




GCI 2023 SUMMER

Pythonによるデータ加工処理の基礎（Pandas）



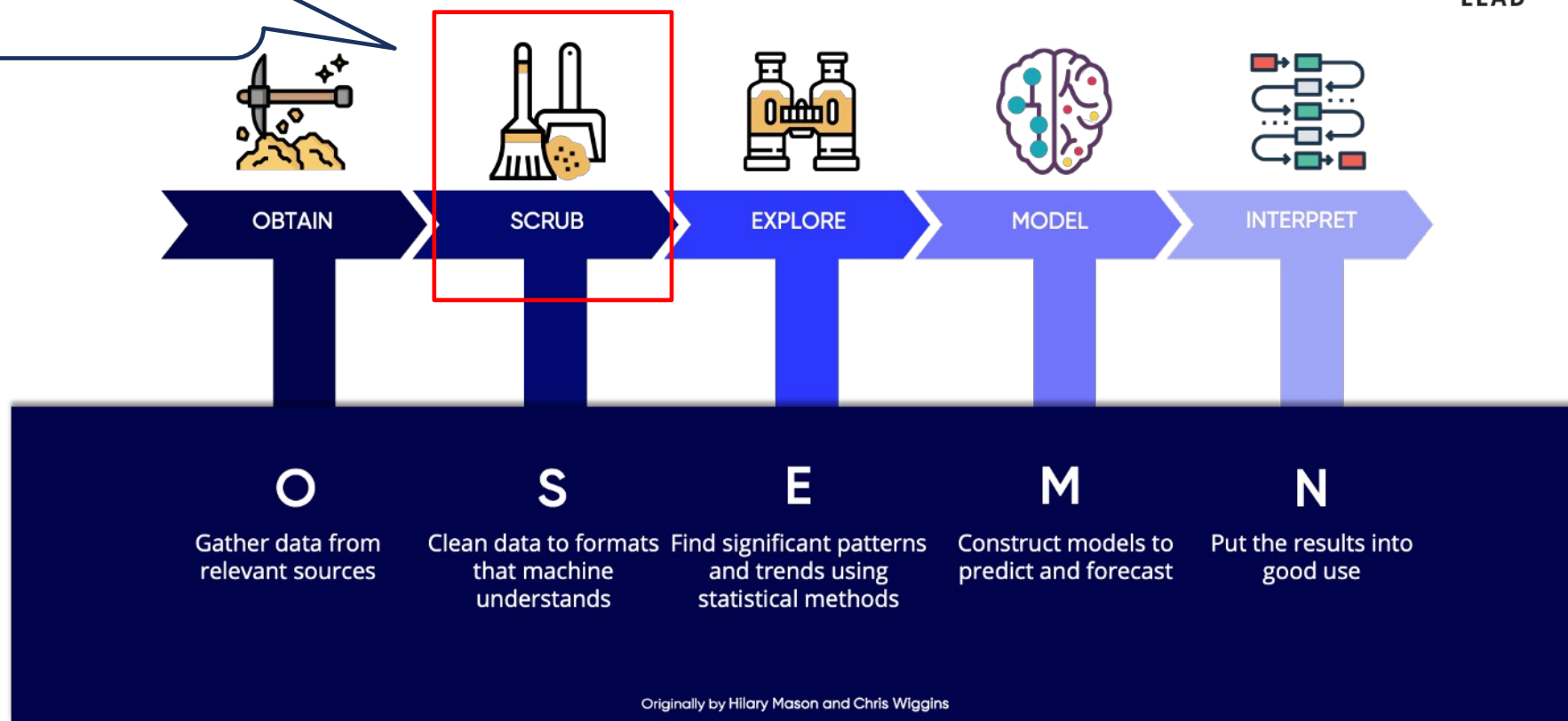
自己紹介

- 東京大学工学系研究科マテリアル工学専攻修士 2 年
- 第 6 期 GCI (2019) 卒業生
- 松尾研では以下の 2 つの共同研究に携わっています
 - 化学プラントにおける異常検知
 - 大規模世界モデル

データサイエンスプロジェクトのフロー

この講義で扱う

Data Science Process



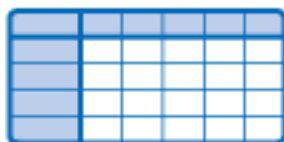
データ分析の8割はモデリング以前の作業

- データに異常値や欠損値がある
- 不要なデータが含まれている
- データの形式がバラバラ
- 扱いにくいデータ形式

データの種類

この講義で扱う

構造化データ



固定長ファイル

Excel / CSV

RDB内のデータ

二次元の表形式など
値が数値・記号で
1テーブルに整理されている

非構造化データ



テキスト

音声

画像

動画

センサーログ

半構造化データ

XML

JSON

html

表形式ではないが
データに規則性がある

データに規則性がなく
表形式に変換できない

Pandas とは

- Pythonを数表として扱うことを提供するライブラリ
- データフレーム形式で様々なデータを加工することができる
- 簡単に言うとExcelを操作するように扱える

