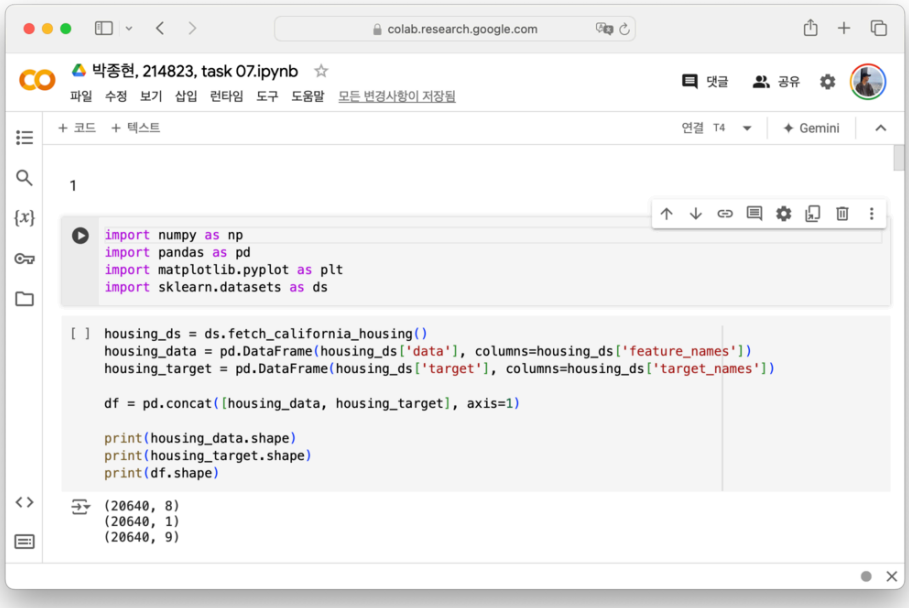
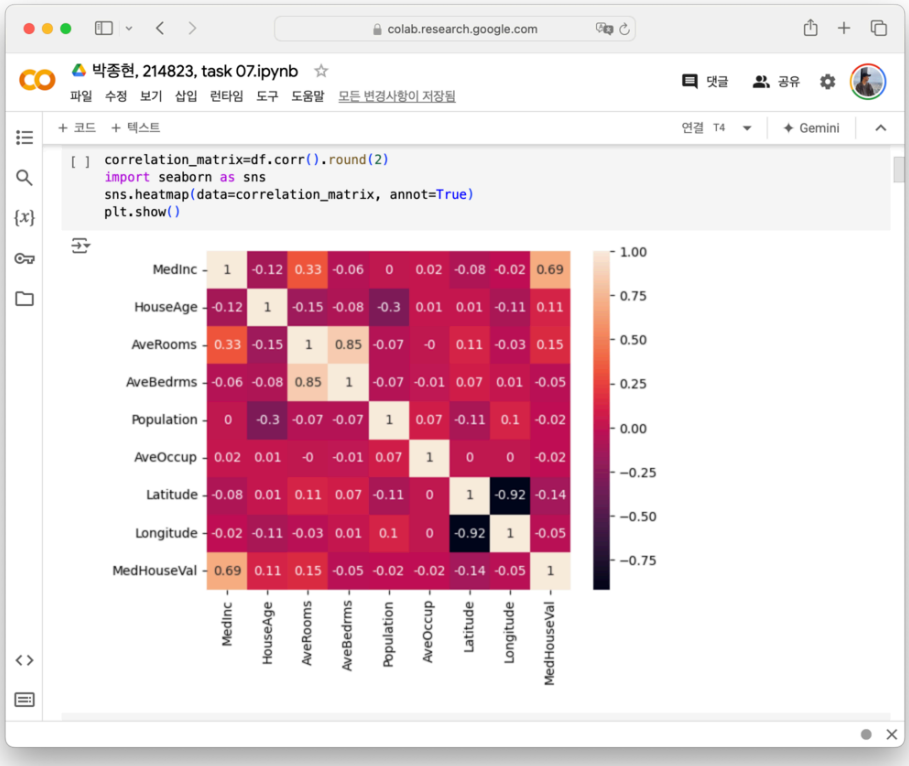
Homework 07

이름: 박종현
학번: 214823

[1] 14-1. 8 16 쪽





colab.research.google.com

박종현, 214823, task 07.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

+ 코드 + 텍스트 연결 T4 Gemini

```
[ ] from sklearn.model_selection import train_test_split
    x_train, x_test, y_train, y_test = train_test_split(housing_data['MedInc'], housing_target['MedHouseV
    print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)

    from sklearn.linear_model import LinearRegression
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.metrics import mean_absolute_error
```

```
(14448,) (6192,) (14448,) (6192,)
```

```
[ ] reg_lr = LinearRegression()
    reg_lr.fit(np.array(x_train).reshape(-1, 1), y_train)
    y_pred_lr = reg_lr.predict(np.array(x_test).reshape(-1, 1))
    mean_absolute_error(y_test, y_pred_lr)
```

```
0.6257812642474934
```

```
[ ] reg_rfr = RandomForestRegressor()
    reg_rfr.fit(np.array(x_train).reshape(-1, 1), y_train)
    y_pred_rfr = reg_rfr.predict(np.array(x_test).reshape(-1, 1))
    mean_absolute_error(y_test, y_pred_rfr)
```

```
0.7190334908322912
```

