

Pythonとデータ分析で初学者から中級者になるためのセミナー

データエンジニアになるためのテクニカルハイライ イトと学習方法

12/20, 2023

辻 真吾 (@tsjshg)

自己紹介

- 辻真吾（つじしんご）
 - 大学の研究所に勤めています
 - Pythonを使ったデータ分析が得意です
 - www.tsjshg.info
- 略歴
 - 大学院修士修了→創業したてのITベンチャー起業（入社時社員7人）
 - Javaを使ったWebアプリ開発
 - 新川のデータセンターにSun Microsystemsのサーバ設置したり
 - 2年半で退職
 - 大学院博士課程（バイオインフォマティクス）
 - 2003年ごろJavaからPythonに乗り換え

おもな著書



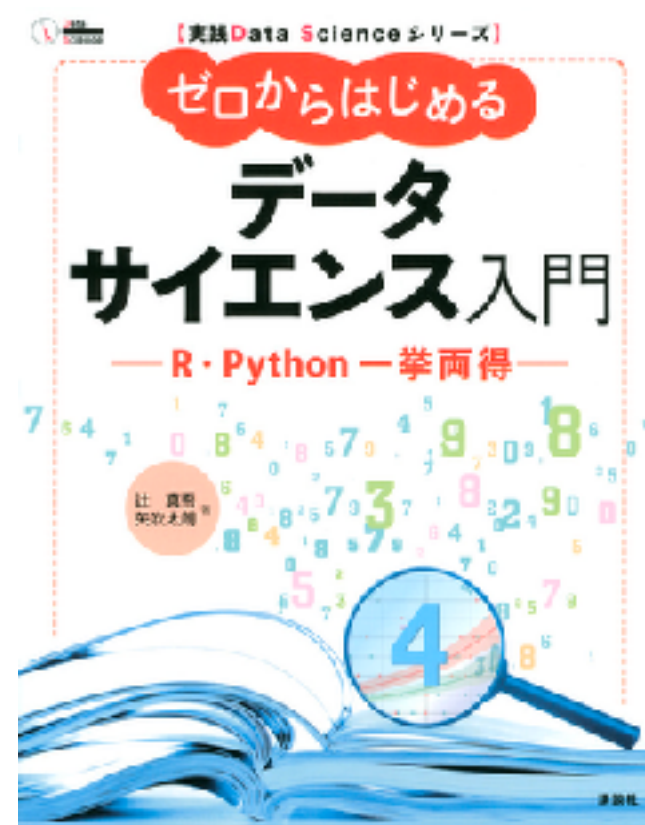
プログラミング未経験者向けに書いたPythonの入門書。いちばん売れたPythonの入門書として有名（2010, 2018改訂）



Pythonでデータサイエンスを実践するために必要な基礎ツールの解説。数学の章を執筆（2018, 2022改訂）



講談社データサイエンス入門シリーズの1冊。大学初等レベルのアルゴリズムとデータ解析の基礎をPythonで学ぶ本（2019）



データサイエンスをRとPythonの両方で学べる本。小学校の同級生との共著（2021）

『Pythonデータ分析 実践ハンドブック』が最近でました！

本日のコードはこちら

<https://colab.research.google.com/drive/16Ts-xME8KAI64sJ6olejWn1RgPi4iTup?usp=sharing>

今日のお話

データエンジニアとして成長するためにはどうしたらいいか？



実践あるのみ！

Kaggle

kaggle

Competitions Datasets Models Code Discussions Courses ...


Search

Sign In

Register

Level up with the largest AI & ML community

Join over 16M+ machine learners to share, stress test, and stay up-to-date on all the latest ML techniques and technologies. Discover a huge repository of community-published models, data & code for your next project.

