表示して確認
```

```
df_train.loc[df_train['PassengerId'].isin([62, 830]), 'Embarked'] = 'C'
```

```
df_train.loc[df_train['PassengerId'].isin([62, 830])]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
61	62	1	1	Icard, Miss. Amelie	female	38.0	0	0	113572	80.0	B28	C
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0	B28	C

5.1.1.3 運賃 (Fare)の補完

最後に、運賃の欠損値の補完を行います。次のように運賃は、チケットのクラスに応じて、大きく異なることがわかるため、チケットのクラスの平均値により補完します。

```
# PclassごとにFareの平均値を表示
```

```
df_train[['Pclass','Fare']].groupby('Pclass').mean()
```

```
Pclass  Fare
```

```
1      84.154687
```

```
2      20.662183
```

```
3      13.675550
```

```
# 欠損値があるレコードを確認
```

```
df_test[df_test['Fare'].isnull()]
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
152	1044	3	Storey, Mr. Thomas	male	60.5	0	0	3701	NaN	NaN	S

PassengerIdが1044のレコードのFareについて、13.675550で埋めた後、実際に埋まっていることを確認します。

```
df_test.loc[df_test['PassengerId'] == 1044, 'Fare'] = 13.675550
```

```
df_test[df_test['PassengerId'] == 1044]
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
152	1044	3	Storey, Mr. Thomas	male	60.5	0	0	3701	13.67555	NaN	S

欠損値を補完することができたか確認してみると、次のようにAge、

Embarked、Fareの欠損値が0と表示され、欠損値を補完できたことがわかります。

```
print('--df_trainの欠損値--')
print(df_train.isnull().sum()) # df_trainの欠損値を表示
print('-'*10)
print('--df_testの欠損値--')
print(df_test.isnull().sum()) # df_testの欠損値を表示
```

--df_trainの欠損値--

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin         687
Embarked       0
dtype: int64
```

--df_testの欠損値--

```
PassengerId    0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin         327
Embarked       0
dtype: int64
```

5.1.2 カテゴリ変数への変換

SexやEmbarkedは、値が文字列になっており、このままではscikit-learnを

使用して分析することができません。そこで、この値を数値に置き換えてやります。

5.1.2.1 Sex (性別)の変換

Sexにはmaleとfemaleが入っているので、それぞれを0と1に変換します。

```
genders = {'male': 0, 'female': 1} # 辞書を作成

# Sexをgendersを用いて変換
df_train['Sex'] = df_train['Sex'].map(genders)
df_test['Sex'] = df_test['Sex'].map(genders)
```

5.1.2.2 Embarked (乗船した港)の変換

Embarkedには、乗船した港に応じてC、Q、Sのいずれかの値になっています。文字列のままだとscikit-learnを使用して予測ができないため、数値データに変換します。

乗船した港のように、どの港から乗船したかという識別子としての意味しか持たない名義特徴量の場合¹、ダミー変数に変換してやります。

ダミー変数に変換とは、次の図のように「港_C」という1つの特徴量に0または1のどちらかを割り当ててやることをいいます。

¹ 乗船した港について、寄港した順序に意味があれば、順番に意味がある順序特徴量と解釈できるかもしれません。

ダミー変数に変換

港		港_C	港_Q	港_S
C	ダミー化	1	0	0
Q		0	1	0
C		1	0	0
S		0	0	1
S		0	0	1

ダミー変数化するに当たっては、pandasのget_dummiesメソッドを用いると、簡単に変換でき「Embarked_C、Embarked_Q、Embarked_S」という3列が追加されます。

```
# ダミー変数化
df_train = pd.get_dummies(df_train, columns=['Embarked'])
df_test = pd.get_dummies(df_test, columns = ['Embarked'])
```

5.1.3 不要な列の削除

今回の予測では、'Name'、'Cabin'、'Ticket'の特徴量は使わないため¹、削除します。

```
df_train.drop(['Name', 'Cabin', 'Ticket'], axis=1, inplace=True)
df_test.drop(['Name', 'Cabin', 'Ticket'], axis=1, inplace=True)
```

¹ この特徴量が常に不要なわけではありません。Kernelsを検索するなどすると、それぞれの特徴量を使った素晴らしい分析に出会うことができます。

5.2 識別器に学習させて予測

ここまで、欠損値を補完し、文字列データを数値化し、不要な特徴量を削除するという前処理が終わりました。ここからは、「ランダムフォレスト」というアルゴリズムを使って、生存予測をしてみたいと思います。

ランダムフォレストについては、章末に説明を記載していますので参考にしてください。

前処理をしてきた、df_trainを'Survived'を除いたX_trainと'Survived'のみのY_trainというデータを作っています。こうすることで、X_trainとY_trainで学習したいという趣旨です。

また、X_testというテストデータを作っています。「.copy」が必要な理由についてはD.4で説明しています。

```
X_train = df_train.drop(['PassengerId', 'Survived'], axis=1) # 不要な列を削除
Y_train = df_train['Survived'] # Y_trainは、df_trainのSurvived列
X_test = df_test.drop('PassengerId', axis=1).copy()
```

```
from sklearn.ensemble import RandomForestClassifier

# ランダムフォレストのインスタンスを作成
forest = RandomForestClassifier(random_state=1)

# X_trainからY_trainを予測するように学習
forest.fit(X_train, Y_train)

# 正解率を表示
acc_log = round(forest.score(X_train, Y_train) * 100, 2)
print(round(acc_log, 2), '%')

96.07 %
```

このように、ランダムフォレストを用いて、X_trainから、Y_trainを予測してみると、96.07%の予測精度がでました。

しかし、この予想はX_trainとY_trainで学習してモデルを作り、その結果を用いてX_trainからY_trainを予測した場合にどのくらい正解できるかという結

果なので、間違いなく過学習¹しています。

そこで、「交差検証」という方法を用いて新しいデータに対してどれくらいの予測精度が出るか検証してみます。

今回は、交差検証の中でもデータを3分割し、3分割したうちの2つをトレーニングデータとし、残り1つをテストデータとして精度を検証する。3分割したデータのそれぞれがテストデータとなるように3回繰り返すという「k分割交差検証(のkが3の場合)」を用いてみます。

```
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
```

```
# 3分割交差検証を指定し、インスタンス化
skf = StratifiedKFold(n_splits=3)

# skf.split(X_train,Ytrain)で、X_trainとY_trainを3分割し、交差検証をする
for train_index, test_index in skf.split(X_train, Y_train):
    X_cv_train = X_train.iloc[train_index]
    X_cv_test = X_train.iloc[test_index]
    y_cv_train = Y_train.iloc[train_index]
    y_cv_test = Y_train.iloc[test_index]
    forest = RandomForestClassifier(random_state=1)
    forest.fit(X_cv_train, y_cv_train) # 学習
    predictions = forest.predict(X_cv_test) # 予測
    # accuracyを表示
    print(round(accuracy_score(y_cv_test,forest.predict(X_cv_test))*100,2))

79.12
81.14
81.14
```

accuracy (正解率)は、79%~81%程度であることがわかります。

それではこのモデルで予測を行なって、提出データを作ってサブミットしてみます。

¹ 過学習とは一般に、トレーニングデータに対し過剰適合しており、未学習のデータに対して適合できていない(正しく答えを出力できない)ことを指す。詳しくは、<https://ja.wikipedia.org/wiki/過剰適合> を参照。

