YSAL 1st team project

PREDICT NEXTPITCH

일시 2023년 10월 9일

발제 Ysal 야구 2팀 (김수환, 송화윤, 정현지)

CONTENTS

01

Intro

Strategic Pitch Location

The Role of Two-Pitch Sequences in Pitching Success

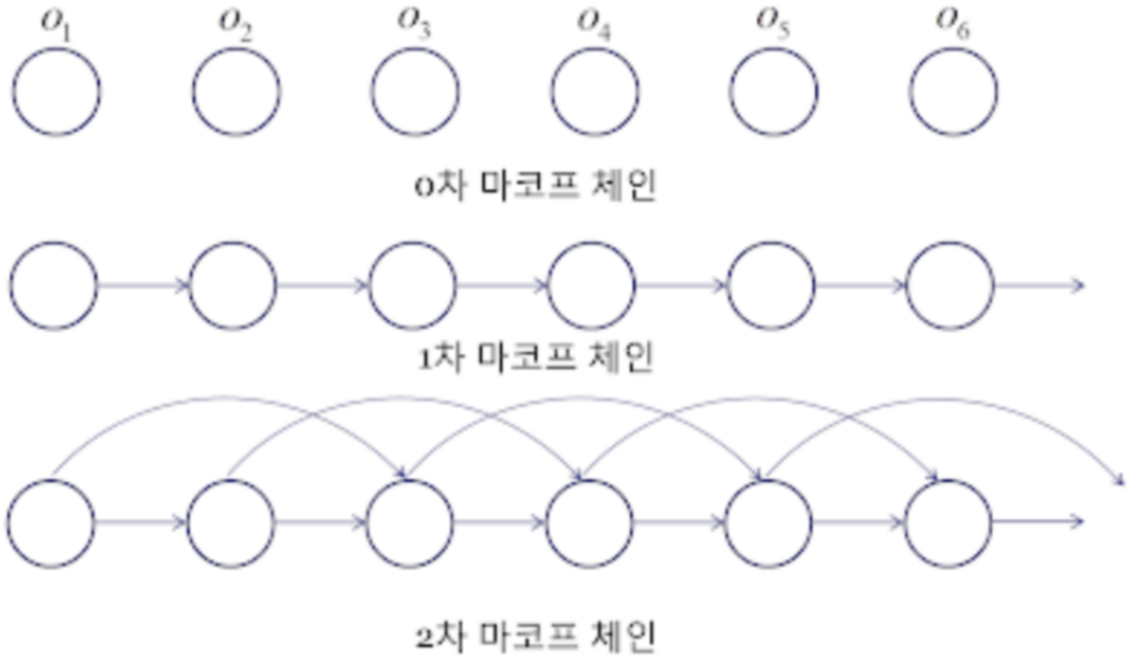
John Z. Clay

Baseball research journal 2023 Spring

https://sabr.org/journal/article/strategic-pitch-location-the-role-of-two-pitch-sequences-in-pitching-success/

Marcov Chain

현재 data는 과거의 data에 영향을 받는다!



Sequential pitch behavior

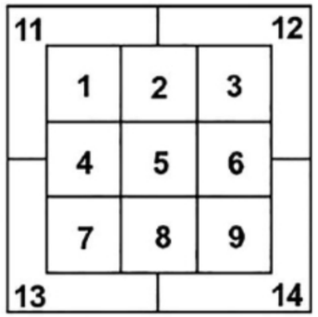
01 Type

- Fastball
- Breaking ball
- Change up

02 Velocity

03 Target Area

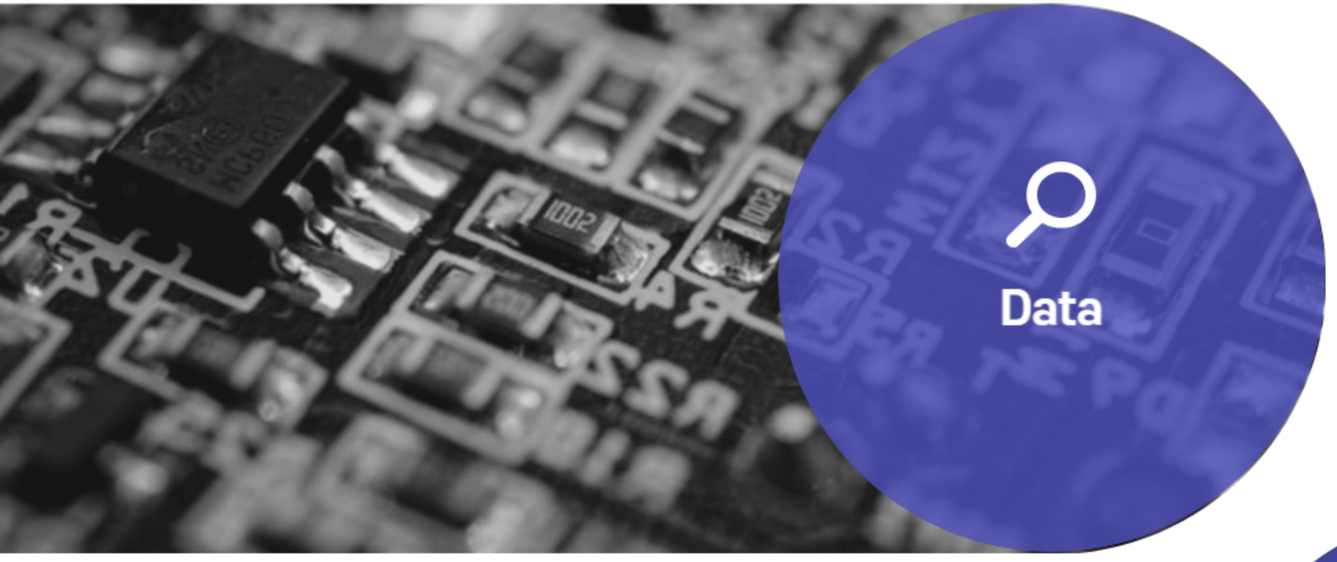
Figure 1. Strike Zone Locations



Depending on the previous action

02

Data



Data

2022 MLB season pitcher data

- > 1500 pitches
- starting pitchers
- https://baseballsavant.mlb.com/statcast_search

