SIGNATE

Confidential

AI Quest: 需要予測・在庫最適化
ガイドコンテンツ

01

当課題特有の注意点について

当課題特有の注意点について

SIGNATE

- **当課題の問題設計やデータの性質を正しく把握した上で、分析を行いましょう。**

✓ 2ヶ月前までの売上履歴をもとに予測できるよう、モデリングする必要がある

2018年	1月	2月	3月	4月	5月	6月	7月	8月	9月	10月	11月	12月
2019年	1月	2月	3月	4月	5月	6月	7月	8月	9月	10月		12月

青: 売上履歴が与えられた期間

赤: 予測対象期間

✓ 予測対象は、2019年12月の「**一ヶ月あたりの商品売上個数**」だが、
学習用として与えられたデータは、2018年1月～2019年10月までの「**一日あたりの売上履歴**」である

✓ 売上履歴は、「**一日の売上数が0個でなかった場合**」のみデータが存在するが、
予測対象は、売上数が0個の場合も含まれる

02

前処理コード例

テーブルの読み込み: pandas.read_csv()

SIGNATE

- **まずは、配布されたCSVファイルを読み込み、中身を確認しましょう。pythonのpandasライブラリにおける例を示します。**

▼read_csv関数の例

ファイル名をクォーテーションマーク(「‘」もしくは「”」)で囲う必要があることに注意しましょう。

```
df = pd.read_csv（’sales_history.csv’）
```

[1] ライブラリ独自の関数を使用する場合は、
事前にライブラリのインポートを行う必要があります。

[2] Jupyter Lab(Notebook)環境を使用する場合は、
読み込んだデータを代入した変数名を記入して
実行するだけで、テーブルの中身を見やすい
レイアウトで表示することができます。

```
In [1]: import pandas as pd

        sales = pd.read_csv('sales_history.csv')

In [2]: sales
```

Out[2]:

	日付	月ブロック	商品ID	店舗ID	販売価格	売上個数
0	2018-01-01	0	1000007	0	250	1.0
1	2018-01-01	0	1000007	15	130	1.0
2	2018-01-01	0	1000008	0	250	1.0
3	2018-01-01	0	1000008	15	130	1.0
4	2018-01-01	0	1000009	15	130	1.0
...	...	...	...	...	...	...

グループ単位の集計: pandas.DataFrame.groupby()

SIGNATE

- **グループごとに値を集計することで、有用な情報を得られる場合があります。グループ分け処理には、pandasのgroupby()メソッドを使用します。**

▼groupby()メソッドの例

他にも、合計値を算出するsum()
値の数を数え上げるcount()などがあります。

```
gp = A.groupby('曜日').mean().reset_index()
```

データフレームA

	曜日	売上数
0	火	2
1	火	4
2	水	5
3	水	4
4	水	9
5	土	12
6	土	19
7	土	26

「火」の平均値: 3

「土」の平均値: 19

データフレーム gp

	曜日	売上数
0	火	3
1	水	6
2	土	19

データの可視化: matplotlib.pyplot.bar()

SIGNATE

- **データをグラフに描画することで、データに対する理解が深まることがあります。データの可視化に便利なmatplotlibライブラリを使ってみましょう。**

▼plt.bar()関数の例

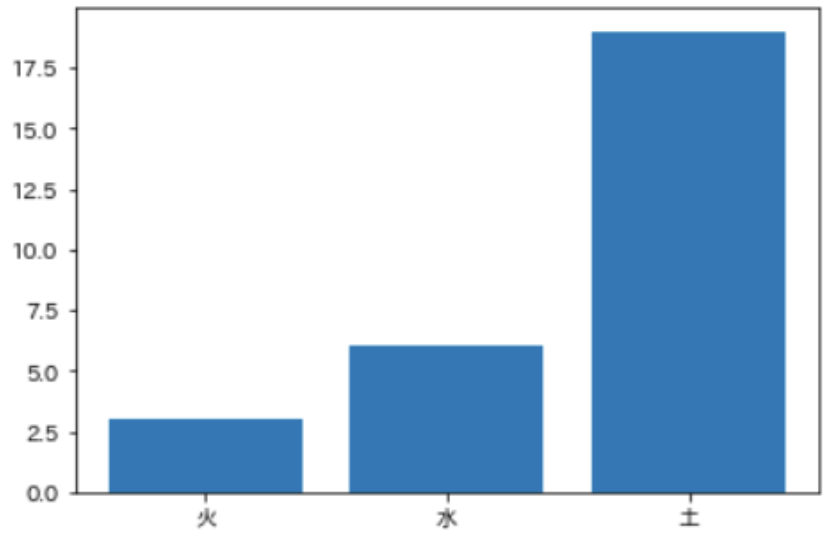
データフレーム名の後ろの[]の中に列名を指定することで、
その列に含まれる値の一覧を取得することができます。

plt.bar(gp['曜日'], gp['売上数'])

matplotlib.pyplotモジュールは、慣例的に
「plt」という省略名でインポートされます。

データフレーム gp

	曜日	売上数
0	火	3
1	水	6
2	土	19



テーブルの結合処理①: pandas.concat()