 Register with Google

Register with Email



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

Kaggleのおすすめポイント

- 実践と言われても何をしたらいいか分からないときにいい
 - データとお題があふれている
- 実際にコンペティションに参加しなくてもいい
 - データだけではなく、世界中の人が書いた実際のコードを見られる
- 賞金総額10万ドルのようなコンペがある
 - お金だけではなく、ポイントやメダルと言ったゲーミフィケーション
- 日本語のサイトSIGNATE (<https://signate.jp/>) もあります

できるデータエンジニアとは？

汚いデータを処理できる

エクセルでこんなデータを受け取った

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1
10	5.4	3.7	1.5	0.2
11	4.8	3.4	1.6	0.2
12	4.8	3	1.4	0.1
13	4.3	3	1.1	0.1
14	5.8	4	1.2	0.2
15	5.7	4.4	1.5	0.4

ファイル名：bad_iris.xlsx

全部で150行あることは、エクセルで簡単に確認できます

(注) ちょっとデータに細工をしています

DataFrameにする

```
bad_iris_df = pd.read_excel('bad_iris.xlsx', index_col=0)
bad_iris_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150 entries, 0 to 149
Data columns (total 4 columns):
#   Column                Non-Null Count  Dtype
---  -
0   sepal length (cm)     150 non-null    float64
1   sepal width (cm)      150 non-null    float64
2   petal length (cm)     149 non-null    float64
3   petal width (cm)      150 non-null    float64
dtypes: float64(4)
memory usage: 5.9 KB
```

何が問題がわかるのでしょうか？

欠損値がある

より明確な欠損値の有無の確認

```
bad_iris_df.isna().any()
sepal length (cm)    False
sepal width (cm)     False
petal length (cm)    True
petal width (cm)     False
dtype: bool
```

欠損値の場所を特定する

```
idx = bad_iris_df['petal length (cm)'].isna()
bad_iris_df[idx]
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
89	5.5	2.5	NaN	1.3

欠損値をそのままにすると

K-meansクラスタリングで3グループに分けようとしてエラー

```
from sklearn.cluster import KMeans  
  
kmeans = KMeans(n_clusters=3)  
kmeans.fit_predict(bad_iris_df)
```

```
-----  
ValueError                                Traceback (most recent call last)  
Cell In[38], line 4  
      1 from sklearn.cluster import KMeans  
      3 kmeans = KMeans(n_clusters=3)  
----> 4 kmeans.fit_predict(bad_iris_df)
```

エラーメッセージが親切 (2022/5のVersion1.1.0から)

```
ValueError: Input X contains NaN.  
KMeans does not accept missing values encoded as NaN natively. For supervised learning, you might want to consider sklearn.ensemble.HistGradientBoostingClassifier and Regressor which accept missing values encoded as NaNs natively. Alternatively, it is possible to preprocess the data, for instance by using an imputer transformer in a pipeline or drop samples with missing values. See https://scikit-learn.org/stable/modules/impute.html You can find a list of all estimators that handle NaN values at the following page: https://scikit-learn.org/stable/modules/impute.html#estimators-that-handle-nan-values
```

こんどはこちら

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1
10	5.4	3.7	1.5	0.2
11	4.8	3.4	1.6	0.2
12	4.8	3	1.4	0.1
13	4.3	3	1.1	0.1
14	5.8	4	1.2	0.2
15	5.7	4.4	1.5	0.4

ファイル名：bad_iris-2.xlsx

(注) 欠損値はありませんが、別の細工をしてあります

よくある事例です

```
bad_iris2_df = pd.read_excel('bad_iris-2.xlsx', index_col=0)
bad_iris2_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150 entries, 0 to 149
Data columns (total 4 columns):
#   Column                Non-Null Count  Dtype
---  -
0   sepal length (cm)     150 non-null    float64
1   sepal width (cm)      150 non-null    float64
2   petal length (cm)     150 non-null    object
3   petal width (cm)      150 non-null    float64
dtypes: float64(3), object(1)
memory usage: 5.9+ KB
```

欠損値はないが、Dtypeが違う . . .

四になってました

87	6.3	2.3	4.4	1.3
88	5.6	3	4.1	1.3
89	5.5	2.5	四	1.3
90	5.5	2.6	4.4	1.2

引数無しのdescribeメソッドで3列しか出ないことでも気付ける