colab.research.google.com

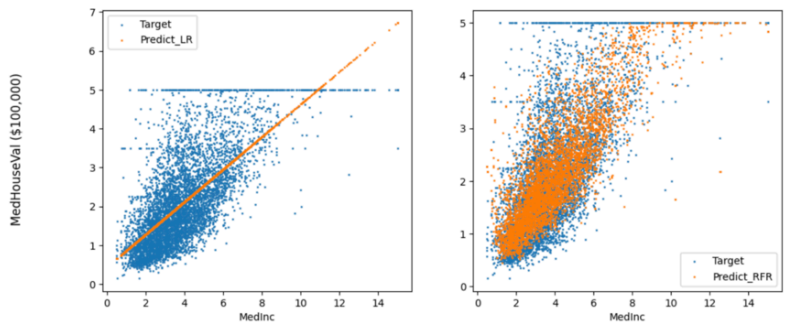
박종현, 214823, task 07.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

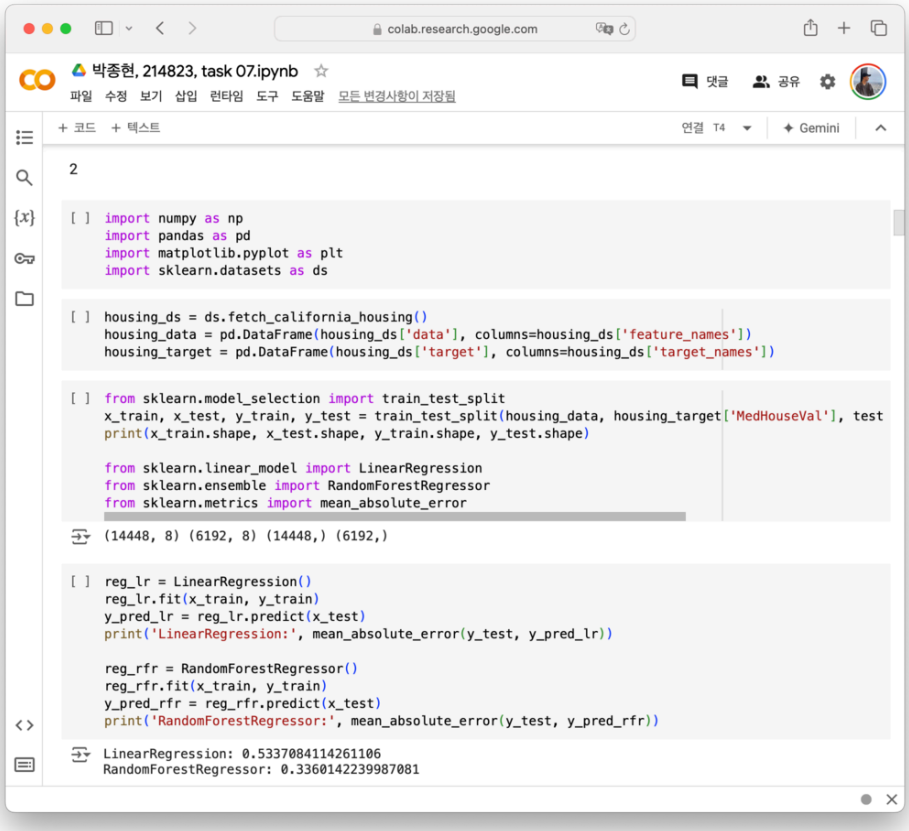
댓글 공유

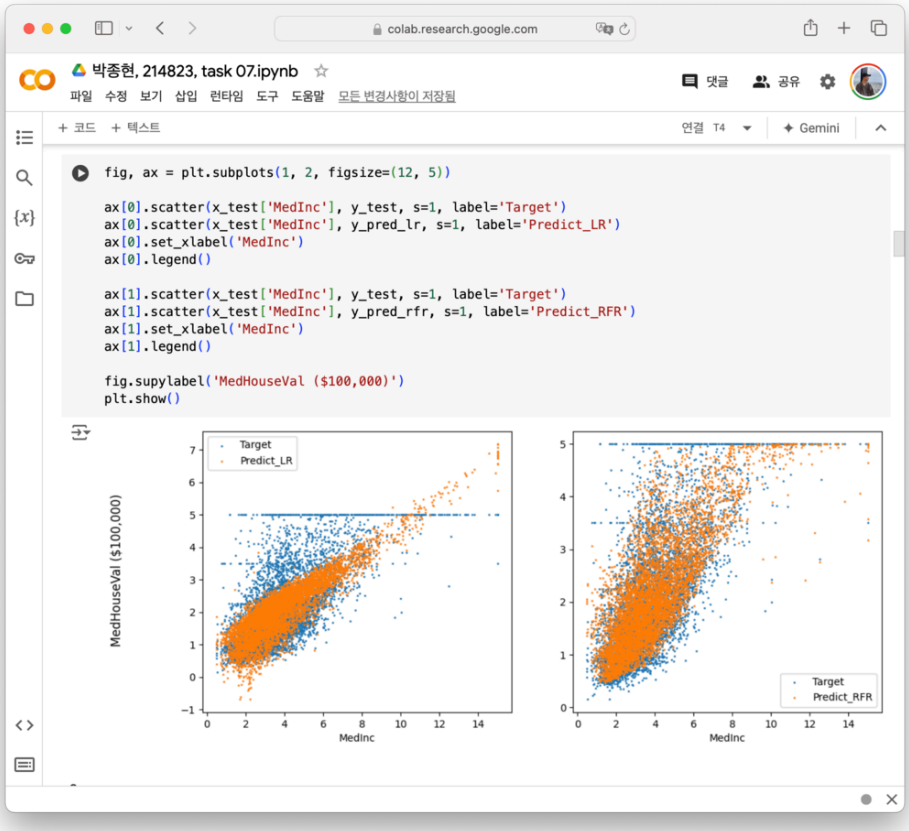
+ 코드 + 텍스트 연결 T4 Gemini

```
[ ] fig, ax = plt.subplots(1, 2, figsize=(12, 5))

    ax[0].scatter(x_test, y_test, s=1, label='Target')
    ax[0].scatter(x_test, y_pred_lr, s=1, label='Predict_LR')
    ax[0].set_xlabel('MedInc')
    ax[0].legend()

    ax[1].scatter(x_test, y_test, s=1, label='Target')
    ax[1].scatter(x_test, y_pred_rfr, s=1, label='Predict_RFR')
    ax[1].set_xlabel('MedInc')
    ax[1].legend()

    fig.supylabel('MedHouseVal ($100,000)')
    plt.show()
```

