Report

2021 Classification 경진대회

작성자	소프트웨어학과 김채은
작성일	2021. 10. 25.
담당교수	오세종 교수님

■ 개요

1. 데이터 셋 준비

- 훈련 셋을 x, y로 나눈다.
- 훈련 데이터를 표준화한다.

2. 훈련 모델 선택

- Logistic Regression, K Nearest Neighbor, Decision Tree, Random Forest, Support Vector Machine 중 가장 성능이 좋은 알고리즘을 선택한다.

3. Feature 선택

- 모델을 학습할 때 사용할 feature를 선택한다.
- 모델의 성능을 높일 수 있는 feature를 선별해야 한다.

4. 하이퍼 파라미터 튜닝

- 하이퍼 파라미터 값에 따라 모델의 성능이 달라지므로 최적의 하이퍼 파라미터 값을 찾아준다.
- 많은 시간이 소요될 수 있으므로 적절한 횟수를 테스트하여 하이퍼 파라미터 값을 찾는다.

5. 모델 평가

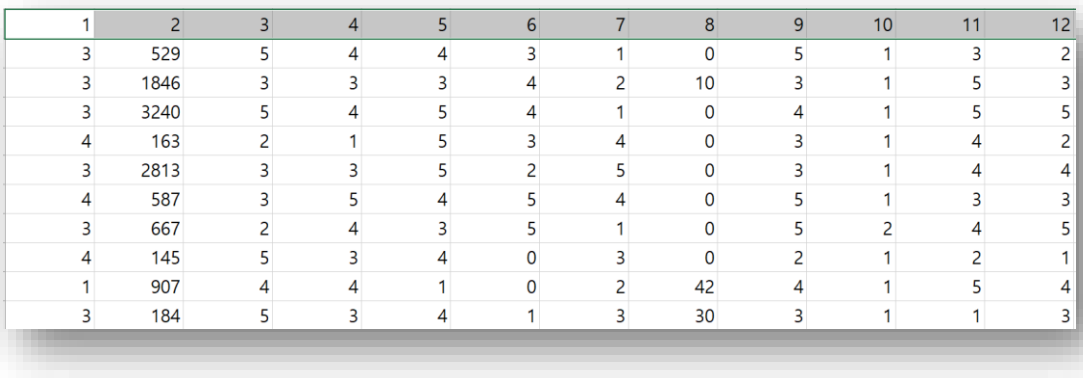
- 모델을 학습시키고 성능을 평가한다.

■ 개발 과정

1. 데이터 셋 준비

```
In [9]: runfile('C:/Users/82104/.spyder-py3/temp.py', wdir='C:/Users/82104/.spyder-py3')
0       1
1       2
2       1
3       1
4       2
       ..
3994    1
3995    1
3996    1
3997    2
3998    1
Name: 1.3, Length: 3999, dtype: int64
```

train_open.csv 파일이 2행부터 읽혀서 1행에 다음과 같이 인덱스를 주었다.



1	2	3	4	5	6	7	8	9	10	11	12
3	529	5	4	4	3	1	0	5	1	3	2
3	1846	3	3	3	4	2	10	3	1	5	3
3	3240	5	4	5	4	1	0	4	1	5	5
4	163	2	1	5	3	4	0	3	1	4	2
3	2813	3	3	5	2	5	0	3	1	4	4
4	587	3	5	4	5	4	0	5	1	3	3
3	667	2	4	3	5	1	0	5	2	4	5
4	145	5	3	4	0	3	0	2	1	2	1
1	907	4	4	1	0	2	42	4	1	5	4
3	184	5	3	4	1	3	30	3	1	1	3

총 4000행의 데이터가 읽힌다.

```
In [10]: runfile('C:/Users/82104/.spyder-py3/temp.py', wdir='C:/Users/82104/.spyder-py3')
0       1
1       1
2       2
3       1
4       1
       ..
3995    1
3996    1
3997    1
3998    2
3999    1
Name: 23, Length: 4000, dtype: int64
```

불러온 trainset을 train_x와 train_y로 나누었다.

```
import pandas as pd
import numpy as np
```

```
trainset                                                              =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/competition/train
_open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

print(train_x)
print(train_y)
```

```
In [24]: runfile('C:/Users/82104/.spyder-py3/temp.py', wdir='C:/Users/82104/.spyder-py3')
      1     2  3  4  5  6  7   8  9  10  ...  13  14  15  16  17  18  19  20  21  22
0     3   529  5  4  4  3  1   0  5   1  ...   2   3   5   1  32   0   2   3   3   2
1     3  1846  3  3  3  4  2  10  3   1  ...   2   3   2   2  22  12   2   1   3   3
2     3  3240  5  4  5  4  1   0  4   1  ...   1   4   1   2  61   0   2   5   4   1
3     4   163  2  1  5  3  4   0  3   1  ...   2   5   5   5  52   7   1   4   4   2
4     3  2813  3  3  5  2  5   0  3   1  ...   2   3   5   4  24   0   1   5   3   1
...  ..   ... .. .. .. .. ..  .. ..  ..  ...  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
3995  1   793  2  1  5  3  4  17  4   1  ...   2   1   2   3  23   1   2   5   5   3
3996  2  1235  5  5  5  4  2   1  2   2  ...   2   4   5   4  25   1   2   3   3   2
3997  5   399  3  5  1  1  1   0  1   2  ...   1   5   1   5  28   0   2   2   4   2
3998  1   328  5  5  2  4  3   0  4   2  ...   2   5   4   4  41   0   1   4   4   2
3999  1  2213  4  4  5  4  3   0  4   1  ...   2   4   1   3  39   0   2   3   3   1

[4000 rows x 22 columns]
0       1
1       1
2       2
3       1
4       1
       ..
3995    1
3996    1
3997    1
3998    2
3999    1
Name: 23, Length: 4000, dtype: int64
```

특성마다 범위와 편차의 차이가 있다. 이런 경우를 두고 스케일이 다르다고 한다. 따라서 scaler를 통해 전처리 과정을 거쳐줄 것이다.

