

# はじめに



## この教材で学べること

本教材は読者が、**自力でKaggleのコンペに参加することができる**ようになることを目指して作成しました。この教材を修了して、ぜひKaggleの世界に飛び込んでみましょう！本教材は以下の二部構成となっています。

- Kaggle基本操作編
- Kaggle実践編

**Kaggle基本操作編**では、Kaggleを利用するにあたって最低限必要となる情報を説明しています。すでに理解されている方は読み飛ばしてしまっても構いません。

**Kaggle実践編**がメインコンテンツです。実際にKaggleノートブックを作成いただき、コードを書いてもらう演習形式となっています。Kaggleの基本的な3ステップである

- EDA
- 特微量エンジニアリング
- モデリング

について順を追って学びます。解答例はKaggleノートブックとして公開しており、教材中にリンクを記載しています。

## この教材の前提知識

- Python文法の基礎知識（for文やif文がわかる）
- 機械学習の基礎知識（教師あり学習と教師なし学習の違いがわかる）

## この教材の対象者

- Kaggleを始めてみたいと思っている人
- Kaggleのアカウントを作ったが、何をしていいかわからない人

## 対応バージョン

プログラミングはKaggleノートブックで実行します。標準的なブラウザが動作する環境であれば、環境は一切問いません。

※なお、Kaggleノートブックは下記のDockerイメージを利用しています。

<https://github.com/Kaggle/docker-python>

## 目次

### Kaggle基本操作編

- コンペティションへの参加
- Overview
- Data
- Notebooks
- Discussion
- Leaderboard
- Rules
- Teams
- My Submissions
- Submit Predictions
- 新規ノートブックの作成

### Kaggle実践編

1. 準備
2. EDA
3. 特徴量エンジニアリング
4. モデリング
5. 提出
6. 発展

## Kaggle基本操作編

---

### コンペティションへの参加

Kaggleにログインしてください。左のメニューから**優勝カップのマーク**をクリックすると、コンペティションの画面に移ります。ここからコンペに参加することになります。

今回はタイタニックコンペに参加してみましょう。[Titanic – Machine Learning from Disaster](#)を選択してください。

**INGV - Volcanic Eruption Prediction**  
Discover hidden precursors in geophysical data to help emergency response  
Playground · 7d to go · 556 Teams Swag

**Titanic - Machine Learning from Disaster**  
Start here! Predict survival on the Titanic and get familiar with ML basics  
Getting Started · Ongoing · 17450 Teams Knowledge

**House Prices - Advanced Regression Techniques**  
Predict sales prices and practice feature engineering, RFs, and gradient boosting  
Getting Started · Ongoing · 4732 Teams Knowledge

**All Competitions**

Active (Not Entered)   Completed   InClass   All Categories ▾   Default Sort ▾

**NFL Big Data Bowl 2021**  
Help evaluate defensive performance on passing plays  
Analytics · 8d to go   \$100,000

次のように個別のコンペ画面に移動するはずです。説明の都合上で違うコンペの画像になっていますが、みなさんはタイタニックコンペの画面が見えているはずです。[Join Competition](#)をクリックするとコンペに参加できます。

**Research Code Competition**

**HuBMAP - Hacking the Kidney**  
Identify glomeruli in human kidney tissue images

InnovationDigi · 726 teams · a month to go (20 days to go until merger deadline)

\$60,000 Prize Money

Overview   Data   Notebooks   Discussion   Leaderboard   Rules   [Join Competition](#)

ここからはコンペの画面について説明します。

## Overview

コンペのメニューから[Overview](#)をクリックします。[Overview](#)ではコンペの概要について確認できます。

[Description](#)では、コンペの背景の説明や開催期間について確認できます。なお、タイタニックコンペは練習用のコンペであるため常時開催しています。

[Description](#)の最下部にある[Points](#)と[Tiers](#)は重要です。ここにメダルの付与対象になっているかが記述されています。メダルを目標にコンペに取り組まれる方は、コンペに参加する前にここをよく確認しましょう。

The screenshot shows the competition page for the "Titanic" competition. It includes a sidebar with user posts and a main content area with statistics, rules, and tags.

- my\_first\_dedicated\_project**: 1 vote · an hour ago
- Titanic : PyTorch Neural Network| GPU | Top 14%🔥**: 4 votes · 8 hours ago
- 鐵達尼生存預測(homework)**: 0 votes · 3 hours ago

**Getting Started Competitions - 100% Public Leaderboard**: 235 replies · a day ago

**Do you keep the 'Cabin' column**: 3 replies · a day ago

**Watch out! Trap of fare**: 7 replies · 18 hours ago

This competition runs indefinitely with a **rolling leaderboard** which invalidates entries after two months.

**17,450 Teams**   **18,144 Competitors**   **81,457 Entries**

**Points**: This competition does not award **ranking points**  
**Tiers**: This competition does not count towards **tiers**

**Tags**: beginner, binary classification, tabular data, categorizationaccuracy

Evaluationでは提出すべき予測結果のフォーマットや評価指標について確認できます。

The screenshot shows the "Evaluation" section of the competition page. It includes a navigation bar and a detailed description of the evaluation process.

**Overview**   Data   Notebooks   Discussion   Leaderboard   Rules   Team   My Submissions   **Submit Predictions**

**Overview**

<b>Description</b>	<b>Goal</b>
<b>Evaluation</b>	It is your job to predict if a passenger survived the sinking of the Titanic or not. For each in the test set, you must predict a 0 or 1 value for the variable.
<b>Frequently Asked Questions</b>	<b>Metric</b>  Your score is the percentage of passengers you correctly predict. This is known as <b>accuracy</b> .
	<b>Submission File Format</b>  You should submit a csv file with exactly 418 entries plus a header row. Your submission will show an error if you have extra columns (beyond PassengerId and Survived) or rows.  The file should have exactly 2 columns: <ul style="list-style-type: none"><li>• PassengerId (sorted in any order)</li><li>• Survived (contains your binary predictions: 1 for survived, 0 for deceased)</li></ul>

## Data

Dataでは、コンペのデータの概要が確認できます。またデータをダウンロードすることもできるので、必要に応じてダウンロードします。本教材ではKaggle上でNotebooks機能を使いますが、後ほど提出方法の確認のためにgender\_submission.csvを利用しますのでダウンロードしておきましょう。

Overview [Data](#) Notebooks Discussion Leaderboard Rules Team My Submissions [Submit Predictions](#)

## Data Description

### Overview

The data has been split into two groups:

- training set (train.csv)
- test set (test.csv)

The training set should be used to build your machine learning models. For the training set, we provide the outcome (also known as the “ground truth”) for each passenger. Your model will be based on “features” like passengers’ gender and class. You can also use [feature engineering](#) to create new features.

The test set should be used to see how well your model performs on unseen data. For the test set, we do not provide the ground truth for each passenger. It is your job to predict these outcomes. For each passenger in the test set, use the model you trained to predict whether or not they survived the sinking of the Titanic.

We also include gender\_submission.csv, a set of predictions that assume all and only female passengers survive, as an example of what a submission file should look like.

## Notebooks

Overview Data [Notebooks](#) Discussion Leaderboard Rules Team [New Notebook](#)

Search notebooks
 Filters

All Your Work Shared With You Favorites Hotness ▾

	<b>Titanic Essentials : Logist,kNN,DecTree,RForest</b>	3
Updated 6h ago 0 comments · Titanic - Machine Learning from Disaster		
	<b>Pycomp: Predicting Survival on Titanic Disaster ✨</b>	7
Updated 8h ago 5 comments · Titanic - Machine Learning from Disaster		
	<b>my_first_dedicated_project</b>	1
Updated 1h ago 0 comments · Titanic - Machine Learning from Disaster		
	<b>Titanic : PyTorch Neural Network  GPU   Top 14%🔥</b>	4
Notebook copied with edits from Manav Dhamani · Updated 8h ago Score: 0.78229 · 4 comments · Titanic - Machine Learning from Disaster		

[Notebooks](#)では、ノートブックの閲覧や実行ができます。Kaggle実践編では[Notebooks](#)機能を利用します。基本的に無料で利用することができます。また一週間単位で実行時間に制限があるもののGPUやTPUを使うこともできます。

## Discussion

Overview Data Notebooks **Discussion** Leaderboard Rules Team My Submissions New Topic

2134 topics Follow Sort by Hotness ▾

All Owned Upvoted Search Topics

	Topic	Last comment	Replies
224	Getting Started Competitions - 100% Public Leaderboard Will Cukierski 6mo ago	by Mukesh Rana 1d ago	235
117	Rolling Leaderboards Will Cukierski 7y ago	by Shuang Mo 4y ago	43
0	Imputing Age Column Jamie Lou 2d ago	by fedesoriano 1d ago	5
0	Do you keep the 'Cabin' column Maria 3d ago	by Yolanda 1d ago	3
2	Dummy variables for 'pclass' Vegard Stenberg 4d ago	by Vegard Stenberg 4d ago	4
2	How to get a better score and reach top 2-3% for the Titanic competition ? Tanmay Unhale 7d ago	by Ritobrata Ghosh 3d ago	7

**Discussion**は、そのコンペに関する議論がかわされます。どういったことを試すとうまくいった・いかなかったという議論がなされるので、実際にコンペに参加する際は必ず目を通すことになるでしょう。またコンペ終了後に自らのソリューションを公開する参加者もいます。彼らの投稿はとても勉強になるので、コンペ後にも活用することになるでしょう。

## Leaderboard

Overview Data Notebooks Discussion **Leaderboard** Rules Team My Submissions **Submit Predictions**

**Public Leaderboard** Private Leaderboard

This leaderboard is calculated with all of the test data.