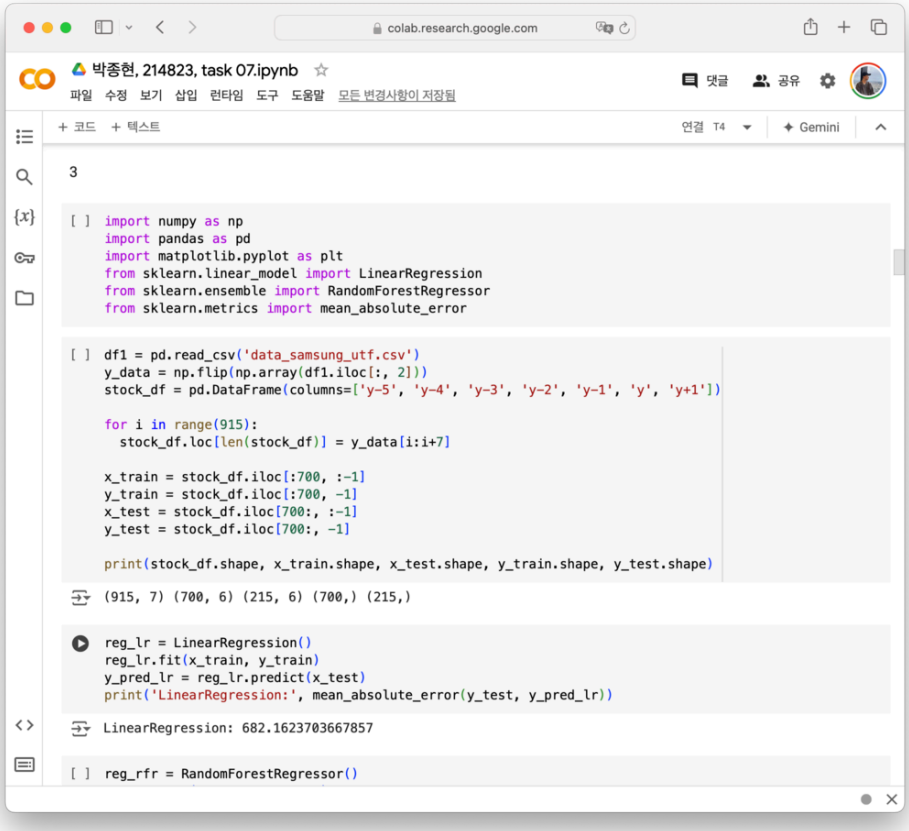


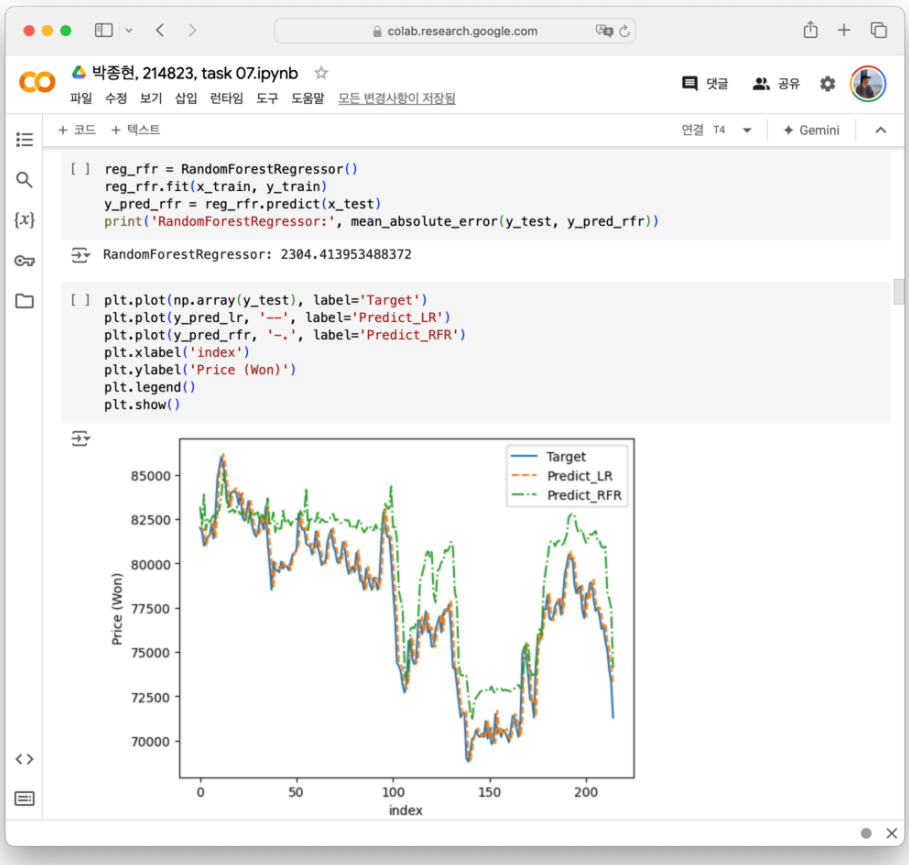
[2] 14-1. 17 18 쪽


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
2
[ ] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.datasets as ds
[ ] housing_ds = ds.fetch_california_housing()
housing_data = pd.DataFrame(housing_ds['data'], columns=housing_ds['feature_names'])
housing_target = pd.DataFrame(housing_ds['target'], columns=housing_ds['target_names'])
[ ] from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(housing_data, housing_target['MedHouseVal'], test
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
(14448, 8) (6192, 8) (14448,) (6192,)
[ ] reg_lr = LinearRegression()
reg_lr.fit(x_train, y_train)
y_pred_lr = reg_lr.predict(x_test)
print('LinearRegression:', mean_absolute_error(y_test, y_pred_lr))
