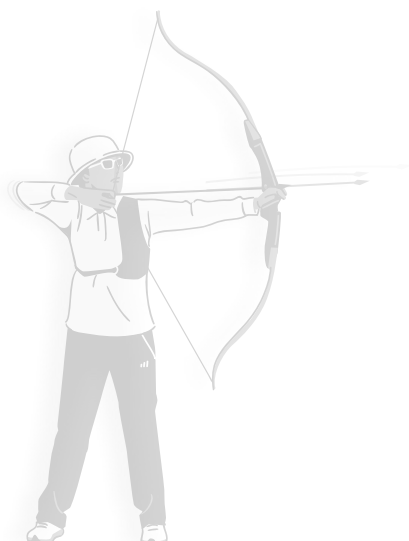


TOKYO 2020

도쿄 올림픽 EDA 및 머신러닝 모델 만들기

메달 개수에 따른 순위예측 모델 & 정보입력시 해당 팀명 출력 모델



2

1

3



멋쟁이사자처럼 13회차 전 예슬



CONTENTS



도쿄 올림픽 2020 EDA

- 선수 분석
- 메달분석 분석
- 참가자 성별 분석



데이터 전처리

- 데이터 라벨링



머신러닝 모델 만들기

- (1) 메달 개수에 따른 순위예측 모델
 - 기본모델(KNN모델) 성능개선
 - 여러가지 모델 비교
- (2) 정보입력시 해당 팀명 출력 모델
 - 기본모델(KNN모델) 만들기

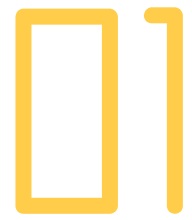


결론



도쿄 올림픽 EDA



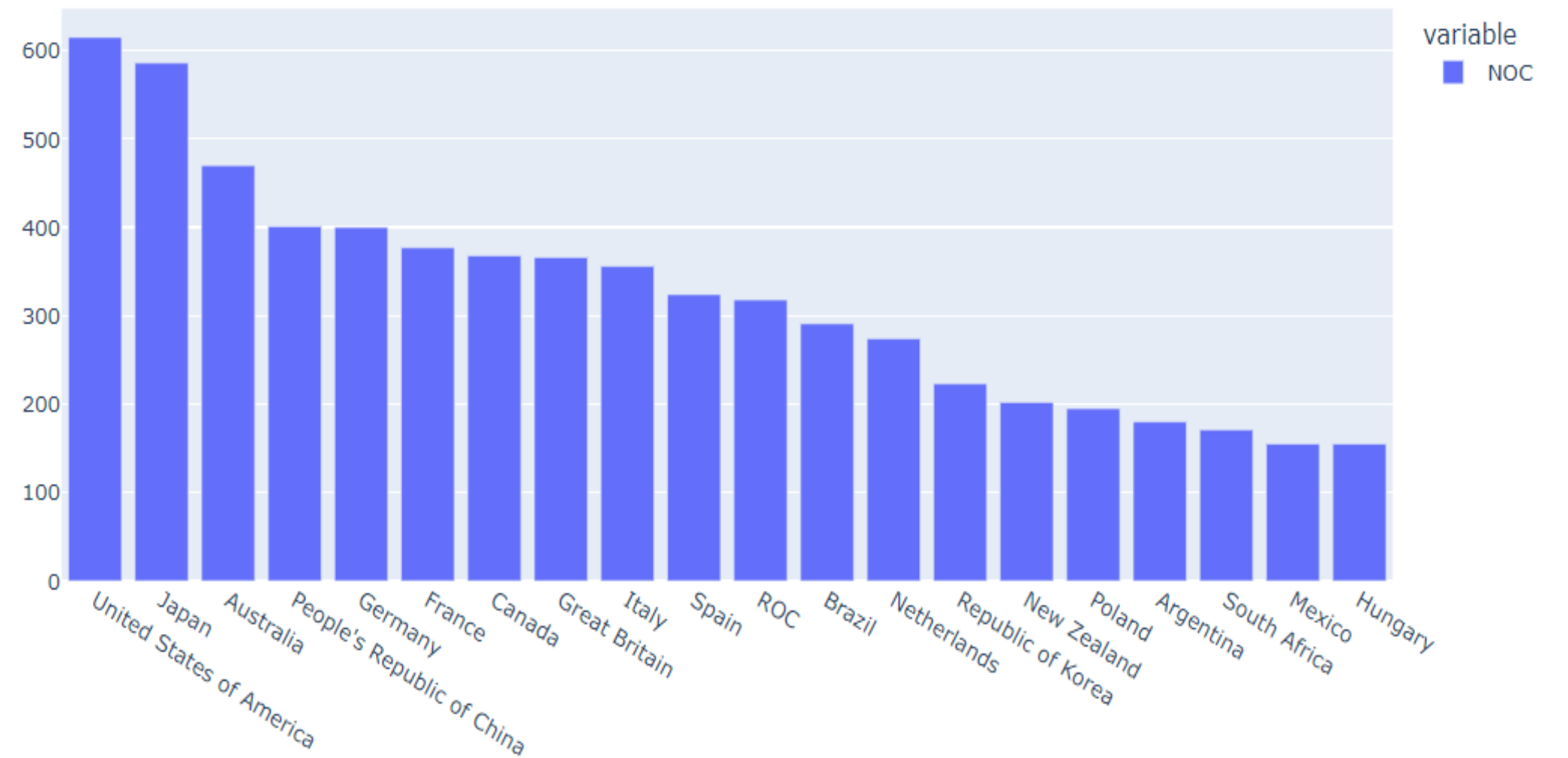


도쿄 올림픽 EDA 선수 분석

```
x = athlete.NOC.value_counts()
```

```
px.bar(x[:20], title="나라별 참가자수 상위 20")
```

나라별 참가자수 상위 20



- 미국,일본,호주 순으로 많은 선수가 참가하였다.
- 참가자 상위20위 안에 일본을 제외한 아시아 국가는 없다.

2

1

3

01

도쿄 올림픽 EDA 메달 분석

금메달 개수

은메달 개수

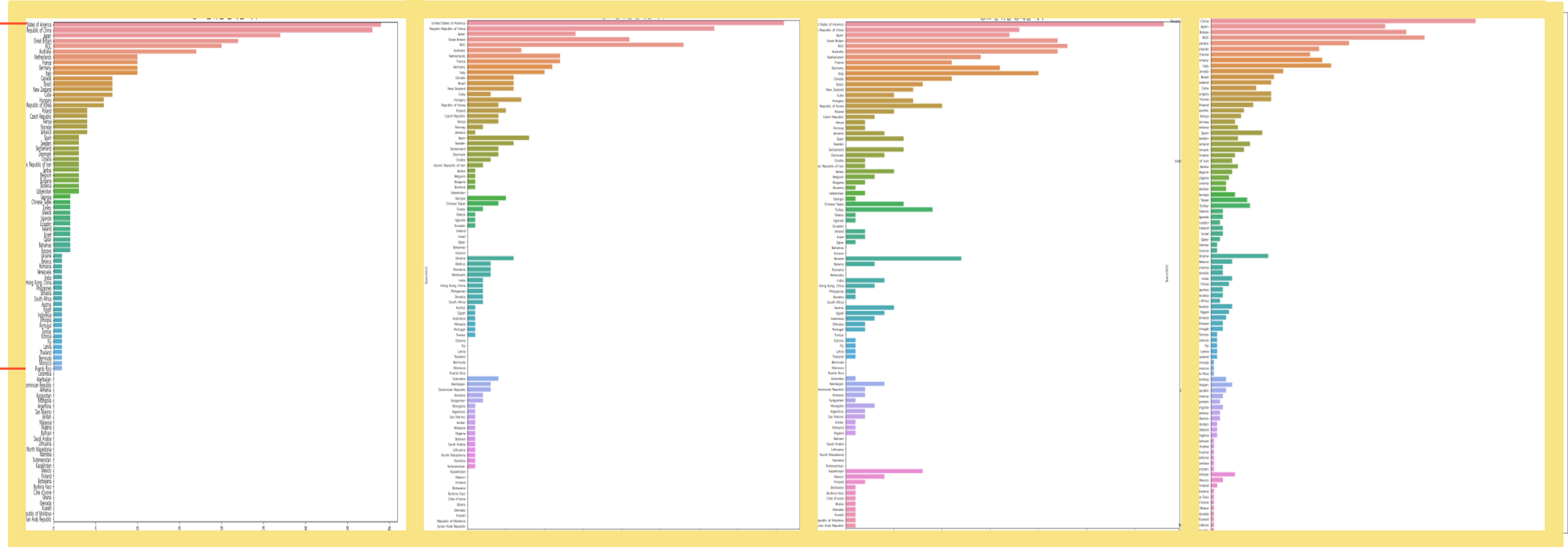
동메달 개수

전체메달 개수

1위

순위

86위



- 올림픽 랭크 순위대로 금메달이 많다.
- 올림픽 랭크는 전체 메달 수와 무관하다.

