AI를 활용한 빅데이터 분석과 응용소프트웨어 개발

개체와 SNP 정보를 이용한 품종 분류 AI 모델 개발

고현서

목차
CONTENTS

개요

- 분석 배경 및 목적
- 대회 설명
- 분석 도구

개요

분석 배경 및 목적

분석 배경

데이콘 주관
‘유전체 정보 품종 분류 AI경진대회’
참가

분석 목적

개체 정보와 SNP 정보를 이용하여
A, B, C 품종을 분류
최고의 품종 구분 정확도를 획득

개요

대회설명

유전체 정보 품종 분류 AI경진대회

알고리즘 | 유전체 | 분류 | Marco F1 score

상금 : 300만원

기간 : 2022.12.12 ~ 2023.01.16 09:59

참가인원 : 총 693명 (468팀) 2022.12.27 기준

주제 : 개체와 SNP 정보를 이용하여 품종 분류 AI 모델 개발

주최 / 주관 : 충남대학교, 티엔티리써치, AI Frenz / 데이콘

대회 설명 : 시장에서 세 품종이 동시에 유통될 때, 각 품종의 고유한 생산품목(우유 및 식육)의 가치 및 가격 산정에
부정유통이 차단되기 위해 현장에서 사용 할 수 있는 AI 모델이 필요합니다.
즉, 많은 SNP 정보를 통해 분류하는 것보다, 보다 더 적은 SNP 정보로 높은 분류 성능을 내는 것이 중요합니다.
따라서 이번 경진대회에서는 개체 정보와 사전에 구성된 15개의 SNP 정보를 바탕으로 품종 분류 모델을
개발해야 합니다.

개요

대회설명

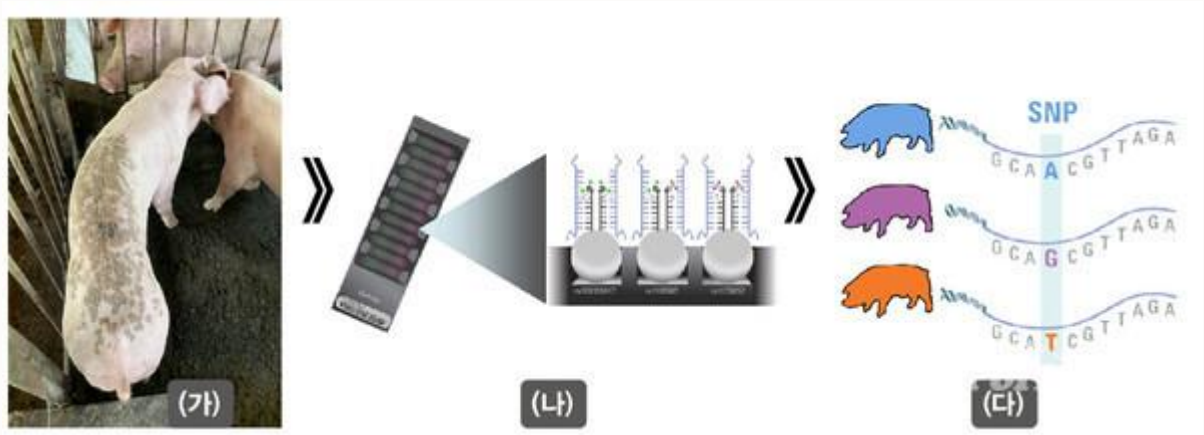
SNP 데이터를 이용한 품종 조성 분석

피그앤포크한돈 | 입력 2021.11.13 | 호수 407 | 댓글 0

- 혈액, 모근으로 돼지의 품종을 확인하는 품질관리 진행

백색돼지를 보면서 이 돼지가 순종인지, F1인지, 비육돈인지를 구분하는 것은 사실상 불가능하다. 겉모습이 모두 비슷하니 돼지고기 품종을 속여 판매하다는 게 그리 어려운 일은 아닐 것이다. 그 예시로 이베리코 돼지고기로 둔갑한 일반 수입 돼지고기, 흑돼지로 속인 백색돼지 등 소비자에게 거짓 정보를 제공하며 판매하는 사례들이 있을 수 있다. 비슷한 사례로 종돈에서 유색이 나타날 경우, 품종을 의심할 수 있기 때문에 품종 확인에 대한 필요성이 높아지고 있는 상황이다.

현재 시대의 흐름은 소비자에게 투명한 판매과정과 정확한 상품 정보를 제공해야 한다. 이에 최근 양돈산업에서도 기존 돼지의 외형적인 모습과 더불어 DNA를 활용한 돼지에 대한 품종 조성(BC, Breed composition)을 좀 더 정확하게 밝혀내는 방향으로 흘러가고 있다. 이는 돼지들의 DNA 내 A, T, G, C와 같은 염기서열의 차이(SNP)가 개체 간 어느 정도인지 파악하여 분자적인 수준에서의 명확한 품종 조성 확인을 가능케 한다. 이러한 분자적인 수준의 품종 조성 확인 방법은 국가기관인 국립축산과학원에서 염소를 이용한 품종 조성 확인 연구, 세계 유수의 연구기관에서 육우, 젖소의 품종 확인 연구가 진행되었던 만큼 신빙성 있는 방법 중 하나이다.



기사 출처 - [한돈] http://www.pignpork.com/news/articleView.html?idxno=542

개요

대회설명

유전체 정보 품종 분류 AI경진대회

알고리즘 | 유전체 | 분류 | Marco F1 score

대회 규칙

- 평가 산식 : Macro F1 Score (Public Score : 전체 테스트 데이터 중 60%/ Private Score : 전체 테스트 데이터
- 평가 방식 : 1차 평가 (리더보드 Private Score) / 2차 평가 (Private Score 상위 10개팀 코드 및 PPT 평가)
- 외부 데이터 사용 불가
- 1일 최대 제출 횟수 3회
- 사용 가능 언어 : Python, R

개요

분석 도구

언어

IDE

데이터 전처리

- 원본 데이터
- EDA
- Label-Encoding

데이터 전처리

원본 데이터

train.csv [파일]

id : 개체 고유 ID 개체정보

father : 개체의 가계 고유 번호 (0 : Unknown)

mother : 개체의 모계 고유 번호 (0 : Unknown)

gender : 개체 성별 (0 : Unknown, 1 : female, 2 : male)

trait : 개체 표현형 정보

15개의 SNP 정보 : SNP_01 ~ SNP_15

class : 개체의 품종 (A,B,C)

	id	father	mother	gender	trait	SNP_01	SNP_02	SNP_03	SNP_04	SNP_05	...	SNP_07	SNP_08	SNP_09	SNP_10	SNP_11	SNP_12	SNP_13	SNP_14	SNP_15	class
0	TRAIN_000	0	0	0	2	G G	A G	A A	G A	C A	...	A A	G G	A A	G G	A G	A A	A A	A A	A A	B
1	TRAIN_001	0	0	0	2	A G	A G	C A	A A	A A	...	A A	G A	A A	A G	A A	G A	G G	A A	A A	C
2	TRAIN_002	0	0	0	2	G G	G G	A A	G A	C C	...	A A	G A	G A	A G	A A	A A	A A	A A	A A	B
3	TRAIN_003	0	0	0	1	A A	G G	A A	G A	A A	...	G G	A A	G G	A G	G G	G G	G G	A A	G G	A
4	TRAIN_004	0	0	0	2	G G	G G	C C	A A	C C	...	A A	A A	A A	G G	A A	A A	A G	A A	G A	C
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
257	TRAIN_257	0	0	0	2	A G	A G	A A	G A	C C	...	A A	G A	A A	G G	A G	G A	A A	A A	A A	B
258	TRAIN_258	0	0	0	2	G G	A A	C A	A A	A A	...	G A	G A	A A	A G	A G	A A	A G	A A	G A	C
259	TRAIN_259	0	0	0	1	A G	G G	A A	G A	A A	...	G G	G A	G A	A A	G G	G G	G G	C A	G G	A
260	TRAIN_260	0	0	0	1	A A	G G	A A	G A	A A	...	G G	A A	G A	A G	A G	G A	G G	C A	G G	A
261	TRAIN_261	0	0	0	2	G G	A G	C A	G G	C C	...	A A	A A	A A	G G	A A	A A	G G	A A	G A	B

데이터 전처리

원본 데이터

test.csv [파일]

id : 개체 고유 ID 개체정보

father : 개체의 가계 고유 번호 (0 : Unknown)

mother : 개체의 모계 고유 번호 (0 : Unknown)

gender : 개체 성별 (0 : Unknown, 1 : female, 2 : male)

trait : 개체 표현형 정보

15개의 SNP 정보 : SNP_01 ~ SNP_15

	id	father	mother	gender	trait	SNP_01	SNP_02	SNP_03	SNP_04	SNP_05	SNP_06	SNP_07	SNP_08	SNP_09	SNP_10	SNP_11	SNP_12	SNP_13	SNP_14	SNP_15
0	TEST_000	0	0	0	1	AG	GG	AA	GA	AA	AG	GG	GA	GA	AG	AG	GA	GG	CA	GA
1	TEST_001	0	0	0	2	GG	AG	CC	GG	CC	AA	AA	AA	AA	GG	AG	AA	AA	AA	AA
2	TEST_002	0	0	0	2	GG	AG	AA	AA	CA	AG	AA	AA	AA	AG	AA	GA	GG	AA	GG
3	TEST_003	0	0	0	2	GG	AG	CA	AA	CC	AA	AA	AA	AA	GG	AA	GA	AG	AA	AA
4	TEST_004	0	0	0	1	AA	GG	AA	GG	AA	GG	GG	AA	GG	AG	GG	GA	GG	AA	GG
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
170	TEST_170	0	0	0	2	AG	GG	CC	AA	CA	AG	AA	GG	AA	GG	GG	AA	AA	AA	GA
171	TEST_171	0	0	0	2	GG	AA	AA	AA	CA	AG	AA	AA	AA	AG	AA	AA	AG	AA	GA
172	TEST_172	0	0	0	2	GG	AA	AA	AA	CA	AG	AA	AA	AA	GG	AG	AA	AG	AA	GG
173	TEST_173	0	0	0	2	AG	GG	CA	GA	CC	GG	AA	GA	AA	GG	AG	AA	AA	AA	AA
174	TEST_174	0	0	0	2	GG	GG	CC	GA	CA	AA	GA	GG	AA	GG	GG	AA	AA	AA	AA

데이터 전처리

원본 데이터

snp_info.csv [파일]

15개의 SNP 세부 정보

name : SNP 명

chrom : 염색체 정보

cm : Genetic distance

pos : 각 마커의 유전체상 위치 정보

	SNP_id	name	chrom	cm	pos
0	SNP_01	BTA-19852-no-rs	2	67.05460	42986890
1	SNP_02	ARS-USMARC-Parent-DQ647190-rs29013632	6	31.15670	13897068
2	SNP_03	ARS-BFGL-NGS-117009	6	68.28920	44649549
3	SNP_04	ARS-BFGL-NGS-60567	6	77.87490	53826064
4	SNP_05	BovineHD0600017032	6	80.50150	61779512
5	SNP_06	BovineHD0600017424	6	80.59540	63048481
6	SNP_07	Hapmap49442-BTA-111073	6	80.78000	64037334
7	SNP_08	BovineHD0600018638	6	82.68560	67510588
8	SNP_09	ARS-BFGL-NGS-37727	6	86.87400	73092782
9	SNP_10	BTB-01558306	7	62.06920	40827112
10	SNP_11	ARS-BFGL-NGS-44247	8	97.17310	92485682
11	SNP_12	Hapmap32827-BTA-146530	9	62.74630	55007839
12	SNP_13	BTB-00395482	9	63.41810	59692848
13	SNP_14	Hapmap40256-BTA-84189	9	66.81970	72822507
14	SNP_15	BovineHD1000000224	10	1.78774	814291

sample_submission.csv [파일] - 제출 양식

id : 개체 샘플 별 고유 ID

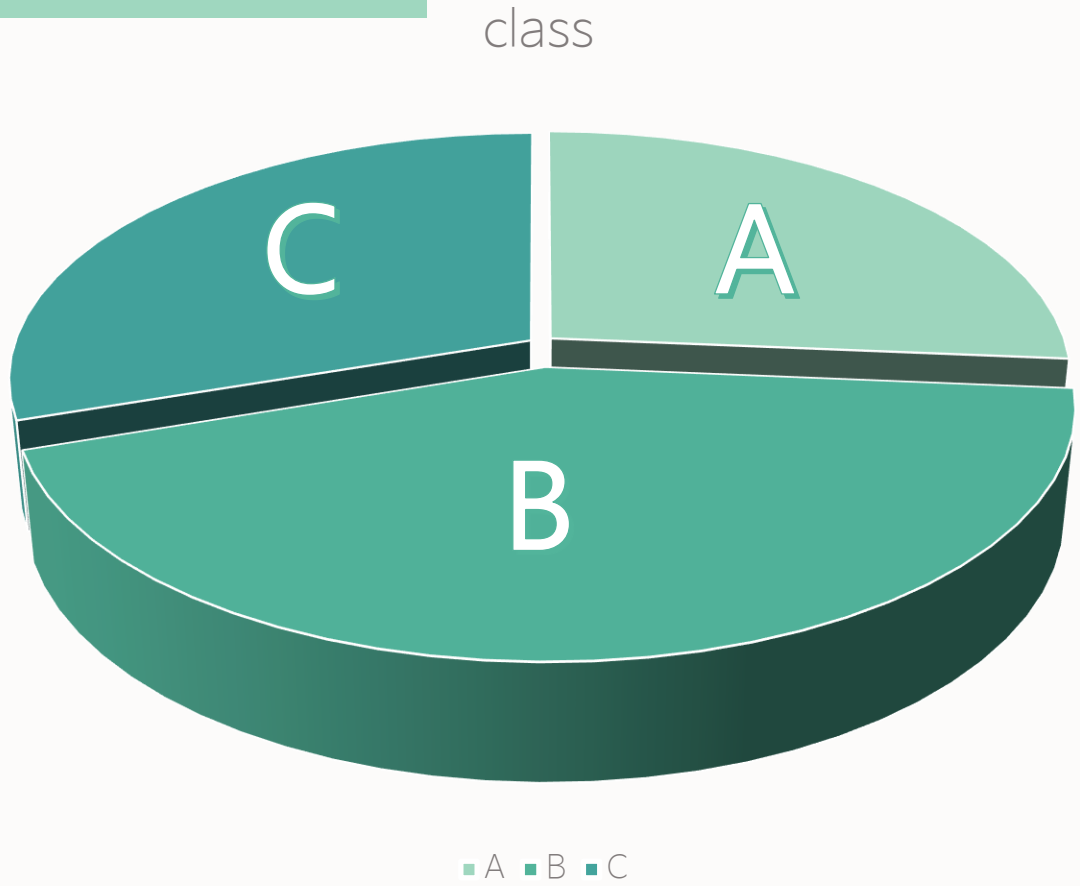
class : 예측한 개체의 품종 (A,B,C)

	id	class
0	TEST_000	A
1	TEST_001	A
2	TEST_002	A
3	TEST_003	A
4	TEST_004	A
...	...	...
170	TEST_170	A
171	TEST_171	A
172	TEST_172	A
173	TEST_173	A
174	TEST_174	A

