4/17(금) 케라스, mnist(손글씨데이터),당뇨, 주식, 암

2020년 4월 17일 금요일 오전 9:10

- 모듈화 : 모듈 독립적 → 조합 →
  (함수, 클래스)
  layer, cost, optimizer, activation

- 케라스 모델 생성 절차

1) 데이터셋 생성
   - 훈련, 검증, 테스트

2) 모델 구성
   - 시퀀스 모델 생성한 다음에 레이어를 추가 (간단한 모델)
   - 복잡한 모델은 케라스 함수 API 사용

3) 모델 학습 과정
   - cost 함수, 최적화 방법 정의
   - compile 함수가 사용됨

4) 모델 학습
   - 트레이닝 데이터로 모델 학습
   - fit 함수가 사용됨

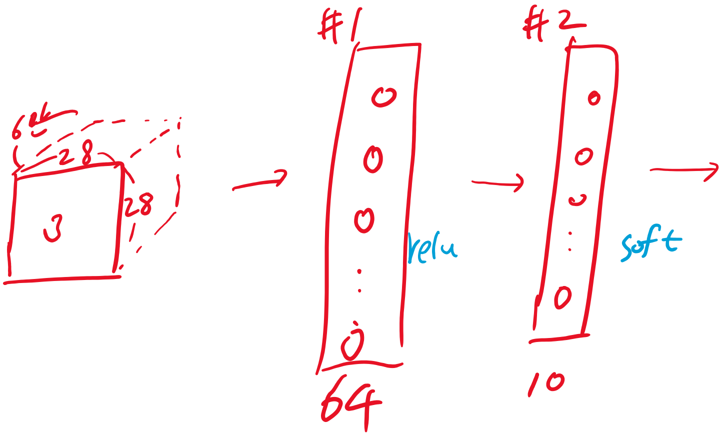
5) 훈련셋, 검증셋의 cost 측정, 정확도 측정

6) 모델 평가
   - test 데이터셋으로 평가
   - evaluate 함수가 사용됨

7) 모델 사용

- 입력 → 모델 → 출력(예측...)
- predict 함수가 사용됨

주피터 콘솔 pip install keras



W개수: 784 x 64

훈련셋 : 모의고사 → 학생 5명 : 모델 5개

검증셋 :

테스트셋 : 수능
작년

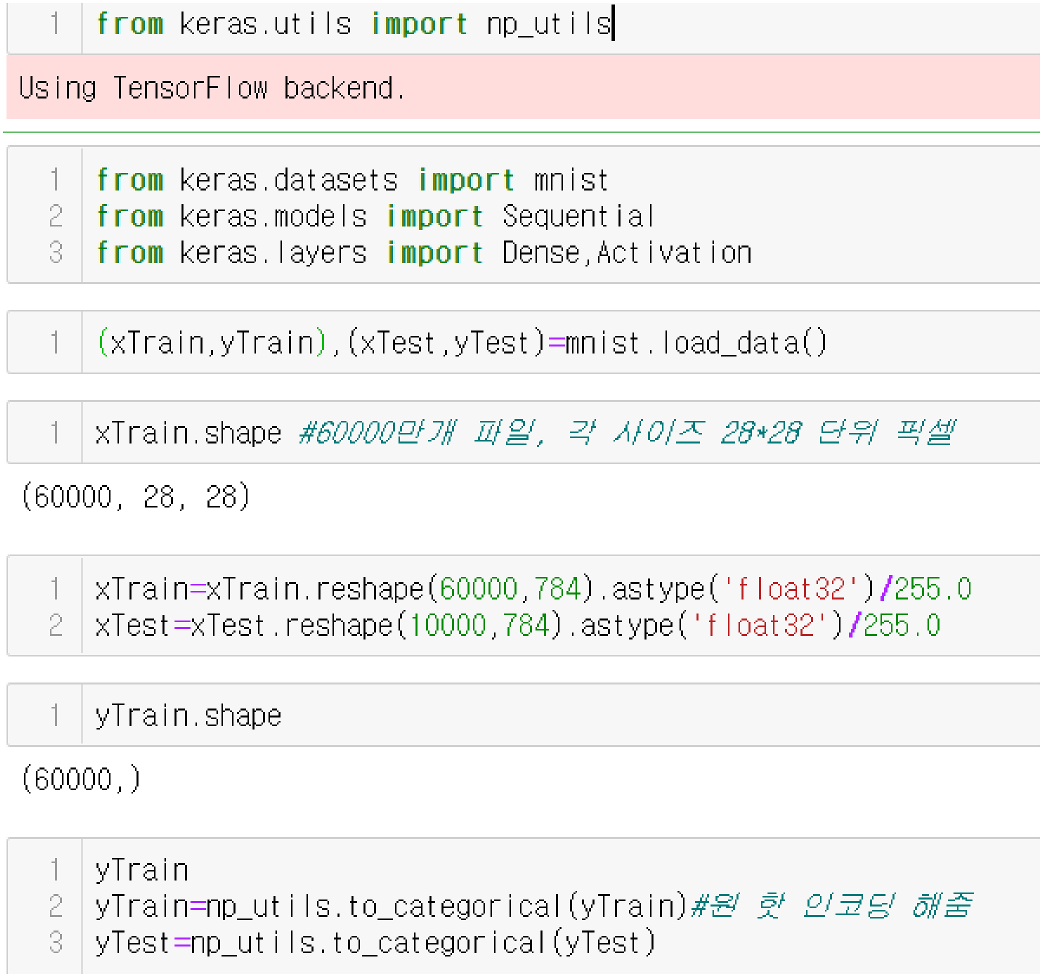
학습? 문제, 답안지 제공

① 훈련셋(70) : 시험셋(30) 실전
작년수능 올해수능

모의고사 1~4회 → 훈련셋 (70)

5회 → 검증셋(30)

하이퍼 파라미터 튜닝



```
from keras.utils import np_utils
```

```
Using TensorFlow backend.
```

```
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense,Activation
```

```
(xTrain,yTrain),(xTest,yTest)=mnist.load_data()
```

```
xTrain.shape #60000만개 파일, 각 사이즈 28*28 단위 픽셀
```

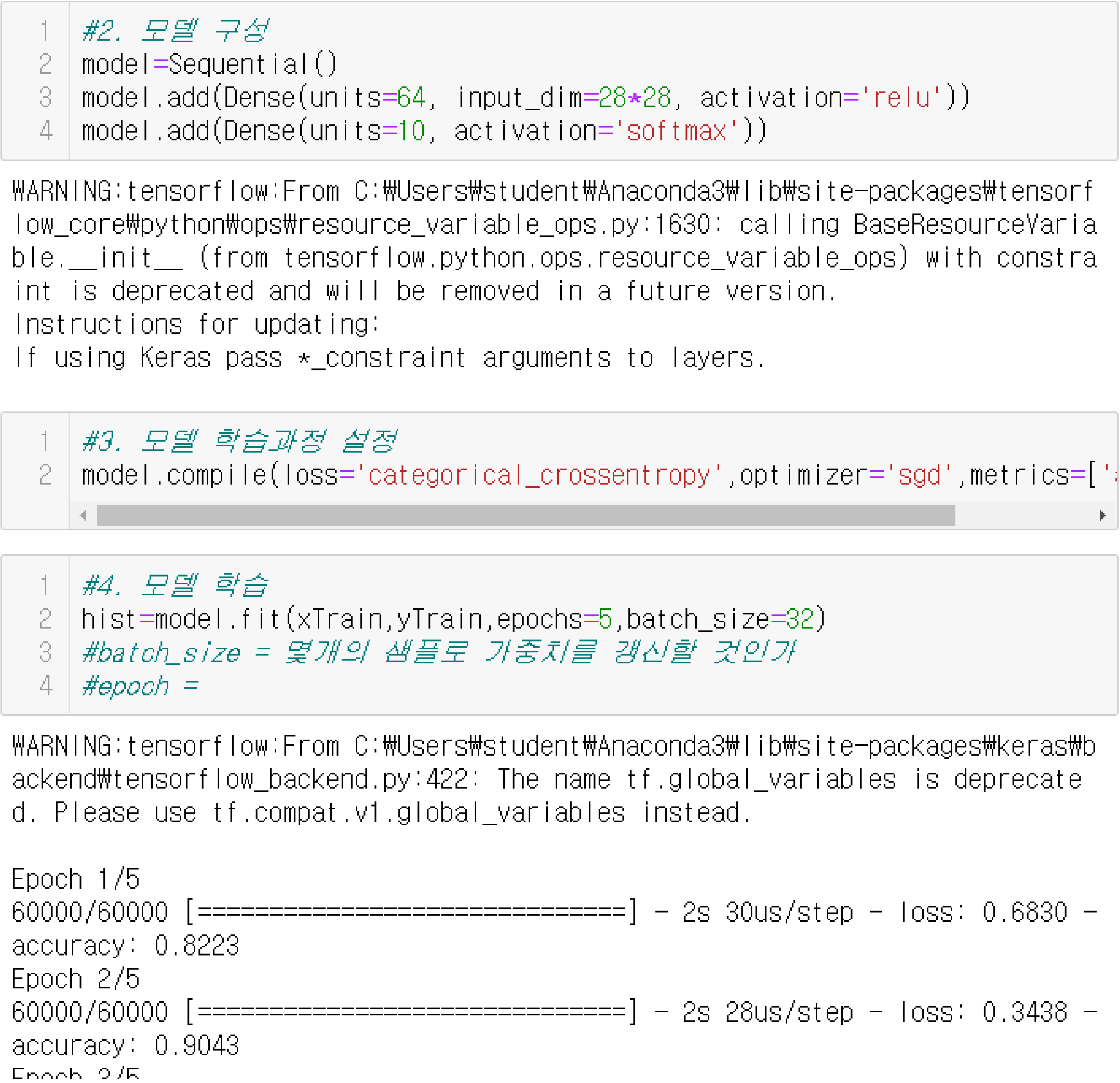
```
(60000, 28, 28)
```

```
xTrain=xTrain.reshape(60000,784).astype('float32')/255.0
xTest=xTest.reshape(10000,784).astype('float32')/255.0
```

```
yTrain.shape
```

```
(60000,)
```

```
yTrain
yTrain=np_utils.to_categorical(yTrain)#원 핫 인코딩 해줌
yTest=np_utils.to_categorical(yTest)
```



