3/19(목)

2020년 3월 19일 목요일 오전 9:15

미분 기초

-운전 : 속도 변화

수학적 표현 ? 사물 변화가 다른 사물에 어떤변화를 주는가

Ex) 시간 변화 -> 속도 변화

강우량 변화 -> 식물의 키 변화

힘 변화 -> 용수철 길이 변화

변화율 = 미분 계수

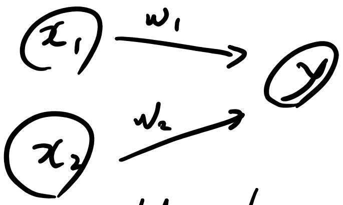
편미분 : 내가 미분하고자 하는 관심있는 값을 제외 하고는 상수로 취급하는 미분

$f(x) = 2x \qquad \frac{df}{dx} = 2$

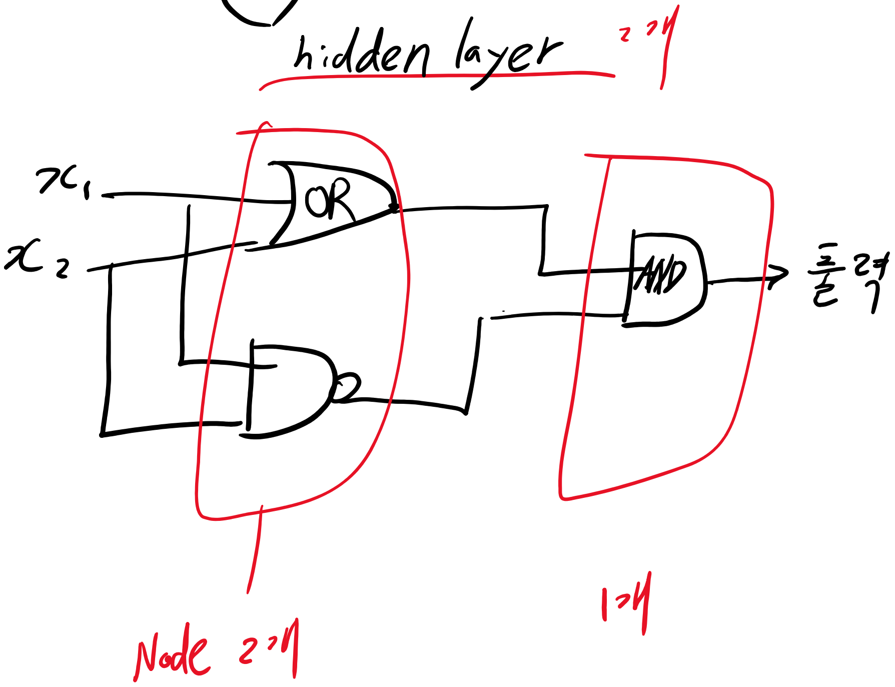
$f(x, y) = xy, \qquad \frac{\partial f}{\partial x} = y$를 상수로 취급

$\frac{\partial f}{\partial x} \Rightarrow y$가 됨

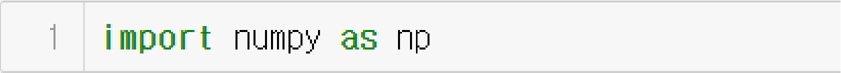
퍼셉트론



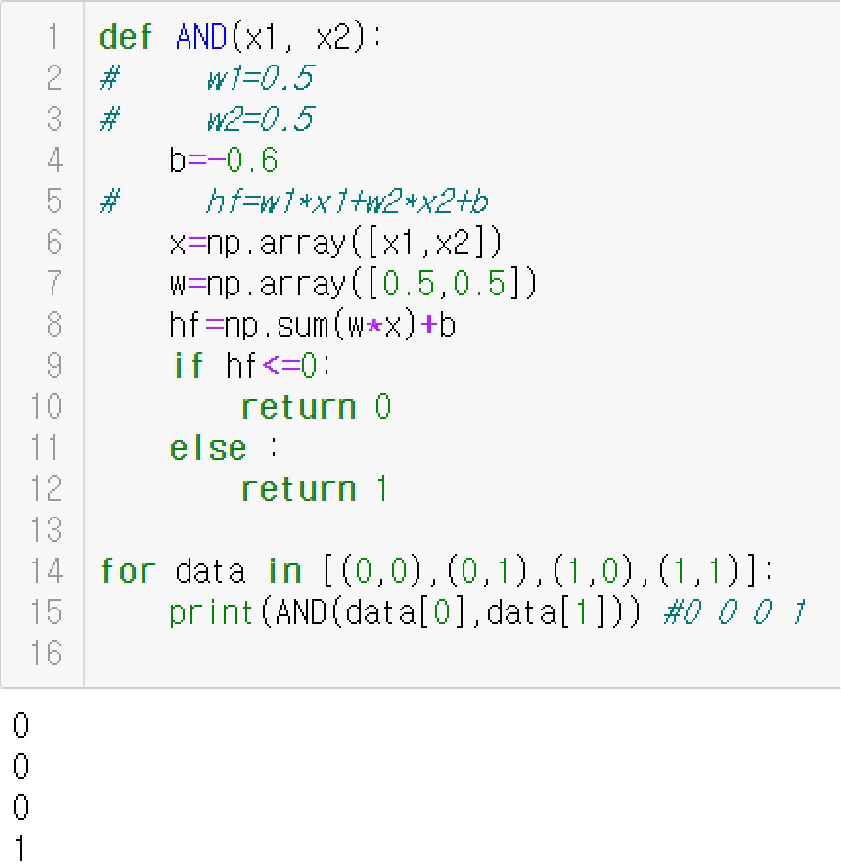
$$y = \begin{cases} 0 : w_1x_1 + w_2x_2 \le \theta \\ 1 : w_1x_1 + w_2x_2 > \theta \end{cases}$$



Node 2기 1기

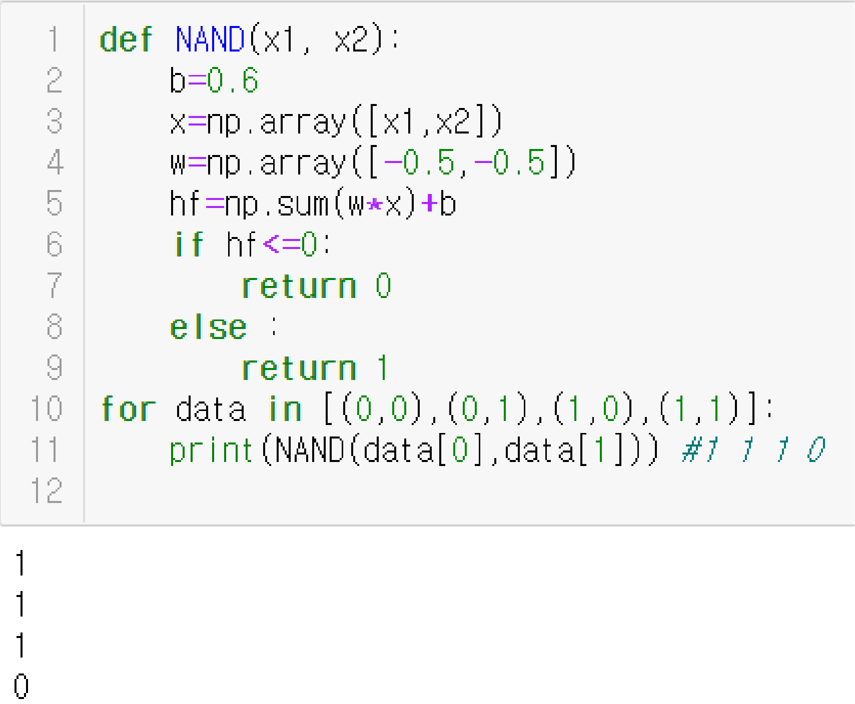


```
import numpy as np
```



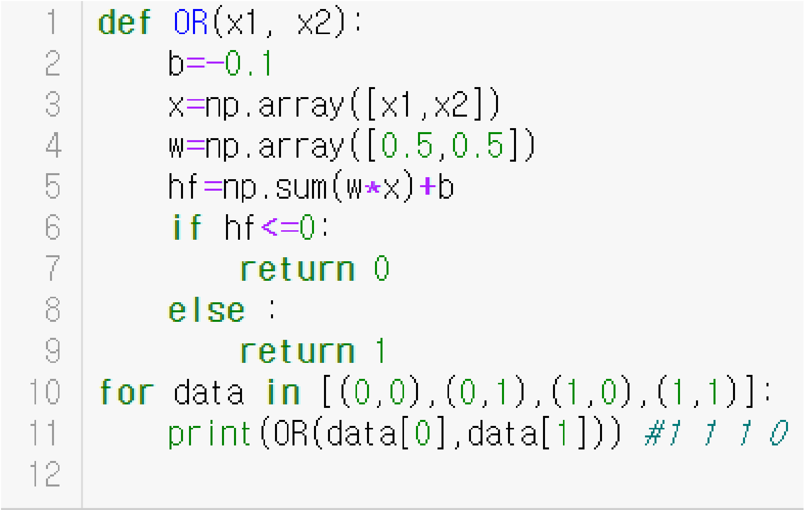
```
def AND(x1, x2):
#     w1=0.5
#     w2=0.5
    b=-0.6
#     hf=w1*x1+w2*x2+b
    x=np.array([x1,x2])
    w=np.array([0.5,0.5])
    hf=np.sum(w*x)+b
    if hf<=0:
        return 0
    else :
        return 1

for data in [(0,0),(0,1),(1,0),(1,1)]:
    print(AND(data[0],data[1])) #0 0 0 1

```

```
0
0
0
1
```



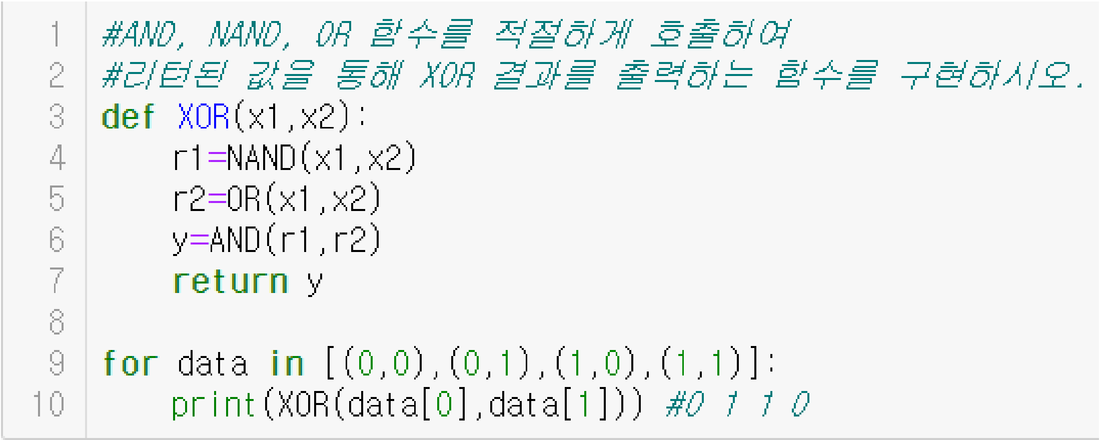
```
def NAND(x1, x2):
    b=0.6
    x=np.array([x1,x2])
    w=np.array([-0.5,-0.5])
    hf=np.sum(w*x)+b
    if hf<=0:
        return 0
    else :
        return 1
for data in [(0,0),(0,1),(1,0),(1,1)]:
    print(NAND(data[0],data[1])) #1 1 1 0

```

```
1
1
1
0
```



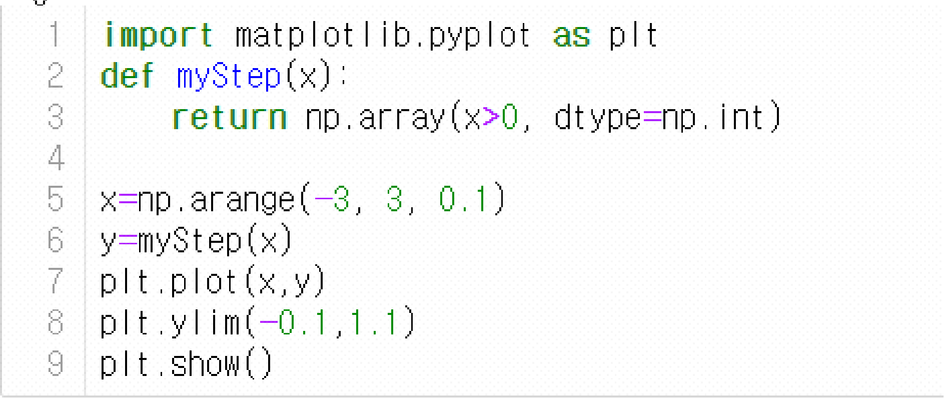
```python
def OR(x1, x2):
    b=-0.1
    x=np.array([x1,x2])
    w=np.array([0.5,0.5])
    hf=np.sum(w*x)+b
    if hf<=0:
        return 0
    else :
        return 1
for data in [(0,0),(0,1),(1,0),(1,1)]:
    print(OR(data[0],data[1])) #1 1 1 0