처음에는 Standard Scaler를 사용했는데, scaler는 -1부터 1까지의 값으로 표준화해준다. 이후 코드를 실행하는데 마이너스 값을 읽어오는데 문제가 발생했다. 따라서 MinMax Scaler를 사용하여 0부터 1까지의 값으로 데이터를 표준화해주었다.

```
import pandas as pd
import numpy as np


from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2



trainset                                                                    =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/competition/train
_open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()


model = LogisticRegression(solver='saga', max_iter=10000)
scores = cross_val_score(model, train_x, train_y)
print(train_x)
print("전처리 전")
print("Accuracy: "+str(scores.mean()))

train_x = scaler.fit_transform(train_x)
scores = cross_val_score(model, train_x, train_y)
print(train_x)
print("전처리 후")
print("Accuracy: "+str(scores.mean()))
```

표준화 전처리 전과 후의 accuracy 결과이다. 전처리 후 accuracy가 약 0.05 향상되었다.

```
In [37]: runfile('C:/Users/82104/.spyder-py3/temp.py', wdir='C:/Users/82104/.spyder-py3')
      1     2  3  4  5  6  7   8  9  10  ...  13  14  15  16  17  18  19  20  21  22
0     3   529  5  4  4  3  1   0  5   1  ...   2   3   5   1  32   0   2   3   3   2
1     3  1846  3  3  3  4  2  10  3   1  ...   2   3   2   2  22  12   2   1   3   3
2     3  3240  5  4  5  4  1   0  4   1  ...   1   4   1   2  61   0   2   5   4   1
3     4   163  2  1  5  3  4   0  3   1  ...   2   5   5   5  52   7   1   4   4   2
4     3  2813  3  3  5  2  5   0  3   1  ...   2   3   5   4  24   0   1   5   3   1
...  ..   ... .. .. .. .. ..  .. ..  ..  ...  ..  ..  ..  ..  ..  ..  ..  ..  ..  ..
3995  1   793  2  1  5  3  4  17  4   1  ...   2   1   2   3  23   1   2   5   5   3
3996  2  1235  5  5  5  4  2   1  2   2  ...   2   4   5   4  25   1   2   3   3   2
3997  5   399  3  5  1  1  1   0  1   2  ...   1   5   1   5  28   0   2   2   4   2
3998  1   328  5  5  2  4  3   0  4   2  ...   2   5   4   4  41   0   1   4   4   2
3999  1  2213  4  4  5  4  3   0  4   1  ...   2   4   1   3  39   0   2   3   3   1

[4000 rows x 22 columns]
전처리 전
Accuracy: 0.78725
```

```
Accuracy: 0.78725
[[0.5        0.09600162 1.         ... 0.6        0.5        0.5       ]
 [0.5        0.36330424 0.6        ... 0.2        0.5        1.        ]
 [0.5        0.64623503 1.         ... 1.         0.75       0.        ]
 ...
 [1.         0.0696164  0.6        ... 0.4        0.75       0.5       ]
 [0.         0.05520601 1.         ... 0.8        0.75       0.5       ]
 [0.         0.43779176 0.8        ... 0.6        0.5        0.        ]]
전처리 후
Accuracy: 0.83825
```

2. 훈련 모델 선택

Logistic Regression, K Nearest Neighbor, Decision Tree, Random Forest, Support Vecotr Machine 중 가장 성능이 좋은 알고리즘을 선택할 것이다. KFold로 데이터를 분할한 후 cross validation을 시행한다.

```
import pandas as pd
import numpy as np


from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import model_selection

from sklearn.feature_selection import RFE



trainset                                                              =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/competition/train
_open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()
model = LogisticRegression(solver='saga', max_iter=10000)

train_x = scaler.fit_transform(train_x)

rfe = RFE(model, n_features_to_select = 20)

fit = rfe.fit(train_x, train_y)
print("Number of features: "+str(fit.n_features_))
temp = fit.support_.tolist()
fs = [idx for idx, x in enumerate(temp) if x==True]
print("Selected features:")
print(fs)
train_x = train_x[:, fs]

models = []
models.append(('LR',              LogisticRegression(solver='saga',
max_iter=10000)))
models.append(('KNN', KNeighborsClassifier()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('RF', RandomForestClassifier()))
models.append(('SVM', SVC()))
```

```
results = []
names = []

for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=42,
shuffle=True)
    cv_results = model_selection.cross_val_score(model, train_x,
train_y, cv=kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "[%s] Accuracy: %f, Standard deviation: %f" % (name,
cv_results.mean(), cv_results.std())
    print(msg)
```

```
Number of features: 20
Selected features:
[0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21]
[LR] Accuracy: 0.838750, Standard deviation: 0.018209
[KNN] Accuracy: 0.821500, Standard deviation: 0.020803
[DT] Accuracy: 0.835500, Standard deviation: 0.024413
[RF] Accuracy: 0.882500, Standard deviation: 0.010000
[SVM] Accuracy: 0.865750, Standard deviation: 0.015085
```

Random Forest 모델의 정확도가 가장 높으므로 이 알고리즘을 선택할 것이다.

