AI 경진대회 코딩 테스트 (B형)

제출자: 이성욱(sungwook.Lee@hyundai.com)
날짜: '21.6/28
문제(Type B): 5차원으로 데이터의 차원 축소, 축소를 통해 정리된 인자들의 특성을 살펴라

OVEVIEW

1. Data Load and View
2. Feature Dimenstion Reduce
3. Feature Character

1. Data Load and View

```
from subprocess import check_output
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
from sklearn.preprocessing import StandardScaler

print(check_output(["ls","input"]).decode('utf8'))
```

```
AI경진대회 예선(B형)_data-set.zip
input.csv
output.csv
```

```
input = pd.read_csv('input/input.csv', header=None)
output = pd.read_csv('input/output.csv', header=None)
output = output.rename(columns={0:'Label'})
```

```
input.values
```

```
array([[  0.83035958,  -0.33025241,  -0.23054277, ...,  -1.02979077,
         -4.27514811,  -0.59929727],
       [ -0.04399859,   0.22065793,   1.60051901, ...,  -1.10753423,
        -20.25542908,  -0.56636377],
       [  0.62671752,   2.10042501,  -0.96579802, ...,  -1.03976259,
        -10.22693074,  -1.05338458],
       ...,
       [ -0.71817134,   0.26945901,   0.53723753, ...,   0.44234589,
        -13.406614  ,   0.85427125],
       [ -0.3884856 ,  -0.20375512,   1.40039956, ...,  -1.08230872,
          40.66522873,  -1.58154278],
       [ -0.09540666,   1.47321441,   1.05998807, ...,  -1.11836725,
          27.74371615,  -1.51622948]])
```

```
# Standard Scaling: Data Normalization -> set mean 0 , and std 1.
stscaler= StandardScaler()
stscaler.fit(input)
input_ = stscaler.transform(input)

y = output
data = pd.DataFrame(input_)
data.head()
```

	0	1	2	3	4	5	6	7	8	9	
0	0.818106	-0.325165	-0.246214	-0.012640	-0.736274	1.205771	-1.074941	0.554851	1.166372	0.445609	1.06589
1	-0.051535	0.224741	1.586533	-0.806663	-1.905261	-1.428647	-1.138761	-0.189653	1.009079	1.932479	0.06046

	0	1	2	3	4	5	6	7	8	9	
2	0.615563	2.101084	-0.982146	-0.264738	1.043126	2.171876	-1.185627	0.103988	0.612144	-0.328839	0.84997
3	0.607127	-2.041983	0.181954	-0.300708	-1.254648	-2.385617	-0.814498	-0.312310	1.381110	-0.885105	-0.5788
4	1.416870	0.361684	0.424445	-1.209425	0.349353	-0.397125	-0.460240	0.267333	-1.291405	1.225944	1.43277

```
data.describe()
```

	0	1	2	3	4	5	6	
count	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	1.000000e+04	1.000000
mean	-3.087669e-17	-1.141309e-17	9.348078e-18	-5.060535e-17	3.215206e-17	3.910205e-17	-1.594280e-17	-4.818368
std	1.000050e+00	1.000050e+00	1.000050e+00	1.000050e+00	1.000050e+00	1.000050e+00	1.000050e+00	1.000050
min	-3.688757e+00	-3.859865e+00	-4.714368e+00	-3.885379e+00	-3.629213e+00	-3.770522e+00	-1.870203e+00	-3.621609
25%	-6.701758e-01	-6.670909e-01	-6.835581e-01	-6.800196e-01	-6.682490e-01	-6.670017e-01	-9.264151e-01	-6.859074
50%	2.088883e-03	4.061022e-03	6.491199e-04	3.440946e-03	-3.559668e-03	6.739244e-03	2.121647e-03	-1.018180
75%	6.776271e-01	6.633211e-01	6.685248e-01	6.733688e-01	6.826127e-01	6.669742e-01	6.198403e-01	6.767314
max	3.786736e+00	3.700328e+00	3.819858e+00	3.731669e+00	3.686054e+00	3.725068e+00	3.512268e+00	3.896716