```

```
0
1
1
1
```

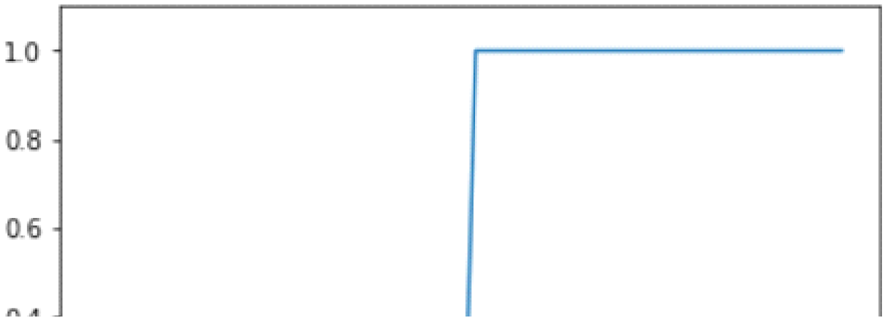


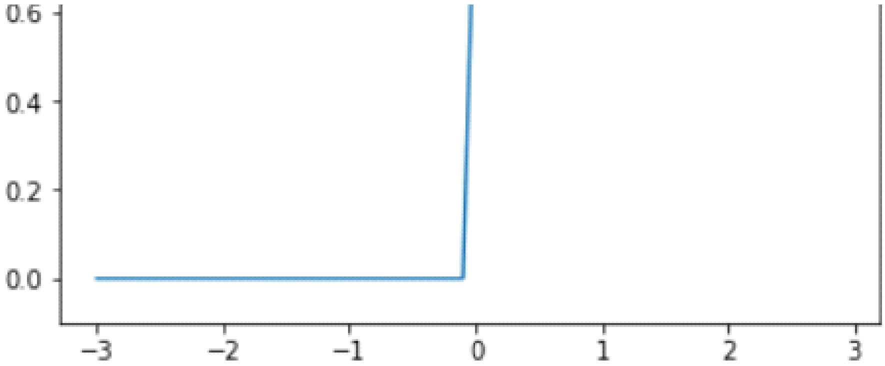
```python
#AND, NAND, OR 함수를 적절하게 호출하여
#리턴된 값을 통해 XOR 결과를 출력하는 함수를 구현하시오.
def XOR(x1,x2):
    r1=NAND(x1,x2)
    r2=OR(x1,x2)
    y=AND(r1,r2)
    return y

for data in [(0,0),(0,1),(1,0),(1,1)]:
    print(XOR(data[0],data[1])) #0 1 1 0
```

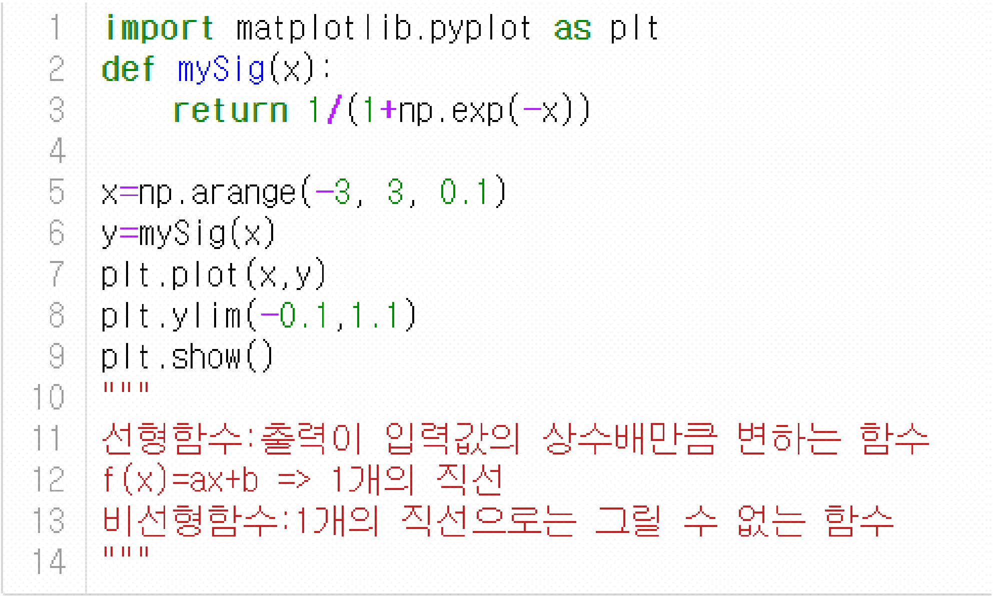
```
0
1
1
0
```



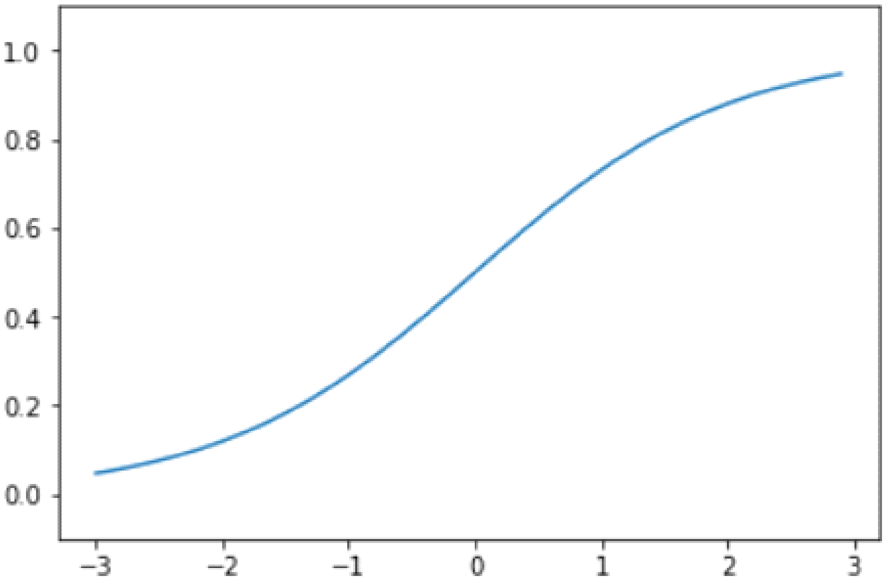
```python
import matplotlib.pyplot as plt
def myStep(x):
    return np.array(x>0, dtype=np.int)

x=np.arange(-3, 3, 0.1)
y=myStep(x)
plt.plot(x,y)
plt.ylim(-0.1,1.1)
plt.show()
```

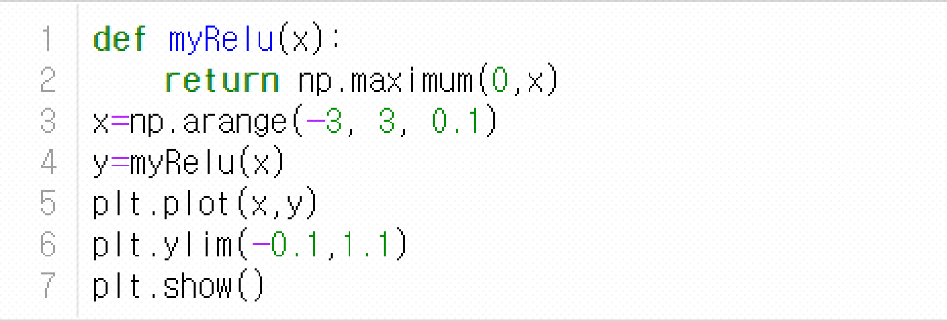




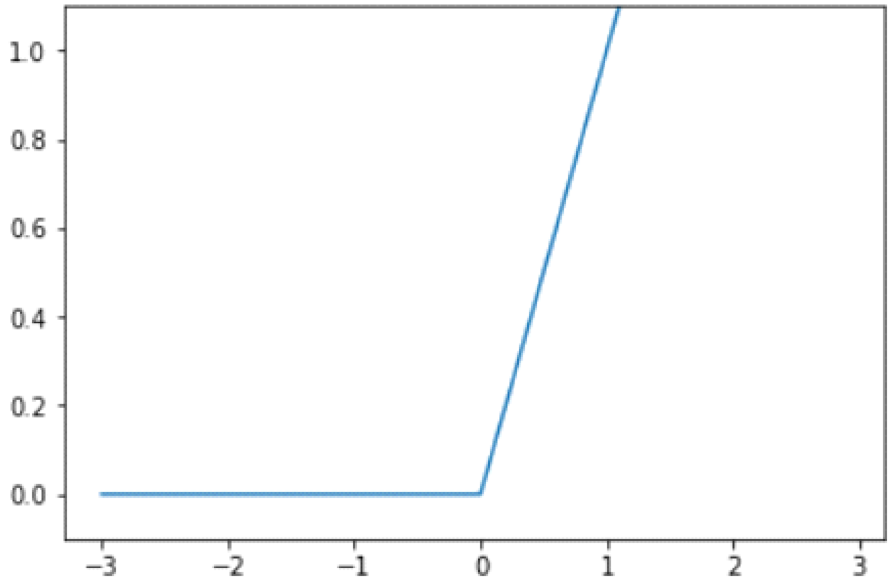


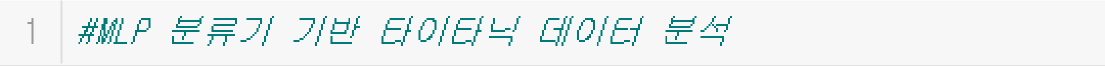
```
import matplotlib.pyplot as plt
def mySig(x):
    return 1/(1+np.exp(-x))

x=np.arange(-3, 3, 0.1)
y=mySig(x)
plt.plot(x,y)
plt.ylim(-0.1,1.1)
plt.show()
"""
선형함수:출력이 입력값의 상수배만큼 변하는 함수
f(x)=ax+b => 1개의 직선
비선형함수:1개의 직선으로는 그릴 수 없는 함수
"""
```



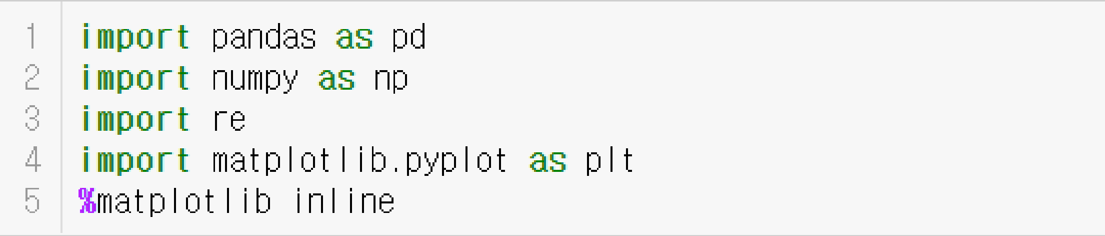


```python
def myRelu(x):
    return np.maximum(0,x)
x=np.arange(-3, 3, 0.1)
y=myRelu(x)
plt.plot(x,y)
plt.ylim(-0.1,1.1)
plt.show()
```

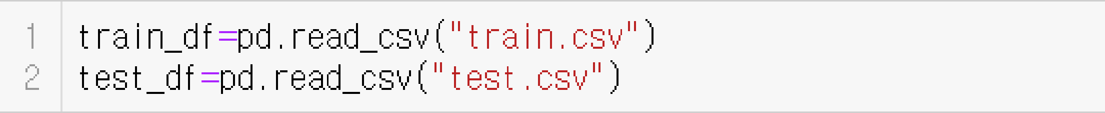




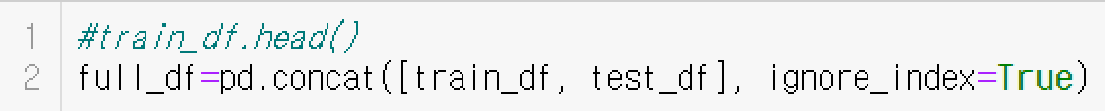
```python
#MLP 분류기 기반 타이타닉 데이터 분석
```



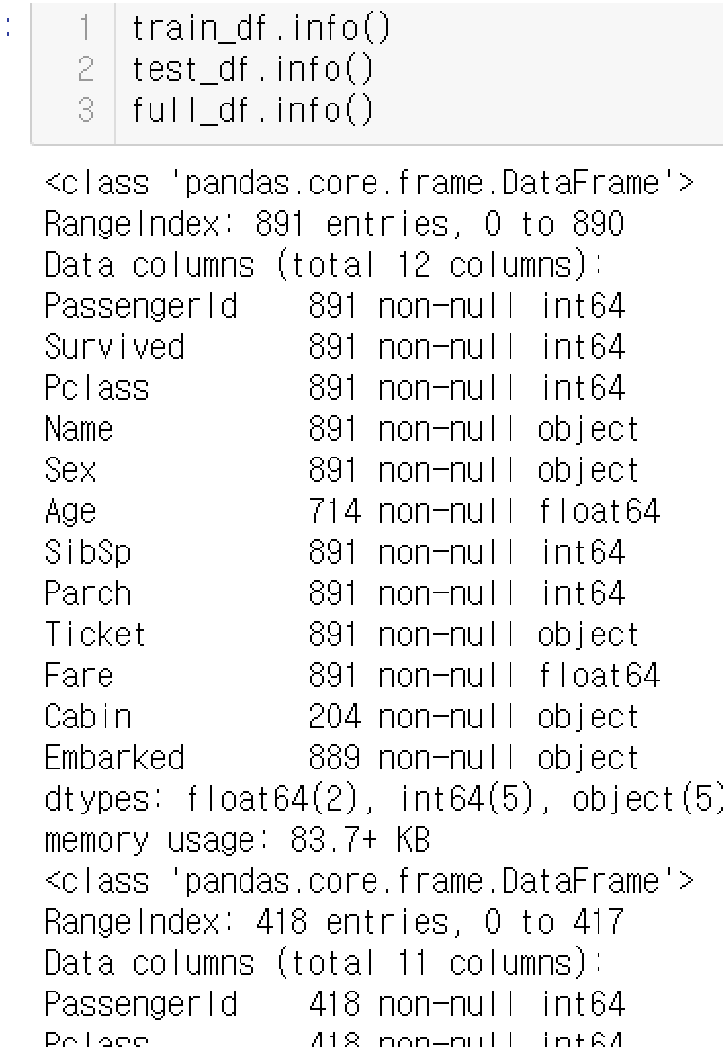
```python
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
%matplotlib inline
```



```python
train_df=pd.read_csv("train.csv")
test_df=pd.read_csv("test.csv")
```