```
bad_iris2_df.describe()
```

	sepal length (cm)	sepal width (cm)	petal width (cm)
count	150.000000	150.000000	150.000000
mean	5.843333	3.057333	1.199333
std	0.828066	0.435866	0.762238
min	4.300000	2.000000	0.100000
25%	5.100000	2.800000	0.300000
50%	5.800000	3.000000	1.300000
75%	6.400000	3.300000	1.800000
max	7.900000	4.400000	2.500000

エラーとしてはstring型をfloat型に変換できない

```
from sklearn.cluster import KMeans
```

```
kmeans = KMeans(n_clusters=3)
```

```
kmeans.fit_predict(bad_iris2_df)
```

```
-----  
ValueError                                Traceback (most recent call last)
```

```
Cell In[39], line 4
```

```
1 from sklearn.cluster import KMeans
```

```
3 kmeans = KMeans(n_clusters=3)
```

```
----> 4 kmeans.fit_predict(bad_iris2_df)
```

```
File ~/.venv/ds3.11/lib/python3.11/site-packages/pandas/core/generic.py:2070, in
```

```
2069 def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
```

```
-> 2070     return np.asarray(self._values, dtype=dtype)
```

```
ValueError: could not convert string to float: '四'
```

実際には、全角の数字が混入するなど、読み込み前のファイルをぱっと確認しただけでは気付けないことも多いが、こうなる前に対処するのができるデータエンジニア（だと思おう）

不揃いなデータと戯れる

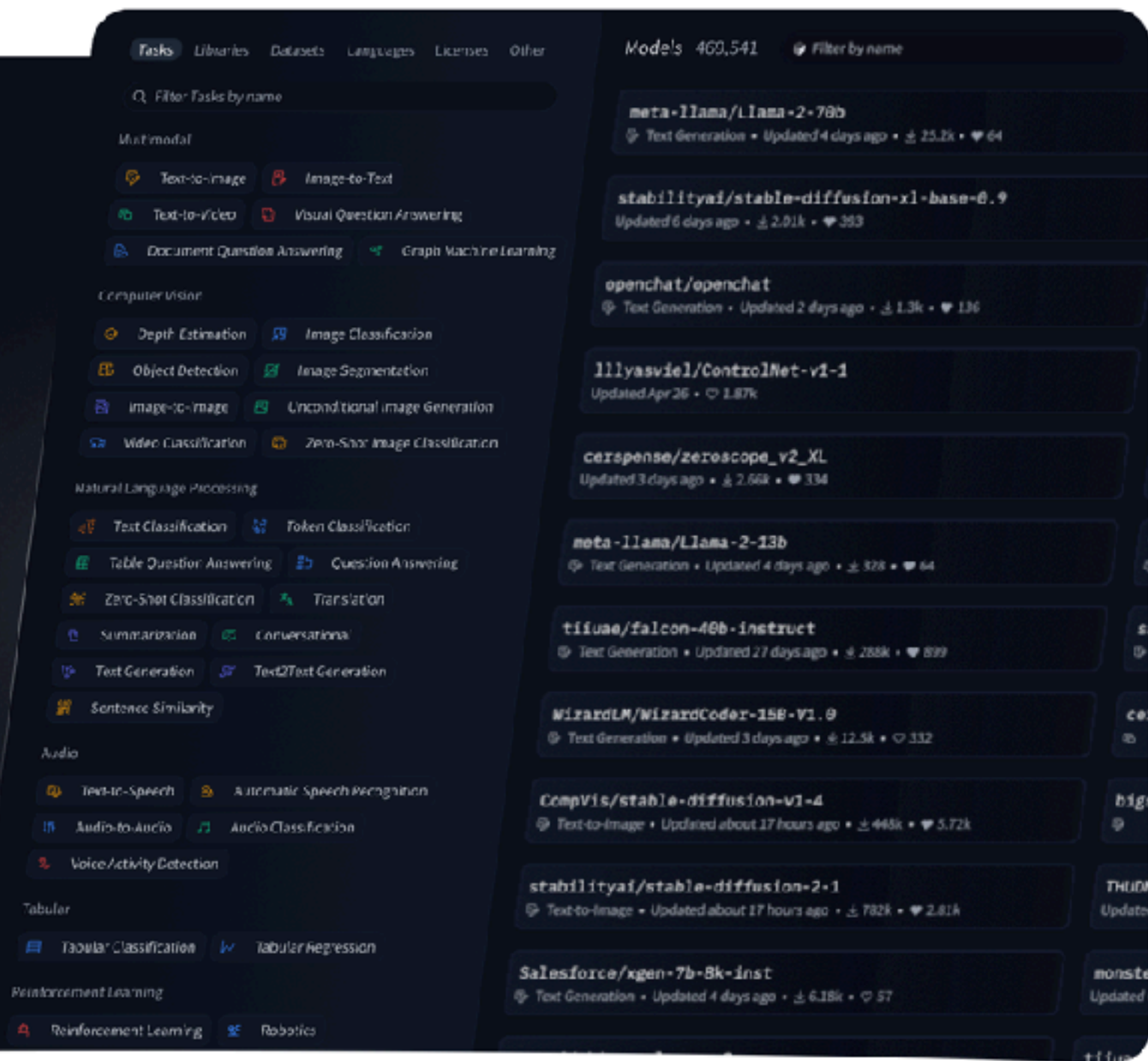
- 実際の仕事では、汚いデータが沢山ある
- Webスクレイピングなどでデータを用意するところからはじめるとうとうしたデータを手に入れる可能性が高い
- 欠損値や予期しない値の混入があることを前提に解析する
 - データ型の違いを常に意識するとよいかも
- (1つの解決策) データの管理にSQLサーバを利用する
 - 列ごとのデータ型を固定できる
 - pandasのDataFrameとも相性抜群
 - 周辺技術へのデータエンジニアには必須

”手軽にできる”を加速するHugging Face



The AI community building the future.

The platform where the machine learning community collaborates on models, datasets, and applications.



<https://huggingface.co/>

Google Colaboratoryがおすすすめ

学習済みの画像分類モデルを使うサンプルコード

