프로젝트 기반 빅데이터 서비스 솔루션 개발 전문과정

교과목명 : 분석라이브러리 활용

- 평가일 : 22.1.21
- 성명 : 이재우
- 점수 :

Q1. 표준정규분포 기반의 2행 3열 배열을 랜덤하게 생성하여 크기, 자료형, 차원을 출력하세요.

In [1]:

```
import numpy as np
array = np.random.randn(2,3)
display(array)

print(array.shape)
print(array.dtype)
print(array.ndim)
```

```
array([[-0.05505599,  0.00496463, -1.01877049],
       [-0.6323609 ,  0.38322885,  1.82983473]])

(2, 3)
float64
2
```

Q2. arange(), reshape() 이용 1차원 2차원 3차원 배열을 아래와 같이 생성하세요.

[0 1 2 3 4 5 6 7 8 9]

[[0 1 2 3 4]
[5 6 7 8 9]]

[[[0 1 2 3 4]
[5 6 7 8 9]]]

In [2]:

```
import numpy as np
array = np.arange(0,10)

print(array)
print()
array1 = array.reshape(2,-1)
print(array1)
print()
array2 = array.reshape(1,2,-1)
print(array2)
```

```
[0 1 2 3 4 5 6 7 8 9]

[[0 1 2 3 4]
 [5 6 7 8 9]]

[[[0 1 2 3 4]
  [5 6 7 8 9]]]
```

Q3. 1 ~ 100 까지 배열에서 3과 7의 공배수인 것만을 출력하세요.

In [3]:

```
import numpy as np
array = np.arange(1,101)
a3_7 = []
for a in array:
    if a%3 == 0 and a%7==0:
        a3_7.append(a)
print(a3_7)
```

```
[21, 42, 63, 84]
```

Q4. 아래 3차원 배열을 생성하여 출력한 후 1차원으로 변환하여 출력하세요.(reshape() 사용)

```
[[[ 0 1 2 3 4]
[ 5 6 7 8 9]]

[[10 11 12 13 14]
[15 16 17 18 19]]

[[20 21 22 23 24]
[25 26 27 28 29]]]
```

In [175]:

```
import numpy as np

array = np.arange(0,30).reshape(3,2,-1)
print(array)
print()
array2 = array.reshape(1,1,30)
array2 = array2[0][0]
print(array2)
```

```
[[[ 0  1  2  3  4]
  [ 5  6  7  8  9]]

 [[10 11 12 13 14]
  [15 16 17 18 19]]

 [[20 21 22 23 24]
  [25 26 27 28 29]]]

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29]
```

Q5. array2d에서 인덱스를 이용해서 값을 선택하고 리스트로 아래와 같이 출력하세요.

arr2d = np.arange(1,10).reshape(3,3)

[3, 6]

[[1, 2],
[4, 5]]

[[1, 2, 3]
[4, 5, 6]]

In [5]:

```
import numpy as np

arr2d = np.arange(1,10).reshape(3,3)

print(arr2d[:2,2])
print()
print(arr2d[:2,:2])
print()
print(arr2d[:2,:])
```

```
[3 6]

[[1 2]
 [4 5]]

[[1 2 3]
 [4 5 6]]
```

Q6. zeros_like, ones_like, full_like 함수 사용 예를 작성하세요.

In [6]:

```
import numpy as np

array = np.empty((5,5))

array_zeros = np.zeros_like(array)
print(array_zeros)
print()

array_ones = np.ones_like(array)
print(array_ones)
print()

array_full = np.full_like(array,7)
print(array_full)
print()
```

```
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]

[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]

[[7. 7. 7. 7. 7.]
 [7. 7. 7. 7. 7.]
 [7. 7. 7. 7. 7.]
 [7. 7. 7. 7. 7.]
 [7. 7. 7. 7. 7.]]
```

Q7. 10 ~ 20 사이의 정수 난수로 10행 5열 2차원 배열을 생성하고 저장한 후 다시 불러내서 출력하세요.