데이터 전처리

EDA

Class 분포 파이차트



code

```
plt.subplots(figsize = (8,8))
plt.pie(train_df['class'].value_counts(), labels = train_df['class'].value_counts().index,
        autopct="%.2f%%", shadow = True, startangle = 90, explode= [0.05, 0.05, 0.05])

plt.title('class A:B:C', size=20)
plt.show()
```

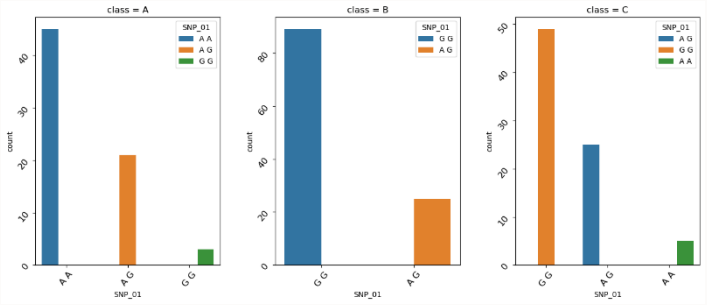
train.csv – class 레코드 빈도수

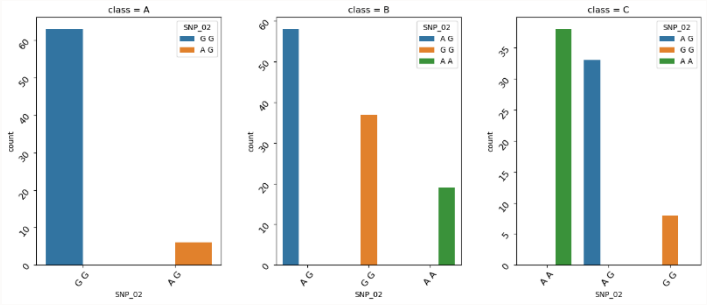
class	개수
A	69
B	114
C	79
총합	262

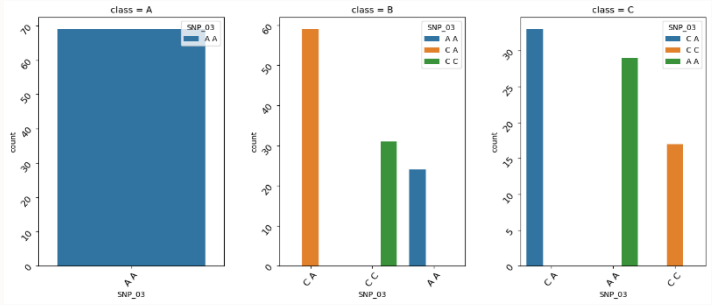
데이터 전처리

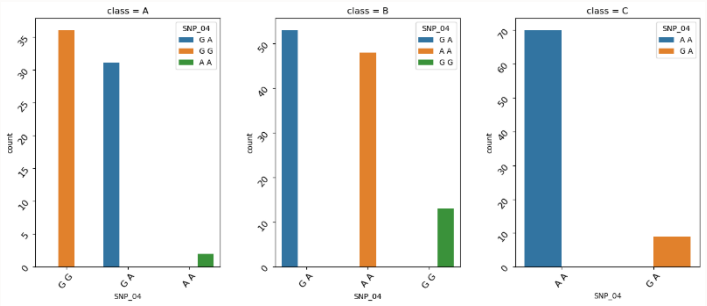
EDA

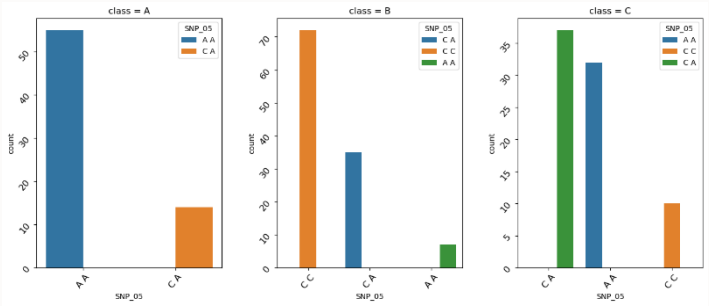
SNP정보별 품종 상관관계

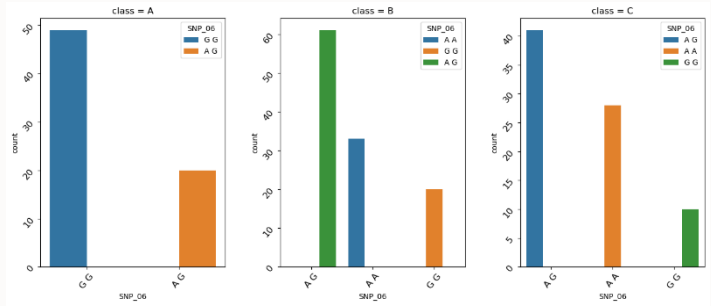


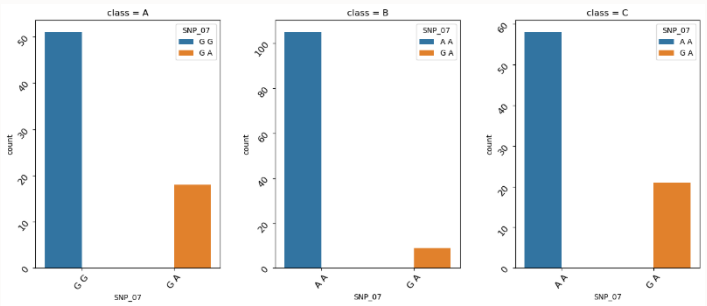


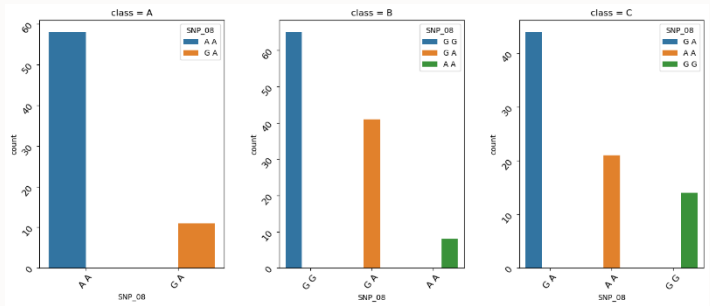


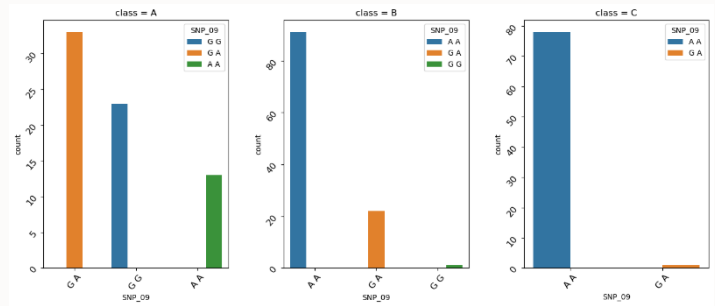








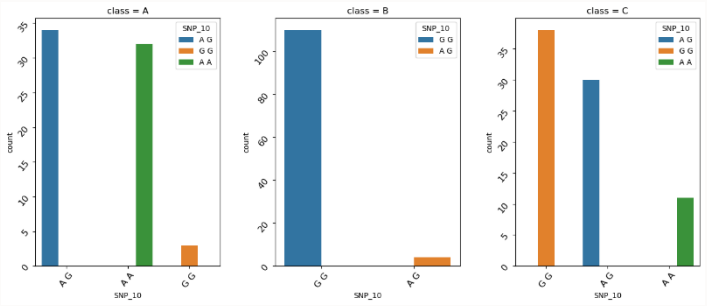


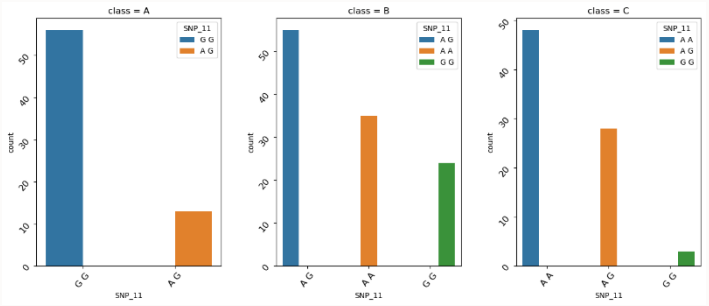


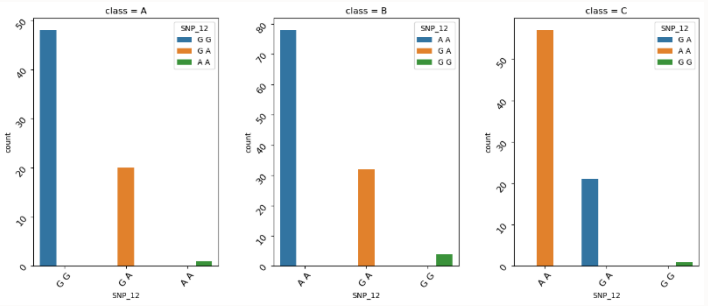
데이터 전처리

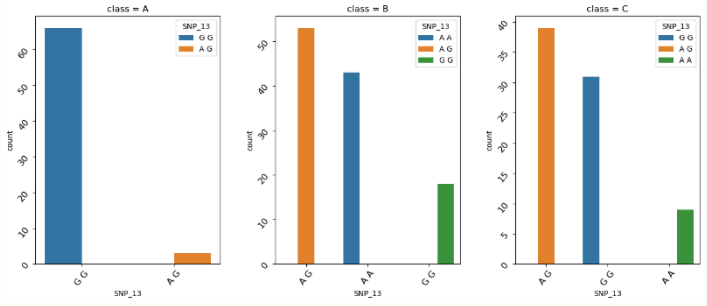
EDA

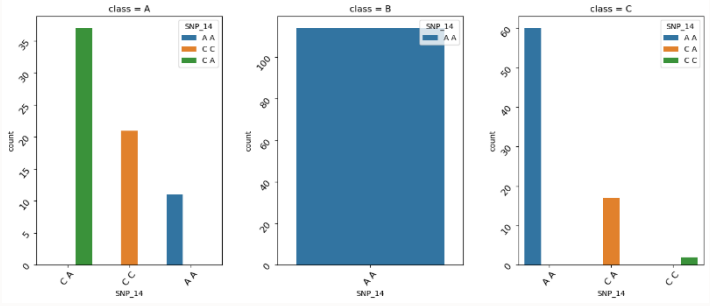
SNP정보별 품종 상관관계

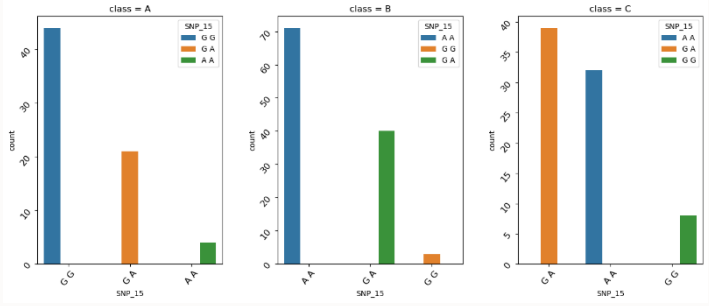












SNP_01 ~ 15 까지의 데이터 레코드 별 class 레코드 분포를 막대그래프로 시각화

데이터 전처리

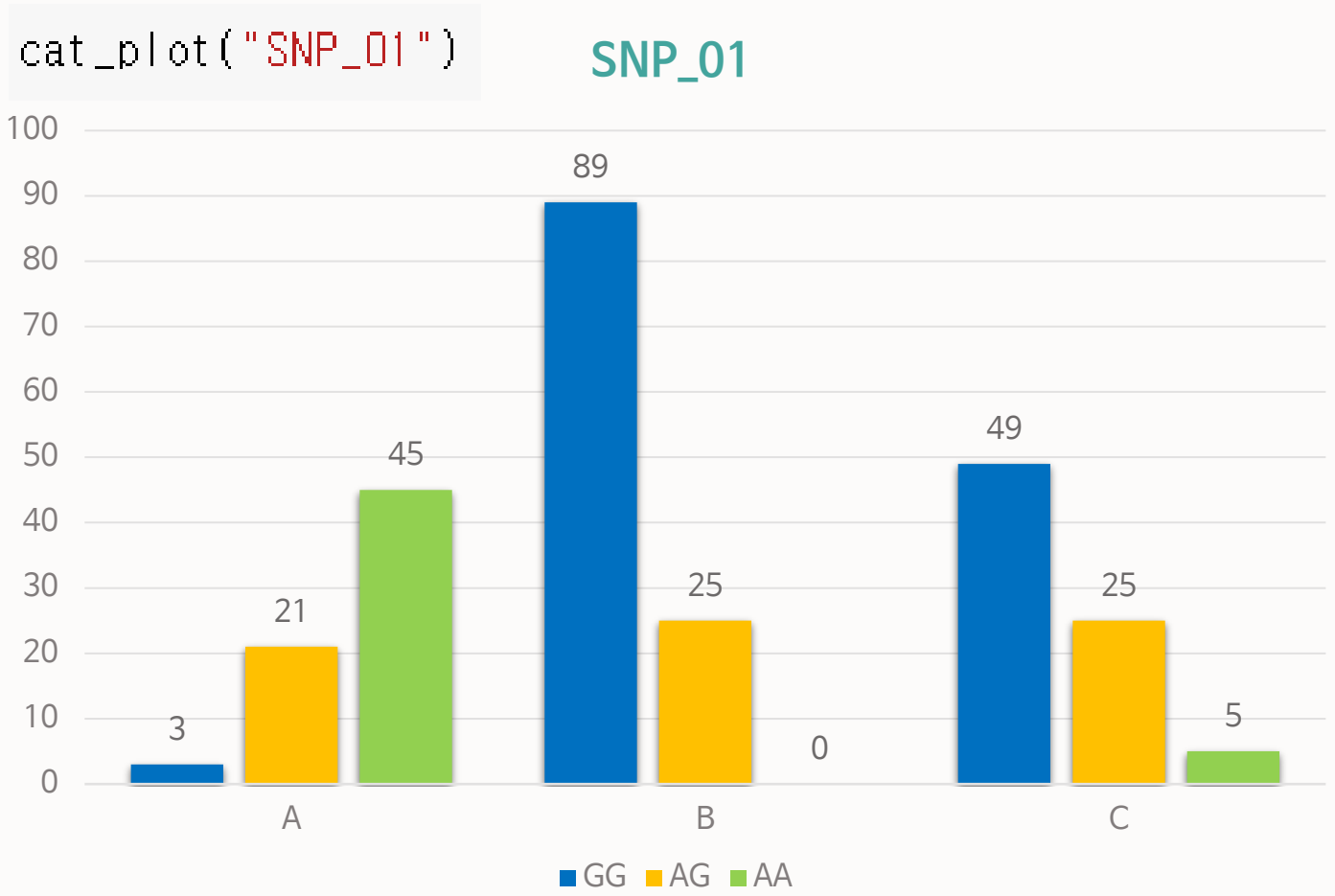
EDA

SNP정보별 품종 상관관계 code

```
# 등급에 따른 차이를 보기 위한 데이터 분류
train_A = train_df[train_df['class']== 'A']
train_B = train_df[train_df['class']== 'B']
train_C = train_df[train_df['class']== 'C']

# class 별 정리
# Categorical 그래프 함수 정의
def cat_plot(column):

  f, ax = plt.subplots(1, 3, figsize=(16, 6))


  sns.countplot(x = column,
                hue = column,
                data = train_A,
                ax = ax[0],
                order = train_A[column].value_counts().index)
  ax[0].tick_params(labelsize=12)
  ax[0].set_title('class = A')
  ax[0].set_ylabel('count')
  ax[0].tick_params(rotation=50)


  sns.countplot(x = column,
                hue = column,
                data = train_B,
                ax = ax[1],
                order = train_B[column].value_counts().index)
  ax[1].tick_params(labelsize=12)
  ax[1].set_title('class = B')
  ax[1].set_ylabel('count')
  ax[1].tick_params(rotation=50)

  sns.countplot(x = column,
                hue = column,
                data = train_C,
                ax = ax[2],
                order = train_C[column].value_counts().index)
  ax[2].tick_params(labelsize=12)
  ax[2].set_title('class = C')
  ax[2].set_ylabel('count')
  ax[2].tick_params(rotation=50)
  plt.subplots_adjust(wspace=0.3, hspace=0.3)
  plt.show()
```

예시

```
cat_plot("SNP_01")
```



* 가독성을 위해 차트 색상을 변경하였습니다.

