# 유전체 정보 품종 분류 AI 경진대회

team : SuperGENOME

1. 개발환경
2. EDA
   - 데이터 확인
   - 데이터 정제
   - 상관관계 확인
3. Feature Engineering
4. Modeling
5. 결과도출

# 1. 개발환경

## 패키지 및 Version 정보

Google Colab

Python - 3.8.10
pandas - 1.3.5
matplotlib - 3.2.2
csv - 1.0
numpy - 1.21.6
seaborn - 0.11.2

# 2. EDA

## 2.1 결측치 확인

```
#결측치 확인
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 262 entries, 0 to 261
Data columns (total 21 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      262 non-null    object
 1   father  262 non-null    int64
 2   mother  262 non-null    int64
 3   gender  262 non-null    int64
 4   trait   262 non-null    int64
 5   SNP_01  262 non-null    object
 6   SNP_02  262 non-null    object
 7   SNP_03  262 non-null    object
 8   SNP_04  262 non-null    object
 9   SNP_05  262 non-null    object
 10  SNP_06  262 non-null    object
 11  SNP_07  262 non-null    object
 12  SNP_08  262 non-null    object
 13  SNP_09  262 non-null    object
 14  SNP_10  262 non-null    object
 15  SNP_11  262 non-null    object
 16  SNP_12  262 non-null    object
 17  SNP_13  262 non-null    object
 18  SNP_14  262 non-null    object
 19  SNP_15  262 non-null    object
 20  class   262 non-null    object
dtypes: int64(4), object(17)
memory usage: 43.1+ KB
```

```
#결측치 확인
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175 entries, 0 to 174
Data columns (total 20 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      175 non-null    object
 1   father  175 non-null    int64
 2   mother  175 non-null    int64
 3   gender  175 non-null    int64
 4   trait   175 non-null    int64
 5   SNP_01  175 non-null    object
 6   SNP_02  175 non-null    object
 7   SNP_03  175 non-null    object
 8   SNP_04  175 non-null    object
 9   SNP_05  175 non-null    object
 10  SNP_06  175 non-null    object
 11  SNP_07  175 non-null    object
 12  SNP_08  175 non-null    object
 13  SNP_09  175 non-null    object
 14  SNP_10  175 non-null    object
 15  SNP_11  175 non-null    object
 16  SNP_12  175 non-null    object
 17  SNP_13  175 non-null    object
 18  SNP_14  175 non-null    object
 19  SNP_15  175 non-null    object
dtypes: int64(4), object(16)
memory usage: 27.5+ KB
```

# 2. EDA

## 2.2 데이터 구성 탐색

**father, mother, gender 변수 drop**

```
snp_num1 = train.columns[train.columns.str.contains('SNP')].tolist()
for num1 in snp_num1:
  def count_into_lst(lst):
      answer=dict()
      for num in train[num1]:
          if num not in answer.keys():
              answer[num]=1
          else:
              answer[num]+=1
      return answer
  num_train1 =count_into_lst(train[num1])

  print(num_train1)
```

```
{'G G': 141, 'A G': 71, 'A A': 50}
{'A G': 97, 'G G': 108, 'A A': 57}
{'A A': 122, 'C A': 92, 'C C': 48}
{'G A': 93, 'A A': 120, 'G G': 49}
{'C A': 86, 'A A': 94, 'C C': 82}
{'A A': 61, 'A G': 122, 'G G': 79}
{'A A': 163, 'G G': 51, 'G A': 48}
{'G G': 79, 'G A': 96, 'A A': 87}
{'A A': 182, 'G A': 56, 'G G': 24}
{'G G': 151, 'A G': 68, 'A A': 43}
{'A G': 96, 'A A': 83, 'G G': 83}
{'A A': 136, 'G A': 73, 'G G': 53}
{'A A': 52, 'G G': 115, 'A G': 95}
{'A A': 185, 'C C': 23, 'C A': 54}
{'A A': 107, 'G G': 55, 'G A': 100}
```

```
snp_num2 = test.columns[test.columns.str.contains('SNP')].tolist()
for num2 in snp_num2:
  def count_into_lst(lst):
      answer=dict()
      for num in train[num2]:
          if num not in answer.keys():
              answer[num]=1
          else:
              answer[num]+=1
      return answer
  num_train2 =count_into_lst(test[num2])

  print(num_train2)
```

```
{'G G': 141, 'A G': 71, 'A A': 50}
{'A G': 97, 'G G': 108, 'A A': 57}
{'A A': 122, 'C A': 92, 'C C': 48}
{'G A': 93, 'A A': 120, 'G G': 49}
{'C A': 86, 'A A': 94, 'C C': 82}
{'A A': 61, 'A G': 122, 'G G': 79}
{'A A': 163, 'G G': 51, 'G A': 48}
{'G G': 79, 'G A': 96, 'A A': 87}
{'A A': 182, 'G A': 56, 'G G': 24}
{'G G': 151, 'A G': 68, 'A A': 43}
{'A G': 96, 'A A': 83, 'G G': 83}
{'A A': 136, 'G A': 73, 'G G': 53}
{'A A': 52, 'G G': 115, 'A G': 95}
{'A A': 185, 'C C': 23, 'C A': 54}
{'A A': 107, 'G G': 55, 'G A': 100}
```

```
gen_im1 = ['father','mother','gender']
for im1 in gen_im1:
  def count_into_lst(lst):
      answer=dict()
      for num in train[im1]:
          if num not in answer.keys():
              e
      retur
  print(cou
```

```
{0: 262}
{0: 262}
{0: 262}
```