In [7]:

```
import numpy as np
np.random.seed(0)
array = np.random.randint(10,21,50).reshape(10,5)
print(array)
np.save('array.npy',array)

arrrr = np.load('array.npy')
arrrr
```

```
[[15 10 13 13 17]
 [19 13 15 12 14]
 [17 16 18 18 20]
 [11 16 17 17 18]
 [11 15 19 18 19]
 [14 13 10 13 15]
 [10 12 13 18 11]
 [13 13 13 17 10]
 [11 19 19 10 20]
 [14 17 13 12 17]]
```

Out[7]:

```
array([[15, 10, 13, 13, 17],
       [19, 13, 15, 12, 14],
       [17, 16, 18, 18, 20],
       [11, 16, 17, 17, 18],
       [11, 15, 19, 18, 19],
       [14, 13, 10, 13, 15],
       [10, 12, 13, 18, 11],
       [13, 13, 13, 17, 10],
       [11, 19, 19, 10, 20],
       [14, 17, 13, 12, 17]])
```

Q8. df = sns.load_dataset('titanic')로 불러와서 다음 작업을 수행한 후 출력하세요.

- 전체 칼럼중 'survived'외에 모든 칼럼을 포함한 df_x를 산출한 후 dataset/df_x.pkl로 저장한다.
- df_x.pkl을 데이터프레임 df_x 이름으로 불러온 후 앞 5개 행을 출력한다.

In [21]:

```
import seaborn as sns
import pandas as pd
import pickle
import numpy as np

df = sns.load_dataset('titanic')
df_x = df.drop('survived',axis = 1)
df_x.to_pickle('dataset/df_x.pkl')

df_x = pd.read_pickle('dataset/df_x.pkl')
df_x.head()
```

Out[21]:

	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	em
0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	
2	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	S
3	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

Q9. df = sns.load_dataset('titanic')로 불러와서 deck 열에서 NaN 갯수를 계산하세요.

In [25]:

```
import seaborn as sns
import pandas as pd
import pickle
import numpy as np

df = sns.load_dataset('titanic')

df['deck'].isnull().sum()
```

```
deck
C     59
B     47
D     33
E     32
A     15
F     13
G      4
dtype: int64
```

Out[25]:

```
688
```

Q10. Q9의 df에서 각 칼럼별 null 개수와 df 전체의 null 개수를 구하세요.

In [31]:

```
import seaborn as sns
import pandas as pd
import pickle
import numpy as np

df = sns.load_dataset('titanic')

print(df.isnull().sum())
print()
print(df.isnull().sum().sum())
```

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64

869
```

아래 tdf 데이터프레임에서 Q11 ~ Q12 작업을 수행하세요.

In [32]:

```
import seaborn as sns
df = sns.load_dataset('titanic')
tdf = df[['survived','sex','age','class']]
tdf.head()
```

Out[32]:

	survived	sex	age	class
0	0	male	22.0	Third
1	1	female	38.0	First
2	1	female	26.0	Third
3	1	female	35.0	First
4	0	male	35.0	Third

Q11. age를 7개 카테고리로 구분하는 새로운 칼럼 'cat_age'를 생성하여 출력하세요. 단, 카테고리 구분을 수행하는 사용자 함수를 만들고 그 함수를 age 칼럼에 매핑하여 결과를 tdf1에 저장하고 출력하세요.

[카테고리]
age <= 5: cat = 'Baby'
age <= 12: cat = 'Child'
age <= 18: cat = 'Teenager'
age <= 25: cat = 'Student'
age <= 60: cat = 'Adult'
age > 60 : cat = 'Elderly'

In [34]:

```
def cat_age(age):
    cat = ''
    if age <= 5: cat = 'Baby'
    elif age <= 12: cat = 'Child'
    elif age <= 18: cat = 'Teenager'
    elif age <= 25: cat = 'Student'
    elif age <= 60: cat = 'Adult'
    elif age > 60 : cat = 'Elderly'
    return cat

tdf1 = tdf.copy()

tdf1['cat_age'] = tdf.age.apply(lambda x:cat_age(x))

tdf1.head()
```

Out[34]:

	survived	sex	age	class	cat_age
0	0	male	22.0	Third	Student
1	1	female	38.0	First	Adult
2	1	female	26.0	Third	Adult
3	1	female	35.0	First	Adult
4	0	male	35.0	Third	Adult

In [64]:

```
tdf1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   survived  891 non-null    int64
 1   sex       891 non-null    object
 2   age       714 non-null    float64
 3   class     891 non-null    object
 4   cat_age   891 non-null    object
 5   sc        891 non-null    object
dtypes: float64(1), int64(1), object(4)
memory usage: 41.9+ KB
```

Q12. tdf1의 sex, class 칼럼을 '_'으로 연결한 'sc'칼럼을 추가한 후 아래와 같이 출력하세요.

In [67]:

```
tdf1['class']=tdf1['class'].astype(str)

s_list = list(tdf1['sex'])
c_list = list(tdf1['class'])
sc_list = []
for i in range(len(s_list)):
    sc_list.append(s_list[i]+'_'+c_list[i])

tdf1['sc'] = sc_list
tdf1['sc']
```

Out[67]:

```
0        male_Third
1      female_First
2      female_Third
3      female_First
4        male_Third
           ...
886     male_Second
887    female_First
888    female_Third
889      male_First
890      male_Third
Name: sc, Length: 891, dtype: object
```

Q13. join() 메소드는 두 데이터프레임의 행 인덱스를 기준으로 결합한다. 2개의 주식데이터를 가져와서 join() 메소드로 아래와 같이 결합한 후 다음 사항을 수행하세요.

- df1과 df2의 교집합만 출력되도록 결합하여 df3에 저장하고 출력
- df3에서 중복된 칼럼을 삭제한 후 블린 인덱싱을 이용하여 eps가 3000 보다 적거나 stock_name이 이마트인 데이터를 선택하여 데이터프레임을 생성하고 df4 이름으로 저장 및 출력하세요.(단, '<' 와 '==' 를 반드시 사용해야 함)

In [93]:

```
import numpy as np
import pandas as pd


df1 = pd.read_excel('./dataset/stock price.xlsx', index_col='id')
df2 = pd.read_excel('./dataset/stock valuation.xlsx', index_col='id')

df3 = df1.join(df2)
df3.dropna(inplace = True)
display(df3)
df4 = df3.copy()
df4.drop('name',axis = 1,inplace = True)
df4 = df4.loc[(df4['stock_name']=='이마트')|(df4.eps<3000)]
display(df4)
```

id	stock_name	value	price	name	eps	bps	per	pb
130960	CJ E&M	58540.666667	98900	CJ E&M	6301.333333	54068.0	15.695091	1.82917
139480	이마트	239230.833333	254500	이마트	18268.166667	295780.0	13.931338	0.86043
145990	삼양사	82750.000000	82000	삼양사	5741.000000	108090.0	14.283226	0.75862
185750	종근당	40293.666667	100500	종근당	3990.333333	40684.0	25.185866	2.47025
204210	모두투어리츠	3093.333333	3475	모두투어리츠	85.166667	5335.0	40.802348	0.65135

id	stock_name	value	price	eps	bps	per	pbr
139480	이마트	239230.833333	254500	18268.166667	295780.0	13.931338	0.860437
204210	모두투어리츠	3093.333333	3475	85.166667	5335.0	40.802348	0.651359

Q14. 배열 a에 대하여 3차원 자리에 2차원을 2차원 자리에 1차원을 1차원 자리에 3차원을 넣어서 변환하여 출력하세요

In [96]:

```
a = np.arange(6).reshape(1,2,3)
print(a,a.shape,'₩n')
a= a.transpose(1,2,0)
print(a,a.shape)
```

```
[[[0 1 2]
  [3 4 5]]] (1, 2, 3)