Analysis

Data analysis

- Set every pitcher's transition matrix
- Grouping pitchers (by K-means clustering)
- Use ML model to predict next pitch

DATA

	pitch_type	game_date	release_speed	release_pos_x	releas
0	FC	2022-10-05	94.4	-0.56	
1	CU	2022-10-05	81.5	-0.37	
2	SI	2022-10-05	97.0	-0.39	
3	CH	2022-10-05	90.4	-0.60	
4	FC	2022-10-05	95.5	-0.42	
...	...	...	...	...	
57758	SL	2022-04-17	82.2	0.73	
57759	FF	2022-04-17	94.7	0.27	
57760	SL	2022-04-17	82.0	0.77	
57761	SL	2022-04-17	84.0	0.75	
57762	SI	2022-04-17	93.9	0.64	

310323 rows × 92 columns

shape = 310323 X 92

Feature Selection

	pitch_name	game_date	batter	pitcher	player_name	description
0	Cutter	2022-10-05	672695	669203	Burnes, Corbin	hit_into_play
1	Curveball	2022-10-05	672695	669203	Burnes, Corbin	ball
2	Sinker	2022-10-05	672695	669203	Burnes, Corbin	ball
3	Changeup	2022-10-05	672695	669203	Burnes, Corbin	foul
4	Cutter	2022-10-05	672695	669203	Burnes, Corbin	ball
...	...	...	...	...	...	...
310318	Slider	2022-04-17	666185	676879	Ashby, Aaron	hit_into_play
310319	4-Seam Fastball	2022-04-17	666185	676879	Ashby, Aaron	foul
310320	Slider	2022-04-17	666185	676879	Ashby, Aaron	called_strike
310321	Slider	2022-04-17	666185	676879	Ashby, Aaron	ball
310322	Sinker	2022-04-17	666185	676879	Ashby, Aaron	called_strike

310323 rows × 17 columns

shape = 310323 * 17

'game_date', 'batter', 'pitcher_x', 'player_name', 'description',
'zone', 'stand', 'p_throws', 'plate_x', 'plate_z',
'inning', 'pitch_number',

DATA

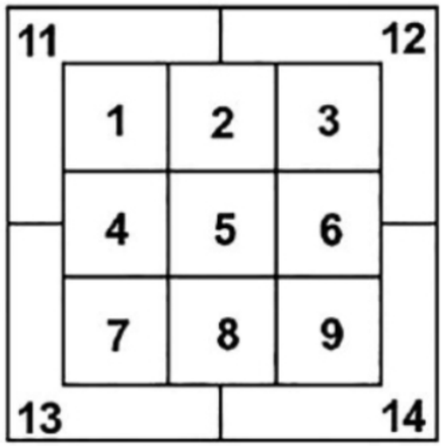
['Cutter','4-Seam Fastball','Split-Finger']
['Knuckle Curve','Curveball', 'Slow Curve']
['Slider','Slurve','Sweeper']
['Changeup']
['Sinker']
['Pitch Out', 'N']

fastball
curve
slider
changeup
sinker
etc

Transition matrix

- Discrete time Marcov chain (DTMC)

Figure 1. Strike Zone Locations



1st 2nd

- T_{ij} : probability that a pitcher transitioned from location i to location j
- # (i -> j) /total num

Transition matrix

Zone Sequence

and

Pitch type Sequence

- 13 Zone location
- Add starting point / Error point
- 15 X 15 matrix

- 총 13개의 pitch type 존재
- 6개로 simplify

Transition matrix code

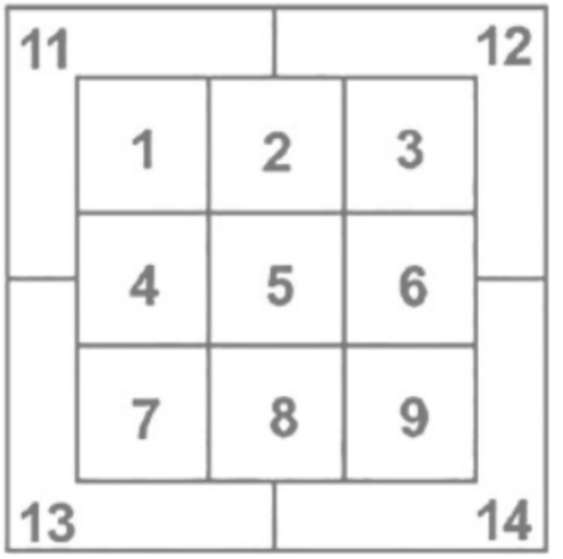
```python
def zonesequence1(data):

    for i in range(len(data)) :

        if data.at[i,"pitch_number"] == 1 :

            data.at[i,"zonesequence1"] = ("S",data.loc[i,"zone"])

        else :
            data.at[i,"zonesequence1"] = (data.loc[i-1,"zone"],data.loc[i,"zone"])

    names_1 = [1,2,3,4,5,6,7,8,9,11,12,13,14,"S"]
    t1 = pd.DataFrame(columns=names_1,index=names_1)

    for i in range(t1.shape[0]) :
        for j in range(t1.shape[1]) :
            a = names_1[i]
            b = names_1[j]
            c = (a,b)
            try:
                t1.loc[a,b] = data["zonesequence1"].value_counts()[c]
            except:    # 예외가 발생했을 때 실행됨
                t1.loc[a,b] = 0

    t1 = t1.div(t1.sum(axis=1),axis=0)

    return t1
```

```python
def pitchsequence1(data):

    data = data.dropna(subset=['pitch_name',"zone"])
    data =data.reset_index(drop=True)

    for i in range(len(data)) :

        if data.at[i,"pitch_number"] == 1 :
            data.at[i,"pitchsequence1"] = ("S",data.loc[i,"pitch_name"])
        else :
            data.at[i,"pitchsequence1"] = (str(data.loc[i-1,"pitch_name"]),str(data.loc[i,"pitch_name"]))

    names_3 = [pitch_name for pitch_name in data.pitch_name.unique()]
    names_3.append("S")
    t3 = pd.DataFrame(columns=names_3,index=names_3)


    for i in range(t3.shape[0]) :
        for j in range(t3.shape[1]) :
            a = names_3[i]
            b = names_3[j]
            c = (a,b)
            try:
                t3.loc[a,b] = data["pitchsequence1"].value_counts()[c]
            except:
                t3.loc[a,b] = 0

    ind = t3[t3.sum(axis=1)==0].index

    #딱 1번 던진 공 제거

    if len(ind) > 0 :
        t3 = t3.drop(labels=ind[0],axis=1)
        t3 = t3.drop(labels=ind[0],axis=0)

    t3 = t3.div(t3.sum(axis=1),axis=0)

    return t3
```

```python
def player_matrix(player_name) :

    data = total[total["player_name"] == player_name]
    data =data.reset_index(drop=True)
    data["zonesequence1"] = tuple
    data["pitchsequence1"] = tuple

    t1 = zonesequence1(data)
    t3= pitchsequence1(data)

    return  t1,t3
```