2. Feature Dimenstion Reduce

- using `PCA` : one of represenative method known as Linear Dimension Reduction. PCA(Principal Component Analysis)

```
from sklearn.decomposition import PCA

pca = PCA(n_components=5)
reduced_dim = pca.fit_transform(data)
reduced_dim = pd.DataFrame(reduced_dim, columns=['principal_comp1','principal_comp2','principal_comp3','principal_comp4','principal_c
reduced_dim
```

	principal_comp1	principal_comp2	principal_comp3	principal_comp4	principal_comp5
0	-1.618868	0.604217	0.781736	-0.547131	-0.846699
1	-1.720829	-0.031994	0.435634	-2.228988	0.958939
2	-1.926017	-1.115552	0.746879	2.104426	-1.363396
3	-1.350258	0.556883	-0.693035	-2.326694	1.298485
4	-0.746567	0.096190	-0.198689	1.189678	-1.028490
...	...	...	...	...	...
9995	3.045021	0.325163	-0.428845	1.365473	-1.105546
9996	-0.575423	0.545010	1.186840	0.204053	-0.563458
9997	1.088324	-0.578450	0.264246	-0.760159	-0.259600
9998	-2.304876	-0.420677	-1.557204	1.736730	1.008181
9999	-2.319175	0.200419	0.456647	0.820593	-0.812472

10000 rows × 5 columns

3. Feature Character

1. Test Classifier using RandomForestClassifier

2. Charateristic: explained variance ratio summation is 34.703%, i.e 34.7% information of original data was remained

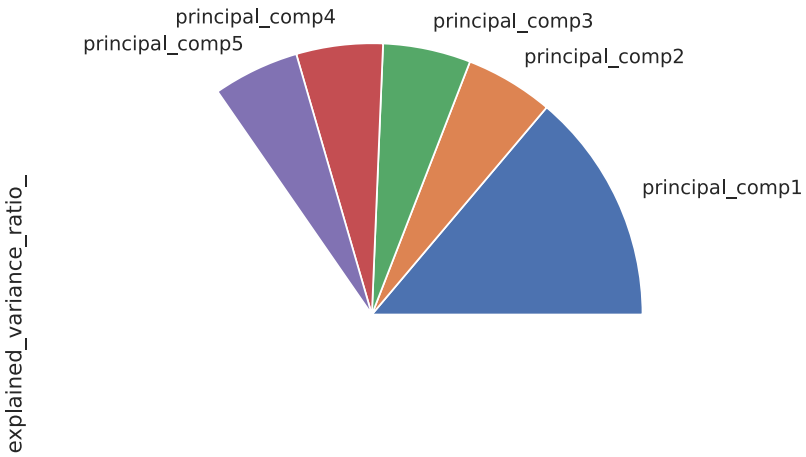
```
from sklearn.ensemble import RandomForestClassifier
# 1) Test Classifier using RandomForestClassifier
clf=RandomForestClassifier(n_estimators=90)
clf.fit(reduced_dim,y)
pred = clf.predict(reduced_dim)

score = clf.score(reduced_dim, y)
print("RandomForestClassifier Score is {} %".format(score*100))
print("Dimension Reduction was executed well!")
```

```
RandomForestClassifier Score is 100.0 %
Dimension Reduction was executed well!
```

```
import numpy as np
# 2) Charateristic
print('Eigen_value :', pca.explained_variance_)
#Explained Variance Ratio는 주성분 벡터가 이루는 축에 투영(projection)한 결과의 분산의 비율을 말하며, 각 eigenvalue의 비율과 같은 의미
print('Explained variance ratio :', pca.explained_variance_ratio_)
print('Information Reduction Percet is {}%'.format(np.sum(pca.explained_variance_ratio_)*100))

df = pd.DataFrame(pca.explained_variance_ratio_, index=['principal_comp1','principal_comp2','principal_comp3','principal_comp4','prin
df.plot.pie(y='explained_variance_ratio_', figsize=(7,7), legend=False)
```

