03

工数予測モデルを構築し、評価せよ

はじめに：本パートで学べる内容

本パート「**モデル開発（PoC）**」では、実際の企業の導入に求められる精度レベルを超えたAIモデルを開発するために必要なスキルを学ぶことができます。

(具体的には、以下の内容を学びます)

- データ前処理
- データ集計/可視化
- モデル開発

「初学者ガイド（サンプルコード付き）」をご参照ください

- 考えてみたが、行き詰ってしまった場合
- 自分なりの検討を深めるために、他にはどんな考え方があるのか知りたい場合

企業へのAI導入を進めるために必要なことを、
このパートを通じて理解しましょう！

演習の進め方

進め方とマイルストン

- 3週間をかけAIモデル開発及びその評価を行っていただきます。
- 本モデル開発はコンペ形式で行います。参加者はコンペ期間中何回でもモデルの評価結果を投稿することができます。
  ただし、1日あたりの投稿回数上限は5回で、毎日深夜0時にリセットされます。また投稿結果はランキングボードに反映されます。
- 自主学習だけではモデル開発が進まない受講者向けに「初学者ガイド（サンプルコード付き）」を用意しています。
  - サンプルコードは、内容を理解し、ご自身で改修していただくことで、学びを深めていただくことを意図したものです。
    特に、モデリングになかなか手が付けられないという場合には有効に使っていただけるものとなっておりますので、必要に応じてご活用ください。
  - 成績上位者には、コードの提出をお願いする可能性があります。

提出物

- 提出用ファイルを作成して、提出
  - ファイルの形式は、sample_submit.csvを参考に作成
  - 右の例のように、提出するCSVファイルには
    「ヘッダー」が含まれないようにしてください

5983行

index	正味作業時間	付帯作業時間
0	予測_0_0	予測_1_0
1	予測_0_1	予測_1_1
2	予測_0_2	予測_1_2
3	予測_0_3	予測_1_3
4	予測_0_4	予測_1_4
…	…	…
5982	予測_0_5982	予測_1_5982

← ヘッダー

演習03：モデリングコンペの実施

これまでの検討を踏まえ、工数予測のAI化が機械の非稼働時間／スタッフの残業削減、工数予測業務の工数削減、納期遅延リスクの低減、従業員満足度の向上に繋がると社内の合意が得られた。これを受け、生産管理課担当者による現行の計算式を用いた工数予測を、AIにより自動化する方向で検討が進められることとなった。

課　題

2020年8月～2021年2月における各機械（印刷機(2, 4, 6, 7, 8号機)、グルアー）で発生した正味作業時間と付帯作業時間を予測するモデルを構築し、その精度を評価せよ。

【留意点】
成績優秀者には、コンペ後集合日程での発表、コードの提出をお願いする可能性がございます。

本課題は実プロジェクトでのPoCにあたるものとして、コンペティション形式で、モデルを構築していただきます。
実際のAI導入プロジェクトにおいては、PoCで精度が確認でき、本格導入が決定した場合は、実業務環境に組み込み、
本番実装・運用することになりますので、単に予測精度を追求するだけでなく、実際に今後運用していくことが可能となるような
予測モデルの作成を心がけてください。

演習対象データ

ABC印刷の2018年10月～2021年2月までの基本データ、加工データ、作業実績データを対象にします。
今回の演習では、印刷機及びグルアーの工数（正味作業時間＋付帯作業時間）の予測を行います。以下が使用するデータの概要です。

データの種類と用途

- データは大きく分けて**学習用**、**評価用**、**提出用**の3種あります
- **学習用**は基本データ、加工データ、作業実績データの3つから構成されていて、前分析を行ったり、モデルに学習させるデータとして利用されるものです
- **評価用**は**学習用**と同様に基本データ、加工データ、作業実績データの3つから構成されていますが、学習したモデルに入力して予測値を出力するためのデータとなります。
  基本的には、基本データと加工データと、作業実績データの”号機名”と”作業日”(それ以外は予測には使えない)を用いて予測を行う形となります
- **提出用**はコンペティションサイトに投稿するためのフォーマットデータとして与えられるもので、応募用サンプルファイル(sample_submit.csv)となります
- 基本データ、加工データ、作業実績データの説明については、以下の表を参照してください。