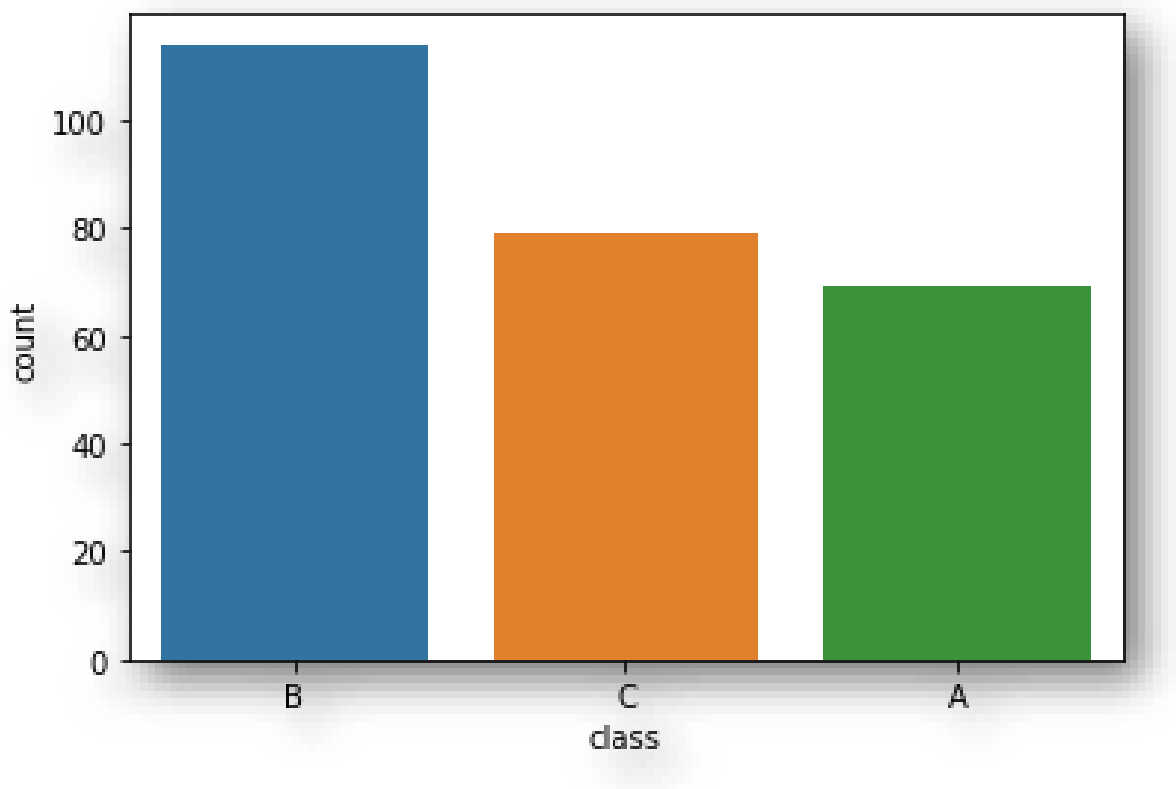
```
gen_im2 = ['father','mother','gender']
for im2 in gen_im2:
  def count_into_lst(lst):
      answer=dict()
      for num in test[im2]:
          if num not in answer.keys():
              answer[num]=1
          else:
              answer[num]+=1
      return answer
  print(count_into_lst(test[im2]))
```

```
{0: 175}
{0: 175}
{0: 175}
```