SIGNATE

- **テーブル型のデータでは、結合処理を頻繁に利用します。**
  まずは、単純に2つのテーブルをつなげる処理の例を確認しましょう。

▼concat関数の例

[ ]が必要なので注意。良く忘れます

```
dfnew = pd.concat（[A, B],sort=False）
```

データフレームA

	val1	val2
0	A1	B1
1	A2	B2
2	A3	B3
3	A4	B4

＋

データフレームB

	val1	val2
4	A5	B5
5	A6	B6
6	A7	B7
7	A8	B8

dfnew

	val1	val2
0	A1	B1
1	A2	B2
2	A3	B3
3	A4	B4
4	A5	B5
5	A6	B6
6	A7	B7
7	A8	B8

2つのテーブルが縦に結合される

テーブルの結合処理②: pandas.merge()

SIGNATE

- 次は「特定の値をヒントとして結合する」場合の例です。

▼merge関数の例

ヒントとなるカラムをオプションで設定

dfnew = pd.merge（A, B, on=“id”）

データフレームA

	val1	id
0	JP	01
1	US	02
2	CN	03
3	GB	04

＋

データフレームB

	id	val2
0	04	イギリス
1	03	中国
2	02	アメリカ
3	01	日本

カラムidをヒントに結合する

dfnew

	val1	id	val2
0	JP	01	日本
1	US	02	アメリカ
2	CN	03	中国
3	GB	04	イギリス

カラムidで同一の値があるもの同士が横に結合される

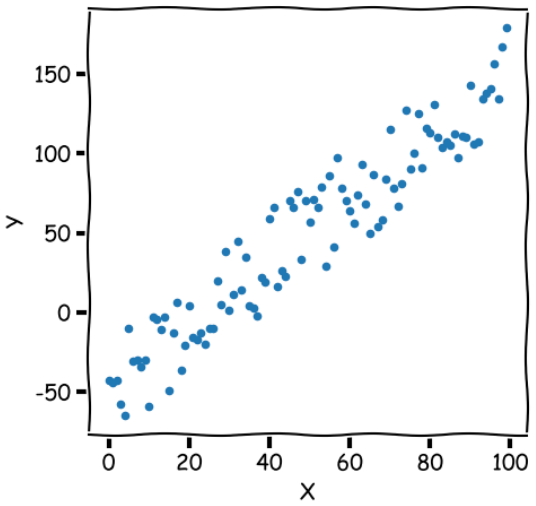
03

特徴量生成・モデリングの方針

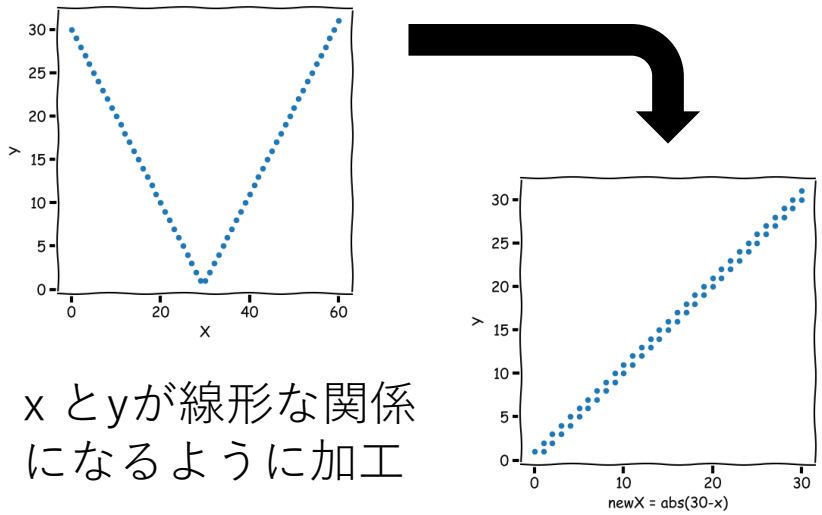
精度向上のコツ①: 特徴量の加工

SIGNATE

- **予測に有効な特徴量を作成・見つけることが重要です。**
  特に、回帰問題の場合には線形性がある特徴量が重要となります。



xとyの相関が高そう
→予測に寄与する変数である可能性



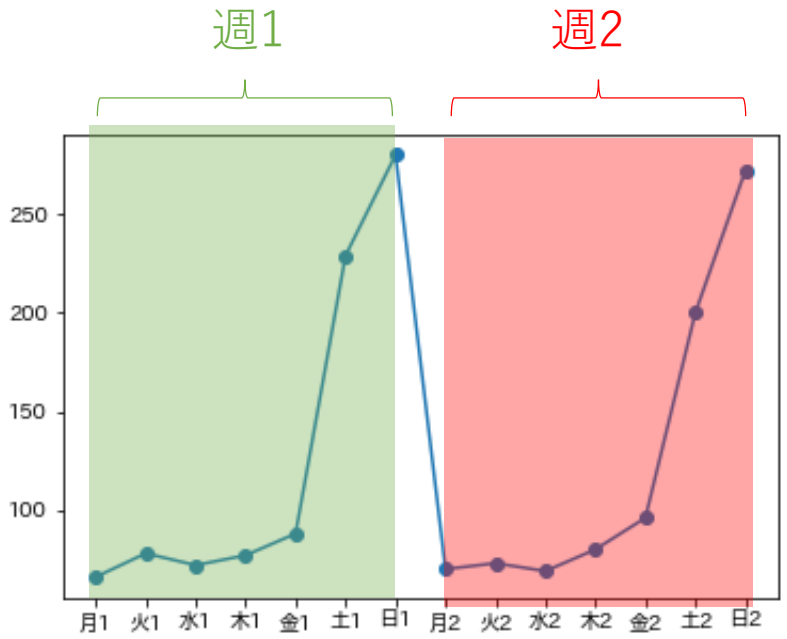