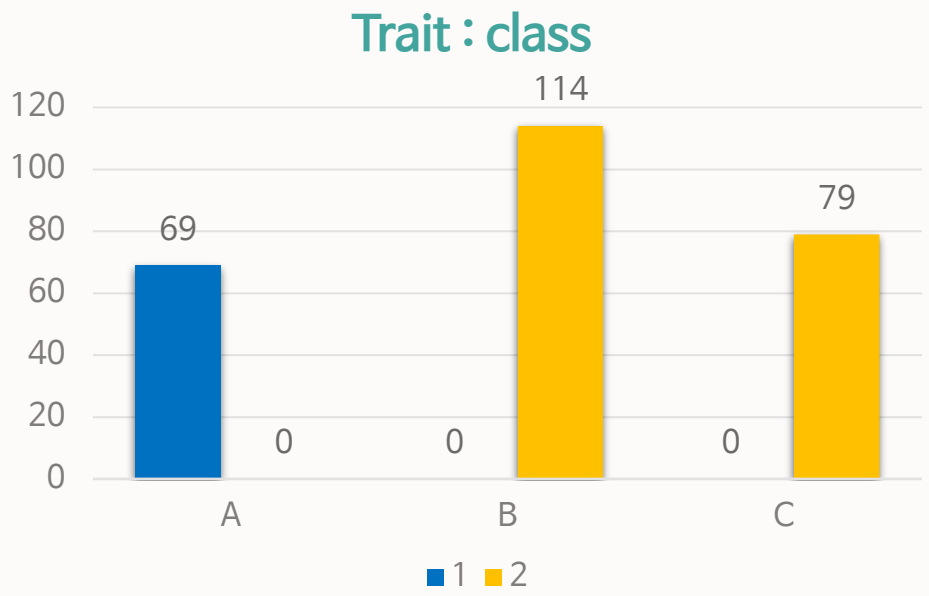
데이터 전처리

EDA

Trait(개체 표현 정보)와 class(품종)의 상관관계

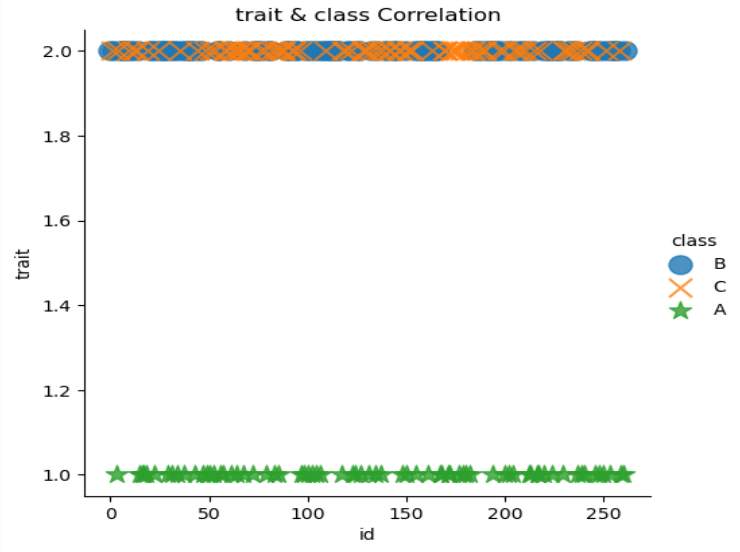
Count plot

```
cat_plot("trait")
```



lmplot

```
sns.lmplot(data = train_df, x = 'id', y = 'trait', scatter_kws = {'s':150}, markers = ['o','x','*'],
           hue = 'class', fit_reg = False)

plt.title('trait & class Correlation')
```



'trait' 이 1인 경우 'class'는 모두 A를 나타내며,
2인 경우 B,C에 분포

* 가독성을 위해 차트 색상을 변경하였습니다.

데이터 전처리

EDA

Trait(개체 표현 정보)와 class(품종)의 상관관계 시각화 code

Count plot

```
cat_plot("trait")
```

lmplot

```
sns.lmplot(data = train_df, x = 'id', y = 'trait', scatter_kws = {'s':150}, markers = ['o','x','*'],
           hue = 'class', fit_reg = False)

plt.title('trait & class Correlation')
```

데이터 전처리

Label-Encoding

Import

```
import pandas as pd
import random
import os
import numpy as np

from sklearn import preprocessing
```

시드값 고정

```
class CFG:
    SEED = 42
def seed_everything(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
seed_everything(CFG.SEED) # Seed 고정
```

데이터 불러오기

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
def get_x_y(df):
    if 'class' in df.columns:
        df_x = df.drop(columns=['id', 'class'])
        df_y = df['class']
        return df_x, df_y
    else:
        df_x = df.drop(columns=['id'])
        return df_x
train_x, train_y = get_x_y(train)
test_x = get_x_y(test)
```

train, test 변수 선언

```
train_x, train_y = get_x_y(train)
test_x = get_x_y(test)
```

데이터 전처리

Label-Encoding

Label-Encoding

```
class_le = preprocessing.LabelEncoder()
snp_le = preprocessing.LabelEncoder()
snp_col = [f'SNP_{str(x).zfill(2)}' for x in range(1,16)]
```

```
snp_data = []
for col in snp_col:
    snp_data += list(train_x[col].values)
```

```
train_y = class_le.fit_transform(train_y)
snp_le.fit(snp_data)
```

```
LabelEncoder()
```

```
for col in train_x.columns:
    if col in snp_col:
        train_x[col] = snp_le.transform(train_x[col])
        test_x[col] = snp_le.transform(test_x[col])
```

예시

train_x

trait	SNP_01	SNP_02	SNP_03	SNP_04	SNP_05	SNP_06	SNP_07	SNP_08	SNP_09	SNP_10	SNP_11	SNP_12	SNP_13	SNP_14	SNP_15
2	5	1	0	4	2	0	0	5	0	5	1	0	0	0	0
2	1	1	2	0	0	1	0	4	0	1	0	4	5	0	0
2	5	5	0	4	3	5	0	4	4	1	0	0	0	0	0
1	0	5	0	4	0	5	5	0	5	1	5	5	5	0	5
2	5	5	3	0	3	0	0	0	0	5	0	0	1	0	4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2	1	1	0	4	3	1	0	4	0	5	1	4	0	0	0
2	5	0	2	0	0	1	4	4	0	1	1	0	1	0	4
1	1	5	0	4	0	1	5	4	4	0	5	5	5	2	5
1	0	5	0	4	0	5	5	0	4	1	1	4	5	2	5
2	5	1	2	5	3	1	0	0	0	5	0	0	5	0	4

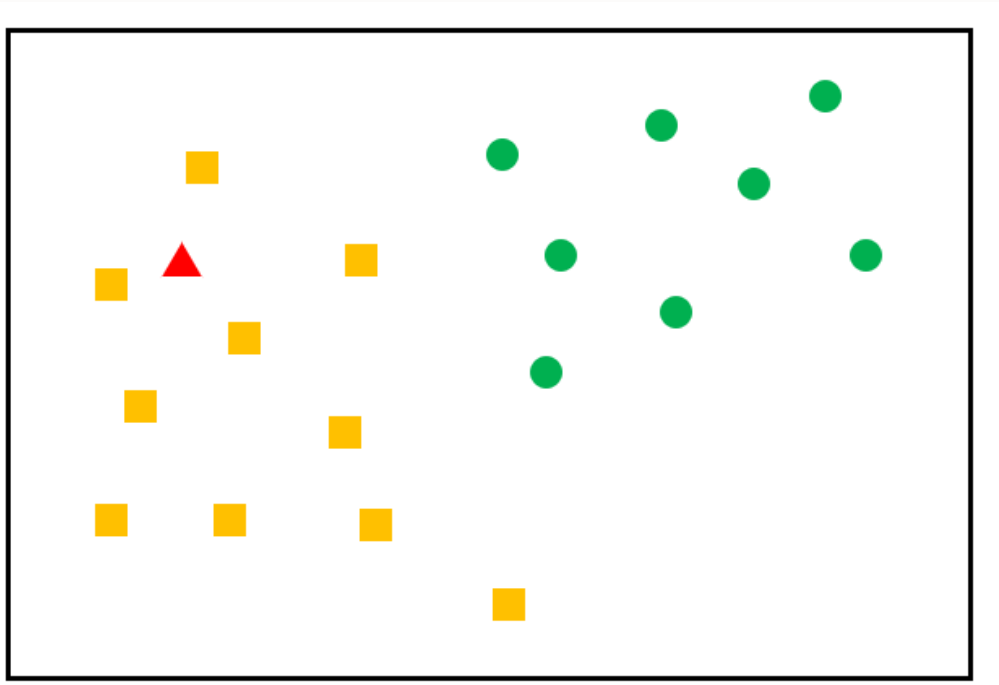
SNP_01~15의 레코드 값이 문자에서 숫자로 변환

분석 모델

- K-최근접 이웃 (kNN)
- Decision tree
- Gaussian Naïve Bayes
- Random Forest
- XGBoost
- LightGBM
- Catboost
- Deeplearning

분석 모델

k-최근접 이웃(k-Nearest Neighbor, kNN)



분류(Classification) 알고리즘, 비슷한 특성을 가진 데이터는 비슷한 범주에 속하는 경향이 있다는 가정하에 사용

주변의 가장 가까운 K개의 데이터를 보고 데이터가 속할 그룹을 판단하는 알고리즘이 K-NN 알고리즘

K의 값에 따라 분류가 달라질 수 있음

분석 모델

k-최근접 이웃(k-Nearest Neighbor, kNN)

kNN 코드 최적의 K값 찾기

```
# 최적의 k 찾기. 3부터 시작해서 max_k_range / 2 까지의 범위.
max_k_range = train.shape[0] // 2
k_list = []

for i in range(3, max_k_range, 2): # 3, 5, .... 39
    k_list.append(i)

cross_validation_scores = []
x_train = train_x[['trait','SNP_01','SNP_02','SNP_03','SNP_04','SNP_05','SNP_06','SNP_07','SNP_08','SNP_09','SNP_1
y_train = train_y
```

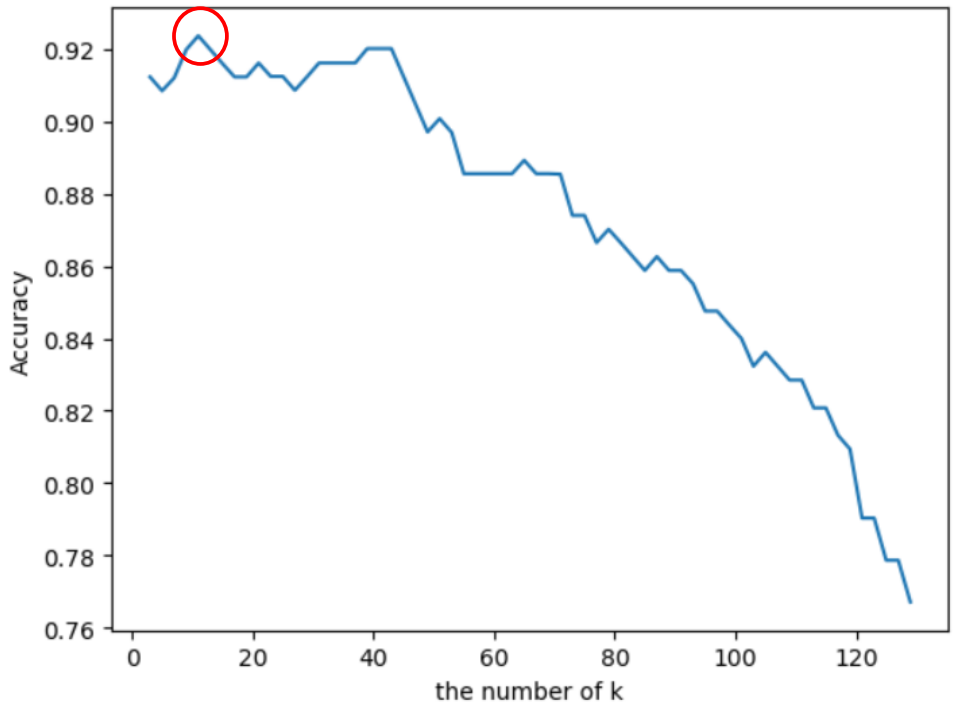
```
# 10-fold cross validation(k-fold 검증)
for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, x_train, y_train, cv=10, scoring='accuracy')
    cross_validation_scores.append(scores.mean())

cross_validation_scores
```

```
[0.9123931623931624,
 0.9085470085470085,
 0.9121082621082621,
 0.91994301994302,
 0.9237891737891738,
 0.91994301994302,
 0.9160968660968661,
 0.9123931623931624,
 0.9123931623931624,
 0.9162393162393163,
 0.9125356125356126,
 0.9125356125356126,
 0.9086894586894587,
 0.9123931623931624,
 0.9162393162393162,
 0.9162393162393162,
```

```
# 최적의 K 값
cvs = cross_validation_scores
k = k_list[cvs.index(max(cross_validation_scores))]
print("최적의 k 값 : " + str(k))
```

최적의 k 값 : 11

```
# 시각화
plt.plot(k_list, cross_validation_scores)
plt.xlabel('the number of k')
plt.ylabel('Accuracy')
plt.show()
```



분석 모델

k-최근접 이웃(k-Nearest Neighbor, kNN)

kNN 코드

K = 11일 때,

```
k = 11
clf = KNeighborsClassifier(n_neighbors=k).fit(train_x, train_y)
print(clf)
```

```
KNeighborsClassifier(n_neighbors=11)
```

```
yhat = clf.predict(test_x)
print("Train set Accuracy :", metrics.accuracy_score(train_y, clf.predict(train_x)))
```

```
Train set Accuracy : 0.9351145038167938
```

```
print(yhat)
```

```
[0 1 2 2 0 1 2 1 0 0 2 1 1 0 1 1 0 1 1 2 1 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 2
 1 1 2 0 1 2 1 1 1 1 2 1 2 0 1 0 1 1 1 2 0 1 2 0 1 2 2 2 0 1 0 0 1 1 1 0 0
 2 1 2 1 1 1 2 1 0 1 1 1 1 1 2 0 1 1 2 1 1 2 0 1 0 2 0 1 1 2 0 0 2 1 0 1 2
 1 1 1 1 0 0 2 1 2 0 1 1 2 2 1 1 1 0 1 0 0 1 1 1 2 0 0 1 0 0 0 2 1 1 2 0 1
 2 0 0 1 0 1 1 0 0 1 2 0 1 2 2 1 0 0 2 1 1 0 1 2 1 1 1]
```