# 2. EDA

## 2.2 데이터 구성 탐색

### Class 변수 확인



# 2. EDA

## 2.3 데이터 정제

### 데이터 타입 변환

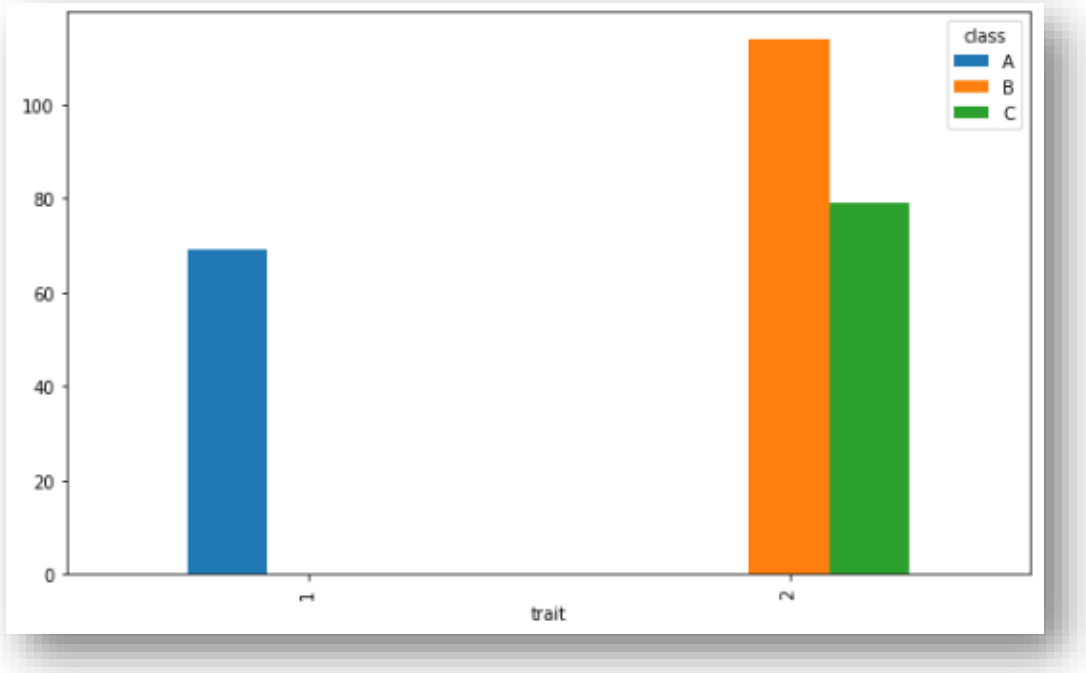
```
# class 변수 A, B, C 각각 0,1,2로 변환
train['class'] = train['class'].map({'A':0, 'B':1, 'C':2}).astype(int)
```

```
# 변수 변환
snp_col = train.columns[train.columns.str.contains('SNP')].tolist()
for col in snp_col:
  train[col] = train[col].map({'A A':1, 'G G':11, 'A G':111, 'G A':1111, 'C C':11111, 'A C':111111, 'C A':1111111}).astype(int)
  test[col] = test[col].map({'A A':1, 'G G':11, 'A G':111, 'G A':1111, 'C C':11111, 'A C':111111, 'C A':1111111}).astype(int)
```