**Raw Data** **Refresh**

#	Team Name	Notebook	Team Members	Score	Entries	Last
1	Zeeshan Patel			1.00000	6	13d
2	Hung Khoi			1.00000	3	2mo
3	Rookie666			1.00000	2	2mo
4	Sharmith jain			1.00000	6	2mo
5	HEroKuma			1.00000	2	2mo
6	Pedro Mendes Odilon			1.00000	10	2mo
7	Sumanth Pobala			1.00000	3	2mo
8	blenderwang			1.00000	20	2mo
9	ChengDaTsai			1.00000	13	2mo
10	Inel Da			1.00000	7	2mo

**Leaderboard**は、スコアの暫定順位表です。一般的には

- Public Leaderboard (Publicテストデータセット)
- Private Leaderboard (Privateテストデータセット)

に分かれており、コンペ中は**Public Leaderboard**しか確認することができません。コンペ終了後に**Private Leaderboard**が発表され順位が決まります。

なぜ、Leader Board（テストデータセット）が分れているかというと、これはある種のハックを防ぐためです。つまり、Leader Boardのスコアを上げるために本質的とはいえないようなソリューションの変更（極端なことを言えば乱数の調整）をできないようにするためです。Leader Board（テストデータセット）が分割されていることで、**Public Leaderboard**に過度にフィットしたソリューションは**Private Leaderboard**で大きく順位を下げてしまう場合があります。

## Rules

**Rules**では、コンペのルールが確認できます。特に以下の情報は重要で、コンペに参加する際は必ず確認すべき項目です。

- チームの人数制限
- 一日の予測提出回数の上限
- スケジュール

Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions Submit Predictions

Privately sharing code or data outside of teams is not permitted. It's okay to share code if made available to all participants on the forums.

## Team Mergers

Team mergers are allowed and can be performed by the team leader. In order to merge, the combined team must have a total submission count less than or equal to the maximum allowed as of the merge date. The maximum allowed is the number of submissions per day multiplied by the number of days the competition has been running.

## Team Limits

There is no maximum team size.

## Submission Limits

You may submit a maximum of 10 entries per day.

You may select up to 5 final submissions for judging.

## Competition Timeline

Start Date: 9/28/2012 9:13 PM UTC

Merger Deadline: None

Entry Deadline: None

End Date: None

This is a fun competition aimed at helping you get started with machine learning. While the Titanic dataset is publicly available on the internet, looking up the answers defeats the entire purpose. So seriously, don't do that.

# Team

現在のチームの状況について確認できます。

Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions Submit Predictions

### Merge Teams

👤 Invite another team to merge with yours.

Team Name

Request Merge

### Pending Merge Requests

You currently have no pending merge requests.

### Teams Proposing a Merge

There are currently no teams proposing a merge with yours.

## My Submissions

提出した予測の一覧を確認することができます。ここからNotebooksに飛ぶ事もできます。

Overview	Data	Notebooks	Discussion	Leaderboard	Rules	Team	My Submissions	Submit Predictions
All	Successful	Selected						
Submission and Description								Public Score
<a href="#">task_3_titanic</a> (version 1/1) 8 months ago by tamref From "task_3_titanic" Script								0.67224
<a href="#">task_3_answer</a>								0.67464
(version 3/3) 8 months ago by tamref From "task_3_answer" Script								0.65071
<a href="#">LightGBM</a>								0.78947
(version 10/11) a year ago by tamref From "LightGBM" Script								0.77033
<a href="#">LightGBM</a>								
(version 9/11) a year ago by tamref From "LightGBM" Script								

## Submit Predictions

Submit Predictionsでは予測結果の提出ができます。

Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions **Submit Predictions**

You have 10 submissions remaining today. This resets 18 hours from now (00:00 UTC).

**Step 1**  
Upload submission file



**File Format**  
Your submission should be in CSV format. You can upload this in a zip/gz/rar/7z archive, if you prefer.

**Number of Predictions**  
We expect the solution file to have 418 prediction rows. This file should have a header row. Please see sample submission file on the [data page](#).

**Step 2**  
Describe submission

Briefly describe your submission

試しに提出をしてみましょう。Step1のアップロードアイコンをクリックし、先ほどダウンロードした [gender\\_submission.csv](#) を提出してみましょう。最終的にスコアの計算まで進めばOKです。

Getting Started Prediction Competition  
A Neural Network in PyTorch for Tabular Data...  
[yashuseth.blog/.../pytorch-neural-network-for...](#)

## Titanic - Machine Learning from Disaster

Start here! Predict survival on the Titanic and get familiar with ML basics

Kaggle · 18,034 teams · Ongoing

Overview Data Notebooks Discussion Leaderboard Rules Team **My Submissions** **Submit Predictions**

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submit.csv	just now	0 seconds	0 seconds	0.78229

**Complete**

[Jump to your position on the leaderboard ▾](#)

本教材では、以後は [Notebooks](#) から提出を行うため、この画面からはアップロードしません。

## 新規ノートブックの作成

さて、**Kaggle実践編**の準備をします。タイタニックコンペの画面で**Notebooks**を開き、**New Notebook**をクリックします。いよいよ実践編へ突入です！

## Kaggle実践編

以後の問題の解答を、**Notebooks**として公開しています。ただ、できるだけ解答を見ずに自ら検索して実装することを推奨します。問題が難しく、どうしても手につかない場合は解答の意図を理解しながらコードを書いてください。

<https://www.kaggle.com/tamreff3290/titanic-lecture-answer>

### 準備

#### 1-1. セルの実行

```
# 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

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 read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

ノートブックを作成した時点で、デフォルトで先頭のセルに数行のコードが実装されています。このセルを実行することで

- ライブラリのインポート
- 入力データのパスの確認

を達成してください。

## 1-2. ライブラリのインポート

本教材で必要なライブラリをインポートします。下記のコードをコピー&ペーストして実行してください。

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_validate
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
import lightgbm as lgb
from sklearn.ensemble import StackingClassifier
from sklearn.feature_selection import SelectKBest
```

## 1-3. データのロード

表計算ライブラリpandasを用いて、**train.csv**,**test.csv**,**gender\_submission.csv**をロードしてください。なお、今後の問題のために、代入する変数名はそれぞれ**train\_df**,**test\_df**,**ss\_df**としておくことを推奨します。

ヒント：ファイルパスは前の問題(1-1)の実行結果を参照すると良いでしょう。

```
/kaggle/input/titanic/train.csv
/kaggle/input/titanic/test.csv
/kaggle/input/titanic/gender_submission.csv
```

# EDA

## 2-1. サンプル数

学習データ(**train.csv**)およびテストデータ(**test.csv**)のサンプル数を調べてください。サンプル数とは表データにおけるレコード数、行数のことです。

ヒント：いくつかの方法があります。

```
train: 891
test: 418
```

## 2-2. 特徴量の数

学習データ([train.csv](#))の特徴量の数を調べてください。

11

ヒント：このコンペの目的やデータの説明を読んで考えてください。タイタニックコンペがどういうタスクなのかを理解しないと、答えは導けないはずです。

## 2-3. データ型

学習データ([train.csv](#))の特徴量について、それぞれのデータ型を調べてください。

ヒント：pandasではこれを一度に確認する方法が用意されています。調べてみてください。

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
 --- 
 0   PassengerId 891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object  
 4   Sex          891 non-null    object  
 5   Age          714 non-null    float64 
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object  
 9   Fare          891 non-null    float64 
 10  Cabin        204 non-null    object  
 11  Embarked     889 non-null    object  
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

## 2-4. 欠損値

タイタニックコンペのデータには欠損値があります。

学習データ([train.csv](#))およびテストデータ([test.csv](#))の特徴量について、欠損値があるかを特徴量ごとに調べてください。

```
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
```

## 2-5. 要約統計量

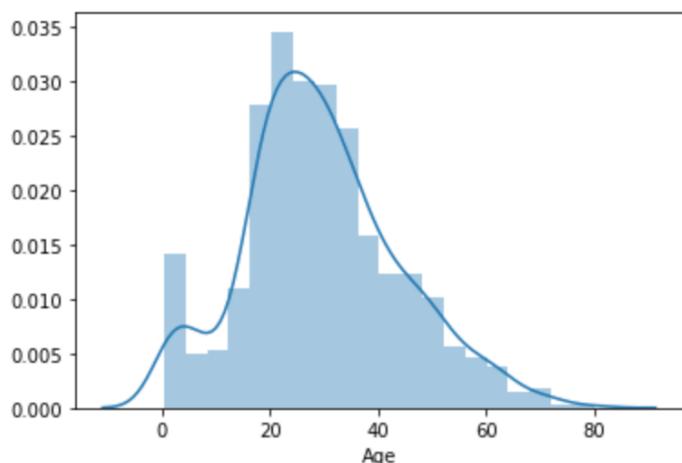
学習データ([train.csv](#))の特徴量ごとに最大値、最小値、平均値などの要約統計量を調べてください。

ヒント：pandasではそれらを一度に確認する方法が用意されています。調べてみてください。

	<b>PassengerId</b>	<b>Survived</b>	<b>Pclass</b>	<b>Age</b>	<b>SibSp</b>	<b>Parch</b>	<b>Fare</b>
<b>count</b>	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
<b>mean</b>	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
<b>std</b>	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
<b>min</b>	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
<b>25%</b>	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
<b>50%</b>	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
<b>75%</b>	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
<b>max</b>	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