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

 [[3]
  [4]
  [5]]] (2, 3, 1)
```

Q15. 'mpg'를 'kpl' 로 환산하여 새로운 열을 생성하고 반올림하여 소수점 아래 둘째 자리까지 처음 5개행을 출력하세요.

In [206]:

```
# read_csv() 함수로 df 생성
import pandas as pd
auto_df = pd.read_csv('./dataset/auto-mpg.csv')
# 열 이름을 지정
auto_df.columns = ['mpg','cylinders','displacement','horsepower','weight',
                   'acceleration','model year','origin','name']
auto_df['kpl'] = auto_df.mpg*(1.60934/3.78541)
auto_df['kpl']=auto_df['kpl'].round(2)
auto_df.head()
```

Out[206]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	name	k
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu	7.(
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320	6.:
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite	7.(
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst	6.:
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino	7.:

Q16. './dataset/stock-data.csv'를 데이터프레임으로 불러와서 datetime64 자료형으로 변환한 후에 년, 월, 일로 분리하고 year를 인덱스로 셋팅하여 출력하세요.

In [205]:

```
from datetime import datetime



df = pd.read_csv('./dataset/stock-data.csv')
df['Date'] = pd.to_datetime(df.Date)
df['year'] = df.Date
df['month'] = df.Date
df['day'] = df.Date
for i in range(len(df)):
    df['year'][i]=(df.Date[i].year)
    df['month'][i]=(df.Date[i].month)
    df['day'][i]=(df.Date[i].day)

df = df.set_index('year')
df.head()
```

```
C:₩Users₩admin₩AppData₩Local₩Temp/ipykernel_6176/2650046712.py:11: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['year'][i]=(df.Date[i].year)
C:₩Users₩admin₩AppData₩Local₩Temp/ipykernel_6176/2650046712.py:12: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['month'][i]=(df.Date[i].month)
C:₩Users₩admin₩AppData₩Local₩Temp/ipykernel_6176/2650046712.py:13: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['day'][i]=(df.Date[i].day)
```

Out[205]:

year	Date	Close	Start	High	Low	Volume	month	day
2018	2018-07-02	10100	10850	10900	10000	137977	7	2
2018	2018-06-29	10700	10550	10900	9990	170253	6	29
2018	2018-06-28	10400	10900	10950	10150	155769	6	28
2018	2018-06-27	10900	10800	11050	10500	133548	6	27
2018	2018-06-26	10800	10900	11000	10700	63039	6	26

Q17. titanic 데이터셋에서 5개 열을 선택해서 class열을 기준으로 그룹화를 수행한 후

아래와 같이 출력하였다. 다음 사항을 출력하세요.

5개 열 : ['age','sex', 'class', 'fare', 'survived']

- 그룹별 평균 출력
- 그룹별 최대값 출력

In [118]:

```
import seaborn as sns
df = sns.load_dataset('titanic')
df = df[['age','sex', 'class', 'fare', 'survived']]
groups1 = df.groupby('class').mean()
display(groups1)
groups2 = df.groupby('class').max()
display(groups2)
```

class	age	fare	survived
First	38.233441	84.154687	0.629630
Second	29.877630	20.662183	0.472826
Third	25.140620	13.675550	0.242363

class	age	sex	fare	survived
First	80.0	male	512.3292	1
Second	70.0	male	73.5000	1
Third	74.0	male	69.5500	1

Q18. titanic 데이터셋에서 'Third'그룹만을 선택해서 group3 이름으로 저장하고 통계 요약표를 출력하세요.

In [119]:

```
import seaborn as sns
df = sns.load_dataset('titanic')
group = df.groupby('class')
group3 = group.get_group('Third')
group3.describe()
```

Out[119]:

	survived	pclass	age	sibsp	parch	fare
count	491.000000	491.0	355.000000	491.000000	491.000000	491.000000
mean	0.242363	3.0	25.140620	0.615071	0.393075	13.675550
std	0.428949	0.0	12.495398	1.374883	0.888861	11.778142
min	0.000000	3.0	0.420000	0.000000	0.000000	0.000000
25%	0.000000	3.0	18.000000	0.000000	0.000000	7.750000
50%	0.000000	3.0	24.000000	0.000000	0.000000	8.050000
75%	0.000000	3.0	32.000000	1.000000	0.000000	15.500000
max	1.000000	3.0	74.000000	8.000000	6.000000	69.550000

Q19. titanic 데이터셋에서 다음 전처리를 수행하세요.

1. df에서 중복 칼럼으로 고려할 수 있는 컬럼들(6개 내외)을 삭제한 후 나머지 컬럼들로 구성되는 데이터프레임을 df1 이름으로 저장 후 출력하세요.
2. df1에서 null값이 50% 이상인 칼럼을 삭제 후 df2 이름으로 저장하고 출력하세요.
3. df2에서 결측값이 있는 age 칼럼에 대해서 평균값으로 대체 처리를 수행하세요.
4. df2에서 결측값이 있는 embarked 칼럼에 대해서 앞행의 값으로 대체 처리를 수행하세요.
5. df2 문자로 되어있는 칼럼들을 레이블 인코딩 수행하여 숫자로 변환 후 df2.info()를 출력하세요

In [156]:

```
import seaborn as sns
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df = sns.load_dataset('titanic')
df1 = df.copy()
df1.drop('pclass',axis = 1,inplace=True)
df1.drop('embark_town',axis = 1,inplace=True)
df1.drop('alive',axis = 1,inplace=True)
df1.drop('adult_male',axis = 1,inplace=True)
df1.drop('who',axis = 1,inplace=True)
df1.drop('alone',axis = 1,inplace=True)
df2 = df1.copy()
df2.drop('deck',axis = 1,inplace = True)
df2.loc[df2['age'].isnull(),['age']] = df2.age.mean()
df2.loc[df2['embarked'].isnull(),['embarked']]
df2.iloc[61,6] = df2.iloc[60,6]
df2.iloc[829,6] = df2.iloc[828,6]

features = ['sex','embarked','class']
for feature in features:
    df2[feature] = le.fit_transform(df2[feature])

df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   survived  891 non-null    int64
 1   sex       891 non-null    int32
 2   age       891 non-null    float64
 3   sibsp     891 non-null    int64
 4   parch     891 non-null    int64
 5   fare      891 non-null    float64
 6   embarked  891 non-null    int32
 7   class     891 non-null    int32
dtypes: float64(2), int32(3), int64(3)
memory usage: 45.4 KB
```

Q20. 보스톤 주택가격 데이터를 탐색한 후 가장 중요한 독립변수 2개를 선정하고 그 이유를 시각화하여 설명하세요.

In [168]:

```
from sklearn.datasets import load_boston
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
# boston 데이타셋 로드
boston = load_boston()


# boston 데이타셋 DataFrame 변환
bostonDF = pd.DataFrame(boston.data , columns = boston.feature_names)

# boston dataset의 target array는 주택 가격임. 이를 PRICE 컬럼으로 DataFrame에 추가함.
bostonDF['PRICE'] = boston.target
print('Boston 데이타셋 크기 :',bostonDF.shape)

for feature in bostonDF.columns:
    tmparray = np.array(bostonDF[feature])
    tmparray = tmparray.reshape(-1,1)
    bostonDF[feature] = scaler.fit_transform(tmparray)

bostonDF.corr()
```

```
Boston 데이타셋 크기 : (506, 14)

C:₩Users₩admin₩anaconda3₩envs₩cakd5₩lib₩site-packages₩sklearn₩utils₩deprecation.p
y:87: FutureWarning: Function load_boston is deprecated; `load_boston` is deprecat
ed in 1.0 and will be removed in 1.2.

    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np


        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="₩s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)
```

```
    for the Ames housing dataset.

  warnings.warn(msg, category=FutureWarning)
```

Out[168]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	(
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-(
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	(
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-(
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	(
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-(
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	(
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-(
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	(
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	(
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-(
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	(
PRICE	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-(