データ分析コンペでの Pandas の使用例

2019年度GCI受講時に実際に提出したものの一部

```
[ ] test = pd.read_csv("googleplay-test.csv")
train = pd.read_csv("googleplay-train.csv")
print(train.shape)
print(test.shape)
```

データの読み込み

```
[ ] train = reduce_mem_usage(train)
test = reduce_mem_usage(test)
print(f'Nan values: {train.isnull().values.any()}')
```

欠損値の確認

```
[ ] # 平均値代入
def fill_genres(train):
    drop_features = ["App", "Size", "Installs", "Type", "Price", "Content Rating", "Genres", "genres_init", "Rating"]
    feats = [f for f in train.columns if f not in drop_features]
    train[feats] = train[feats].fillna(train[feats].mean())
    return train
```

欠損値の補完

```
[ ] # 回帰代入
import sklearn.linear_model as lm
def fill_size(train):
    reg = lm.LinearRegression()
    indexer = train['size_log'].isnull()
    reg.fit(train.loc[~indexer, ['installs_num_log', 'review_log', 'reviews_installs']], train.loc[~indexer, 'size_log'])
    predicted = reg.predict(train.loc[indexer, ['installs_num_log', 'review_log', 'reviews_installs']])
    train.loc[indexer, 'size_log'] = predicted
    return train
```

データの確認

```
[ ] test.head()
```

	App	Reviews	Size	Installs	Type	Price	Content Rating	Genres
0	FP Markets	1	2.0M	100+	Free	0	Everyone	Finance
1	Diabetes, Blood Pressure, Health Tracker App	309	5.9M	10,000+	Free	0	Everyone	Medical
2	Frontback - Social Photos	19446	Varies with device	1,000,000+	Free	0	Mature 17+	Social
3	e-DN - den digitala tidningen från Dagens Nyheter	160	32M	50,000+	Free	0	Everyone 10+	News & Magazines
4	Roulette Advisor LITE	41	1.4M	5,000+	Free	0	Everyone	Casino

データの整形

```
def make_size_list(train):
    values = train.loc[:, "Size"].values
    size_list = list()
    for i in range(len(values)):
        size_list.append([values[i][-1]])
    dummy_df = pd.get_dummies(pd.DataFrame(size_list, columns=["size_list"]), drop_first=True, dummy_na=True)
    train = pd.merge(train, dummy_df, left_index=True, right_index=True)
    print("make_size_list")
    return train

# 正規分布的にするためlogをとる
def make_size_log(train):
    values = train.loc[:, "Size"].values
    size_log = list()
    for i in range(len(values)):
        value = values[i]
        if value[-1] == 'M':
            size_log.append([np.log10(float(value[0:-1])) + 6])
        elif value[-1] == 'k':
            size_log.append([np.log10(float(value[0:-1])) + 3])
```

```
# genres内での相対的な値
def make_genres_mean(train):
    features = ["installs_num_log", "price_num", "size_log", "review_log", "reviews_rate", "reviews_installs", "name_len", "genres_num", "name_upper", "name_alnum", "name_decimal"]
    for x in features:
        train[f'mean_{x}'] = train.groupby('genres_init')[x].transform(sum)/train['genres_count']
        train[f'x_rate'] = (train[x] / train[f'mean_{x}'] + 1).replace([np.inf, -np.inf, np.nan], 0)
    print("make_genres_mean")
    return train
```

```
[ ] def featurizing_train(train):
    train = make_size_list(train)
    train = make_size_log(train)
    train = make_price_num(train)
    train = make_rating_dummies(train)
    train = make_genres_num(train)
    train = make_genres_init(train)
    train = make_review_log(train)
    train = make_installs_num(train)
    train = make_installs_num_log(train)
    train = make_genres_init_dummies(train)
    train = make_reviews_rate(train)
    train = make_content_rating_dummies(train)
    train = make_name_len(train)
    train = make_reviews_installs(train)
    train = make_name_words(train)
    train = make_name_upper(train)
    train = make_genres_count(train)
    train = make_genres_mean(train)
    return train
```

以降、データ分析やモデルの構築

Pandas と Numpy の使い分け

Pandasの強み

- 数値データや時系列データ，文字列を処理する関数が揃っている
- データ集約などの統計処理に強い
- 時系列データに強い
- csv, excel, jsonなどの様々なファイル形態を読み書きすることができる

Pandasの弱み

- 速度が遅い
- メモリ消費が激しい

Pandas のデータ型

この形式を主に使用する

	apple
0	2
1	3
2	2

Series

	apple	orange
0	2	4
1	3	1
2	2	6

DataFrame

Excelのようなイメージ

Columns (例)

