Jane Street Market Prediction

2017010715허지혜

1. 대회 소개

Jane Street

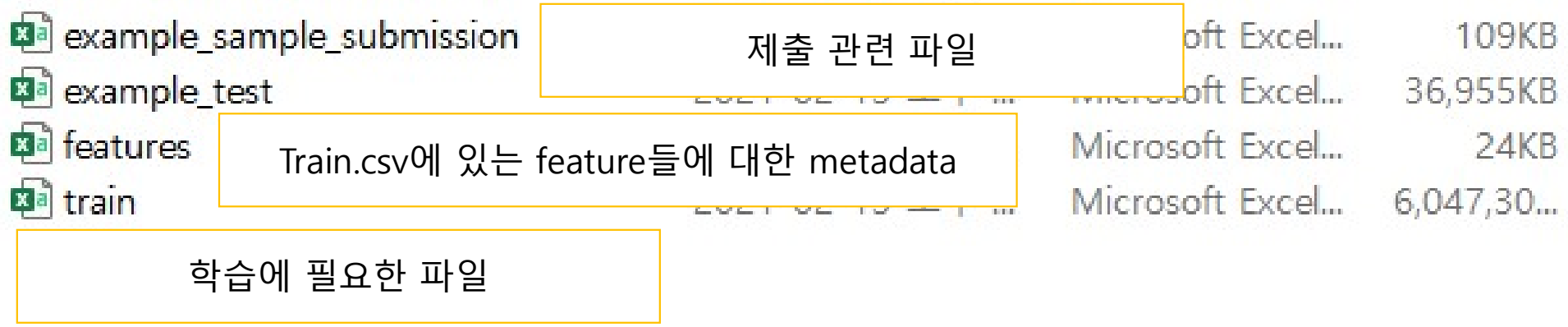
뉴욕, 런던, 홍콩, 암스테르담에 사무소를 두고 있고 전 세계에서 운영하고 있는 회사이다.
1200명 정도의 직원을 보유한 글로벌 독점 무역 회사이다.
ETC(주식,선물,상품,옵션,채권,통화) 와 같은 다양한 금융 상품을 거래한다.
거래를 할 때 회사 안의 독점 모델을 기반으로 한다.

1. 대회 소개

실제 시장 결과를 토대로 얻은 여러 FEATURE들로 되어있는 데이터를 보고
1(받다)/0(거절하다) 를 결정해서 최대 수익을 내는 자체 거래 모델을 만드는 문제

=> 이진 분류

2. 데이터셋 확인


example_sample_submission
example_test
features
train
제출 관련 파일
Train.csv에 있는 feature들에 대한 metadata
학습에 필요한 파일
Microsoft Excel...
109KB
36,955KB
24KB
6,047,30...

2. 데이터셋 확인

	date	weight	resp_1	resp_2	resp_3	resp_4	resp	feature_0	feature_
0	0	0.000000	0.009916	0.014079	0.008773	0.001390	0.006270	1	-1.8727
1	0	16.673515	-0.002828	-0.003226	-0.007319	-0.011114	-0.009792	-1	-1.3495
2	0	0.000000	0.025134	0.027607	0.033406	0.034380	0.023970	-1	0.8127
3	0	0.000000	-0.004730	-0.003273	-0.000461	-0.000476	-0.003200	-1	1.1743
4	0	0.138531	0.001252	0.002165	-0.001215	-0.006219	-0.002604	1	-3.1720
...	...	...	...	...	...	...	...	...	
2390486	499	0.000000	0.000142	0.000142	0.005829	0.020342	0.015396	1	-1.6493
2390487	499	0.000000	0.000012	0.000012	-0.000935	-0.006326	-0.004718	1	2.4329
2390488	499	0.000000	0.000499	0.000499	0.007605	0.024907	0.016591	1	-0.6224
2390489	499	0.283405	-0.000156	-0.000156	-0.001375	-0.003702	-0.002004	-1	-1.4637
2390490	499	0.000000	-0.001855	-0.001855	-0.001194	-0.000864	-0.001905	-1	-1.8171

2390491 rows × 138 columns

Date : 날짜
Weight : 얼마나 넣을지 가중치
Resp_{1,2,3,4} : time horizon
Features_{0,...,129} : 익명화 됨

```
df_train.columns
```

```
Index(['date', 'weight', 'resp_1', 'resp_2', 'resp_3', 'resp_4', 'resp',
       'feature_0', 'feature_1', 'feature_2',
       ...
       'feature_121', 'feature_122', 'feature_123', 'feature_124',
       'feature_125', 'feature_126', 'feature_127', 'feature_128',
       'feature_129', 'ts_id'],
      dtype='object', length=138)
```

2. 데이터셋 확인

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2390491 entries, 0 to 2390490
Columns: 139 entries, date to action
dtypes: float64(135), int64(4)
memory usage: 2.5 GB
```