3. Feature 선택

처음에 Logistic Regression 모델로 Feature를 선택하고 Random Forest에 그대로 적용했는데, 문득 모델 알고리즘이 달라지면 feature selection에 따른 결과도 달라지지 않을까 하는 생각이 들어 Random Forest 모델으로 다시 feature selection을 진행하였다. 예상대로 결과가 달라졌다.

feature selection은 Filter method, Backward elimination, 두 가지 평가방식을 사용하여 진행했다. Logistic Regression 모델에서 feature selection을 진행하며 Filter method보다 Backward elimination을 사용했을 때 feature selection의 정확도가 높다는 것을 확인하고, Random Forest 모델에서는 진행할 땐 Backward elimination을 통한 feature selection만을 진행했다.

Filter method는 각 feature가 독립적이라는 가정 하에서 feature의 중요도를 평가하는 방법이다. SelectKBest 모듈을 통해 각 feature의 평가점수를 얻을 수 있다.

```
import pandas as pd
import numpy as np


from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2



trainset                                                                =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/competition/train_open.c
sv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()


train_x = scaler.fit_transform(train_x)

test = SelectKBest(score_func=chi2, k=train_x.shape[1])
fit = test.fit(train_x, train_y)

print(np.round(fit.scores_, 3))
f_order = np.argsort(-fit.scores_)
print(f_order.tolist())
```

```
In [44]: runfile('C:/Users/82104/.spyder-py3/temp.py', wdir='C:/Users/82104/.spyder-py3')
[5.13000e-01 5.38570e+01 4.26980e+01 3.30910e+01 2.64000e-01 1.11431e+02
 2.10000e-01 2.44700e+00 2.10000e-02 5.49510e+02 6.50000e-01 4.14280e+01
 2.06000e-01 2.89670e+01 1.40000e-02 1.27900e+00 1.52000e-01 3.28700e+00
 5.00000e-03 2.65730e+01 9.96280e+01 2.36521e+02]
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7, 15, 10, 0, 4, 6, 12, 16, 8, 14, 18]
```

중요도에 따라 feature 조합을 만들어 모델의 성능을 테스트하였다.

```
import pandas as pd
import numpy as np


from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2



trainset                                                                =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/competition/train
_open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()
model = LogisticRegression(solver='saga', max_iter=10000)

train_x = scaler.fit_transform(train_x)

test = SelectKBest(score_func=chi2, k=train_x.shape[1])
fit = test.fit(train_x, train_y)

print(np.round(fit.scores_, 5))
f_order = np.argsort(-fit.scores_)
print(f_order.tolist())

acc_selected = np.zeros(train_x.shape[1] + 1)

for i in range(1, train_x.shape[1] + 1):
    fs = f_order[0:i].tolist()
    print(fs)
    train_x_selected = train_x[:, fs]
    scores = cross_val_score(model, train_x_selected, train_y, cv =
10)
    print(scores.mean())
    acc_selected[i] = scores.mean()
print("Best Accuracy: "+str(acc_selected.max()))
```

```
[9]
0.67175
[9, 21]
0.76075
[9, 21, 5]
0.8087500000000001
[9, 21, 5, 20]
0.8055
[9, 21, 5, 20, 1]
0.8094999999999999
[9, 21, 5, 20, 1, 2]
0.826
[9, 21, 5, 20, 1, 2, 11]
0.82775
[9, 21, 5, 20, 1, 2, 11, 3]
0.8385
[9, 21, 5, 20, 1, 2, 11, 3, 13]
0.83625
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19]
0.8380000000000001
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17]
0.83825
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7]
0.83675
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7, 15]
0.83725
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7, 15, 10]
0.83575
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7, 15, 10, 0]
0.83625
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7, 15, 10, 0, 4]
0.8370000000000001
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7, 15, 10, 0, 4, 6]
0.8370000000000001
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7, 15, 10, 0, 4, 6, 12]
0.83725
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7, 15, 10, 0, 4, 6, 12, 16]
0.8385
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7, 15, 10, 0, 4, 6, 12, 16, 8]
0.8390000000000001
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7, 15, 10, 0, 4, 6, 12, 16, 8, 14]
0.8377500000000001
[9, 21, 5, 20, 1, 2, 11, 3, 13, 19, 17, 7, 15, 10, 0, 4, 6, 12, 16, 8, 14, 18]
0.8380000000000001
Best Accuracy: 0.8390000000000001
```

최고 accuracy는 약 0.84이다. 14, 18 feature가 들어가니 accuracy가 하락하는 것을 볼 수 있다.

Backward elimination을 사용한 Feature selection을 진행했다.

```
import pandas as pd
import numpy as np


from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import RFE
```

```
trainset                                                                =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/competition/train
_open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()
model = LogisticRegression(solver='saga', max_iter=10000)

train_x = scaler.fit_transform(train_x)

acc_selected = np.zeros(train_x.shape[1] + 1)

for i in range(1, train_x.shape[1] + 1):
    rfe = RFE(model, n_features_to_select = i)
    fit = rfe.fit(train_x, train_y)
    print(fit.support_.tolist())
    temp = fit.support_.tolist()
    fs = [idx for idx, x in enumerate(temp) if x==True]
    print(fs)
```