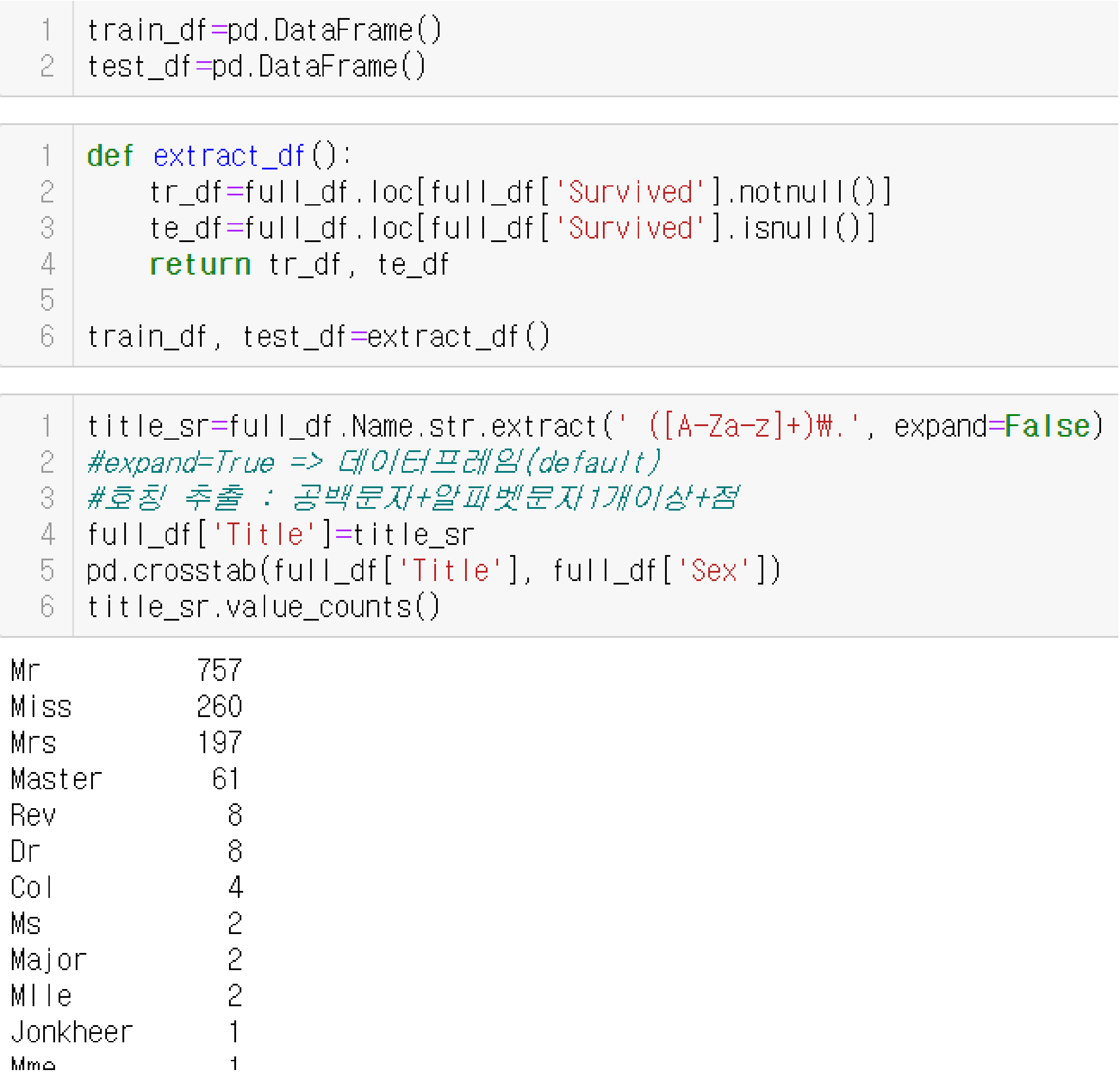


```python
#train_df.head()
full_df=pd.concat([train_df, test_df], ignore_index=True)
```



```
train_df.info()
test_df.info()
full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
```

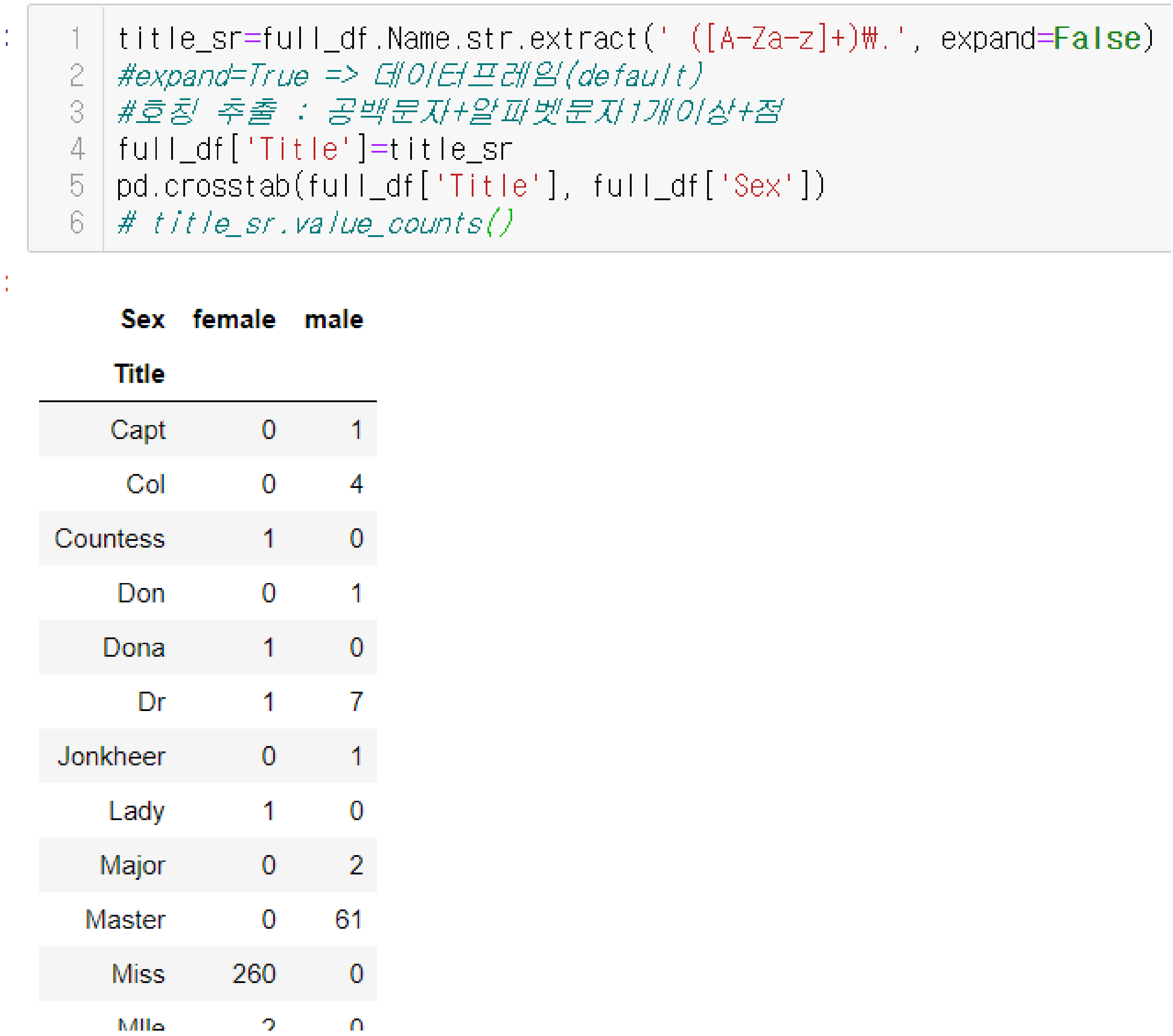


```
train_df=pd.DataFrame()
test_df=pd.DataFrame()
```

```
def extract_df():
    tr_df=full_df.loc[full_df['Survived'].notnull()]
    te_df=full_df.loc[full_df['Survived'].isnull()]
    return tr_df, te_df

train_df, test_df=extract_df()
```

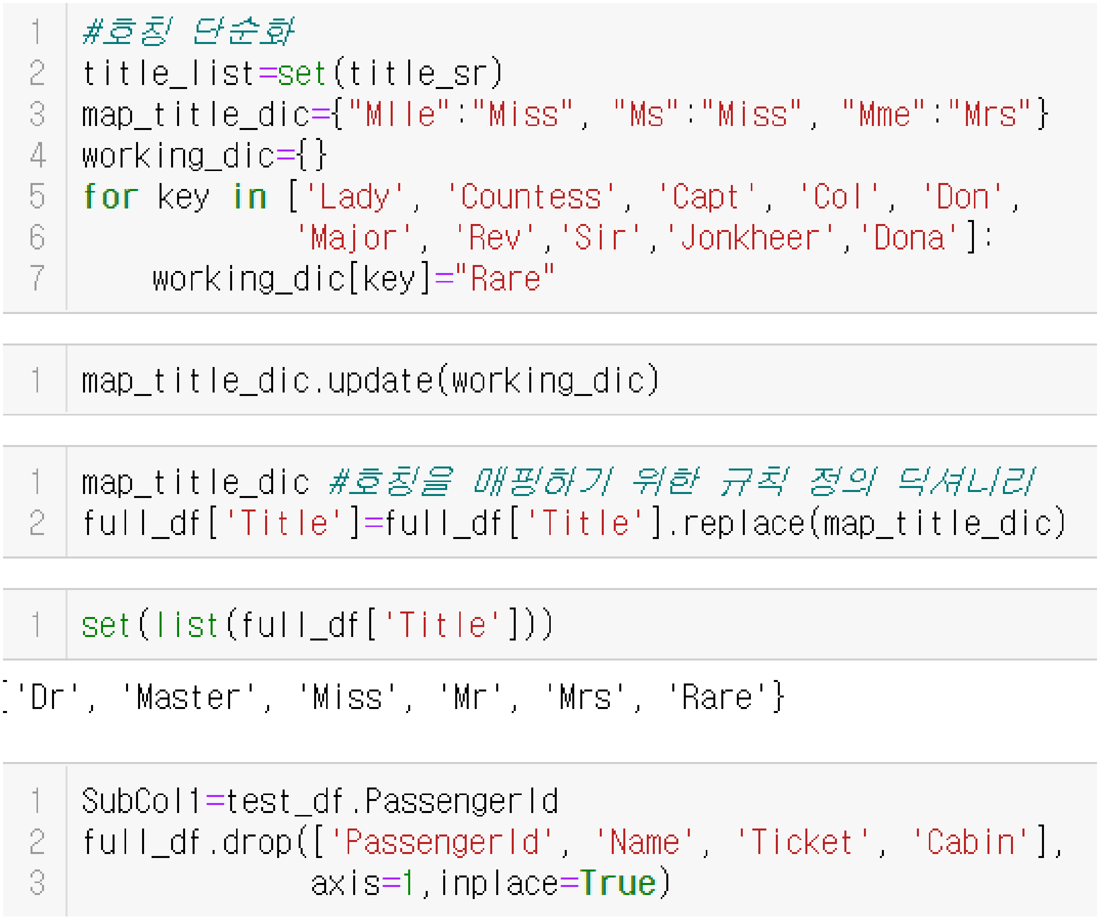
```
title_sr=full_df.Name.str.extract(' ([A-Za-z]+)₩.', expand=False)
#expand=True => 데이터프레임(default)
#호칭 추출 : 공백문자+알파벳문자1개이상+점
full_df['Title']=title_sr
pd.crosstab(full_df['Title'], full_df['Sex'])
title_sr.value_counts()
```

```
Mr          757
Miss        260
Mrs         197
Master       61
Rev           8
Dr            8
Col           4
Ms            2
Major         2
Mlle          2
Jonkheer      1
Mme           1
```



```python
title_sr=full_df.Name.str.extract(' ([A-Za-z]+)₩.', expand=False)
#expand=True => 데이터프레임(default)
#호칭 추출 : 공백문자+알파벳문자1개이상+점
full_df['Title']=title_sr
pd.crosstab(full_df['Title'], full_df['Sex'])
# title_sr.value_counts()
```