```
submit = pd.read_csv('open/sample_submission.csv')
```

```
submit['class'] = class_le.inverse_transform(yhat)
```

```
submit.to_csv('./answer/kNN_submit.csv', index=False)
```

kNN_submit.scv

id	class
TEST_000	A
TEST_001	B
TEST_002	C
TEST_003	C
TEST_004	A
TEST_005	B
TEST_006	C
TEST_007	B
TEST_008	A
•	
•	
•	
TEST_172	B
TEST_173	B
TEST_174	B

분석 모델

k-최근접 이웃(k-Nearest Neighbor, kNN)

Public Score 결과 K = 11일 때,

제출 번호	제출 파일	제출일자	점수	최종제출여부
783310	kNN_submit.csv edit	2022-12-27 18:40:06	0.9622367466	☐

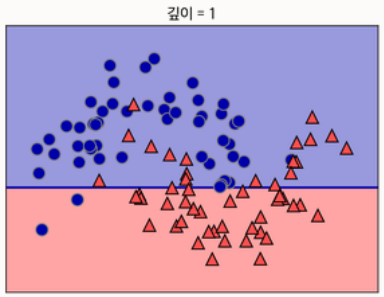
kNN(k-최근접 이웃)모델로

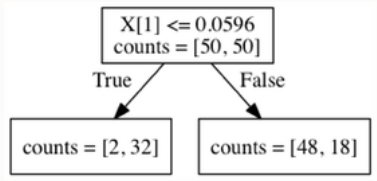
K = 11 일 때 분석한 결과

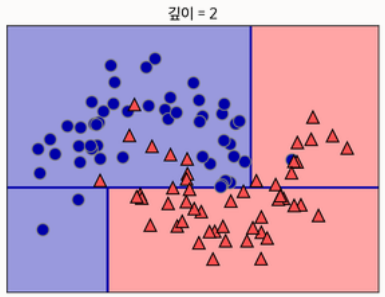
0.9351145038167938 정확도(train set 기준)에

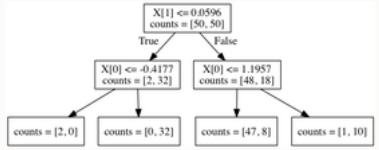
0.9622367466의 결과 도출

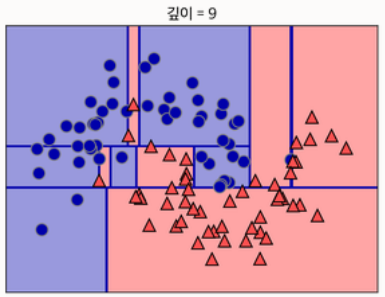
분석 모델
Decision Tree











분류(Classification) 알고리즘, 데이터를 분석하여 이들 사이에 존재하는 패턴을 예측 가능한 규칙들의 조합으로 나타내며,

그 모양이 '나무'와 유사하여 '의사결정 나무 ' 라 지칭

문을 던져서 대상을 좁혀나가는 '스무고개' 놀이와 비슷한 개념

depth가 너무 깊을 시 학습데이터에만 최적화 된 '과적합' 우려

분석 모델

Decision Tree

Decision Tree 코드

```
clf = tree.DecisionTreeClassifier(random_state = 42)
clf.fit(x_train, y_train)
```

```
DecisionTreeClassifier(random_state=42)
```

```
preds = clf.predict(x_test)
print('done')
```

```
done
```

```
preds
```

```
array([0, 1, 2, 1, 0, 1, 2, 2, 0, 0, 2, 1, 1, 0, 1, 1, 0, 1, 1, 2, 1, 1,
       2, 0, 1, 2, 1, 0, 0, 1, 0, 0, 1, 2, 0, 2, 2, 1, 1, 2, 0, 1, 2, 1,
       1, 1, 1, 2, 1, 2, 0, 1, 0, 1, 1, 1, 2, 0, 1, 2, 0, 1, 2, 2, 2, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 2, 1, 2, 1, 1, 1, 2, 1, 0, 1, 1, 1, 1, 1,
       2, 0, 1, 1, 2, 1, 1, 2, 0, 1, 0, 2, 0, 1, 1, 2, 0, 0, 2, 1, 0, 1,
       2, 1, 1, 1, 1, 0, 0, 2, 1, 1, 0, 1, 1, 2, 2, 1, 2, 1, 0, 1, 0, 0,
       1, 1, 1, 2, 0, 0, 1, 0, 0, 0, 2, 1, 1, 2, 0, 1, 2, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 2, 0, 1, 2, 2, 1, 0, 0, 2, 1, 1, 0, 1, 2, 2, 1, 1])
```

```
submit = pd.read_csv('./sample_submission.csv')
```

```
submit['class'] = class_le.inverse_transform(preds)
```

```
submit.to_csv('./answer/Decisiontree_submit.csv', index=False)
```

Decisiontree_submit.scv

id	class
TEST_000	A
TEST_001	B
TEST_002	C
TEST_003	B
TEST_004	A
TEST_005	B
TEST_006	C
TEST_007	C
TEST_008	A

•

•

•

TEST_172	C
TEST_173	B
TEST_174	B

분석 모델

Decision Tree

Public Score 결과

제출 번호	제출 파일	제출일자	점수	최종제출여부
779250	Decisiontree_submit.csv 의사결정트리 edit	2022-12-20 13:12:59	0.9354953039	☐

DecisionTree(의사결정트리)모델로 분석한 결과

1.0 만점에 0.93549539039의 결과 도출

분석 모델

Gaussian Naïve Bayes

➢ 베이즈 이론

$$P(A \mid B) = \frac{P(B|A)\,P(A)}{P(B)}$$

- P(A | B) : 어떤 사건 B가 일어났을 때 사건 A가 일어날 확률.
- P(B | A) : 어떤 사건 A가 일어났을 때 사건 B가 일어날 확률.
- P(A) : 어떤 사건 A가 일어날 확률.

➢ 조건부 확률

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

➢ 치킨 집에 저녁 성인 손님이 주문을 할 때 맥주를 주문할지 안 할지 예측.

➢ 저녁에 성인 손님이 한 명 와서 주문 시 나이브 베이즈 공식에 따르면 손님이 맥주를 주문할 확률.

- P(주문|저녁, 성인)

  = P(저녁, 성인|주문) * P(주문) / P(저녁, 성인)

➢ 결합 확률 : 두 가지 이상의 사건이 동시에 발생하는 확률.

- P(A, B) = P(A|B) P(B)

시간	성인여부	맥주
오전	성인	주문 안 함
오전	미성년자	주문 안 함
점심	성인	주문함
점심	미성년자	주문 안 함
점심	미성년자	주문 안 함
저녁	성인	주문함
저녁	성인	주문함
저녁	성인	주문함
저녁	성인	주문 안 함
저녁	미성년자	주문 안 함

- 기존 손님들의 주문 내역

분류(Classification) 알고리즘, 모든 데이터를 독립적인 사건으로 가정

특징들의 값들이 정규 분포(가우시안 분포)돼 있다는 가정하에 조건부 확률을 계산

문서 분류 및 스팸 메일 분류에 강한 면모

독립적인 사건이라고 가정하기에 다른 분류모델에 제약

분석 모델

Gaussian Naïve Bayes

Gaussian Naïve Bayes 코드

Model Fit

```
# 학습데이터로 clf를 학습합니다.
clf = GaussianNB()
clf.fit(train_x, train_y)
```

```
GaussianNB()
```

Inference

```
# 테스트 데이터로 clf를 테스트합니다.
preds = clf.predict(test_x) # 예측치
print('done')
```

```
done
```

```
preds
```

```
array([0, 1, 2, 1, 0, 1, 2, 1, 0, 0, 2, 1, 1, 0, 1, 1, 0, 1, 1, 2, 1, 1,
       1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 2, 0, 1, 2, 1, 1, 2, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 2, 0, 1, 2, 0, 1, 2, 2, 1, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 2, 1, 2, 1, 1, 1, 2, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 2, 0, 1, 0, 2, 0, 1, 1, 1, 0, 0, 2, 1, 0, 1,
       2, 1, 1, 1, 1, 0, 0, 2, 1, 1, 0, 1, 1, 2, 2, 1, 2, 1, 0, 1, 0, 0,
       1, 1, 1, 2, 0, 0, 1, 0, 0, 0, 2, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 2, 1, 1, 0, 0, 2, 1, 1, 0, 1, 2, 1, 1, 1])
```

Submission

```
submit = pd.read_csv('open/sample_submission.csv')
```

```
submit['class'] = class_le.inverse_transform(preds)
```

```
submit.to_csv('./answer/Gaussian.csv', index=False)
```

Gaussian.scv

id	class
TEST_000	A
TEST_001	B
TEST_002	C
TEST_003	B
TEST_004	A
TEST_005	B
TEST_006	C
TEST_007	B
TEST_008	A
TEST_009	A
TEST_010	C

•

•

TEST_171	C
TEST_172	B
TEST_173	B
TEST_174	B

분석 모델

Gaussian Naïve Bayes

Public Score 결과

제출 번호	제출 파일	제출일자	점수	최종제출여부
779240	submit.csv edit	2022-12-20 12:47:19	0.8854854855	☐

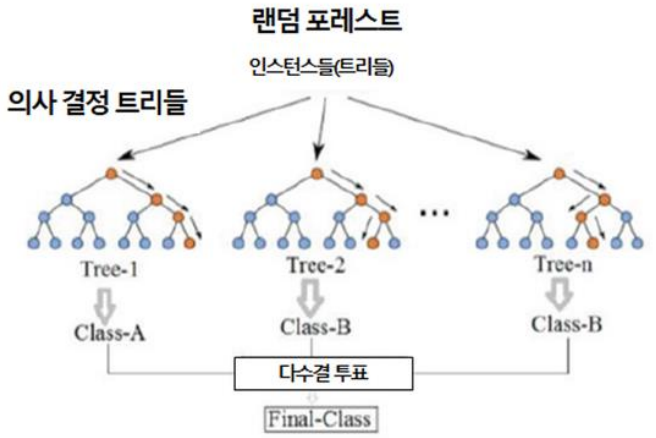
Gaussian Naïve Bayes(가우시안 나이브베이즈)모델로 분석한 결과

1.0 만점에 0.88545848555의 결과 도출

분석 모델

랜덤 포레스트(RandomForest)

- 배깅(bagging)
  - 한 가지 분류 모델을 여러 개 만들어서 서로 다른 학습 데이터로 학습시킨 후(부트스트랩), 동일한 테스트 데이터에 대한 서로 다른 예측값들을 투표를 통해(aggregating) 가장 높은 예측값으로 최종 결론을 내리는 앙상블 기법.
  - 어원 : 부트스트랩(bootstrap) + 어그리게이팅(aggregating)
  - 과대적합이 쉬운 모델에 상당히 적합한 앙상블.



Random Forest의 대표적인 파라미터

- n_estimators (int) : 내부에서 생성할 결정 트리의 개수
- crierion (str) : 정보량 계산 시 사용할 수식 (분류 모델 : gini / entropy, 회귀 모델 : mse / rmse)
- max_depth (int) : 생성할 트리의 높이
- min_samples_split (int) : 분기를 수행하는 최소한의 데이터 수
- max_leaf_nodes (int) : 리프 노드에서 가지고 있을 수 있는 최대 데이터 수
- random_state (int) : 내부적으로 사용되는 난수값
- n_jobs (int) : 병렬처리에 사용할 CPU 수
- class_weight (dict) : 학습 시 클래스의 비율에 맞춰 손실값에 가중치를 부여 (분류 모델에서만 쓰임)

배깅을 통해 여러가지 의사결정나무를 예측하는 모델

의사결정 트리의 과대적합을 보완

조정해야 할 파라미터가 많음

분석 모델

랜덤 포레스트(RandomForest)

RandomForest 코드

```
# 2nd

# 하이퍼파리미터 값을 먼저 임의로 설정하여 모델 생성 후 학습하고 GridSearchCV를 통해 최적의 하이퍼 파라미터 값 구하기.

params = { 'n_estimators' : [300, 400, 500],
           'max_depth' : [4, 5, 6, 7, 8],
           'min_samples_leaf' : [1, 2, 3],
           'min_samples_split' : [2, 3, 4]
            }

# RandomForestClassifier 객체 생성 후 GridSearchCV 수행
rf_clf = RandomForestClassifier(random_state = 42, n_jobs = -1)
grid_cv = GridSearchCV(rf_clf, param_grid = params, cv = 3, n_jobs = -1)
grid_cv.fit(train_x, train_y)

print('최적 하이퍼 파라미터: ', grid_cv.best_params_)
print('최고 예측 정확도: {:.4f}'.format(grid_cv.best_score_))
```

```
최적 하이퍼 파라미터:  {'max_depth': 8, 'min_samples_leaf': 1, 'min_samples_split': 4, 'n_estimators': 300}
최고 예측 정확도: 0.9619
```

```
#위의 결과로 나온 최적 하이퍼 파라미터로 다시 모델을 학습하여 테스트 세트 데이터에서 예측 성능을 측정
rf_clf1 = RandomForestClassifier(n_estimators = 300,
                                 max_depth = 8,
                                 min_samples_leaf = 1,
                                 min_samples_split = 4,
                                 random_state = 42,
                                 n_jobs = -1)
rf_clf1.fit(train_x, train_y)
pred = rf_clf1.predict(test_x)
```

```
print(pred)
```

```
[0 1 2 1 0 1 2 1 0 0 2 1 1 0 1 1 0 1 1 2 1 1 1 0 1 1 1 0 0 1 0 0 1 2 0 1 2
 1 1 2 0 1 2 1 1 1 1 2 1 2 0 1 0 1 1 1 2 0 1 2 0 1 2 2 2 0 1 0 0 1 1 1 0 0
 2 1 2 1 1 1 2 1 0 1 1 1 1 1 2 0 1 1 2 1 1 2 0 1 0 2 0 1 1 2 0 0 2 1 0 1 2
 1 1 1 1 0 0 2 1 1 0 1 1 2 2 1
 2 0 0 1 0 1 1 0 0 1 2 0 1 2 2
```

```
submit = pd.read_csv('open/sample_submission.csv')
```

```
submit['class'] = class_le.inverse_transform(pred)
```

```
submit.to_csv('./answer/RF_submit.csv', index=False)
```

RF_submit.scv

id	class
TEST_000	A
TEST_001	B
TEST_002	C
TEST_003	B
TEST_004	A
TEST_005	B
TEST_006	C
TEST_007	B
TEST_008	A
•	
•	
•	
TEST_172	C
TEST_173	B
TEST_174	B

분석 모델

랜덤 포레스트(RandomForest)

Public Score 결과

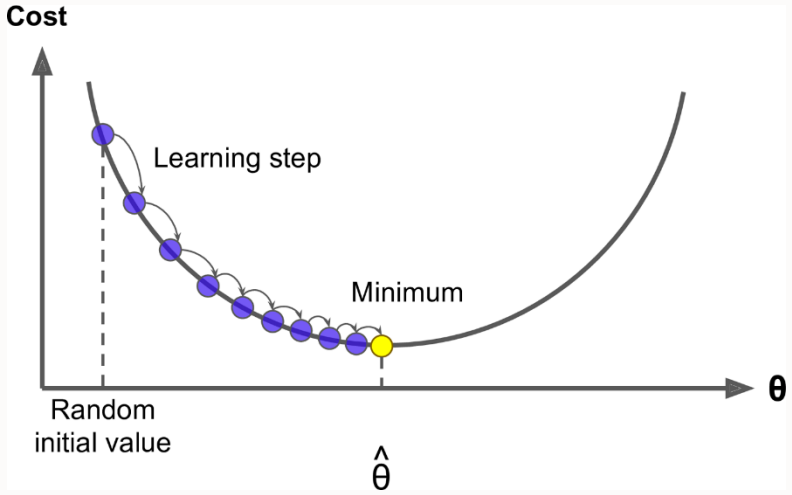
제출 번호	제출 파일	제출일자	점수	최종제출여부
783193	RF_submit.csv 2nd - RF edit	2022-12-27 16:25:42	0.9622367466	☐

RandomForest(랜덤 포레스트) 모델로 분석한 결과

0.9619 정확도(train set 기준)에

0.9622367466의 결과 도출

분석 모델
XGBOOST



과적합 방지가 가능한 규제가 포함

CART (Classification And Regression Tree)를 기반으로 하므로 분류와 회귀 둘 다 가능

앙상블 부스팅의 특징인 가중치 부여를 경사하강법 (gradient descent) 선정

분석 모델

XGBOOST

XGBOOST