for문을 통해 몇 개의 feature를 선택할 것인지 설정하고 반복하였다. 다음과 같이 True와 False 값으로 선정된 feature가 무엇인지 구분이 된다. True와 False로 이루어진 배열에서 인덱스를 추출했다.

```
[False, False, False, False, False, True, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False]
[5]
[False, False, True, False, False, True, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False]
[2, 5]
[False, False, True, False, False, True, False, False, False, False, False, False, False, False,
False, False, False, True, False, False, False, False]
[2, 5, 17]
[False, False, True, False, False, True, False, False, False, True, False, False, False, False,
False, False, False, True, False, False, False, False]
[2, 5, 9, 17]
[False, False, True, False, False, True, False, False, False, True, False, False, False, True,
False, False, False, False, True, False, False, False, False]
[2, 5, 9, 13, 17]
[False, False, True, False, False, True, False, False, False, True, False, False, False, True,
False, False, False, False, True, False, False, True, False]
[2, 5, 9, 13, 17, 20]
```

```
import pandas as pd
import numpy as np


from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import RFE



trainset                                                                =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/competition/train
_open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()
model = LogisticRegression(solver='saga', max_iter=10000)

train_x = scaler.fit_transform(train_x)

acc_selected = np.zeros(train_x.shape[1] + 1)

for i in range(1, train_x.shape[1] + 1):
    rfe = RFE(model, n_features_to_select = i)
    fit = rfe.fit(train_x, train_y)
    temp = fit.support_.tolist()
    fs = [idx for idx, x in enumerate(temp) if x==True]
    print(fs)
    train_x_selected = train_x[:, fs]
    scores = cross_val_score(model, train_x_selected, train_y, cv =
10)
    print(scores.mean())
    acc_selected[i] = scores.mean()

print("Best Accuracy: "+str(acc_selected.max()))
```

선택된 feature 조합으로 accuracy를 측정한 결과이다.

```
[5]
0.78725
[2, 5]
0.7805
[2, 5, 17]
0.7895
[2, 5, 9, 17]
0.82725
[2, 5, 9, 13, 17]
0.82825
[2, 5, 9, 13, 17, 20]
0.83375
[2, 5, 7, 9, 13, 17, 20]
0.8347499999999999
[2, 5, 7, 9, 11, 13, 17, 20]
0.8332499999999999
[1, 2, 5, 7, 9, 11, 13, 17, 20]
0.8332500000000002
[1, 2, 3, 5, 7, 9, 11, 13, 17, 20]
0.835
[1, 2, 3, 5, 7, 9, 11, 13, 17, 20, 21]
0.8390000000000001
[1, 2, 3, 5, 7, 9, 11, 13, 17, 19, 20, 21]
0.83675
[0, 1, 2, 3, 5, 7, 9, 11, 13, 17, 19, 20, 21]
0.83775
```

```
[0, 1, 2, 3, 5, 7, 9, 11, 13, 14, 17, 19, 20, 21]
0.8387500000000001
[0, 1, 2, 3, 5, 7, 9, 10, 11, 13, 14, 17, 19, 20, 21]
0.83725
[0, 1, 2, 3, 5, 7, 9, 10, 11, 12, 13, 14, 17, 19, 20, 21]
0.8375
[0, 1, 2, 3, 5, 7, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 21]
0.8397500000000001
[0, 1, 2, 3, 5, 7, 8, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 21]
0.8397499999999999
[0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 21]
0.8397500000000001
[0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21]
0.8402499999999999
[0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]
0.8394999999999999
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]
0.8380000000000001
Best Accuracy: 0.8402499999999999
```

[0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21] 일 때 0.84로, 6, 15를 제외했을 때 accuracy 가장 높은 것을 볼 수 있다. 몇 개의 feature를 가져와야 6, 15를 제외하는지 확인하고 적용시킨다.

```
print("Number of features: "+str(fit.n_features_))
```

Random Forest 모델에서 Backward elimination을 통한 feature selection을 진행했다.

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import model_selection

from sklearn.feature_selection import RFE



trainset                                                                =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/competition/train
_open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()
model = RandomForestClassifier()

train_x = scaler.fit_transform(train_x)

acc_selected = np.zeros(train_x.shape[1] + 1)

for i in range(1, train_x.shape[1] + 1):
    rfe = RFE(model, n_features_to_select = i)
    fit = rfe.fit(train_x, train_y)
    temp = fit.support_.tolist()
    fs = [idx for idx, x in enumerate(temp) if x==True]
    print(fs)
    train_x_selected = train_x[:, fs]
    scores = cross_val_score(model, train_x_selected, train_y, cv =
10)
    print(scores.mean())
    acc_selected[i] = scores.mean()

print("Best Accuracy: "+str(acc_selected.max()))
```

```
[1]
0.58025
[1, 5]
0.74
[1, 5, 16]
0.7597500000000001
[1, 5, 16, 21]
0.7985
[1, 5, 9, 16, 21]
0.8240000000000001
[1, 5, 9, 16, 20, 21]
0.8484999999999999
[1, 2, 5, 9, 16, 20, 21]
0.8615
[1, 2, 3, 5, 9, 16, 20, 21]
0.8702500000000001
[1, 2, 3, 5, 9, 13, 16, 20, 21]
0.8825
[1, 2, 3, 5, 9, 11, 13, 16, 20, 21]
0.8880000000000001
[1, 2, 3, 5, 9, 11, 13, 16, 17, 20, 21]
0.88575
[1, 2, 3, 5, 9, 11, 13, 16, 17, 19, 20, 21]
0.8862500000000001
[1, 2, 3, 5, 9, 11, 13, 15, 16, 17, 19, 20, 21]
0.885
```