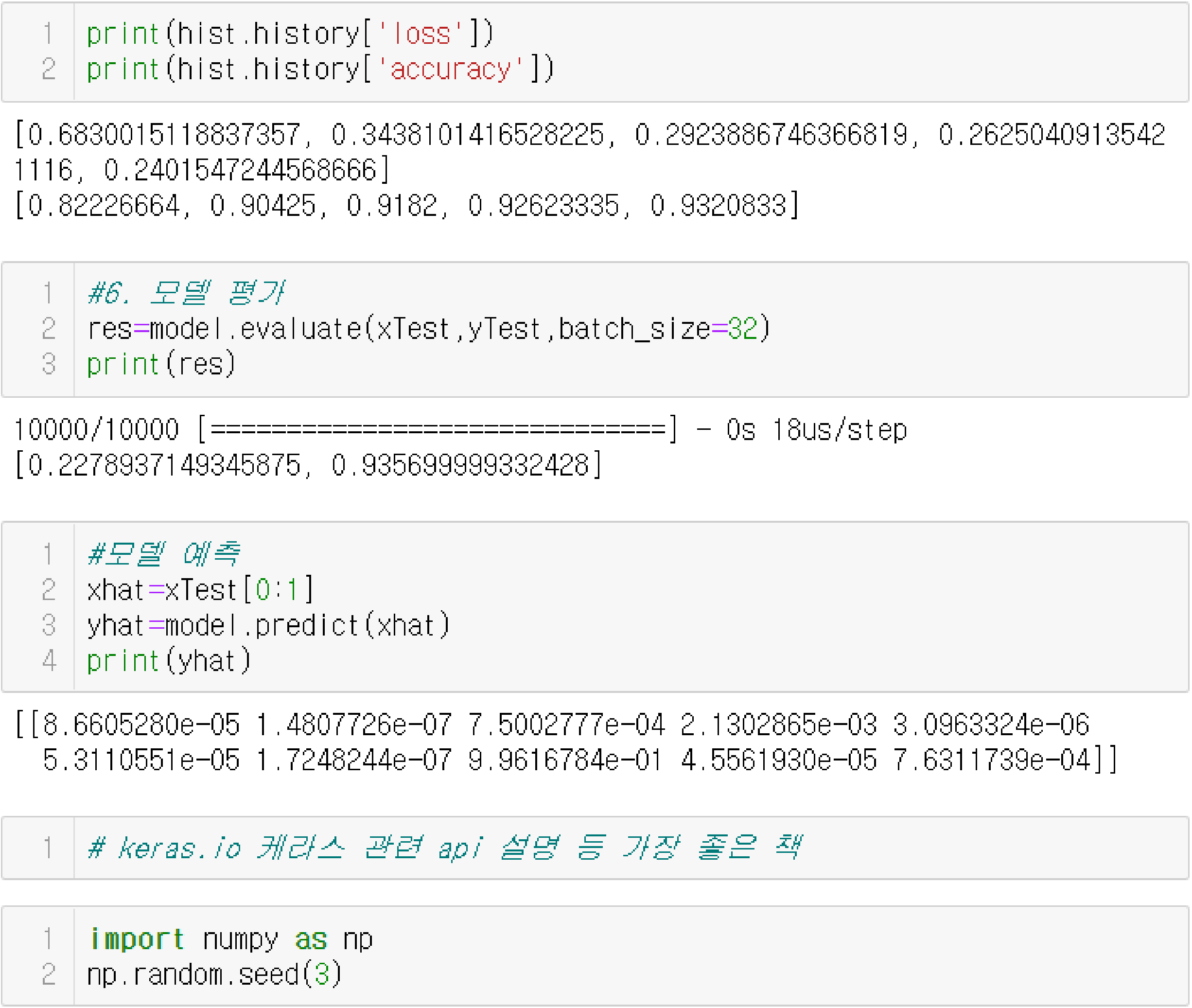
```
#2. 모델 구성
model=Sequential()
model.add(Dense(units=64, input_dim=28*28, activation='relu'))
model.add(Dense(units=10, activation='softmax'))
```

```
WARNING:tensorflow:From C:₩Users₩student₩Anaconda3₩lib₩site-packages₩tensorf
low_core₩python₩ops₩resource_variable_ops.py:1630: calling BaseResourceVaria
ble.__init__ (from tensorflow.python.ops.resource_variable_ops) with constra
int is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

```
#3. 모델 학습과정 설정
model.compile(loss='categorical_crossentropy',optimizer='sgd',metrics=['
```

```
#4. 모델 학습
hist=model.fit(xTrain,yTrain,epochs=5,batch_size=32)
#batch_size = 몇개의 샘플로 가중치를 갱신할 것인가
#epoch =
```

```
WARNING:tensorflow:From C:₩Users₩student₩Anaconda3₩lib₩site-packages₩keras₩b
ackend₩tensorflow_backend.py:422: The name tf.global_variables is deprecate
d. Please use tf.compat.v1.global_variables instead.

Epoch 1/5
60000/60000 [==============================] - 2s 30us/step - loss: 0.6830 -
accuracy: 0.8223
Epoch 2/5
60000/60000 [==============================] - 2s 28us/step - loss: 0.3438 -
accuracy: 0.9043
Epoch 3/5
```



```python
print(hist.history['loss'])
print(hist.history['accuracy'])
```

```
[0.6830015118837357, 0.3438101416528225, 0.2923886746366819, 0.2625040913542
1116, 0.24015472444568666]
[0.82226664, 0.90425, 0.9182, 0.92623335, 0.9320833]
```

```python
#6. 모델 평가
res=model.evaluate(xTest,yTest,batch_size=32)
print(res)
```

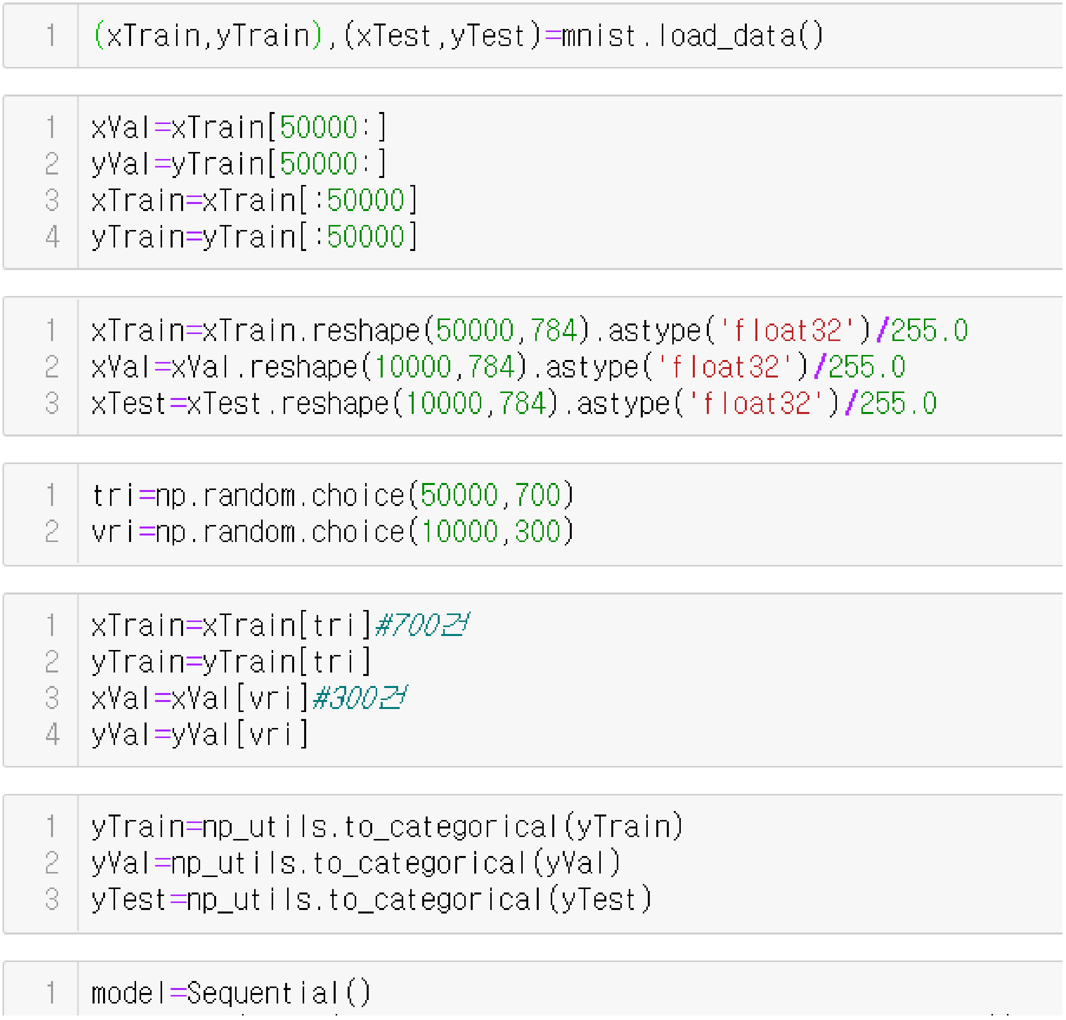
```
10000/10000 [==============================] - 0s 18us/step
[0.2278937149345875, 0.9356999999332428]
```

```python
#모델 예측
xhat=xTest[0:1]
yhat=model.predict(xhat)
print(yhat)
```

```
[[8.6605280e-05 1.4807726e-07 7.5002777e-04 2.1302865e-03 3.0963324e-06
  5.3110551e-05 1.7248244e-07 9.9616784e-01 4.5561930e-05 7.6311739e-04]]
```