reg_rfr = RandomForestRegressor()
reg_rfr.fit(x_train, y_train)
y_pred_rfr = reg_rfr.predict(x_test)
print('RandomForestRegressor:', mean_absolute_error(y_test, y_pred_rfr))
LinearRegression: 0.5337084114261106
RandomForestRegressor: 0.3360142239987081

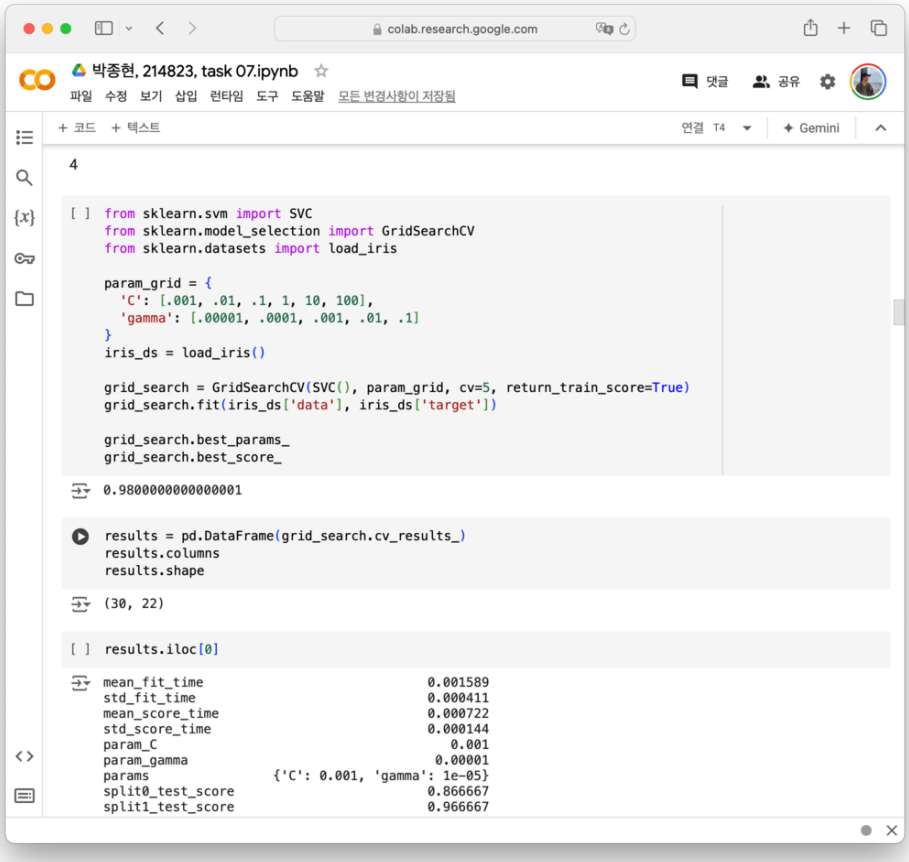

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
ax[0].scatter(x_test['MedInc'], y_test, s=1, label='Target')
ax[0].scatter(x_test['MedInc'], y_pred_lr, s=1, label='Predict_LR')
ax[0].set_xlabel('MedInc')
ax[0].legend()
ax[1].scatter(x_test['MedInc'], y_test, s=1, label='Target')
ax[1].scatter(x_test['MedInc'], y_pred_rfr, s=1, label='Predict_RFR')
ax[1].set_xlabel('MedInc')
ax[1].legend()
fig.supylabel('MedHouseVal ($100,000)')
plt.show()
Target
Predict_LR
MedHouseVal ($100,000)
MedInc
Target
Predict_RFR
MedInc