```
df_train.describe()
```

	date	weight	resp_1	resp_2	resp_3	resp_4	
count	2.390491e+06	2.390491e+06	2.390491e+06	2.390491e+06	2.390491e+06	2.390491e+06	2.39
mean	2.478668e+02	3.031535e+00	1.434969e-04	1.980749e-04	2.824183e-04	4.350201e-04	4.08
std	1.522746e+02	7.672794e+00	8.930163e-03	1.230236e-02	1.906882e-02	3.291224e-02	2.69
min	0.000000e+00	0.000000e+00	-3.675043e-01	-5.328334e-01	-5.681196e-01	-5.987447e-01	-5.
25%	1.040000e+02	1.617400e-01	-1.859162e-03	-2.655044e-03	-5.030704e-03	-9.310415e-03	-7.
50%	2.540000e+02	7.086770e-01	4.552665e-05	6.928179e-05	1.164734e-04	1.222579e-04	8.63
75%	3.820000e+02	2.471791e+00	2.097469e-03	2.939111e-03	5.466336e-03	9.804649e-03	7.54
max	4.990000e+02	1.672937e+02	2.453477e-01	2.949339e-01	3.265597e-01	5.113795e-01	4.48

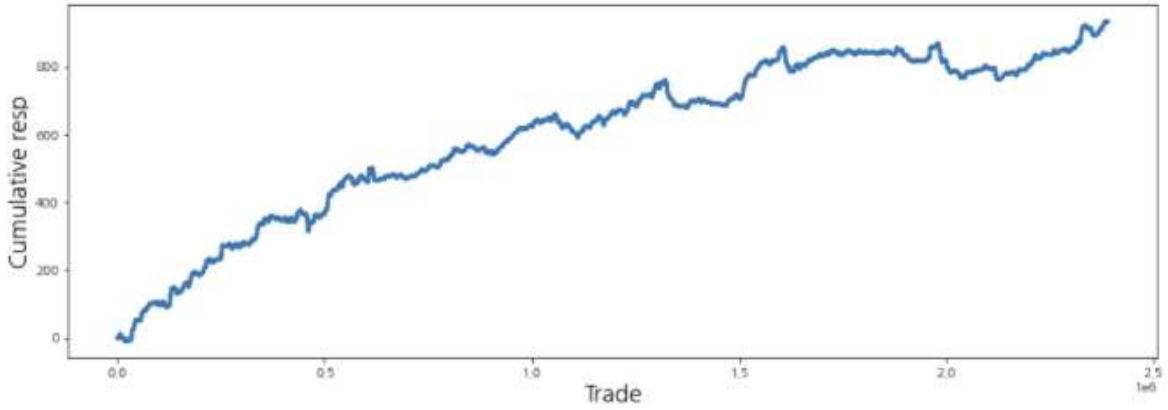
8 rows × 139 columns

3. 탐색적 데이터 분석

Resp

```
fig, ax = plt.subplots(figsize=(15,5))
balance = pd.Series(df_train['resp']).cumsum() # 누적합
ax.set_xlabel("Trade", fontsize=18)
ax.set_ylabel("Cumulative resp", fontsize=18)
balance.plot(lw=3)
```

