データサイエンス実習
最終課題 発表

三宅研 M2 高林 秀

1. 問題設定

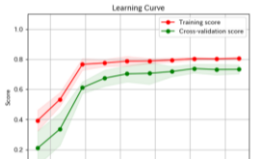
- 選択した問題：選択肢① プロ野球Freakから、ホームゲームの観客数を予測する回帰モデルを作る

【問題条件】

- 予測するチーム：横浜DeNAベイスターズ
- 目的変数：ホームゲームの観客数
- 説明変数
  - 試合日程（年,月,日）
  - 勝敗
  - スコア
  - 対戦相手
  - 先発投手

4. 使用したモデル

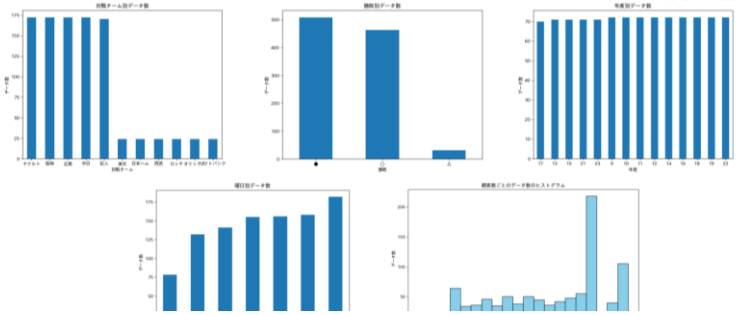
- 決定木：`DecisionTreeRegressor`
- 標準化と交差検証により適切なハイパーパラメータの探索を行い、精度向上を狙った。
  - 交差検証：`GridSearchCV(…, cv=10, scoring='r2', n_jobs=-1)`
  - 標準化　：`StandardScaler`
- 学習後に、学習過程を可視化して確認することで、過学習の有無を確認した。




```
grid_params = {
    'regressor__max_depth': [2, 4, 6, 8, 10, 12],
    'regressor__min_samples_split': [2, 5, 10],
    'regressor__min_samples_leaf': [1, 2, 4, 6, 8, 10, 12],
}
```

2. データの概要：データ概要

- 今回使用するデータセットのデータ数については下記のような分布になっている。



内容

1. 問題設定
2. データの概要
3. 整形と前処理
4. 使用したモデル
5. モデルの分析
6. まとめ

1. 問題設定

- 選択した問題：選択肢① プロ野球Freakから、ホームゲームの観客数を予測する回帰モデルを作る

【問題条件】

- 予測するチーム：**横浜DeNAベイスターズ**
- 目的変数：ホームゲームの観客数
- 説明変数
  - 試合日程 (年,月,日)
  - 勝敗
  - スコア
  - 対戦相手
  - 先発投手

日付	観客数	勝敗	スコア	対戦相手	先発投手	試合時間	球場
4月7日(火)	20,168 人	●	1－5	巨人	寺原	3:45	横浜
4月8日(水)	16,361 人	●	1－12	巨人	工藤	3:10	横浜
4月9日(木)	16,691 人	●	2－9	巨人	ウォーランド	3:15	横浜
4月10日(金)	12,791 人	○	9－1	ヤクルト	三浦	2:46	横浜
4月11日(土)	17,817 人	●	0－3	ヤクルト	グリン	3:23	横浜
4月12日(日)	17,972 人	○	5－3	ヤクルト	小林	3:13	横浜
4月17日(金)	12,250 人	●	1－5	阪神	三浦	2:56	横浜
4月18日(土)	25,773 人	●	4－9	阪神	グリン	4:02	横浜
4月19日(日)	24,491 人	○	4－2	阪神	小林	3:20	横浜
4月21日(火)		－	中止	広島			横浜
4月22日(水)	16,948 人	●	4－5	広島	寺原	3:19	横浜
4月23日(木)	16,772 人	●	0－2	広島	ウォーランド	2:43	横浜
5月2日(土)	21,080 人	○	6－1	中日	三浦	2:36	横浜
5月3日(日)	23,920 人	●	0－2	中日	グリン	3:14	横浜
5月4日(月)	22,529 人	●	2－4	中日	小林	3:18	横浜

2. データの概要：データの取り込み

- Pandasの`read_html`関数により、Web上からダウンロードした。
- 年ごとにURLが異なるため、繰り返し処理で年度を変えて取得、その後一元化した。

```python
# インポートするデータの条件指定
data_year_range: const[tuple] = (9, 23)
team: const[str] = "baystars"
base_url: const[str] = "https://baseball-freak.com/audience/"
```