基本/加工/作業実績データの詳細

種別	内容	ファイル名	説明
学習用	基本データ	base_train.csv	• 受注が決まった段階で記録されるデータ • “受注番号”についてユニーク
	加工データ	processing_train.csv	• 決まった受注に対する各工程・機械に関するデータ • “受注番号”×”号機名”(“号機コード”)についてユニーク
	作業実績データ	actual_train.csv	• “作業日”が2018年10月~2020年7月 • 日々の機械の稼働実績を記載したデータ • 正味作業時間と付帯作業時間が含まれる • “受注番号”×”号機名”(“号機コード”)についてユニーク
評価用	基本データ	base_test.csv	• 受注が決まった段階で記録されるデータ • “受注番号”についてユニーク
	加工データ	processing_test.csv	• 決まった受注に対して各工程・機械に関するデータ • “受注番号”×”号機名”(“号機コード”)についてユニーク
	作業実績データ	actual_test.csv	• “作業日”が2020年8月~2021年2月 • 日々の機械の稼働実績を記載したデータ • 正味作業時間と付帯作業時間は含まれない • “受注番号”×”号機名”(“号機コード”)についてユニーク
提出用	応募用サンプルファイル	sample_submit.csv	• コンペへ投稿する際のフォーマットデータ • このファイルと同じフォーマットになるように予測結果ファイルを作成

データ確認時の留意事項 (担当者から受領した資料の内容)

「作業実績データ」の定義

- 時間に関するデータの定義は以下の通り
- 合計時間は、2020年2月4日以降データ定義を変更しており注意が必要

入力項目名	定義	
	2020年2月3日まで	2020年2月4日以降
所要時間	開始してから終了するまでの時間	変更なし
作業時間	正味作業時間 (本生産が行われる時間)	変更なし
残業時間	非稼働時間 (生産に関与していない時間 (問い合わせ対応、予期せぬ停止等))	変更なし
合計時間	作業時間 + 残業時間	付帯作業時間 (本生産のための準備・後処理時間)

※2020年2月3日までの付帯作業時間は、「所要時間 - (作業時間+残業時間)」で算出可能
※データは手入力のため、上記定義の通りに入力されていない場合がある

主要なカラムの定義

対象ファイル	項目名	定義
加工データ	受注番号	営業が受注したオーダーごとに振り分けられる番号
	号機コード	生産に利用する機械のコード 機械毎に振られている
	号機名	生産を行う機械の名称
	工程名	受注した商品を製造するためのプロセスの名称
	数量1	受注に対して、生産する数量
	数量項目名1	数量1の単位名称
作業実績データ	作業日	機械を稼働させた日付

生産管理課
担当者

今回お渡しするデータは見方に注意が必要だったり、印刷業界もなかなか特殊で分かりにくい内容もあると思うので、ご留意いただきたい点をこちらの資料にまとめました。特に「作業実績データ」は、まとめてお渡ししているので分かりにくくなっていますが、工場で入力する項目名や定義の変更があったので、ご留意ください。申し訳ありませんが、すべてのデータカラムについて定義をまとめた資料がないため、ヒアリングでお話した内容とこの資料から読み解いていただければと思います。

補助教材① 前処理コード例 (1/7) テーブルの読み込み: pandas.read_csv()

まずは与えられたcsvファイルをPython上で読み込んで中身を確認しましょう

pandasというライブラリの read_csv という関数を使用した例

パス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

```
actual_train = pd.read_csv('./train/actual_train.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] JupyterLab (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

```
[1]: import pandas as pd

     actual_train = pd.read_csv('./train/actual_train.csv')