Sex	female	male
Title		
Capt	0	1
Col	0	4
Countess	1	0
Don	0	1
Dona	1	0
Dr	1	7
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	61
Miss	260	0
Mlle	2	0



```python
#호칭 단순화
title_list=set(title_sr)
map_title_dic={"Mlle":"Miss", "Ms":"Miss", "Mme":"Mrs"}
working_dic={}
for key in ['Lady', 'Countess', 'Capt', 'Col', 'Don',
            'Major', 'Rev','Sir','Jonkheer','Dona']:
    working_dic[key]="Rare"
```

```python
map_title_dic.update(working_dic)
```

```python
map_title_dic #호칭을 매핑하기 위한 규칙 정의 딕셔너리
full_df['Title']=full_df['Title'].replace(map_title_dic)
```

```python
set(list(full_df['Title']))
```

```
{'Dr', 'Master', 'Miss', 'Mr', 'Mrs', 'Rare'}
```

```python
SubCol1=test_df.PassengerId
full_df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'],
             axis=1,inplace=True)
```

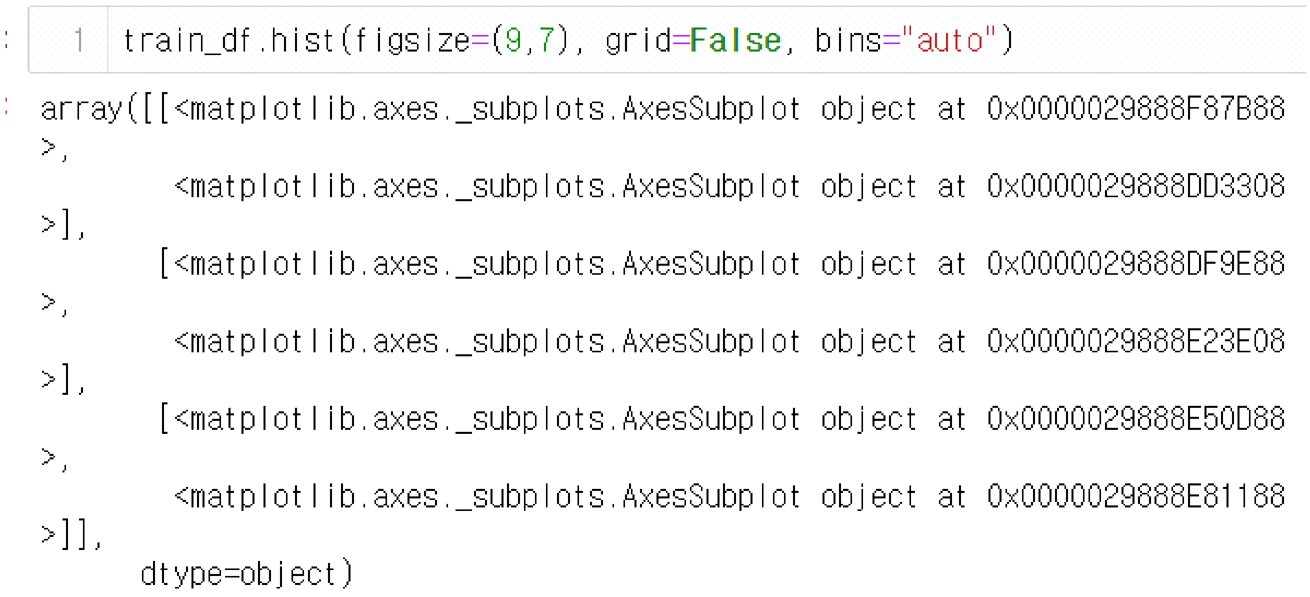
```
train_df, test_df=extract_df()
```

```
train_df
#Pclass별(1,2,3) 생존자(Survived) 평균
train_df[['Pclass','Survived']].groupby(['Pclass'],as_index=False).mean(
```

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

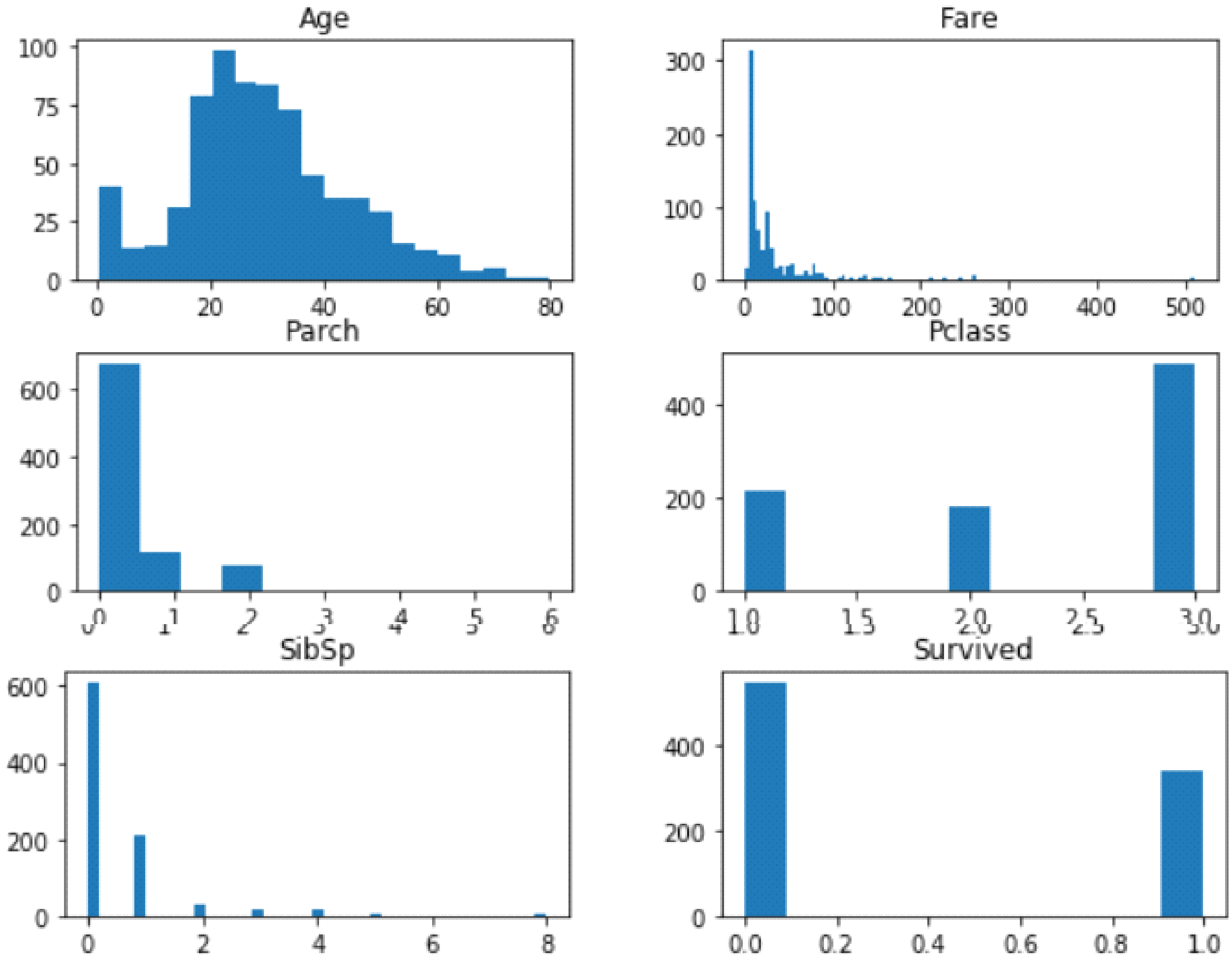
```
feature_list=list(full_df)
for f in feature_list:
    print(f+" "+ str(len(full_df[f].value_counts())))
```

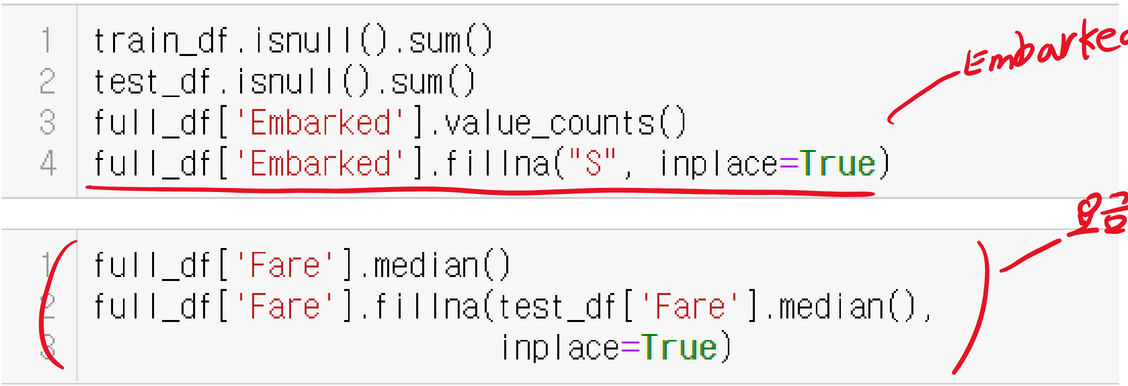
```
Age 98
Embarked 3
Fare 281
Parch 8
Pclass 3
Sex 2
SibSp 7
Survived 2
Title 6
```



```
train_df.hist(figsize=(9,7), grid=False, bins="auto")
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000029888F87B88
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029888DD3308
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029888DF9E88
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029888E23E08
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000029888E50D88
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000029888E81188
>]],
      dtype=object)
```



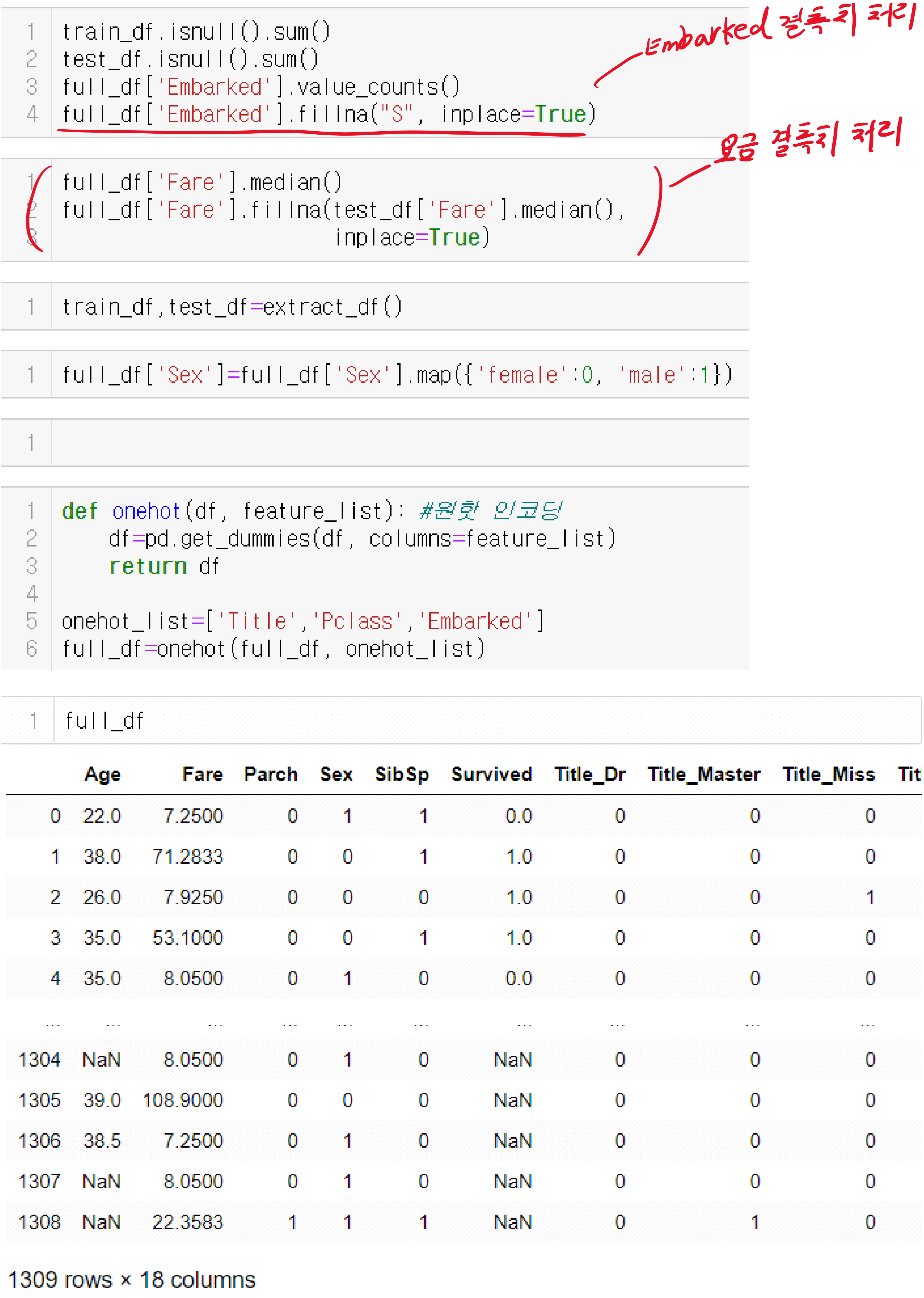


```
train_df.isnull().sum()
test_df.isnull().sum()
full_df['Embarked'].value_counts()
full_df['Embarked'].fillna("S", inplace=True)
```