```python
# keras.io 케라스 관련 api 설명 등 가장 좋은 책
```

```python
import numpy as np
np.random.seed(3)
```



```python
(xTrain,yTrain),(xTest,yTest)=mnist.load_data()
```

```python
xVal=xTrain[50000:]
yVal=yTrain[50000:]
xTrain=xTrain[:50000]
yTrain=yTrain[:50000]
```

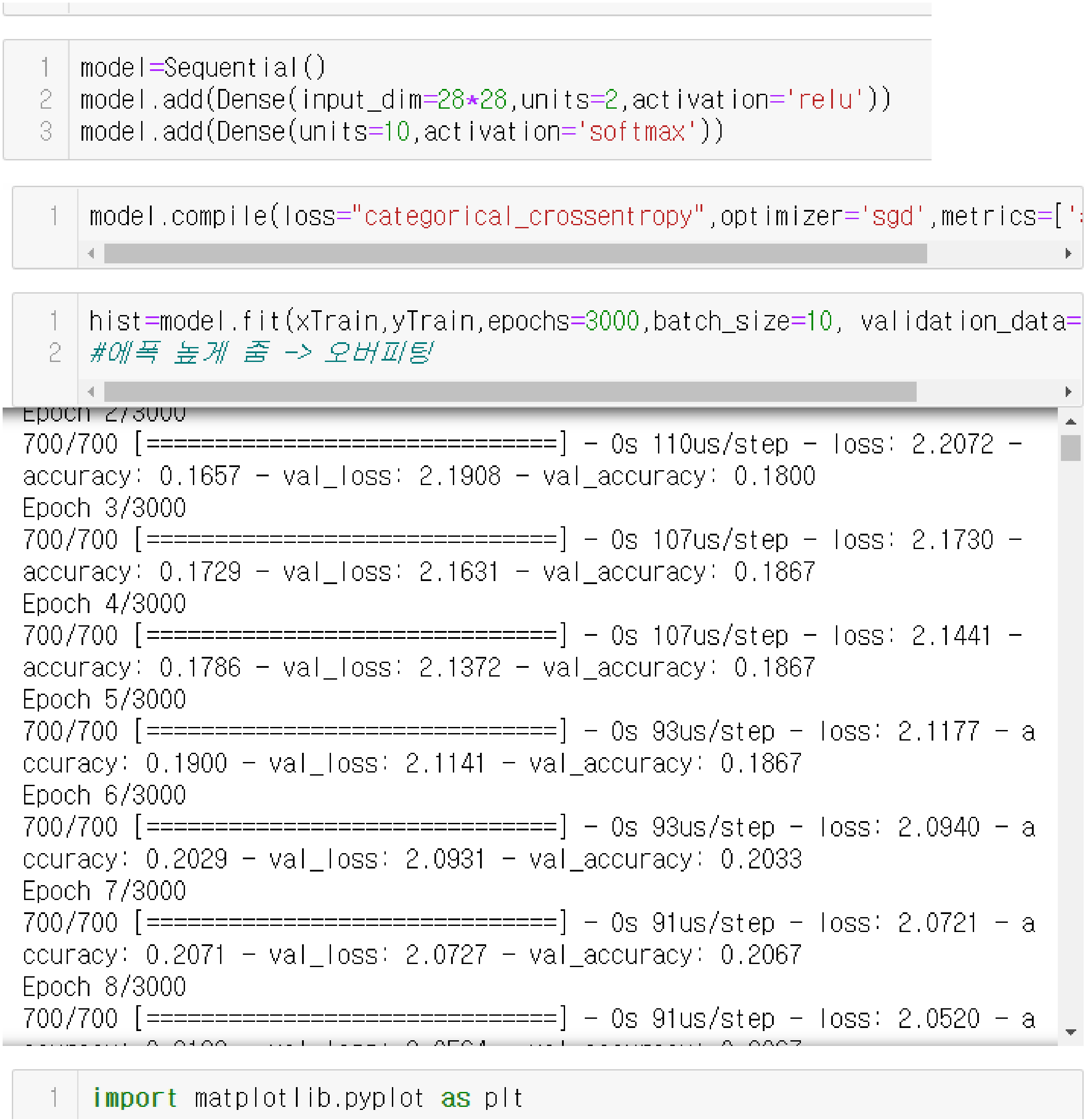
```python
xTrain=xTrain.reshape(50000,784).astype('float32')/255.0
xVal=xVal.reshape(10000,784).astype('float32')/255.0
xTest=xTest.reshape(10000,784).astype('float32')/255.0
```

```python
tri=np.random.choice(50000,700)
vri=np.random.choice(10000,300)
```

```python
xTrain=xTrain[tri]#700건
yTrain=yTrain[tri]
xVal=xVal[vri]#300건
yVal=yVal[vri]
```

```python
yTrain=np_utils.to_categorical(yTrain)
yVal=np_utils.to_categorical(yVal)
yTest=np_utils.to_categorical(yTest)
```

```python
model=Sequential()
```



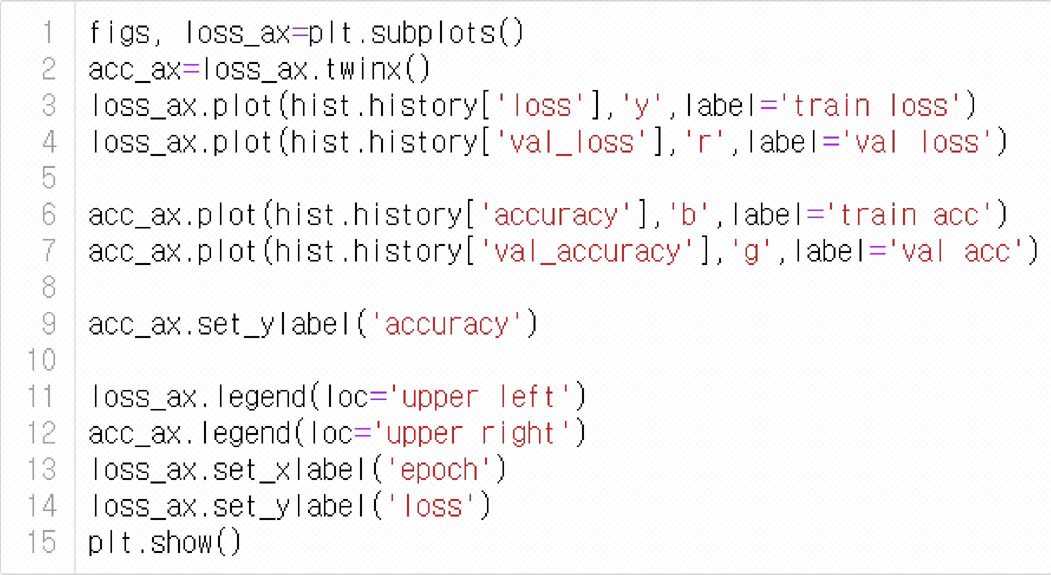
```python
model=Sequential()
model.add(Dense(input_dim=28*28,units=2,activation='relu'))
model.add(Dense(units=10,activation='softmax'))
```

```python
model.compile(loss="categorical_crossentropy",optimizer='sgd',metrics=['
```

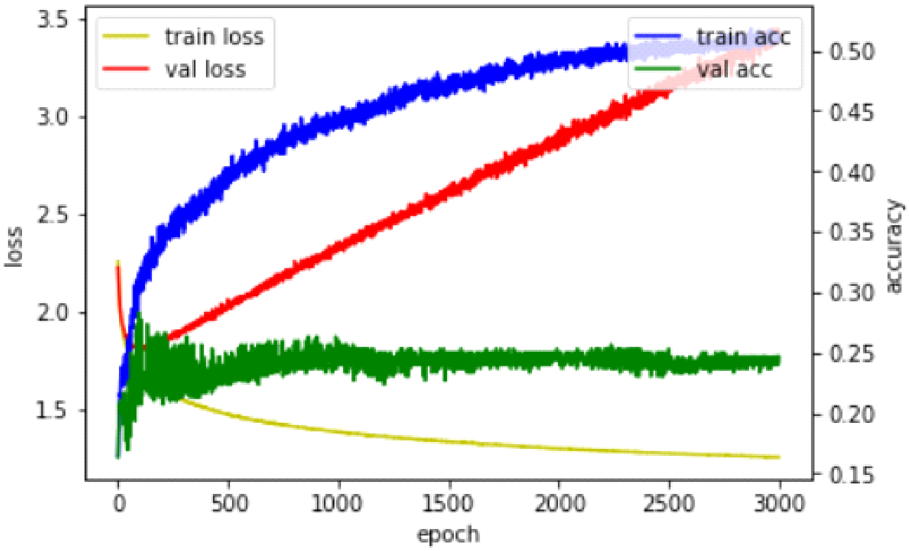
```python
hist=model.fit(xTrain,yTrain,epochs=3000,batch_size=10, validation_data=
#에폭 높게 줌 -> 오버피팅
```

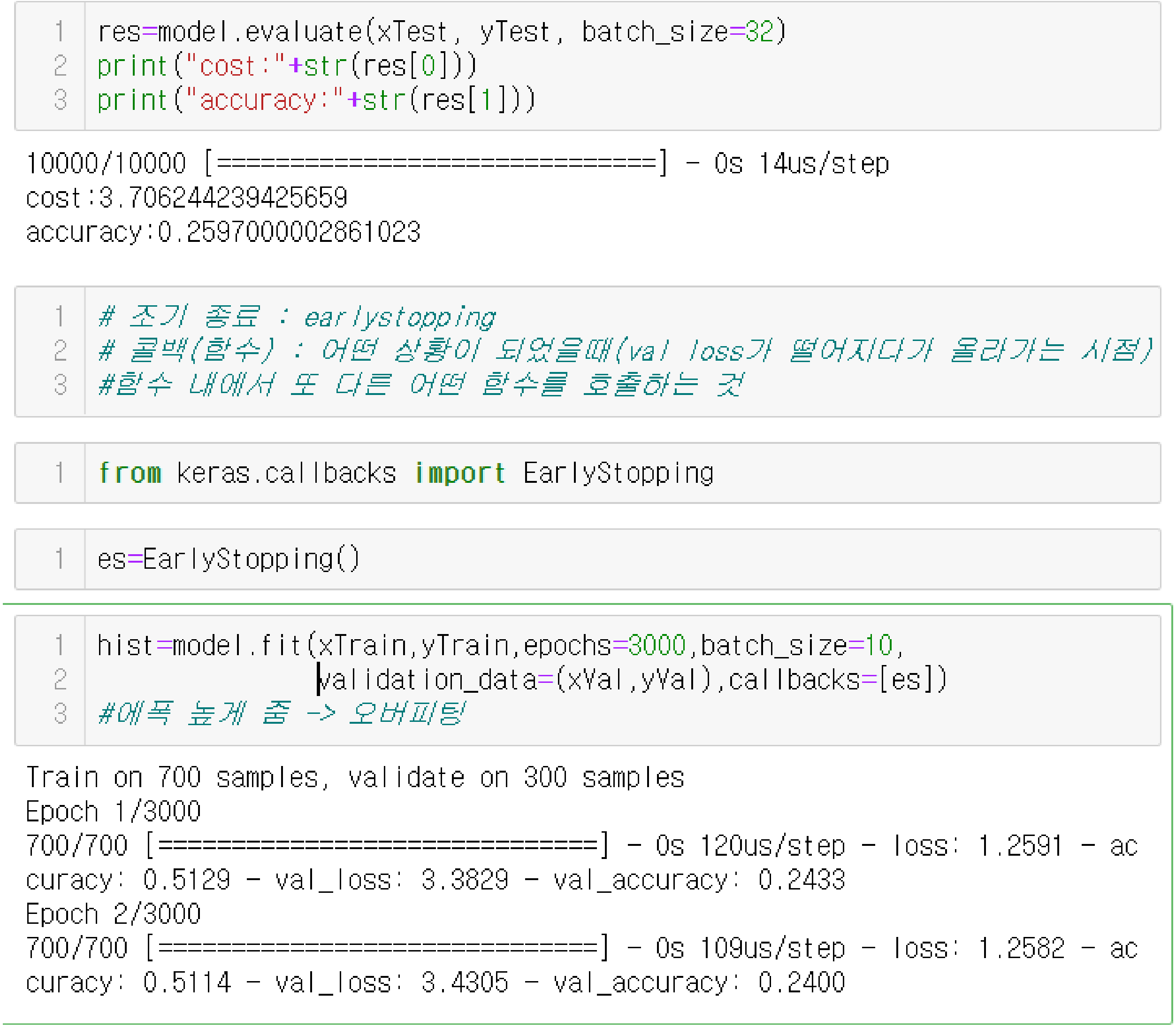
```
Epoch 2/3000
700/700 [==============================] - 0s 110us/step - loss: 2.2072 -
accuracy: 0.1657 - val_loss: 2.1908 - val_accuracy: 0.1800
Epoch 3/3000
700/700 [==============================] - 0s 107us/step - loss: 2.1730 -
accuracy: 0.1729 - val_loss: 2.1631 - val_accuracy: 0.1867
Epoch 4/3000
700/700 [==============================] - 0s 107us/step - loss: 2.1441 -
accuracy: 0.1786 - val_loss: 2.1372 - val_accuracy: 0.1867
Epoch 5/3000
700/700 [==============================] - 0s 93us/step - loss: 2.1177 - a
ccuracy: 0.1900 - val_loss: 2.1141 - val_accuracy: 0.1867
Epoch 6/3000
700/700 [==============================] - 0s 93us/step - loss: 2.0940 - a
ccuracy: 0.2029 - val_loss: 2.0931 - val_accuracy: 0.2033
Epoch 7/3000
700/700 [==============================] - 0s 91us/step - loss: 2.0721 - a
ccuracy: 0.2071 - val_loss: 2.0727 - val_accuracy: 0.2067
Epoch 8/3000
700/700 [==============================] - 0s 91us/step - loss: 2.0520 - a
```