```
[1, 2, 3, 5, 7, 9, 11, 13, 15, 16, 17, 19, 20, 21]
0.8869999999999999
[1, 2, 3, 5, 7, 9, 11, 13, 14, 15, 16, 17, 19, 20, 21]
0.88275
[1, 2, 3, 5, 6, 7, 9, 11, 13, 14, 15, 16, 17, 19, 20, 21]
0.88125
[0, 1, 2, 3, 5, 6, 7, 9, 11, 13, 14, 15, 16, 17, 19, 20, 21]
0.88225
[0, 1, 2, 3, 4, 5, 6, 7, 9, 11, 13, 14, 15, 16, 17, 19, 20, 21]
0.88175
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13, 14, 15, 16, 17, 19, 20, 21]
0.8835
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 19, 20, 21]
0.882
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21]
0.88225
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]
0.8815
Best Accuracy: 0.8880000000000001
```

[1, 2, 3, 5, 9, 11, 13, 16, 20, 21] 일 때, 0.89로 가장 높은 accuracy가 도출되었다. 0, 4, 6, 7, 8, 10, 12, 14, 15, 17, 18, 19번 feature가 제외되었다.

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

```
from sklearn import model_selection

from sklearn.feature_selection import RFE



trainset                                                                =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/competition/train
_open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()
model = RandomForestClassifier()
train_x = scaler.fit_transform(train_x)

rfe = RFE(model, n_features_to_select = 10)
fit = rfe.fit(train_x, train_y)
temp = fit.support_.tolist()
fs = [idx for idx, x in enumerate(temp) if x==True]
print("Number of features: " +str(fit.n_features_))
print("Features: ", end="")
print(fs)
train_x_selected = train_x[:, fs]
scores = cross_val_score(model, train_x_selected, train_y, cv = 10)
print("Accuracy: "+str(scores.mean()))
```

```
Number of features: 10
Features: [1, 2, 3, 5, 9, 11, 13, 16, 20, 21]
Accuracy: 0.8895
```

선정된 10개의 feature를 확인했다.

*** test set size 선택**

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import model_selection

from sklearn.feature_selection import RFE

trainset                                                                    =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/competition/train_open.c
sv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()
model = RandomForestClassifier(random_state=42)
train_x = scaler.fit_transform(train_x)

rfe = RFE(model, n_features_to_select = 10)
fit = rfe.fit(train_x, train_y)
temp = fit.support_.tolist()
fs = [idx for idx, x in enumerate(temp) if x==True]
print("Number of features: " +str(fit.n_features_))
print("Features: ", end="")
print(fs)
train_x = train_x[:, fs]

size = [0.2, 0.25, 0.3]
train_x_all = train_x
train_y_all = train_y
temp = np.zeros([50, 3])
for i in range(50):
    for j in range(3):
        train_x, test_x, train_y, test_y = train_test_split(train_x_all,
train_y_all, test_size=size[j])
        model.fit(train_x, train_y)
        accuracy = model.score(test_x, test_y)
        temp[i][j] = accuracy
    print(temp[i])

avg = np.mean(temp, 0).tolist()
for idx, i in enumerate(avg):
    print("Case: test size = "+str(size[idx]))
    print(i)
```

train set 과 test set이 어떻게 나누어지냐에 따라서 다른 결과를 보인다. 0.01 차이는 결과에 큰 영향이 없을 것 같으나 가장 accuracy 값이 좋게 나오는 0.3을 test set 사이즈의 비율로 선택했다.

```
[0.8725 0.891  0.8975]
[0.875      0.888      0.89083333]
[0.875      0.881      0.88666667]
[0.9    0.879  0.8925]
[0.88375    0.877      0.88333333]
[0.89125 0.878   0.8975 ]
[0.87875    0.896      0.89083333]
[0.90625 0.893   0.9    ]
[0.8875     0.887      0.87333333]
[0.8975 0.878  0.88  ]
[0.8975     0.875      0.89333333]
[0.87875 0.89    0.885  ]
[0.86625 0.886   0.8775 ]
[0.88625    0.889      0.87833333]
[0.8875     0.896      0.88083333]
[0.8975     0.885      0.88166667]
[0.9075     0.882      0.89416667]
[0.875      0.892      0.89416667]
Case: test size = 0.2
0.8871249999999999
Case: test size = 0.25
0.8861600000000002
Case: test size = 0.3
0.8884999999999998
```

4. Hyper parameter tuning

RandomizedSearchCV와 GridSearchCV를 통해 하이퍼 파라미터 튜닝을 진행했다. 이 부분은 수행시간이 굉장히 오래걸려서 Spyder로 진행 중이던 개발을 Colab으로 옮겼다. 수행시간에 큰 차이가 있지는 않았지만 Colab은 진행경과를 표시해주어서 어느정도 수행시간을 예측할 수 있었기 때문이다.

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import model_selection
from sklearn.model_selection import RandomizedSearchCV


from sklearn.feature_selection import RFE

trainset                                                              =
pd.read_csv('C:/Users/82104/Desktop/deeplearning/competition/train
_open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()
model = RandomForestClassifier(random_state=42)
train_x = scaler.fit_transform(train_x)

rfe = RFE(model, n_features_to_select = 10)
fit = rfe.fit(train_x, train_y)
temp = fit.support_.tolist()
fs = [idx for idx, x in enumerate(temp) if x==True]
print("Number of features: " +str(fit.n_features_))
print("Features: ", end="")
print(fs)
train_x = train_x[:, fs]

train_x_all = train_x
train_y_all = train_y

train_x, test_x, train_y, test_y = train_test_split(train_x_all,
train_y_all, test_size=0.3)

n_estimators = [int(x) for x in np.linspace(start = 100, stop =
3000, num = 30)]
```