```
# optuna 적용
import sklearn

X_train, X_test, y_train, y_test = train_test_split(train_x, train_y, test_size=0.33, random_state=42)
X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train, test_size=0.25, random_state=42)

def objective(trial):
  param = {
      "objective":"multi:softprob",
      "eval_metric":"error",
      "tree_method":"exact","gpu_id":0,
      "booster":"gbtree",
      "verbosity":0,
      "num_class":3,
      "max_depth":trial.suggest_int("max_depth", 1, 10),
      "learning_rate":trial.suggest_uniform("learning_rate",0.0001,0.99),
      "n_estimators":trial.suggest_int("n_estimators",1000,10000,step=100),
      "colsample_bytree":trial.suggest_float("colsample_bytree",0.5, 1.0),
      "colsample_bylevel":trial.suggest_float("colsample_level",0.5, 1.0),
      "colsample_bynode":trial.suggest_float("colsample_bynode",0.5, 1.0),
      "reg_lambda":trial.suggest_uniform("reg_lambda",1e-2, 1),
      "reg_alpha":trial.suggest_uniform("reg_alpha",1e-2, 1),
    "subsample":trial.suggest_discrete_uniform("subsample",0.6, 1.0, 0.05),
      "min_child_weight":trial.suggest_int("min_child_weight",2,15),
    "gamma":trial.suggest_float("gamma",0.0001,1.0,log=True)
}
 model = XGBClassifier(**param, random_state=42)
 bst = model.fit(X_train, y_train)
 preds = bst.predict(X_validation)
 pred_lables = np.rint(preds)
  accuracy = sklearn.metrics.accuracy_score(y_validation, pred_lables)
  return accuracy
```

XGBOOST_submit.scv

id	class
TEST_000	A
TEST_001	B
TEST_002	C
TEST_003	B
TEST_004	A
TEST_005	B
TEST_006	C
TEST_007	B
TEST_008	A
•	
•	
•	
TEST_172	C
TEST_173	B
TEST_174	B

분석 모델
XGBOOST

XGBOOST

```
if __name__=="__main__":

  train_start = time.time()

  study = optuna.create_study(direction="maximize")
  study.optimize(objective, n_trials=1000, show_progress_bar=True)

  print("Number of finshed trials: ", len(study.trials))
  print("Best trial:")

  trial = study.best_trial

  print("Accuracy:{}".format(trial.value))
  print(" Best hyperparameters:")
  for key, value in trial.params.items():
    print("{}:{}".format(key, value))


  clf = XGBClassifier(**study.best_params, random_state=42, use_label_encoder=False)
  clf.fit(X_train, y_train)
  preds = clf.predict(X_test)
  accuracy = sklearn.metrics.accuracy_score(y_test, preds)

  print("Accuracy: {}",format(accuracy))
```

분석 모델
XGBOOST

XGBOOST

```
# gridsearch 적용
# 하이퍼 파라미터 값 찾기

from sklearn.model_selection import GridSearchCV

param_xgb = {"max_depth": [1,2,3,6,8],
             "min_child_weight" : [1,2,3,4,5],
             "n_estimators": [1000,1100,1200,1300],
             'gamma' : [0.00001,0.00001,0.00001, 0.0001, 0.001],
             'learning_rate': [0.01, 0.02, 0.03,0.04,0.05]
             }

gscv_xgb = GridSearchCV (estimator = XGBClassifier(), param_grid = param_xgb, scoring ='accuracy', cv = 3,
                         refit=True, n_jobs= 1, verbose=2)
gscv_xgb.fit(train_x, train_y)
```

```
from sklearn import decomposition

 #줄이고 싶은 차원 수 지정
pca = decomposition.PCA(n_components=1)

# X에 오리지널 데이터 넣기
train_x['PCA'] = pca.fit_transform(train_x)
train_x

test_x['PCA'] = pca.transform(test_x)
test_x
```

분석 모델

XGBOOST

XGBOOST

	trait	SNP_01	SNP_02	SNP_03	SNP_04	SNP_05	SNP_06	SNP_07	SNP_08	SNP_09	SNP_10	SNP_11	SNP_12	SNP_13	SNP_14	SNP_15	PCA
0	1	1	5	0	4	0	1	5	4	4	1	1	4	5	2	4	7.897005
1	2	5	1	3	5	3	0	0	0	0	5	1	0	0	0	0	-6.015086
2	2	5	1	0	0	2	1	0	0	0	1	0	4	5	0	5	0.132941
3	2	5	1	2	0	3	0	0	0	0	5	0	4	1	0	0	-5.783503
4	1	0	5	0	5	0	5	5	0	5	1	5	4	5	0	5	13.486259
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
170	2	1	5	3	0	2	1	0	5	0	5	5	0	0	0	4	-3.255729
171	2	5	0	0	0	2	1	0	0	0	1	0	0	1	0	4	-3.861143
172	2	5	0	0	0	2	1	0	0	0	5	1	0	1	0	5	-4.857870
173	2	1	5	2	4	3	5	0	4	0	5	1	0	0	0	0	-2.864874
174	2	5	5	3	4	2	0	4	5	0	5	5	0	0	0	0	-3.308072

분석 모델

XGBOOST

XGBOOST

```
kmeans_train = train_x
kmeans = KMeans(n_clusters=3, random_state=42).fit(kmeans_train)
train_x['cluster'] = kmeans.predict(kmeans_train)
test_x['cluster'] = kmeans.predict(test_x)
train_x
```

trait	SNP_01	SNP_02	SNP_03	SNP_04	SNP_05	SNP_06	SNP_07	SNP_08	SNP_09	SNP_10	SNP_11	SNP_12	SNP_13	SNP_14	SNP_15	PCA	cluster
2	5	1	0	4	2	0	0	5	0	5	1	0	0	0	0	-7.592571	2
2	1	1	2	0	0	1	0	4	0	1	0	4	5	0	0	-1.492393	1
2	5	5	0	4	3	5	0	4	4	1	0	0	0	0	0	-1.285883	1
1	0	5	0	4	0	5	5	0	5	1	5	5	5	0	5	13.568591	0
2	5	5	3	0	3	0	0	0	0	5	0	0	1	0	4	-4.883778	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2	1	1	0	4	3	1	0	4	0	5	1	4	0	0	0	-3.670829	2
2	5	0	2	0	0	1	4	4	0	1	1	0	1	0	4	-3.209077	1
1	1	5	0	4	0	1	5	4	4	0	5	5	5	2	5	10.555078	0
1	0	5	0	4	0	5	5	0	4	1	1	4	5	2	5	11.703959	0
2	5	1	2	5	3	1	0	0	0	5	0	0	5	0	4	-2.486841	1

분석 모델
XGBOOST

XGBOOST

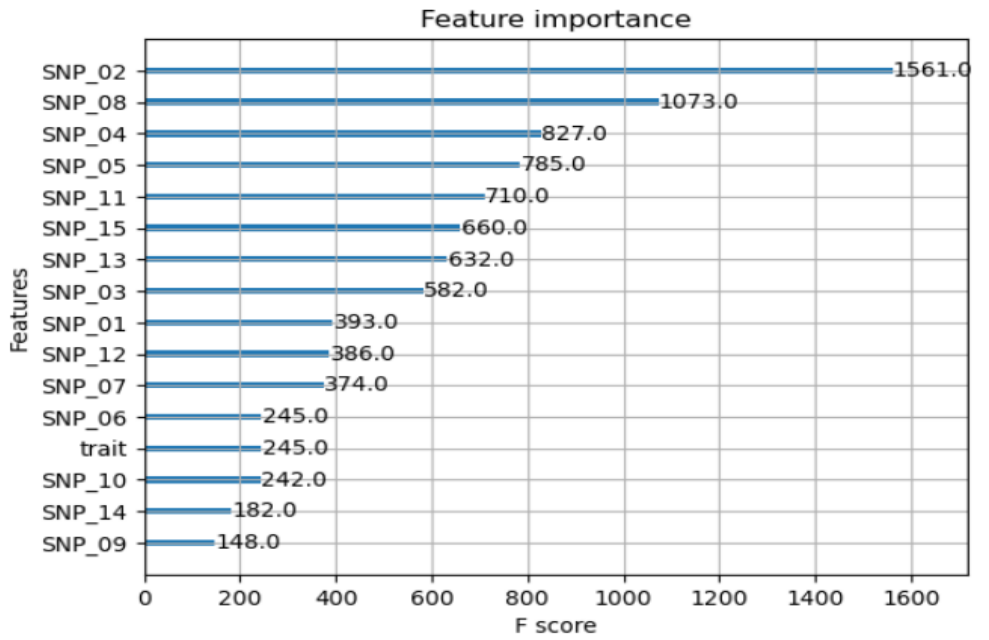
```
xgb_wrapper = XGBClassifier(booster='gbtree',
                            gamma=1e-05,
                            max_depth=3,
                            min_child_weight=1,
                            n_estimators=1000,
                            random_state=42,
                            learning_rate=0.03)
model = xgb_wrapper.fit(train_x, train_y, eval_metric='merror')
plot_importance(model)
```

```
preds = xgb_wrapper.predict(test_x)
print(preds)
```

```
[0 1 2 1 0 1 2 1 0 0 2 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 0 0 1 0 0 1 2 0 1 2
 1 1 2 0 1 2 1 1 1 1 2 1 2 0 1 0 1 1 1 2 0 1 2 0 1 2 2 2 0 1 0 0 1 1 1 0 0
 2 1 2 1 1 1 2 1 0 1 1 1 1 1 2 0 1 1 2 1 1 2 0 1 0 2 0 1 1 2 0 0 2 1 0 1 2
 1 1 1 1 0 0 2 1 2 0 1 1 2 2 1 1 1 0 1 0 0 1 1 1 2 0 0 1 0 0 0 2 1 1 1 0 1
 2 0 0 1 0 1 1 0 0 1 2 0 1 2 2 1 0 0 2 1 1 0 1 2 2 1 1]
```

plot_importance(model)

```
<AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'>
```



분석 모델

XGBOOST

XGBOOST

```
submit = pd.read_csv('open/sample_submission.csv')
```

```
submit['class'] = class_le.inverse_transform(preds)
```

```
submit.to_csv('./answer/xgboost40.csv', index=False)
```

Public Score 결과

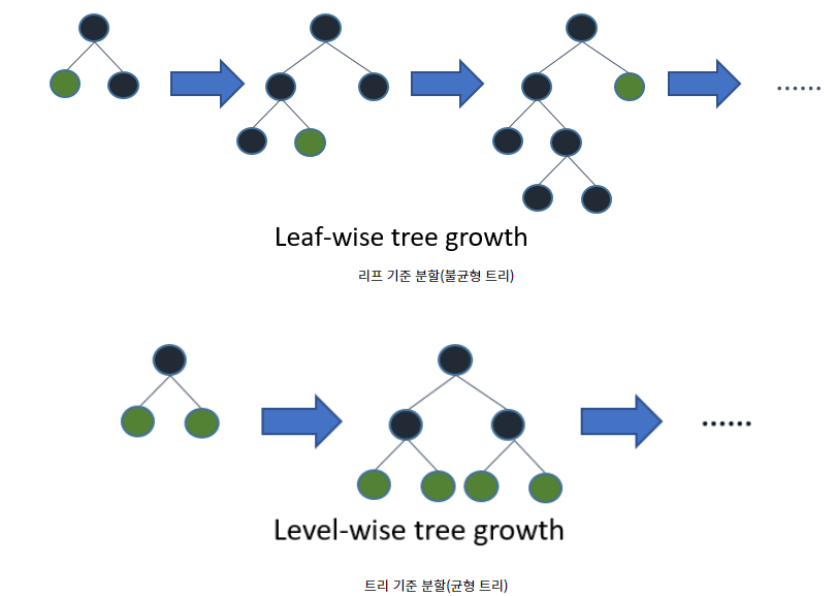
제출 번호	제출 파일	제출일자	점수	최종제출여부
781783	xgb_submit10.csv edit	2022-12-24 13:04:13	0.9719171917	☐

XGBOOST모델로 분석한 결과

1.0 만점에 0.97191719171의 결과 도출

분석 모델

LightGBM



LightGBM 주요 파라미터

num_leaves: 클수록 정확도는 높아지지만 오버피팅 발생 가능
min_data_in_leaf: 클수록 오버피팅 방지
max_depth: 위 두개 파라미터와 결합하여 오버피팅 방지
objective: 사용하는 데이터셋의 타겟팅 값의 형태에 따라 조정 필요
metric: 성능 평가를 어떤 것으로 할 것인지 조정 필요

리프기준 분할 방식 사용

XGBoost의 성능개선 버전으로 시간이 1.5배정도 짧게 소요

메모리 사용량이 상대적으로 적음

적은 데이터셋(공식문건 기준 1만건 이하)에서 과대적합 발생 가능성 다수

분석 모델

LightGBM

LightGBM 코드 1st (파라미터 조정 전)

```
# 1st

lgb_dtrain = lgb.Dataset(data = train_x, label = train_y) # 학습 데이터를 LightGBM 모델에 맞게 변환
lgb_param = {'max_depth': 10, # 트리 깊이
             'learning_rate': 0.01, # Step Size
             'n_estimators': 100, # Number of trees, 트리 생성 개수
             'objective': 'multiclass', # 목적 함수
             'num_class': len(set(train_y)) + 1} # 파라미터 추가, Label must be in [0, num_class) -> num_class보다 1 커야한다.
lgb_model = lgb.train(params = lgb_param, train_set = lgb_dtrain) # 학습 진행
lgb_model_predict = np.argmax(lgb_model.predict(test_x), axis = 1) # 평가 데이터 예측, Softmax의 결과값 중 가장 큰 값의 Label로 예측
```

```
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[LightGBM] [Warning] Accuracy may be bad since you didn't explicitly set num_leaves OR 2^max_depth > num_leaves. (num_leaves=31).
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000096 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 48
[LightGBM] [Info] Number of data points in the train set:
[LightGBM] [Info] Start training from score -1.334238
[LightGBM] [Info] Start training from score -0.832146
[LightGBM] [Info] Start training from score -1.198897
[LightGBM] [Info] Start training from score -34.538776
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
```