```python
import matplotlib.pyplot as plt
```



```
figs, loss_ax=plt.subplots()
acc_ax=loss_ax.twinx()
loss_ax.plot(hist.history['loss'],'y',label='train loss')
loss_ax.plot(hist.history['val_loss'],'r',label='val loss')

acc_ax.plot(hist.history['accuracy'],'b',label='train acc')
acc_ax.plot(hist.history['val_accuracy'],'g',label='val acc')

acc_ax.set_ylabel('accuracy')

loss_ax.legend(loc='upper left')
acc_ax.legend(loc='upper right')
loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
plt.show()
```





```
res=model.evaluate(xTest, yTest, batch_size=32)
print("cost:"+str(res[0]))
print("accuracy:"+str(res[1]))
```

```
10000/10000 [==============================] - 0s 14us/step
cost:3.706244239425659
accuracy:0.2597000002861023
```

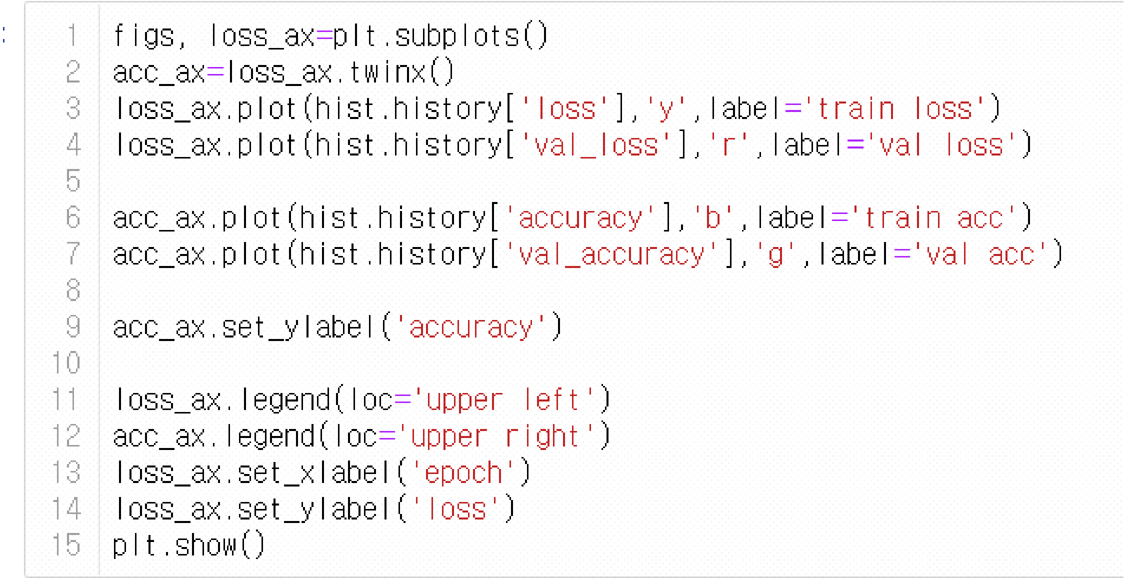
```
# 조기 종료 : earlystopping
# 콜백(함수) : 어떤 상황이 되었을때(val loss가 떨어지다가 올라가는 시점)
#함수 내에서 또 다른 어떤 함수를 호출하는 것
```

```
from keras.callbacks import EarlyStopping
```

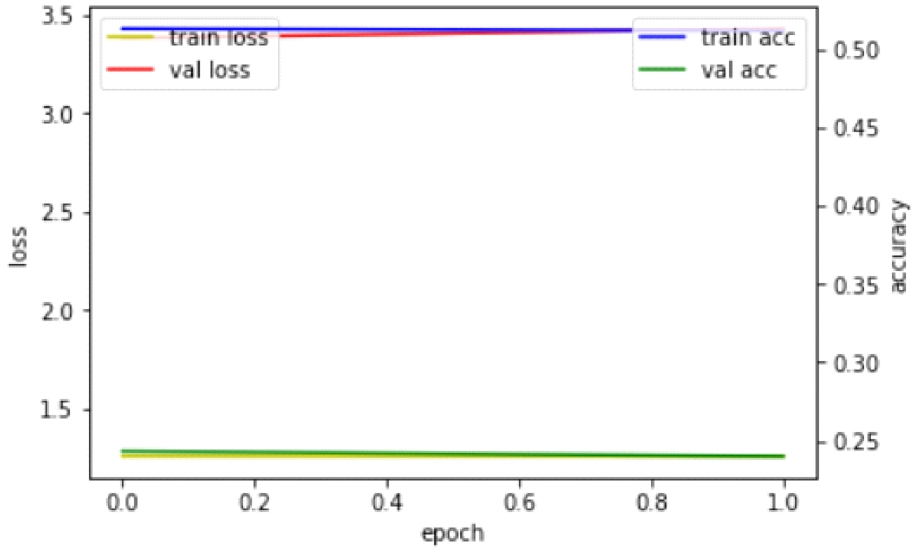
```
es=EarlyStopping()
```

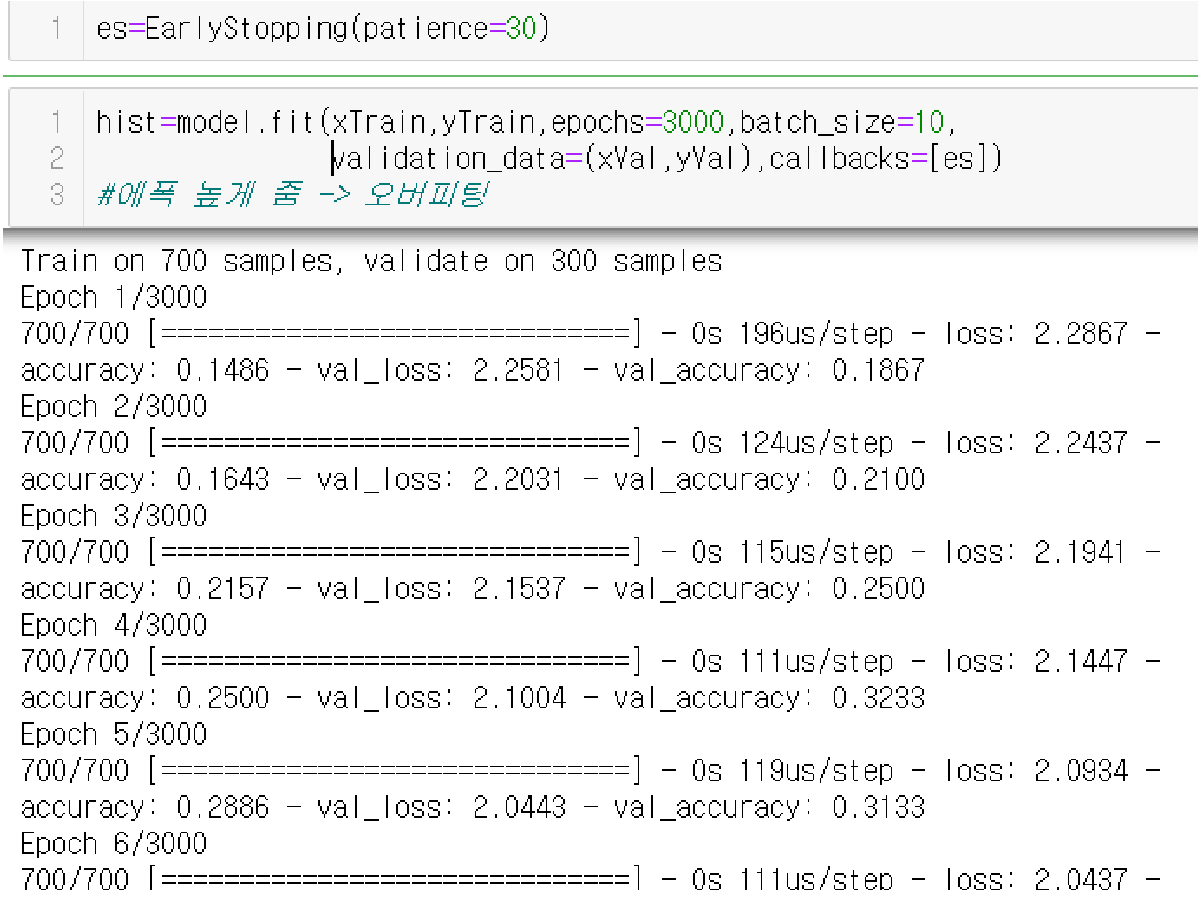
```
hist=model.fit(xTrain,yTrain,epochs=3000,batch_size=10,
               validation_data=(xVal,yVal),callbacks=[es])
#에폭 높게 줌 -> 오버피팅
```