[2]: actual_train
```

[2]:

	作業実績番号	行番号	事業所コード	事業所名	部門コード	部門名	工程コード	工程名	号機コード	号機名	...	更新担当者コード	更新担当者名	更新端末コード	削除日	削除フラグ	削除担当者コード	削除担当者名	削除端末コード	取込キー	備考
0	D18000925	1	4.0	NaN	2010.0	NaN	1	プリプレス	1	プリプレス	...	4154.0	NaN	MHC476-1808	« NULL »	0.0	« NULL »	NaN	« NULL »	201810	NaN
1	E19053080	1	1.0	NaN	4040.0	NaN	41	検査	45	検査	...	177.0	NaN	MHC479-1809	« NULL »	0.0	« NULL »	NaN	« NULL »	« NULL »	NaN

補助教材① 前処理コード例 (2/7) 日付データへの変換: pandas.to_datetime ()

日付を日付データとして変換しましょう

Pandas というライブラリの to_datetime という関数を使用した例

“作業日” を日付データとして改めて定義しなおす

```
actual_train['作業日'] = pd.to_datetime(actual_train['作業日'])
```

- 特定の期間のデータを取り出す処理が書きやすくなる
- 年や月などのデータを取得しやすくなる

補助教材① 前処理コード例 (3/7) テーブルのフィルタリング処理①

関心のあるデータのみ抽出してみましょう

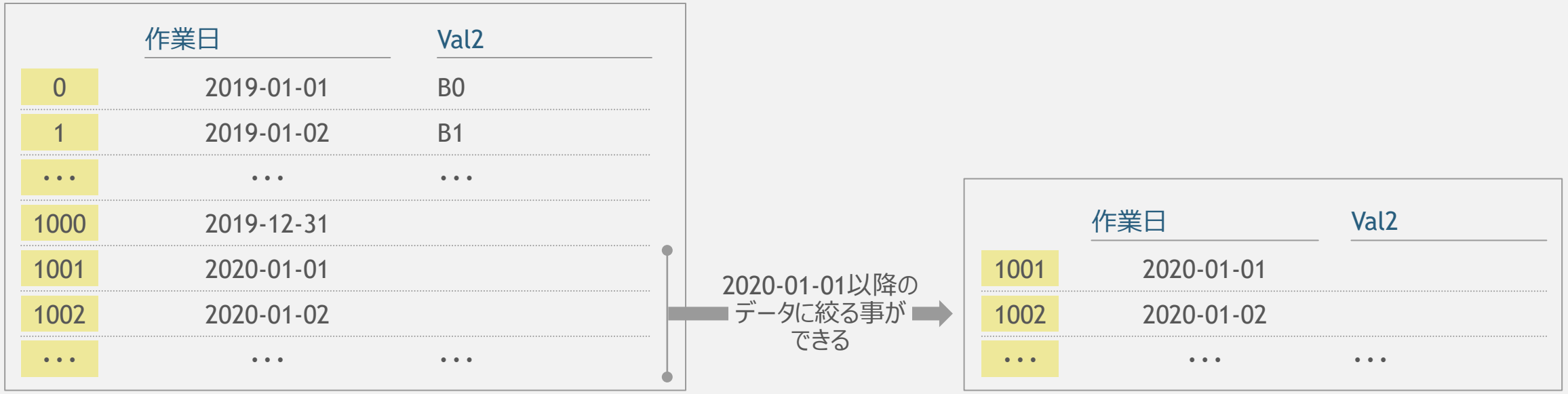
Pandas というライブラリを使用した例

- “[]”の中に代入する式によって自由に抽出の仕方を指定できます
- フィルタリングにより、異常値が入っているレコードの除去などに使えます (特定の値は除外するなど)

E,g) “作業日”を2020以降に絞りたい場合 (※ 事前に作業日の値を日付データに変換しておく必要があります)

```
filtered = actual_train[actual_train['作業日']>='2020-01-01']
```

イメージ図



補助教材① 前処理コード例 (4/7) テーブルの結合処理①: pandas.concat ()

2つ以上のテーブルを連結してみましょう

Pandas というライブラリの concat という関数を使用した例

[ ] が必要なので注意。良く忘れます