## 2-6. ヒストグラム

学習データ([train.csv](#))の乗客の年齢分布をヒストグラムで示してください。

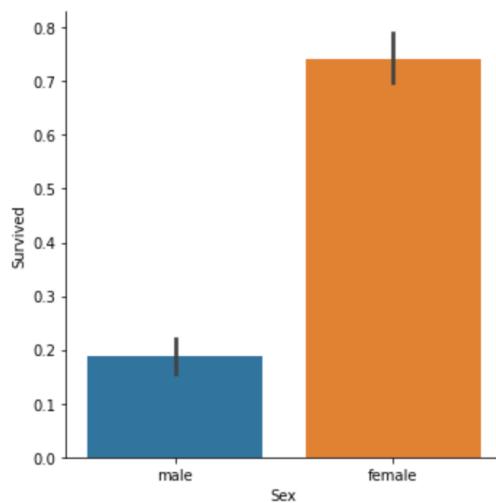


## 2-7. 集計

学習データ([train.csv](#))を用いて、性別の違いが生存率に影響を与えていそうか調べてください。ここでは統計的検定などの必要はなく、確認方法は問いません。（例えば、グラフ・表など）

```
Sex
female    0.742038
male      0.188908
Name: Survived, dtype: float64
```

<b>Survived</b>	
<b>Sex</b>	
female	0.742038
male	0.188908



## 2-8. クロス集計

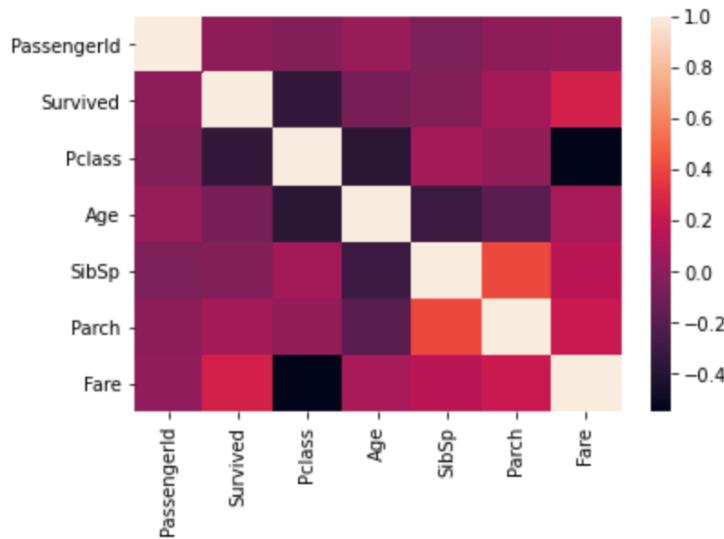
学習データ([train.csv](#))を用いて、クロス集計を行い「性別と生存者数」および「PClassと生存者数」の関係をそれぞれ調べてください。

Survived	0	1
Sex		
female	81	233
male	468	109

## 2-9. 相関係数

学習データ([train.csv](#))の特徴量のうち数値型のものについて相関係数を調べ、グラフで表示してください。

ヒント：グラフの描画にはseabornなどのライブラリを用いると良いでしょう。



## 特徴量エンジニアリング

### 3-1. ベースラインと精度評価

特微量エンジニアリングを始める前に、ベースラインとなるモデルを作成をし精度評価を行えるようにしておきます。このようなコードを実装しておくことで、特微量エンジニアリングの効果を評価することができるようになります。モデリングについては後ほど学習しますので、ここではソースコードをそのままコピー&ペーストして動かすことができれば問題ありません。精度評価の詳細が気になる方は補足を読んでください。

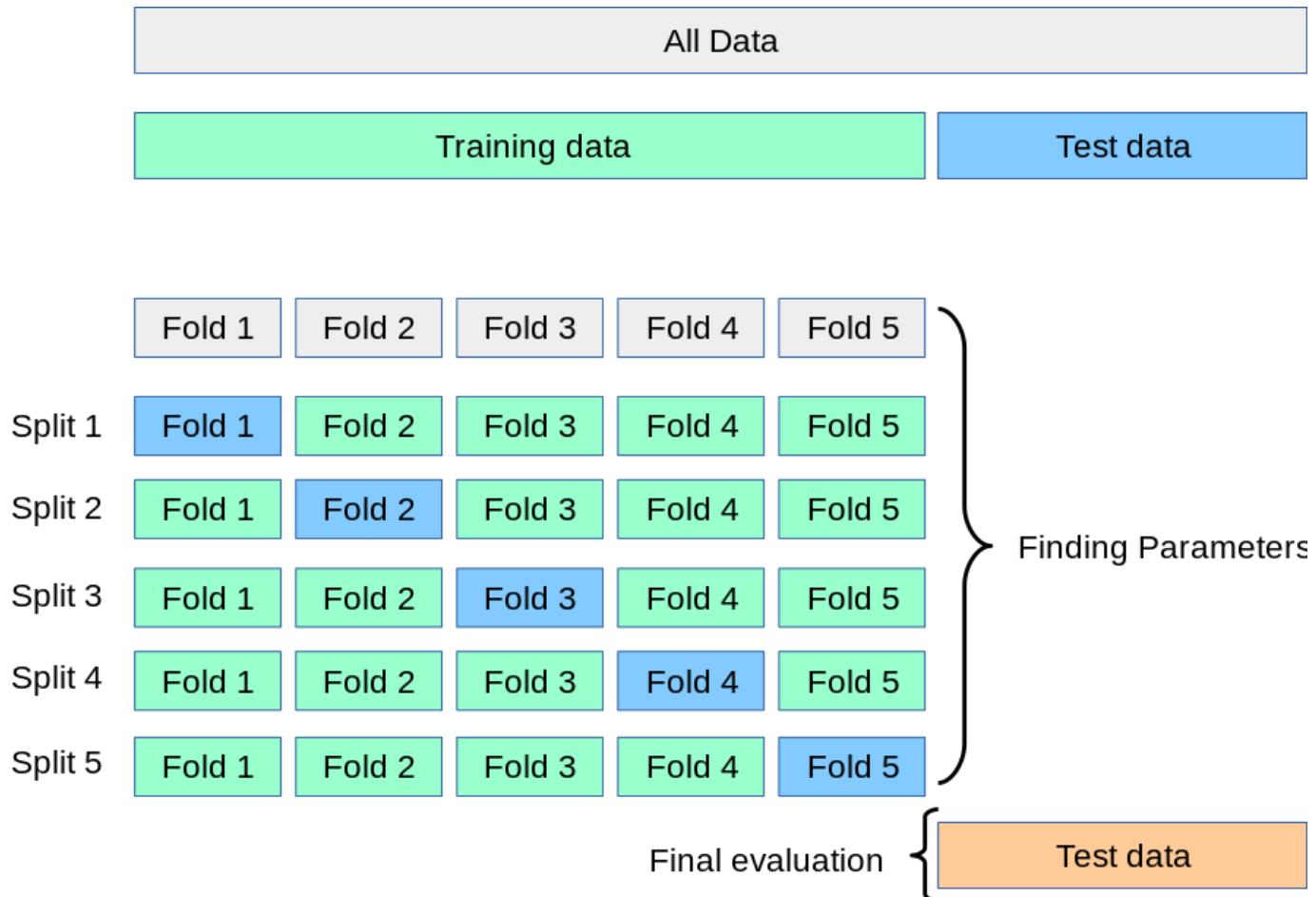
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_validate
numeric_columns = ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']
lr = LogisticRegression()
cv_results = cross_validate(lr,
train_df[numeric_columns].fillna(train_df.median()), train_df['Survived'],
scoring='accuracy')
print ('cv-score-mean', cv_results['test_score'].mean())
```

乱数の影響にもよりますが、概ね0.69程度の数字が出力されればうまく動作しています。

## 精度評価の補足

ここでは交差検証（クロスバリデーション）という方法を用いて精度評価を行っています。交差検証とは以下のようなアルゴリズムです。

1. 学習データをk分割する
2. k-1個のセットで学習する
3. 残りの1セット（検証データ）で評価する
4. 2~3を複数パターン繰り返す（通常は全部でkパターン）



※画像は[https://scikit-learn.org/stable/modules/cross\\_validation.html](https://scikit-learn.org/stable/modules/cross_validation.html)を利用させていただきました。

`sklearn.model_selection.cross_val_score`は交差検証が実行して、`scores`というディクショナリを返却します。このディクショナリは`test_score`に、交差検証の各パターンにおける検証データのスコアを格納したリストを保持しています。先のソースコードではこのリストの平均値を求めて、`cv-score-mean`として表示しています。

なお、より直接的にスコアを求めるなら、`sklearn.model_selection.cross_val_score`を使うこともできますが、こちらを利用した場合はその後にモデルを呼び出すことができないため、テストデータに対する予測ができません。そのため、この問題では`sklearn.model_selection.cross_val_score`を使用しました。

### 3-2. カテゴリ変数

3-1. ベースラインと精度評価では、一部の特徴量しか利用しませんでした。具体的には数値データ（データ型がfloat64またはint64）のみを扱っていました。

ここから、カテゴリ変数も扱うことを考えましょう。カテゴリ変数とは「数値で表せないか、あるいは数値で表せるが表すことに意味がない変数」です。例えば

- 性別
- 氏名
- 住所
- ID などが該当します。

2-3. データ型でデータ型がobjectとなっていた特徴量が候補となります。2-3の解法の一つは下記のコードです。実行して、objectを確認してみましょう。