✓ 0.0s

```python
# 指定範囲年のデータのインポート
datas: list[pd.DataFrame] = []
for year in tqdm(range(data_year_range[0], data_year_range[1]+1), desc="Fetching Data...."):
    if(year == 20):
        continue
    else:
        url = f"{base_url}{year:02}/{team}.html"
        dfs = pd.read_html(url)
        datas.append(dfs[2])
else:
    print("Done")
```

✓ 4.2s

```
Fetching Data....: 100%|██████████| 15/15 [00:04<00:00,  3.56it/s]
```

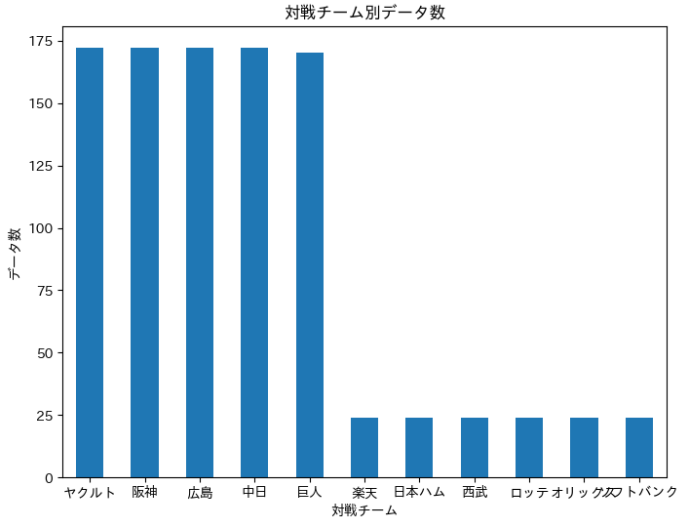
```python
# 年数のカラムを追加し、DataFrameを１つにまとめる。
for df, year in zip(datas,  np.arange(data_year_range[0], data_year_range[1]+1).tolist()):
    df['年'] = year

dataset = pd.concat(datas, ignore_index=True)
dataset
```

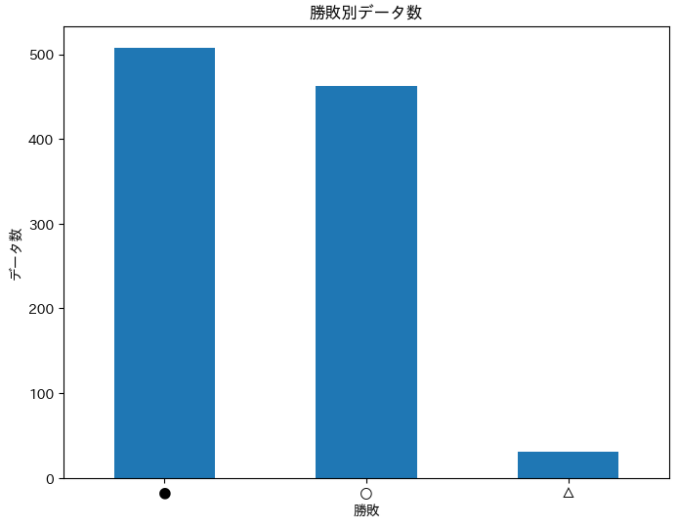
✓ 0.0s

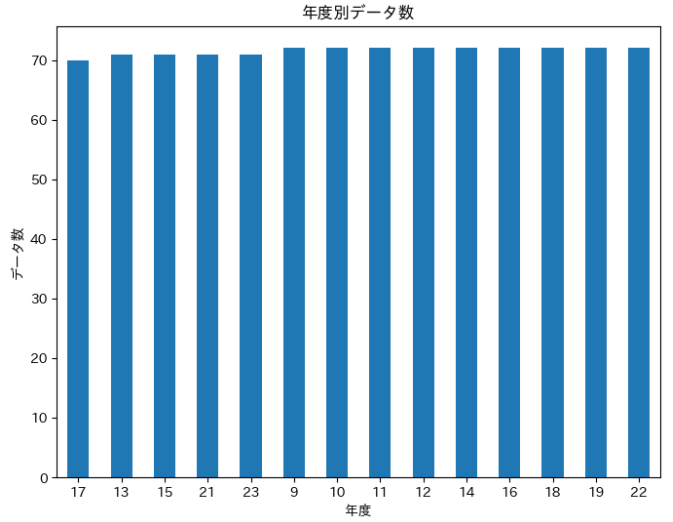
	日付	観客数	勝敗	スコア	対戦相手	先発投手	試合時間	球場	日付.1	年
0	4月7日(火)	20,168 人	●	1 - 5	巨人	寺原	3:45	横浜	4月7日(火)	9
1	4月8日(水)	16,361 人	●	1 - 12	巨人	工藤	3:10	横浜	4月8日(水)	9
2	4月9日(木)	16,691 人	●	2 - 9	巨人	ウォーランド	3:15	横浜	4月9日(木)	9