```
<AxesSubplot:xlabel='Trade', ylabel='Cumulative resp'>
```

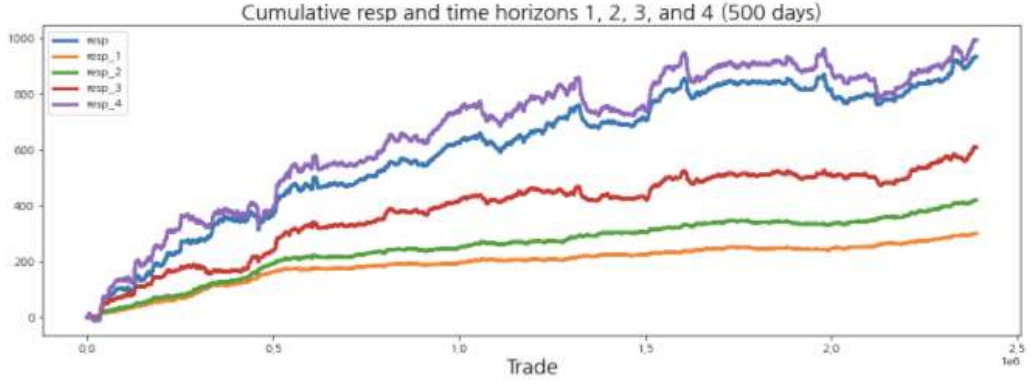


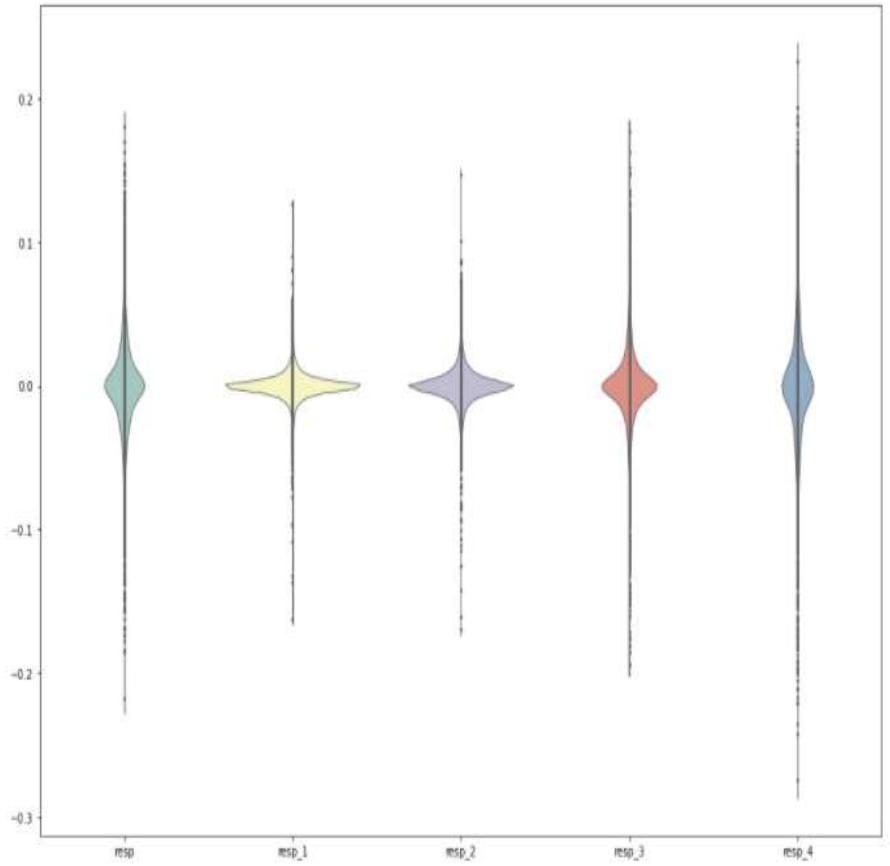
Resp*weight =return이라는 설명을 보아 가격 변동률 을 나타냄 => 종속 변수 나눌 때 기준

3. 탐색적 데이터 분석

Resp_{1,2,3,4}

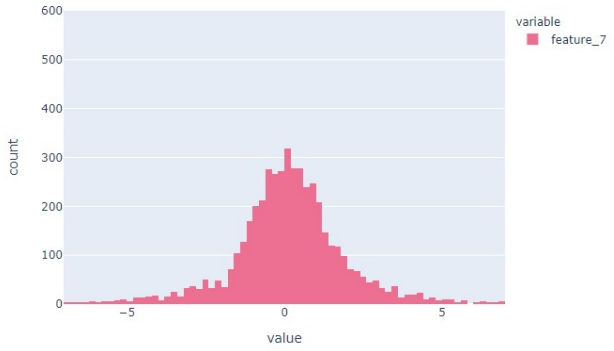
```
fig, ax = plt.subplots(figsize=(15, 5))
balance= pd.Series(df_train['resp']).cumsum()
resp_1= pd.Series(df_train['resp_1']).cumsum()
resp_2= pd.Series(df_train['resp_2']).cumsum()
resp_3= pd.Series(df_train['resp_3']).cumsum()
resp_4= pd.Series(df_train['resp_4']).cumsum()
ax.set_xlabel ("Trade", fontsize=18)