```
train_df.info()
```

### 3-3. ユニークな要素の個数

One-Hotエンコーディングと呼ばれる処理をして、カテゴリ変数を数値に変換します。ただし、すべてのカテゴリ変数をOne-Hotエンコーディングするべきではありません。なぜでしょうか。以下の画像は、SexとEmbarkedのエンコーディングの結果です。

	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	0	1	0	0	1
1	1	0	1	0	0
2	1	0	0	0	1
3	1	0	0	0	1
4	0	1	0	0	1
...	...	...	...	...	...
886	0	1	0	0	1
887	1	0	0	0	1
888	1	0	0	0	1
889	0	1	1	0	0
890	0	1	0	1	0

891 rows × 5 columns

画像からわかるように、それぞれのカテゴリ変数に対して、ユニークな要素の個数の特徴量が生成されます。つまり、カテゴリ変数によっては膨大な特徴量が生成されてしまいます。機械学習の世界では、特徴量が膨大になると過学習の問題などが発生しうるため、一般には不用意に特徴量を増やしすぎることは推奨されません。（増やしてから後ほど選択する手法を取る場合もありますが、ここでは触れません。）

One-Hotエンコーディングするカテゴリ変数を絞り込むために、それぞれのカテゴリ変数のユニークな要素の個数を確認してみましょう。

ヒント：pandasではこれを一度に確認する方法が用意されています。調べてみてください。

### 3-4. One-Hotエンコーディング

3-3. ユニークな要素の個数の結果から、カテゴリ変数としてSexとEmbarkedは使っても問題がなさそうです。

3-1. ベースラインと精度評価のコードを参考にしながら、数値データに加えて、SexとEmbarkedをOne-Hotエンコーディングで生成した特徴量を使って、モデルを学習して精度評価を行ってください。

乱数の影響にもありますが、概ね0.79程度の数字が出力されればうまく動作しています。

ヒント：One-Hotエンコーディングにはpandas\_get\_dummies、データの結合にはpandas.concatなどを使うと良いでしょう。

	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	0	1	0	0	1
1	1	0	1	0	0
2	1	0	0	0	1
3	1	0	0	0	1
4	0	1	0	0	1
...	...	...	...	...	...
886	0	1	0	0	1
887	1	0	0	0	1
888	1	0	0	0	1
889	0	1	1	0	0
890	0	1	0	1	0

891 rows × 5 columns

### 3-5. ダミーコーディング

One-Hotエンコーディングでは性別を表すために2つの特徴量 (`Sex_female`と`Sex_male`) を生成しました。しかし、これらの特徴量はどちらかが1ならどちらかが0なわけですから、情報量としては1つの特徴量で十分です。同様の議論で、ユニークな要素の個数がk個である場合、k-1個の特徴量で十分であるという考え方があります。この考え方に基づいた特徴量の生成方法にダミーコーディングがあります。ダミーコーディングの結果は以下のようになります。

	Sex_male	Embarked_Q	Embarked_S
0	1	0	1
1	0	0	0
2	0	0	1
3	0	0	1
4	1	0	1
...	...	...	...
886	1	0	1
887	0	0	1
888	0	0	1
889	1	0	0
890	1	1	0

891 rows × 3 columns

3-4. One-Hotエンコーディングのコードを参考にしながら、数値データに加えて、`Sex`と`Embarked`をダミーコーディングで生成した特徴量を使って、モデルを学習して精度評価を行ってください。

乱数の影響にもよりますが、概ね0.79程度の数字が出力されればうまく動作しています。

### 3-6. 文字列処理

変数`Cabin`に着目してみましょう。