Embarked 결측치 처리

```
full_df['Fare'].median()
full_df['Fare'].fillna(test_df['Fare'].median(),
                       inplace=True)
```

요금 결측치 처리



```
train_df.isnull().sum()
test_df.isnull().sum()
full_df['Embarked'].value_counts()
full_df['Embarked'].fillna("S", inplace=True)
```

Embarked 결측치 처리

```
full_df['Fare'].median()
full_df['Fare'].fillna(test_df['Fare'].median(),
                                  inplace=True)
```

요금 결측치 처리

```
train_df,test_df=extract_df()
```

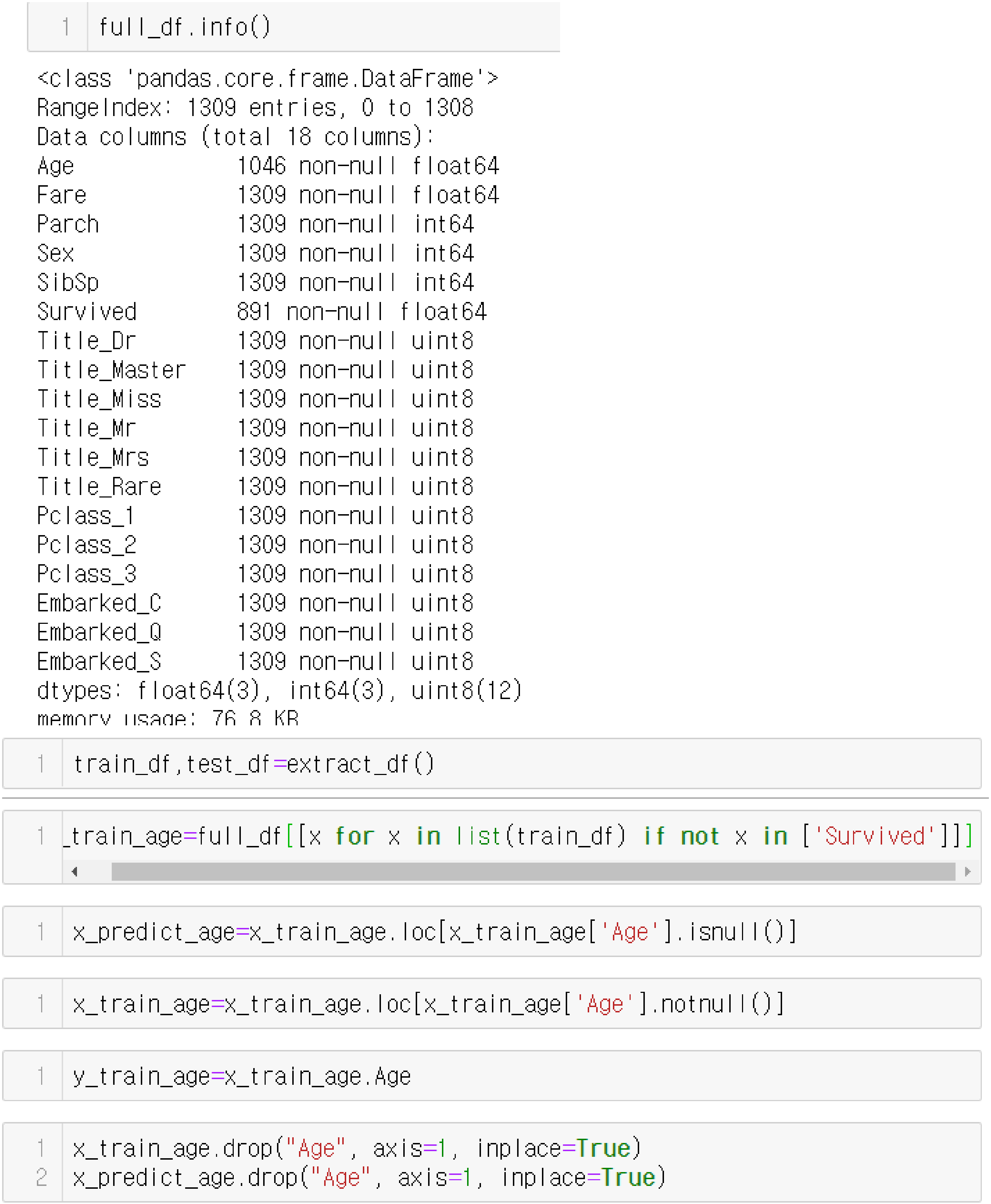
```
full_df['Sex']=full_df['Sex'].map({'female':0, 'male':1})
```

```
def onehot(df, feature_list): #원핫 인코딩
    df=pd.get_dummies(df, columns=feature_list)
    return df

onehot_list=['Title','Pclass','Embarked']
full_df=onehot(full_df, onehot_list)
```

```
full_df
```

	Age	Fare	Parch	Sex	SibSp	Survived	Title_Dr	Title_Master	Title_Miss	Tit
0	22.0	7.2500	0	1	1	0.0	0	0	0	
1	38.0	71.2833	0	0	1	1.0	0	0	0	
2	26.0	7.9250	0	0	0	1.0	0	0	1	
3	35.0	53.1000	0	0	1	1.0	0	0	0	
4	35.0	8.0500	0	1	0	0.0	0	0	0	
...	...	...	...	...	...	...	...	...	...	
1304	NaN	8.0500	0	1	0	NaN	0	0	0	
1305	39.0	108.9000	0	0	0	NaN	0	0	0	
1306	38.5	7.2500	0	1	0	NaN	0	0	0	
1307	NaN	8.0500	0	1	0	NaN	0	0	0	
1308	NaN	22.3583	1	1	1	NaN	0	1	0	

1309 rows × 18 columns



```
full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 18 columns):
Age             1046 non-null float64
Fare            1309 non-null float64
Parch           1309 non-null int64
Sex             1309 non-null int64
SibSp           1309 non-null int64
Survived        891 non-null float64
Title_Dr        1309 non-null uint8
Title_Master    1309 non-null uint8
Title_Miss      1309 non-null uint8
Title_Mr        1309 non-null uint8
Title_Mrs       1309 non-null uint8
Title_Rare      1309 non-null uint8
Pclass_1        1309 non-null uint8
Pclass_2        1309 non-null uint8
Pclass_3        1309 non-null uint8
Embarked_C      1309 non-null uint8
Embarked_Q      1309 non-null uint8
Embarked_S      1309 non-null uint8
dtypes: float64(3), int64(3), uint8(12)
memory usage: 76.8 KB
```

```
train_df,test_df=extract_df()
```

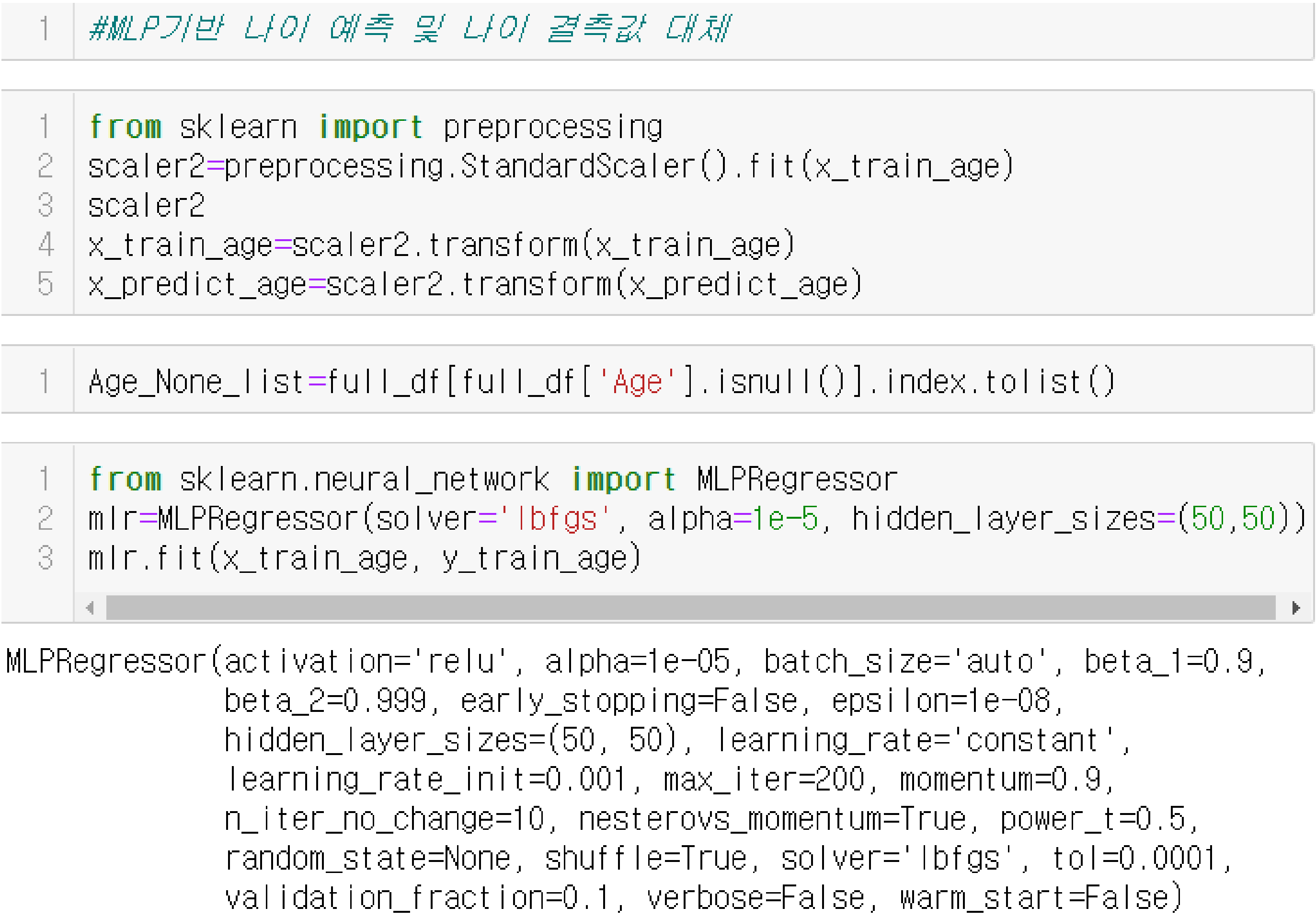
```
_train_age=full_df[[x for x in list(train_df) if not x in ['Survived']]]
```

```
x_predict_age=x_train_age.loc[x_train_age['Age'].isnull()]
```

```
x_train_age=x_train_age.loc[x_train_age['Age'].notnull()]
```

```
y_train_age=x_train_age.Age
```

```
x_train_age.drop("Age", axis=1, inplace=True)
x_predict_age.drop("Age", axis=1, inplace=True)
```