```
data_all = pd.concat([actual_train, actual_test])
```

	Val1	Val2
0	A0	B0
1	A1	B1
2	A2	B2

＋

	Val1	Val2
3	A3	B3
4	A4	B4
5	A5	B5

＝

	Val1	Val2
0	A0	B0
1	A1	B1
2	A2	B2
3	A3	B3
4	A4	B4
5	A5	B5

- モデルを学習させるときは一つのまとまったデータを学習データとして入力する
- まとめて一つにすることで分析しやすくなる場合がよくある

補助教材① 前処理コード例 (5/7) テーブルの結合処理②: pandas.merge ()

2つのテーブルを結合してみましょう

Pandas というライブラリの merge という関数を使用した例

結合するキーを指定

```
merged = pd.merge(actual_processing, actual_base, on = '受注番号')
```

```
merged = pd.merge(actual_train, actual_processing, on = ['受注番号','号機名'])
```

複数指定する場合は
”[]” で囲って列挙する

“Val1” をキーとして結合

	Val1	Val2
0	A0	B0
1	A1	B1
2	A2	B2

＋

	Val1	Val3
3	A0	C3
4	A4	C4
5	A1	C5

＝

	Val1	Val2	Val3
0	A0	B0	C3
1	A1	B1	C5

- モデルを学習させるときは一つのまとまったデータを学習データとして入力する
- まとめて一つにすることで分析しやすくなる場合がよくある

補助教材① 前処理コード例 (6/7) グループ単位の集計: pandas.DataFrame.groupby ()

データをグループごとに集計してみましょう

Pandas というライブラリにおける groupbyメソッド を使用した例

作業日ごとにデータをグループ分け

```
grouped = actual_train.groupby('作業日').apply(lambda x:x['所要時間'].sum())
```

所要時間の合計を求める

.groupby ('Val1').apply(lambda x:x ['Val2'].sum())

- 様々な粒度でデータを分析できる
- 特徴量加工の幅が広がり、利用できる

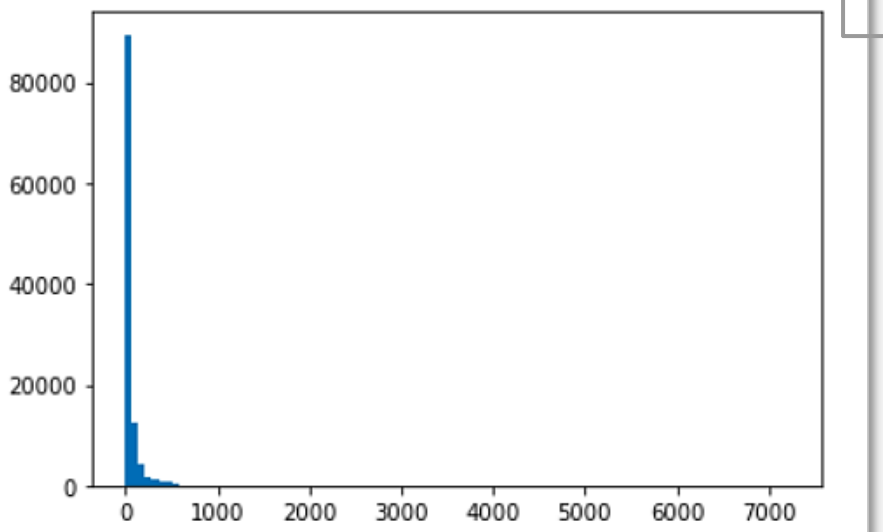
補助教材① 前処理コード例 (7/7) データの可視化 (ヒストグラム): matplotlib.pyplot.hist ()

データのヒストグラムを描画してみましょう

matplotlib というライブラリのpyplotのhist関数を使用した例