```
1 from transformers import AutoImageProcessor, ResNetForImageClassification
2 import torch
3 from datasets import load_dataset
4
5 # モデルの準備
6 processor = AutoImageProcessor.from_pretrained("microsoft/resnet-50")
7 model = ResNetForImageClassification.from_pretrained("microsoft/resnet-50")
```

```
1 # サンプルデータの準備
2 dataset = load_dataset("huggingface/cats-image")
3 image = dataset["test"]["image"][0]
```

サンプルデータも
Hugging Faceにある



```
1 def predict_image(image):
2     # 画像データの前処理
3     inputs = processor(image, return_tensors="pt")
4     with torch.no_grad():
5         logits = model(**inputs).logits
6     # 1000個のクラスの中から画像に写っているモノを予想
7     predicted_label = logits.argmax(-1).item()
8     return model.config.id2label[predicted_label]
```

```
1 predict_image(image)
```

```
'tiger cat'
```


自分で用意した画像を使うには？

Hugging Faceにある訓練済みのmicrosoft/resnet-50 モデルを使って、自分で用意した画像を読み込ませるにはどうしたらいいですか？

≡ Answer

Hugging Faceのmicrosoft/resnet-50モデルを使用して、自分で用意した画像を読み込ませるには、以下の手順に従います。

1. まず、Hugging FaceのTransformersライブラリをインストールします。
2. 次に、PyTorchやTensorFlowなどのバックエンドをインストールします。
3. その後、次のコードを使用して、画像を読み込んでモデルに渡します。

```
python
from PIL import Image
import torch
from torchvision import transforms
from transformers import AutoModel, AutoTokenizer