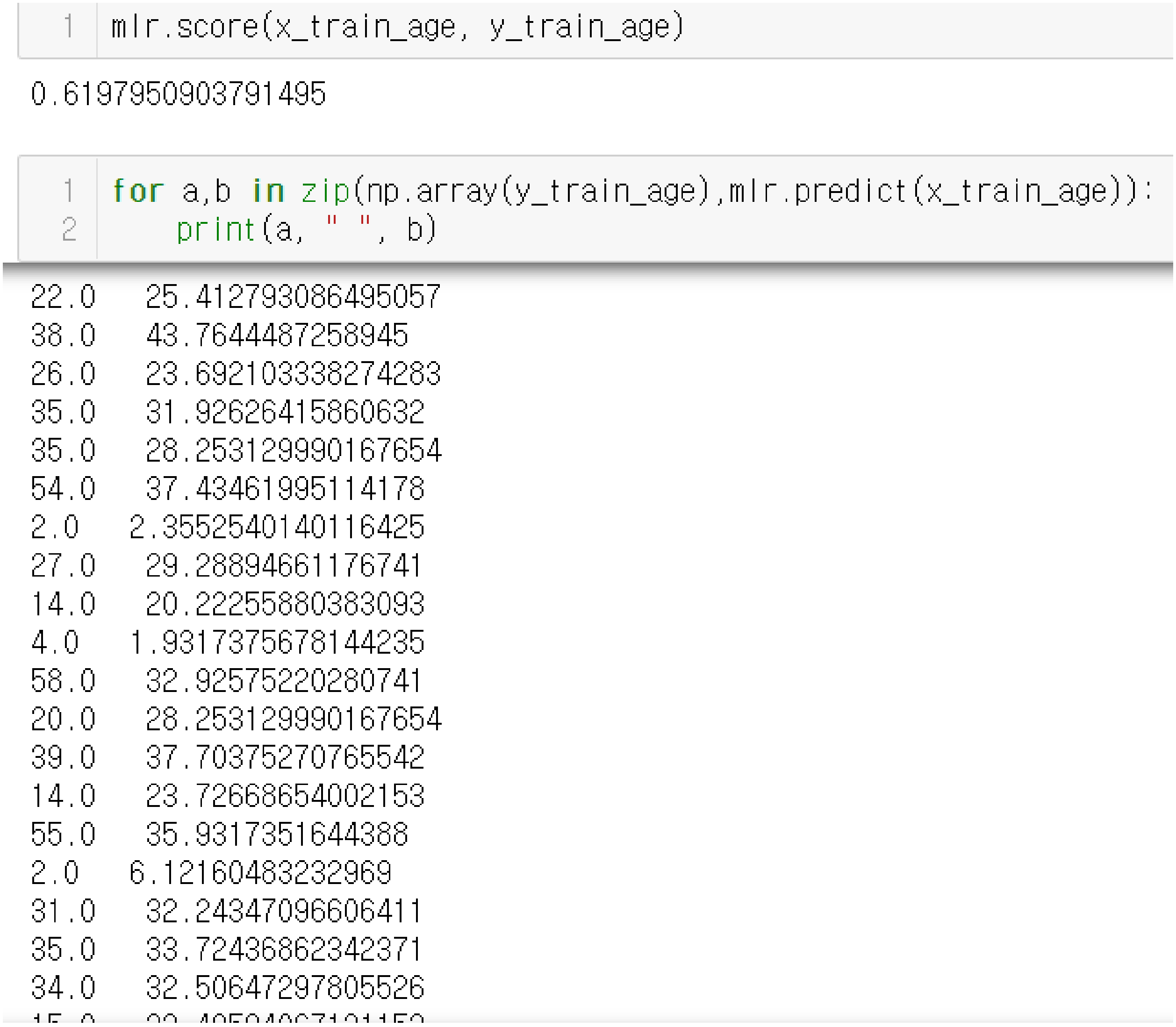
```
#MLP기반 나이 예측 및 나이 결측값 대체
```

```
from sklearn import preprocessing
scaler2=preprocessing.StandardScaler().fit(x_train_age)
scaler2
x_train_age=scaler2.transform(x_train_age)
x_predict_age=scaler2.transform(x_predict_age)
```

```
Age_None_list=full_df[full_df['Age'].isnull()].index.tolist()
```

```
from sklearn.neural_network import MLPRegressor
mlr=MLPRegressor(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(50,50))
mlr.fit(x_train_age, y_train_age)
```

```
MLPRegressor(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
             beta_2=0.999, early_stopping=False, epsilon=1e-08,
             hidden_layer_sizes=(50, 50), learning_rate='constant',
             learning_rate_init=0.001, max_iter=200, momentum=0.9,
             n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
             random_state=None, shuffle=True, solver='lbfgs', tol=0.0001,
             validation_fraction=0.1, verbose=False, warm_start=False)
```

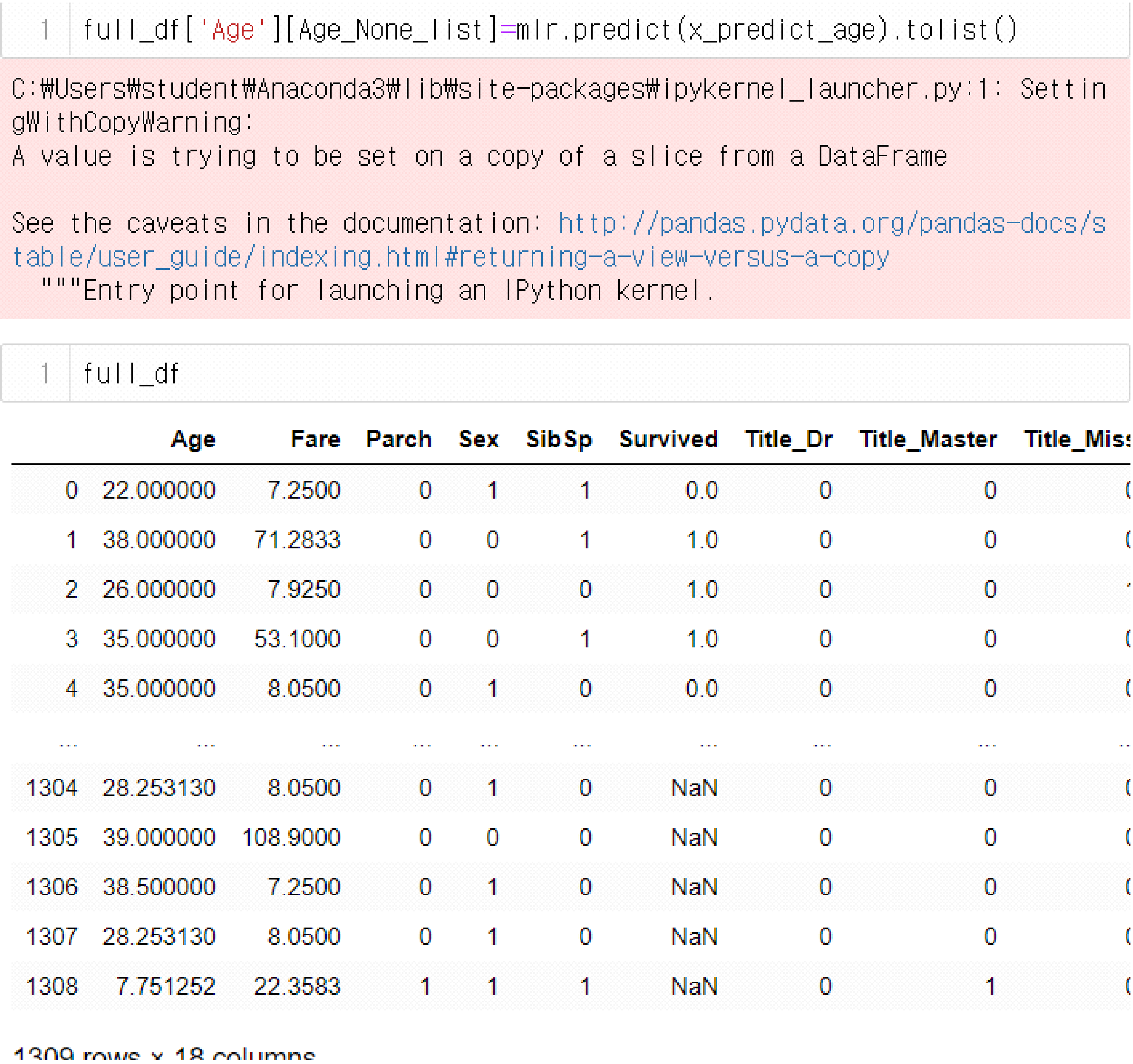


```
mlr.score(x_train_age, y_train_age)
```

```
0.6197950903791495
```

```
for a,b in zip(np.array(y_train_age),mlr.predict(x_train_age)):
    print(a, " ", b)
```

```
22.0   25.412793086495057
38.0   43.7644487258945
26.0   23.692103338274283
35.0   31.92626415860632
35.0   28.253129990167654
54.0   37.43461995114178
2.0   2.3552540140116425
27.0   29.28894661176741
14.0   20.22255880383093
4.0   1.9317375678144235
58.0   32.92575220280741
20.0   28.253129990167654
39.0   37.70375270765542
14.0   23.72668654002153
55.0   35.9317351644388
2.0   6.12160483232969
31.0   32.24347096606411
35.0   33.72436862342371
34.0   32.50647297805526
15.0   22.40524067121152
```



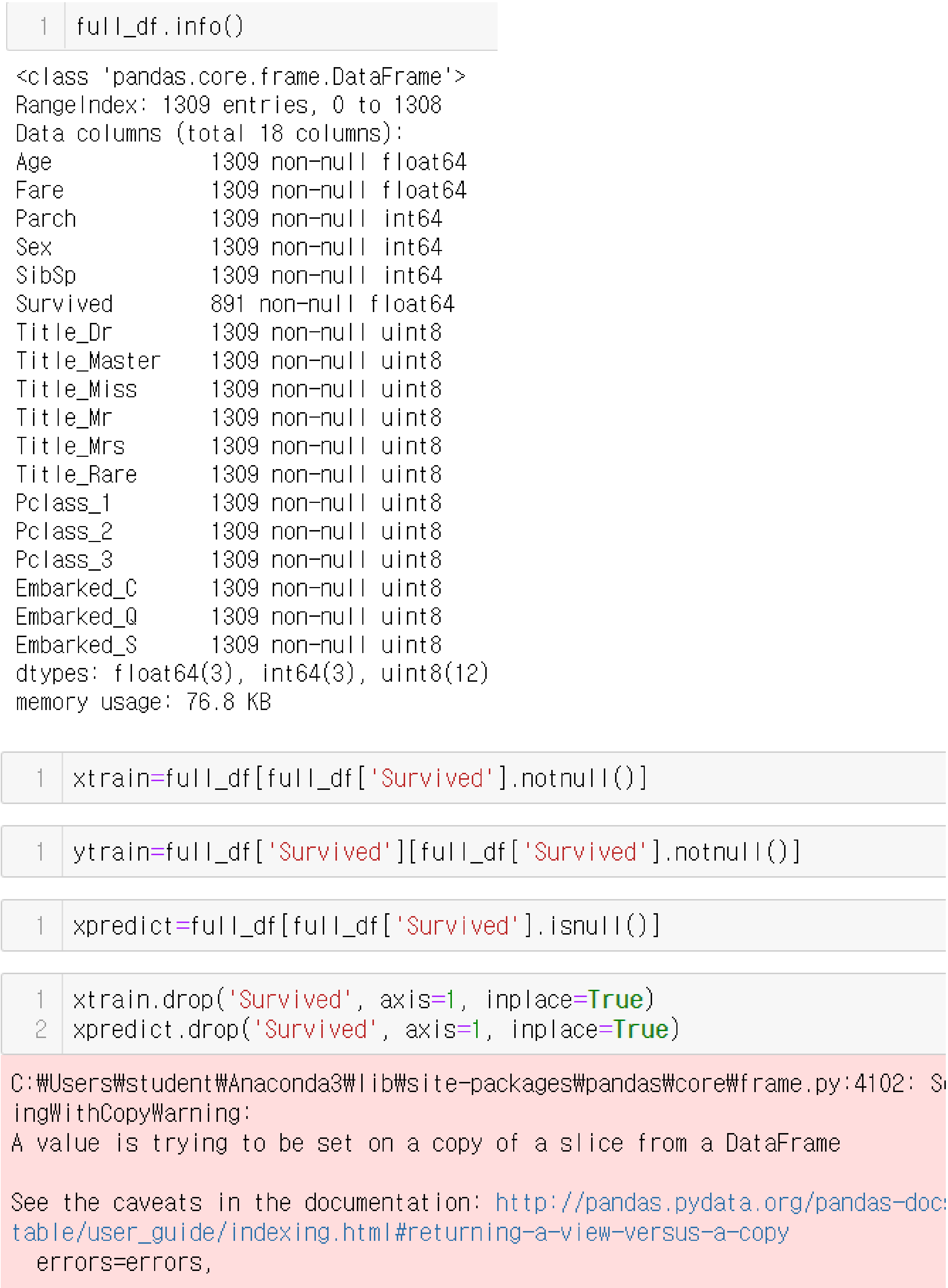
```python
full_df['Age'][Age_None_list]=mlr.predict(x_predict_age).tolist()
```

```
C:\Users\student\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

```python
full_df
```

	Age	Fare	Parch	Sex	SibSp	Survived	Title_Dr	Title_Master	Title_Miss
0	22.000000	7.2500	0	1	1	0.0	0	0	0
1	38.000000	71.2833	0	0	1	1.0	0	0	0
2	26.000000	7.9250	0	0	0	1.0	0	0	1
3	35.000000	53.1000	0	0	1	1.0	0	0	0
4	35.000000	8.0500	0	1	0	0.0	0	0	0
...	...	...	...	...	...	...	...	...	..
1304	28.253130	8.0500	0	1	0	NaN	0	0	0
1305	39.000000	108.9000	0	0	0	NaN	0	0	0
1306	38.500000	7.2500	0	1	0	NaN	0	0	0
1307	28.253130	8.0500	0	1	0	NaN	0	0	0
1308	7.751252	22.3583	1	1	1	NaN	0	1	0

1309 rows × 18 columns