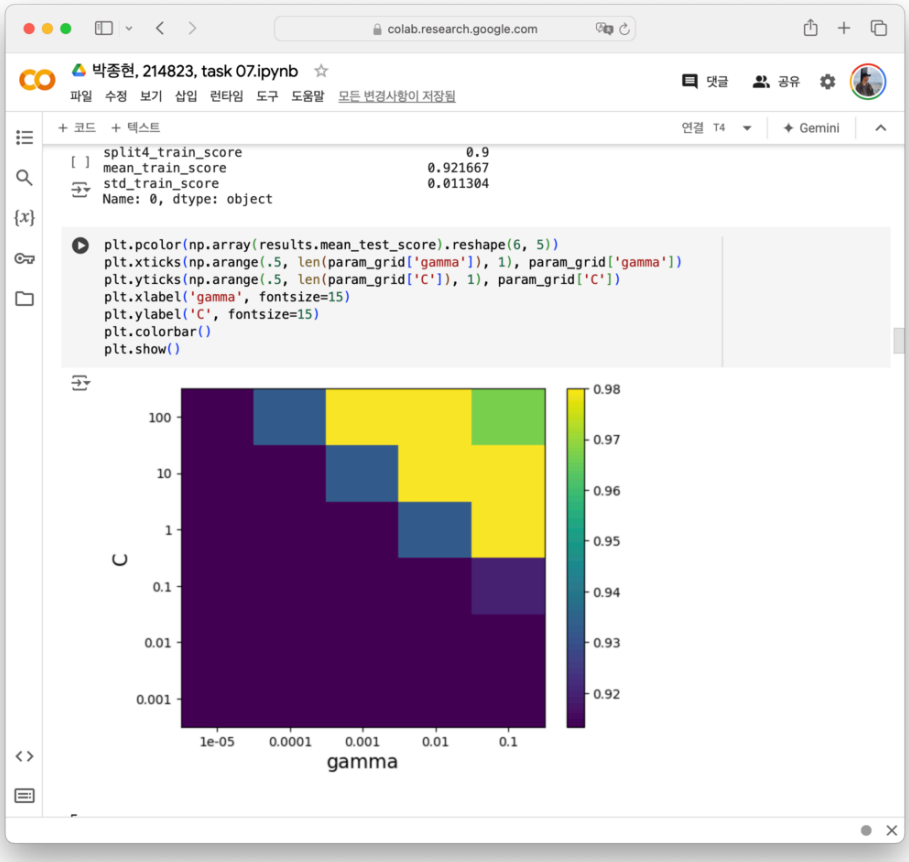
[3] 14-1. 23 25 쪽


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
3
[ ] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
[ ] df1 = pd.read_csv('data_samsung_utf.csv')
y_data = np.flip(np.array(df1.iloc[:, 2]))
stock_df = pd.DataFrame(columns=['y-5', 'y-4', 'y-3', 'y-2', 'y-1', 'y', 'y+1'])
for i in range(915):
stock_df.loc[len(stock_df)] = y_data[i:i+7]
x_train = stock_df.iloc[:700, :-1]
y_train = stock_df.iloc[:700, -1]
x_test = stock_df.iloc[700:, :-1]
y_test = stock_df.iloc[700:, -1]
print(stock_df.shape, x_train.shape, x_test.shape, y_train.shape, y_test.shape)
(915, 7) (700, 6) (215, 6) (700,) (215,)
reg_lr = LinearRegression()
reg_lr.fit(x_train, y_train)
y_pred_lr = reg_lr.predict(x_test)
print('LinearRegression:', mean_absolute_error(y_test, y_pred_lr))
LinearRegression: 682.1623703667857
[ ] reg_rfr = RandomForestRegressor()