ax.set_title ("Cumulative resp and time horizons 1, 2, 3, and 4 (500 days)", fontsiz
balance.plot(lw=3)
resp_1.plot(lw=3)
resp_2.plot(lw=3)
resp_3.plot(lw=3)
resp_4.plot(lw=3)
plt.legend(loc="upper left");
```

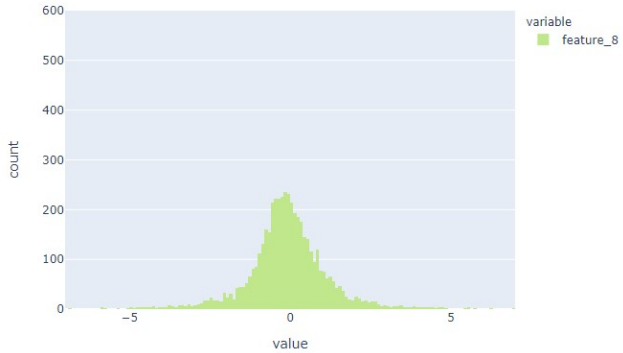


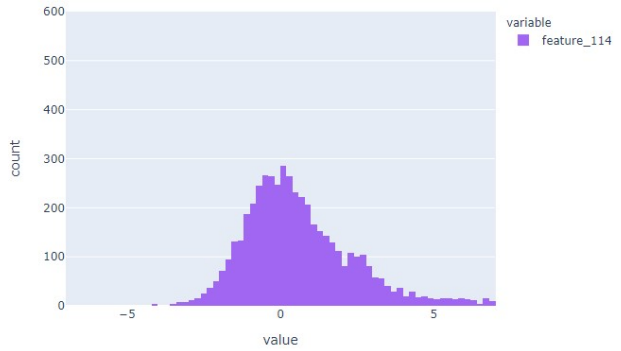


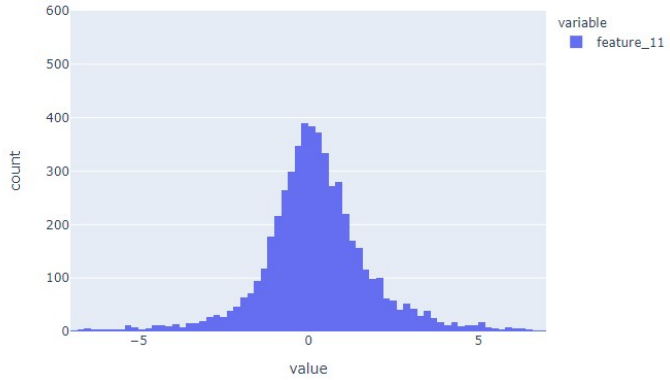
3. 탐색적 데이터 분석

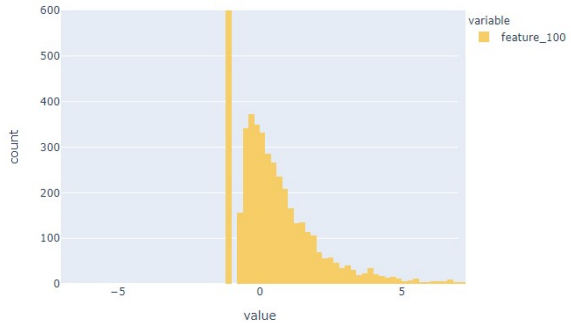
Features_{0,...,129}

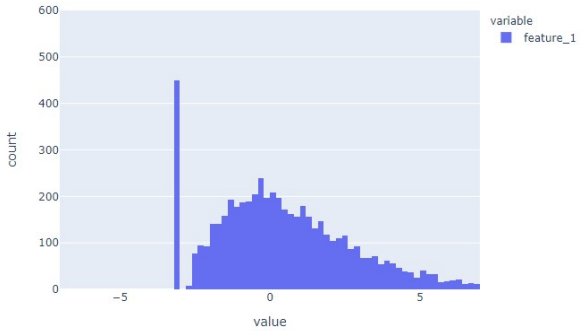








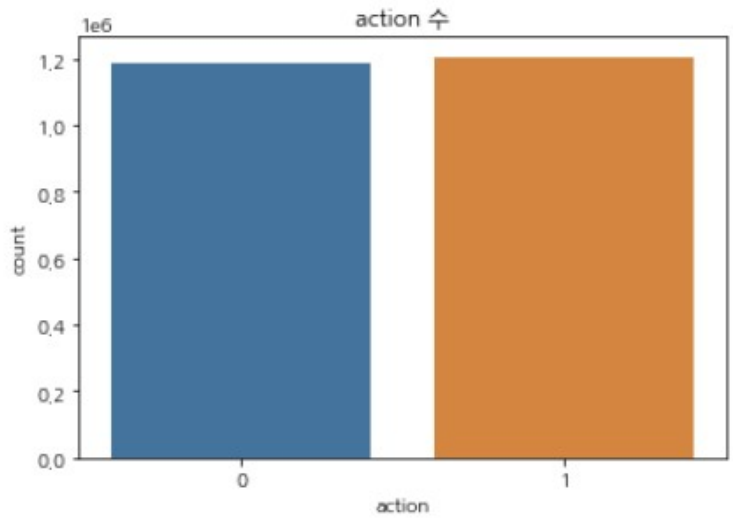




3. 탐색적 데이터 분석

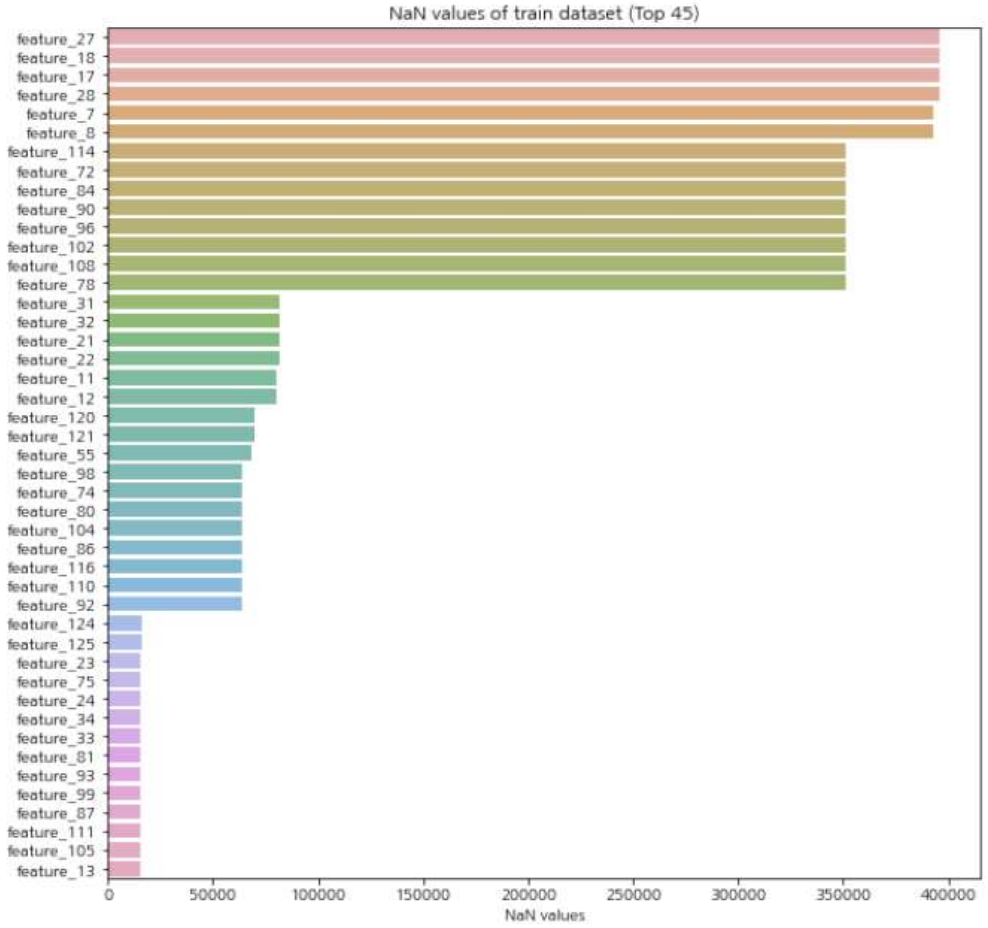
action