```
print(lgb_model_predict)
```

```
[0 1 2 1 0 2 2 1 0 0 2 1 1 0 1 1 0 1 1 2 1 1 1 0 1 1 1 0 0 1 0 0 1 2 0 1 2
 1 1 2 0 1 2 1 1 1 1 2 1 2 0 1 0 1 1 1 2 0 1 2 0 1 2 2 2 0 1 0 0 1 1 1 0 0
 2 1 2 1 1 1 2 1 0 1 1 1 1 1 2 0 1 1 2 1 1 2 0 1 0 2 0 1 1 2 0 0 2 1 0 1 2
 1 1 1 1 0 0 2 1 1 0 1 1 2 2 1 1 1 0 1 0 0 1 1 1 2 0 0 1 0 0 0 2 1 1 1 0 1
 2 0 0 1 0 1 1 0 0 1 2 0 1 2 1 1 0 0 2 1 1 0 1 2 2 1 1]
```

```
submit = pd.read_csv('open/sample_submission.csv')
```

```
submit['class'] = class_le.inverse_transform(lgb_model_predict)
```

```
submit.to_csv('./answer/LGBM_submit.csv', index=False)
```

LGBM_submit.scv

id	class
TEST_000	A
TEST_001	B
TEST_002	C
TEST_003	B
TEST_004	A
TEST_005	C
TEST_006	C
TEST_007	B
TEST_008	A
⋮	
TEST_172	C
TEST_173	B
TEST_174	B

분석 모델

LightGBM

LightGBM 결과

제출 번호	제출 파일	제출일자	점수	최종제출여부
781531	submit.csv 1st - LightGBM edit	2022-12-23 17:56:22	0.9622367466	☐

LightGBM모델로 분석한 결과(1st - 파라미터 조정 전)

0.9622367466의 결과 도출

분석 모델

LightGBM

LightGBM 코드 2nd (파라미터 조정 후)

```
# 2nd

lgb_dtrain = lgb.Dataset(data = train_x, label = train_y) # 학습 데이터를 LightGBM 모델에 맞게 변환
lgb_param = {'max_depth': 10, # 트리 깊이
             'num_leaves': 100,
             'min_data_in_leaf': 10,
             'learning_rate': 0.03, # Step Size
             'n_estimators': 100, # Number of trees, 트리 생성 개수
             'objective': 'multiclass', # 목적 함수
             'num_class': len(set(train_y)) + 1} # 파라미터 추가, Label must be in [0, num_class) -> num_class보다 1 커야한다.
lgb_model = lgb.train(params = lgb_param, train_set = lgb_dtrain) # 학습 진행
lgb_model_predict = np.argmax(lgb_model.predict(test_x), axis = 1) # 평가 데이터 예측, Softmax의 결과값 중 가장 큰 값의 Label로 예측
```

```
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000065 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 48
[LightGBM] [Info] Number of data points in the train set:
[LightGBM] [Info] Start training from score -1.334238
[LightGBM] [Info] Start training from score -0.832146
[LightGBM] [Info] Start training from score -1.198897
[LightGBM] [Info] Start training from score -34.538776
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
[LightGBM] [Warning] No further splits with positive gain
```

```
print(lgb_model_predict)
```

```
[0 1 2 1 0 1 2 1 0 0 2 1 1 0 1 1 0 1 1 2 1 1 1 0 1 1 1 0 0 1 0 0 1 2 0 1 2
 1 1 2 0 1 2 1 1 1 1 2 1 2 0 1 0 1 1 1 2 0 1 2 0 1 2 2 2 0 1 0 0 1 1 1 0 0
 2 1 2 1 1 1 2 1 0 1 1 1 1 1 2 0 1 1 2 1 1 2 0 1 0 2 0 1 1 2 0 0 2 1 0 1 2
 1 1 1 1 0 0 2 1 1 0 1 1 2 2 1 2 1 0 1 0 0 1 1 1 2 0 0 1 0 0 0 2 1 1 1 0 1
 2 0 0 1 0 1 1 0 0 1 2 0 1 2 2 1 0 0 2 1 1 0 1 2 2 1 1]
```

```
submit = pd.read_csv('open/sample_submission.csv')
```

```
submit['class'] = class_le.inverse_transform(lgb_model_predict)
```

```
submit.to_csv('./answer/LGBM_submit.csv', index=False)
```

LGBM_submit.scv

id	class
TEST_000	A
TEST_001	B
TEST_002	C
TEST_003	B
TEST_004	A
TEST_005	B
TEST_006	C
TEST_007	B
TEST_008	A
⋮	
TEST_172	C
TEST_173	B
TEST_174	B

분석 모델

LightGBM

LightGBM 결과

제출 번호	제출 파일	제출일자	점수	최종제출여부
781803	submit.csv 2nd - LightGBM edit	2022-12-24 14:47:23	0.9622367466	☐

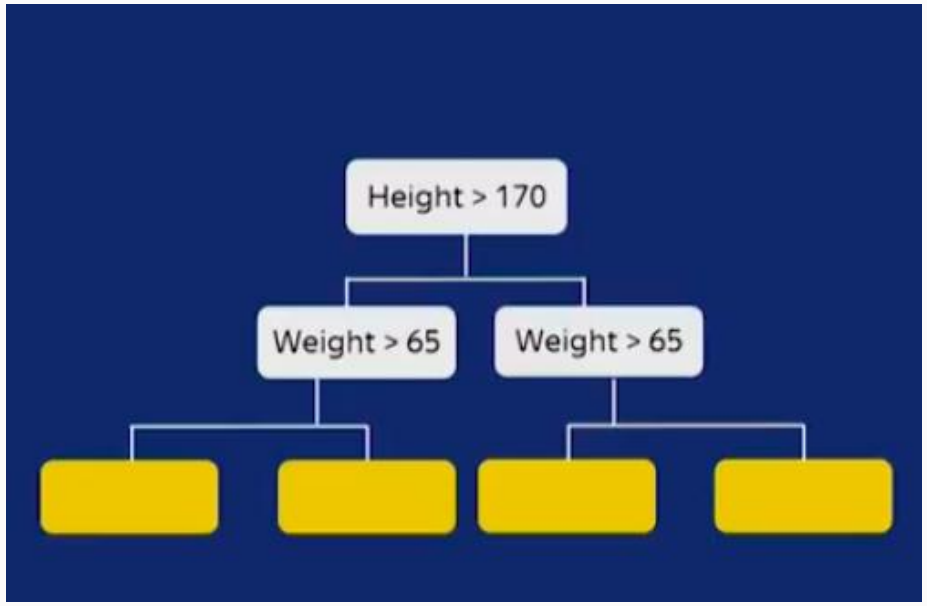
LightGBM모델로 분석한 결과(2nd - 파라미터 조정 후)

0.9622367466의 결과 도출(1st와 결과 동일.)

과적합으로 예상됨.

분석 모델

CatBoost



Parameters

X
y
cat_features
text_features
embedding_features
sample_weight
baseline
use_best_model
eval_set
verbose
logging_level

plot
plot_file
column_description
silent
early_stopping_rounds
save_snapshot
snapshot_file
snapshot_interval
init_model
log_cout
log_cerr

Categorical Boosting의 약자로 Catergorical feature를 처리하는데 최적화

대칭모양의 트리를 생성

구현하기 쉽고 성능이 강력

데이터 대부분이 수치형 변수인 경우, Light GBM 보다 학습 속도가 느림

파라미터 설명 - https://catboost.ai/en/docs/concepts/python-reference_catboostclassifier_fit#x

분석 모델

CatBoost

CatBoost 코드

```
import random
import os
from sklearn import preprocessing, decomposition
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings, random
warnings.filterwarnings(action='ignore')

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import StratifiedKFold
from sklearn.cluster import KMeans
from catboost import CatBoostClassifier, Pool
```

```
train = pd.read_csv('/content/sample_data/train.csv')
test = pd.read_csv('/content/sample_data/test.csv')
```

	id	father	mother	gender	trait	SNP_01	SNP_02	SNP_03
0	TRAIN_000	0	0	0	2	G G	A G	A A
1	TRAIN_001	0	0	0	2	A G	A G	C A
2	TRAIN_002	0	0	0	2	G G	G G	A A
3	TRAIN_003	0	0	0	1	A A	G G	A A
4	TRAIN_004	0	0	0	2	G G	G G	C C
...	...	...	...	...	...	...	...	...
257	TRAIN_257	0	0	0	2	A G	A G	A A
258	TRAIN_258	0	0	0	2	G G	A A	C A
259	TRAIN_259	0	0	0	1	A G	G G	A A
260	TRAIN_260	0	0	0	1	A A	G G	A A
261	TRAIN_261	0	0	0	2	G G	A G	C A

262 rows × 21 columns

SNP_10	SNP_11	SNP_12	SNP_13	SNP_14	SNP_15	class
G G	A G	A A	A A	A A	A A	B
A G	A A	G A	G G	A A	A A	C
A G	A A	A A	A A	A A	A A	B
A G	G G	G G	G G	A A	G G	A
G G	A A	A A	A G	A A	G A	C
...	...	...	...	...	...	...
G G	A G	G A	A A	A A	A A	B
A G	A G	A A	A G	A A	G A	C
A A	G G	G G	G G	C A	G G	A
A G	A G	G A	G G	C A	G G	A
G G	A A	A A	G G	A A	G A	B

분석 모델

CatBoost

CatBoost 코드 - 전처리

*18 페이지 참조

```
def get_x_y(df):
    if 'class' in df.columns:
        df_x = df.drop(columns=['id', 'class','father','mother','gender'])
        df_y = df['class']
        return df_x, df_y
    else:
        df_x = df.drop(columns=['id','father','mother','gender'])
        return df_x

train_x, train_y = get_x_y(train)
test_x = get_x_y(test)
train_x
```

Train, Test 셋의 전처리

의미가 없는 id, father, mother, gender 의 값을 삭제하고 Train의 target인 'class' 항목을 따로 분류하였다.

	trait	SNP_01	SNP_02	SNP_03	SNP_04	SNP_05	SNP_06	SNP_07	SNP_08	SNP_09	SNP_10	SNP_11	SNP_12	SNP_13	SNP_14	SNP_15
0	2	GG	AG	AA	GA	CA	AA	AA	GG	AA	GG	AG	AA	AA	AA	AA
1	2	AG	AG	CA	AA	AA	AG	AA	GA	AA	AG	AA	GA	GG	AA	AA
2	2	GG	GG	AA	GA	CC	GG	AA	GA	GA	AG	AA	AA	AA	AA	AA
3	1	AA	GG	AA	GA	AA	GG	GG	AA	GG	AG	GG	GG	GG	AA	GG
4	2	GG	GG	CC	AA	CC	AA	AA	AA	AA	GG	AA	AA	AG	AA	GA
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
257	2	AG	AG	AA	GA	CC	AG	AA	GA	AA	GG	AG	GA	AA	AA	AA
258	2	GG	AA	CA	AA	AA	AG	GA	GA	AA	AG	AG	AA	AG	AA	GA
259	1	AG	GG	AA	GA	AA	AG	GG	GA	GA	AA	GG	GG	GG	CA	GG
260	1	AA	GG	AA	GA	AA	GG	GG	AA	GA	AG	AG	GA	GG	CA	GG
261	2	GG	AG	CA	GG	CC	AG	AA	AA	AA	GG	AA	AA	GG	AA	GA

262 rows x 16 columns

분석 모델

CatBoost

CatBoost 코드 - LabelEncoder()를 사용한 전처리

*18 페이지 참조

```
class_le = preprocessing.LabelEncoder()
snp_le = preprocessing.LabelEncoder()
snp_col = [f'SNP_{str(x).zfill(2)}' for x in range(1,16)]

snp_data = []
for col in snp_col:
    snp_data += list(train_x[col].values)

train_y = class_le.fit_transform(train_y)
snp_le.fit(snp_data)
```

```
LabelEncoder()
```

```
for col in train_x.columns:
    if col in snp_col:
        train_x[col] = snp_le.transform(train_x[col])
        test_x[col] = snp_le.transform(test_x[col])
```

	trait	SNP_01	SNP_02	SNP_03	SNP_04	SNP_05	SNP_06	SNP_07	SNP_08
0	2	5	1	0	4	2	0	0	5
1	2	1	1	2	0	0	1	0	4
2	2	5	5	0	4	3	5	0	4
3	1	0	5	0	4	0	5	5	0
4	2	5	5	3	0	3	0	0	0
...	...	...	...	...	...	...	...	...	...
257	2	1	1	0	4	3	1	0	4
258	2	5	0	2	0	0	1	4	4
259	1	1	5	0	4	0	1	5	4
260	1	0	5	0	4	0	5	5	0
261	2	5	1	2	5	3	1	0	0

262 rows × 16 columns

분석 모델
CatBoost

CatBoost 코드 - PCA 전처리

```
#줄이고 싶은 차원 수 지정
pca = decomposition.PCA(n_components=2)

# X에 오리지널 데이터 넣기
pca_x = pca.fit_transform(train_x)
train_pca = pd.DataFrame(data=pca_x, columns = ['pc1', 'pc2'])
# 분산비율 확인
pca.explained_variance_ratio_
```

```
array([0.45543534, 0.08824541])
```

```
pca_t = pca.transform(test_x)
test_pca = pd.DataFrame(data=pca_t, columns = ['pc1', 'pc2'])
pca.explained_variance_ratio_
```

```
array([0.45543534, 0.08824541])
```

```
train_x[['pc1','pc2']] = train_pca
```

```
test_x[['pc1','pc2']] = test_pca
```

pc1	pc2
-5.368759	-2.435071
-1.055281	3.238915
-0.909257	-3.231096
9.594443	-0.441934
-3.453353	1.055642
...	...
-2.595668	-2.564450
-2.269160	3.581536
7.463567	-0.083800
8.275949	0.807607
-1.758462	0.661203

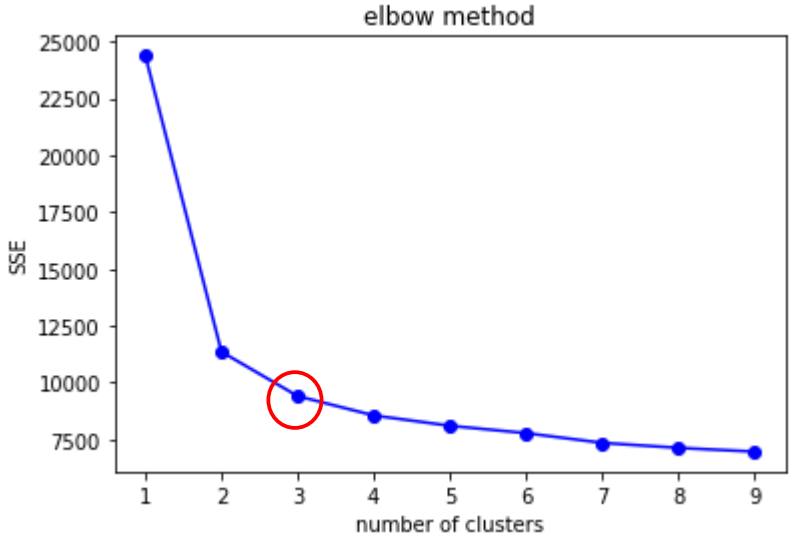
분석 모델

CatBoost

CatBoost 코드