Transition matrix

"Ohtani, Shohei"



	1	2	3	4	5	6	7	8	9	11	12	13	14	15	S
1	8	4	1	14	10	6	2	6	5	15	3	4	20	0	0
2	1	5	2	3	4	5	5	3	9	8	4	6	23	0	0
3	1	2	2	1	3	11	0	2	1	12	5	3	11	0	0
4	1	8	5	13	6	3	4	11	5	23	5	11	26	0	0
5	5	5	4	10	14	8	5	6	14	21	7	14	47	0	0
6	7	7	3	8	12	7	4	4	6	16	7	9	40	0	0
7	2	6	2	4	4	0	2	3	2	6	2	10	17	0	0
8	3	2	5	4	12	8	3	4	2	11	2	15	27	0	0
9	1	6	3	5	7	9	7	5	7	11	9	8	37	0	0
11	18	14	4	13	31	20	14	19	15	45	7	28	45	0	0
12	3	5	4	11	11	11	5	4	11	10	5	4	12	0	0
13	9	7	2	13	16	13	8	7	4	19	7	17	31	0	0
14	24	18	18	45	52	42	17	36	38	40	32	35	135	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	42	25	16	42	65	52	21	42	46	80	31	38	161	0	0

	1	2	3	4	5	6	S
1	520	54	344	0	13	0	0
2	115	8	57	0	3	0	0
3	361	26	349	0	33	0	0
4	0	0	0	0	0	0	0
5	20	0	50	0	15	0	0
6	0	0	0	0	0	0	0
S	247	135	246	0	33	0	0

	1	2	3	4	5	6	S
1	0.558539	0.058002	0.369495	0.0	0.013963	0.0	0.0
2	0.628415	0.043716	0.311475	0.0	0.016393	0.0	0.0
3	0.469441	0.033810	0.453836	0.0	0.042913	0.0	0.0
4	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0
5	0.235294	0.000000	0.588235	0.0	0.176471	0.0	0.0
6	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0
S	0.373676	0.204236	0.372163	0.0	0.049924	0.0	0.0

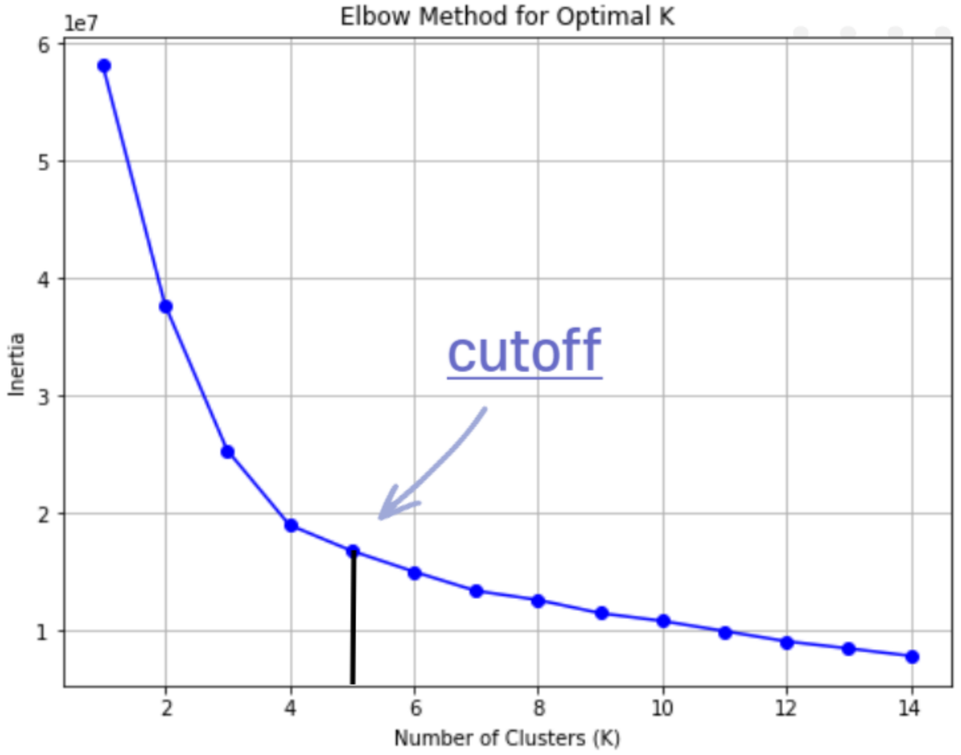
Clustering

according to their similarities in the transition variables

Euclidean distance
Make similarity matrix

K-means

Elbow plot
determine K



Select k-means model with k=5

Pitch Group

	player_name	zone_group	pitch_type_group
0	Ashby, Aaron	1	1
1	Strider, Spencer	0	4
2	Rogers, Trevor	2	4
3	Skubal, Tarik	2	2
4	Wells, Tyler	0	4
...	...	...	...
125	Cease, Dylan	3	4
126	Wainwright, Adam	0	1
127	Alcantara, Sandy	0	3
128	Cole, Gerrit	0	4
129	Burnes, Corbin	3	4