```
# 예측하려는 값이 반반이니 굳이 데이터를 생성하지 않아도 될듯될듯
import seaborn as sns
import matplotlib.pyplot as plt
plt.rcParams['font.family'] = 'NanumGothic'
sns.countplot(x='action',data=df_train)
plt.title("action 수")
plt.show()
```




```
df_train['action'] = 0
df_train.loc[(df_train['resp']>0),'action']=1
```

3. 탐색적 데이터 분석



결측치가 존재

=> df_train.fillna(df_train.mean(), inplace=True)

```
date            0
weight          0
resp_1          0
resp_2          0
resp_3          0
                ..
feature_127     0
feature_128     0
feature_129     0
ts_id           0
action          0
Length: 139, dtype: int64
```

3. 탐색적 데이터 분석

Weignt가 0인 거래는 완전성을 위해 의도적으로 포함 시켰지만 기여하지 않는다.

=> df_train = df_train[df_train['weight'] != 0]

	date	weight	resp_1	resp_2	resp_3	resp_4	resp	feature_0	feature_
1	0	16.673515	-0.002828	-0.003226	-0.007319	-0.011114	-0.009792	-1	-1.3495
4	0	0.138531	0.001252	0.002165	-0.001215	-0.006219	-0.002604	1	-3.1720
6	0	0.190575	-0.001939	-0.002301	0.001088	0.005963	0.000709	-1	-3.1720
7	0	3.820844	0.017395	0.021361	0.031163	0.036970	0.033473	-1	0.4460
8	0	0.116557	-0.005460	-0.007301	-0.009085	-0.003546	-0.001677	1	-3.1720
...	...	...	...	...	...	...	...	...	
2390444	499	56.694795	0.001607	0.001607	-0.001245	-0.012068	-0.010023	-1	1.5386
2390446	499	1.650055	0.004523	0.004523	0.003172	-0.013886	-0.013637	1	0.2703
2390478	499	0.895142	0.000486	0.000486	-0.004090	-0.008105	-0.005441	-1	-0.1343
2390481	499	2.967272	0.000298	0.000298	-0.005393	-0.012472	-0.006681	-1	-0.7795
2390489	499	0.283405	-0.000156	-0.000156	-0.001375	-0.003702	-0.002004	-1	-1.4637

1981287 rows × 139 columns

3. 탐색적 데이터 분석

독립 변수 : feature_{0,...,128}
종속 변수 : 0 아니면 1

4. 모델링

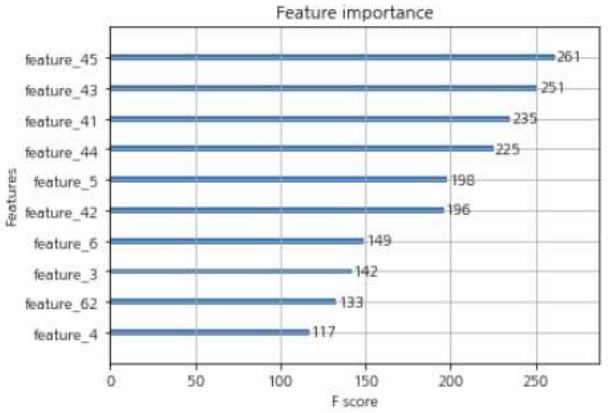
1) XGBoost