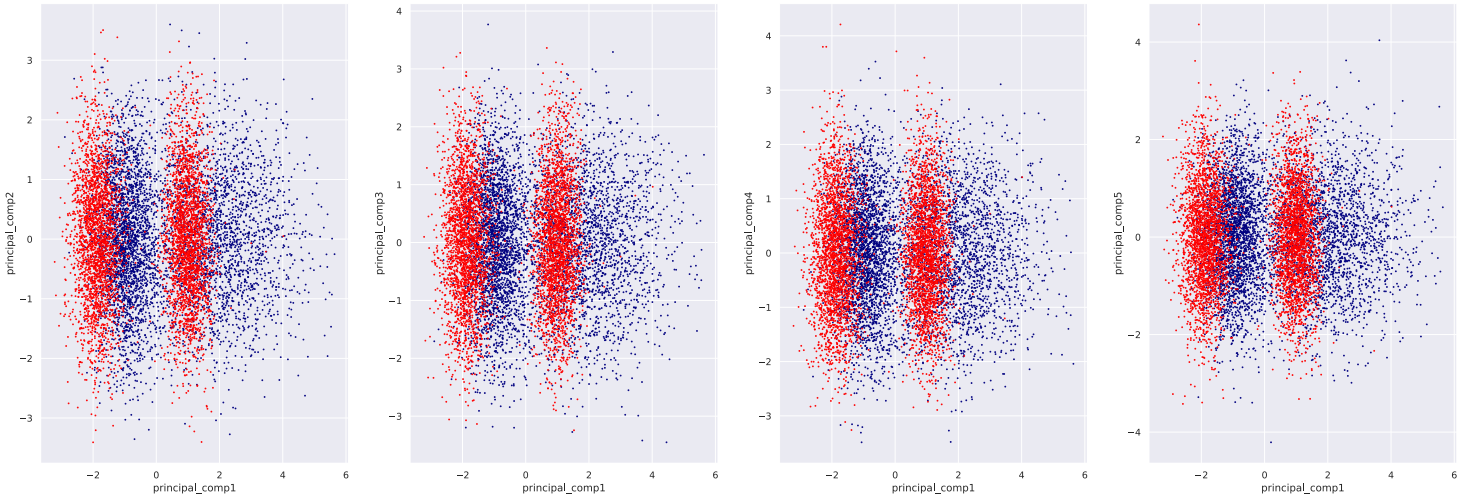
```
Eigen_value : [2.76572688 1.05425708 1.04845199 1.03346204 1.02878758]
Explained variance ratio : [0.13827252 0.05270758 0.05241736 0.05166793 0.05143424]
Information Reduction Percet is 34.64996247376067%
```

```
<AxesSubplot:ylabel='explained_variance_ratio_'>
```