```
[12]: from matplotlib import pyplot as plt
      %matplotlib inline

[16]: plt.hist(actual_train['所要時間'],bins=100)
      plt.show()
```

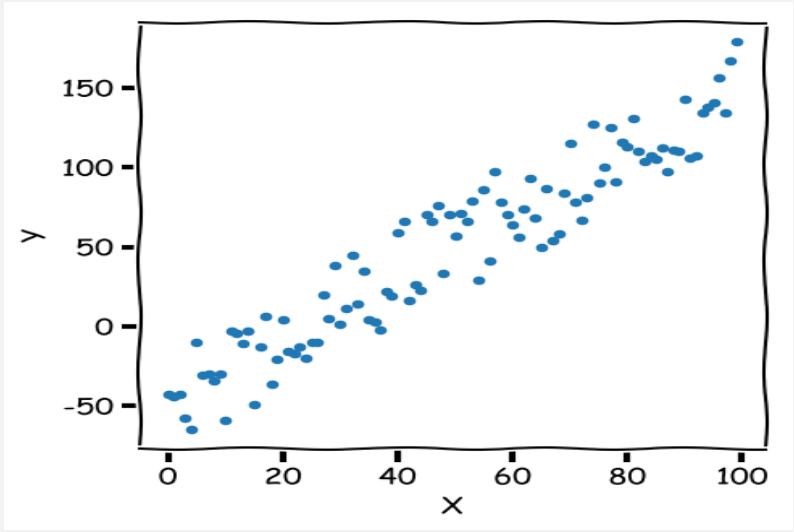


binsでビンの数を指定して粒度を調整できます

- 目的変数などの数量データの分布を確認するときに使用できる

補助教材② 特徴量生成・モデリングの方針(1/3)
精度向上のコツ 1.特徴量の加工

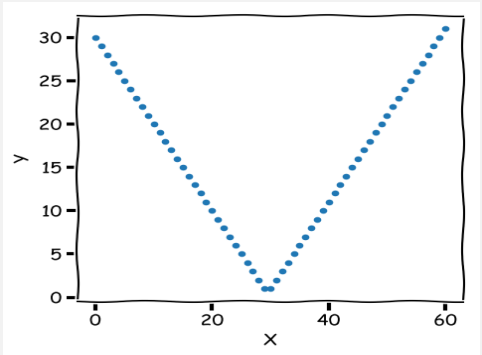
予測に有効な特徴量を作成・見つけることが重要です。
（特徴量とは、分析対象データの中の、予測の手掛かりとなる変数のこと）
特に、回帰問題の場合には線形性がある特徴量が重要となります。

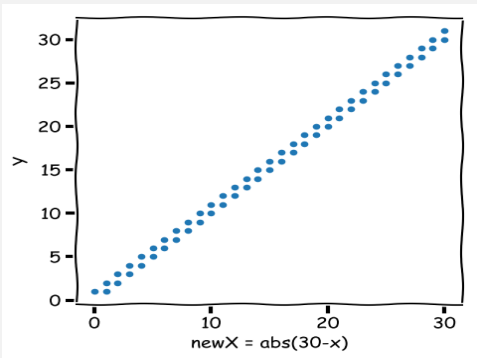


xとyの相関が高そう
→予測に寄与する変数である可能性

※PythonではmatplotlibなどのライブラリでMAE分析可能

応用レベル



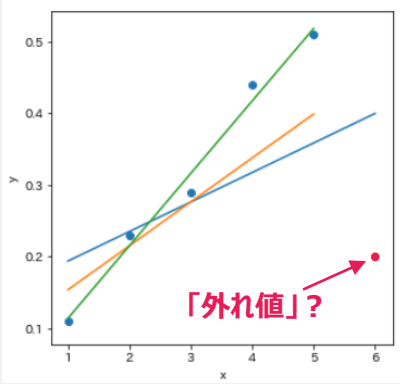


線形回帰モデルを利用する場合、
ｘとyが線形な関係になるように加工

補助教材② 特徴量生成・モデリングの方針(2/3)
精度向上のコツ 2.外れ値の除外、量的データのカテゴリ化

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」といった工夫で予測精度が改善することがあります。
手元のデータで検証してみましょう。

外れ値を外す、丸める

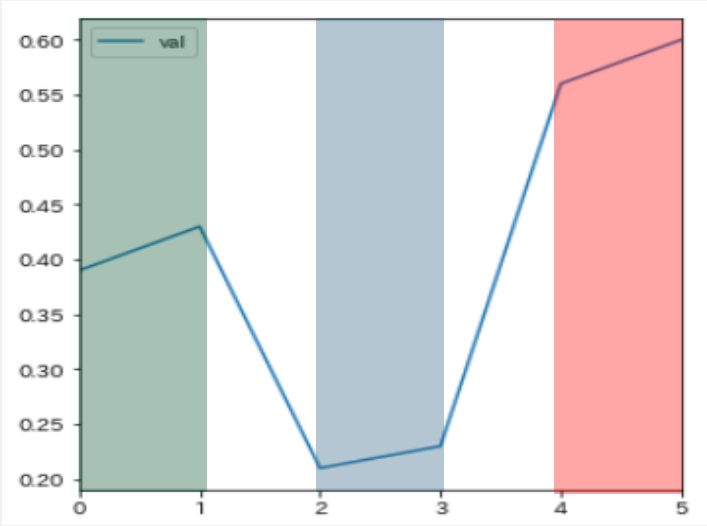


青線：
外れ値に手を加えなかった場合の回帰直線
橙線：
xが最大で5となるよう丸めた場合の回帰直線
緑線：
外れ値を除去した場合の回帰直線

上図の**青線**のように、**「外れ値」**の存在に影響を受けてモデルの予測傾向が大きく変容してしまうことがある。そんなときは、
①「外れ値」を特定の値域に丸める (橙線)
②「外れ値」を除外する (緑線)
といった処理を加えることで、(**「外れ値」**以外の)より一般的なデータに適合したモデルを構築することができる。
※ ただし、データの性質や予測評価指標によっては**「外れ値」**をあえてそのまま残すほうが望ましい場合もあることに注意