```
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 300, num = 30)]
max_depth.append(None)
min_samples_split = [2, 3, 5, 7, 8, 10]
min_samples_leaf = [1, 2, 3, 4, 5]
bootstrap = [True, False]
random_grid = {'n_estimators': n_estimators,
 'max_features': max_features,
 'max_depth': max_depth,
 'min_samples_split': min_samples_split,
 'min_samples_leaf': min_samples_leaf,
 'bootstrap': bootstrap
}

base_model = RandomForestClassifier(random_state=42)
base_model.fit(train_x, train_y)
base_accuracy = base_model.score(test_x, test_y)

rf_random = RandomizedSearchCV(estimator = base_model,
param_distributions = random_grid, n_iter = 100, cv = 10, verbose=2,
random_state=42, n_jobs = -1)
rf_random.fit(train_x, train_y)
print(rf_random.best_params_)
best_random_model = rf_random.best_estimator_
best_random_accuracy = best_random_model.score(test_x, test_y)
print('base acc: {0:0.5f}. best acc:
{1:0.5f}'.format(base_accuracy, best_random_accuracy))
print('Improvement of {:0.5f}%.'.format(100 * (best_random_accuracy
- base_accuracy) / base_accuracy))
```

먼저 RandomizeSearchCV를 통해 하이퍼 파라미터 튜닝을 진행했다. 처음 100번의 fit을 진행했을 때는 오히려 기본 accuracy보다 결과가 좋지 않게 나왔다. 그래서 1000번의 fit을 진행했더니 0.09% accuracy가 향상했다.

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
{'n_estimators': 2500, 'min_samples_split': 7, 'min_samples_leaf': 2, 'max_features': 'sqrt',
'max_depth': 210, 'bootstrap': True}
base acc: 0.89833. best acc: 0.89167
Improvement of -0.74212%.
```

```
Fitting 10 folds for each of 100 candidates, totalling 1000 fits
{'n_estimators': 600, 'min_samples_split': 10, 'min_samples_leaf': 2, 'max_features': 'auto',
'max_depth': 30, 'bootstrap': False}
base acc: 0.88833. best acc: 0.88917
Improvement of 0.09381%.
```

매개변수 값을 랜덤하게 추출하는 범위가 너무 넓어서 accuracy의 향상이 좋지 않은 것이라고 추측되어 어느정도 범위를 조정한 후 다시 랜덤하게 추출하기로 결정했다. GridSearchCV를 통해 하이퍼 파라미터 튜닝을 진행했다. 하이퍼 파라미터에 따른 accuracy 순위를 오름차순으로 정렬한 뒤 csv로 받았다.

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import model_selection
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV



from sklearn.feature_selection import RFE

trainset = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/train_
open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()
model = RandomForestClassifier(random_state=42)
train_x = scaler.fit_transform(train_x)

rfe = RFE(model, n_features_to_select = 10)
fit = rfe.fit(train_x, train_y)
temp = fit.support_.tolist()
fs = [idx for idx, x in enumerate(temp) if x==True]
print("Number of features: " +str(fit.n_features_))
print("Features: ", end="")
print(fs)
train_x = train_x[:, fs]

train_x_all = train_x
train_y_all = train_y
```

```
train_x, test_x, train_y, test_y = train_test_split(train_x_all, trai
n_y_all, test_size=0.3)

n_estimators = [int(x) for x in np.linspace(start = 100, stop = 3000,
 num = 30)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 300, num = 30)]
max_depth.append(None)
min_samples_split = [2, 3, 5, 7, 8, 10]
min_samples_leaf = [1, 2, 3, 4, 5]
bootstrap = [True, False]

default_model = RandomForestClassifier(random_state=42)
default_model.fit(train_x, train_y)
default_accuracy = default_model.score(test_x, test_y)


param_grid = {
    'n_estimators': [1000, 2000],
    'max_depth': [50, 100],
    'min_samples_split': [2, 5, 10],
    'max_features': ['auto', 'sqrt'],
    'min_samples_leaf' : [1, 2, 3],
    'bootstrap':[True, False]
}

grid_search = GridSearchCV(estimator=default_model, param_grid=param_
grid, cv=5, n_jobs= -1, scoring='f1_weighted', verbose=10)

grid_search.fit(train_x, train_y)

print(grid_search.best_params_)
best_random_model = grid_search.best_estimator_
best_random_accuracy = best_random_model.score(test_x, test_y)
print('Default acc: {0:0.5f}. Best acc: {1:0.5f}'.format(default_accu
racy, best_random_accuracy))
print('Improvement of {:0.5f}%.'.format(100 * (best_random_accuracy -
 default_accuracy) / default_accuracy))
scores = pd.DataFrame(grid_search.cv_results_)
print(scores[['params','mean_test_score', 'rank_test_score']])

scores.sort_values(by=['rank_test_score'], axis=0, inplace=True)
scores.to_csv('sorted_scores.csv')
```

```
{'bootstrap': False, 'max_depth': 50, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 2000}
Default acc: 0.87917. Best acc: 0.87500
Improvement of -0.47393%.
                                                params  ...  rank_test_score
0    {'bootstrap': True, 'max_depth': 50, 'max_feat...  ...               25
1    {'bootstrap': True, 'max_depth': 50, 'max_feat...  ...               57
2    {'bootstrap': True, 'max_depth': 50, 'max_feat...  ...               81
3    {'bootstrap': True, 'max_depth': 50, 'max_feat...  ...               13
4    {'bootstrap': True, 'max_depth': 50, 'max_feat...  ...               33
..                                                 ...  ...              ...
139  {'bootstrap': False, 'max_depth': 100, 'max_fe...  ...               85
140  {'bootstrap': False, 'max_depth': 100, 'max_fe...  ...               61
141  {'bootstrap': False, 'max_depth': 100, 'max_fe...  ...               85
142  {'bootstrap': False, 'max_depth': 100, 'max_fe...  ...               53
143  {'bootstrap': False, 'max_depth': 100, 'max_fe...  ...               77