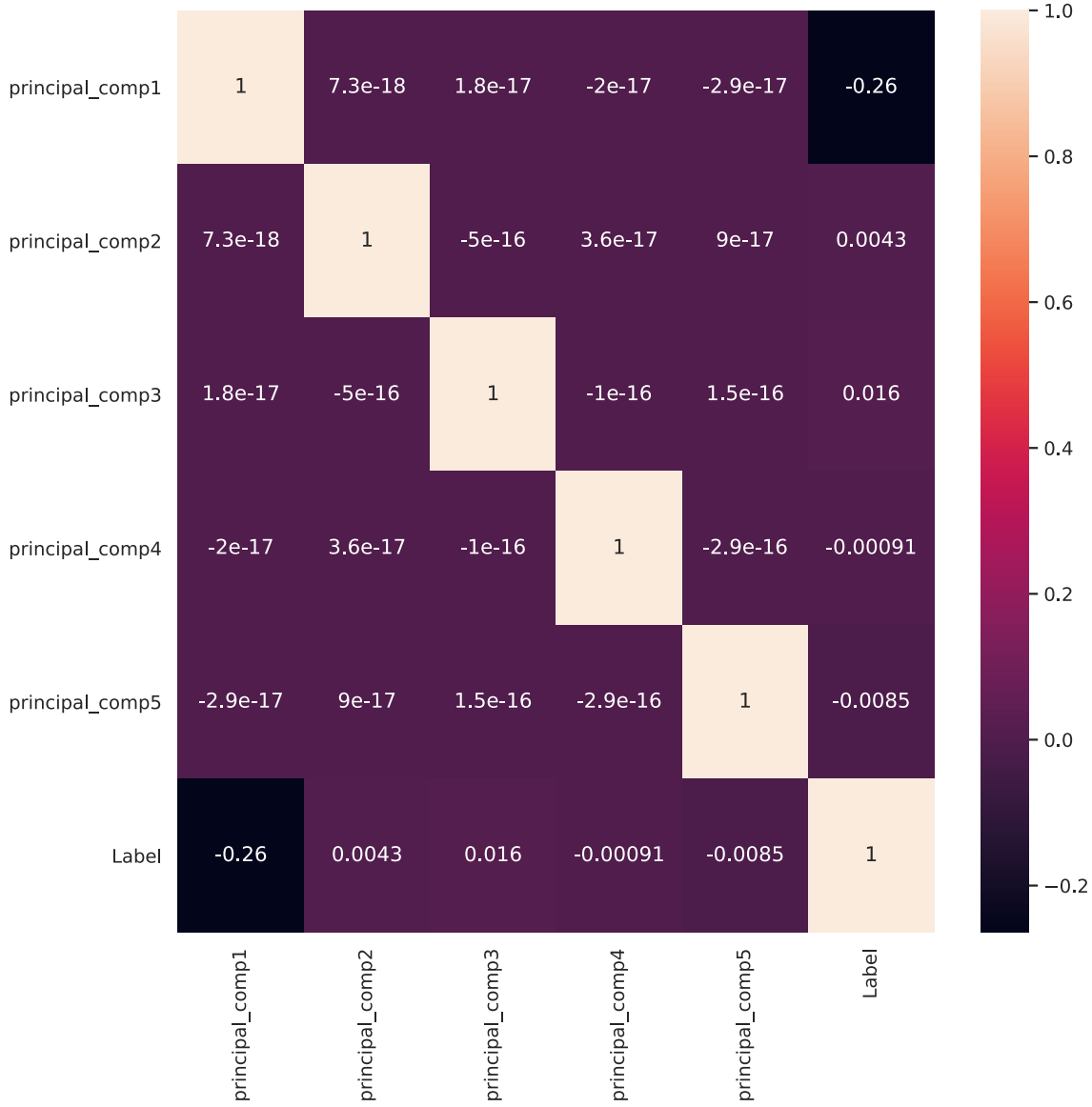
```
plt.figure()
colors=['navy','red']

aug = pd.concat([reduced_dim, y],axis=1)

fig=plt.figure(figsize=(30,10))
ax1=fig.add_subplot(1,4,1)
ax2=fig.add_subplot(1,4,2)
ax3=fig.add_subplot(1,4,3)
ax4=fig.add_subplot(1,4,4)

for color, label in zip(colors, [0, 1]):
    ax1.scatter(x=aug.loc[aug['Label'] == label]['principal_comp1'], y=aug.loc[aug['Label'] == label]['principal_comp2'], c=color,  s
    ax1.set_xlabel('principal_comp1')
    ax1.set_ylabel('principal_comp2')

for color, label in zip(colors, [0, 1]):
    ax2.scatter(x=aug.loc[aug['Label'] == label]['principal_comp1'], y=aug.loc[aug['Label'] == label]['principal_comp3'], c=color,  s
    ax2.set_xlabel('principal_comp1')
    ax2.set_ylabel('principal_comp3')

for color, label in zip(colors, [0, 1]):
    ax3.scatter(x=aug.loc[aug['Label'] == label]['principal_comp1'], y=aug.loc[aug['Label'] == label]['principal_comp4'], c=color,  s
    ax3.set_xlabel('principal_comp1')
    ax3.set_ylabel('principal_comp4')

for color, label in zip(colors, [0, 1]):
    ax4.scatter(x=aug.loc[aug['Label'] == label]['principal_comp1'], y=aug.loc[aug['Label'] == label]['principal_comp5'], c=color,  s
    ax4.set_xlabel('principal_comp1')
    ax4.set_ylabel('principal_comp5')
```

```
<Figure size 432x288 with 0 Axes>
```




```
corrmat = aug.corr()
plt.figure(figsize=(10,10))
sns.heatmap(corrmat, annot=True)
```

```
<AxesSubplot:>
```


principal_comp1
principal_comp2
principal_comp3
principal_comp4
principal_comp5
Label
1
7.3e-18
1.8e-17
-2e-17
-2.9e-17
-0.26
7.3e-18
1
-5e-16
3.6e-17
9e-17
0.0043
1.8e-17
-5e-16
1
-1e-16
1.5e-16
0.016
-2e-17
3.6e-17
-1e-16
1
-2.9e-16
-0.00091
-2.9e-17
9e-17
1.5e-16
-2.9e-16
1
-0.0085
-0.26
0.0043
0.016
-0.00091
-0.0085
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

끝