```
train_df['Cabin'].value_counts()
```

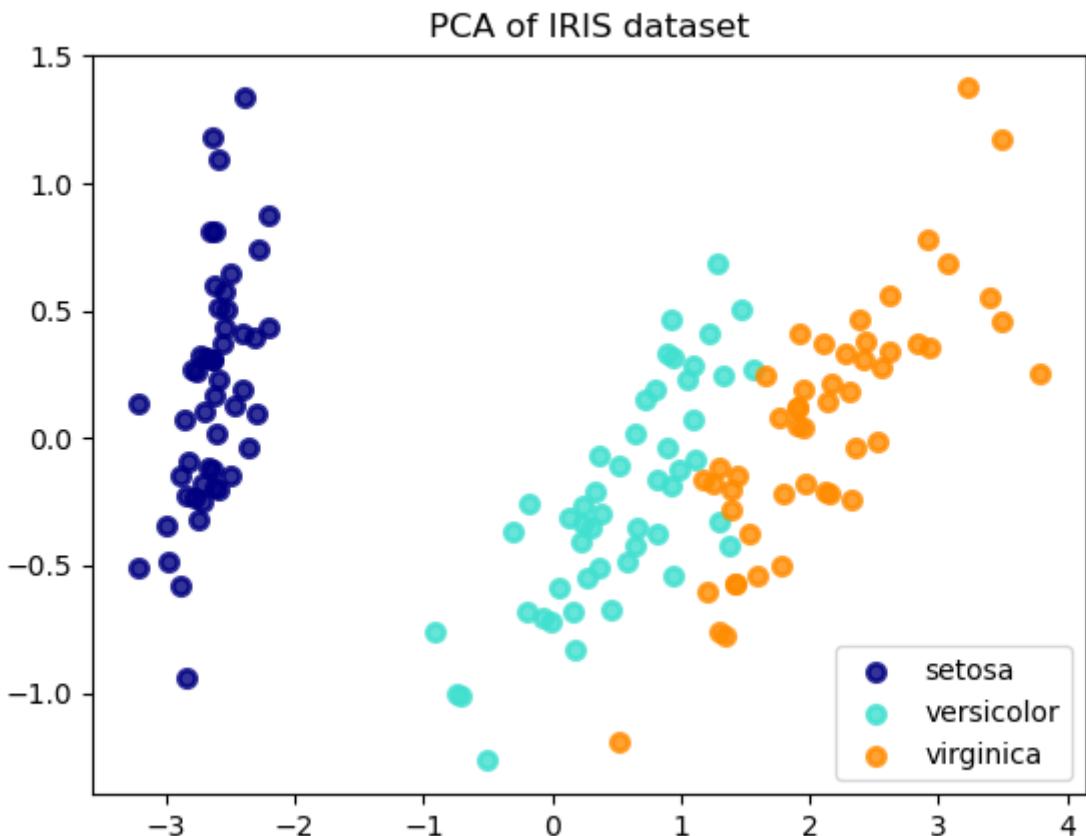
```
B96 B98      4
C23 C25 C27  4
G6          4
F33          3
C22 C26      3
...
A26          1
A6           1
A28          1
E34          1
B78          1
Name: Cabin, Length: 147, dtype: int64
```

船室（Cabin）はアルファベットから始まるようです。このアルファベットは船室の位置やグレードを表していると予想できます。特徴量として検討してみる価値がありそうです。[3-5. ダミーコーディング](#)のコードを参考にしながら、[3-5](#)の特徴量に加えて、Cabinの頭文字をダミーコーディングで生成した特徴量を使って、モデルを学習して精度評価を行ってください。

乱数の影響にもよりますが、概ね0.79程度の数字が出力されればうまく動作しています。

### 3-7. 主成分分析

教師なし学習の有名なアルゴリズムの一つに主成分分析（PCA）があります。ここでは主成分分析の数学的な意味については詳しくは触れませんが、特徴量エンジニアリングの文脈において、主成分分析は特徴量を要約する働きがあると理解してください。



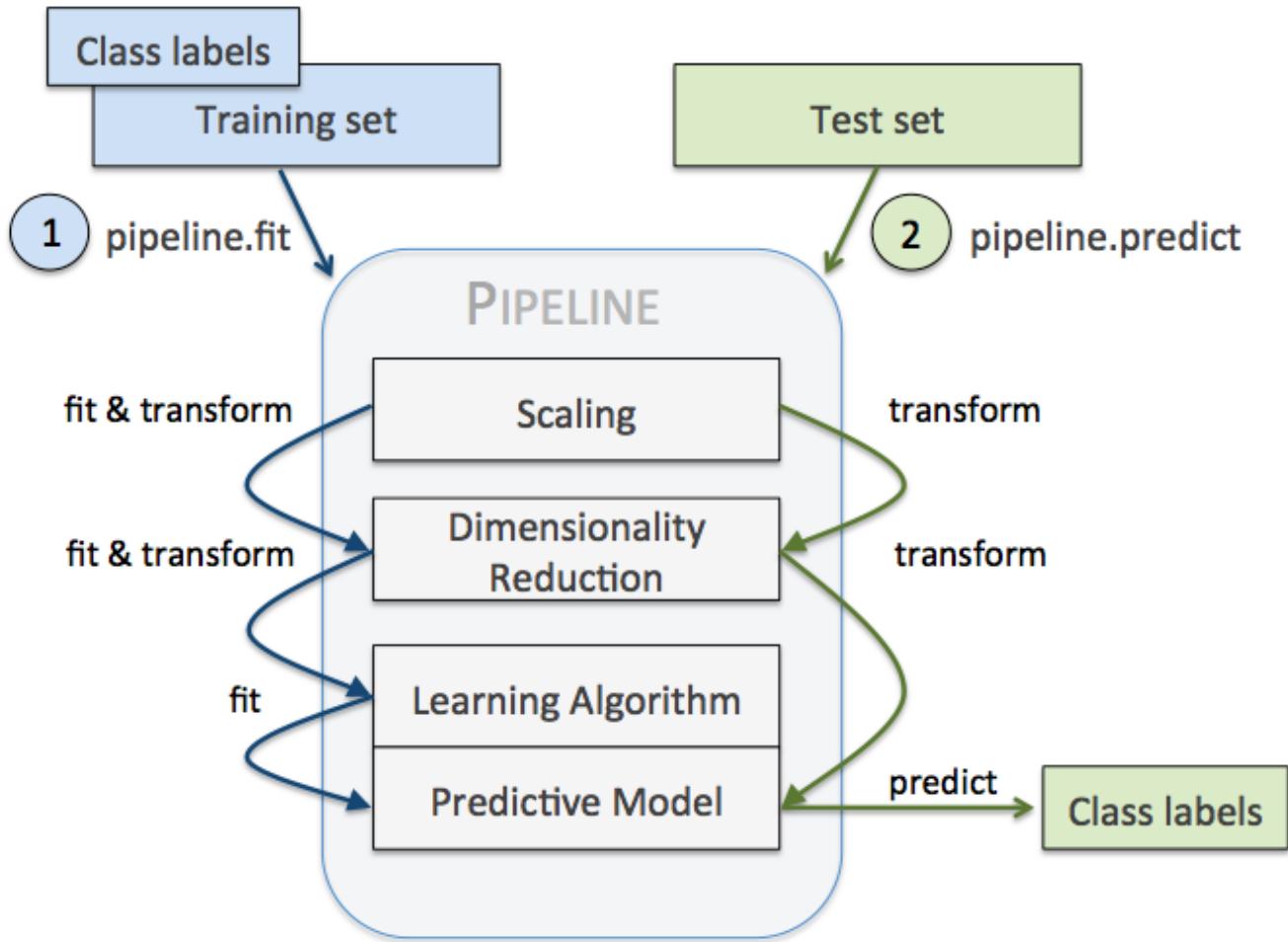
※画像は<https://scikit-learn.org/stable/modules/decomposition.html>を利用させていただきました。

[3-6](#)の特徴量を主成分分析した特徴量を使って、モデルを学習して精度評価を行ってください。

乱数の影響にもよりますが、概ね0.80程度の数字が出力されればうまく動作しています。

### 3-8. パイプライン

特徴量エンジニアリングなどの前処理や機械学習のアルゴリズム実装をすすめていくと、処理が肥大化し複雑になります。そういった課題に対処するために処理をまとめることができます。これをパイプライン処理といいます。



※画像は<https://iaml.it/blog/optimizing-sklearn-pipelines>を利用させていただきました。

`sklearn.pipeline.make_pipeline`を使って、下記のパイプライン処理を実装してください。

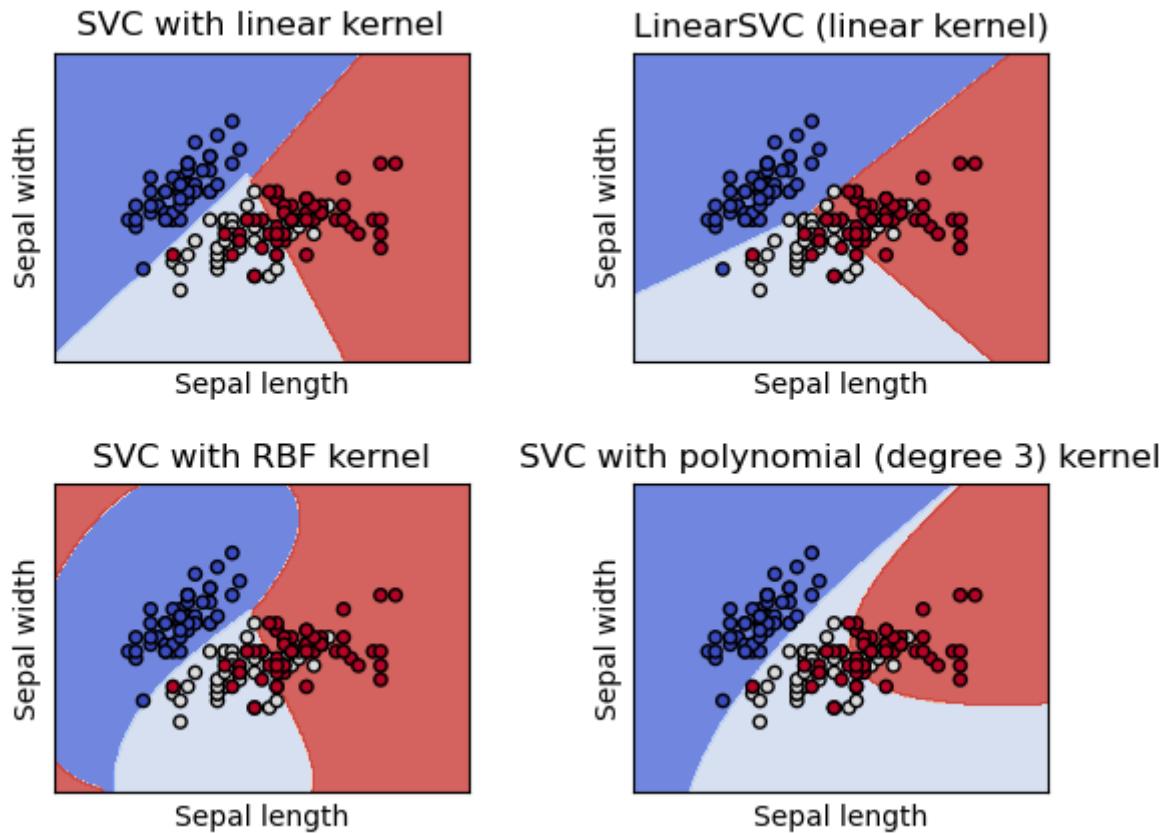
- `sklearn.preprocessing.StandardScaler`による前処理
- 主成分分析
- ロジスティック回帰の学習

乱数の影響にもよりますが、概ね0.79程度の数字が出力されればうまく動作しています。

## モーデリング

本章ではモーデリングについて学びます。ここまでではベースラインとしてロジスティック回帰を用いてきましたが、他のモデルを試してみたりハイパーパラメータの調整に取り組んでみましょう。

### 4-1. サポートベクターマシン



※画像は<https://scikit-learn.org/stable/modules/svm.html>を利用させていただきました。

教師あり学習のアルゴリズムの一つにサポートベクターマシン（SVM）があります。[3-8. パイプライン](#)で利用したコードを使って、SVMを学習して精度評価を行ってください。

乱数の影響にもよりますが、概ね0.81程度の数字が出力されればうまく動作しています。

#### 4-2. サポートベクターマシーンのハイパーパラメータ

[4-1. サポートベクターマシーン](#)で利用したコードを使って、サポートベクターマシーンを学習して精度評価を行ってください。ただし、コストパラメータ  $C=0.1$  としてください。この値が大きいほど、正則化が強く作用するため、過学習が発生しにくくなります。

乱数の影響にもよりますが、概ね0.77程度の数字が出力されればうまく動作しています。

#### 4-3. ニューラルネットワーク

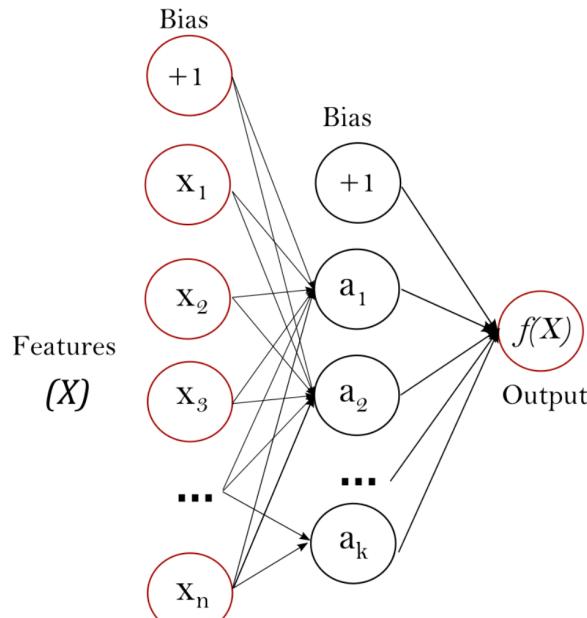


Figure 1 : One hidden layer MLP.

※画像は[https://scikit-learn.org/stable/modules/neural\\_networks\\_supervised.html](https://scikit-learn.org/stable/modules/neural_networks_supervised.html)を利用させていただきました。

教師あり学習のアルゴリズムの一つにニューラルネットワークがあります。複雑なネットワークを利用したい場合は **Keras**, **TensorFlow**, **PyTorch**などを用いますが、シンプルなネットワークであれば **scikit-learn** で対応できます。**4-1. サポートベクターマシン** で利用したコードを参考に、ニューラルネットワークを学習して精度評価を行ってください。

乱数の影響にもよりますが、概ね0.81程度の数字が出力されればうまく動作しています。

なお、ニューラルネットワークを学習した際に、以下のような警告が表示されることがあります。