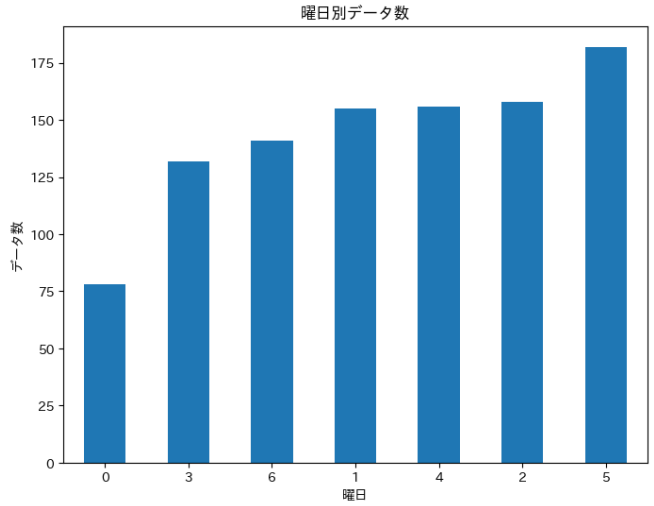
2. データの概要：データ概要

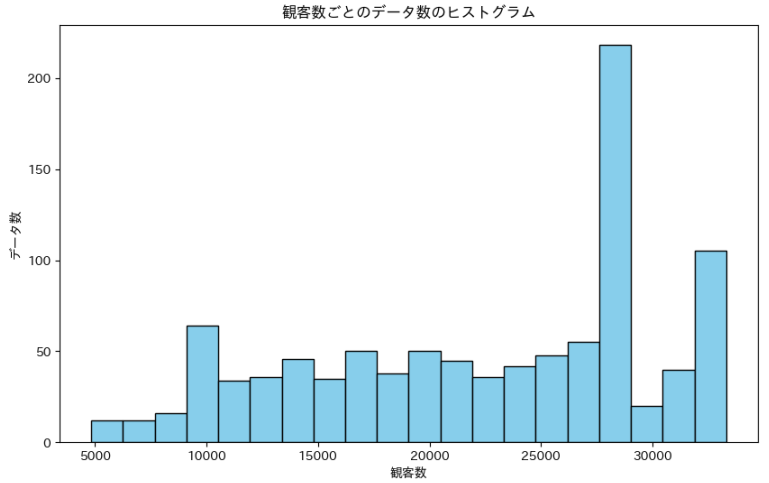
- 今回使用するデータセットのデータ数については下記のような分布になっている。









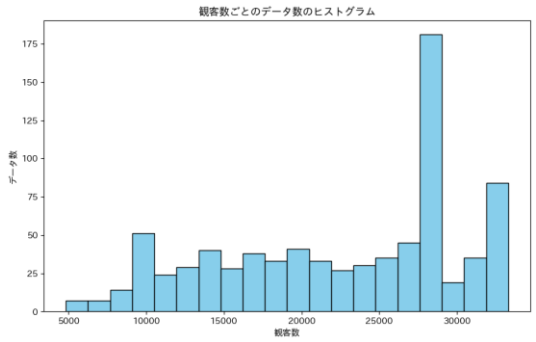


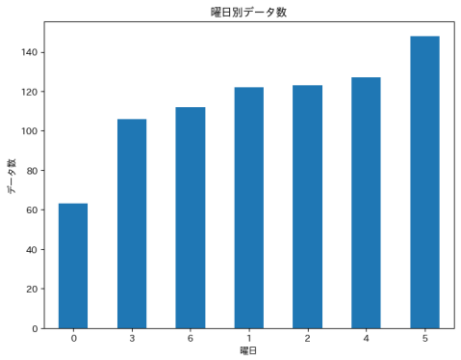
2. データの概要：訓練・テスト別

- 訓練用データとテストデータの分割割合は、8:2として分割した。

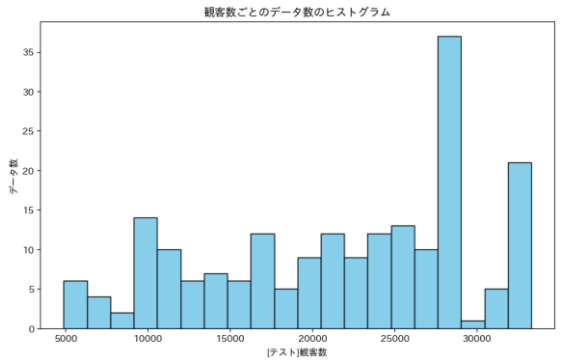
```
TrainData Size (801, 95)
TestData Size (201, 95)
```

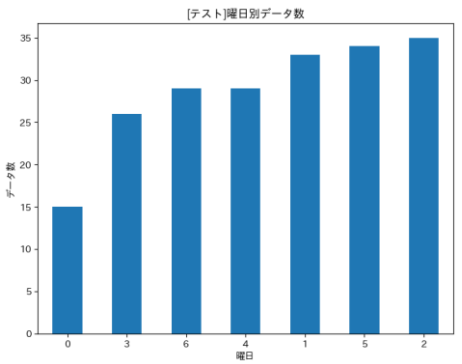
訓練用データ





テスト用データ





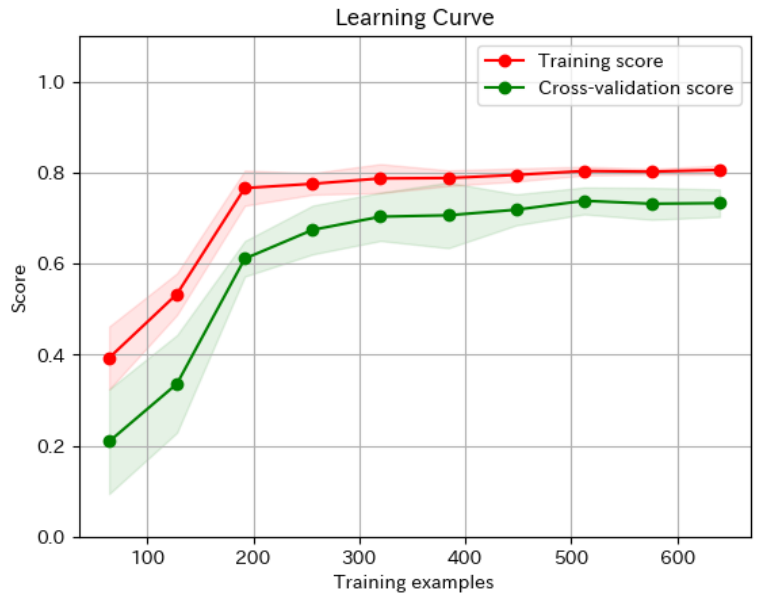
3. 整形と前処理

- 中止ゲームについては、そのままでは欠損値として自動処理できなかったので、以下の修正を行った。
  - 勝敗　：'-' という値を、欠損値として扱い、`np.nan`代入
  - スコア：'中止'という値を、〃
- 観客数のデータから、単位（人）を除去
- スコアを自軍スコアと相手スコアに分ける
- 勝ち（〇）＝ true, 負け（●） ＝ falseとしてエンコーディングする
- 対戦相手と先発投手をone-hotエンコーディングする
- 年、月、日、曜日、四半期のカラムを追加

	観客数	自軍スコア	相手スコア	勝利	対戦相手_オリックス	対戦相手_ソフトバンク	対戦相手_ヤクルト	対戦相手_ロッテ	対戦相手_中日	対戦相手_巨人	...	先発投手_阿斗里	先発投手_須田	先発投手_飯塚	先発投手_高崎	先発投手_高橋	年	月	日	曜日	四半期
0	20168	1	5	False	False	False	False	False	False	True	...	False	False	False	False	False	2009	4	7	1	2
1	16361	1	12	False	False	False	False	False	False	True	...	False	False	False	False	False	2009	4	8	2	2
2	16691	2	9	False	False	False	False	False	False	True	...	False	False	False	False	False	2009	4	9	3	2
3	12791	9	1	True	False	False	True	False	False	False	...	False	False	False	False	False	2009	4	10	4	2
4	17817	0	3	False	False	False	True	False	False	False	...	False	False	False	False	False	2009	4	11	5	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1066	33271	1	0	True	False	False	False	False	False	True	...	False	False	False	False	False	2022	9	25	6	3
1067	33262	1	0	True	False	False	False	False	False	True	...	False	False	False	False	False	2022	9	26	0	3
1068	33254	3	11	False	False	False	True	False	False	False	...	False	False	False	False	False	2022	9	27	1	3
1069	33267	5	3	True	False	False	False	False	False	False	...	False	False	False	False	False	2022	9	29	3	3
1070	32782	4	0	True	False	False	False	False	True	False	...	False	False	False	False	False	2022	10	1	5	4