[144 rows x 3 columns]
```

rank_test_	param_bootstrap	param_max_depth	param_max_features	param_mi	param_min_samples_split	param_n_estimators
1	FALSE	100	sqrt	1	10	2000
1	FALSE	50	auto	1	10	2000
1	FALSE	50	sqrt	1	10	2000
1	FALSE	100	auto	1	10	2000
5	FALSE	50	auto	1	10	1000
5	FALSE	100	auto	1	10	1000
5	FALSE	100	sqrt	1	10	1000
5	FALSE	50	sqrt	1	10	1000
9	FALSE	50	auto	1	5	2000
9	FALSE	100	auto	1	5	2000
9	FALSE	100	sqrt	1	5	2000
9	FALSE	50	sqrt	1	5	2000
13	TRUE	50	sqrt	1	5	2000
13	TRUE	100	auto	1	5	2000
13	TRUE	100	sqrt	1	5	2000
13	TRUE	50	auto	1	5	2000
17	FALSE	100	sqrt	1	2	2000
17	FALSE	50	sqrt	1	2	2000
17	FALSE	100	auto	1	2	2000
17	FALSE	50	auto	1	2	2000
21	FALSE	100	sqrt	2	10	1000

5. 모델 평가

주어진 test set으로 모델 평가를 진행하면서 하이퍼 파라미터 튜닝을 반복했다. 예측 결과를 csv로 받았다.

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import model_selection
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV



from sklearn.feature_selection import RFE

trainset = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/train_
open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

testset = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/test_op
en.csv')

print(testset)

scaler = MinMaxScaler()
scaler.fit(train_x)
train_x = scaler.transform(train_x)
testset = scaler.transform(testset)
print(testset)

model = RandomForestClassifier(random_state=42)

rfe = RFE(model, n_features_to_select = 10)
fit = rfe.fit(train_x, train_y)
temp = fit.support_.tolist()
fs = [idx for idx, x in enumerate(temp) if x==True]
```

```
print("Number of features: " +str(fit.n_features_))
print("Features: ", end="")
print(fs)
# print(train_x)
train_x = train_x[:, fs]
# print(train_x, test_x.iloc[:, fs])
testset = testset[:, fs]
print(train_x, train_y)

model = RandomForestClassifier(bootstrap=False, max_depth=100, max_fe
atures='sqrt', min_samples_split=10, min_samples_leaf=1, n_estimators
 = 2000, random_state=1234)
model.fit(train_x, train_y)

def save_submission2(pred):
 file_name = "32191197_김채은.csv"
 np.savetxt(file_name, pred, fmt="%s", delimiter=",")

pred = model.predict(testset)
submission = pd.DataFrame(pred)
save_submission2(submission)
```

	A	B	C	D
1	answer			
2	2			
3	2			
4	1			
5	1			
6	2			
7	1			
8	1			
9	2			
10	2			
11	1			
12	2			

위와 같은 결과가 나왔다.

RandomizedSearchCV를 통한 하이퍼 파라미터 튜닝을 다시 진행했다.

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import model_selection
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV


from sklearn.feature_selection import RFE

trainset = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/train_
open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

scaler = MinMaxScaler()
model = RandomForestClassifier(random_state=42)
train_x = scaler.fit_transform(train_x)

rfe = RFE(model, n_features_to_select = 10)
fit = rfe.fit(train_x, train_y)
temp = fit.support_.tolist()
fs = [idx for idx, x in enumerate(temp) if x==True]
print("Number of features: " +str(fit.n_features_))
print("Features: ", end="")
print(fs)
train_x = train_x[:, fs]

train_x_all = train_x
train_y_all = train_y

train_x, test_x, train_y, test_y = train_test_split(train_x_all, trai
n_y_all, test_size=0.3)

n_estimators = [int(x) for x in np.linspace(start = 100, stop = 3000,
 num = 30)]
```

```
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 300, num = 30)]
max_depth.append(None)
min_samples_split = [2, 3, 5, 7, 8, 10]
min_samples_leaf = [1, 2, 3, 4, 5]
bootstrap = [True, False]
random_grid = {'n_estimators': n_estimators,
 'max_features': max_features,
 'max_depth': max_depth,
 'min_samples_split': min_samples_split,
 'min_samples_leaf': min_samples_leaf,
 'bootstrap': bootstrap
}

default_model = RandomForestClassifier(random_state=42)
default_model.fit(train_x, train_y)
default_accuracy = default_model.score(test_x, test_y)

param_grid = {
    'n_estimators': [2000, 2500],
    'max_depth':  [int(x) for x in np.linspace(50, 110, num = 11)] ,
    'min_samples_split': [10, 15],
    'max_features': ['sqrt'],
    'min_samples_leaf' : [1],
    'bootstrap':[False]
}

rf = RandomForestClassifier(random_state=1234)
rf_random = RandomizedSearchCV(estimator = rf, param_distributions =
random_grid, n_iter = 100,
cv = 10, verbose=2, random_state=42, n_jobs = -1)
rf_random.fit(train_x, train_y)


best_random_model = rf_random.best_estimator_
best_random_accuracy = best_random_model.score(test_x, test_y)

print('Default acc: {0:0.5f}. Best acc: {1:0.5f}'.format(default_accu
racy, best_random_accuracy))
print('Improvement of {:0.5f}%.'.format(100 * (best_random_accuracy -
 default_accuracy) / default_accuracy))
scores = pd.DataFrame(rf_random.cv_results_)
print(scores[['params','mean_test_score', 'rank_test_score']])