2

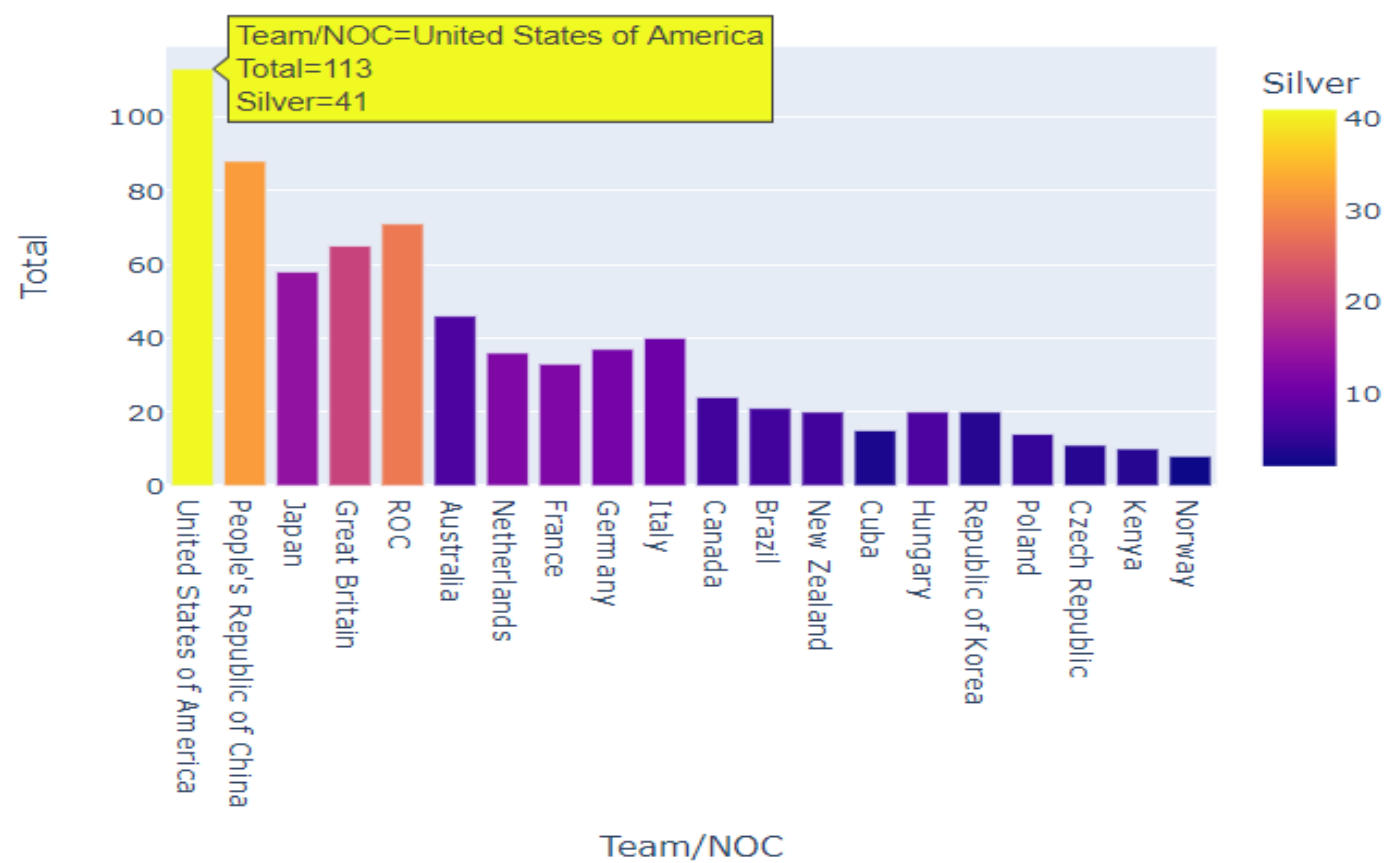
1

3

01 도쿄 올림픽 EDA 메달 분석

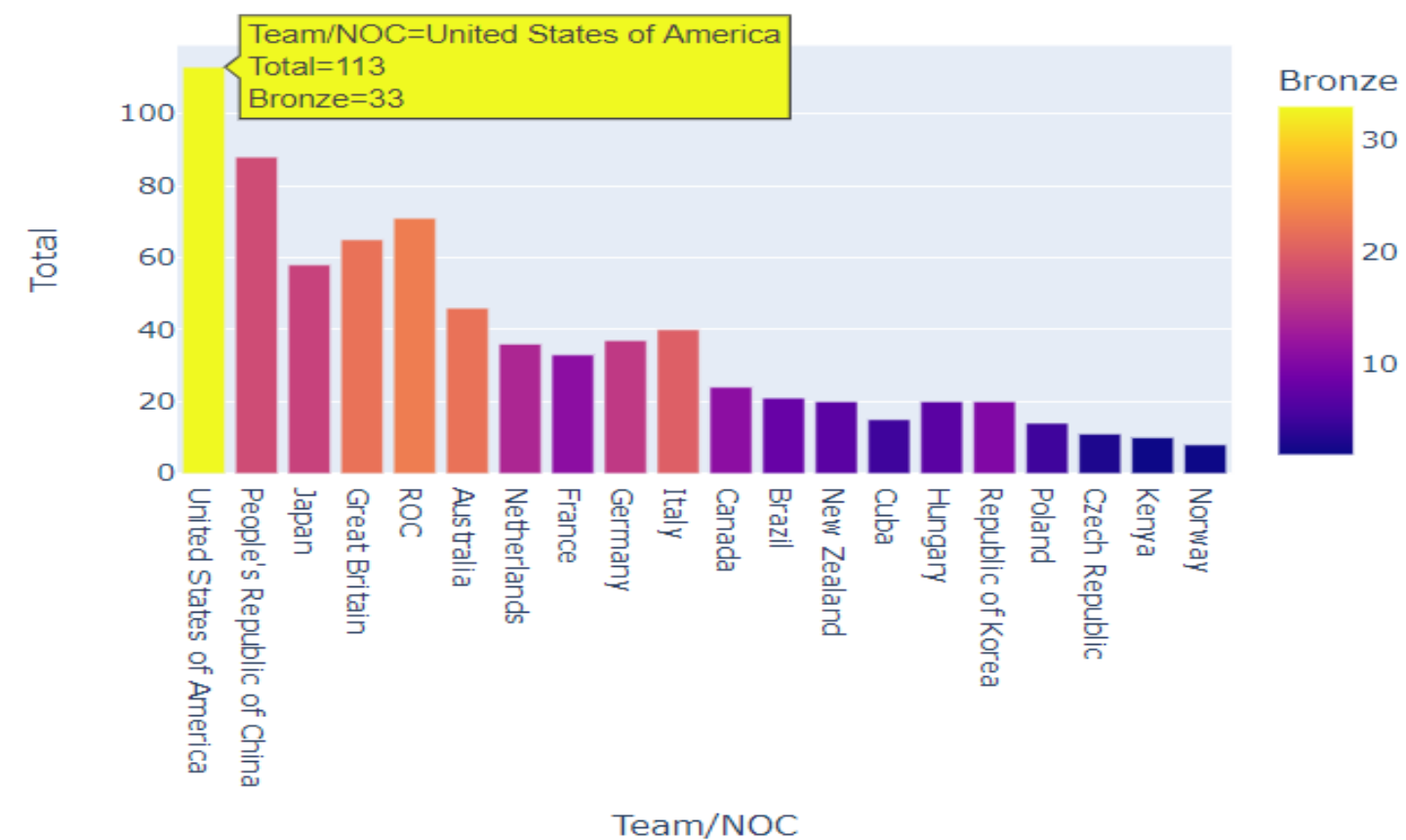
```
px.bar(medal[:20], x="Team/NOC", y="Total", color="Silver",  
       title="상위 20개국의 전체 메달 및 은메달 비율 ")
```

상위 20개국의 전체 메달 및 은메달 비율



```
px.bar(medal[:20], x="Team/NOC", y="Total", color="Bronze",  
       title="상위 20개국의 전체 메달 및 동메달 비율 ")
```

상위 20개국의 전체 메달 및 동메달 비율



- 올림픽 랭크 순위는 은메달 동메달 비율과 무관하다.