```
clf.fit(X_train, y_train)
```

```
C:₩Users₩hu612₩Anaconda3₩lib₩site-packages₩xgboost₩sklearn.py:888: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[01:34:13] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=8, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
from xgboost import plot_importance
plot_importance(clf,max_num_features=10)
```

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



4. 모델링

2) 의사결정나무

모형 요약

지정 사항	성장방법	CHAID
	종속변수	action
	독립변수	feature_0, feature_1, feature_2, feature_3, feature_4, feature_5, feature_6, feature_7, feature_8, feature_9, feature_10, feature_11, feature_12, feature_13, feature_14, feature_15, feature_16, feature_17, feature_18, feature_19, feature_20, feature_21, feature_22, feature_23, featur feature_25, featur feature_27, featur feature_29, featur feature_31, featur feature_33, featur feature_35, featur feature_37, featur feature_39, featur feature_41, featur feature_43, featur feature_45, featur feature_47, featur feature_49, featur feature_51, featur feature_53, featur feature_55, featur
		feature_127, feature_128, feature_129
	검증	교차 검증
	최대 나무 깊이	3
	부모 노드의 최소 케이스	100
	자식 노드의 최소 케이스	50
결과	독립변수 포함	feature_35, feature_16, feature_44
	노드 수	7
	터미널 노드 수	4
	깊이	2

분류

관측	예측 0	예측 1	정확도 퍼센트
0	111	53	67.7%
1	58	107	64.8%
전체 퍼센트	51.4%	48.6%	66.3%

성장방법: CHAID
종속변수: action

111과 107 case가 정확하게 분류되었고 전체 정확도는 66.3% 이다.

4. 모델링

3) 로지스틱 회귀

방정식의 변수

		B	S.E.	Wald	자유도	유의확률	Exp(B)
0 단계	상수항	.018	.001	154.338	1	.000	1.018

모형 요약

단계	-2 로그 우도	·와 Snell의 제곱	Nagelkerke R-제곱
1	2739684.71[a]	.003	.005

a. 모수 추정값이 .001보 에 변경되어 계산반복수

블록 방법 : 시작

분류표[a,b]

			예측 action		
	관측됨		0	1	분류정확 %
0 단계	action	0	0	981900	.0
		1	0	999387	100.0
	전체 퍼센트				50.4

a. 모형에 상수항이 있습니다.

b. 절단값은 .500입니다.

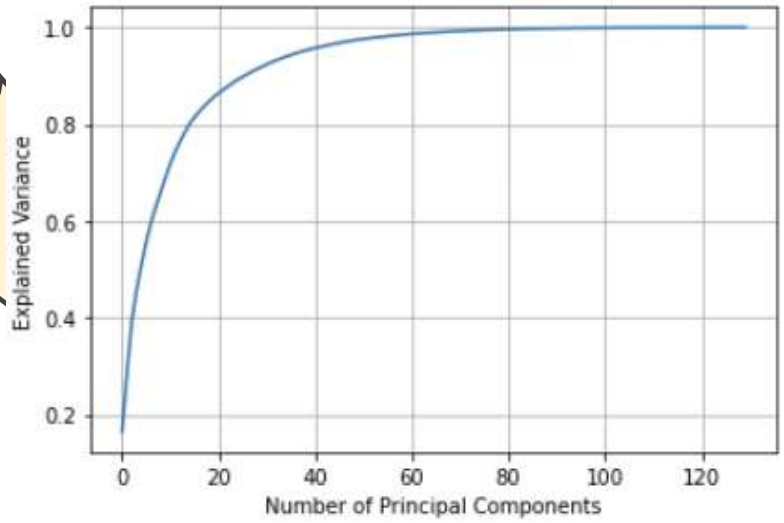
분류표[a]

			예측 action		
	관측됨		0	1	분류정확 %
1 단계	action	0	138	26	84.1
		1	27	138	83.6
	전체 퍼센트				83.9

a. 절단값은 .500입니다.

4. 모델링

PCA




```
import numpy as np
cumsum = np.cumsum(pca.explained_variance_ratio_)
d = np.argmax(cumsum >= 0.95) + 1
print('선택할 차원 수 :', d)
```

선택할 차원 수 : 39

```
pca.explained_variance_ratio_
```

```
array([0.16511826, 0.1175708 , 0.11000283, 0.06565214, 0.05103314,
       0.04739297, 0.04076871, 0.03284667, 0.03051317, 0.02984129,
       0.02802329, 0.02438513, 0.02118943, 0.01939493, 0.01773622,
       0.01333488, 0.01187031, 0.01094993, 0.00962451, 0.00941349,
       0.00787931, 0.00757311, 0.00712786, 0.00683244, 0.0062854 ,
       0.00605761, 0.00567961, 0.0052402 , 0.00517126, 0.00467653,
       0.00461633, 0.00445658, 0.00412807, 0.00400131, 0.00367621,
       0.0034522 , 0.00332541, 0.00302417, 0.00272071])
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=0.95) # 95% 이상의 분산의 설명력을 갖는 차원 축소
pc = pca.fit_transform(X_)
```