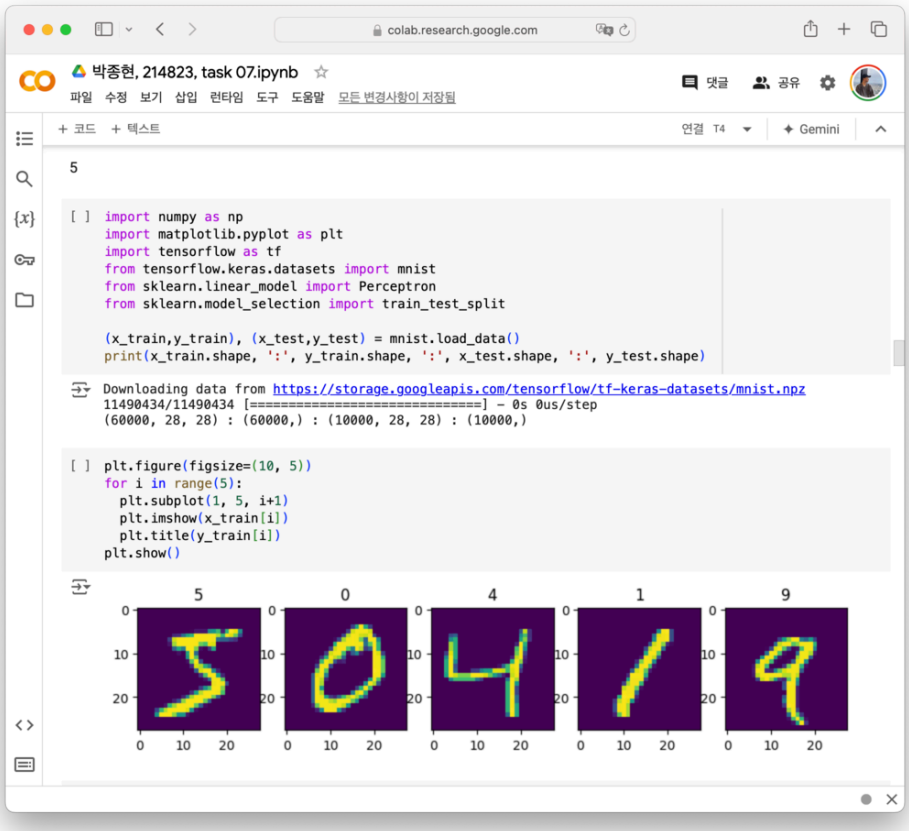

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] reg_rfr = RandomForestRegressor()
reg_rfr.fit(x_train, y_train)
y_pred_rfr = reg_rfr.predict(x_test)
print('RandomForestRegressor:', mean_absolute_error(y_test, y_pred_rfr))
RandomForestRegressor: 2304.413953488372
[ ] plt.plot(np.array(y_test), label='Target')
plt.plot(y_pred_lr, '--', label='Predict_LR')
plt.plot(y_pred_rfr, '-.', label='Predict_RFR')
plt.xlabel('index')
plt.ylabel('Price (Won)')
plt.legend()
plt.show()
Target
Predict_LR
Predict_RFR
Price (Won)
85000
82500
80000
77500
75000
72500
70000
0
50
100
150
200
index

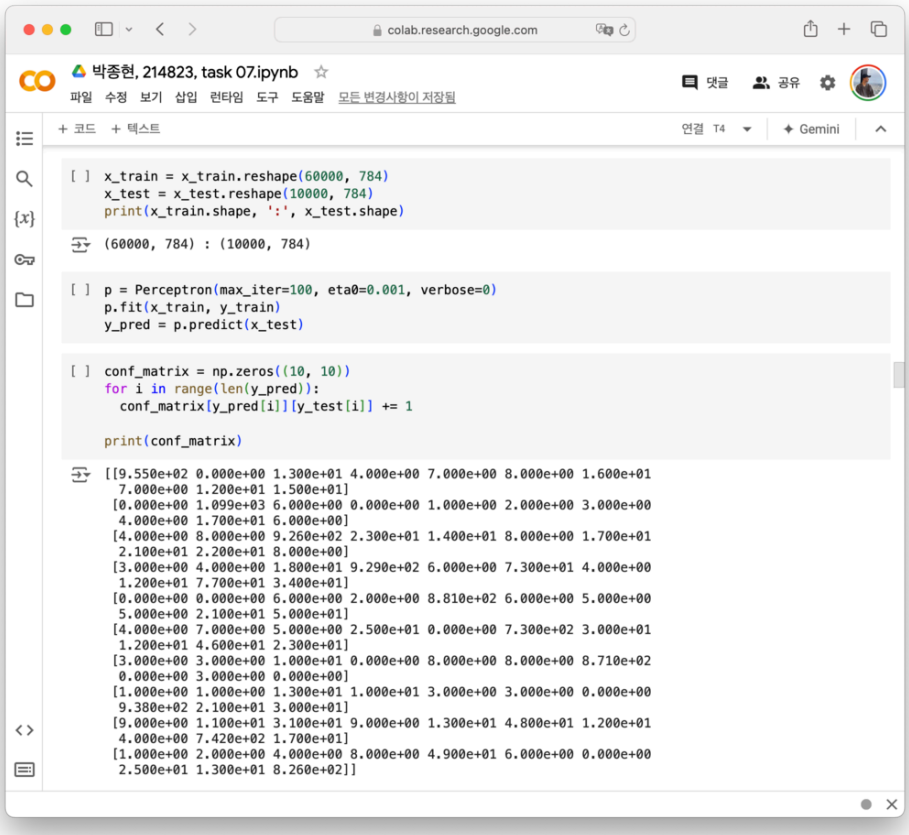
[4] 14-1. 32 35 쪽


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
4
[ ] from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import load_iris
param_grid = {
'C': [.001, .01, .1, 1, 10, 100],
'gamma': [.00001, .0001, .001, .01, .1]
}
iris_ds = load_iris()
grid_search = GridSearchCV(SVC(), param_grid, cv=5, return_train_score=True)
grid_search.fit(iris_ds['data'], iris_ds['target'])
grid_search.best_params_
grid_search.best_score_
0.9800000000000001
results = pd.DataFrame(grid_search.cv_results_)
results.columns
results.shape
(30, 22)
[ ] results.iloc[0]
mean_fit_time 0.001589
std_fit_time 0.000411
mean_score_time 0.000722
std_score_time 0.000144
param_C 0.001
param_gamma 0.00001
params {'C': 0.001, 'gamma': 1e-05}
split0_test_score 0.866667
split1_test_score 0.966667