```
full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 18 columns):
Age               1309 non-null float64
Fare              1309 non-null float64
Parch             1309 non-null int64
Sex               1309 non-null int64
SibSp             1309 non-null int64
Survived          891 non-null float64
Title_Dr          1309 non-null uint8
Title_Master      1309 non-null uint8
Title_Miss        1309 non-null uint8
Title_Mr          1309 non-null uint8
Title_Mrs         1309 non-null uint8
Title_Rare        1309 non-null uint8
Pclass_1          1309 non-null uint8
Pclass_2          1309 non-null uint8
Pclass_3          1309 non-null uint8
Embarked_C        1309 non-null uint8
Embarked_Q        1309 non-null uint8
Embarked_S        1309 non-null uint8
dtypes: float64(3), int64(3), uint8(12)
memory usage: 76.8 KB
```

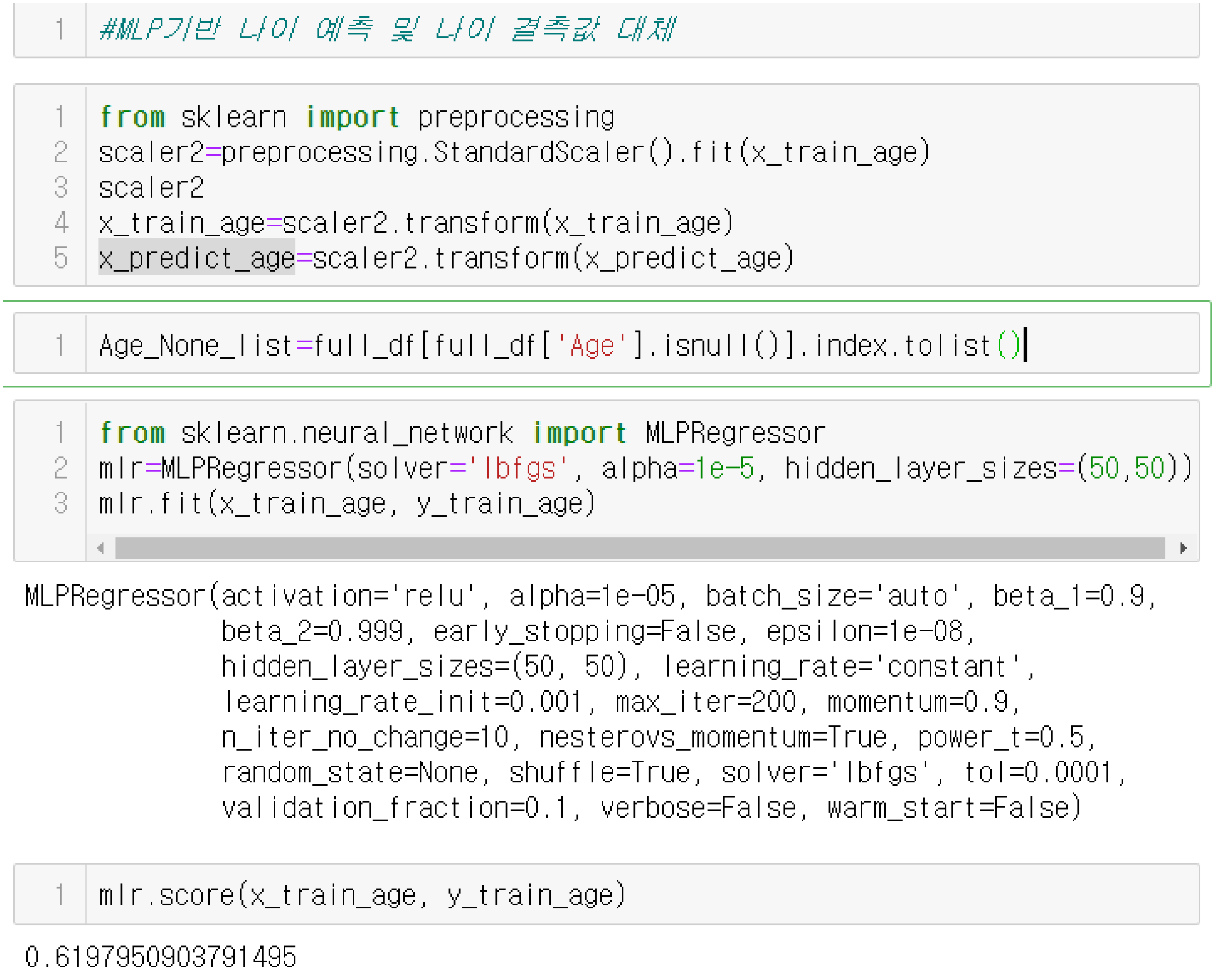
```
xtrain=full_df[full_df['Survived'].notnull()]
```

```
ytrain=full_df['Survived'][full_df['Survived'].notnull()]
```

```
xpredict=full_df[full_df['Survived'].isnull()]
```

```
xtrain.drop('Survived', axis=1, inplace=True)
xpredict.drop('Survived', axis=1, inplace=True)
```

```
C:₩Users₩student₩Anaconda3₩lib₩site-packages₩pandas₩core₩frame.py:4102: S
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-doc
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
```



```python
#MLP기반 나이 예측 및 나이 결측값 대체
```

```python
from sklearn import preprocessing
scaler2=preprocessing.StandardScaler().fit(x_train_age)
scaler2
x_train_age=scaler2.transform(x_train_age)
x_predict_age=scaler2.transform(x_predict_age)
```

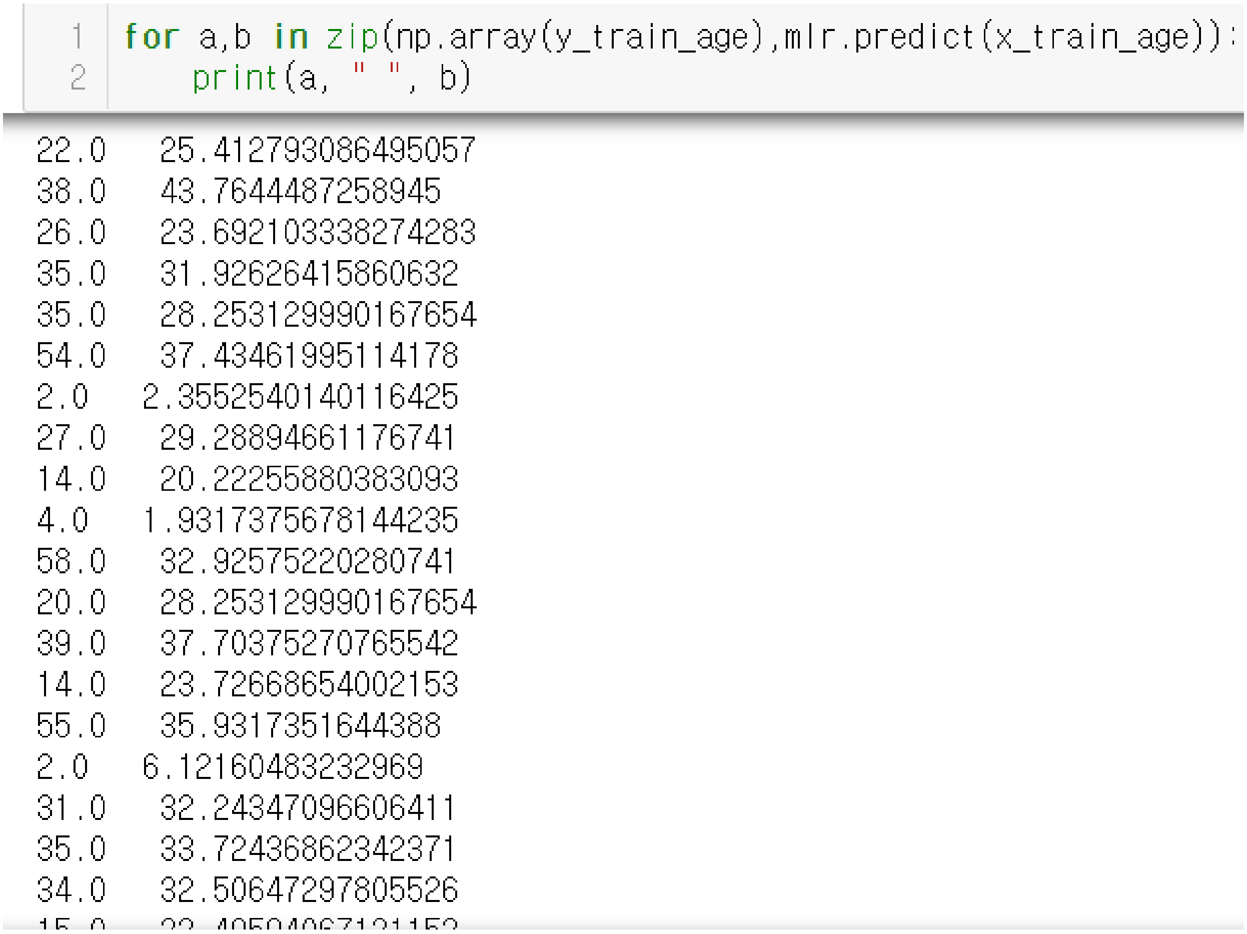
```python
Age_None_list=full_df[full_df['Age'].isnull()].index.tolist()
```

```python
from sklearn.neural_network import MLPRegressor
mlr=MLPRegressor(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(50,50))
mlr.fit(x_train_age, y_train_age)
```

```
MLPRegressor(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
             beta_2=0.999, early_stopping=False, epsilon=1e-08,
             hidden_layer_sizes=(50, 50), learning_rate='constant',
             learning_rate_init=0.001, max_iter=200, momentum=0.9,
             n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
             random_state=None, shuffle=True, solver='lbfgs', tol=0.0001,
             validation_fraction=0.1, verbose=False, warm_start=False)
```

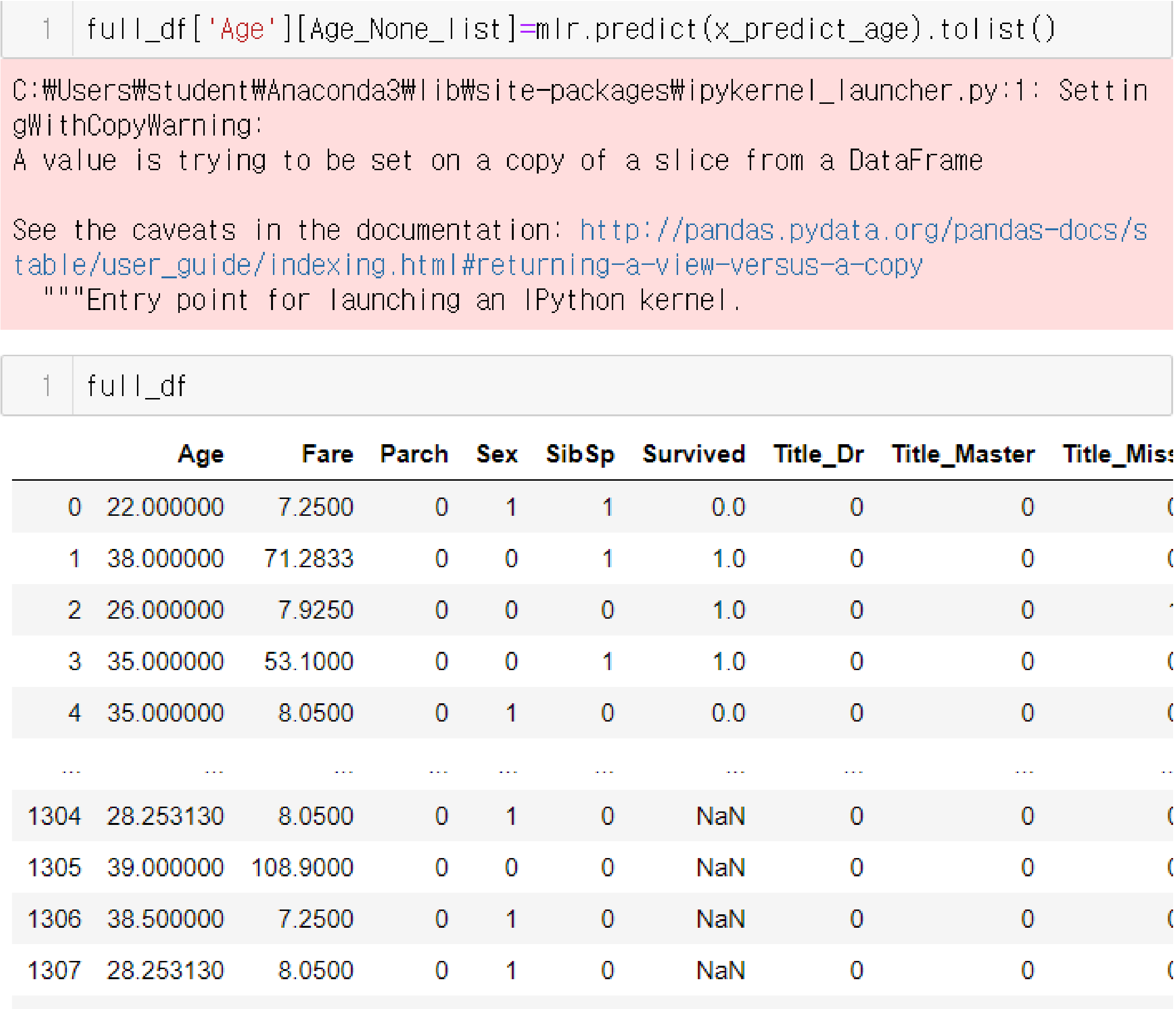
```python
mlr.score(x_train_age, y_train_age)
```

```
0.6197950903791495
```



```python
for a,b in zip(np.array(y_train_age),mlr.predict(x_train_age)):
    print(a, " ", b)
```