```
/opt/conda/lib/python3.7/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:585:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached
and the optimization hasn't converged yet.
% self.max_iter, ConvergenceWarning)
```

`max_iter` の値が小さく、`max_iter` に達しても学習が収束しなかったことを示す警告です。精度を見ていただくとわかるように、他のモデルと比較しても遜色ない精度が出る程度には学習できているので、本教材では無視していくだけ構いません。気になる方は、以後 `max_iter` の数を増やしていただいても構いませんが、その分だけ処理に時間がかかるようになります。もし、実務等で本格的にニューラルネットワークを使う場合は GPU の利用をおすすめします。その場合は **scikit-learn** では対応していないため、先程紹介したようなフレームワークを利用することになります。

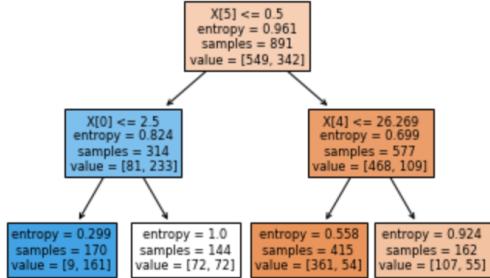
#### 4-4. ニューラルネットワークのハイパーパラメータ

**4-3. ニューラルネットワーク** で利用したコードを使って、ニューラルネットワークを学習して精度評価を行ってください。ただし、ニューラルネットワークの層数を4層（出力層含む）としてください。なお、中間層のニューロン数はいずれも10としてください。

乱数の影響にもよりますが、概ね0.82程度の数字が出力されればうまく動作しています。

## 4-5. 決定木

教師あり学習のアルゴリズムの一つに決定木があります。



画像を見るとわかるように、データから「どのように条件分岐させればよいか？」を導出するアルゴリズムです。4-1. サポートベクターマシンで利用したコードを参考に、決定木を学習して精度評価を行ってください。

乱数の影響にもよりますが、概ね0.74程度の数字が出力されればうまく動作しています。

## 4-6. 決定木のハイパーパラメータ

4-5. 決定木で利用したコードを使って、決定木を学習して精度評価を行ってください。ただし、下記のハイパーパラメータを利用して下さい。

```
params = {
    "max_depth": 3,
    "criterion": "entropy"
}
```

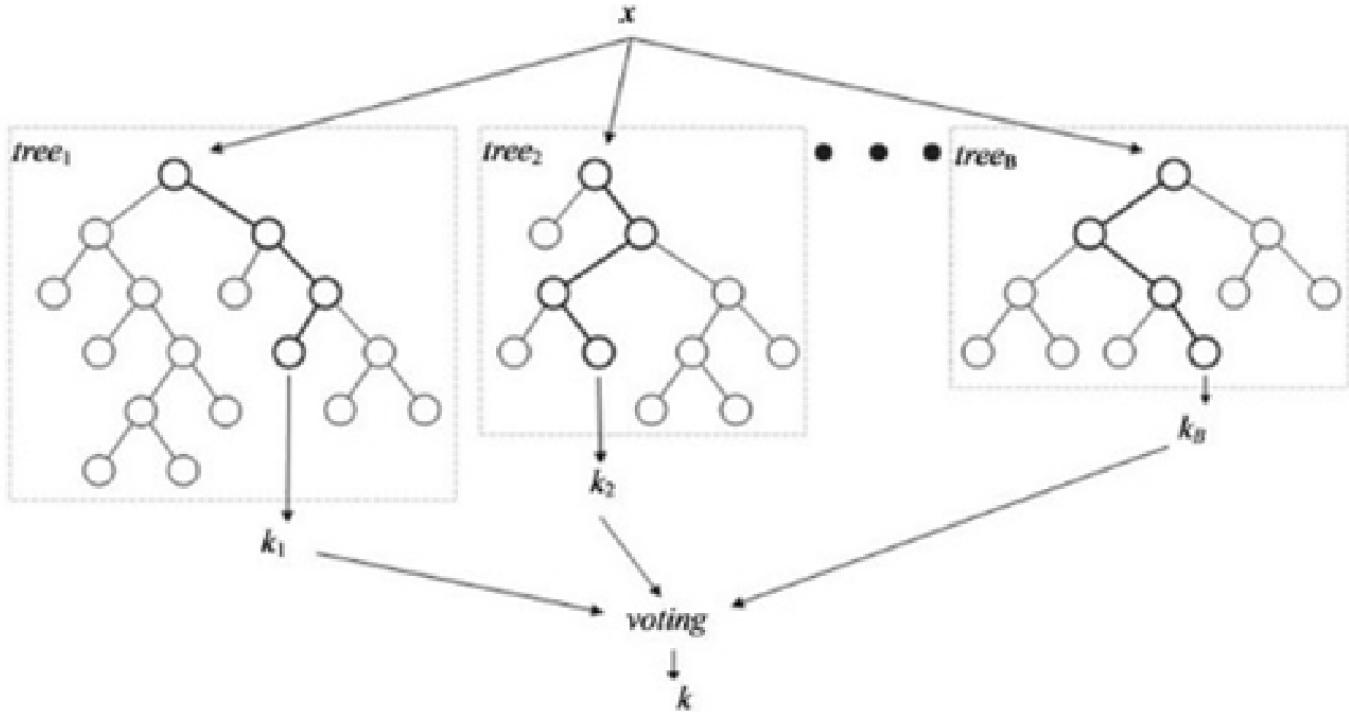
なお、他にも以下のようなハイパーパラメータが存在します。詳細はドキュメント (<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier>) を参考にしてください。

- criterion: 分割する際の指標
- splitter: 特徴量やしきい値の選択方法
- max\_depth: 決定木の深さの上限（添付した画像の場合は2に相当）
- min\_samples\_split: ノードを分岐対象とするために必要な分岐元ノードのサンプルの最小値
- min\_samples\_leaf: ノードを分岐対象とするために必要な分岐先ノードのサンプルの最小値
- max\_features: 分岐条件を探索するにあたりどの程度の特徴量を探索するか（splitter="best"で有効）
- max\_leaf\_nodes: 決定木の葉の数の上限
- class\_weight: クラスごとの重み付け

乱数の影響にもよりますが、概ね0.78程度の数字が出力されればうまく動作しています。

## 4-7. ランダムフォレスト

# Random Forest Classifier



※画像は<https://www.mygreatlearning.com/blog/random-forest-algorithm/>を利用させていただきました。

教師あり学習のアルゴリズムの一つにランダムフォレストがあります。ランダムフォレストの決定木を発展させたものです。

1. 学習データからブートストラップ（復元抽出）し、Nパターンのサブセットを作る
2. それぞれのサブセットを使って、N個の決定木を学習する
3. テストデータに対して、N個の決定木の多数決によって予測値を決定する

4-1. サポートベクターマシンで利用したコードを参考に、ランダムフォレストを学習して精度評価を行ってください。

乱数の影響にもありますが、概ね0.79程度の数字が出力されればうまく動作しています。

## 4-8. ランダムフォレストのハイパープラメータ

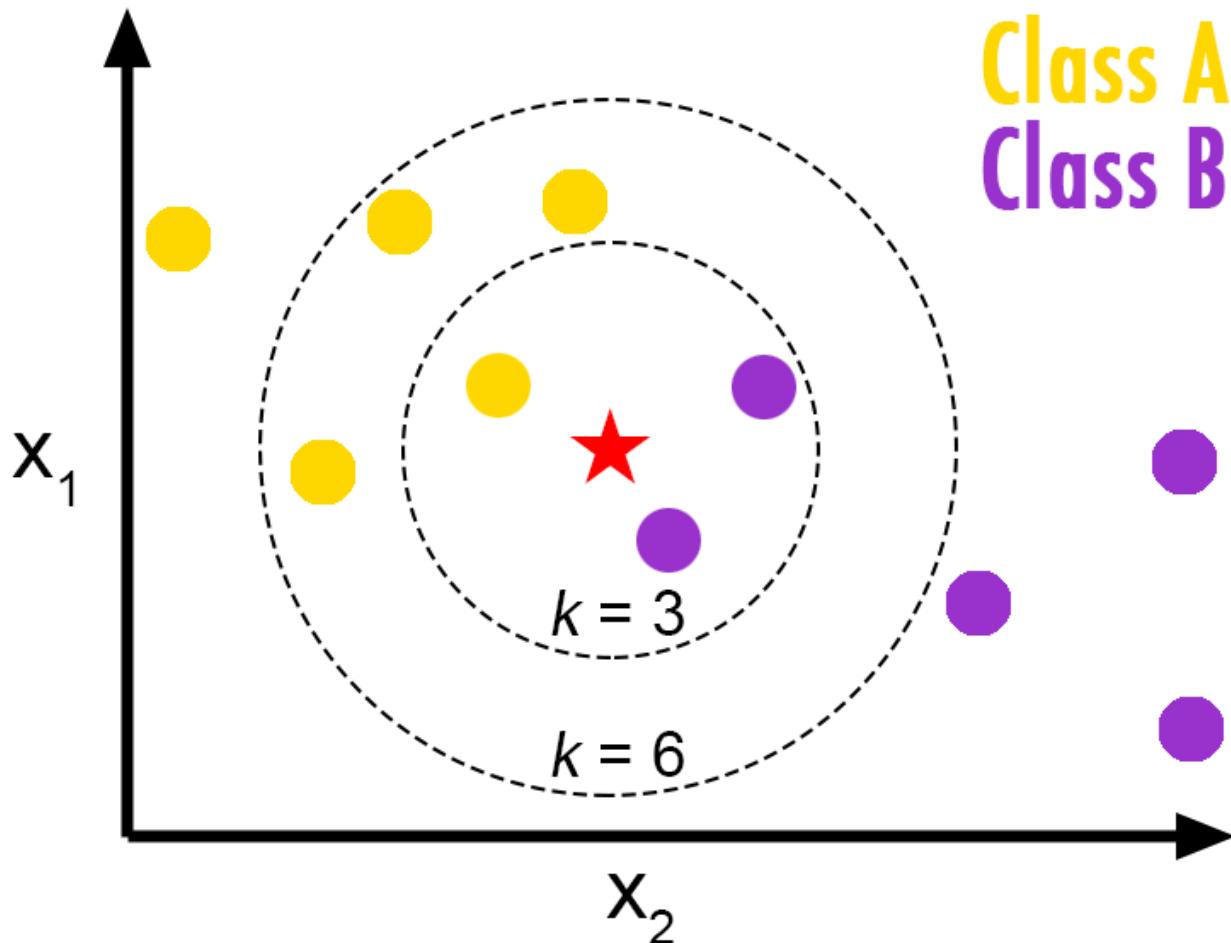
そのアルゴリズムの特性から、決定木のハイパープラメータと重複する部分が多いです。

4-7. ランダムフォレストで利用したコードを使って、ランダムフォレストを学習して精度評価を行ってください。ただし、下記のハイパープラメータを利用して下さい。