x とyが線形な関係
になるように加工

線形回帰モデルを利用する場合

精度向上のコツ②: ラグ特徴量の生成

SIGNATE

- **時系列情報を扱う課題では、過去の実績値が特徴量として、有効に機能する場合があります。このような特徴量をラグ特徴量といいます。**



✓ 実績値に周期性があり、
おおむね一週間前と近い値になる傾向がある。

→ **一週間前の同曜日の実績値**が特徴量として有効に機能するのでは？

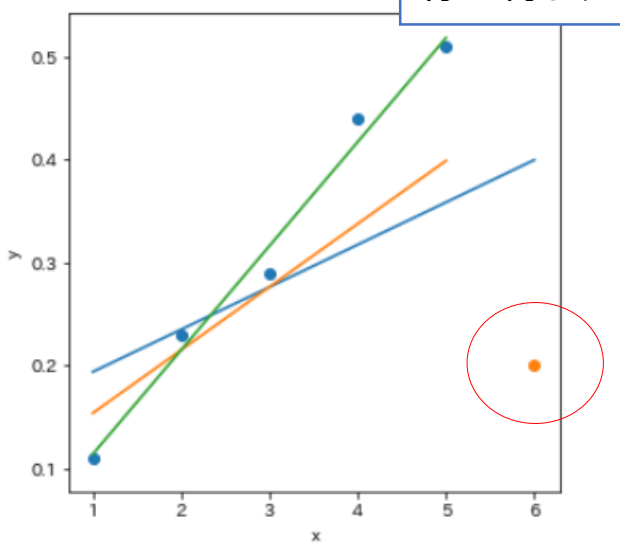
精度向上のコツ③: 外れ値の除外、量的データのカテゴリ化

SIGNATE

- 他にも「外れ値を除外する」ことや、
  「量的データをカテゴリ化する」といった工夫も考えられます。

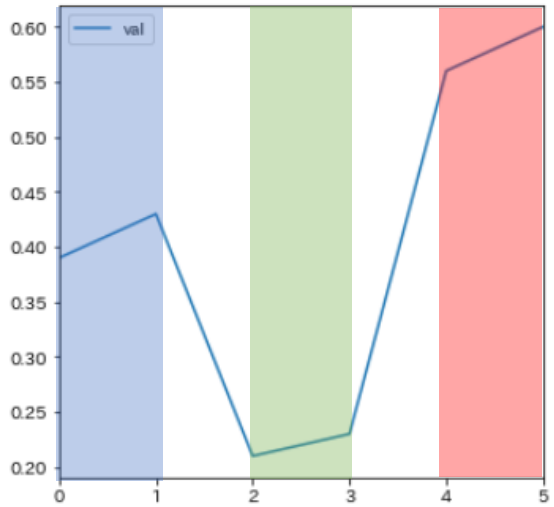
外れ値を外す、丸める

青：未加工
緑：if x>5 then x=5
緑：除去



外れ値？

量的データを質的データに変換



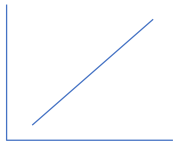
線形性が見えずらい変数についてはビニングに
よってカテゴリ変数化して扱うのも有効

精度向上のコツ④: モデルの選択

SIGNATE

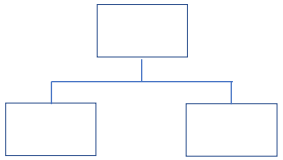
- **モデルの特性を踏まえて、課題に適したモデルを選択することも重要です。**

線形回帰モデル



- ✓ 単調性（xが増えればyも増）により、学習データ内の目的変数の値域を超えて予測できる
- ✓ 交互作用（AかつB等の複合的な条件）がモデル内に内包されていない為、特徴量を作る必要がある
- ✓ 相関が互いに強い説明変数を入れると係数が不安定（多重共線性）

決定木モデル



- ✓ あくまで学習データを基準に目的変数の値域が決まる為、学習データの値域を超えた予測はできない
- ✓ 交互作用がモデル内に内包されている為、交互作用に関する特徴量を作る必要がない
- ✓ 多重共線性を考慮しなくてよい

SIGNATE

Find the sign of changing times.