zone group 1

	player_name	zone_group	pitch_type_group
0	Ashby, Aaron	1	1
7	Luzardo, Jesús	1	2
8	Lodolo, Nick	1	3
12	Peterson, David	1	2
31	Kershaw, Clayton	1	4
33	Detmers, Reid	1	4
43	Steele, Justin	1	4
54	Snell, Blake	1	4
65	Hill, Rich	1	0
78	Lauer, Eric	1	4
115	Rodón, Carlos	1	4
119	Valdez, Framber	1	4

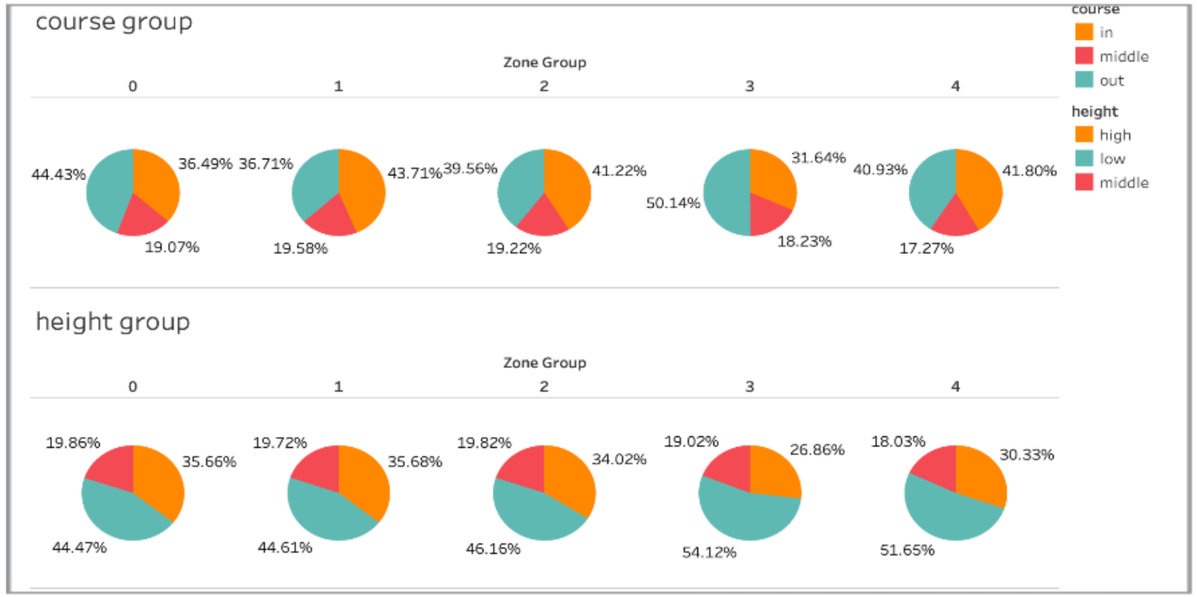
73	Ohtani, Shohei	0	4
74	Ryan, Joe	0	4
78	Lauer, Eric	1	4
81	Manaea, Sean	2	4
82	Urías, Julio	2	4
86	Giolito, Lucas	0	4
98	Verlander, Justin	0	4
100	Taillon, Jameson	0	4
106	Bieber, Shane	3	4
109	López, Pablo	4	4
111	Gallen, Zac	4	4
113	Darvish, Yu	0	4
115	Rodón, Carlos	1	4
118	Gilbert, Logan	0	4
125	Cease, Dylan	3	4
128	Cole, Gerrit	0	4