```
# 学習と予測を行う
forest = RandomForestClassifier(random_state=1)
forest.fit(X_train, Y_train)
Y_prediction = forest.predict(X_test)
submission = pd.DataFrame({
    'PassengerId': df_test['PassengerId'],
    'Survived': Y_prediction
})
submission.to_csv('submission.csv', index=False)
```

Your submission scored 0.71770, which is not an improvement of your best score. Keep trying!

サブミットするとスコアは0.71770でした。女性を生存と予測し男性を死亡と予測するgender_submission.csvのスコアである0.76555よりも悪い結果となりました。¹

次節からは、この節で作ったランダムフォレストのモデルを改良して、0.76555を超えるモデルの作成を目指します。

5.3 モデルの改良

5.2節で作成したモデルではスコアが0.71770と低い結果となりました。このように既存のモデルのスコアが悪い場合、改善するために、欠損値の補完方法の変更、特徴量選択等の前処理の変更、ハイパーパラメーターの変更、アルゴリズムの変更など色々な方法が考えられます。この本では欠損値補完方法の変更、特徴量を追加、連続データの離散化という3つの方法を行います。

5.3.1 年齢の補完方法を変更する

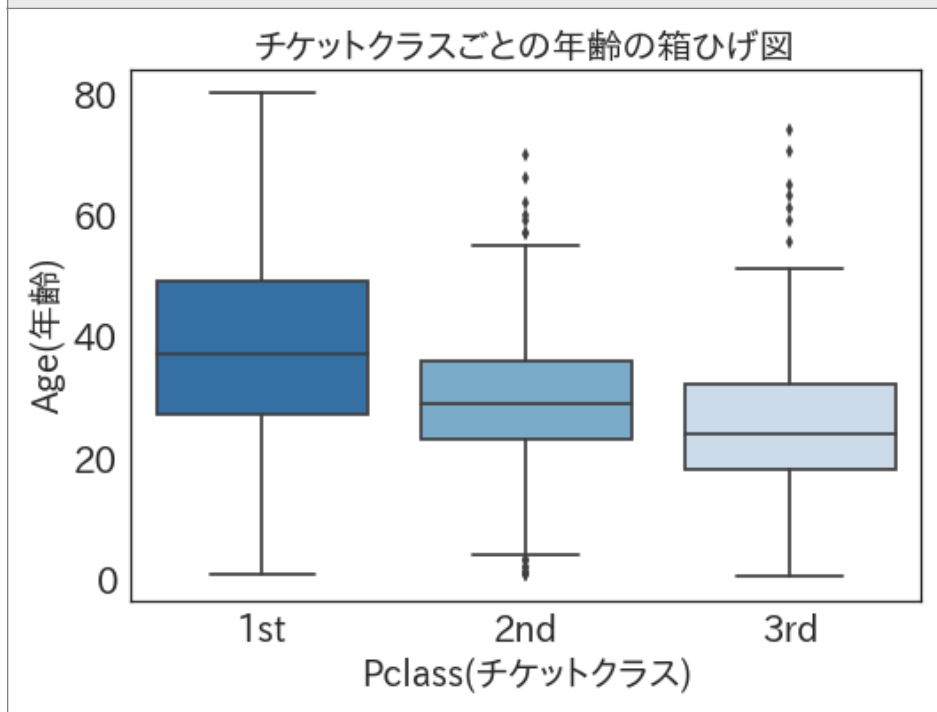
年齢は、5.1.1.1では、単純に年齢の平均値を代入しました。しかし単純に平均値を代入することは意味がないばかりか、欠損になっているデータを削除するよりも悪い結果をもたらすこともあります。

4.2.9で見たように、チケットのクラスに応じて大きく生存割合が異なりまし

¹ スコアがgender_submission.csvよりも低い理由については、本章末のエクササイズにしています。

た。そこでチケットのクラスと年齢に何かしらの関係がないかと思い、可視化してみます。

```
sns.boxplot(x='Pclass', y='Age', data=df_train)
plt.xticks([0.0, 1.0, 2.0], ['1st', '2nd', '3rd'])
plt.title('チケットクラスごとの年齢の箱ひげ図')
plt.xlabel('Pclass(チケットクラス)')
plt.ylabel('Age(年齢)')
```



箱ひげ図を描いてみたところ、1stクラスの客は年齢が高く、2nd、3rdとクラスが下がるにつれて、年齢が下がっています。

このチケットのクラスが下がると年齢も下がるというのは、当時の社会状況は分からないものの、現代の感覚からいけば納得できる結果です。

そこで、チケットのクラス毎の年齢の平均値により、年齢を補完することにします。

```
# PclassごとにAgeの平均を算出
df_train.groupby('Pclass').mean()['Age']
```

```
Pclass
1    38.233441
2    29.877630
3    25.140620
Name: Age, dtype: float64
```

チケットのクラスごとの年齢の平均値が分かったため、この値を四捨五入した値でdf_trainとdf_testの年齢の欠損値を補完します。

```
# Ageがnullの場合に、Pclassに応じてAgeに代入する関数
def impute_age(cols):
    Age = cols[0]
    Pclass = cols[1]

    if pd.isnull(Age):
        if Pclass == 1:
            return 39
        elif Pclass == 2:
            return 30
        else:
            return 25
    else:
        return Age

# df_train['Age']の欠損値に代入
df_train['Age'] = df_train[['Age', 'Pclass']].apply(impute_age, axis = 1)

# df_test['Age']の欠損値に代入
df_test['Age'] = df_test[['Age', 'Pclass']].apply(impute_age, axis = 1)
```

このようにAgeの欠損値をPclass毎の年齢の平均値で補完し、5.2と同じように、ランダムフォレストを用いて予測をし¹サブミットすると、スコアは0.72727となります。5.2のように年齢を全データの平均値で補完したスコアの0.71770と比べ、約1ポイント上がりました。

欠損値の補完方法を変更して、うまく補完してやることでスコアが改善されたということです。

¹ サンプルコードはサポートサイトを参照ください。

5.3.2 連続変数の離散化

これまでは、年齢と運賃を連続データのままで特に処理をせずに予測をしてきました。

4.2.14で運賃を4分割したように、年齢と運賃をカテゴリデータに変換してみます。

```
data = [df_train, df_test]
for df in data:
    # Fareのカテゴリ変数化
    df.loc[ df['Fare'] <= 7.91, 'Fare'] = 0
    df.loc[(df['Fare'] > 7.91) & (df['Fare'] <= 14.454), 'Fare'] = 1
    df.loc[(df['Fare'] > 14.454) & (df['Fare'] <= 31), 'Fare'] = 2
    df.loc[ df['Fare'] > 31, 'Fare'] = 3
    df['Fare'] = df['Fare'].astype(int)

    # Ageのカテゴリ変数化
    df.loc[ df['Age'] <= 16, 'Age'] = 0
    df.loc[(df['Age'] > 16) & (df['Age'] <= 32), 'Age'] = 1
    df.loc[(df['Age'] > 32) & (df['Age'] <= 48), 'Age'] = 2
    df.loc[ df['Age'] > 48, 'Age'] = 3
    df['Age'] = df['Age'].astype(int)
```

5.3.1のモデルを'Fare'と'Age'をこのようにカテゴリ変数化し、予測を行うと0.75119というスコアになります。

カテゴリ変数化を行うことで、5.3.1のスコア0.72727から0.02以上もスコアを上げることができたことがわかります。

5.3.3 特徴量の重要度を確認する

scikit-learnのランダムフォレストでは、学習済みのインスタンスに`feature_importances_`メソッドを用いることで、特徴量の重要度を取得することができます。5.3.2のモデルについて取得してみます。

```
forest.feature_importances_
array([0.13924783, 0.41004894, 0.11409453, 0.09676177,
       0.07999719, 0.10232023, 0.02503838, 0.02067357, 0.01181757])
```

このままでは、どの特徴量の重要度かわからないので、列名も一緒に表示するようにしてやります。

```
for i,k in zip(X_train.columns,forest.feature_importances_):  
    print(i,round(k,4))
```

```
Pclass 0.1392  
Sex 0.41  
Age 0.1141  
SibSp 0.0968  
Parch 0.08  
Fare 0.1023  
Embarked_0 0.025  
Embarked_1 0.0207  
Embarked_2 0.0118
```

Sexは重要な特徴量となっていることがわかります。反対に、SibSp、ParchとEmbarkedのダミー変数は重要な特徴量となっていないことがわかります。

5.3.4 特徴量の見直し

5.3.3では、SibSp、ParchとEmbarkedのダミー変数が重要な特徴量ではないことがわかりました。そこでSibSpとParchから、4.2.13のように家族のサイズを表す「FamilySize」と1人で乗船しているかを示す「IsAlone」作成し特徴量に加えます。また、Embarkedはダミー変数化しない¹ことにします。

このコードをスクリプトで書くと次のように書くことができます。

¹ 結果的に3.2.2のスコアと同一になるため、ダミー変数化しないことにしています。重要ではない特徴量は除外するのが一般的です。

```

import pandas as pd
from sklearn.ensemble import RandomForestClassifier

df_train = pd.read_csv('../input/train.csv')
df_test = pd.read_csv('../input/test.csv')

# Embarkedの補完
df_train.loc[df_train["PassengerId"].isin([62, 830]), 'Embarked'] = 'C'

# Fareの補完
df_test.loc[df_test["PassengerId"] == 1044, 'Fare'] = 13.675550

# Age変換のための関数
def impute_age(cols):
    Age = cols[0]
    Pclass = cols[1]
    if pd.isnull(Age):
        if Pclass == 1:
            return 39
        elif Pclass == 2:
            return 30
        else:
            return 25
    else:
        return Age

data = [df_train, df_test]
for df in data:
    # Ageの補完
    df['Age'] = df[['Age', 'Pclass']].apply(impute_age, axis = 1)