```
def elbow(data, length):
    sse = [] # sum of squre error 오차제곱합
    for i in range(1, length):
        kmeans = KMeans(n_clusters=i)
        kmeans.fit(data)
        # SSE 값 저장
        sse.append(kmeans.inertia_)
    plt.plot(range(1, length), sse, 'bo-')
    plt.title("elbow method")
    plt.xlabel("number of clusters")
    plt.ylabel("SSE")
    plt.show()
elbow(train_x, 10)
```

Elbow(train_x, 10)




```
kmeans_train = train_x
kmeans = KMeans(n_clusters=3, random_state=42).fit(kmeans_train)
train_x['cluster'] = kmeans.predict(kmeans_train)
test_x['cluster'] = kmeans.predict(test_x)
```

['cluster']

	pc1	pc2	cluster
	-5.368759	-2.435071	2
	-1.055281	3.238915	1
	-0.909257	-3.231096	2
	9.594443	-0.441934	0
	-3.453353	1.055642	1
	...	...	...
	-2.595668	-2.564450	2
	-2.269160	3.581536	1
	7.463567	-0.083800	0
	8.275949	0.807607	0
	-1.758462	0.661203	1

N_clusters 의 개수 지정.

Elbow method를 이용하여 최적의 cluster의 수를 3으로 지정하여 K-means clustering을 진행하였다.

분석 모델

CatBoost

CatBoost 코드

```
X_train, X_test, y_train, y_test = train_test_split(train_x, train_y, test_size=0.20)
```

```
seed = 42
n_fold = 15   #FOLD 10
```

```
X = train_x
y = train_y
X_test = X_test
```

#변수선언

```
skfold = StratifiedKFold(n_splits=n_fold, shuffle=True, random_state=seed)
folds=[]
for train_idx, valid_idx in skfold.split(X_train, y_train):
        folds.append((train_idx, valid_idx))

cat_cols = ['trait','SNP_01','SNP_02','SNP_03','SNP_04','SNP_05','SNP_06','SNP_07','SNP_08',
            'SNP_09','SNP_10','SNP_11','SNP_12','SNP_13','SNP_14','SNP_15','cluster']
for fold in range(n_fold):
  print(f'₩n---------------- Fold {fold} ----------------₩n')
  train_idx, valid_idx = folds[fold]
  X_train, X_valid, y_train, y_valid = X.iloc[train_idx], X.iloc[valid_idx], y[train_idx], y[valid_idx]
  train_data = Pool(data=X_train, label=y_train, cat_features=cat_cols)

  valid_data = Pool(data=X_valid, label=y_valid, cat_features=cat_cols)

  model_cat = CatBoostClassifier(n_estimators= 50000, eta =0.005, max_depth =3)
  model_cat.fit(train_data, eval_set=valid_data, early_stopping_rounds=100, verbose=100)
```

X_train

```
X_train
```

	trait	SNP_01	SNP_02	SNP_03	SNP_04	SNP_05	SNP_06	SNP_07	SNP_08
16	1	0	5	0	5	0	5	5	0
236	2	1	1	3	4	2	1	0	5
70	2	1	0	2	0	2	1	0	4
29	1	1	5	0	4	0	5	5	0
132	2	5	5	3	4	2	5	0	5
...	...	...	...	...	...	...	...	...	...
2	2	5	5	0	4	3	5	0	4
21	2	5	1	3	0	3	1	0	5

분석 모델

CatBoost

CatBoost 코드

```
---------------- Fold 0 ----------------

0:      learn: 1.0925407        test: 1.0936900 best: 1.0936900 (0)     total: 58.9ms   remaining: 49m 6s
100:    learn: 0.7220881        test: 0.7963329 best: 0.7963329 (100)   total: 522ms    remaining: 4m 18s
200:    learn: 0.5405143        test: 0.6594119 best: 0.6594119 (200)   total: 952ms    remaining: 3m 55s
300:    learn: 0.4323714        test: 0.5870054 best: 0.5870054 (300)   total: 1.44s    remaining: 3m 57s
400:    learn: 0.3617106        test: 0.5365966 best: 0.5365966 (400)   total: 1.86s    remaining: 3m 50s
500:    learn: 0.3112203        test: 0.5078857 best: 0.5078857 (500)   total: 2.32s    remaining: 3m 48s
600:    learn: 0.2768744        test: 0.4920962 best: 0.4920962 (600)   total: 2.77s    remaining: 3m 47s
700:    learn: 0.2507172        test: 0.4833682 best: 0.4832729 (696)   total: 3.18s    remaining: 3m 43s
800:    learn: 0.2307821        test: 0.4737162 best: 0.4734819 (798)   total: 3.63s    remaining: 3m 43s
900:    learn: 0.2142986        test: 0.4602757 best: 0.4601130 (899)   total: 4.07s    remaining: 3m 41s
1000:   learn: 0.1992387        test: 0.4567678 best: 0.4561051 (992)   total: 4.55s    remaining: 3m 42s
1100:   learn: 0.1876761        test: 0.4553101 best: 0.4533715 (1038)  total: 4.98s    remaining: 3m 41s
Stopped by overfitting detector  (100 iterations wait)

bestTest = 0.4533715006
bestIteration = 1038

Shrink model to first 1039 iterations.
```

```
---------------- Fold 14 ----------------

0:      learn: 1.0931162        test: 1.0918114 best: 1.0918114 (0)     total: 4.48ms   remaining: 3m 43s
100:    learn: 0.7420089        test: 0.6463141 best: 0.6463141 (100)   total: 467ms    remaining: 3m 50s
200:    learn: 0.5671599        test: 0.4310281 best: 0.4310281 (200)   total: 886ms    remaining: 3m 39s
300:    learn: 0.4632065        test: 0.3098644 best: 0.3098644 (300)   total: 1.31s    remaining: 3m 36s
400:    learn: 0.3974634        test: 0.2402173 best: 0.2402173 (400)   total: 1.75s    remaining: 3m 37s
500:    learn: 0.3492542        test: 0.1928532 best: 0.1928532 (500)   total: 2.2s     remaining: 3m 37s
600:    learn: 0.3145197        test: 0.1602673 best: 0.1602673 (600)   total: 2.65s    remaining: 3m 38s
700:    learn: 0.2860738        test: 0.1358232 best: 0.1358232 (700)   total: 3.11s    remaining: 3m 38s
800:    learn: 0.2658653        test: 0.1196454 best: 0.1196454 (800)   total: 3.54s    remaining: 3m 37s
900:    learn: 0.2473040        test: 0.1068030 best: 0.1067033 (898)   total: 3.97s    remaining: 3m 36s
1000:   learn: 0.2307553        test: 0.0955129 best: 0.0955129 (1000)  total: 4.4s     remaining: 3m 35s
1100:   learn: 0.2176349        test: 0.0868370 best: 0.0868370 (1100)  total: 4.82s    remaining: 3m 34s
1200:   learn: 0.2062318        test: 0.0791818 best: 0.0791818 (1200)  total: 5.26s    remaining: 3m 33s
1300:   learn: 0.1964240        test: 0.0738577 best: 0.0738577 (1300)  total: 5.71s    remaining: 3m 33s
1400:   learn: 0.1877031        test: 0.0684382 best: 0.0684382 (1400)  total: 6.13s    remaining: 3m 32s
1500:   learn: 0.1788341        test: 0.0635340 best: 0.0635340 (1500)  total: 6.57s    remaining: 3m 32s
1600:   learn: 0.1709832        test: 0.0595601 best: 0.0595601 (1600)  total: 7.01s    remaining: 3m 31s
1700:   learn: 0.1643495        test: 0.0560072 best: 0.0560072 (1700)  total: 7.46s    remaining: 3m 31s
1800:   learn: 0.1587472        test: 0.0536242 best: 0.0536242 (1800)  total: 7.91s    remaining: 3m 31s
```

fold 0~14

Model_cat.fit

```
21100:  learn: 0.0161660        test: 0.0068396 best: 0.0068396 (21100) total: 1m 35s   remaining: 2m 10s
21200:  learn: 0.0160794        test: 0.0068069 best: 0.0068068 (21196) total: 1m 36s   remaining: 2m 10s
21300:  learn: 0.0160034        test: 0.0068036 best: 0.0067888 (21254) total: 1m 36s   remaining: 2m 9s
Stopped by overfitting detector  (100 iterations wait)

bestTest = 0.006788769588
bestIteration = 21254

Shrink model to first 21255 iterations.
```

분석 모델
CatBoost

CatBoost 코드 - 모델을 이용한 예측

```
print(model_cat.get_all_params())
```

```
{'nan_mode': 'Min', 'eval_metric': 'MultiClass', 'combinations_ctr': ['Borders:CtrBorderCount=15:CtrBorderType=Uniform:TargetBorderCount=2:Targ
```

```
preds = model_cat.predict(X_test)
print('Done.')
```

```
Done.
```

```
from sklearn.metrics import accuracy_score as acc
acc(y_test, preds)
```

```
1.0
```

```
f1_score(y_test, preds, average='micro')
```

```
1.0
```

```
preds = model_cat.predict(train_x)
print('Done.')
acc(train_y, preds)
```

```
Done.
0.9961832061068703
```

```
f1_score(train_y, preds, average='micro')
```

```
0.9961832061068703
```

```
preds = model_cat.predict(test_x)
print('Done.')
print(preds)
```

```
Done.
[[0]
 [1]
 [2]
 [1]
 [0]
 [1]
```

분석 모델

CatBoost

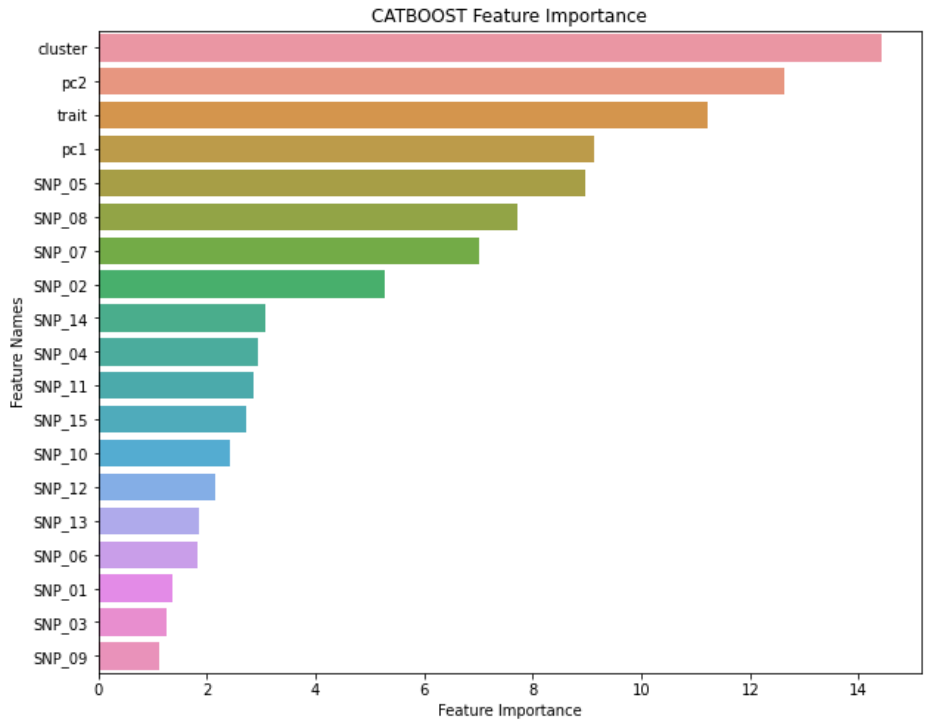
CatBoost 코드 – feature

```
def plot_feature_importance(importance,names,model_type):

    feature_importance = np.array(importance)
    feature_names = np.array(names)

    data={'feature_names':feature_names,'feature_importance':feature_importance}
    fi_df = pd.DataFrame(data)

    fi_df.sort_values(by=['feature_importance'], ascending=False,inplace=True)

    plt.figure(figsize=(10,8))

    sns.barplot(x=fi_df['feature_importance'], y=fi_df['feature_names'])

    plt.title(model_type + ' Feature Importance')
    plt.xlabel('Feature Importance')
    plt.ylabel('Feature Names')
```

```
submit = pd.read_csv('/content/sample_data/sample_submission.csv')

submit['class'] = class_le.inverse_transform(preds)

submit.to_csv('/content/sample_data/catfin.csv', index=False)
```

Plot_feature_importance

```
plot_feature_importance(model_cat.get_feature_importance(),X_test.columns,'CATBOOST')
```



Catfin.csv

```
id,class
TEST_000,A
TEST_001,B
TEST_002,C
TEST_003,B
TEST_004,A
TEST_005,B
TEST_006,C
TEST_007,B
TEST_008,A
TEST_009,A
TEST_010,C
TEST_011,B
TEST_012,C
TEST_013,A
TEST_014,B
TEST_015,B
TEST_016,A
TEST_017,B
```

분석 모델

CatBoost

Public Score 결과

제출 번호	제출 파일	제출일자	점수	최종제출여부
783007	catfin2.csv edit	2022-12-27 12:48:55	0.9719171917	☑

CatBoost모델로 분석한 결과

1.0 만점에 0.97191719171의 결과 도출

분석 모델

Deep Learning - 분류방식 SGDClassifier

Deep Learning 코드

```
# 신경망 생성. Dense 4개. Dropout layer 2개. 활성화 함수는 입력층은 ReLU,나머지는 softmax 사용.
# 컬럼의 수가 16개 (trait, SNP_01~15) 이므로 input_shape은 (16,)이 된다.
# 출력층 값은 ABC 3개중에 하나가 나와야 하므로 unit=3로 지정.

model = keras.Sequential()
model.add(keras.layers.Dense(units=100, activation='ReLU', input_shape=(16,)))
model.add(keras.layers.Dropout(0.3))
model.add(keras.layers.Dense(units=100))
model.add(keras.layers.Dropout(0.3))
model.add(keras.layers.Dense(units=100))

model.add(keras.layers.Dense(units=3, activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               1700

 dropout (Dropout)           (None, 100)               0

 dense_1 (Dense)             (None, 100)               10100

 dropout_1 (Dropout)         (None, 100)               0

 dense_2 (Dense)             (None, 100)               10100

 dense_3 (Dense)             (None, 3)                 303

=================================================================
Total params: 22,203
Trainable params: 22,203
Non-trainable params: 0
_________________________________________________________________
```

분석 모델

Deep Learning - 분류방식 SGDClassifier

Deep Learning 코드

```
]: # compile 작업. 최적화 알고리즘 adam사용. 다중분류 손실함수. 정확도 사용.

   model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
```

```
]: # 600번 학습. (횟수는 줄일 수 있을 것 같다.)

   checkpoint_cb =  keras.callbacks.ModelCheckpoint('SGDclassifer_den5.h5', save_best_only=True)
   early_stopping_cb = keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True)

   history = model.fit(X_train, y_train, epochs=600)
```

```
7/7 [==============================] - 0s 2ms/step - loss: 0.1727 - accuracy: 0.9330
Epoch 589/600
7/7 [==============================] - 0s 2ms/step - loss: 0.1250 - accuracy: 0.9522
Epoch 590/600
7/7 [==============================] - 0s 2ms/step - loss: 0.1281 - accuracy: 0.9378
```

```
: # 정확도 출력

  model.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 5ms/step - loss: 0.3026 - accuracy: 0.9434
```

```
: [0.3026180565357208, 0.9433962106704712]
```

분석 모델

Deep Learning - 분류방식 SGDClassifier

Deep Learning 코드

In [16]:
```
# 확률적 경사 하강법 분류를 사용.