1列目 2列目 ... 5列目

	datetime	y	week	soldout	name
1行目	0 2013-11-18	90	月	0	厚切りイカフライ
2行目	1 2013-11-19	101	火	1	手作りヒレカツ
...	2 2013-11-20	118	水	0	白身魚唐揚げ野菜あん
...	3 2013-11-21	120	木	1	若鶏ピリ辛焼
5行目	4 2013-11-22	130	金	1	ビッグメンチカツ

index (行)

より詳しく知りたい方へ

- PandasをはじめNumpy, Matplotlibなどの基本を網羅
- 色々な参考書がありますが、実際に手を動かして学ぶことが重要です



演習

- 実際にコードを動かしてみましょう
- PandasではDataFrame形式を扱う関数がたくさんあります
- もちろん全て覚える必要はなく、なんとなくどのようなことができるのかを理解できるだけで十分です

階層型インデックス

	店舗名	商品名	売上
0	A店	商品1	100
1	A店	商品2	200
2	B店	商品1	150
3	B店	商品2	250



		店舗名	商品名	売上
店舗名	商品名			
A店	商品1	A店	商品1	100
	商品2	A店	商品2	200
B店	商品1	B店	商品1	150
	商品2	B店	商品2	250

データそのものは変わらないが、視認性が向上し、フィルタリングや集約の計算がしやすくなる

階層型インデックス

		Osaka	Tokyo	Osaka
		Red	Blue	Red
a	1	0	1	2
	2	3	4	5
b	2	6	7	8

level0 level1

level0

level1 level-1

Stack

level1

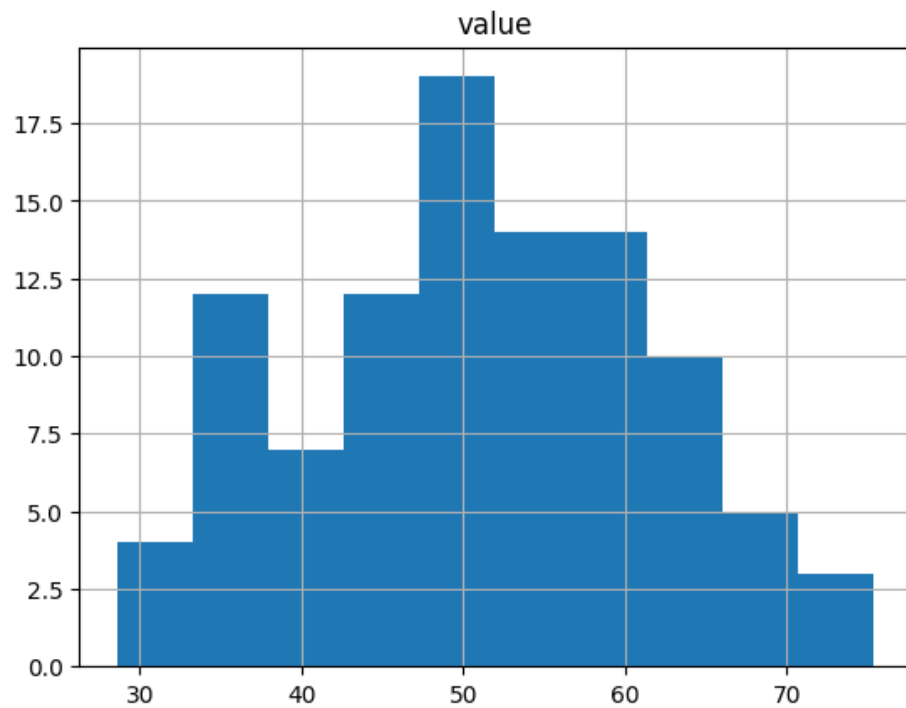
			Osaka	Tokyo
a	1	Blue	0	1.0
		Red	2	NaN
	2	Blue	3	4.0
		Red	5	NaN
b	2	Blue	6	7.0
		Red	8	NaN

level0

			Blue	Red
a	1	Osaka	NaN	0
		Tokyo	1.0	2
	2	Osaka	NaN	3
		Tokyo	4.0	5
b	2	Osaka	NaN	6
		Tokyo	7.0	8

ビン分割

```
data = DataFrame({"value": np.random.normal(50, 10, 100)})
```



cut() : データを等間隔で分割

```
pd.cut(data["value"], bins=5, labels=["1", "2", "3", "4", "5"])
```

1	2	3	4	5
16	19	33	24	8

qcut() : データを等個数で分割

```
pd.qcut(data["value"], q=5, labels=["1", "2", "3", "4", "5"])
```

1	2	3	4	5
20	20	20	20	20

どちらの関数を使用するかは、データの性質に応じて決定する必要がある