    # 性別の変換
    df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})

    # Embarked
    df['Embarked'] = df['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).astype(int)

    # Fareのカテゴリ変数化
    df.loc[ df['Fare'] <= 7.91, 'Fare'] = 0
    df.loc[(df['Fare'] > 7.91) & (df['Fare'] <= 14.454), 'Fare'] = 1
    df.loc[(df['Fare'] > 14.454) & (df['Fare'] <= 31), 'Fare'] = 2

```

```

df.loc[ df['Fare'] > 31, 'Fare'] = 3
df['Fare'] = df['Fare'].astype(int)

# Ageのカテゴリ変数化
df.loc[ df['Age'] <= 16, 'Age'] = 0
df.loc[(df['Age'] > 16) & (df['Age'] <= 32), 'Age'] = 1
df.loc[(df['Age'] > 32) & (df['Age'] <= 48), 'Age'] = 2
df.loc[ df['Age'] > 48, 'Age'] = 3
df['Age'] = df['Age'].astype(int)

# FamilySizeとIsAloneの作成
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1
df['IsAlone'] = 0
df.loc[df['FamilySize'] == 1, 'IsAlone'] = 1

# 不要な列の削除
df_train.drop(['Name', 'Cabin', 'Ticket', 'SibSp', 'Parch'], axis=1, inplace=True)
df_test.drop(['Name', 'Cabin', 'Ticket', 'SibSp', 'Parch'], axis=1, inplace=True)

# X_train、Y_train、X_testを作成
X_train = df_train.drop(['PassengerId', 'Survived'], axis=1)
Y_train = df_train['Survived']
X_test = df_test.drop('PassengerId', axis=1).copy()

# 学習
forest = RandomForestClassifier(random_state=1)
forest.fit(X_train, Y_train)
Y_pred = forest.predict(X_test)

# 提出データの作成
submission = pd.DataFrame({
    'PassengerId': df_test['PassengerId'],
    'Survived': Y_pred})
submission.to_csv('submit.csv', index=False)

```

このデータをサブミットすると、スコアは0.76555であり、3.2.2で予め用意されている提出サンプル (gender_submission.csv)を提出したスコアと同じスコアになりました。

なお、先ほどのスクリプトで作成した学習データを表示してみると次のよう

にとっても簡素なデータになることがわかります。前処理を行いここまで簡素なデータに変換し、そこそこのスコアを出すことができました。

X_train.head()							
	Pclass	Sex	Age	Fare	Embarked	FamilySize	IsAlone
0	3	0	1	0	0	2	0
1	1	1	2	3	1	2	0
2	3	1	1	1	0	1	1
3	1	1	2	3	0	2	0
4	3	0	2	1	0	1	1

5.4 まとめ

モデルの改善を行い、ようやく予め用意されている提出サンプルと同じスコアをだすことができました。この本での説明はここまでになりますが、例えば次のことを試して、モデルの改善などを行ってみましょう。きっとKaggleが楽しくなりますし、更に良いモデルを作成したくなります。

【この本では試さなかった試みや、改善点、エクササイズなど】

1. 5.3.4のモデルに対し、5.3.3と同様に特徴量の重要度にアクセスし、重要ではない特徴量があれば、削除した上で予測を行うこと。
2. 次の特徴量をモデルに用いるなどすること。
 - a. 4.2.15でNameから作成した特徴量である「Title_num」を用いる。
 - b. 乗船した港 (embarked)について、実際の航海でどのような順序で寄港したか確認。それぞれの生存割合の確認と、何故生存割合に差があるの推測。
 - c. キャビン番号 (cabin)が実際にどのように付けられていたのか調べ、有効な特徴量を作成。
3. ランダムフォレストには、インスタンス作成時に決めるハイパーパラメータが複数あります。このハイパーパラメータを変更し、予測をすること。
4. 3.のハイパーパラメータを決めるには、グリッドサーチと言われる手法

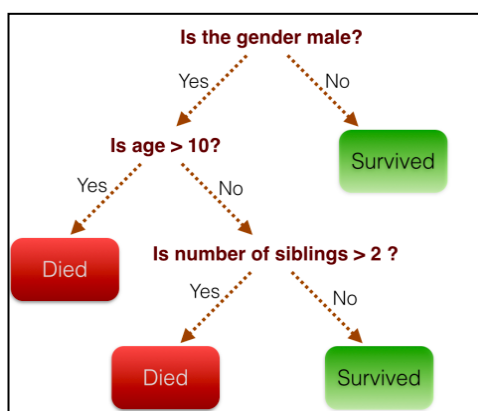
が有効になります。グリッドサーチを用いて良いハイパーパラメータを決定すること。

5. 5.2のランダムフォレストのモデルは、複数の特徴量を用いているにもかかわらず、予め用意されている提出サンプルのモデルよりも悪いスコアとなりました。ランダムフォレストについて書籍やWebで調べて何故なのか考えてみることに。
6. Kaggleのカーネルを見ると、この本と同じランダムフォレストを用いて、高得点を出しているものが沢山あります。どのようにして高得点を出しているのか分析し、真似をすること。
7. ランダムフォレスト以外にも機械学習のアルゴリズムは沢山あります¹。他のアルゴリズムを用いて、予測を行ってみることに。

ランダムフォレストについて

ランダムフォレストで用いられる「決定木」は、下図²のように、「男か女か」、「10歳以上か未満か」というように、データを当てはまりがよいように分割していき、その分割結果のグループごとに、分類結果を出力するアルゴリズムです。分割が枝分かれし木のように見えることから決定木と言われます。

ランダムフォレストとは、アンサンブル学習法(複数の識別器を集めて、予測を行うこと)の一つであり、「データのランダムサンプリングを複数回行い、使用する特徴量をランダムに選択し、決定木を複数作成し、複数の決定木の多数決により出力を決める」識別器です。ランダムサンプリングされたデータによって学習した複数の決定木を使用するため、「ランダムフォレスト」と言われます。詳細については、「統計的学習の基礎 (共立出版,2014)15章」やWikipedia³などを参照ください。



¹ scikit-learnのチートシート (http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html, <https://www.datacamp.com/community/blog/scikit-learn-cheat-sheet>)などを参照

² <https://qiita.com/yshi12/items/6d30010b353b084b3749>より引用

³<https://ja.wikipedia.org/wiki/ランダムフォレスト>

付録

A GoogleColaboratoryの使い方

これまでは、ローカルPCにPythonをインストールして使用するのが普通でしたが、Googleが2017年10月に「GoogleColaboratory」という素晴らしいツールを公開し、ブラウザ上でPythonを動かせるようになりました。

この章では、GoogleColaboratoryとは何か、Kaggleのカーネル環境との違い、Kaggleでの使用方法などを説明します。

この本のコードを実行したりタイタニックコンペに参加するだけであれば、Kaggleのカーネル環境やGoogleColaboratoryという無料でブラウザから使える環境で十分です。

A.1 GoogleColaboratoryとは何か

GoogleColaboratoryは、機械学習の教育と研究の普及を促進するために作られたグーグルのプロジェクトです。

GoogleColaboratoryのノートブックでは、ブラウザからPythonを実行でき、また、ノートブックはGoogleドライブに保存され、GoggleドキュメントやGoggleスプレッドシートと同じように共有することができます。

A.2 Kaggleのカーネル環境との違い

Kaggleのカーネル環境とGoogleColaboratoryとの機能を比較してみます。

	GoogleColaboratory	Kaggleのカーネル
料金	無料	無料

スペック - CPU - メモリ - Diskスペース - 使用時間	6CPU 13GB 40GB 連続12時間まで	4CPU 16GB 1GB 1つの処理で60分まで
GPU利用可否	使用可能 (Tesla K80)	使用不可 ¹
Kaggleのデータの 利用のしやすさ	やや面倒 (後述)	超簡単
Kaggleへの submitのしやすさ	一手間あり (後述)	超簡単 (1クリックで 可能)

以上のように、GoogleColaboratoryは、GPUが使えるため、ディープラーニングなど、計算量が大きく並列化に向いている処理をする場合には非常に便利です。一方でKaggleとのデータのやりとりには、一手間あります。

A.3 Kaggleでの使用方法

GoogleColaboratoryは様々な使い方ができますが、この本ではKaggleのデータを扱うという点とmatplotlibを使う際に日本語を使うという2点に絞って説明します。その他の一般的な使用例は私のブログ²にリンク等をまとめましたのでそちらも参考にしてください。

A.3.1 データの読み込み

Kaggleのデータの読み込み方です。Dropbox³を使用して読み込む方法と、Googleドライブにアップロードしたデータを読み込む方法⁴を説明します。

¹ BETA版ですが、Kaggleのカーネル環境でもGPUが使えるようです。私のブログ (<http://www.currypurin.com/entry/2018/04/15/124324>)を参照ください。

² <http://www.currypurin.com/entry/2018/03/08/013325>

³ dropbox (<https://www.dropbox.com/>)は、クラウドストレージサービス。2GBまでは無料でデータを保存でき、任意の端末からアクセスが可能。

⁴ GitHubにデータをあげておいて、ダウンロード (clone)する方法も非常に簡単です。私のブログ (<http://www.currypurin.com/entry/2018/03/14/100250>)に説明を書きましたので参考にしてください。

A.3.1.1 Dropboxを使用して読み込み

Dropboxに読み込みたいデータをアップロードし、ファイルの右側にある「共有」ボタンをクリックして共有URLを取得します。



GoogleColaboratoryのセルから、次のように!wget 先ほどコピーした共有URLと入力すると、データの読み込みがはじまります。