```
params = {
    "n_estimators":100,
    "max_depth":5,
    "criterion":"entropy",
}
```

乱数の影響にもよりますが、概ね0.80程度の数字が出力されればうまく動作しています。

#### 4-9. k近傍法



※画像は<https://dslytics.wordpress.com/2017/11/16/classification-series-5-k-nearest-neighbors-knn/>を利用させていただきました。

**k近傍法**はシンプルなアルゴリズムです。予測したいサンプルから特徴量空間上で距離が近い順にk個の点を参照し、そのラベルの多数決でクラス分類を行います。

4-1. サポートベクターマシンで利用したコードを参考に、k近傍法を学習して精度評価を行ってください。

乱数の影響にもよりますが、概ね0.80程度の数字が出力されればうまく動作しています。

#### 4-10. k近傍法のハイパーパラメータ

**k近傍法**の主要なハイパーパラメータに**k**があります。これは**n\_neighbors**として指定できます。4-9. k近傍法で利用したコードを使って、k近傍法を学習して精度評価を行ってください。ただし、ハイパーパラメータとして **n\_neighbors=6** を利用してください。

#### 4-11. ナイーブベイズ分類器

ベイズの定理を用いたアルゴリズムです。スパムメールのフィルタに用いられる（いた）ことで有名です。数学的にもシンプルですが、それなりに精度が出るのでベースラインとして用いられることがあります。

4-1. サポートベクターマシンで利用したコードを参考に、ナイーブベイズ分類器を学習して精度評価を行ってください。特微量にどのような分布を仮定するかバリエーションがありますが、ここではベルヌーイ分布を仮定したnaive\_bayes.BernoulliNBを用いて実装してください。

乱数の影響にもよりますが、概ね0.77程度の数字が出力されればうまく動作しています。

#### 4-12. ナイーブベイズ分類機のハイパーパラメータ

4-11. ナイーブベイズ分類器で利用したコードを使って、ナイーブベイズ分類器を学習して精度評価を行ってください。ただし、ハイパーパラメータとしてalpha=0.1を利用してください。

#### 4-13. LightGBM



最後にLightGBMを利用してみましょう。LightGBMは木系の教師あり学習モデルの一つで、テーブルデータに対して非常に高い精度を発揮することからKaggleでは根強い人気があります。4-1. サポートベクターマシンで利用したコードを参考に、LightGBMを学習して精度評価を行ってください。

乱数の影響にもよりますが、概ね0.79程度の数字が出力されればうまく動作しています。

#### 4-14. LightGBMのハイパーパラメータ

LightGBMは非常に強力なモデルですが、ややハイパーパラメータのチューニングが難しい傾向にあります。

筆者がタイタニックコンペ用にチューニングした下記のパラメータを使い、4-13. LightGBMで利用したコードを参考に、LightGBMを学習して精度評価を行ってください。

乱数の影響にもよりますが、概ね0.82程度の数字が出力されればうまく動作しています。

```
params = {
    "num_leaves": 16,
    "min_data_in_leaf": 16,
    "max_depth": 6,
    "learning_rate": 0.05,
    "feature_fraction": 0.8,
    "bagging_freq": 1,
    "bagging_fraction": 0.8,
    "lambda_l1": 0.15,
    "lambda_l2": 0.15
}
```

## 提出

さて、最後にLightGBMを使って提出用のcsvを作成します。再度、学習をしなおしてもよいですが、クロスバリデーションで学習した5モデルを再利用しましょう。この方法はアンサンブルの効果があるため、単一のモデルの場合

よりも精度がやや改善します。Kaggleではよく利用される方法です。

## 5-1. return\_estimator

`cross_validate`のパラメータとして`return_estimator`を与えることで、学習後のモデルを`estimator`として取得することができます。この方法で、`cross_validate`の返却値に`estimator`が含まれるように実装してください。[4-6. LightGBMのハイパーパラメータ](#)で利用したコードを参考にするとよいでしょう。

ヒント：ダミーコーディングは学習データとテストデータを結合して処理する必要があることに注意してください。

## 5-2. テストデータの予測

**5-1. `return_estimator`**の結果を用いて、テストデータに対する予測値を取得してください。予測値はサンプルごとに5パターン得られるので、それらの多数決により0か1かを判定してください。