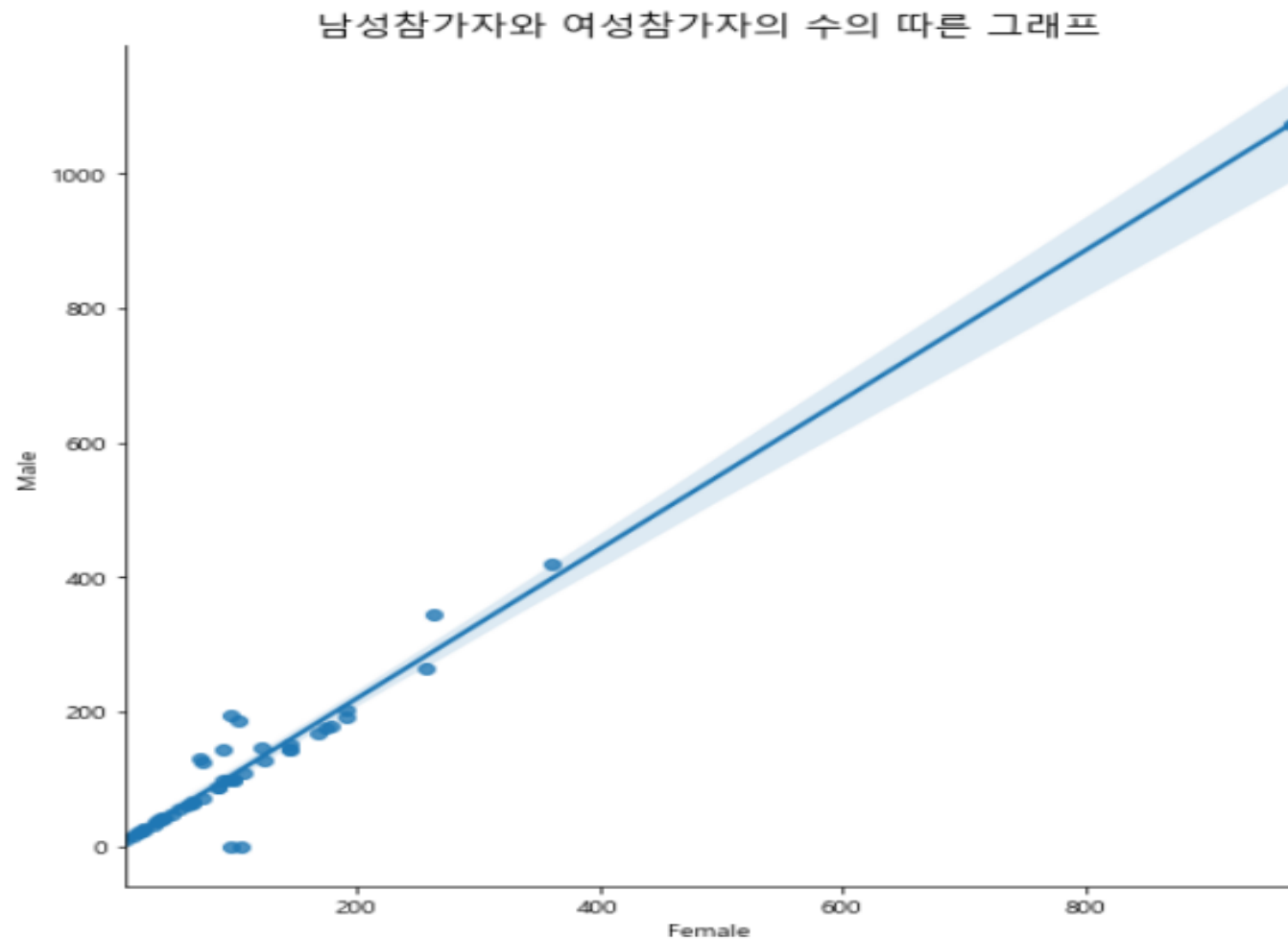
2

1

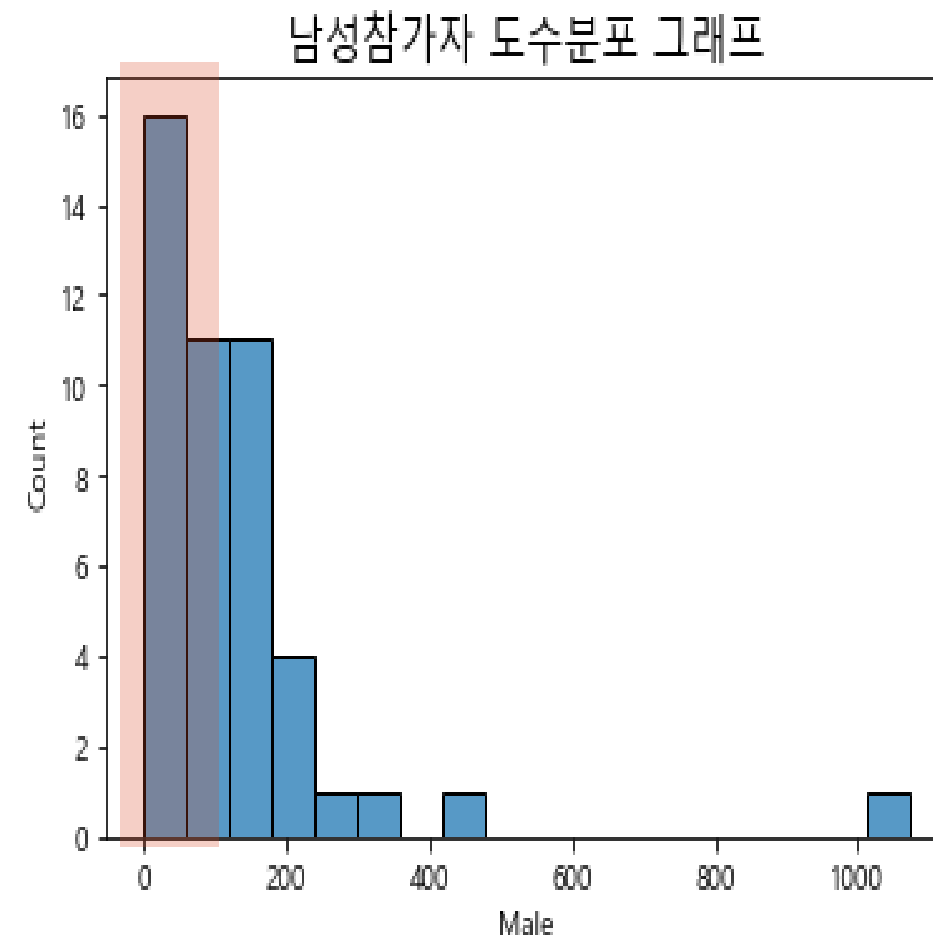
3

01 도쿄 올림픽 EDA 성별 분석

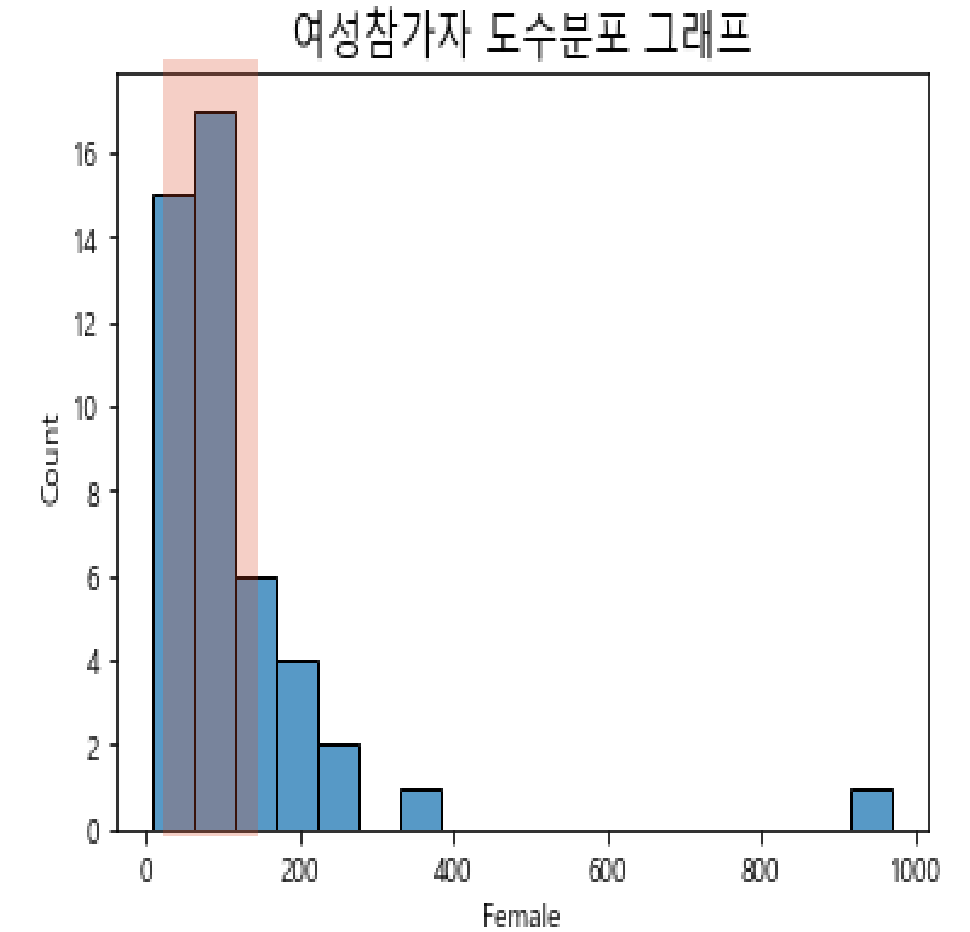
```
sns.lmplot(x='Female', y='Male', data=gender, size=7)
plt.title("남성참가자와 여성참가자의 수의 따른 그래프", fontsize=15)
plt.show()
```



```
sns.histplot(gender.Male)
plt.title("남성참가자 도수분포 그래프", fontsize=15)
```



```
sns.histplot(gender.Female)
plt.title("여성참가자 도수분포 그래프", fontsize=15)
```



- 남성참가자와 여성참가자의 수는 대등하다.
- 400명 이상 참가하는 종목은 거의 없다.
- 종목별 로 여성 참가자는 [100명 이하 참가자] 남성 참가자는 [약70명 이하] 참가자가 가장 많다.