```
1 !wget https://www.dropbox.com/s/l7bvsh5wo80t0sm/test.csv
```

データの読み込みが終了した後に、!!lsと入力すると、test.csvが読み込めたことがわかるので、これまでのようにpandasで読み込んで使いましょう。

```
# ファイル一覧の表示
```

```
!!ls
```

```
# pandasで読み込み
```

```
import pandas as pd
df_test = pd.read_csv('test.csv')
```

```
datalab test.csv
```

A.3.1.2 GoogleDriveを使用して読み込み

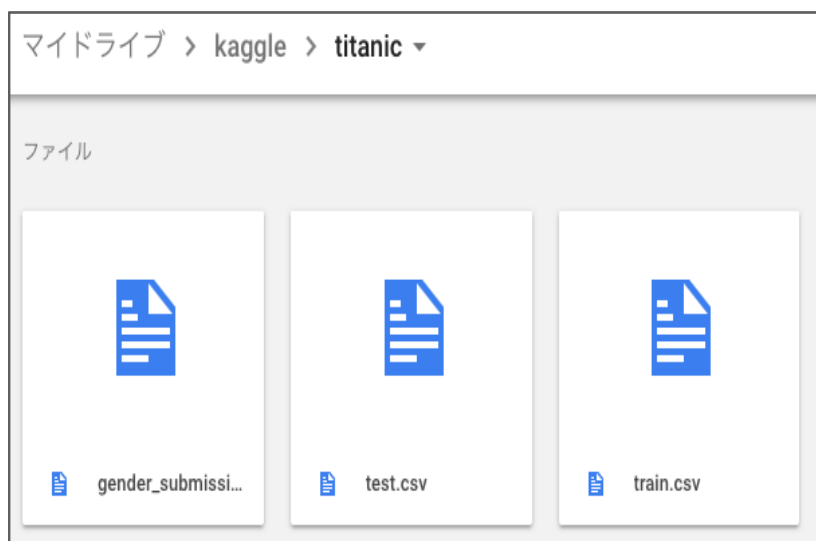
GoogleDriveは、デフォルトで15GBまでの容量が提供されるため、Dropboxでは容量が足りずに扱えない場合や、Googleのサービスだけで完結させたい場合などに、使うと良いと思います。

GoogleDriveを使用する場合は、Dropboxよりも少し面倒な手順となり、次のようになります。

1. GoogleDriveへファイルのアップロード
 2. ファイルの共有設定
 3. GoogleColaboratoryからGoogleDriveへの認証
 4. GoogleColaboratoryからGoogleDriveのファイルにアクセス
- 以下順に手順を記載します。

1.GoogleDriveへファイルのアップロード

GoogleDriveにファイルをアップロードします。今回はタイタニックコンペの3つのファイルをアップロードしました。



2.ファイルの共有設定

ファイルを右クリックし、「共有可能なリンクを取得」をクリックします。リンクが表示されるので、リンクをコピーしておきます。

3.GoogleColaboratoryからGoogleDriveへの認証

GoogleColaboratoryで次のコードを走らせます。

```
!pip install -U -q PyDrive

from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

すると、「Go to the following link in your browser:」と表示されるので、リンク先にアクセスし、「許可」をクリックします。すると「このコードをコピーし、アプリケーションに切り替えて貼り付けてください。」と表示されるので、GoogleColaboratoryに戻り、「Enter verification code:」に貼り付けます。

以上により、GoogleColaboratoryからGoogleDriveへアクセス可能となりました。

4.GoogleColaboratoryからGoogleDriveのファイルにアクセス

次のコードの「hoge」の箇所を、2.でコピーしたリンクのid=より後の部分に変更します。また「test.csv」の箇所を、適当な文字列に変更します。すると、読み込むことができます。

```
id = 'hoge' # 1.でコピーしたid= より後の部分
downloaded = drive.CreateFile({'id': id})
downloaded.GetContentFile('test.csv') # test.csvを適当な文字列に変更

# pandasで読み込み
import pandas as pd
# test.csvを先ほど書き換えた適当な文字列に変更
df_test = pd.read_csv('test.csv')
```

以上で、Kaggleからダウンロードしたデータをdf_testとして取り込むことができました。Kaggleのノートブックと見た目は若干異なりますが、同じように扱うことができます。

A.3.2 データのダウンロード

Kaggleへデータをサブミットするに辺り、GoogleColaboratory上のデータをダウンロードする必要があります。ダウンロードは次のようにして行います。(df_submissionというデータフレームをsubmission.csvというデータにしてローカルにダウンロードする場合のコードです)

```
#データフレームをcsv形式で書き出す
df_submission.to_csv('submission.csv', index=False)

#GoogleColaboratory特有のライブラリを使い、ダウンロードする
from google.colab import files
files.download('submission.csv')
```

ローカルにダウンロードされるので、3.4で説明したようにサブミットします。

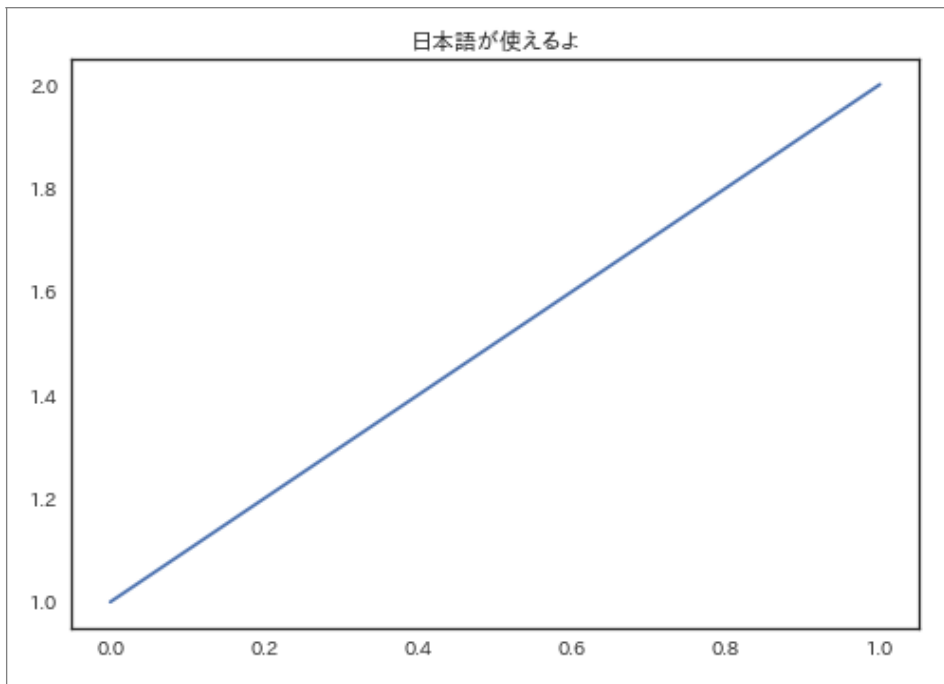
A.4 matplotlibで日本語を使う方法

図のタイトルを日本語にしようとしたりするなどmatplotlibで日本語を使おうとすると、うまく表示されません。日本語を表示したい場合は、以下の3行を実行しておく、日本語を表示することができます¹。

```
!apt-get -y install fonts-ipafont-gothic
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = 'IPAPGothic'

# 以下具体例
plt.plot([1,2])
plt.title('日本語が使えるよ')
```

¹ この方法で日本語が表示される場合と、表示されずエラーが表示される場合が発生しました。対処法は私のブログ (<http://www.currypurin.com/entry/2018/03/25/015759>)に記載していますので参照ください。



B 可視化の方法(matplotlibでの可視化)

この章では、5章で行った可視化について、入門者を対象に説明します。

B.1 pyplotでの可視化

Pythonでの可視化は、matplotlibというライブラリを使用する行うのが一般的です。特にmatplotlibに含まれるpyplotというモジュールを使用すれば、オブジェクト指向を意識せずに簡単に可視化ができます。