```
Train on 700 samples, validate on 300 samples
Epoch 1/3000
700/700 [==============================] - 0s 120us/step - loss: 1.2591 - ac
curacy: 0.5129 - val_loss: 3.3829 - val_accuracy: 0.2433
Epoch 2/3000
700/700 [==============================] - 0s 109us/step - loss: 1.2582 - ac
curacy: 0.5114 - val_loss: 3.4305 - val_accuracy: 0.2400
```



```python
figs, loss_ax=plt.subplots()
acc_ax=loss_ax.twinx()
loss_ax.plot(hist.history['loss'],'y',label='train loss')
loss_ax.plot(hist.history['val_loss'],'r',label='val loss')

acc_ax.plot(hist.history['accuracy'],'b',label='train acc')
acc_ax.plot(hist.history['val_accuracy'],'g',label='val acc')

acc_ax.set_ylabel('accuracy')

loss_ax.legend(loc='upper left')
acc_ax.legend(loc='upper right')
loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
plt.show()
```

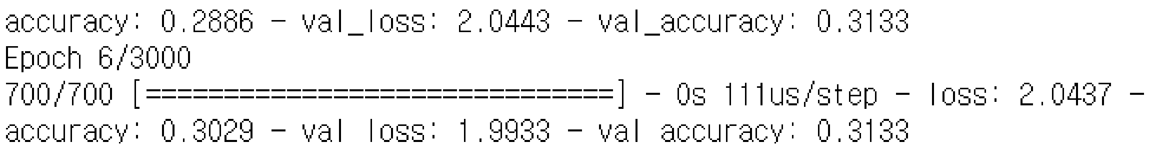




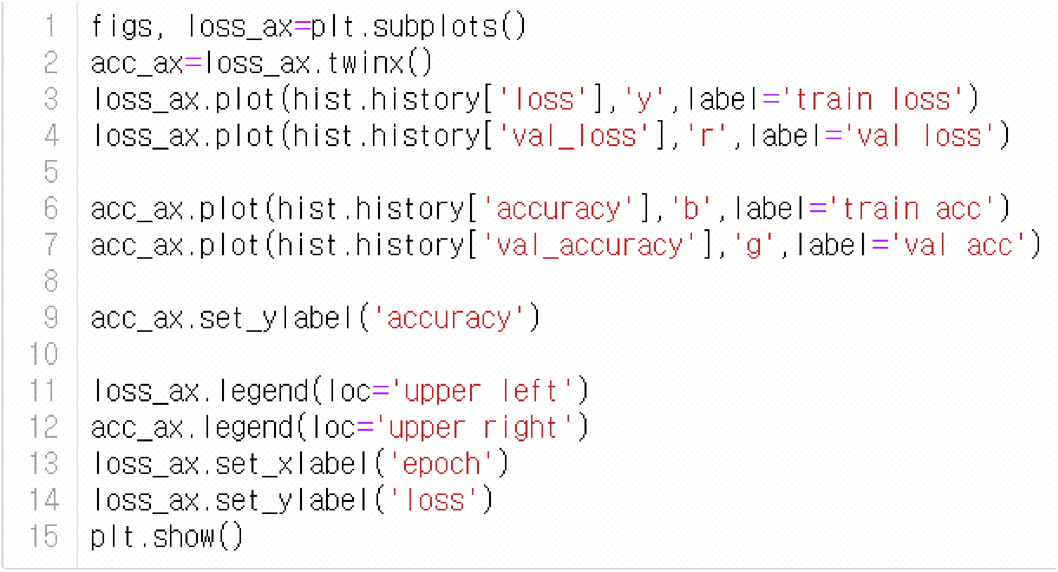
```python
es=EarlyStopping(patience=30)
```

```python
hist=model.fit(xTrain,yTrain,epochs=3000,batch_size=10,
               validation_data=(xVal,yVal),callbacks=[es])
#에폭 높게 줌 -> 오버피팅
```

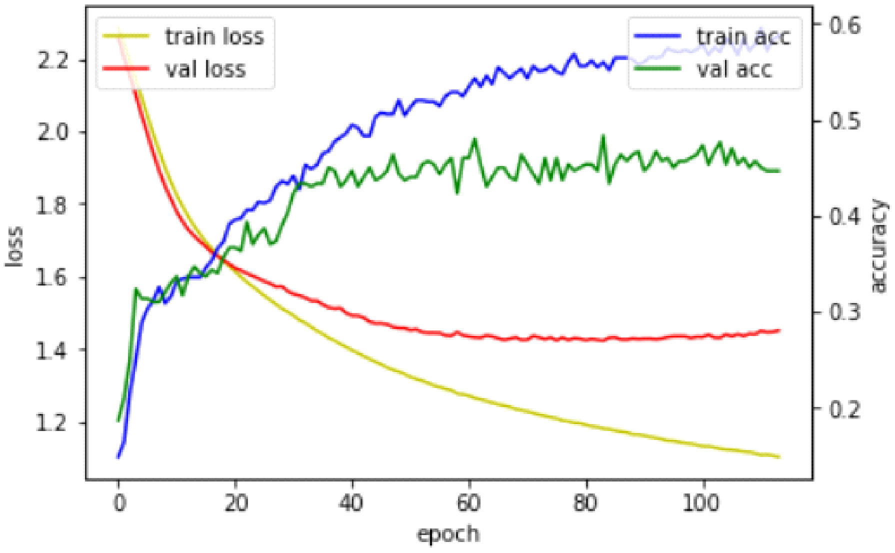
```
Train on 700 samples, validate on 300 samples
Epoch 1/3000
700/700 [==============================] - 0s 196us/step - loss: 2.2867 -
accuracy: 0.1486 - val_loss: 2.2581 - val_accuracy: 0.1867
Epoch 2/3000
700/700 [==============================] - 0s 124us/step - loss: 2.2437 -
accuracy: 0.1643 - val_loss: 2.2031 - val_accuracy: 0.2100
Epoch 3/3000
700/700 [==============================] - 0s 115us/step - loss: 2.1941 -
accuracy: 0.2157 - val_loss: 2.1537 - val_accuracy: 0.2500
Epoch 4/3000
700/700 [==============================] - 0s 111us/step - loss: 2.1447 -
accuracy: 0.2500 - val_loss: 2.1004 - val_accuracy: 0.3233
Epoch 5/3000
700/700 [==============================] - 0s 119us/step - loss: 2.0934 -
accuracy: 0.2886 - val_loss: 2.0443 - val_accuracy: 0.3133
Epoch 6/3000
700/700 [==============================] - 0s 111us/step - loss: 2.0437 -
```

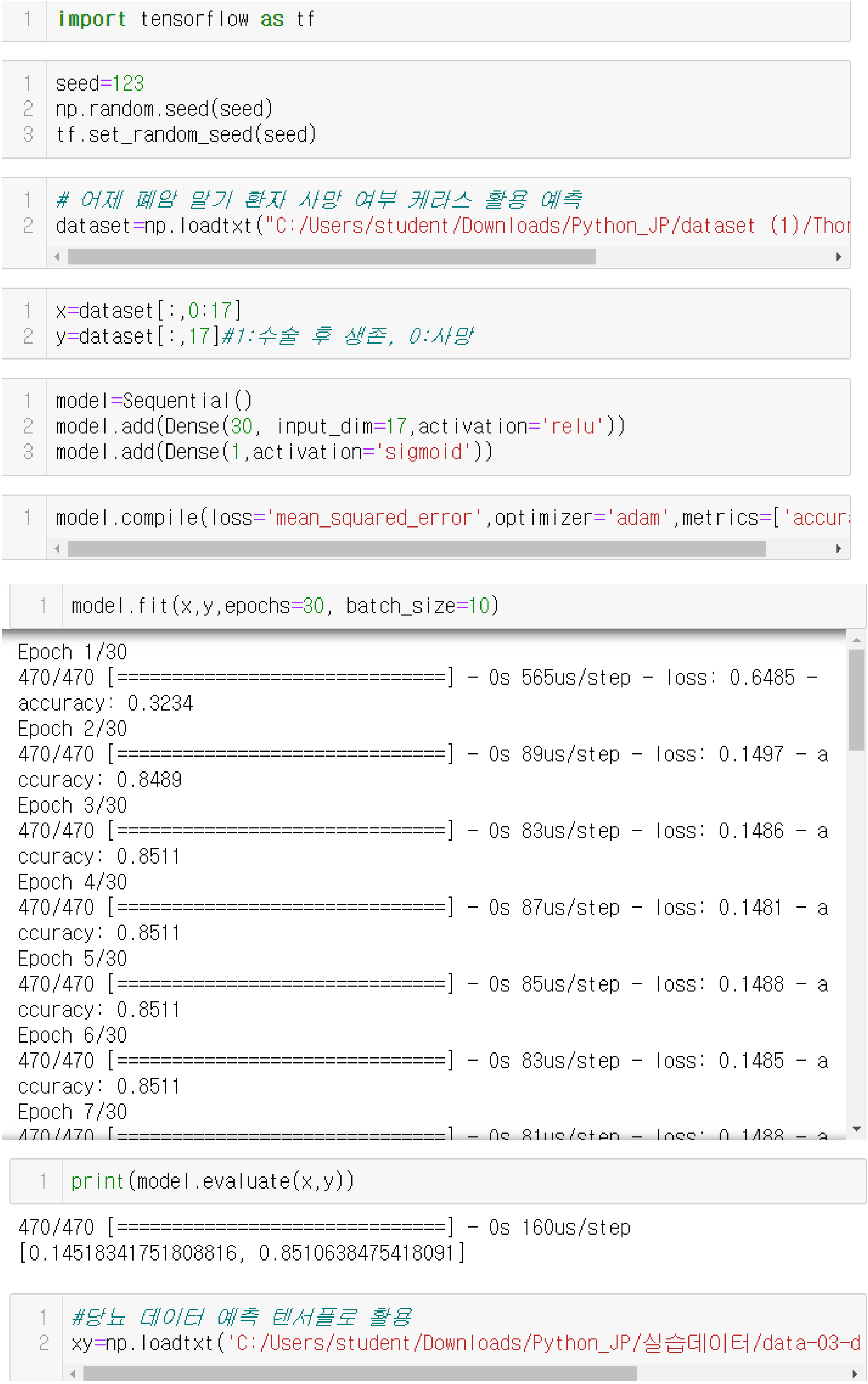


```
accuracy: 0.2886 - val_loss: 2.0443 - val_accuracy: 0.3133
Epoch 6/3000
700/700 [==============================] - 0s 111us/step - loss: 2.0437 -
accuracy: 0.3029 - val loss: 1.9933 - val accuracy: 0.3133
```