応用レベル

量的データを質的データに変換



線形性が見えづ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

補助教材② 特徴量生成・モデリングの方針(3/3)
精度向上のコツ 3.モデルの選択

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ため、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)

主な用途

例❶：体重の予測(単回帰)
- 身長から体重を予測する

例❷：店舗売上の予測(重回帰)
- 来客数、駅からの距離、駅乗降者数などから店舗の売上を予測する

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ため、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ため、交互作用に関する特徴量を作らなくてもよい場合がある
- ✓ 多重共線性を考慮しなくてよい

主な用途

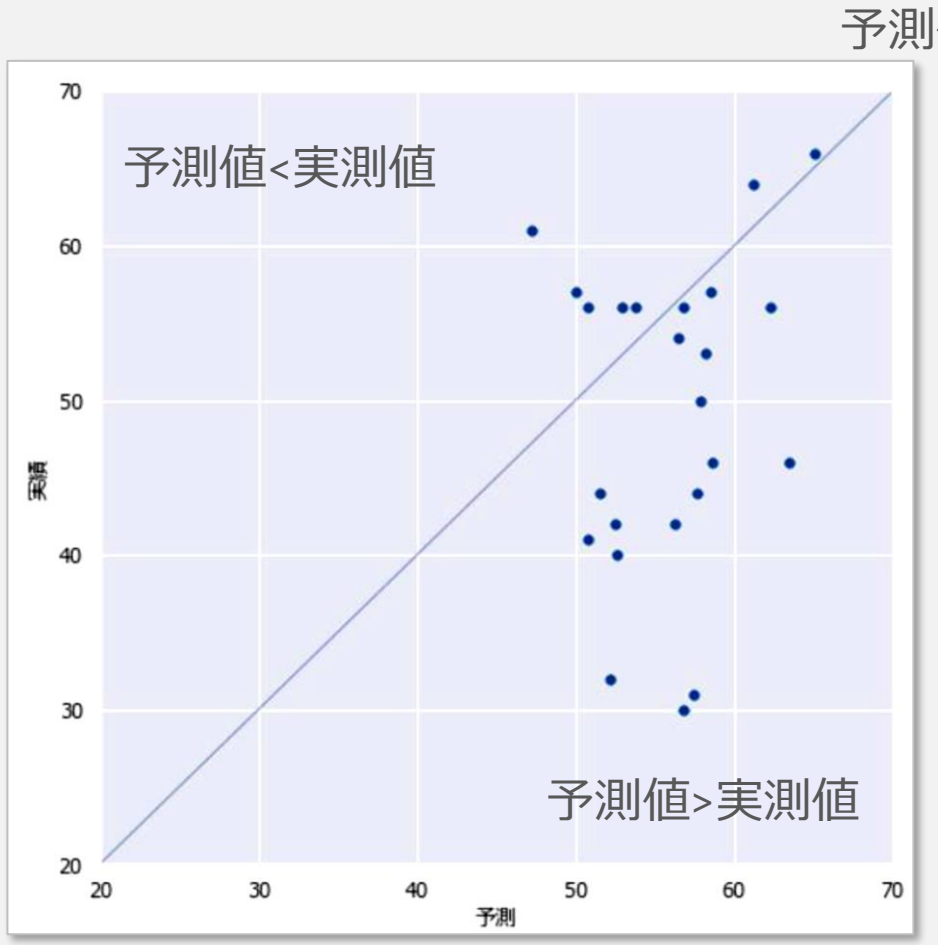
例❶：顧客の行動予測
- 過去のデータから顧客が次に商品Aを購入するか否か予測する

例❷：事象の発生原因の特定
- あるサービスの解約につながる要因・要素を探索する

補助教材③ モデル特性の把握 (1/4) 予測値と実測値の比較

X軸に予測値、y軸に実測値を描画、上振れ/下振れ傾向を把握

- これは最低限必要な基本確認となります
- 必要に応じ、特定商品カテゴリごと等、より細かい粒度で傾向を確認することもあります



※ Pythonではmatplotlibやpandasなどのライブラリで分析可能

補助教材③ モデル特性の把握(2/4)
MAE (Mean Absolute Error)

- 定義

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_{obs,i} - y_{pred,i}\right|$$

N　　：データ数
$y_{obs,i}$　：i番目の実測値
$y_{pred,i}$　：i番目の予測値

- 特徴
  - 実測値と予測値の差の**絶対値**の平均をとっている
  - 直感的に分かりやすい
- 使用場面
  - 直感的なわかりやすさを求める場合