また、可視化する際は、4章でも使用したようにseabornというライブラリも使うことが多いので、私は通常一緒にimportしています。

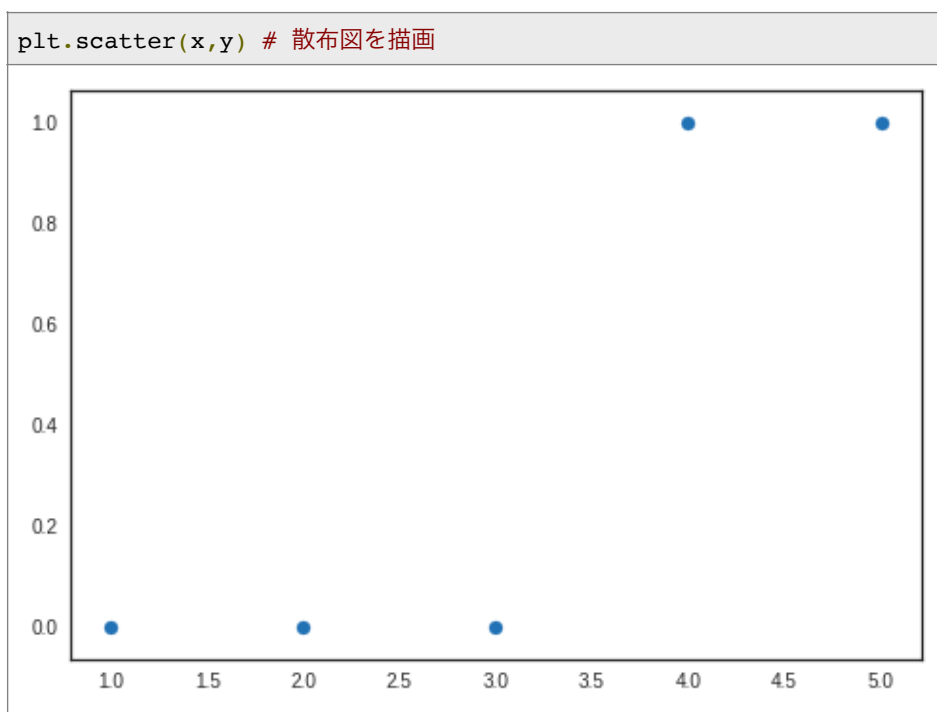
```
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np # numpyも使用するので、併せてimport
```

pyplotの代表的なメソッドに次のものがあります。

関数	説明	引数の説明
scatter(x, y)	散布図	x:x軸の値、y:y軸の値
plot(x, y)	折れ線グラフ	同上
bar(x, y)	棒グラフ	同上
title(s)	タイトルの設定	s:表のタイトル
xlim(xmin,xmax)	x軸の範囲の設定	xmin:x軸の最小値、xmax:x軸の最大値
ylim(ymin,ymax)	y軸の範囲の設定	ymin:y軸の最小値、ymax:y軸の最大値
legend()	凡例の表示	
sns.countplot(x,hue,date)	データの件数を集計し、ヒストグラムを描く	x:集計対象の列名、hue:細かく分割する際の列名、date:データフレーム
sns.heatmap(data)	ヒートマップの表示	data:ndarray形式に変換可能な2次元のデータ。pandasのデータフレームも指定可能

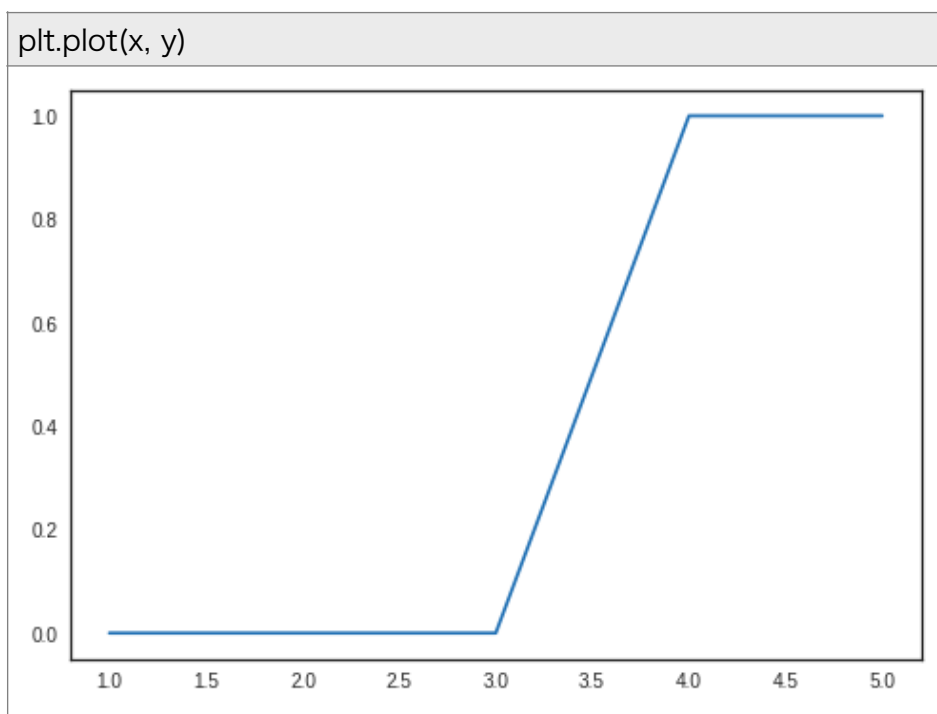
```
x = np.array([1,2,3,4,5]) # xにnumpyの配列を代入
y = np.array([0,0,0,1,1]) # yにnumpyの配列を代入
```


B.1.1 散布図



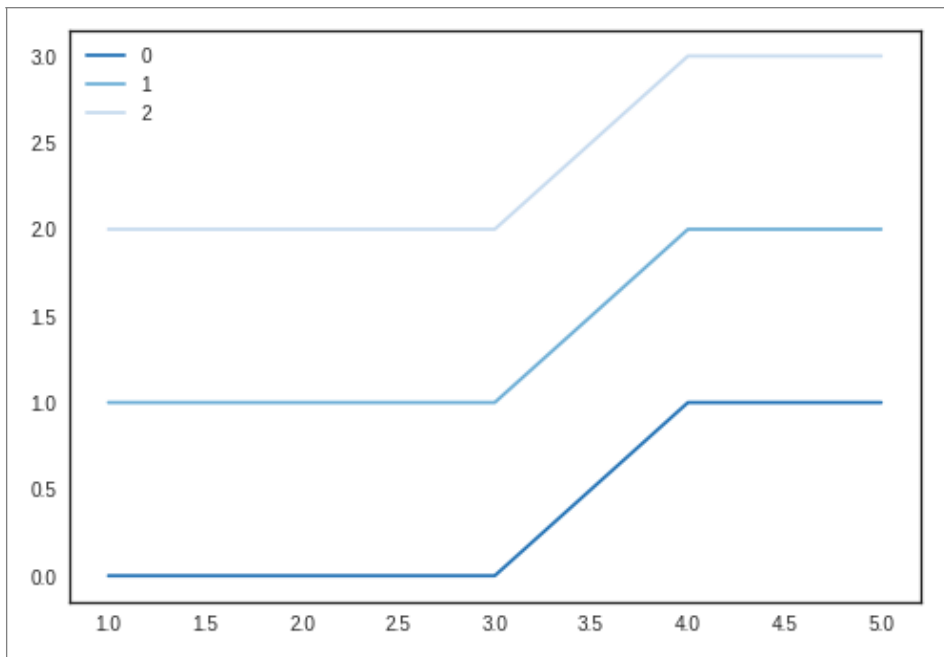
x,yの組みの5点(1,0)、(2,0)、(3,0)、(4,1)、(5,1)からなる散布図を描くことができました。

B.1.2 折れ線グラフ



先ほどと同じ5点を順に結んだ折れ線グラフを描くことができました。
同時に複数の折れ線を描いたり、凡例を表示したりしてみます。

```
plt.plot(x,y, label = '0') # labelを0に設定  
plt.plot(x,y+1, label = '1') # yをy+1に設定、labelは1に設定  
plt.plot(x,y+2, label = '2') # yをy+2に設定、labelは2に設定  
plt.legend() # 凡例を表示
```