2

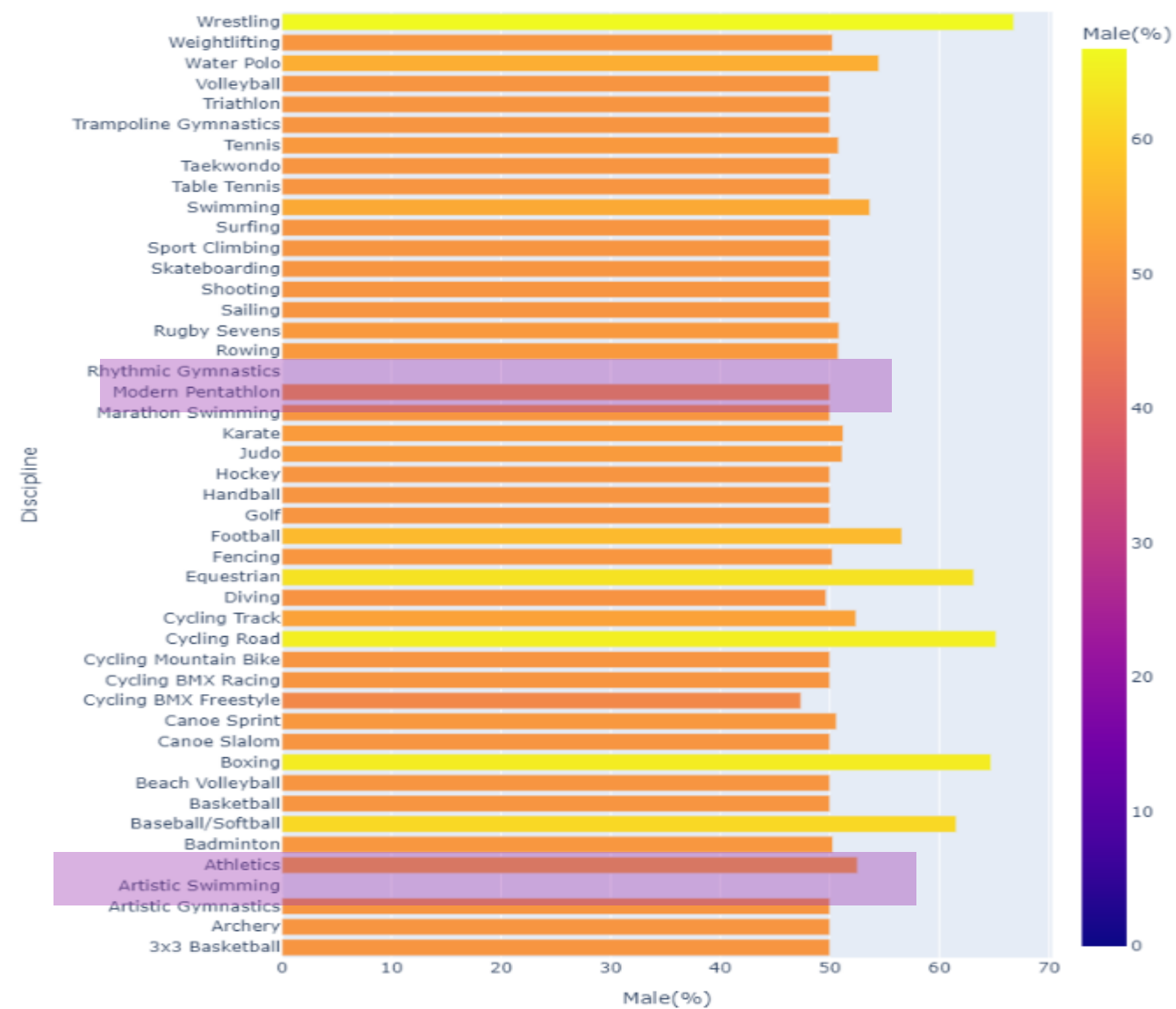
1

3

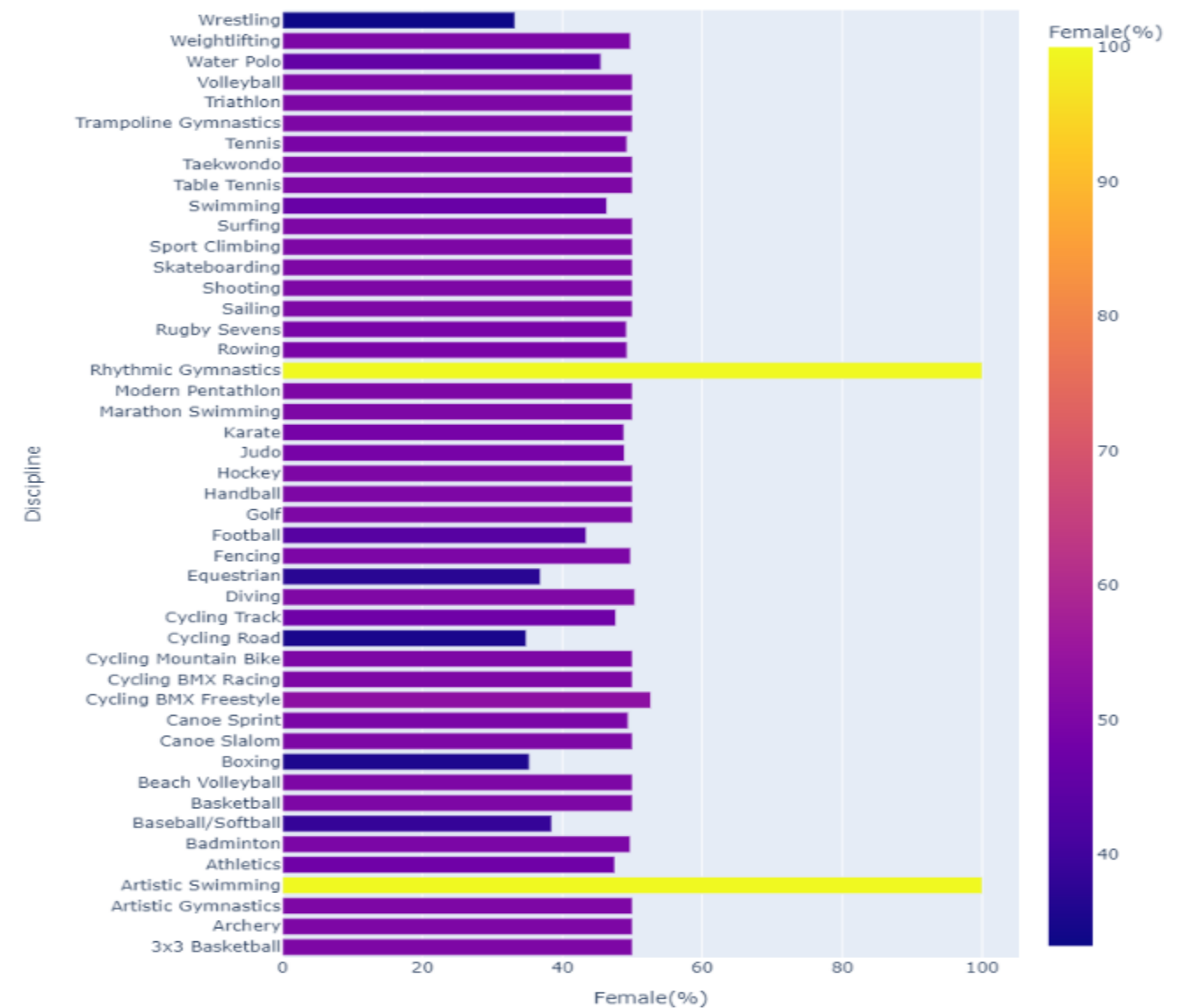
01

도쿄 올림픽 EDA 성별 분석

종목별 참가자 및 남성 비율



종목별 참가자 및 여성 비율



2

1

3

- 여성은 전종목에 참여하였지만, 남성은 두 종목의 경기([Artistic Swimming], [Rhythmic Gymnastics])에 참가하지 않았다.
- 대부분 종목에서 남성의 비율이 절반(50%) 이상이다. 그 중 가장 많은 비율의 남성이 참가한 종목은 [Wrestling]이다.

01 도쿄 올림픽 EDA 성별 분석

[성별 비율] 새로운 column 추가

```
gender["Female(%)"] = gender['Female'] / gender['Total'] * 100
gender["Male(%)"] = gender['Male'] / gender['Total'] * 100
gender.head()
```

[여성 > 남성] 데이터 추출

```
gender_2 = gender.loc[(gender.Female > gender.Male), :]
gender_2
```

	Discipline	Female	Male	Total	Female(%)	Male(%)
3	Artistic Swimming	105	0	105	100.000000	0.000000
12	Cycling BMX Freestyle	10	9	19	52.631579	47.368421
17	Diving	72	71	143	50.349650	49.650350
28	Rhythmic Gymnastics	96	0	96	100.000000	0.000000

종목별 참가자

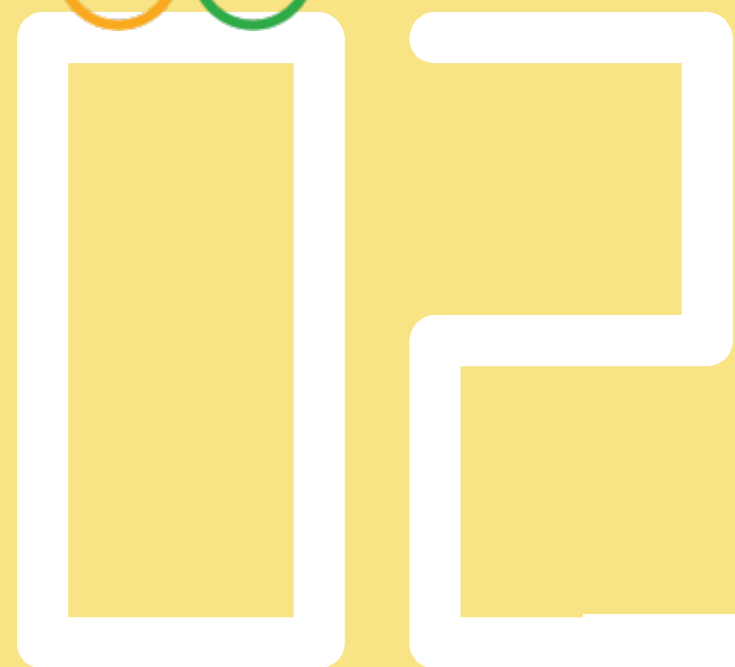


- 전체 46종목중 여성 참가자가 더 많은 종목은 4종목이다.
- 그 중 [Artistic Swimming]과 [Rhythmic Gymnastics]는 여성 참가자만 있다.

2

1

3



데이터 전처리



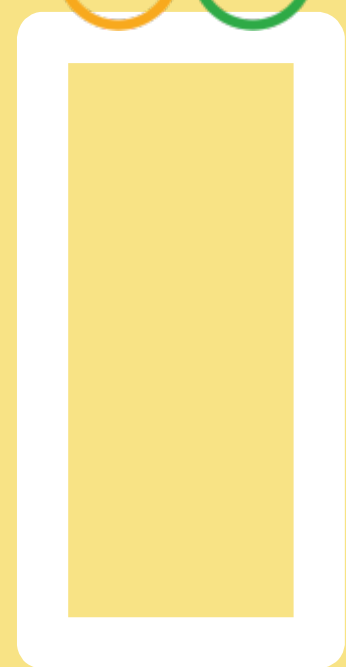
02 데이터 전처리 데이터 라벨링

[나라명 라벨링] 새로운 컬럼 추가

```
ex_x = LabelEncoder()  