  - どのような実測値に対しても等しく当てられていることを評価したい場合
  - 一部の予測が大きくズレることを比較的許容できる場合
- 実装
  - Pythonではscikit-learnやnumpyで実装できる

補助教材③ モデル特性の把握(3/4)
RMSE (Root Mean Squared Error)

- 定義

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_{obs,i} - y_{pred,i}\right)^2}$$

N　　：データ数
$y_{obs,i}$　：i番目の実測値
$y_{pred,i}$　：i番目の予測値

- 特徴
  - 実測値と予測値の差の**2乗**のルートをとっている
  - **大きなズレ(実測値と予測値に極端な開きがあるもの)があった場合、誤差が極端に大きくなる**
- 使用場面
  - 回帰問題において最も一般的な指標であるため、多くの場合に用いられる
  - 一部の予測が大きくズレることを許容したくない場合
- 実装
  - Pythonではscikit-learnやnumpyで実装できる

補助教材③ モデル特性の把握(4/4)
RMSLE (Root Mean Squared Logarithmic Error)

- 定義

※定義式内の「+1」はyの値が0の場合に計算不可能となることを防ぐため

$$RMSLE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\log\left(y_{pred,i}+1\right) - \log\left(y_{obs,i}+1\right)\right)^2}$$

N　　：データ数
$y_{obs,i}$　：i番目の実測値
$y_{pred,i}$　：i番目の予測値

- 特徴
  - 実測値と予測値の「**対数の差の2乗**」の平均のルートをとっている
  - 予測値が実測値よりも**小さく振れた場合に誤差が大きくなる**
  - log A−log$B$=log A/B となるため、RMSLEは、データの**『スケール』そのものが影響することはなく、『2つの値の割合』を考慮する**
- 使用場面
  - 実測値の範囲が極端に広い場合(スケールの影響を受けないようにするため)
  - 実測値よりも低い値を予測することを回避したい場合
- 実装
  - Pythonではscikit-learnやnumpyで実装できる