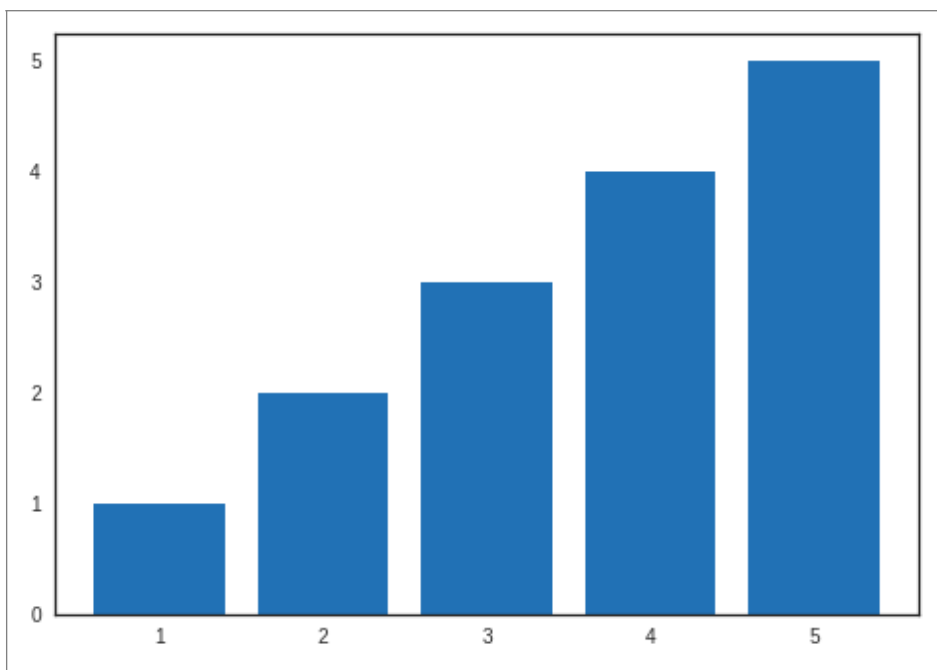


凡例が左上に表示され、3つの折れ線グラフが同時に表示されます。

B.1.3 棒グラフ

次に棒グラフを表示してみます。

```
plt.bar(x,x)
```



x軸の値が1、2、3、4、5で、それぞれの高さが1、2、3、4、5である棒グラフを書くことができました。

B.1.4 複数の図を同時に描く

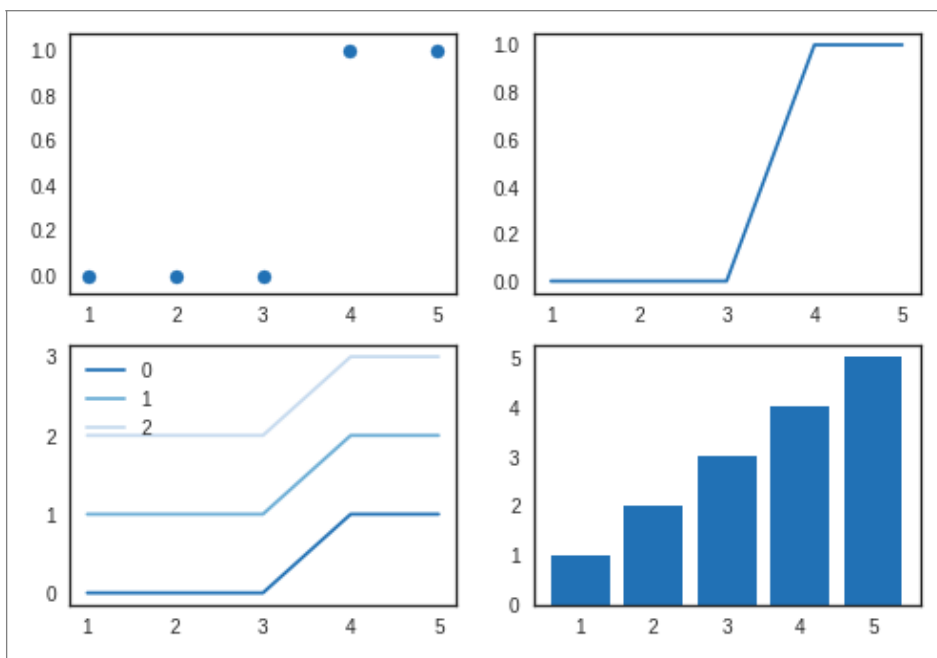
複数の図を同時に描く際は、pyplotのsubplotメソッドを用いて、次のようにsubplotの引数で位置を指定してから、描画のコードを書きます。

```
plt.subplot(221) # 2行2列の1つめ
plt.scatter(x,y)

plt.subplot(222) # 2行2列の2つめ
plt.plot(x,y)

plt.subplot(2,2,3) # 2行2列の3つめ、このように、区切りも可能
plt.plot(x,y, label = '0')
plt.plot(x,y+1, label = '1')
plt.plot(x,y+2, label = '2')
plt.legend();

plt.subplot(224) # 2行2列の4つめ
plt.bar(x,x)
```



B.2 pyplotのfigure関数を用いる方法

これまでのように簡単な可視化を行うには、pyplotの関数を使うと便利なのですが、少し凝った可視化をする場合などにはfigure関数を使い、明示的にFigureのインスタンスとAxesインスタンスを作成して¹可視化されることがあります。

その場合の方法を知らないと、コードを読むことができなくなるため、簡単な例を紹介します。²

¹ B.1のようにpyplotの関数を用いる場合には、裏側でFigureクラスとAxesクラスがインスタンス化されているので、簡便に描画が可能です。

² 更に詳しく知りたい方は公式チュートリアル (<https://matplotlib.org/tutorials/index.html>)や、「早く知っておきたかったmatplotlibの基礎知識、あるいは見た目の調整が捗るArtistの話」(<https://qiita.com/skotaro/items/08dc0b8c5704c94eafb9>)を読むことをお勧めします。

```

# Figureのインスタンスを生成
fig = plt.figure()

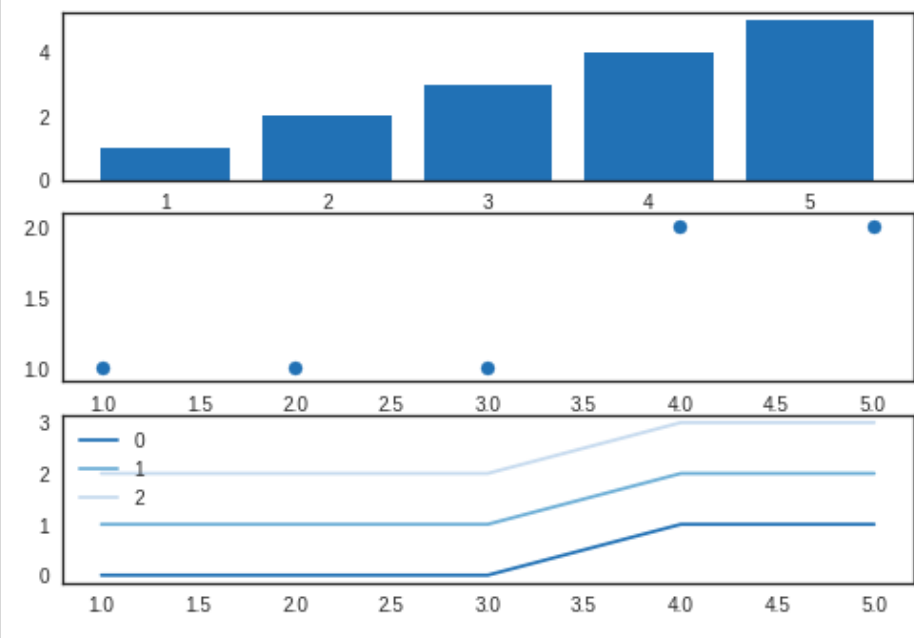
# Axesのインスタンスを生成
ax1 = fig.add_subplot(311)
ax2 = fig.add_subplot(312)
ax3 = fig.add_subplot(313)

# ax1,ax2,ax3にplot
ax1.bar(x, x)

ax2.scatter(x, y+1)

ax3.plot(x,y, label = '0')
ax3.plot(x,y+1, label = '1')
ax3.plot(x,y+2, label = '2')
ax3.legend(loc='upper left')

```



なお、ここまで説明したKaggleのノートブックやGoogleColaboratoryを使用せずに、C章で説明するJupyterノートブックを用いる場合には、matplotlibで可視化するにあたり、`%matplotlib inline` と1度実行しておく必要があります。

C Pythonのインストール

この章では、ローカル環境¹にPythonの環境を構築するための方法を紹介します。

C.1 Anacondaのインストール

Pythonインストールする際は、Continuum Analytics社が提供する、Anacondaというパッケージを使うのが簡単です。Anacondaは、Pythonに加え、Pythonを用いてデータサイエンスをする際に使用するライブラリの多くが含まれており、環境の構築がAnacondaのインストールのみで終わるためです。