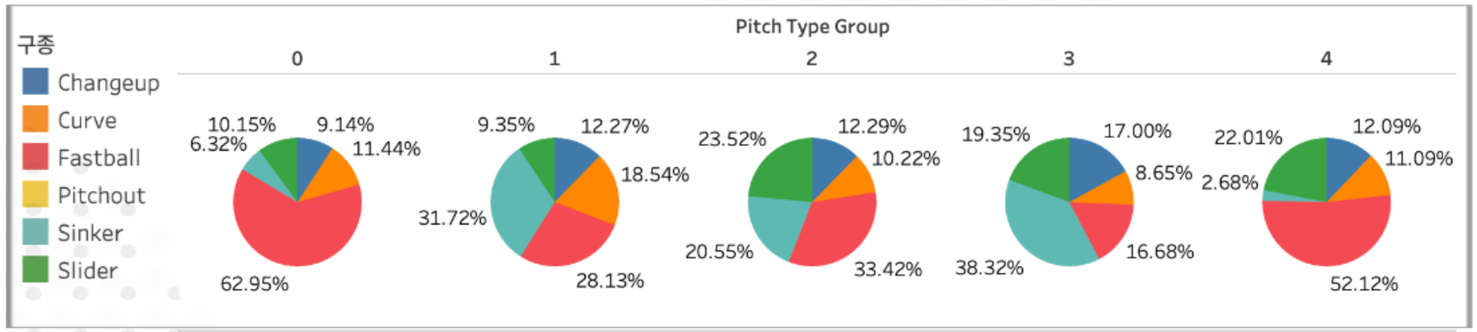
Pitch type group 4

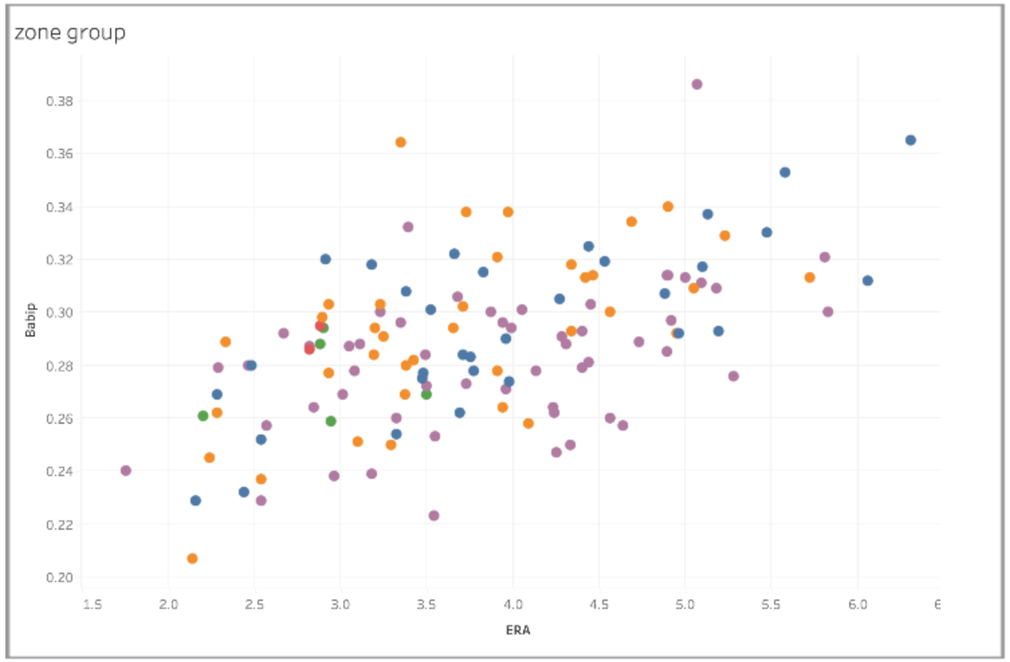
03

Visualization

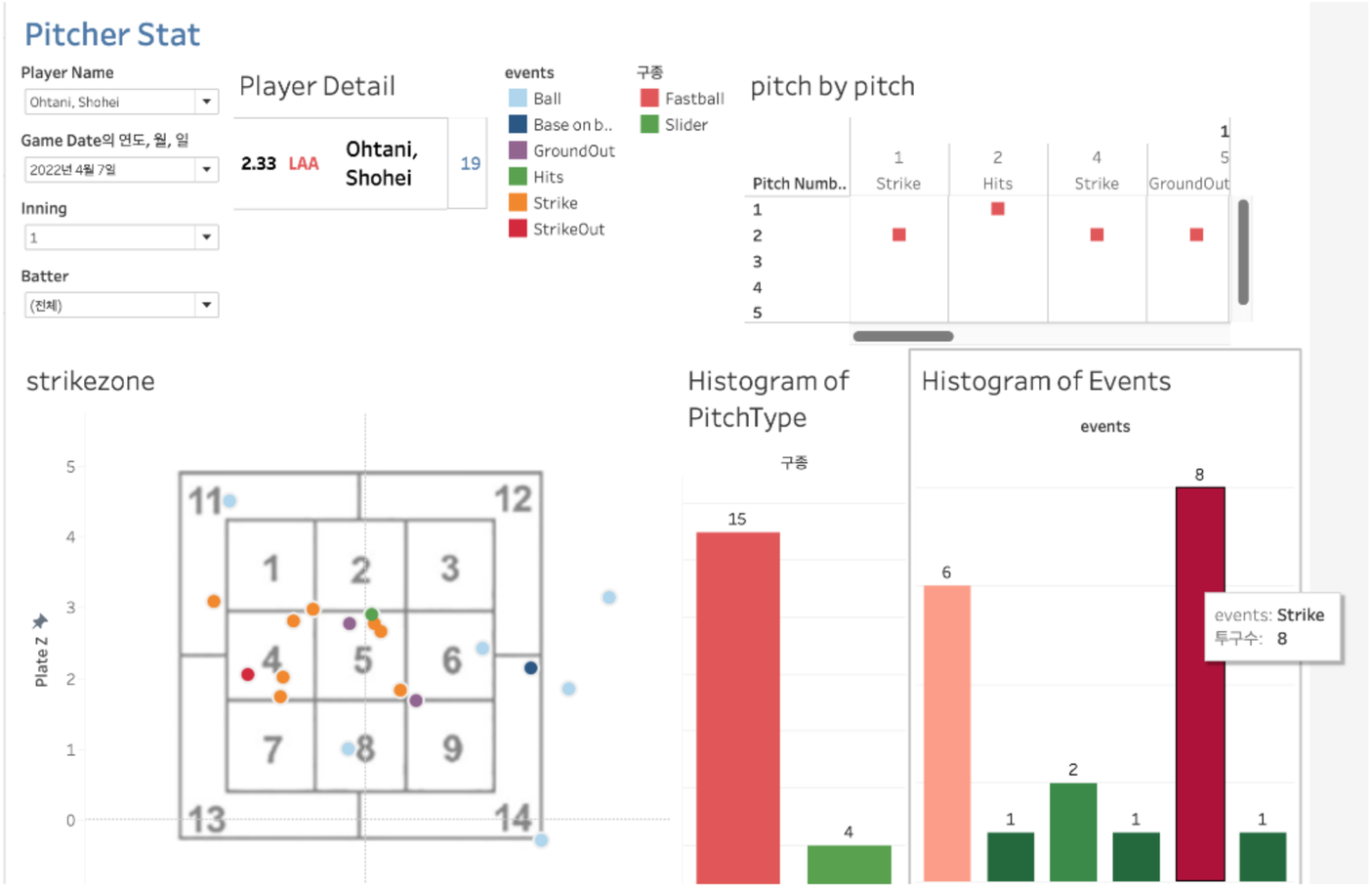
Graph


course group
Zone Group
0
1
2
3
4
44.43%
36.49%
19.07%
36.71%
43.71%
19.58%
39.56%
41.22%
19.22%
50.14%
31.64%
18.23%
40.93%
41.80%
17.27%
course
in
middle
out
height
high
low
middle
height group
Zone Group
0
1
2
3
4
19.86%
35.66%
44.47%
19.72%
35.68%
44.61%
19.82%
34.02%
46.16%
19.02%
26.86%
54.12%
18.03%
30.33%
51.65%