```python
figs, loss_ax=plt.subplots()
acc_ax=loss_ax.twinx()
loss_ax.plot(hist.history['loss'],'y',label='train loss')
loss_ax.plot(hist.history['val_loss'],'r',label='val loss')

acc_ax.plot(hist.history['accuracy'],'b',label='train acc')
acc_ax.plot(hist.history['val_accuracy'],'g',label='val acc')

acc_ax.set_ylabel('accuracy')

loss_ax.legend(loc='upper left')
acc_ax.legend(loc='upper right')
loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
plt.show()
```





```
import tensorflow as tf
```

```
seed=123
np.random.seed(seed)
tf.set_random_seed(seed)
```

```
# 어제 폐암 말기 환자 사망 여부 케라스 활용 예측
dataset=np.loadtxt("C:/Users/student/Downloads/Python_JP/dataset (1)/Thor
```

```
x=dataset[:,0:17]
y=dataset[:,17]#1:수술 후 생존, 0:사망
```

```
model=Sequential()
model.add(Dense(30, input_dim=17,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
```

```
model.compile(loss='mean_squared_error',optimizer='adam',metrics=['accur
```

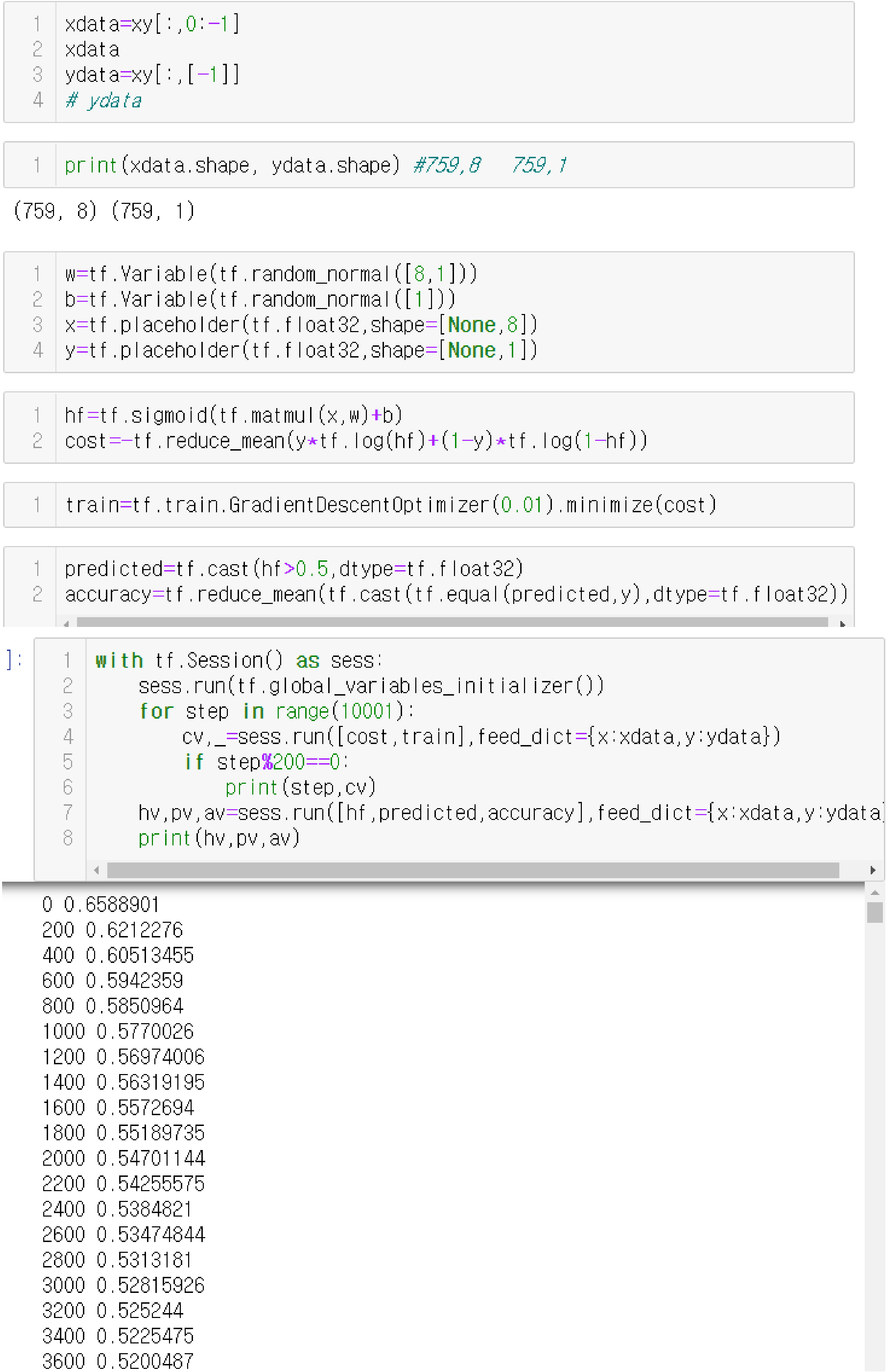
```
model.fit(x,y,epochs=30, batch_size=10)
```

```
Epoch 1/30
470/470 [==============================] - 0s 565us/step - loss: 0.6485 -
accuracy: 0.3234
Epoch 2/30
470/470 [==============================] - 0s 89us/step - loss: 0.1497 - a
ccuracy: 0.8489
Epoch 3/30
470/470 [==============================] - 0s 83us/step - loss: 0.1486 - a
ccuracy: 0.8511
Epoch 4/30
470/470 [==============================] - 0s 87us/step - loss: 0.1481 - a
ccuracy: 0.8511
Epoch 5/30
470/470 [==============================] - 0s 85us/step - loss: 0.1488 - a
ccuracy: 0.8511
Epoch 6/30
470/470 [==============================] - 0s 83us/step - loss: 0.1485 - a
ccuracy: 0.8511
Epoch 7/30
470/470 [==============================] - 0s 81us/step - loss: 0.1488 - a
```

```
print(model.evaluate(x,y))
```

```
470/470 [==============================] - 0s 160us/step
[0.14518341751808816, 0.8510638475418091]
```

```
#당뇨 데이터 예측 텐서플로 활용
xy=np.loadtxt('C:/Users/student/Downloads/Python_JP/실습데이터/data-03-d
```



```
xdata=xy[:,0:-1]
xdata
ydata=xy[:,[-1]]
# ydata
```

```
print(xdata.shape, ydata.shape) #759,8   759,1
```

```
(759, 8) (759, 1)
```

```
w=tf.Variable(tf.random_normal([8,1]))
b=tf.Variable(tf.random_normal([1]))
x=tf.placeholder(tf.float32,shape=[None,8])
y=tf.placeholder(tf.float32,shape=[None,1])
```

```
hf=tf.sigmoid(tf.matmul(x,w)+b)
cost=-tf.reduce_mean(y*tf.log(hf)+(1-y)*tf.log(1-hf))
```

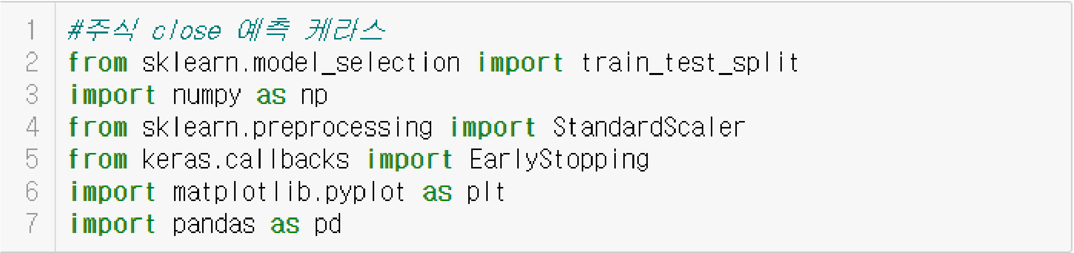
```
train=tf.train.GradientDescentOptimizer(0.01).minimize(cost)
```

```
predicted=tf.cast(hf>0.5,dtype=tf.float32)
accuracy=tf.reduce_mean(tf.cast(tf.equal(predicted,y),dtype=tf.float32))
```

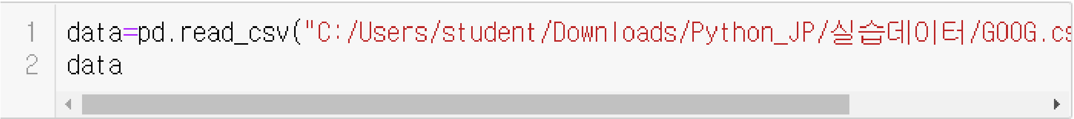
]:

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for step in range(10001):
        cv,_=sess.run([cost,train],feed_dict={x:xdata,y:ydata})
        if step%200==0:
            print(step,cv)
    hv,pv,av=sess.run([hf,predicted,accuracy],feed_dict={x:xdata,y:ydata
    print(hv,pv,av)
```

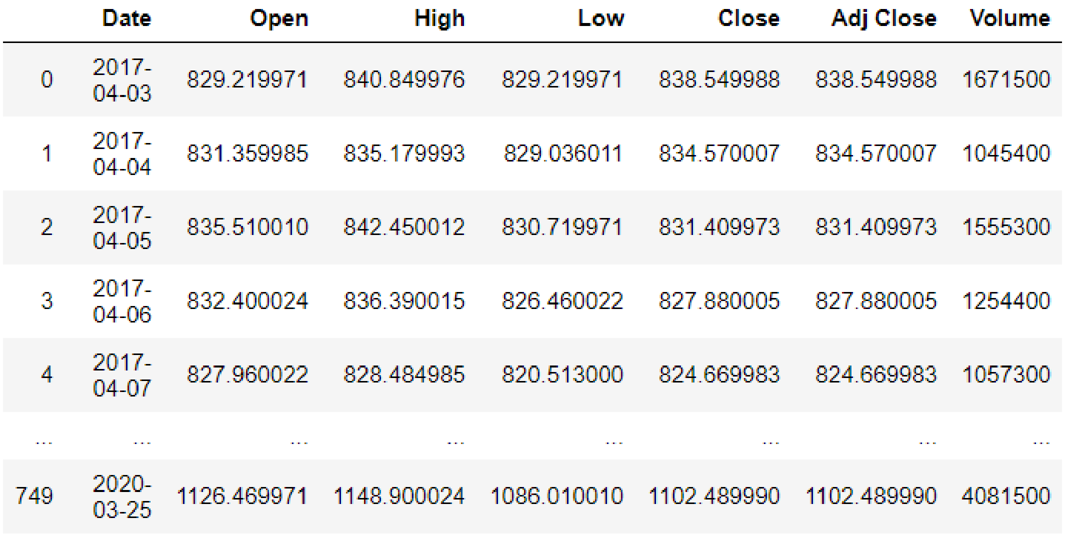
```
0 0.6588901
200 0.6212276
400 0.60513455
600 0.5942359
800 0.5850964
1000 0.5770026
1200 0.56974006
1400 0.56319195
1600 0.5572694
1800 0.55189735
2000 0.54701144
2200 0.54255575
2400 0.5384821
2600 0.53474844
2800 0.5313181
3000 0.52815926
3200 0.525244
3400 0.5225475
3600 0.5200487
```