インストール方法は、ANACONDAのサイト²から、OSに応じたインストーラー³をダウンロードします。インストーラーを起動するとPythonを含むパッケージ一式がインストールされます。

C.2 Jupyterノートブックについて

Jupyterノートブックは、GoogleColaboratoryのようにノートブックと呼ばれる形式で作成したプログラムを実行し、実行結果を記録しながら、データの分析作業を進めるためのツールです。

Anacondaには、Jupyterノートブックも含まれており、Anacondaをインストールすると、Jupyterノートブックも使用可能となります。

D Pythonの基本

Pythonを全く触ったことがない方を対象に、Pythonではこんなに簡単に計

¹ ローカル環境は、自分の好きなマシンを使えたり、インターネット環境が必要なかったりなど当然便利な点もありますし、凝ったことをやろうとするとローカル環境が必要になるかと思います。

² <https://www.anaconda.com/download/>

³ Python3系のインストーラーと2系のインストーラーがありますが、特に理由がなければ、3系を選ぶと良いでしょう。

算ができるよということを書いてみます。

次の3つの例題のような簡単な計算や処理が複数の方法できるようになることをゴールとします。

例題

- 1. 「1+2+3・・・98+99+100」を計算
- 2. 「1から10数のうち奇数を二乗したリスト」を取得
- 3. リストの要素を整数型にする

D.1 四則演算

足し算や引き算等の四則演算などは、次のように数と演算子を並べることで計算することができます。

1 + 5 - 2 #足し算と引き算
4

主な演算記号は次の通りです。

演算子	説明	利用例	利用例の結果
+	和	5 + 3	8
-	差	5 - 2	3
*	積	5 * 3	15
/	商	5 / 2	2.5
//	商 (小数点切り捨て)	5 // 2	2
%	商の余り	10 % 3	1
x ** y	xのy乗	2 ** 3	8
round(x, n)	xをn桁に丸める	round(3.5555,2)	3.56
math.floor(x)	xの小数点を切り捨て	math.floor(3.5)	3.5
math.ceil(x)	xの小数点を切り上げ	math.ceil(3.5)	3.6

それでは、例題1.の1から100までを足すという問題をやってみます。

次のように1から100までの100個の数字を全て書くことで計算することができます。

1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+ 19+20+21+22+23+24+25+26+27+28+29+30+31+32+33+34 +35+36+37+38+39+40+41+42+43+44+45+46+47+48+49+5 0+51+52+53+54+55+56+57+58+59+60+61+62+63+64+65+ 66+67+68+69+70+71+72+73+74+75+76+77+78+79+80+81 +82+83+84+85+86+87+88+89+90+91+92+93+94+95+96+9 7+98+99+100
5050

しかし、このように、全て書くのは大変ですし、一つでも誤って書くと答えが異なってしまうので、もっと簡単に書く必要があります。そこでfor構文という繰り返しの構文を用いて書いてみることにします。

-for構文- (else以下は省略可能)
for ループ変数 in 範囲:
 繰り返す処理 (ループ変数を使用可能)
else:
 繰り返し後の処理 (ループ変数を使用可能)

for文は、このように書くことができ、範囲には、リストなどの「Iterable」なオブジェクトが入り、else以下は必要が無い時は省略可能です。

例えば1から10までを合計する計算は次のように書くことができます。

<pre>sum_num = 0 # 合計をいれておく変数、初期値は0 # numには1から10が順に代入されるfor構文 for num in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]: sum_num = sum_num + num #sum_num + numを計算しsum_numに代入 print(sum_num) # sum_numを表示</pre>
55

inの後ろの範囲の部分に[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]という全ての合計したい数字が含まれるリストを書きましたが、1から100までのリス

トを書くのは大変です。そこでrange ()という数字のリストを簡単につくることができる関数が用意されています。

```
range([start,] stop[, step])
```

startからstop未満の最大の整数までのstep刻みのリストを返す関数。
startを省略すると0start、stepを省略すると1刻み。

range(m,n)はmからn-1までのリストを返すため、mからn-1まで繰り返し処理を行うことができます。つまり1から100までの繰り返し処理をするために、1から100までのリストが欲しい場合はrange(1,101)と書けば良いということです。

```
sum_num = 0 # 合計をいれておく変数、初期値は0

for num in range(1, 101): # iに0から100を順次代入している
    sum_num = sum_num + num # sum_num + num をsum_numに代入

print(sum_num)
```

5050

このように短く書くことができました。

しかし、この書き方もfor構文を使うため、1行には収まらず数行書く必要があるためやや不便です。そこで次のようにrangeと、リストの数値の合計を返す関数であるsum関数を用いて次のように書くことができます。

```
sum(range(1, 101))
```

5050

sum()は、引数 (後ろの括弧内の値のこと)のリストを合計する関数です。そのため、range(1, 101)という1から始まり、100までのリストを合計することができます。

D.2 リスト、リスト内包表記

1から10のうち奇数を二乗したリストを取得してみます。まずは、全てを自力で書く方法です。

```
[1**2, 3**2, 5**2, 7**2, 9**2]
```

```
[1, 9, 25, 49, 81]
```

これは、先ほどのfor構文とrange関数を用いて次のように書くことができます。

```
square_list = [] # 空のリストを定義
for i in range(1,10,2): #1から10未満までの2刻みのリストをiに代入
    square_list.append(i**2) #リストの末尾にiの二乗を追加

print(square_list)
```

```
[1, 9, 25, 49, 81]
```

しかし、これも何行か書く必要があるため大変です。そこでPythonでは、次のようにリスト内包表記という表記方法が用意されており、これを使えば1行で書くことができます。

[式 for ループ変数 in 範囲] と書くことで、リストが取得可能。

式には、ループ変数を使うこともできるし、使わないこともできる。

先ほどのfor文は、リスト内包表記を用いて次のように書くことができました。

```
[i**2 for i in range(1, 10, 2)]
```

```
[1, 9, 25, 49, 81]
```

D.3 リストの要素を整数型にする

float型 (少数型)であるリストの要素をint型 (整数型)に変換してみます。Pythonにはintという関数があり、後ろの引数を整数型に変換することができます。

```
int(3.5)
```

```
3
```

for文を使い、ひとつずつint関数を用いて整数にしてみます。

```
int_list = [] # 空のリストを定義
```

```
for i in [1.0, 2.0, 3.5, 4.0]:
```

```
    int_list.append(int(i)) # リストの要素をひとつずつint型にする
```

```
print(int_list)
```

```
[1, 2, 3, 4]
```

これをリスト内包表記を使って書くと次のようになります。

```
[int(i) for i in [1.0, 2.0, 3.5, 4.0]]
```

```
[1, 2, 3, 4]
```

以上のように、リスト内包表記を使えば、簡単に (1行で)変換することができます。

D.4 .copyが必要な理由

5.2で、`X_test = df_test.drop('PassengerId', axis=1).copy()`とした際に、copyが必要な理由を具体例で説明します。

具体例

1行目が[1, 2]、2行目が[2, 3]であるdfというデータフレームを作ります。

```
df = pd.DataFrame(data=[[1,1),(2,3)])  
df
```

1, 1 ← 上記データフレームをこのように表示することにします。
2, 3

df1 = df としたのち、df1の0行,0列を10に書き換えます。

```
df1 = df  
df1.iloc[0,0] = 10 # df1の0行,0列を10に書き換え  
df1
```

10, 1
2, 3

dfを表示します。

```
df
```

10, 1
2, 3

df1の値を変更したのに、dfの値も変更されてしまいました。

df1 = dfというのは、同じ場所を参照すると言う意味で、どちらかの値を書き換えると、もう一方の値も書き換わってしまいます。

そのため、ディープコピーと言われる参照先の値を新しい場所にコピーする処理をしたい場合には、copy()としてやる必要があります。

このようにすることで、5.2の例で言えば、X_testの値を書き換えても、df_testの値は変わらないことになります。

E Kaggleの称号と用語集

E.1 Kaggleの称号の説明

Kaggleには、コンペに参加し上位の成績を残すことで獲得することができる「グランドマスター」や「マスター」等の称号があります。称号をもらえる条件等について説明します。なお更に詳しくは、公式サイト¹や私のブログ²に書いた説明を参照ください。

称号	条件
Grandmaster	5 gold medals Solo gold medal
Master	1 gold medal 2 silver medals
Expert	2 bronze medals
Contributor	プロフィールの完成や1stサブミットの実施 等
Novice	Kaggleに登録