medal['Team/NOC_lbl'] = ex_x.fit_transform(medal['Team/NOC'])  
medal
```

	Rank	Team/NOC	Gold	Silver	Bronze	Total	Rank by Total	Team/NOC_lbl
0	1	United States of America	39	41	33	113	1	90
1	2	People's Republic of China	38	32	18	88	2	64
2	3	Japan	27	14	17	58	5	45
3	4	Great Britain	22	21	22	65	4	33
4	5	ROC	20	28	23	71	3	70
...
88	86	Ghana	0	0	1	1	77	32
89	86	Grenada	0	0	1	1	77	35
90	86	Kuwait	0	0	1	1	77	50
91	86	Republic of Moldova	0	0	1	1	77	72
92	86	Syrian Arab Republic	0	0	1	1	77	83

- 범주형 column(나라명)을 수치형column(나라명 라벨링)으로 변경
- 두 번째 모델(정보입력시 해당 팀명 출력 모델)의 정확도를 높이기 위함



머신러닝 모델



03 머신러닝 모델 (1) 메달 개수에 따른 순위예측 모델

- 기본모델(KNN모델) 만들기

1. 데이터 나누기

```
sel = ['Gold', "Silver", "Bronze"]

X_n = medal[sel]
y_n = medal['Rank']

X_n_train, X_n_test, y_n_train, y_n_test = train_test_split(X_n, y_n,
                                                            random_state=1)
```

2. KNN 모델 학습

```
knn = KNeighborsClassifier()
knn.fit(X_n_train, y_n_train)
```

3. KNN 모델 정확도 구하기

```
knn_tr = knn.score(X_n_train, y_n_train)
knn_test = knn.score(X_n_test, y_n_test)

print("훈련 데이터셋 정확도 : {:.2f}".format(knn_tr))
print("테스트 데이터 셋 정확도 : {:.2f}".format(knn_test))
```