Pitch Type Group
구종
Changeup
Curve
Fastball
Pitchout
Sinker
Slider
0
1
2
3
4
10.15%
9.14%
6.32%
11.44%
62.95%
9.35%
12.27%
18.54%
31.72%
28.13%
23.52%
12.29%
10.22%
20.55%
33.42%
19.35%
17.00%
8.65%
38.32%
16.68%
22.01%
12.09%
11.09%
2.68%
52.12%


zone group
Babip
0.38
0.36
0.34
0.32
0.30
0.28
0.26
0.24
0.22
0.20
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
ERA

Tableau

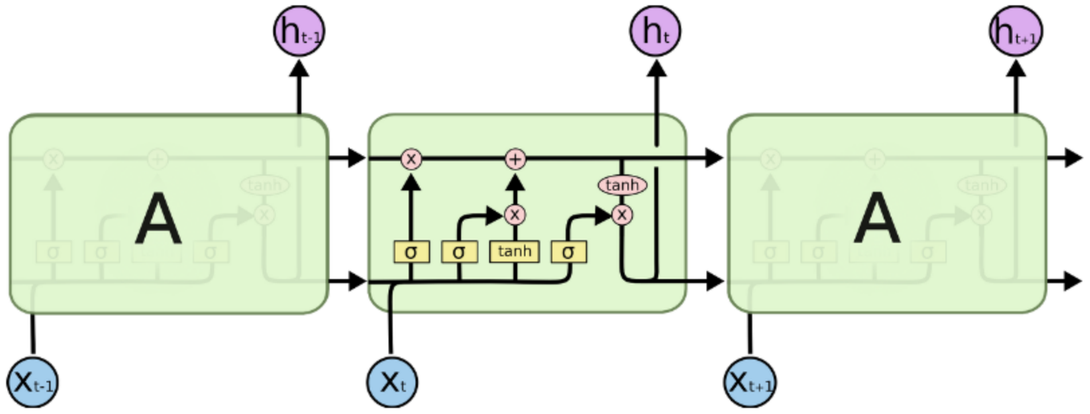

Pitcher Stat
Player Name
Ohtani, Shohei
Game Date의 연도, 월, 일
2022년 4월 7일
Inning
1
Batter
(전체)
Player Detail
2.33 LAA
Ohtani, Shohei
19
events
Ball
Base on b..
GroundOut
Hits
Strike
StrikeOut
구종
Fastball
Slider
pitch by pitch
Pitch Numb..
1 Strike
2 Hits
4 Strike
5 GroundOut
strikezone
Plate Z
11
12
13
14
Histogram of PitchType
구종
15
4
Histogram of Events
events
6
1
2
1
8
1
events: Strike
투구수: 8

04

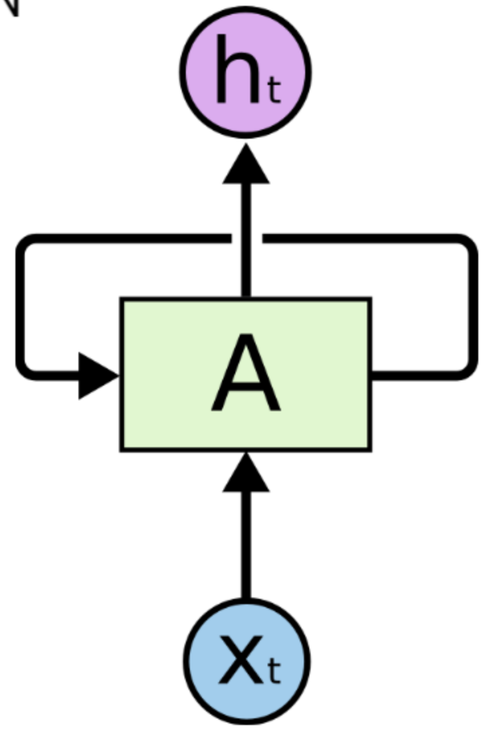
Modeling

1st trial : RNN / LSTM

LSTM(Long Short-Term memory network)



RNN



1st trial : RNN / LSTM

- Data

	pitch_name	pitcher_x	description	stand	p_throws	plate_x	plate_z	in
0	5	676879	83	2	1	0.71	2.94	
1	3	676879	66	2	1	-1.29	0.84	
2	3	676879	83	2	1	-0.40	2.27	
3	1	676879	83	2	1	1.15	3.25	
4	3	676879	71	2	1	0.84	2.33	
...	...	...	...	...	...	...	...	
5863	1	669203	66	1	2	1.69	1.42	
5864	4	669203	83	1	2	0.09	1.36	
5865	5	669203	66	1	2	2.09	2.88	
5866	2	669203	66	1	2	-0.70	0.84	
5867	1	669203	71	1	2	0.47	2.96	

player1

AFFECT!