# 2. EDA

## 2.4 상관관계 확인

**trait 변수에 따른 class 구분**



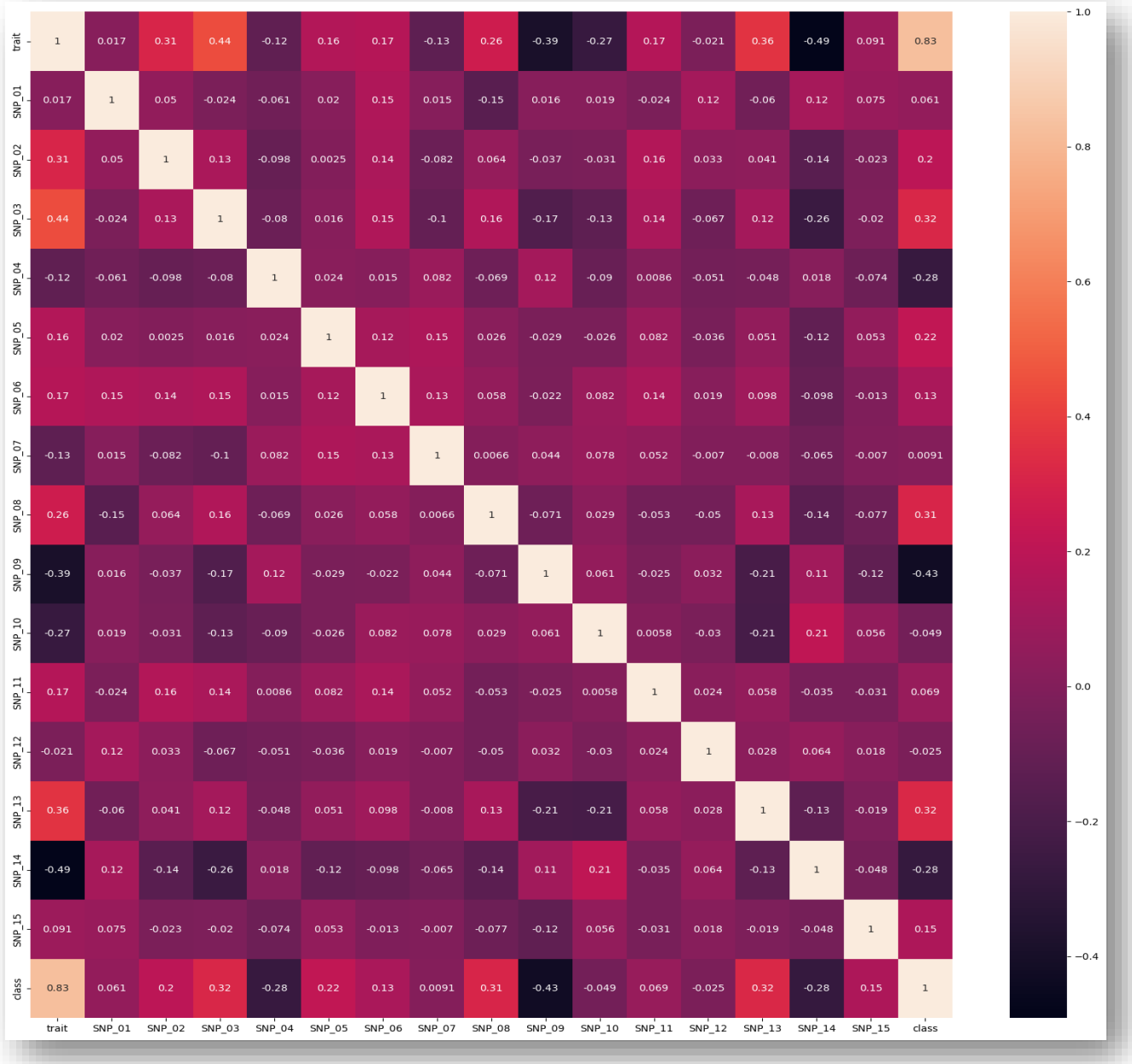

**Trait 변수가 1일 경우, 'A' class**
**그 외의 경우, 'B'/'C' class**

# 2. EDA

## 2.4 상관관계 확인

최소한의 SNP 사용으로
정확도를 높여야 하므로
상관관계 확인

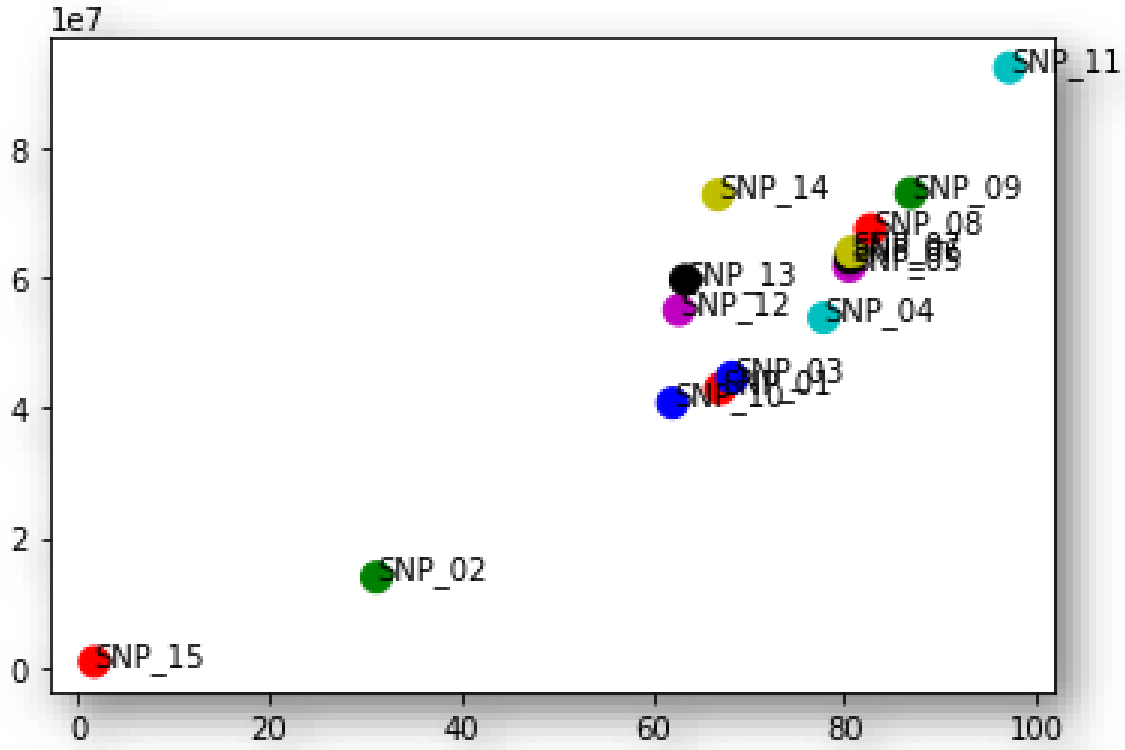
trait 변수와 class 변수에
각 SNP가 가진
상관계수 확인하여,
영향이 낮은 SNP 요소 탐색



# 2. EDA

## 2.4 상관관계 확인

| | 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 |




SNP_info 정보 확인하여,
cm과 pos가 그래프 상 비슷한 구간에 위치할 경우,
같은 배열을 가진 것으로 판단 → 파생변수 생성 시도

# 3. Feature Engineering

## 파생변수 생성

```
# 파생변수 생성
train['SNP_1013'] = train['SNP_10'] + train['SNP_13']
test['SNP_1013'] = test['SNP_10'] + test['SNP_13']
```

```
#변수 선택
X_train = train.drop(['id', 'father', 'mother', 'gender','class','SNP_01','SNP_06','SNP_10','SNP_12','SNP_13'], axis = 1)
y_train = train['class']
X_test = test.drop(['id', 'father', 'mother', 'gender','SNP_01','SNP_06','SNP_10','SNP_12','SNP_13'], axis = 1)
```

cm, pos가 비슷한 SNP_10, SNP_13의 배열이 서로 같다고 가정 (ex : AA GG = GG AA)
info 그래프를 통해 cm과 pos의 좌표에서 비슷한 위치에 있는 SNP_10과 SNP_13을 연결

# 4. Modeling

## Tree model 중 성능 비교위해 voting 사용

```
#lightGBM, ExtraTreesClassifier, RandomForestClassifier 선택
model = lgbm.LGBMClassifier()

et_cls = ExtraTreesClassifier(n_estimators=500, min_samples_leaf=5, min_samples_split=7, max_features=12)