# 画像の読み込み
image = Image.open("your_image_path.jpg")
```

すごいものがあふれている

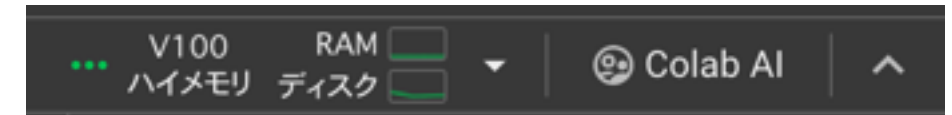
- Hugging FaceとGoogle ColabのおかげでWebブラウザ1つで画像分類モデルをすぐに試すことができる
- わからないことがあったら大規模言語モデルを基盤にした生成AIを使ってみる
 - 暗中模索ではなく、簡単に一筋の光が見える
- PIL (Pillow) はPythonで画像を扱うときによく使われる外部ライブラリ
 - 『Pythonデータ分析 実践ハンドブック』の8章にくわしい解説があります

流行の大規模言語モデルを試してみる

モデルの準備

```
1 # https://huggingface.co/elyza/ELYZA-japanese-Llama-2-7b-instruct
2
3 import torch
4 from transformers import AutoModelForCausalLM, AutoTokenizer
5
6 # モデルの準備に5分ほどかかります。
7 model_name = "elyza/ELYZA-japanese-Llama-2-7b-instruct"
8 tokenizer = AutoTokenizer.from_pretrained(model_name)
9 model = AutoModelForCausalLM.from_pretrained(model_name, torch_dtype="auto")
```

```
1 def prompt(text, default_system_prompt="", do_sample=False,
2             max_tokens=1024, model=model, tokenizer=tokenizer):
3     B_INST, E_INST = "[INST]", "[/INST]"
4     B_SYS, E_SYS = "<<SYS>>\n", "\n<</SYS>>\n\n"
5     DEFAULT_SYSTEM_PROMPT = default_system_prompt
6     if torch.cuda.is_available():
7         model = model.to("cuda")
8     prompt = "{bos_token}{b_inst} {system}{prompt} {e_inst} ".format(
9         bos_token=tokenizer.bos_token,
10        b_inst=B_INST,
11        system=f"{B_SYS}{DEFAULT_SYSTEM_PROMPT}{E_SYS}",
12        prompt=text,
13        e_inst=E_INST,
14    )
15    with torch.no_grad():
16        token_ids = tokenizer.encode(prompt, add_special_tokens=False, return_tensors="pt")
17        # generateメソッドに引数でバラバラに渡す方法は、deprecatedらしい
18        config = GenerationConfig(max_new_tokens=max_tokens,
19                                  do_sample=do_sample,
20                                  pad_token_id = tokenizer.pad_token_id,
21                                  eos_token_id = tokenizer.eos_token_id)
22        output_ids = model.generate(token_ids.to(model.device), generation_config=config)
23        output = tokenizer.decode(output_ids.tolist()[0][token_ids.size(1) :], skip_special_tokens=True)
24        return output
```



ハイメモリのインスタンスに接続

指示を受け取って返答する関数

それっぽい返答が来る

```
1 sys_prompt = "あなたはデータサイエンス、AI、機械学習アルゴリズムの専門家です。"  
2 text = "データエンジニアとして成長するには何が必要か教えてください。"  
3  
4 print(prompt(text, sys_prompt))
```

データエンジニアとして成長するために必要なことを以下に示します。

1. プログラミングスキル: データサイエンスの分析や機械学習のアルゴリズムを実装するにはプログラミングスキルが必要です。PythonやRなどの言語を学ぶ必要があります。
2. データ分析の知識: データサイエンスの分析方法や機械学習のアルゴリズムを選択するにはデータ分析の基礎知識が必要です。統計学や機械学習の基礎を学ぶ必要があります。
3. ビジネス知識: データサイエンスのアルゴリズムを実装する際にはビジネスの知識が必要です。顧客のニーズやビジネス目標を理解する必要があります。
4. コミュニケーションスキル: データサイエンティストは開発チームや顧客と協力して作業を行うことがあります。そのため、コミュニケーションスキルを磨く必要があります。
5. 経験: データサイエンスのアルゴリズムを実際に複数のプロジェクトで使用し、経験を積むことでスキルを向上させることができます。
6. 学習意欲: データサイエンスや機械学習のアルゴリズムは日々進化しているため、常に新しい知識を学ぶ必要があります。

パラメータを変更して、返答がすこし変わる様子