훈련 데이터셋 정확도 : 0.41
테스트 데이터 셋 정확도 : 0.25

2

1

3

03 머신러닝 모델 기본모델(KNN모델) 성능 개선

- k값에 따른 정확도 구하기

```
tr_knn = []
test_knn = []
k_nums = range(1,22,2)

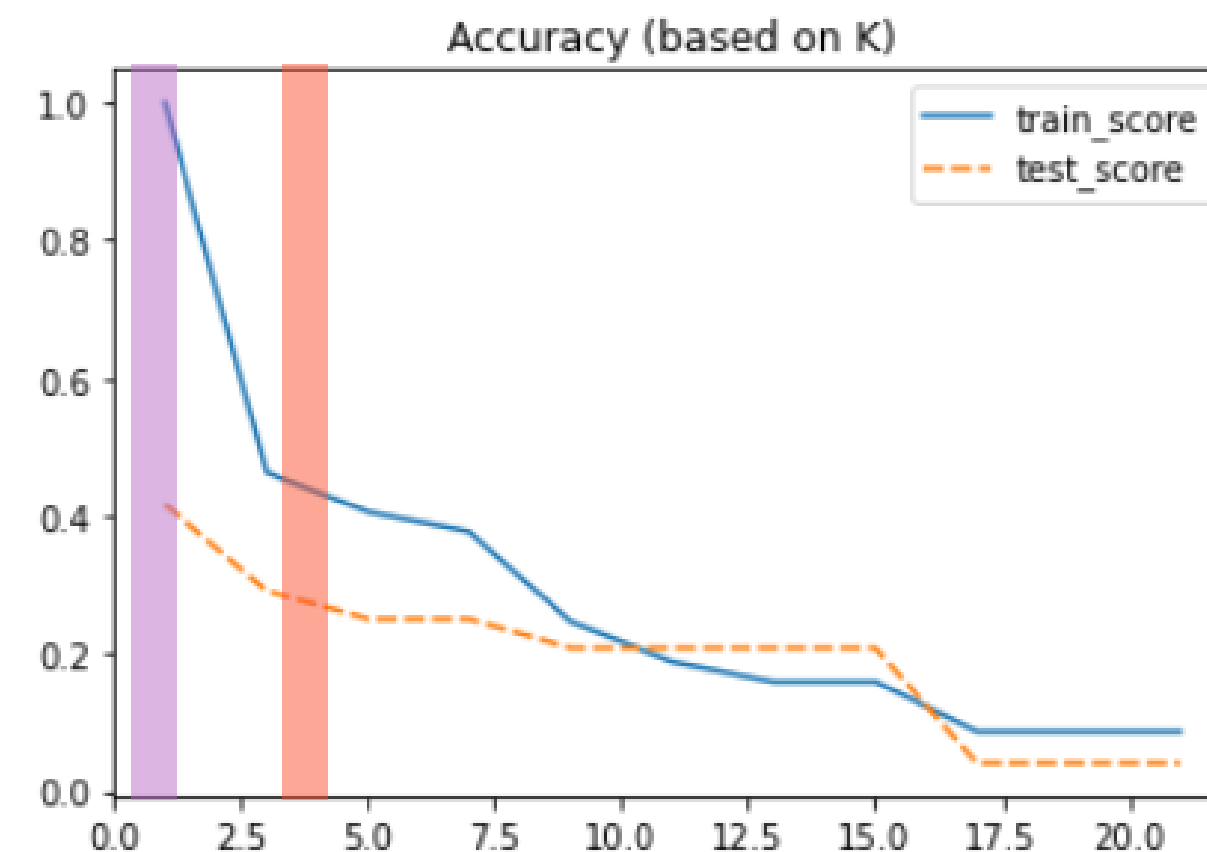
for n in k_nums:
    knn = KNeighborsClassifier(n_neighbors=n)
    knn.fit(X_n_train, y_n_train)

    knn_tr = knn.score(X_n_train, y_n_train)
    knn_test = knn.score(X_n_test, y_n_test)

    tr_knn.append(knn_tr)
    test_knn.append(knn_test)

    print("k :", n)
    print("학습용 셋 정확도 : {:.3f}".format(knn_tr))
    print("테스트용 셋 정확도 : {:.3f}".format(knn_test))
```

	train_score	test_score
1	1.000000	0.416667
3	0.463768	0.291667
5	0.405797	0.250000
7	0.376812	0.250000
9	0.246377	0.208333
11	0.188406	0.208333
13	0.159420	0.208333
15	0.159420	0.208333
17	0.086957	0.041667
19	0.086957	0.041667
21	0.086957	0.041667



train 정확도는 k=1일때 가장 높지만, test 셋과의 정확도 차이가 많이나므로 과적합됨을 알 수 있다.
따라서, k=3일 때 가장 높은 정확도를 보이므로 k=3을 채택한다.

- test_size에 따른 정확도 구하기

```
for i in range(1,6,1):
    X_n_train, X_n_test, y_n_train, y_n_test = train_test_split(X_n,y_n,
                                                                test_size=(i/10),
                                                                random_state=3)

    knn = KNeighborsClassifier(n_neighbors=1).fit(X_n_train,y_n_train)

    tr_score = knn.score(X_n_train,y_n_train)
    test_score = knn.score(X_n_test,y_n_test)
    print("학습용 : {}, 테스트용 : {}".format(10-i, i))
    print("학습용 : {:.3f}, 테스트용 : {:.3f}\n".format(tr_score, test_score))
```

```
학습용 : 9, 테스트용 : 1
학습용 : 1.000, 테스트용 : 0.500

학습용 : 8, 테스트용 : 2
학습용 : 1.000, 테스트용 : 0.526

학습용 : 7, 테스트용 : 3
학습용 : 1.000, 테스트용 : 0.500

학습용 : 6, 테스트용 : 4
학습용 : 1.000, 테스트용 : 0.395

학습용 : 5, 테스트용 : 5
학습용 : 1.000, 테스트용 : 0.362
```

- total feature 추가

```
sel_t = ['Gold', "Silver", "Bronze", "Total"]
```

```
X_t = medal[sel_t]
y_t = medal['Rank']
```

```
X_t_train, X_t_test, y_t_train,|
y_t_test=train_test_split(X_t,y_t,random_state=1)
```

```
print("훈련 데이터셋 정확도 : {:.2f}".format(knn_t_tr))
print("테스트 데이터 셋 정확도 : {:.2f}".format(knn_t_test))
```

훈련 데이터셋 정확도 : 0.41
 테스트 데이터 셋 정확도 : 0.29

test_size와 관계없이 학습용 셋의 정확도는 변화가 없으므로 테스트용 셋 정확도가 가장 높은 8:2 비율을 채택한다.
 total(전체메달수)를 feature에 추가함으로써 성능이 약간 향상됨을 알 수 있다.



머신러닝 모델 기본모델(KNN모델) 성능 개선

기본 모델 정확도
훈련 데이터셋 정확도: 0.41
테스트 데이터셋 정확도: 0.25

-가중치 값을 통한 성능개선

```
medal['Gold_1000'] = medal['Gold'] * 1000  
medal['Silver_100'] = medal['Silver'] * 100  
medal['Bronze_10'] = medal['Bronze'] * 10
```

Gold X 1000, Silver X 100, Bronze X 10
각각가중치를 두어 새로운 colmn 생성

-가중치 값 조절 후 정확도

```
knn_1000_tr = knn_1000.score(X_1000_train, y_1000_train)  
knn_1000_test = knn_1000.score(X_1000_test, y_1000_test)
```

```
print("훈련 데이터셋 정확도 : {:.2f}".format(knn_1000_tr))  
print("테스트 데이터 셋 정확도 : {:.2f}".format(knn_1000_test))
```

훈련 데이터셋 정확도 : 0.41
테스트 데이터 셋 정확도 : 0.25

가중치 값과 무관하게 이전 기본모델과 정확도가 같음.
따라서 가중치 값에 따른 성능개선이 되지 않음.