rf_cls = RandomForestClassifier(n_estimators=500, min_samples_leaf=5, min_samples_split=7, max_features=12)
lg_cls = model.fit(X_train,y_train)

# 모델 voting
voting = VotingClassifier(
    estimators=[
        ('et', et_cls),
        ('rf', rf_cls),
        ('lg', lg_cls)
    ]
)
voting.fit(X_train, y_train)
```

```
VotingClassifier(estimators=[('et',
                              ExtraTreesClassifier(max_features=12,
                                                   min_samples_leaf=5,
                                                   min_samples_split=7,
                                                   n_estimators=500)),
                             ('rf',
                              RandomForestClassifier(max_features=12,
                                                     min_samples_leaf=5,
                                                     min_samples_split=7,
                                                     n_estimators=500)),
                             ('lg', LGBMClassifier())])
```

# 5. 결과도출

```
# int -> object로 재변환 후 저장
pred = voting.predict(X_test)
sub['class'] = pred

sub['class'] = sub['class'].map({0:'A', 1:'B', 2:'C'}).astype(object)
sub.to_csv("/content/sample_data/gendata.csv", index = False)
```

| | id | class |
|---|---|---|
| 0 | TEST_000 | A |
| 1 | TEST_001 | B |
| 2 | TEST_002 | C |
| 3 | TEST_003 | C |
| 4 | TEST_004 | A |

# 감사합니다