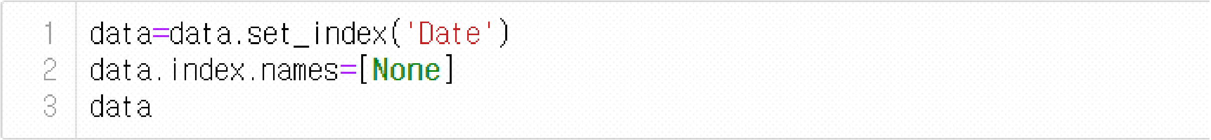
```python
#주식 close 예측 케라스
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import StandardScaler
from keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import pandas as pd
```



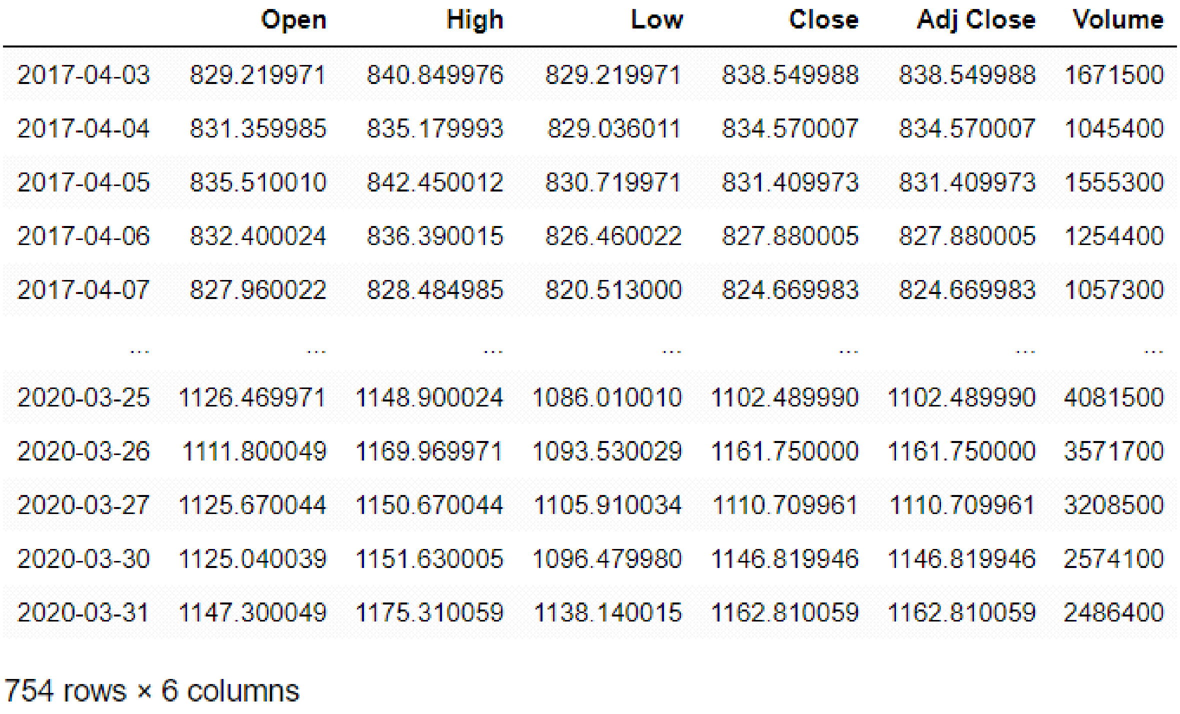
```python
data=pd.read_csv("C:/Users/student/Downloads/Python_JP/실습데이터/GOOG.cs
data
```



	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-04-03	829.219971	840.849976	829.219971	838.549988	838.549988	1671500
1	2017-04-04	831.359985	835.179993	829.036011	834.570007	834.570007	1045400
2	2017-04-05	835.510010	842.450012	830.719971	831.409973	831.409973	1555300
3	2017-04-06	832.400024	836.390015	826.460022	827.880005	827.880005	1254400
4	2017-04-07	827.960022	828.484985	820.513000	824.669983	824.669983	1057300
...	...	...	...	...	...	...	...
749	2020-03-25	1126.469971	1148.900024	1086.010010	1102.489990	1102.489990	4081500

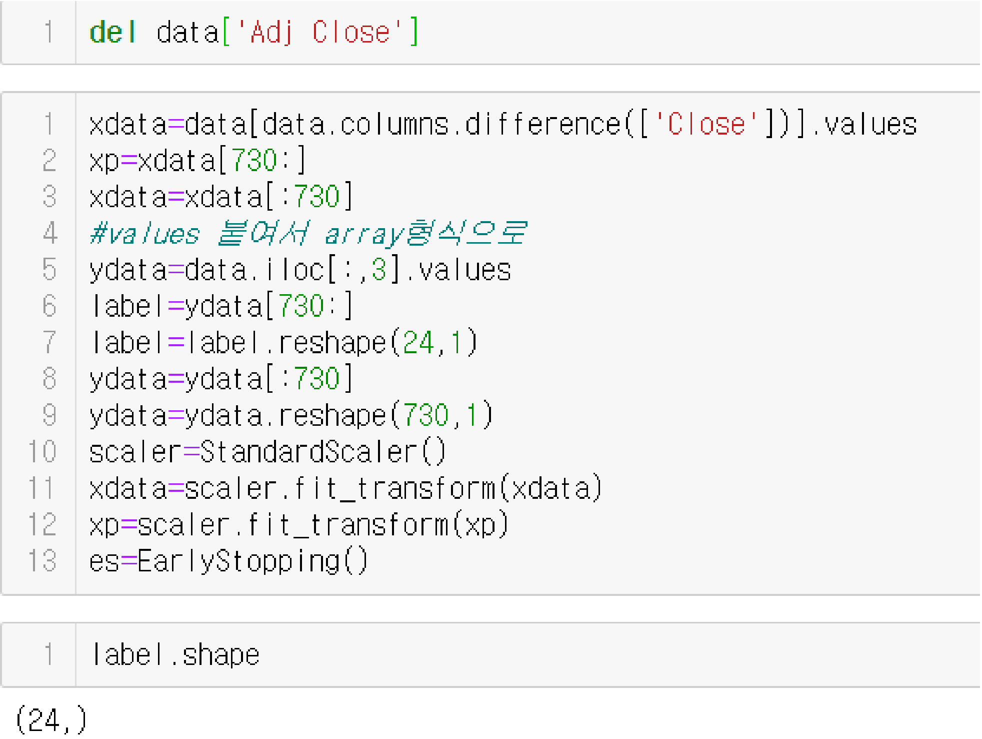


```python
data=data.set_index('Date')
data.index.names=[None]
data
```



	Open	High	Low	Close	Adj Close	Volume
2017-04-03	829.219971	840.849976	829.219971	838.549988	838.549988	1671500
2017-04-04	831.359985	835.179993	829.036011	834.570007	834.570007	1045400
2017-04-05	835.510010	842.450012	830.719971	831.409973	831.409973	1555300
2017-04-06	832.400024	836.390015	826.460022	827.880005	827.880005	1254400
2017-04-07	827.960022	828.484985	820.513000	824.669983	824.669983	1057300
...	...	...	...	...	...	...
2020-03-25	1126.469971	1148.900024	1086.010010	1102.489990	1102.489990	4081500
2020-03-26	1111.800049	1169.969971	1093.530029	1161.750000	1161.750000	3571700
2020-03-27	1125.670044	1150.670044	1105.910034	1110.709961	1110.709961	3208500
2020-03-30	1125.040039	1151.630005	1096.479980	1146.819946	1146.819946	2574100
2020-03-31	1147.300049	1175.310059	1138.140015	1162.810059	1162.810059	2486400

754 rows × 6 columns

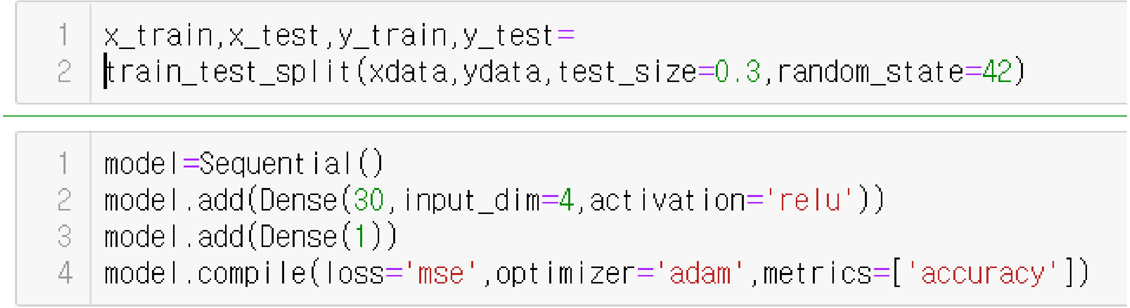


```
del data['Adj Close']
```

```
xdata=data[data.columns.difference(['Close'])].values
xp=xdata[730:]
xdata=xdata[:730]
#values 붙여서 array형식으로
ydata=data.iloc[:,3].values
label=ydata[730:]
label=label.reshape(24,1)
ydata=ydata[:730]
ydata=ydata.reshape(730,1)
scaler=StandardScaler()
xdata=scaler.fit_transform(xdata)
xp=scaler.fit_transform(xp)
es=EarlyStopping()
```