Kaggleに登録すると、全員がNoviceから始まります。プロフィールを完成させ、サブミットすることなどにより、Contributorの称号がもらえます。

Expert以上の称号を得るためには、コンペで上位に入りメダルを獲得する必要があります。メダルの獲得条件は次のようになっています。

Grandmasterの条件のSolo gold medalとは、複数人での参加ではなく、単独参加でgold medalを取得しているということです。

¹ <https://www.kaggle.com/progression>

² <http://www.currypurin.com/entry/2018/02/21/011316>

	0~99 チーム	100~249 チーム	250~999 チーム	1000 チーム以上
Bronze	上位40%	上位40%	上位100人	上位10%
Silver	上位20%	上位20%	上位50人	上位5%
Gold	上位10%	上位10人	上位10人と 0.2%*1	上位10人と 0.2%*1

*1:上位10人と0.2%の意味は、コンペに参加するチームが500チーム増えるごとに、与えられるメダルが1つ増えるということ¹。

E.2 Kaggle用語集

用語	意味
EDA (Exploratory Data Analysis)	たまたにEDAというタイトルだったり、タグがついているカーネルがありますが、これは、探索的データ解析と訳されます。 探索的データ解析とは、モデルを仮定せずに、フラットな状態で、データを眺めてみようという意味で、コンペにエントリーした際にとりあえずEDAと書いてあるカーネル見ると、どのようなコンペか確認することができます。
Kaggle api	Kaggle 公式のapiであり、コンペ一覧の取得、データのダウンロード、サブミット等が可能。(2018年4月15日現在、ベータ版)
Kaggle learn	KaggleのLearnというページ ² には、ハンズオンでデータサイエンスを学べる次のノートブックが公開されています。 Machine Learning、Pandas、Data Visualisation、SQL、R、Deep Learning

¹ たとえば、500チームが参加するコンペには全部で11個のgold medalが与えられ、5000チームが参加するコンペでは全部で20個のgold medalが与えられます。

² <https://www.kaggle.com/learn/overview>

kaggler-ja	日本のKagglerのslackコミュニティ。Grandmasterからビギナーまでの幅広い層が情報交換をしている。
Kaggle Rankings	これまでのコンペで獲得したポイントに応じたKaggleのランキング。ポイントは時間とともに減衰するため、ランキングは最近のコンペで獲得したポイントの影響が大きく、競走的なランキングとなる。

F データ分析の勉強方法

完全に私見になってしまいますが、データ分析について学ぶ際にどのように学ぶのが1番よいか、考えている方法を記載します。

データ分析 (機械学習等)について学ぶ場合は、実戦から入ってしまうのが勉強方法として最適であり、Kaggleから始めるのが1番良いのではないかと私は考えています。以下のような順序で学んでみではいかがでしょうか。¹

F.1 Kaggleで初サブミット

まずは何もわからなくてもよいので、とりあえずサブミットしましょう。

やったことあるのと、ないのではとてつもない差があると思います。この本の3章を読みながら、何はともあれ初サブミットしましょう。

F.2 Kaggleを楽しむ

初サブミット後は、Kaggleを楽しむと良いでしょう。

この本の4章や5章を見ながら、タイタニックチュートリアルをやってみてもいいし、現在開催中のコンペや過去のコンペで興味を持ったカーネルを眺めてみるなどすれば、雰囲気がつかめるし、自分が学ばなければいけないことがわかると思います。

¹ 私も現在このように学んでいる途中です。結果を定期的にブログなどに書きたいと思います。

F.3 知識をインプットする必要

F.2 のカーネルだけで学べる方はずっとそれで良いと思うのですが、通常は入門用の書籍等でのインプットをしたくなると思います。以下はその場合のお勧めです。全てを順番にということではなく、気に入ったものを取りましょう。

F.3.1 本で学ぶ

私も次の2冊で学びながら、この本を書いています。どちらの本もとてもオススメです。

- Sebastian Raschka, Python機械学習プログラミング, インプレス, 2016, クイープ訳, 福島 真太郎 監訳 (原著: Python Machine Learning)
- Andreas C. Muller and Sarah Guido, Pythonではじめる機械学習, オライリージャパン, 2017, 中田 秀基訳 (原著: Introduction to Machine Learning with Python)

また、Pythonの知識が十分じゃない場合は、並行してPythonの基礎がわかる本を読むとよいと思います。Pythonの初心者向けの本は分かりやすくて素晴らしい本が多いので、本屋で気に入った本を選べば、良書に巡り会えると思います。

F.3.2 動画で学ぶ

動画のコースも最近とても充実してきました。巷では英語のコースがよく勧められています。ここでは、日本語で学ぶことができるコースでお勧めのものを紹介します。

次の動画は、どちらもUdemyの動画で、内容が素晴らしいと思います。定価は10,000円overですが、セールの時に買うと、格安で買うことができます。

- Toru Tamaki, Pythonで機械学習: scikit-learnで学ぶ識別入門, Udemy
- Shingo Tsuji, Pierian Data International by Jose Portilla, 【世界で5万人が受講】 実践 Python データサイエンス, Udemy

F.3.3 カーネルで学ぶ

本や動画で学習しながら、カーネルで学習した単語を検索し、どのように実戦で使われているかを確認したりすると理解が深まると思います。

また、英語ができる方などはおそらくカーネルだけでも、かなりのところまで行けるのではないのでしょうか。

F.4 実践

F.2の知識のインプットと平行して行えますが、カーネルを読みながら、ひたすら実践してトライ&エラーを重ねるのがとても勉強になります。カーネルや自分のブログにトライ&エラーを書いておくと、すごい方からアドバイスがもらえて、ものすごく勉強になります。

F.5 網羅的に理解する

コンペに何度も参加し、色々な分析手法を身につけあとに、網羅的でしっかりと理論を書いてある本を学ぶと、多くの学びがあるのではと思います。

おすすめの書籍としては、「Trevor Hastie and Robert Tibshirani and Jerome Friedman,統計的学習の基礎,共立出版,2014,杉山 将 ほか監訳 (原著: The Elements of Statistical Learning)」が非常にわかりやすくしっかり書かれているため良いのではと思っています。

F.6 その後

ここまでくれば、自分でやることはみつかると思います。Kaggleで上位を目指したり、論文を読んで実装してみたり色々できるのではないのでしょうか。

あとがき

データ分析もしたこともなかった素人が、競馬の分析がしたくて、データ分析の勉強を初めて約1年6ヶ月。毎週金曜日にデータ分析の勉強会などに参加し教えてもらいながら勉強をすすめ、なんとかこの本を書き上げることができた理解度まで到達することができました。

この本を書くために、かなりの時間を投入しましたが、これからはKaggleで良い順位をとるために、多くの時間を投入していく所存です。

Kaggleでソロ金メダル、そして1位をとることを目標に全力で挑戦していきますので、これからも応援よろしくお願いします。

また、この本を読んでKaggleに興味を持った方は、是非Kaggleと一緒にやってみましょう。わたしのグループも現在のところ毎週金曜日に新宿で勉強会を開いていますので、参加いただければと思います。

この本は今年の1月から全力で書き上げました。読んでいただいた方に少しでもためになる箇所があったとしたらとても嬉しいです。

しかし全力で書き上げたものの、内容や本の構成は改善できることが数多くあり、Kaggleのチュートリアルとして更に良い内容となるように、アップデートを行い、改訂版を出すことも今後の目標となりました。

この本を書くにあたり、執筆中に原稿を読み貴重なコメントをいただいた以下の方々にはとても感謝しております。おかげで、内容のよくない初稿から修正を重ねこの本が出来上がり、また修正を重ねる過程でとても勉強になりました。時間等の兼ね合いで反映できないコメントもありましたが、改訂版を出す際には絶対に反映させます。

@hKosho, @Yoshiki, @norinory, @osawat, @rakuda1007,
@sakurasaku, @takenaka, @ysaito (いずれもslackのID)

また、金曜日の勉強会に参加いただいた皆様ありがとうございました。いつも刺激を受け楽しく勉強することができ、この本を書くことができました。

カレーちゃん

<http://www.currypurin.com/>

<https://twitter.com/currypurin>

手にとっていただき、読んでいただきありがとうございました。

この本を読んだ感想やツッコミなどお待ちしております。twitter又はpeingまでどしどしお寄せください。それが更なるこのチュートリアルのパワーアップに繋がりますので。

また、本を読んで実践してみた結果をブログ等で公開いただけると大変嬉しいです。

本当は、背表紙にチートシートをつける予定だったのですが、間に合いませんでした。改訂版ではチートシートをつけて戻ってきます。

kaggleのチュートリアル

2018年 4月22日 技術書典4 初版第1刷発行

著 者 カレーちゃん

発行所 データリファインメント

印刷所 日光企画

(C)2018データリファインメント