```
22.0   25.412793086495057
38.0   43.7644487258945
26.0   23.692103338274283
35.0   31.92626415860632
35.0   28.253129990167654
54.0   37.43461995114178
2.0   2.3552540140116425
27.0   29.28894661176741
14.0   20.22255880383093
4.0   1.9317375678144235
58.0   32.92575220280741
20.0   28.253129990167654
39.0   37.70375270765542
14.0   23.72668654002153
55.0   35.9317351644388
2.0   6.12160483232969
31.0   32.24347096606411
35.0   33.72436862342371
34.0   32.50647297805526
15.0   22.40504067121153
```

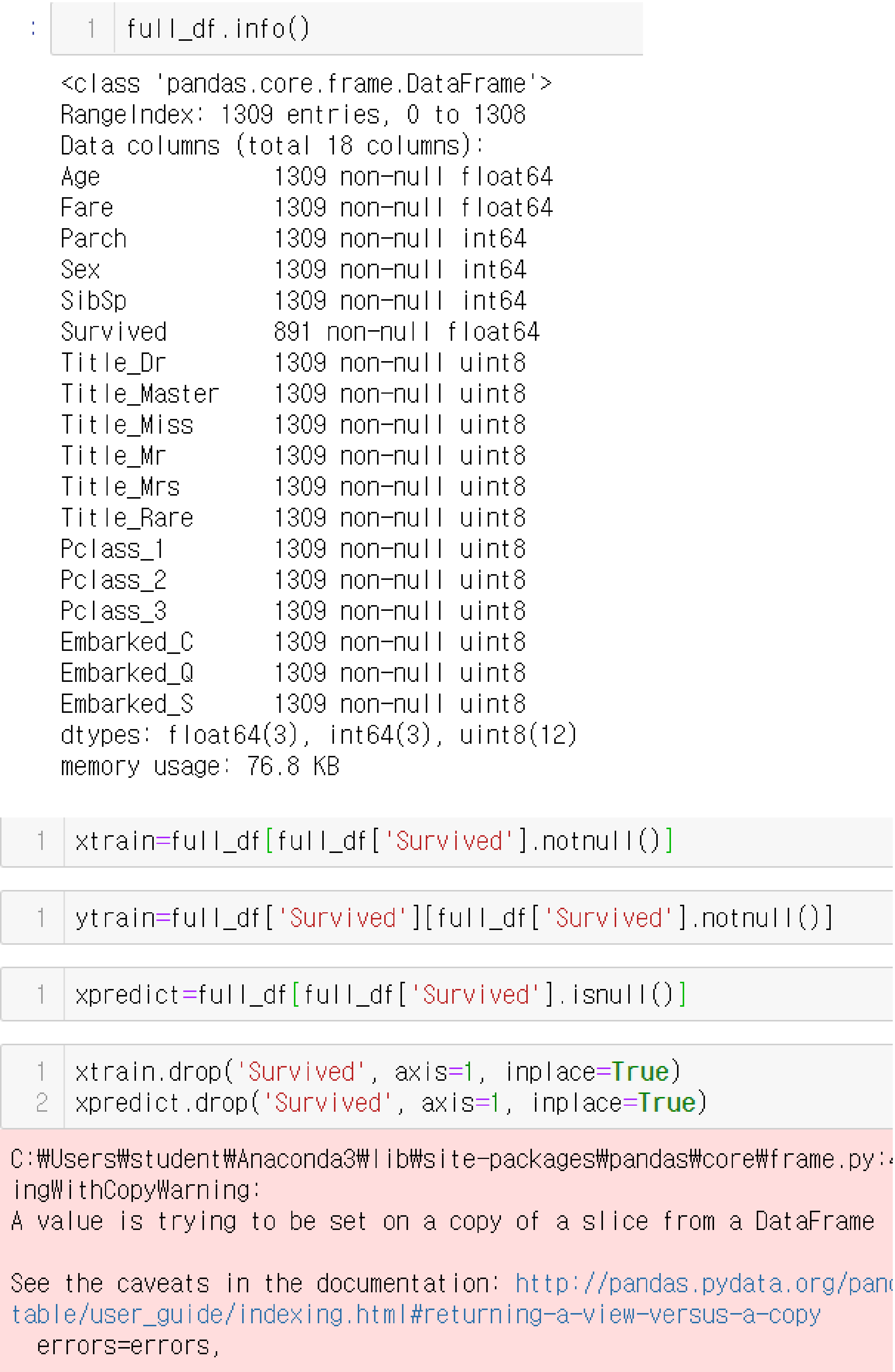


```python
full_df['Age'][Age_None_list]=mlr.predict(x_predict_age).tolist()
```

```
C:₩Users₩student₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:1: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

```python
full_df
```

	Age	Fare	Parch	Sex	SibSp	Survived	Title_Dr	Title_Master	Title_Mis
0	22.000000	7.2500	0	1	1	0.0	0	0	0
1	38.000000	71.2833	0	0	1	1.0	0	0	0
2	26.000000	7.9250	0	0	0	1.0	0	0	1
3	35.000000	53.1000	0	0	1	1.0	0	0	0
4	35.000000	8.0500	0	1	0	0.0	0	0	0
...	...	...	...	...	...	...	...	...	..
1304	28.253130	8.0500	0	1	0	NaN	0	0	0
1305	39.000000	108.9000	0	0	0	NaN	0	0	0
1306	38.500000	7.2500	0	1	0	NaN	0	0	0
1307	28.253130	8.0500	0	1	0	NaN	0	0	0



```
full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 18 columns):
Age             1309 non-null float64
Fare            1309 non-null float64
Parch           1309 non-null int64
Sex             1309 non-null int64
SibSp           1309 non-null int64
Survived        891 non-null float64
Title_Dr        1309 non-null uint8
Title_Master    1309 non-null uint8
Title_Miss      1309 non-null uint8
Title_Mr        1309 non-null uint8
Title_Mrs       1309 non-null uint8
Title_Rare      1309 non-null uint8
Pclass_1        1309 non-null uint8
Pclass_2        1309 non-null uint8
Pclass_3        1309 non-null uint8
Embarked_C      1309 non-null uint8
Embarked_Q      1309 non-null uint8
Embarked_S      1309 non-null uint8
dtypes: float64(3), int64(3), uint8(12)
memory usage: 76.8 KB
```

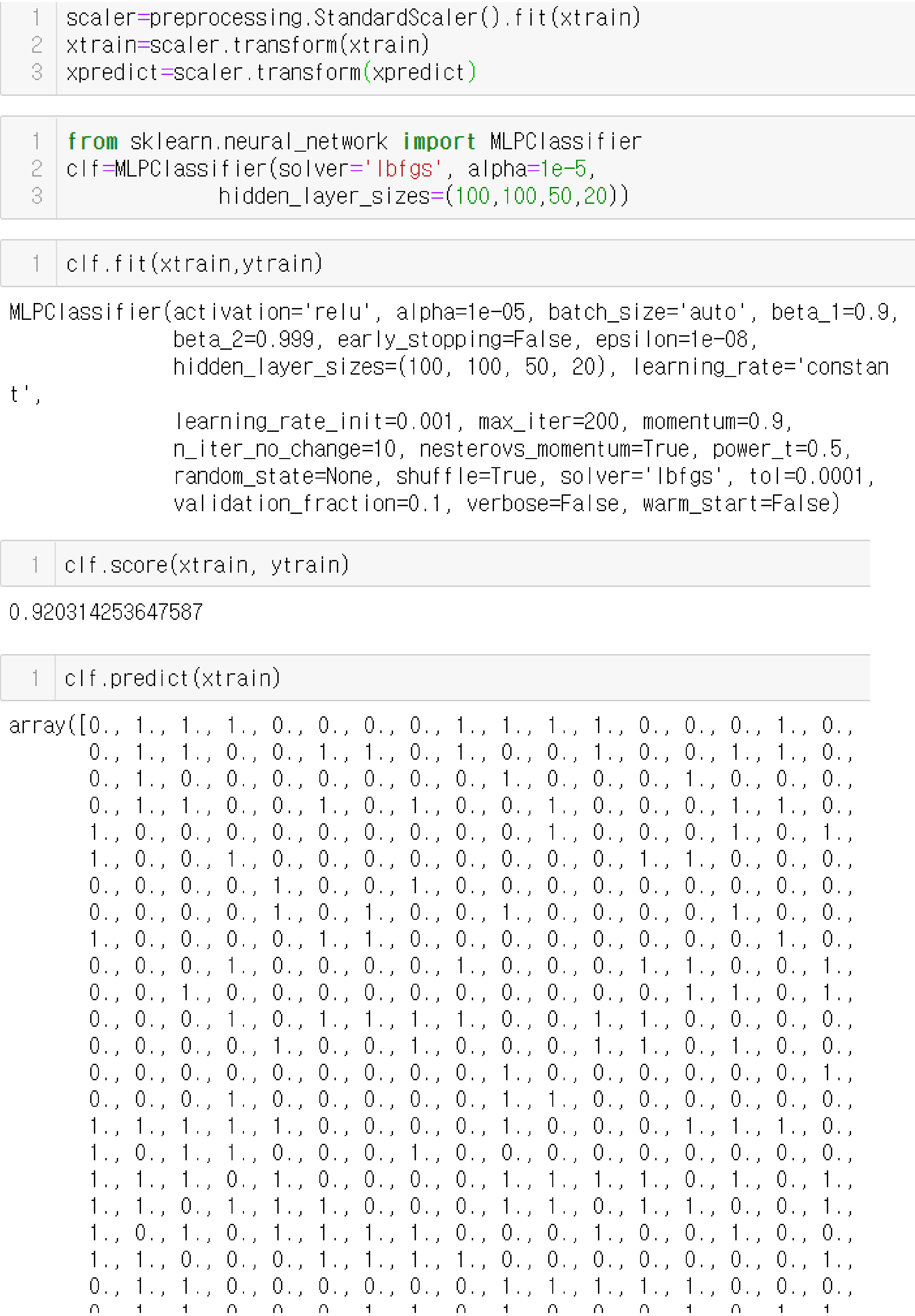
```
xtrain=full_df[full_df['Survived'].notnull()]
```

```
ytrain=full_df['Survived'][full_df['Survived'].notnull()]
```

```
xpredict=full_df[full_df['Survived'].isnull()]
```

```
xtrain.drop('Survived', axis=1, inplace=True)
xpredict.drop('Survived', axis=1, inplace=True)
```

```
C:₩Users₩student₩Anaconda3₩lib₩site-packages₩pandas₩core₩frame.py:
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pand
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
```



```python
scaler=preprocessing.StandardScaler().fit(xtrain)
xtrain=scaler.transform(xtrain)
xpredict=scaler.transform(xpredict)
```

```python
from sklearn.neural_network import MLPClassifier
clf=MLPClassifier(solver='lbfgs', alpha=1e-5,
                  hidden_layer_sizes=(100,100,50,20))
```

```python
clf.fit(xtrain,ytrain)
```

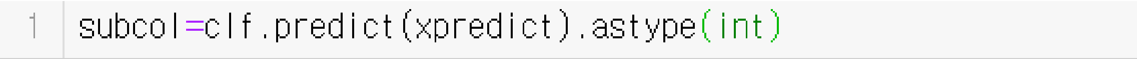
```
MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100, 100, 50, 20), learning_rate='constan
t',
              learning_rate_init=0.001, max_iter=200, momentum=0.9,
              n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
              random_state=None, shuffle=True, solver='lbfgs', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)
```

```python
clf.score(xtrain, ytrain)
```

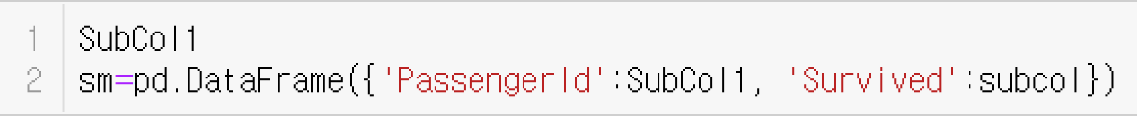
```
0.9203142536475870
```

```python
clf.predict(xtrain)
```

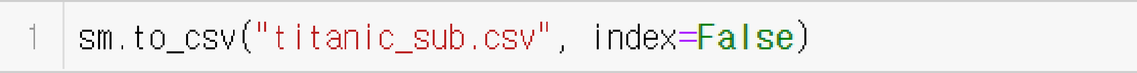
```
array([0., 1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 1., 0.,
       0., 1., 1., 0., 0., 1., 1., 0., 1., 0., 0., 1., 0., 0., 1., 1., 0.,
       0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0.,
       0., 1., 1., 0., 0., 1., 0., 1., 0., 0., 1., 0., 0., 0., 1., 1., 0.,
       1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 1.,
       1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0.,
       0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 0., 1., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0.,
       1., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0.,
       0., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 0., 1.,
       0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 1.,
       0., 0., 0., 1., 0., 1., 1., 1., 1., 0., 0., 1., 1., 0., 0., 0., 0.,
       0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 1., 0., 1., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1.,
       0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0.,
       1., 1., 1., 1., 1., 0., 0., 0., 0., 1., 0., 0., 0., 1., 1., 1., 0.,
       1., 0., 1., 1., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       1., 1., 1., 0., 1., 0., 0., 0., 0., 1., 1., 1., 1., 0., 1., 0., 1.,
       1., 1., 0., 1., 1., 1., 0., 0., 0., 1., 1., 0., 1., 1., 0., 0., 1.,
       1., 0., 1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0.,
       1., 1., 0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 1.,
       0., 1., 1., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0.,
       0   1   1   0   0   0   1   1   0   1   0   0   0   1   0   1   1
```



```
subcol=clf.predict(xpredict).astype(int)
```



```
SubCol1
sm=pd.DataFrame({'PassengerId':SubCol1, 'Survived':subcol})
```



```
sm.to_csv("titanic_sub.csv", index=False)
```