```
X_1 = pd.DataFrame(pc)
X_1
```

	0	1	2	3	4	5	6	7	8	9	...	29	30	31	
0	-1.709433	-1.663395	0.567048	4.475464	2.124899	-0.473302	0.735927	0.774307	1.005804	-1.794576	...	0.097861	0.422965	0.578824	0
1	1.277469	-1.267713	0.867440	6.089463	0.320856	1.597281	-0.760054	-1.187987	-1.223627	-0.997223	...	-0.585504	-0.815827	0.585593	1
2	-4.637392	-2.025102	3.682530	6.089639	-1.090982	0.263074	0.352546	-2.941379	-0.427533	-2.172190	...	0.213968	0.098593	-0.706036	0
3	-2.923084	-1.433641	-2.711706	6.235479	-1.637121	-1.125479	-1.124936	1.417878	1.412773	-1.943002	...	0.954591	0.241888	-1.724264	0
4	0.796217	-5.187286	0.997004	6.887823	-3.286283	-0.004450	-1.386140	-2.611509	-0.745652	-0.764889	...	0.068187	-0.497145	-0.264170	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1981282	-3.253631	1.860364	4.978060	-1.976941	3.723683	1.039812	-0.455548	3.340591	5.803692	-0.114286	...	0.651736	0.869918	0.691915	-1
1981283	6.816826	2.790268	-0.303738	-3.123044	-2.904719	2.595225	-0.372002	1.604832	3.901712	2.340397	...	-0.217093	-0.375996	0.314315	0
1981284	-2.470101	-0.590523	1.005870	-4.145536	0.032018	1.013273	0.576703	-2.638471	3.558125	1.180044	...	-0.131717	0.342199	0.072717	0
1981285	-3.468934	-0.599246	-0.263997	-4.226054	-0.589807	1.347740	-0.806845	-2.330167	2.051927	2.288797	...	0.314696	0.412427	-0.255663	-0
1981286	-0.004525	5.967921	15.477562	-3.504058	2.751327	-8.575804	3.393580	2.233357	-0.972393	-1.102197	...	-0.530141	0.282473	-0.045390	-1

1981287 rows × 39 columns

4. 모델링

XGBoost

```
from sklearn.model_selection import train_test_split
y = data['action']
X_train, X_test, y_train, y_test = train_test_split(X_1,y,test_size=0.2)
```

```
print(X_train.shape, y_train.shape)
```

```
(1306504, 39) (1306504,)
```

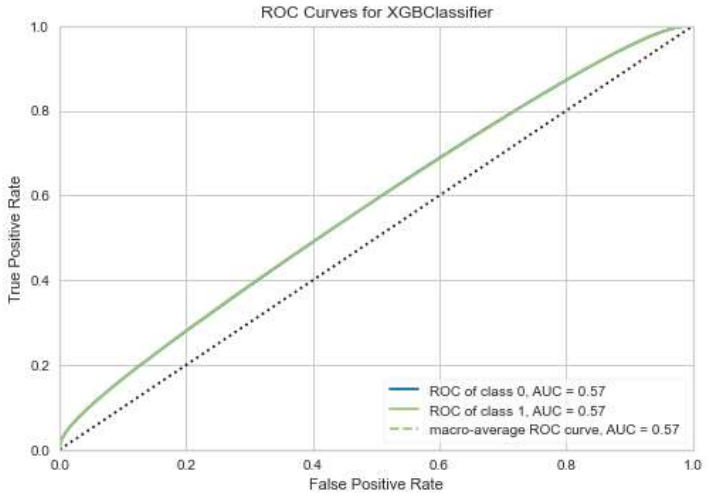
```
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(X_train, y_train)
```

```
C:₩Users₩home₩anaconda3₩lib₩site-packages₩xgboost₩sklearn.py:888: UserWarning: The use of label encoder in XG
will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoc
BClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[03:05:02] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Startin
lt evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explici
like to restore the old behavior.

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=4, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
print(xgb.score(X_train,y_train))
```

```
0.5438575006276292
```

```
from yellowbrick.classifier import ROCAUC
visualizer = ROCAUC(xgb, classes=[0, 1], micro=False, macro=True, per_class=True)
visualizer.fit(X_train, y_train)
visualizer.score(X_train, y_train)
visualizer.show()
```



4. 결론

아쉬운 점

```
from ngboost import NGBClassifier
from ngboost.distns import k_categorical, Bernoulli
from sklearn.datasets import load_breast_cancer

ngb_cat = NGBClassifier(Dist=k_categorical(3), verbose=False)
ngb_cat.fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
<ipython-input-17-4a60b5bdd2a6> in <module>
      4
      5 ngb_cat = NGBClassifier(Dist=k_categorical(3), verbose=False)
----> 6 ngb_cat.fit(X_train, y_train)

~₩Anaconda3₩lib₩site-packages₩ngboost₩ngboost.py in fit(self, X, Y, X_val,
Y_val, sample_weight, val_sample_weight, train_loss_monitor, val_loss_moni
tor, early_stopping_rounds)
    244
    245         for itr in range(self.n_estimators):
--> 246             _, col_idx, X_batch, Y_batch, weight_batch, P_batch =
self.sample(
    247                 X, Y, sample_weight, params
    248             )