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] split4_train_score 0.9
mean_train_score 0.921667
std_train_score 0.011304
Name: 0, dtype: object
plt.pcolor(np.array(results.mean_test_score).reshape(6, 5))
plt.xticks(np.arange(.5, len(param_grid['gamma']), 1), param_grid['gamma'])
plt.yticks(np.arange(.5, len(param_grid['C']), 1), param_grid['C'])
plt.xlabel('gamma', fontsize=15)
plt.ylabel('C', fontsize=15)
plt.colorbar()
plt.show()
100
10
1
0.1
0.01
0.001
C
1e-05
0.0001
0.001
0.01
0.1
gamma
0.98
0.97
0.96
0.95
0.94
0.93
0.92

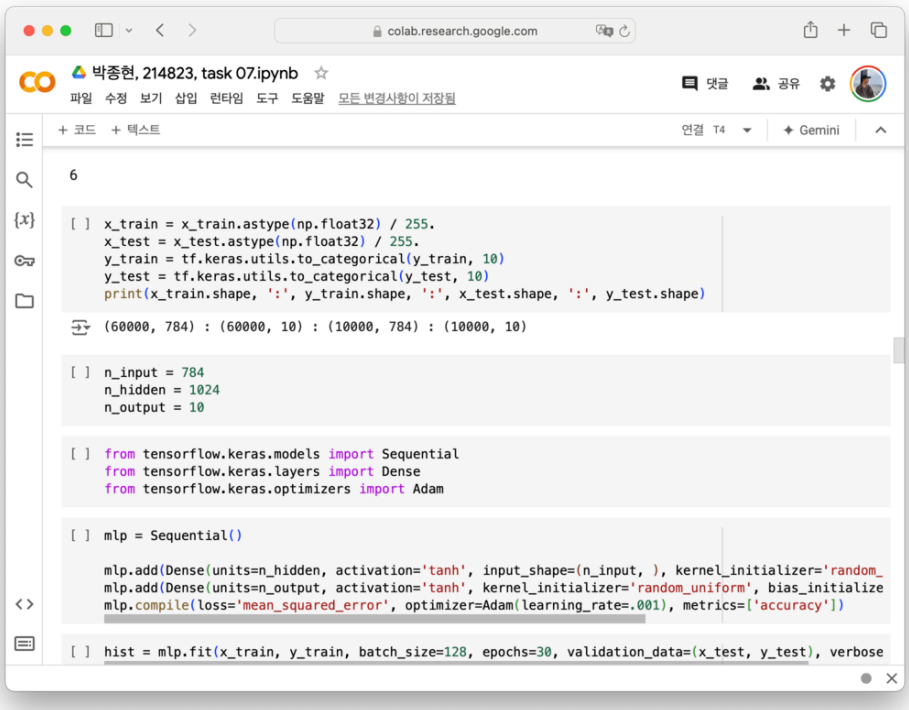
[5] 14-2. 7 11 쪽

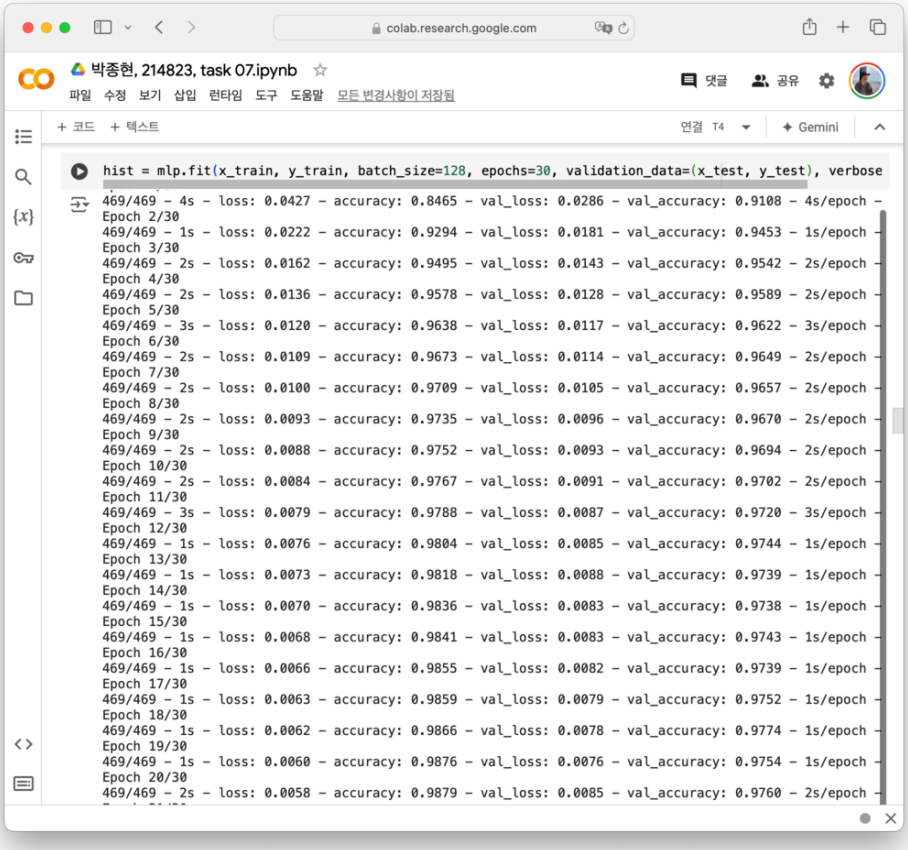

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
5
[ ] import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
(x_train,y_train), (x_test,y_test) = mnist.load_data()
print(x_train.shape, ':', y_train.shape, ':', x_test.shape, ':', y_test.shape)
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==========] - 0s 0us/step
(60000, 28, 28) : (60000,) : (10000, 28, 28) : (10000,)
[ ] plt.figure(figsize=(10, 5))
for i in range(5):
plt.subplot(1, 5, i+1)
plt.imshow(x_train[i])
plt.title(y_train[i])
plt.show()
5
0
4
1
9