2

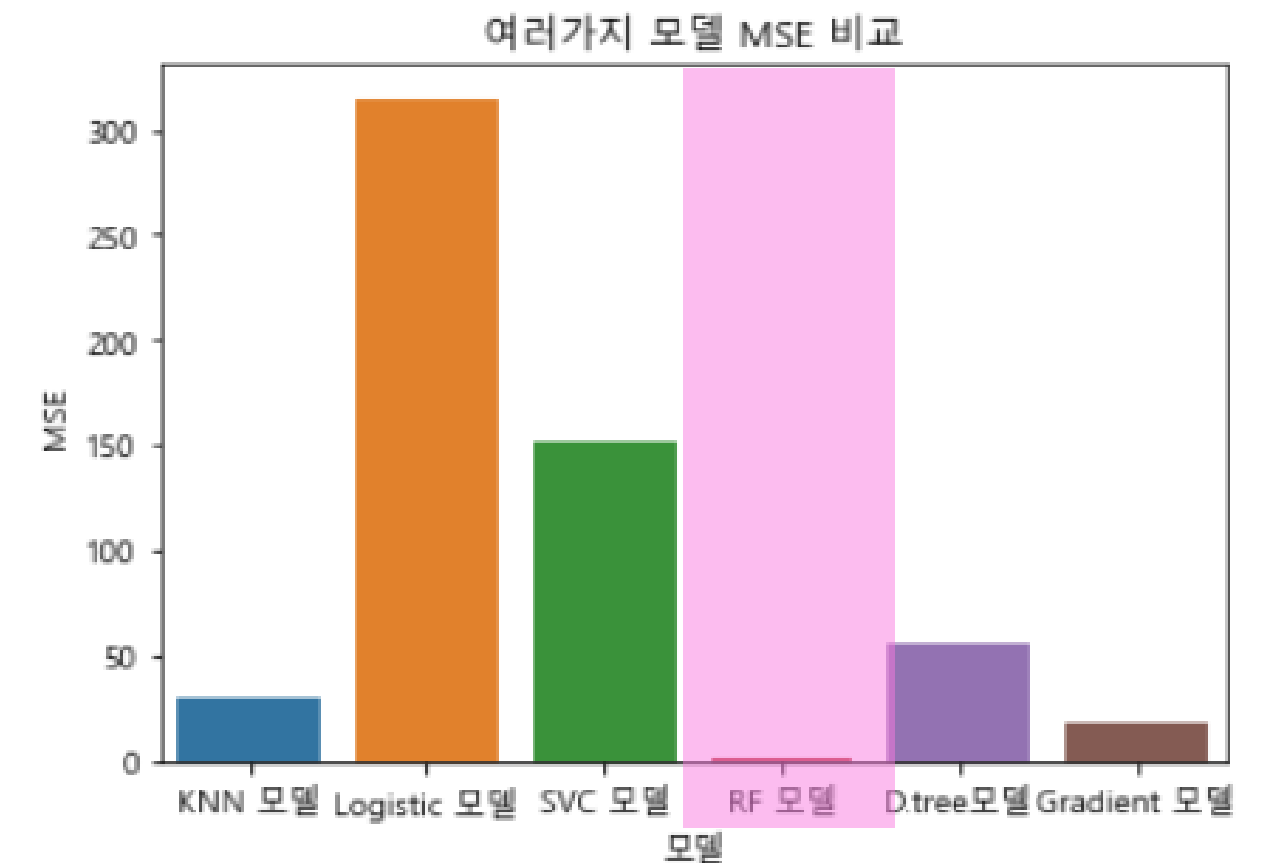
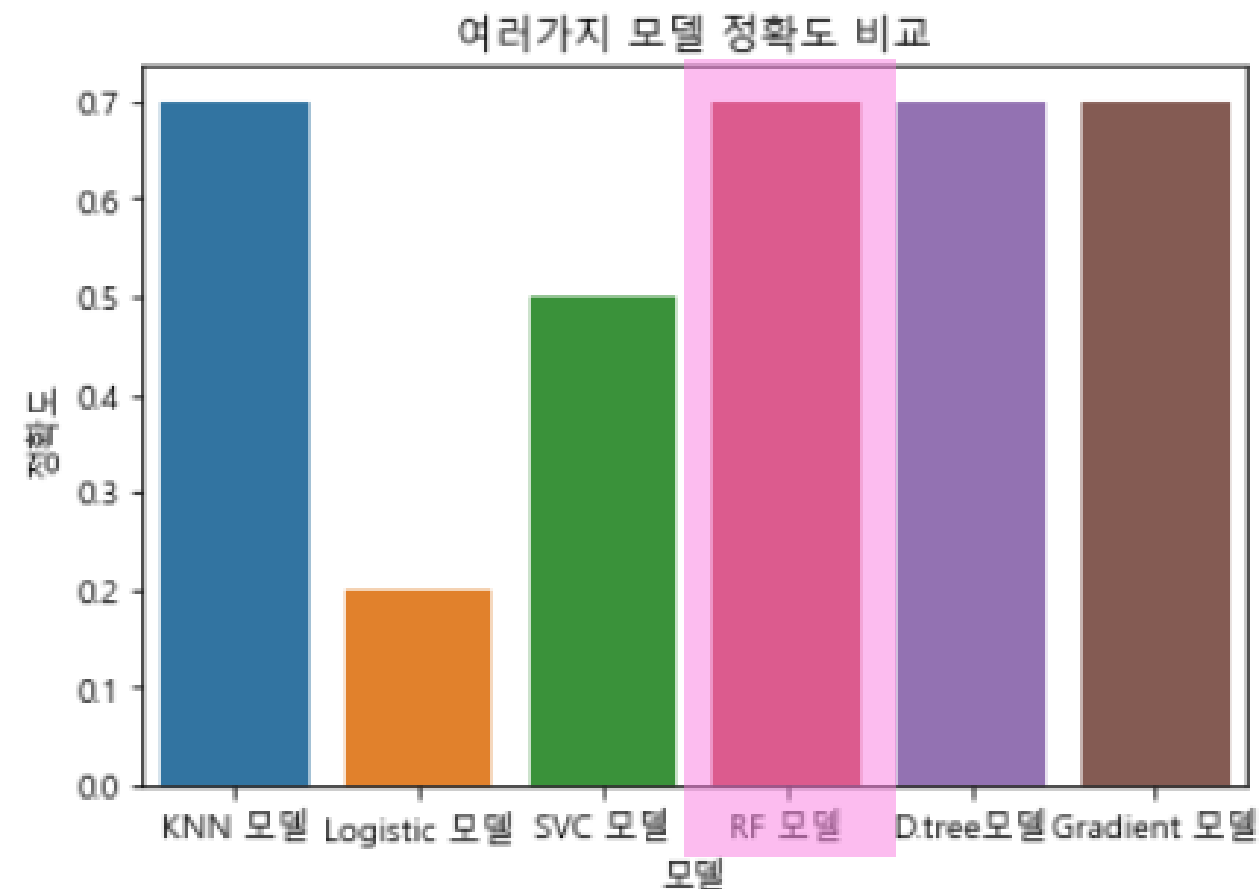
1

3

03 머신러닝 모델 다양한 모델 비교

KNeighborsClassifier, LogisticRegression, LinearSVC, RandomForestClassifier, DecisionTreeClassifier, GradientBoostingClassifier 모델

	모델	정확도	오차	MSE
0	KNN 모델	0.7	0.300000	29.7
1	Logistic 모델	0.2	0.378313	314.9
2	SVC 모델	0.5	-0.078313	151.9
3	RF 모델	0.7	0.300000	1.8
4	D.tree모델	0.7	0.300000	56.3
5	Gradient 모델	0.7	0.300000	18.2



logistic 모델과 SVC 모델을 제외한 모든 모델들의 정확도가 0.7로 가장 높았다.
MSE가 가장 낮게 나온 모델은 RandomForestClassifier 모델로 수치는 1.80이다.
따라서, 가장 최적의 모델은 RandomForestClassifier이다.