```
array([0., 0., 0., 0., 0.2, 0., 0.8, 0., 1., 0., 0., 0., 0., 1.,
       0., 1., 1., 0., 0.2, 0.2, 0.8, 0., 1., 1., 0.6, 1., 0.,
       1., 0., 0., 0., 0., 0., 0., 0.2, 0.2, 0., 0.8, 0.,
       0.8, 0., 1., 0., 1., 1., 0., 0.4, 0., 1., 0., 0.4, 0.,
       1., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 1., 1.,
       1., 1., 0., 0.2, 0.6, 1., 0., 0., 1., 1., 0.2, 0., 1.,
       0., 1., 0.8, 0., 0., 0., 0., 1., 0.6, 1., 1., 0.,
       0., 1., 0., 0., 1., 0., 0.6, 0., 1., 0., 0., 0., 0.,
       0.6, 0., 0., 0., 0., 0., 1., 0.8, 0.8, 1., 0., 0.,
       0.8, 0.2, 1., 1., 0., 1., 0., 0., 0.8, 0., 0.8, 0., 0.,
       0., 0.2, 0., 0., 0., 0., 0., 0.4, 0., 0., 1., 0.4,
       0., 0., 0., 1., 0., 0.2, 0., 1., 0., 0., 0., 0., 0.,
       1., 0.6, 0.2, 0.8, 1., 0.8, 1., 0., 0., 0., 0., 0., 1.,
       0.8, 0., 0., 0., 0., 1., 1., 0.4, 0.8, 0.8, 0., 1.,
       1., 0., 1., 0., 1., 0., 0., 0., 0.2, 0.4, 0., 1.,
       0., 1., 0.4, 0., 0., 0.6, 1., 0., 1., 0., 0., 0.4, 0.,
       0.8, 0., 0., 0., 1., 1., 0., 0., 1., 1., 0., 0., 1.])
```

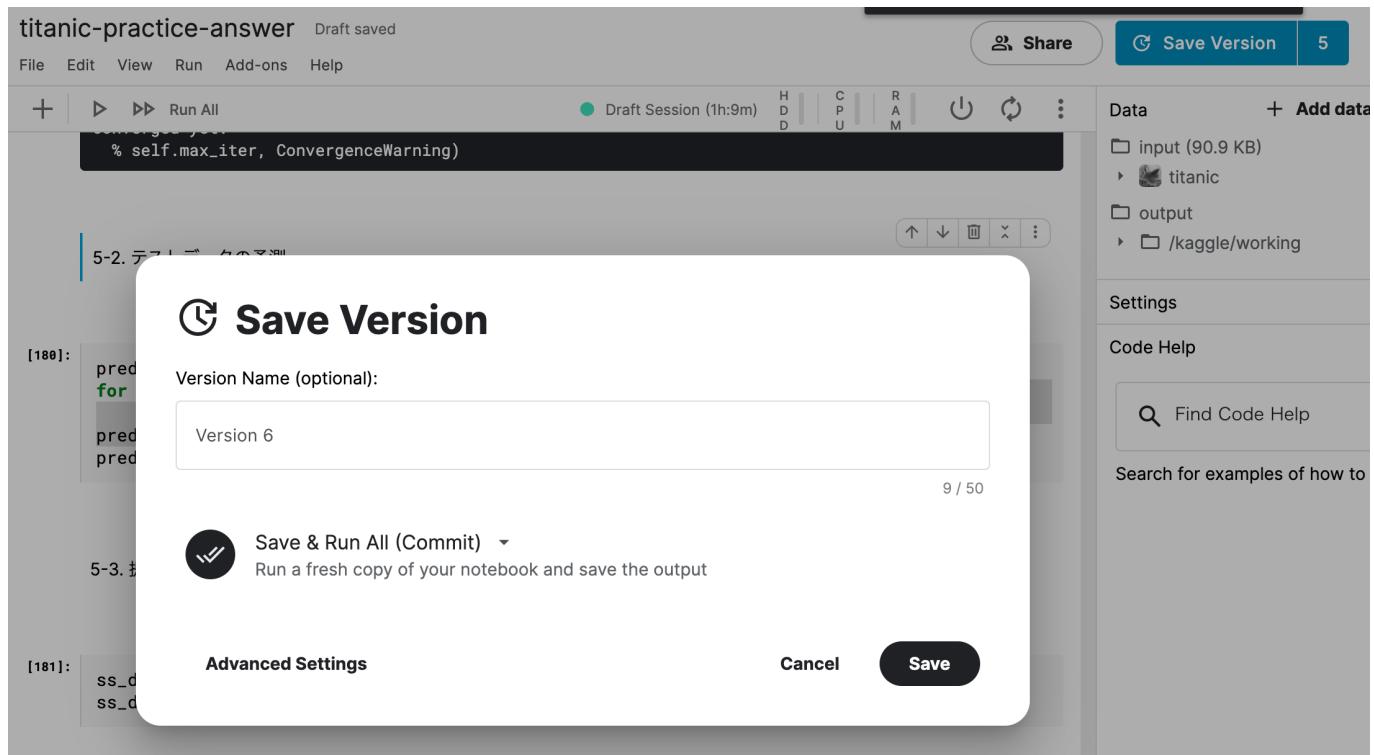
```
array([0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0])
```

## 5-3. 提出用csvの保存

最後に提出用csvを保存するコードを実装します。

## 5-4. 提出

1. 画面右上にあるSave Versionをクリックして、Saveをクリックします。



1. メニューからNotebooks > Your Workに進み、先程作成したNotebooksをクリックします。

2. 画面右にあるOutputをクリックし、表示されるSubmitをクリックします

PassengerId	Survived
892	0
893	0
894	0
895	0
896	0
897	0
898	1
899	0

Version 6 of 6

Notebook  
Input (1)  
**Output**  
Execution Info  
Log  
Comments (0)

お疲れさまでした！あとは改善あるのみです。これまで学習してきたことを復習しながら、

- 特徴量エンジニアリング
- モデリング

してみましょう。

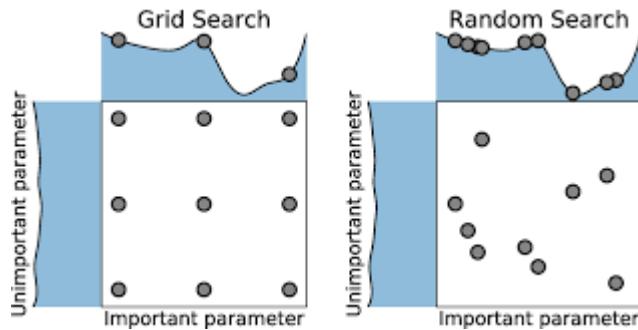
一つの目安として、Public Leaderboardにおけるスコアが0.79を超えたなら学習量としては十分です。タイタニックコンペを卒業して次のコンペにトライしてみましょう！

## 発展

ここまでKaggleにおける一通りの流れを説明しました。本章では、より発展的な内容を扱います。コンペに参加していればこれらの知識が役立つ場面があるでしょう。

## 6-1. グリッドサーチ

`cross_validate`を用いてハイパーパラメータの調整をしてもよいですが、ハイパーパラメータをどのように調整すればよいでしょうか。その方法の一つにグリッドサーチがあります。その名の通りハイパーパラメータをグリッド状に配置した点とみなして、候補となる点（ハイパーパラメータ）を試験的に学習して評価する方法です。点の数だけ、学習と評価を繰り返すことになるため、計算コストが高い点に注意が必要です。



※画像は<https://medium.com/@cj12fv/an-intro-to-hyper-parameter-optimization-using-grid-search-and-random-search-d73b9834ca0a>を利用させていただきました。

サポートベクターマシンのハイパーパラメータをグリッドサーチを使ってチューニングし、テストデータを予測してみましょう。

ヒント：ループを使って実装しても良いですが、`sklearn`に`sklearn.model_selection.GridSearchCV`として実装されており、こちらを使うのが良いでしょう。`GridSearchCV.fit`は、探索により得られた最適なハイパーパラメータを用いて、最後に学習データ全体を使って学習を行います。

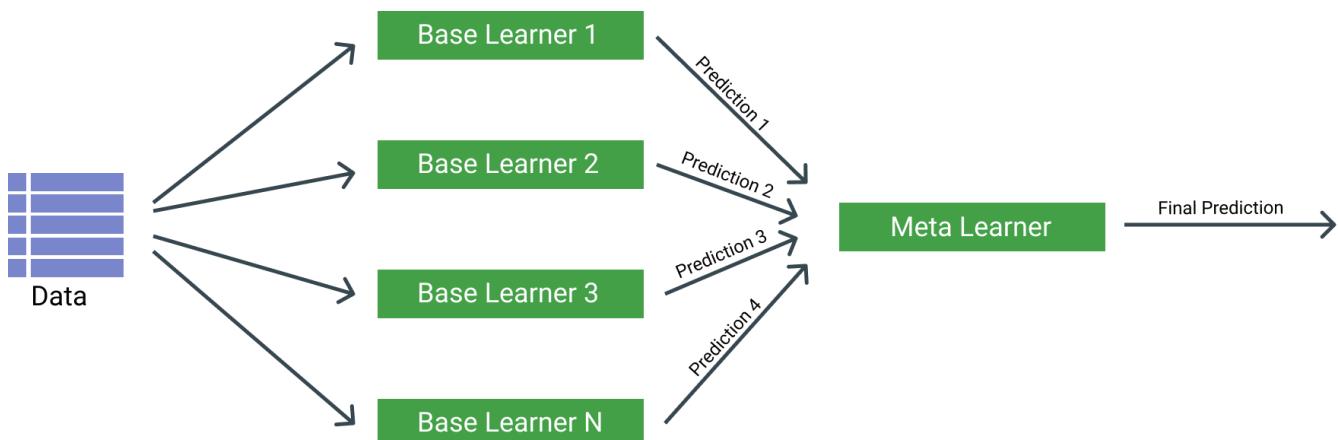
## 6-2. ブレンディング

ブレンディングは最も簡単なアンサンブルの一つです。複数のモデルで学習を行い、分類問題の場合は単純に多数決をとります。バリデーションスコアに応じて重みを付ける場合もあります。

複数のモデルを学習し、ブレンディングを行ってください。

## 6-3. スタッキング

スタッキングは、学習済みモデルの予測結果を後段のモデルの入力に用いて学習するアルゴリズムです。次の画像を見るとイメージしやすいでしょう。



※画像は<https://towardsdatascience.com/stacking-made-easy-with-sklearn-e27a0793c92b?gi=de9b5304f2ae>を利用させていただきました。

下記のモデルで学習を行い、その後段にロジスティック回帰を用いるスタッキングを実装してください。

```
estimators = [('nb', nb), ('knn', knn), ('rfc', rfc), ('lgbm', lgbm), ]
```

ヒント：`sklearn.ensemble.StackingClassifier`を使うと良いでしょう。

## 6-4. 特徴選択

特徴選択には大きく二種類の方法があります。

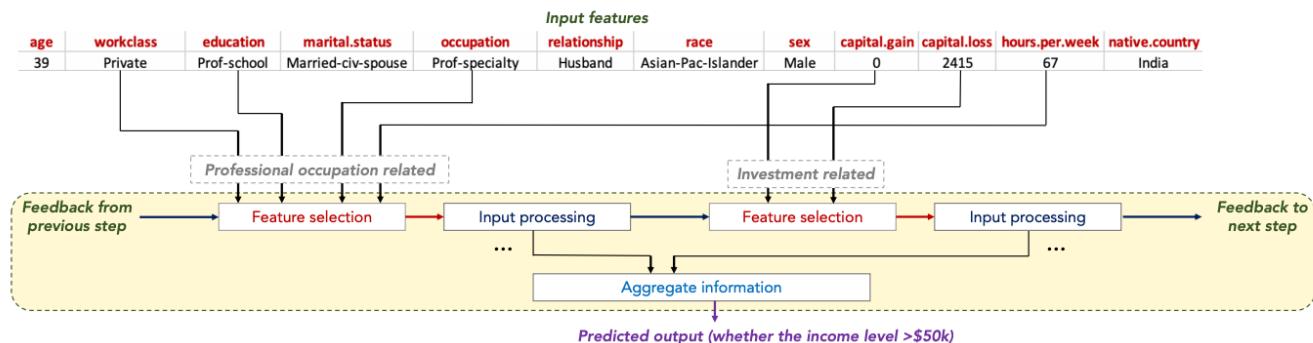
- フィルタ法
- ラッパー法

ラッパー法はモデルの学習が必要ですが、フィルタ法は不要です。フィルタ法の一つで統計検定に基づく方法である `sklearn.feature_selection.SelectKBest` を用いて、特徴選択を行ってください。なお、この処理も `sklearn.pipeline.Pipeline` に含めることができます。

## 6-5. TabNet

TabNetは2019年に公開されたNNのアーキテクチャで、テーブルデータを学習するためのモデルです。

<https://arxiv.org/abs/1908.07442>



GCP AI Platformの組み込みアルゴリズムとしても採用されていました、期待度が高いモデルです。

<https://cloud.google.com/ai-platform/training/docs/algorithms>

残念ながら `scikit-learn` には実装がありませんが、`scikit-learn` に似たインターフェイスの実装があるのでこちらを使います。

<https://github.com/dreamquark-ai/tabnet>

4-1. サポートベクターマシンで利用したコードを参考に、`TabNet`を学習して精度評価を行ってください。なお、`Kaggle`のコンテナには標準でインストールされていないので、下記のマジックコマンドでインストールして、インポートしてください。

```
!pip install pytorch-tabnet  
from pytorch_tabnet.tab_model import TabNetClassifier
```