player2

- Model

```
for i, _ in enumerate(dataset):
    idx_in = i + seq_len
    idx_out = idx_in + steps
    if idx_out > len(dataset):
        break
    seq_x = dataset[i:idx_in, :-1]
    if single_output:
        seq_y = dataset[idx_out - 1 : idx_out, -1]
    else:
        seq_y = dataset[idx_in:idx_out, -1]
    X.append(seq_x)
    y.append(seq_y)
```

```
X = np.array(X)
y = np.array(y)
```

```
X.shape
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_val, Y_train, Y_val = train_test_split(X, y, shuffle=True, test_size=0.3, random_state=42)
```

```
model1=Sequential()
```

```
model1.add(tf.keras.layers.LSTM(128,dropout=0.2,recurrent_dropout=0.2))
model1.add(Dense(14, activation='softmax'))  # 가능한 14개의 위치에 대한 softmax 출력
```

```
model1.compile(loss='categorical_crossentropy',optimizer='adam',metrics=["mse", "accuracy"])
```

```
history = model1.fit(X_train, Y_train, batch_size=32,epochs=10,validation_data=(X_val, Y_val))
```

- > 다른 시퀀스의 pitch or 다른 pitcher의 공도 참조하는 문제가 생김

- > pitcher number가 좋은 변수로 작용하지 않고 오히려 결과를 왜곡시키는 경향이 있었음

-> 우리가 관심있는 것은 장기 기억이 아니라 two pitch sequence (바로 직전 과거) 이기때문에 0
굳이 딥러닝과 같은 무거운 모델을 쓸 필요가 없었음

2nd trial : Machine learning

- Data

	pitch_name	pitcher_x	description	stand	p_throws	plate_x	plate_z	inning	pitch_number	zone_group	prev_zone	y
0	5	676879	83	2	1	0.71	2.94	1	1	1	0.0	3.0
1	3	676879	66	2	1	-1.29	0.84	1	2	1	3.0	13.0
2	3	676879	83	2	1	-0.40	2.27	1	3	1	13.0	4.0
3	1	676879	83	2	1	1.15	3.25	1	4	1	4.0	12.0
4	3	676879	71	2	1	0.84	2.33	1	5	1	12.0	14.0
...	...	...	...	...	...	...	...	...	...	...	...	...
305863	1	669203	66	1	2	1.69	1.42	3	2	3	8.0	14.0
305864	4	669203	83	1	2	0.09	1.36	3	3	3	14.0	14.0
305865	5	669203	66	1	2	2.09	2					
305866	2	669203	66	1	2	-0.70	0					
305867	1	669203	71	1	2	0.47	2					

-> 바로 직전에 던진 투구 위치를 새로운 컬럼으로 넣어줌
-> pitch group 과 같이 zone예측에 관련없는 변수는 제거해줌
-> 이전 위치가 다음 위치에 미치는 영향을 알고 싶은 것이므로,
직접적인 투수 성능 지표(ERA/OBP 등)은 누락시킴

바로 직전에 던진 피치 타입을 새로운 컬럼으로 넣어줌 ->
zone group 과 같이 pitch type 예측에 관련없는 변수는 제거해줌 ->

	pitcher_x	description	zone	stand	p_throws	plate_x	plate_z	inning	pitch_number	pitch_type_group	prev_pitch	y
0	676879	83	3.0	2	1	0.71	2.94	1	1	1	0	5
1	676879	66	13.0	2	1	-1.29	0.84	1	2	1	5	3
2	676879	83	4.0	2	1	-0.40	2.27	1	3	1	3	3
3	676879	83	12.0	2	1	1.15	3.25	1	4	1	3	1
4	676879	71	14.0	2	1	0.84	2.33	1	5	1	1	3
...	...	...	...	...	...	...	...	...	...	...	...	...
305563	669203	66	13.0	1	2	-0.54	1.40	4	2	4	2	1
305564	669203	66	14.0	1	2	0.69	0.96	4	3	4	1	1
305565	669203	66	13.0	1	2	-0.01	0.72	4	4	4	1	4
305566	669203	83	5.0	1	2	-0.15	2.54	4	5	4	4	1
305567	669203	71	8.0	1	2	-0.04	2.02	4	6	4	1	1

2nd trial : Machine learning

Search best model for predicting next Zone location

```
best_model = compare_models(exclude = ['svm','rbfsvm'], sort='Accuracy')
```

Initiated	06:55:07
Status	Fitting 5 Folds
Estimator	Gradient Boosting Classifier

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.8864	0.9954	0.8900	0.8883	0.8891	0.8741	0.8741	31.9820
qda	Quadratic Discriminant Analysis	0.8446							
dt	Decision Tree Classifier	0.8362							
ridge	Ridge Classifier	0.4534							
knn	K Neighbors Classifier	0.4056							
ada	Ada Boost Classifier	0.3277							
nb	Naive Bayes	0.3178							
lr	Logistic Regression	0.1851							

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.8905	0.9967	0.8941	0.8924	0.8932	0.8787	0.8787	395.8360
xgboost	Extreme Gradient Boosting	0.8871	0.9965	0.8908	0.8888	0.8897	0.8750	0.8750	27.0980
rf	Random Forest Classifier	0.8864	0.9954	0.8900	0.8883	0.8891	0.8741	0.8741	31.9820
qda	Quadratic Discriminant Analysis	0.8446	0.9916	0.8510	0.8473	0.8486	0.8279	0.8279	0.5720
et	Extra Trees Classifier	0.8446	0.9908	0.8500	0.8444	0.8470	0.8277	0.8278	33.4380
dt	Decision Tree Classifier	0.8362	0.9112	0.8396	0.8402	0.8399	0.8185	0.8185	1.6060
lda	Linear Discriminant Analysis	0.7321	0.9782	0.7465	0.7401	0.7341	0.7033	0.7045	1.6740
lightgbm	Light Gradient Boosting Machine	0.6749	0.8293	0.6663	0.6915	0.6581	0.6407	0.6464	79.3280
ridge	Ridge Classifier	0.4534	0.0000	0.4601	0.3309	0.3165	0.3646	0.3890	0.3360
knn	K Neighbors Classifier	0.4056	0.7851	0.4058	0.4347	0.4173	0.3447	0.3455	6.3520
ada	Ada Boost Classifier	0.3277	0.7656	0.3764	0.1437	0.2067	0.2092	0.2438	10.5200
nb	Naive Bayes	0.3178	0.7918	0.3504	0.3093	0.2846	0.2203	0.2425	0.4460
lr	Logistic Regression	0.1851	0.4994	0.2127	0.0394	0.0664	0.0000	0.0000	6.6580