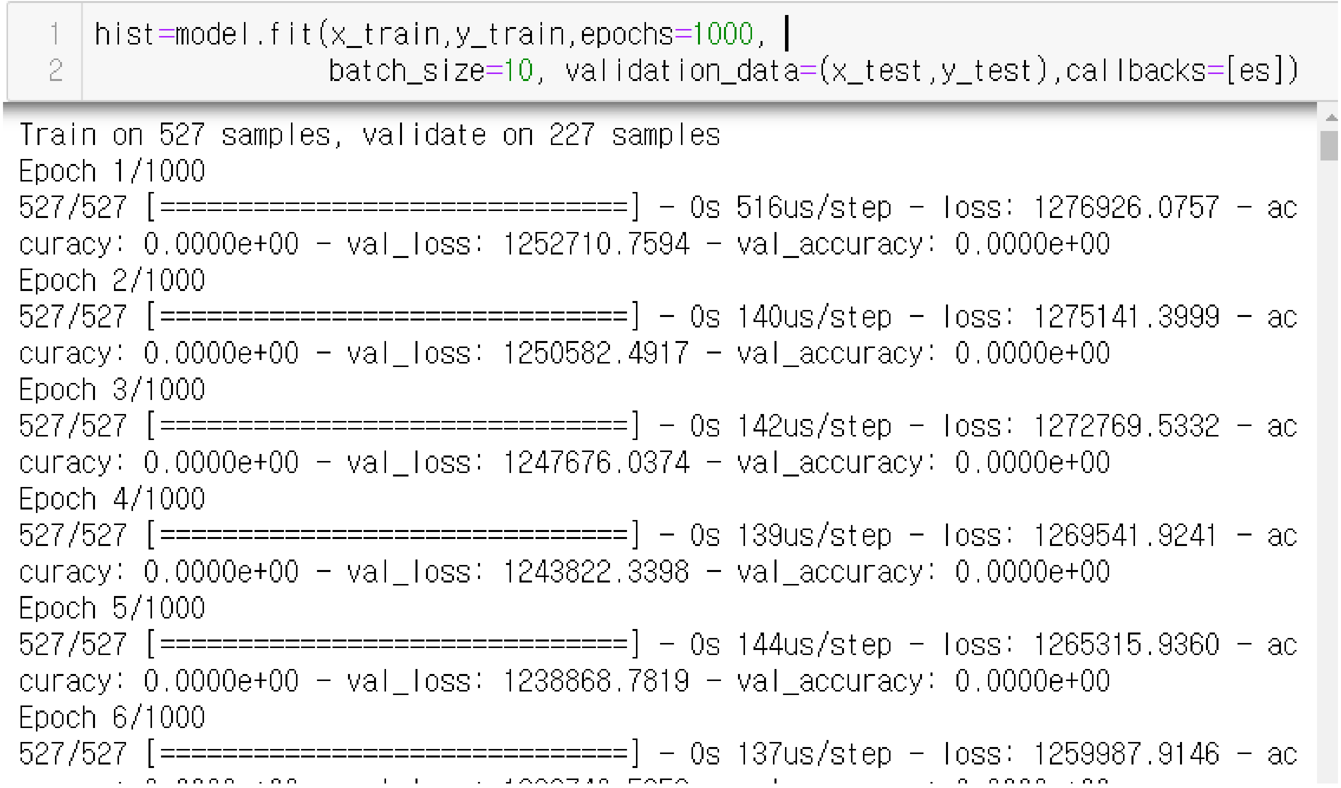
```
label.shape
```

```
(24,)
```



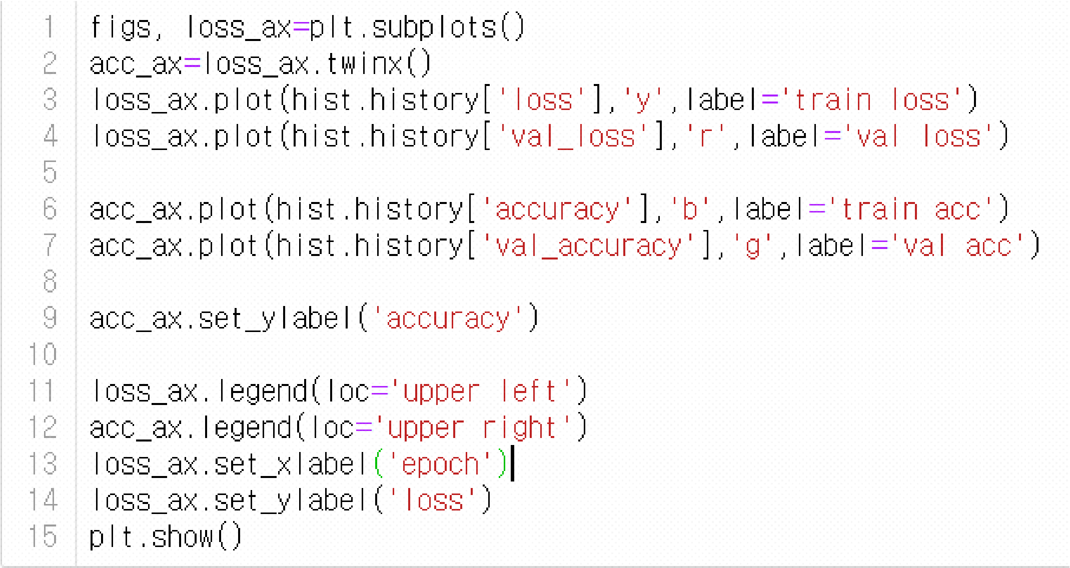
```
x_train,x_test,y_train,y_test=
train_test_split(xdata,ydata,test_size=0.3,random_state=42)
```

```
model=Sequential()
model.add(Dense(30,input_dim=4,activation='relu'))
model.add(Dense(1))
model.compile(loss='mse',optimizer='adam',metrics=['accuracy'])
```

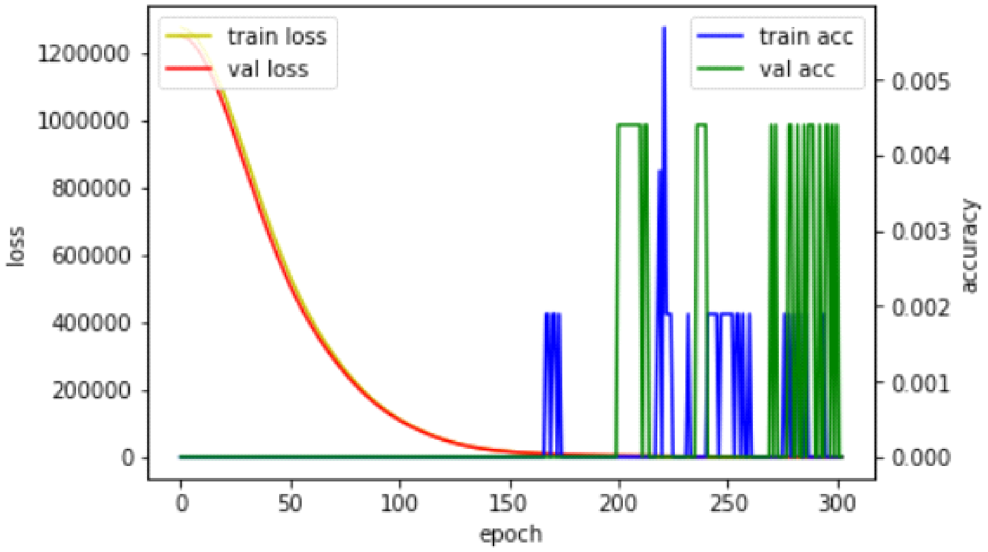


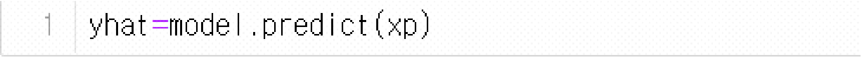
```
hist=model.fit(x_train,y_train,epochs=1000,
               batch_size=10, validation_data=(x_test,y_test),callbacks=[es])
```

```
Train on 527 samples, validate on 227 samples
Epoch 1/1000
527/527 [==============================] - 0s 516us/step - loss: 1276926.0757 - ac
curacy: 0.0000e+00 - val_loss: 1252710.7594 - val_accuracy: 0.0000e+00
Epoch 2/1000
527/527 [==============================] - 0s 140us/step - loss: 1275141.3999 - ac
curacy: 0.0000e+00 - val_loss: 1250582.4917 - val_accuracy: 0.0000e+00
Epoch 3/1000
527/527 [==============================] - 0s 142us/step - loss: 1272769.5332 - ac
curacy: 0.0000e+00 - val_loss: 1247676.0374 - val_accuracy: 0.0000e+00
Epoch 4/1000
527/527 [==============================] - 0s 139us/step - loss: 1269541.9241 - ac
curacy: 0.0000e+00 - val_loss: 1243822.3398 - val_accuracy: 0.0000e+00
Epoch 5/1000
527/527 [==============================] - 0s 144us/step - loss: 1265315.9360 - ac
curacy: 0.0000e+00 - val_loss: 1238868.7819 - val_accuracy: 0.0000e+00
Epoch 6/1000
527/527 [==============================] - 0s 137us/step - loss: 1259987.9146 - ac
```

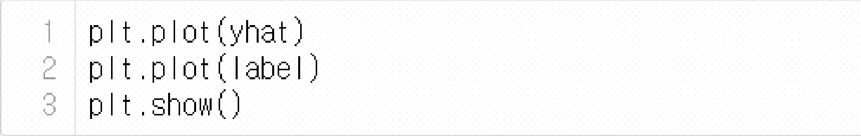


```
figs, loss_ax=plt.subplots()
acc_ax=loss_ax.twinx()
loss_ax.plot(hist.history['loss'],'y',label='train loss')
loss_ax.plot(hist.history['val_loss'],'r',label='val loss')

acc_ax.plot(hist.history['accuracy'],'b',label='train acc')
acc_ax.plot(hist.history['val_accuracy'],'g',label='val acc')

acc_ax.set_ylabel('accuracy')

loss_ax.legend(loc='upper left')
acc_ax.legend(loc='upper right')
loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
plt.show()
```

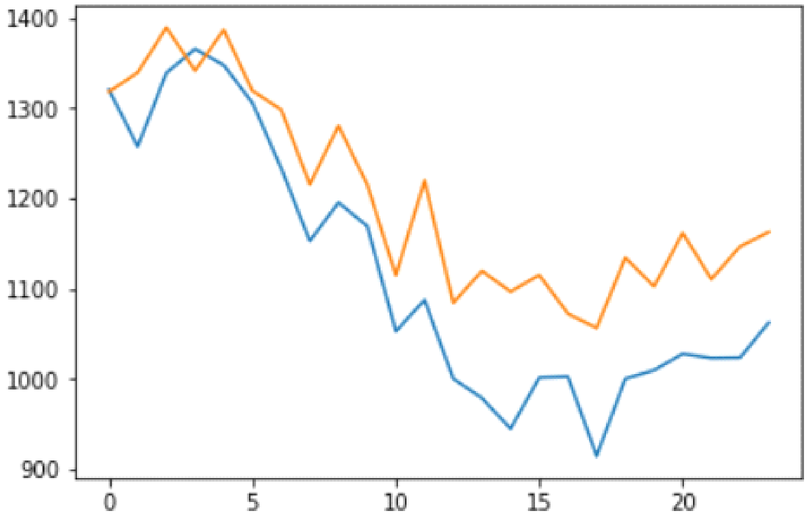




```
yhat=model.predict(xp)
```



```
plt.plot(yhat)
plt.plot(label)
plt.show()
```



1400
1300
1200
1100
1000
900
0
5
10
15
20