4. 使用したモデル

- 決定木：`DecisionTreeRegressor`
- 標準化と交差検証により適切なハイパーパラメータの探索を行い、精度向上を狙った。
  - 交差検証：`GridSearchCV(…, cv=10, scoring='r2', n_jobs=-1)`
  - 標準化　：`StandardScaler`
- 学習後に、学習過程を可視化して確認することで、過学習の有無を確認した。




```python
grid_params = {
    'regressor__max_depth': [2, 4, 6, 8, 10, 12],
    'regressor__min_samples_split': [2, 5, 10],
    'regressor__min_samples_leaf': [1, 2, 4, 6, 8, 10, 12],
}
```

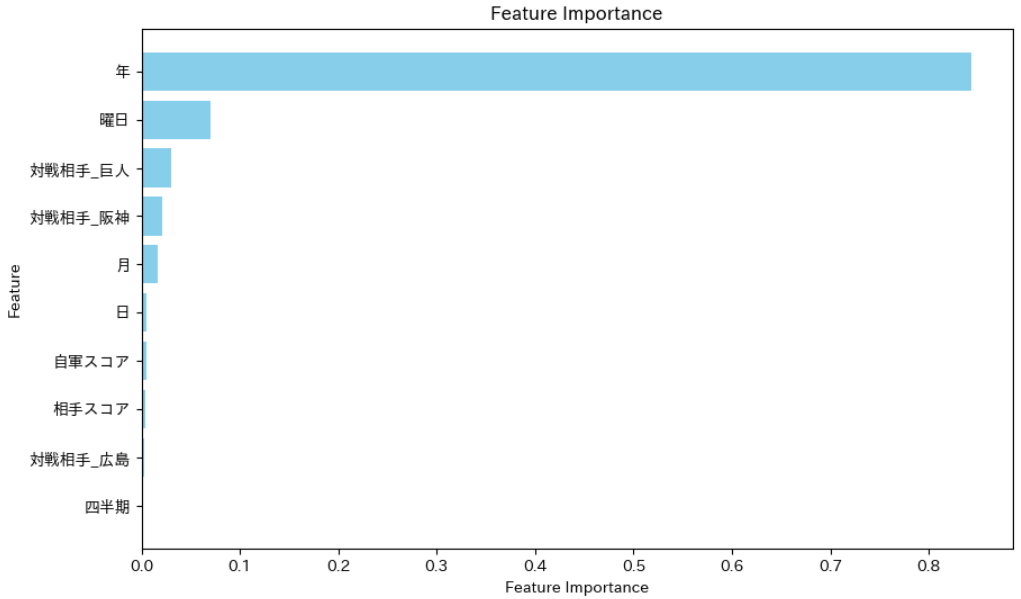
5. モデルの分析

- 今回のモデルのパラメータは以下のようになった。

```
model param: {
'regressor__max_depth': 6,
'regressor__min_samples_leaf': 8,
'regressor__min_samples_split’: 2
}
```

- また、 R2スコアは以下の通り
  - 訓練時：0.74
  - テスト時：0.77

- 特徴重要度をプロットした結果
- コロナの関係か、年数がかなり影響している
- 次いで、曜日の重要度が高い。



6. まとめ

- 横浜ベイスターズの2009年～2023年のホームゲームの観客数を予測する回帰モデルを作成した。
- 決定木ベースのモデルを採用
- 標準化と交差検証、グリッドサーチによるハイパーパラメータ探索で可能な限り精度向上に努めた。
- 結果、訓練時よりも、テスト時のR2スコアの方を高くすることができ、一定の精度を確保できた。
- 特徴重要度の計算から、年数と曜日が結果に影響を与えていると推測。

【感想】

- データのサンプリングを行えば、もう少し精度を出せたかも
  - チーム別 × 観客数をプロットし、偏りを確認すべき？
- 生データ整形に時間がかかり、もっと適切な精度向上のための前処理を組めたかもしれない。。。