Gradient
Extreme Gradient Boosting
is the best model
with 0.89 accuracy

2nd trial : Machine learning

Search best model for predicting next Zone location

선택된 모델 (Gradient Boosting Classifier) 로 하이퍼 파라미터 튜닝 진행

```
#gradient boosting
model_GB = GradientBoostingClassifier(random_state=42)

gridParams = {
    "n_estimators" :[700, 800, 900],
    "max_depth" : [8, 9 ,10],
    "learning_rate" :[0.1, 0.05],
    'min_samples_leaf': [0.1,0.3]
}


grid_cv_GB = GridSearchCV(model_GB, param_grid=gridParams, cv=5 , scoring="accuracy", n_jobs=-1)
grid_cv_GB.fit(x_train, y_train)

print('최적 하이퍼 파라미터: ', grid_cv_GB.best_params_)
```

```
최적 하이퍼 파라미터:  {'learning_rate': 0.1, 'max_depth': 8, 'min_samples_leaf': 0.1, 'n_estimators': 700}
```

result

	y	pred
0	13	13
1	7	13
2	14	14
3	8	8
4	12	2
...	...	...
38995	14	14
38996	14	14
38997	12	12
38998	13	13
38999	3	3

39000 rows × 2 columns

```
accuracy = accuracy_score(original, y_pred)
accuracy
```

```
0.8821025641025642
```

2nd trial : Machine learning

Search best model for predicting next pitch type

```
[12] best_model = compare_models(exclude = ['svm','rbfsvm'], sort='Accuracy')
```

Initiated	06:36:36
Status	Fitting 5 Folds
Estimator	Gradient Boosting Classifier

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.5960	0.8327	0.4615	0.5369	0.4904	0.4225	0.4282	17.5220
knn	K Neighbors Classifier	0.5489							
dt	Decision Tree Classifier	0.4899							
ridge	Ridge Classifier	0.4653							
nb	Naive Bayes	0.4470							
lr	Logistic Regression	0.4294							
ada	Ada Boost Classifier	0.4223							
qda	Quadratic Discriminant Analysis	0.1035							

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.6237	0.8656	0.5012	0.5745	0.5294	0.4650	0.4700	7.0760
rf	Random Forest Classifier	0.5960	0.8327	0.4615	0.5369	0.4904	0.4225	0.4282	17.5220
gbc	Gradient Boosting Classifier	0.5813	0.8246	0.4255	0.5351	0.4618	0.3901	0.4009	81.5280
et	Extra Trees Classifier	0.5812	0.8222	0.4540	0.5135	0.4782	0.4053	0.4091	14.8780
lightgbm	Light Gradient Boosting Machine	0.5648	0.7651	0.4455	0.4991	0.4671	0.3857	0.3887	21.5960
knn	K Neighbors Classifier	0.5489	0.7798	0.4091	0.4769	0.4374	0.3553	0.3598	3.0660
dt	Decision Tree Classifier	0.4899	0.6573	0.4095	0.4050	0.4071	0.3057	0.3057	1.2820
lda	Linear Discriminant Analysis	0.4687	0.7084	0.2306	0.3480	0.2515	0.1793	0.2003	0.4300
ridge	Ridge Classifier	0.4653	0.0000	0.1972	0.3238	0.1992	0.1461	0.1783	0.2860
nb	Naive Bayes	0.4470	0.6540	0.1281	0.2419	0.1172	0.0608	0.1130	0.3380
lr	Logistic Regression	0.4294	0.4938	0.0859	0.0369	0.0516	0.0000	0.0000	1.3220
dummy	Dummy Classifier	0.4294	0.5000	0.0859	0.0369	0.0516	0.0000	0.0000	0.2740
ada	Ada Boost Classifier	0.4223	0.6262	0.1005	0.1939	0.0766	0.0216	0.0430	4.2440
qda	Quadratic Discriminant Analysis	0.1035	0.1499	0.0639	0.0895	0.0708	0.0583	0.0602	0.3160

-> XGBOOST

2nd trial : Machine learning

Search best model for predicting next pitch type

선택된 모델 (XGboost classifier) 로 하이퍼 파라미터 튜닝 진행

```
from xgboost import XGBClassifier
from sklearn.model_selection import StratifiedKFold, GridSearchCV

model_XGB = XGBClassifier(eval_metric='mlogloss', silent=True)

param_grid={'booster' :['gbtree'],
                  'max_depth':[6,8,10],
                  'min_child_weight':[1,3, 5, 7],
                  'gamma':[0,1,2,3],
                  'nthread':[4],
                  'colsample_bytree':[0.5,0.8],
                  'colsample_bylevel':[0.6,0.9],
                  'n_estimators':[100, 150, 200],
                  'random_state':[42]}


grid_cv_XGB=GridSearchCV(model_XGB, param_grid=param_grid, cv=StratifiedKFoldd(5),scoring='accuracy',n_jobs=-1)
grid_cv_XGB.fit(x_train, y_train)
print('최적 하이퍼 파라미터: ', grid_cv_XGB.best_params_)
print('최고 예측 정확도: {:.4f}'.format(grid_cv_XGB.best_score_))
```

result

	pitch_name	pred
0	4	3
1	4	4
2	2	5
3	4	4
4	3	5
...	...	...
38995	1	1
38996	4	2
38997	5	1
38998	2	4
38999	1	1

39000 rows × 2 columns

```
ACCURACY(test_y, y_pred)
```

0.6177435897435898

05

- **Discussion**
- **Q &A**