clf = SGDClassifier(random_state=CFG.SEED)
clf.fit(train_x, train_y)
```

Out[16]:
```
SGDClassifier(random_state=42)
```

In [17]:
```
# # 예측 및 테스트 값 출력

preds = clf.predict(test_x)
print('Done.')
print(preds)
preds.shape
preds.ndim
```

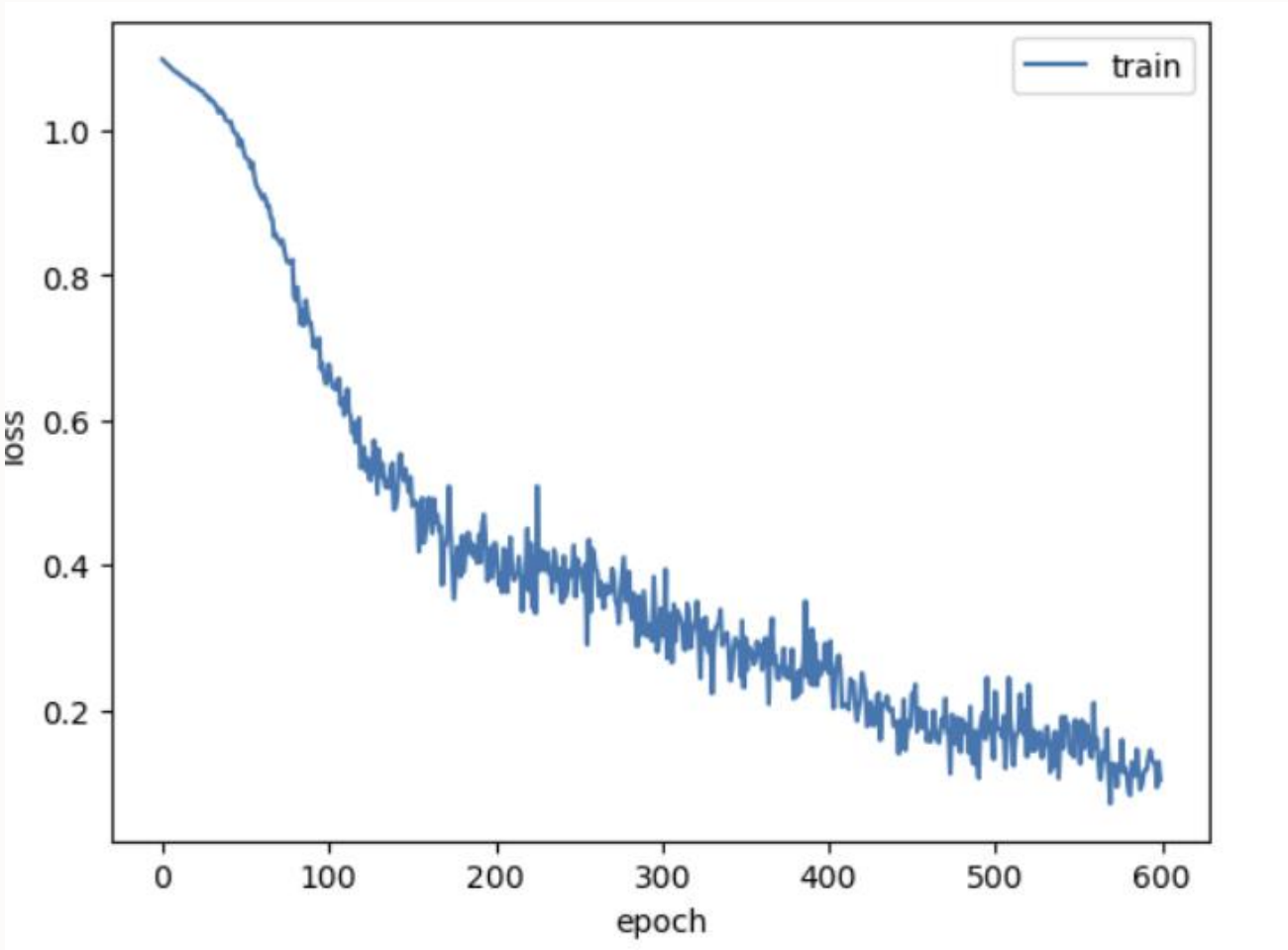
```
Done.
[0 1 2 1 0 1 2 1 0 0 2 1 1 0 1 1 0 1 1 2 1 1 1 0 1 1 1 0 0 1 0 0 1 2 0 1 2
 1 1 2 0 1 2 1 1 1 1 2 1 2 0 1 0 1 1 1 2 0 1 2 0 1 2 2 2 0 1 0 0 1 1 1 0 0
 2 1 2 1 1 1 2 1 0 1 1 1 1 1 2 0 1 1 1 1 1 2 0 1 0 2 0 1 1 2 0 0 2 1 0 1 2
 1 1 1 1 0 0 2 1 2 0 1 1 2 2 1 1 1 0 1 0 0 1 1 1 2 0 0 1 0 0 0 2 1 1 1 0 1
 2 0 0 1 0 1 1 0 0 1 2 0 1 2 2 1 0 0 2 1 1 0 1 2 2 1 1]
```

Out[17]:
```
1
```

분석 모델

Deep Learning - 분류방식 SGDClassifier

Deep Learning 코드



SGDClassifierDropout_Dense6.csv

	A	B
1	id	class
2	TEST_000	A
3	TEST_001	B
4	TEST_002	C
5	TEST_003	B
6	TEST_004	A
7	TEST_005	B
8	TEST_006	C
9	TEST_007	B
10	TEST_008	A

.

.

173	TEST_171	C
174	TEST_172	C
175	TEST_173	B
176	TEST_174	B

분석 모델

Deep Learning - 분류방식 SGDClassifier

Public Score 결과

제출 번호	제출 파일	제출일자	점수	최종제출여부
782415	SGDClassifierDropout_Dense6.csv edit	2022-12-26 12:53:43	0.9622367466	☐

Deep Learning - 분류방식 SGDClassifier 로 분석한 결과

1.0 만점에 0.9622367466의 결과 도출

분석 모델

Deep Learning - 분류방식 RandomForestClassifier

Deep Learning 코드

```
# 신경망 생성. Dense 5개. 활성화 함수는 입력층은 ReLU,출력층은 softmax 사용.
# 컬럼의 수가 16개 (trait, SNP_01~15) 이므로 input_shape은 (16,)이 된다.
# 출력층 값은 ABC 3개중에 하나가 나와야 하므로 unit=3로 지정.

model = keras.Sequential()
model.add(keras.layers.Dense(units=100, activation='ReLU', input_shape=(16,)))
model.add(keras.layers.Dense(units=100))
model.add(keras.layers.Dense(units=100))
model.add(keras.layers.Dense(units=100))


model.add(keras.layers.Dense(units=3, activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               1700

 dense_1 (Dense)             (None, 100)               10100

 dense_2 (Dense)             (None, 100)               10100

 dense_3 (Dense)             (None, 100)               10100

 dense_4 (Dense)             (None, 3)                 303

=================================================================
Total params: 32,303
Trainable params: 32,303
Non-trainable params: 0
_________________________________________________________________
```

분석 모델

Deep Learning - 분류방식 RandomForestClassifier

Deep Learning 코드

```
# compile 작업. 최적화 알고리즘 adam사용. 다중분류 손실함수. 정확도 사용.

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics='accuracy')
```

```
history = model.fit(X_train, y_train, epochs=600)
```

```
Epoch 1/600
7/7 [==============================] - 0s 2ms/step - loss: 1.0864 - accuracy: 0.5885
Epoch 2/600
7/7 [==============================] - 0s 1ms/step - loss: 1.0623 - accuracy: 0.6986
Epoch 3/600
7/7 [==============================] - 0s 2ms/step - loss: 1.0395 - accuracy: 0.6986
```

```
model.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.2430 - accuracy: 0.9245

[0.24296587705612183, 0.9245283007621765]
```

분석 모델

Deep Learning - 분류방식 RandomForestClassifier

Deep Learning 코드

```
# 랜덤포레스트 분류 사용.

clf = RandomForestClassifier(random_state=CFG.SEED)
clf.fit(train_x, train_y)
```

```
RandomForestClassifier(random_state=42)
```

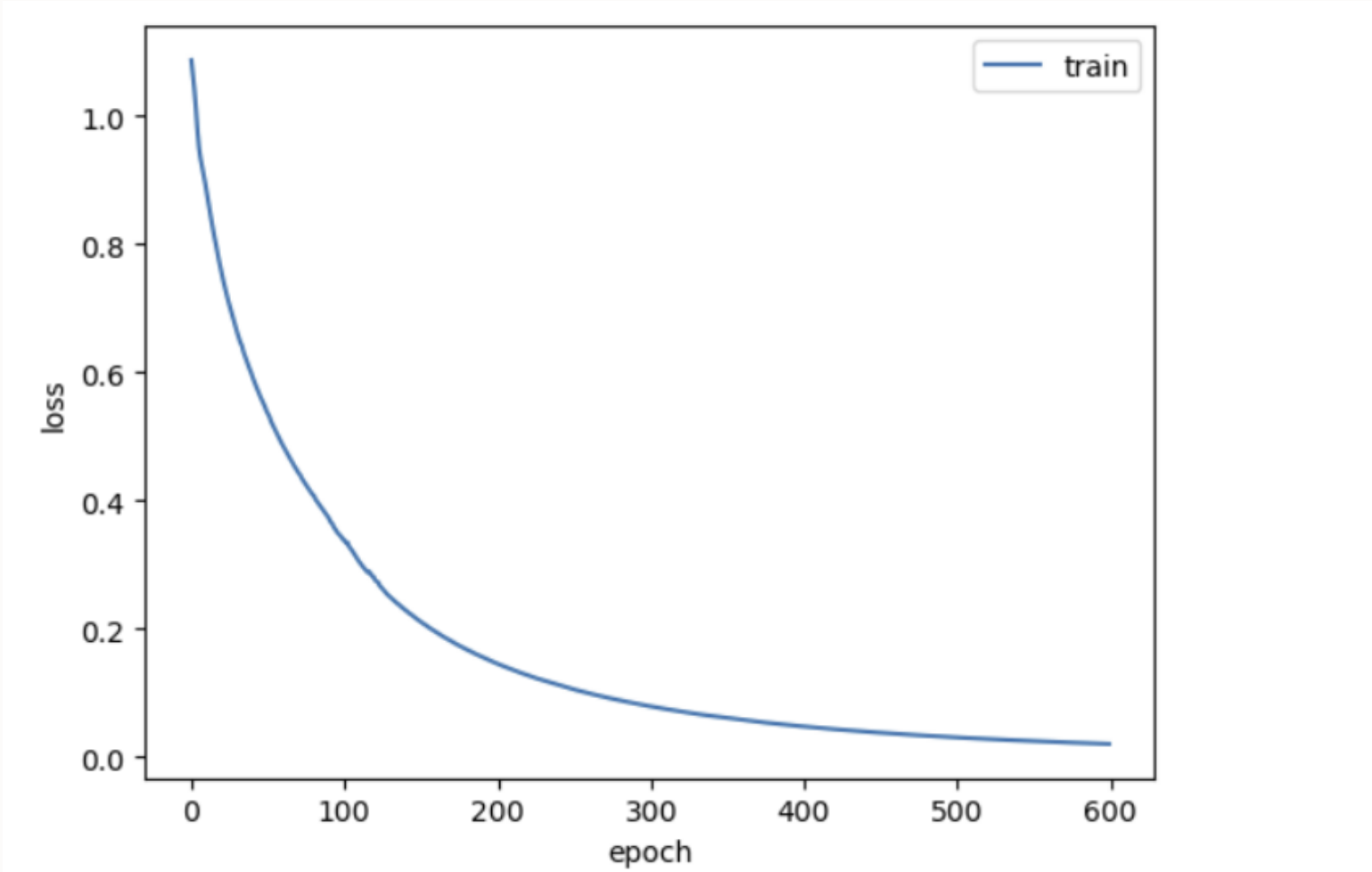
```
# 예측 및 테스트 값 출력

preds = clf.predict(test_x)
print('Done.')
print(preds)
preds.shape
preds.ndim
```

```
Done.
[0 1 2 1 0 1 2 1 0 0 2 1 2 0 1 1 0 1 1 2 1 1 1 0 1 1 1 0 0 1 0 0 1 2 0 1 2
 1 1 2 0 1 2 1 1 1 1 2 1 2 0 1 0 1 1 1 2 0 1 2 0 1 2 2 2 0 1 0 0 1 1 1 0 0
 2 1 2 1 1 1 2 1 0 1 1 1 1 1 2 0 1 1 2 1 1 2 0 1 0 2 0 1 1 2 0 0 2 1 0 1 2
 1 1 1 1 0 0 2 1 1 0 1 1 2 2 1 2 1 0 1 0 0 1 1 1 2 0 0 1 0 0 0 2 1 1 1 0 1
 2 0 0 1 0 1 1 0 0 1 2 0 1 2 2 1 0 0 2 1 1 0 1 2 2 1 1]

1
```

분석 모델

Deep Learning - 분류방식 RandomForestClassifier

Deep Learning 코드



submitRandomReLU.csv

	A	B
1	id	class
2	TEST_000	A
3	TEST_001	B
4	TEST_002	C
5	TEST_003	B
6	TEST_004	A
7	TEST_005	B
8	TEST_006	C
9	TEST_007	B
10	TEST_008	A
11	TEST_009	A
12	TEST_010	C
	⋮	
174	TEST_172	C
175	TEST_173	B
176	TEST_174	B

분석 모델

Deep Learning - 분류방식 RandomForestClassifier

Public Score 결과

제출 번호	제출 파일	제출일자	점수	최종제출여부
780809	submitRandomReLU.csv edit	2022-12-22 18:15:05	0.9622367466	☐

Deep Learning - 분류방식 RandomForestClassifier 로 분석한 결과

1.0 만점에 0.9622367466의 결과 도출

분석 결과

분석 결과

Catboost : 0.9719171917

XGBosst : 0.9719171917

LightGBM : 0.9622367466

Random Forest : 0.9622367466

K-최근접 이웃 (kNN) : 0.9622367466

DLN - RandomForestClassifier : 0.9622367466

DLN - SGDClassifier : 0.9622367466

Decision tree : 0. 9354953039

Gaussian Naïve Bayes : 0.8854584855

Public score 최고점은 catboost

분석 결과

Catboost 1위선정 이유

Catboost

```
preds = model_cat.predict(train_x)
print('Done.')
acc(train_y, preds)
```

```
Done.
0.9961832061068703
```

```
f1_score(train_y, preds, average='micro')
```

```
0.9961832061068703
```

XGBoost

```
preds = xgb_wrapper.predict(train_x)
```

```
accuracy_score(train_y, preds)
```

```
0.9923664122137404
```

```
f1_score(train_y,preds, average= 'micro')
```

```
0.9923664122137404
```

경진대회에 제출한 Public Score는 동률이나

Train 데이터 기준 accuracy와 F1_score 모두

Catboost가 XGBoost에 비해 점수가 앞서 1위에 선정

분석 결과

대회 순위

1	33_33	33	0.99078
2	지웅몬	지웅 be 가영	0.99078
3	사랑해요여러분	사랑	0.99078
139	숫자123	숫자	0.97191

22.12.28 기준 총 489팀 중 139위 기록 중

입상을 하진 못했지만,

여러 알고리즘을 사용해서

실무에서 사용한 데이터를 다루어 봤다는 점에 의의

감사합니다!🙏