~₩Anaconda3₩lib₩site-packages₩ngboost₩ngboost.py in sample(self, X, Y, samp
le_weight, params)
    131             idxs,
    132             col_idx,
--> 133             X[idxs, :][:, col_idx],
    134             Y[idxs],
    135             weight_batch,

MemoryError: Unable to allocate 1.54 GiB for an array with shape (130, 1585029) a
nd data type float64
```

NGBoost

2019년 10월 9일 스탠퍼드 대학교에서 발표한
새로운 부스팅 알고리즘이다.

1. 다른 부스팅 알고리즘보다 성능이 괜찮다.
2. 예측의 불확실성까지 측정해준다.
3. 컴퓨팅 시간이 오래 걸린다.

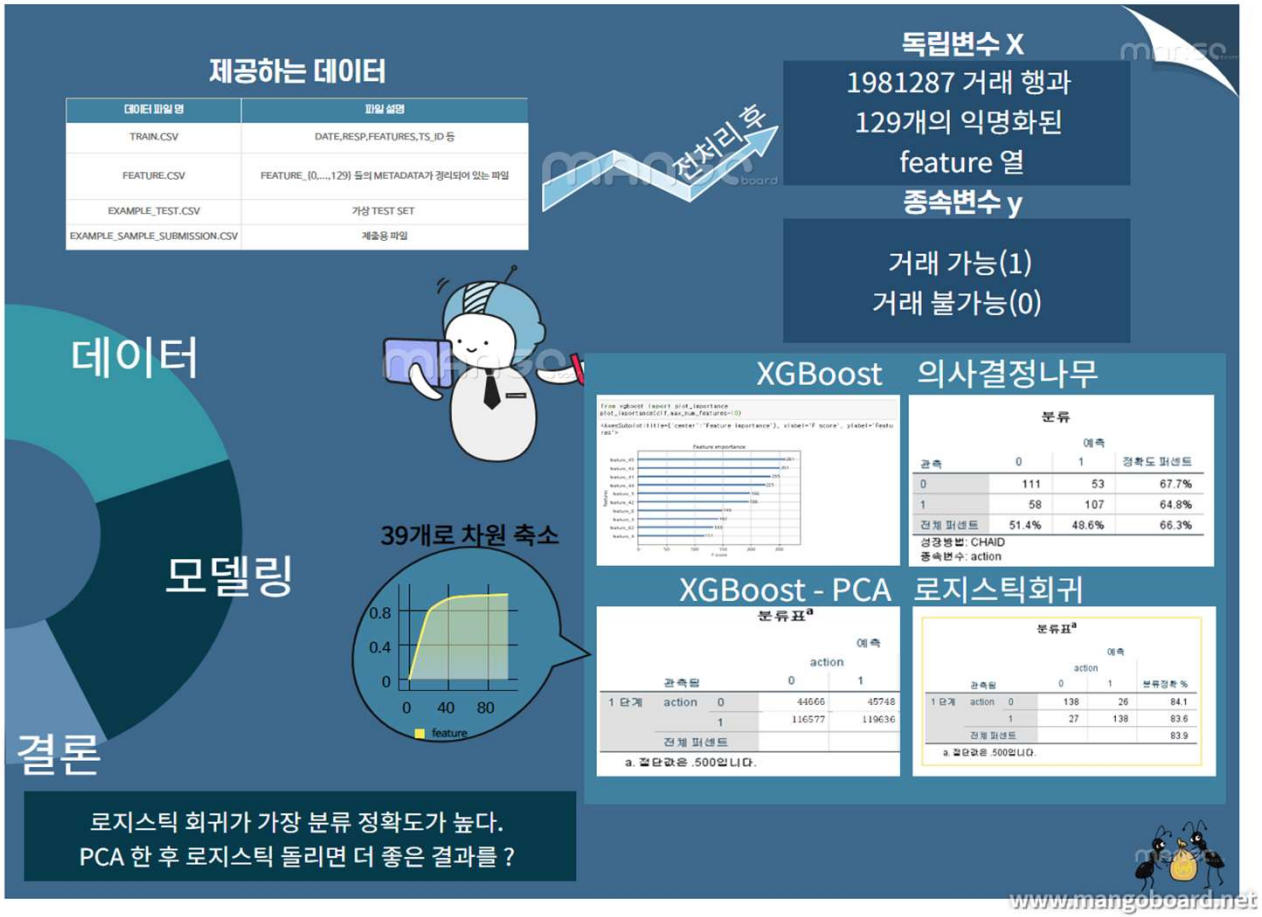
4. 결론

돌렸던 모델 중에서는 로지스틱 회귀가 가장 결과가 좋다.
하지만 설명력이 좋은 모델은 아니다.

PCA를 써서 39개로 변수를 줄여서 하는게 129개를 다 넣은 것 보다 효과가 좋다.

6. 인포그래픽

'쉬운' 디자인 제작 플랫폼, 망고보드



END