```
1 print(prompt(text, sys_prompt, do_sample=True))
```

データエンジニアとして成長するために必要なことをお教えします。

データエンジニアとして成長するために必要なことは、以下の5つです。

1. 基礎知識: データサイエンス、AI、機械学習の基礎知識を確実に身につけます。
2. プログラミングスキル: データサイエンスや機械学習を行うために必要なプログラミング技術を身につけます。
3. データ分析のスキル: データを整理・並列化・可視化する必要があります。
4. データサイエンスのアルゴリズム: 機械学習のアルゴリズム、特にディープラーニングやニューラルネットワークを学びます。
5. ビジネス知識: データサイエンスをビジネス上で活かすために、ビジネス知識を学び、業界の動向を把握します。

これらをバランスよく身につけることで、成長が期待できます。

何かやろうとすると必ずエラーに遭遇

```
1 kmeans = KMeans(n_clusters=3)
2 kmeans.fit_predict(bad_iris2_df)
```

ValueError Traceback (most recent call last)
<ipython-input-9-e9406eb8b0dc> in <cell line: 2>()
1 kmeans = KMeans(n_clusters=3)
----> 2 kmeans.fit_predict(bad_iris2_df)

5 frames

/usr/local/lib/python3.10/dist-packages/pandas/core/generic.py in __array__(self, dtype)
2068
2069 def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
-> 2070 return np.asarray(self._values, dtype=dtype)
2071
2072 def __array_wrap__(

ValueError: could not convert string to float: '四'

エラーの説明 ← これを押す

先ほどの「四」が入っているDataFrameをColabで実行

解決策を教えてください



あなた

エラーの内容をご説明ください。

```
ValueError: could not convert string to float: '四'
```



Colab AI

The error is caused by the value '四' in the column 'sepal length (cm)' of the `bad_iris2_df` DataFrame. This value cannot be converted to a float, which is required by the `fit_predict()` method of the `KMeans` class.

To fix the error, you can remove the row containing the value '四' from the `bad_iris2_df` DataFrame. You can do this using the following code:

```
bad_iris2_df = bad_iris2_df[bad_iris2_df['sepal length (cm)'] != '四']
```

[Use code with caution](#)

Once you have removed the row containing the value '四', you can fit the `KMeans` model to the `bad_iris2_df` DataFrame. You can do this using the following code:

```
kmeans = KMeans(n_clusters=3)
kmeans.fit_predict(bad_iris2_df)
```

[Use code with caution](#)

この回答を評価  

AIアシスト機能はプログラミングの強い味方

- 「エラーの説明」は強力
 - もちろん的外れのこともあるが、こんな機能はかつて無かったので、プログラミング学習の敷居がまた大きく下がった印象
- コード自動生成もある
 - メソッドや属性の入力補完ではなく、pickleでデータを保存するコードなど複数行がまるまる出てくる
 - ColabPro（月額約1,000円）以上の機能みたいです
 - GitHub Copilotを含め、コード自動生成は私自身実は避けてきた
 - あまりの便利さに驚いた・・・

まとめ

- データエンジニアとし成長するには実践あるのみ！
 - コードを書くスキル
 - 周辺技術（例：SQLなど）の知識
- とにかくなにかやってみる
 - Kaggleでデータを探したり、Hugging Faceでモデルを探したり
- ひとまず実行してみたあとは
 - 仕組みに興味をもって調べてみる→周辺技術に関する知識の拡大
- AIの活用
 - 生成系AIにはとにかく聞いてみればいい
 - コードを書くときにAIアシスト機能を活用すると負担がだいぶ減る

ご清聴ありがとうございました

月に1回zoomで気軽に参加できる勉強会

『みんなのPython勉強会』を開催中

次回は2024年1月25日（木）19:00～なんと100回記念

<https://startpython.connpass.com/>