2

1

3

03 머신러닝 모델 (2) 정보입력시 해당 팀명 출력 모델

금,은,동 메달 개수입력 시 해당 나라명 출력하는 모델

1. 데이터 나누기

```
sel = ['Gold', 'Silver', 'Bronze']  
X = medal[sel]  
y = medal['Team/NOC_1b1']  
  
X_train, X_test, y_train, y_test = train_test_split(X, y,  
                                                    test_size = 0.1,  
                                                    random_state = 0)
```

2. 데이터 입력받기

```
print("금, 은, 동메달의 개수를 입력하세요")  
a, b, c = input().split()  
inp_value = [a, b, c]
```

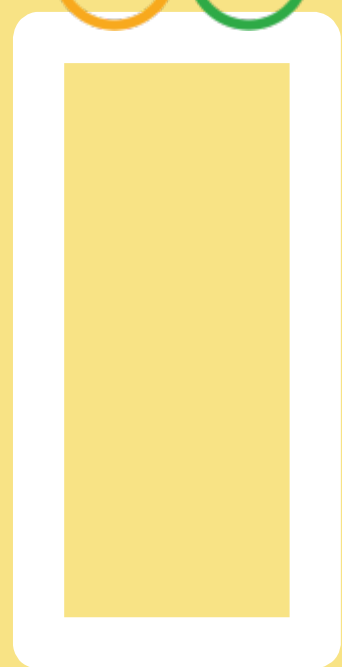
3. 데이터 출력

```
pred = model.predict([inp_value])  
pred[0]
```

4. loc를 통해 해당 출력받은 데이터와 일치하는 나라명 출력

```
sel_medal = medal.loc[medal['Team/NOC_1b1'] == pred[0], "Team/NOC"]  
print("예측 나라는 :", sel_medal.values)
```

예측 나라는 : ['Botswana']



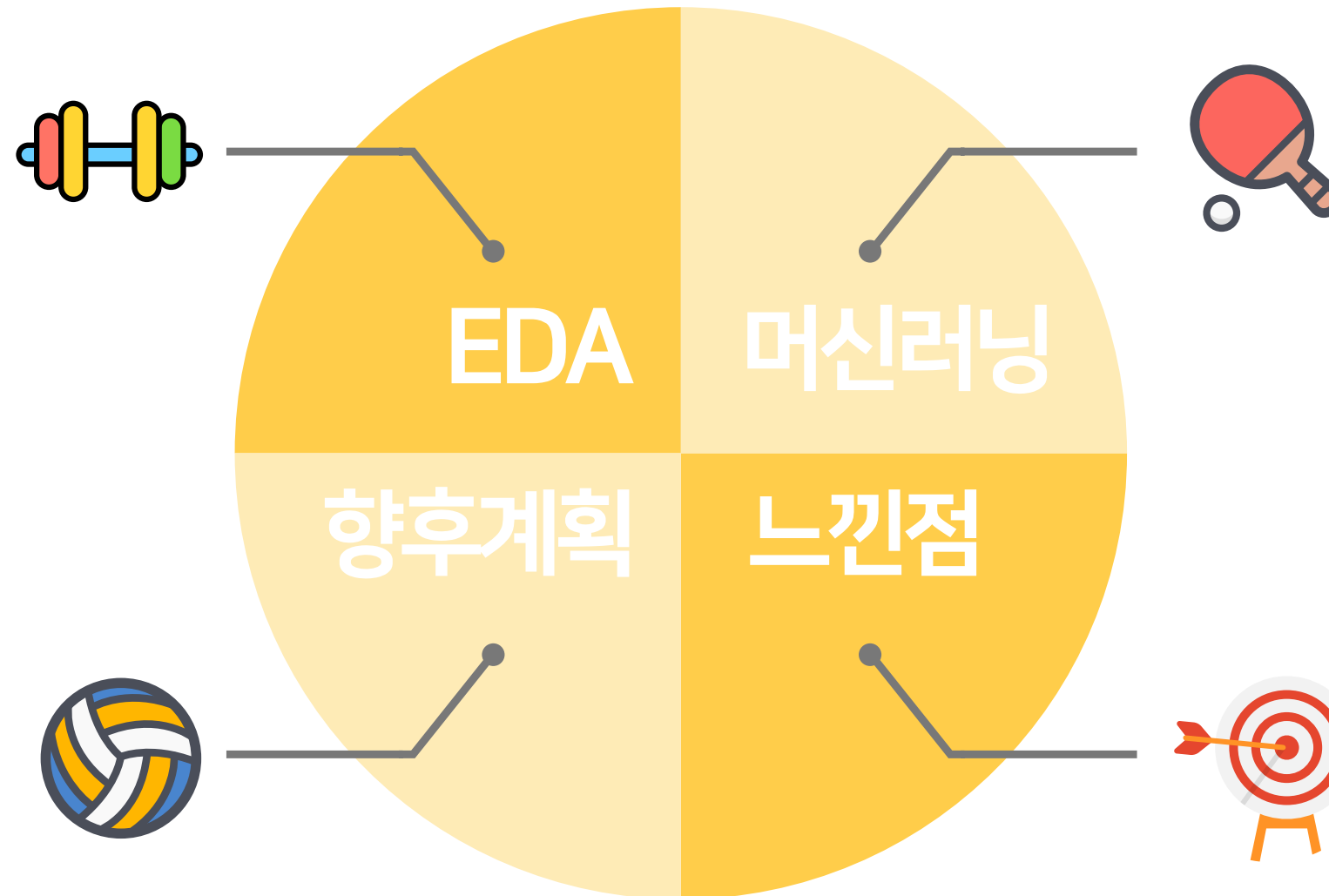
결론



04 결론

도쿄올림픽 상위 메달국가는 유럽 및 아메리카이며, 선수 성비는 남자가 약간 우세하다. 또한 여성은 올림픽 전종목에 출전하였다.

- EDA : 지도 시각화 및 가설설정하여 심도있는 분석 진행
- 머신러닝 : feature 추가를 통한 성능개선 및 추가 모델 비교



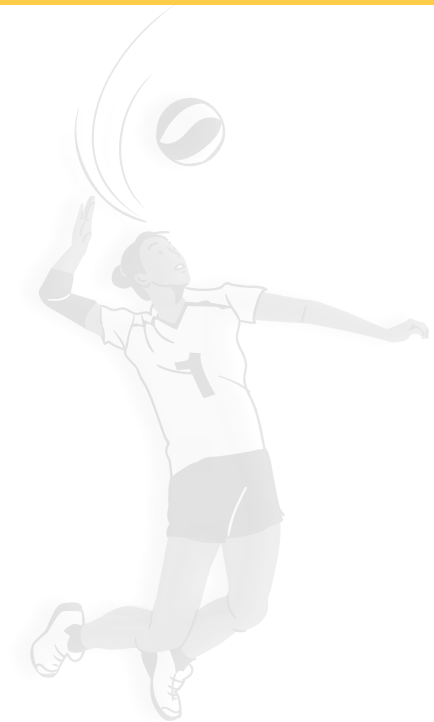
최적의 모델의 위한 K값은 3이며, 학습용과 테스트용셋의 최적 비율은 8:2이다. 대부분 모델이 정확도가 비슷하지만 R/F모델이 mse가 가장 낮아 최적모델이다.

파이썬 라이브러리를 활용하여 데이터를 분석하면서 그간 배웠던 내용을 복습할 수 있었다. 또한 머신러닝 모델을 만들고 성능개선 및 비교를 통해 ML 기본에 대해 이해할 수 있었다.

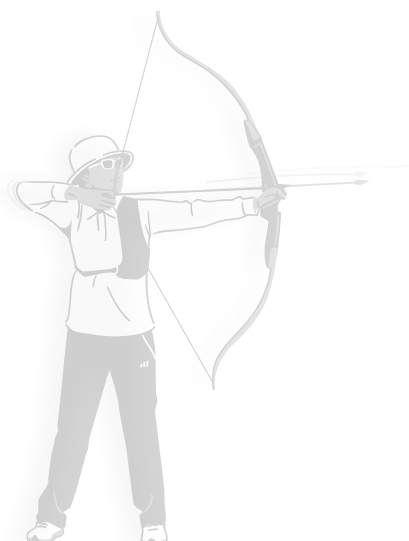
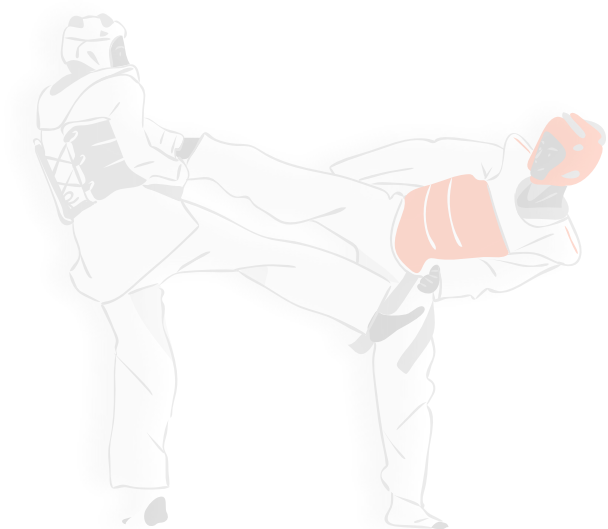
2

1

3



감사합니다



2

1

3



멋쟁이사자처럼 13회차 전 예슬