scores.sort_values(by=['rank_test_score'], axis=0, inplace=True)
scores.to_csv('sorted_scores.csv')
```

param_n_estimators	param_min_samples_	param_min_samp	param_max_features	param_max_depth	param_bootstrap
3000	3	1	sqrt	100	TRUE
2700	7	1	sqrt	10	TRUE
3000	7	2	auto	250	FALSE
100	2	1	sqrt	140	TRUE
600	8	1	auto	60	FALSE
2200	3	1	sqrt	180	TRUE

하이퍼 파라미터 값에 따른 accuracy를 표로 정리했다.

n_esimators	min_samples_split	min_samples_leaf	max_features	max_depth	bootstrap	accuracy
3000	3	1	Sqrt	100	True	0.91
2700	7	1	Sqrt	10	True	0.92
3000	7	2	Auto	250	False	0.91
100	2	1	Sqrt	140	True	0.91
600	8	1	Auto	60	False	0.91
2200	3	1	Sqrt	180	True	0.91

하이퍼 파라미터 값이 바뀜에 따라 feature selection에 따른 결과도 달라질 수 있을 것 같았다.

그래서 가장 accuracy가 높았던 하이퍼 파라미터에, 모든 feature를 선택하여 다시 테스트 해보았다.

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import model_selection
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV


from sklearn.feature_selection import RFE
```

```
trainset = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/train_
open.csv')

train_x = trainset.iloc[:, 0:22]
train_y = trainset.iloc[:, -1]

testset = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/test_op
en.csv')

print(testset)

scaler = MinMaxScaler()
scaler.fit(train_x)
train_x = scaler.transform(train_x)
testset = scaler.transform(testset)
print(testset)

model = RandomForestClassifier(random_state=42)

rfe = RFE(model, n_features_to_select = 22)
fit = rfe.fit(train_x, train_y)
temp = fit.support_.tolist()
fs = [idx for idx, x in enumerate(temp) if x==True]
print("Number of features: " +str(fit.n_features_))
print("Features: ", end="")
print(fs)
# print(train_x)
train_x = train_x[:, fs]
# print(train_x, test_x.iloc[:, fs])
testset = testset[:, fs]
print(train_x, train_y)

model = RandomForestClassifier(bootstrap=True, max_depth=10, max_feat
ures='sqrt', min_samples_split=7, min_samples_leaf=1, n_estimators =
2700, random_state=1234)
model.fit(train_x, train_y)

def save_submission2(pred):
 file_name = "32191197_김채은.csv"
 np.savetxt(file_name, pred, fmt="%s", delimiter=",")

pred = model.predict(testset)
submission = pd.DataFrame(pred)
save_submission2(submission)
```

놀랍게도 지금까지의 결과 중 가장 높은 값이 나왔다. 모든 feature를 선택하고 하이퍼 파라미터 튜닝을 다시 진행했다. 다음은 1위부터 4위까지의 테스트 결과이다.

n_estimators	min_samples_split	min_samples_leaf	max_features	Max_depth	bootstrap	accuracy
1400	2	1	sqrt	180	false	0.92
2300	3	1	sqrt	120	false	0.92
600	8	1	auto	60	false	0.92
2100	2	1	auto	60	false	0.92
2500	3	1	auto	170	false	0.92

모든 feature selection에 대해 하이퍼 파라미터 튜닝을 진행할 시간이 없어서 이대로 마무리 지었다. 다음은 나의 테스트 로그이다.

My test log

date	sid	acc
2021-10-22	32191197	0.91
2021-10-24	32191197	0.91
2021-10-24	32191197	0.92
2021-10-24	32191197	0.91
2021-10-24	32191197	0.91
2021-10-24	32191197	0.91
2021-10-24	32191197	0.91
2021-10-24	32191197	0.92
2021-10-24	32191197	0.91
2021-10-25	32191197	0.92
2021-10-25	32191197	0.92
2021-10-25	32191197	0.92
2021-10-25	32191197	0.92
2021-10-25	32191197	0.92
2021-10-25	32191197	0.92

■ 경진대회 참여소감

딥러닝/클라우드를 수강하면서 매주 많은 양의 과제가 제공되었다. 과제를 할 당시에는 지치고 힘들었는데, 경진대회를 위한 모델을 개발하면서 과제를 하면서 실습해보았던 내용을 많이 이용할 수 있었다. 과제를 통해 미리 실습해보지 않았다면 이번 경진대회를 진행하는 데 훨씬 많은 시간이 소요되고 진행하기가 굉장히 어려웠을 것 같다.

처음에는 accuracy가 너무 낮게 나와 막막했는데, 진행 과정을 거쳐가면서 조금씩 조금씩 향상되는 accuracy를 보는 게 큰 재미였던 것 같다. 과제를 진행할수록 어떻게 하면 좋은 예측 결과를 낼 수 있을지, 더 많이 고민하게 되고 열정이 생겼던 것 같다. 특히 마지막 단계에서 feature selection과 hyper parameter tuning을 반복 진행했으면 더 높은 accuracy를 찾을 수 있었을 것 같은데, 시간적인 한계와 업로드 수 제한으로 모두 테스트해보지 못한 것이 아쉽다.

딥러닝 모델을 가지고 accuracy를 테스트해본 경험은 있었지만, feature를 선택하는 것부터 하이퍼 파라미터 튜닝까지 전 과정을 진행해본 것은 이번이 처음이다. 결과가 어떻게 나올지 기다리면서, 오랜만에 가슴을 두근거리게 만드는 개발 과제를 할 수 있어 기뻤다.