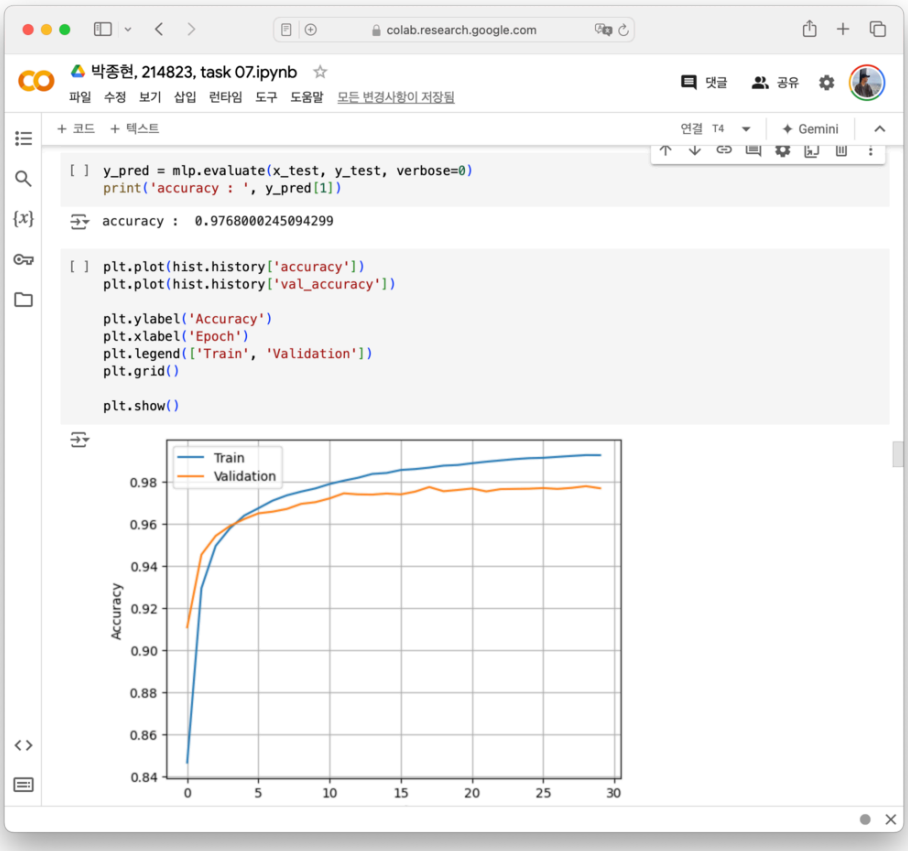

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
print(x_train.shape, ':', x_test.shape)
(60000, 784) : (10000, 784)
[ ] p = Perceptron(max_iter=100, eta0=0.001, verbose=0)
p.fit(x_train, y_train)
y_pred = p.predict(x_test)
[ ] conf_matrix = np.zeros((10, 10))
for i in range(len(y_pred)):
conf_matrix[y_pred[i]][y_test[i]] += 1
print(conf_matrix)
[[9.550e+02 0.000e+00 1.300e+01 4.000e+00 7.000e+00 8.000e+00 1.600e+01
7.000e+00 1.200e+01 1.500e+01]
[0.000e+00 1.099e+03 6.000e+00 0.000e+00 1.000e+00 2.000e+00 3.000e+00
4.000e+00 1.700e+01 6.000e+00]
[4.000e+00 8.000e+00 9.260e+02 2.300e+01 1.400e+01 8.000e+00 1.700e+01
2.100e+01 2.200e+01 8.000e+00]
[3.000e+00 4.000e+00 1.800e+01 9.290e+02 6.000e+00 7.300e+01 4.000e+00
1.200e+01 7.700e+01 3.400e+01]
[0.000e+00 0.000e+00 6.000e+00 2.000e+00 8.810e+02 6.000e+00 5.000e+00
5.000e+00 2.100e+01 5.000e+01]
[4.000e+00 7.000e+00 5.000e+00 2.500e+01 0.000e+00 7.300e+02 3.000e+01
1.200e+01 4.600e+01 2.300e+01]
[3.000e+00 3.000e+00 1.000e+01 0.000e+00 8.000e+00 8.000e+00 8.710e+02
0.000e+00 3.000e+00 0.000e+00]
[1.000e+00 1.000e+00 1.300e+01 1.000e+01 3.000e+00 3.000e+00 0.000e+00
9.380e+02 2.100e+01 3.000e+01]
[9.000e+00 1.100e+01 3.100e+01 9.000e+00 1.300e+01 4.800e+01 1.200e+01
4.000e+00 7.420e+02 1.700e+01]
[1.000e+00 2.000e+00 4.000e+00 8.000e+00 4.900e+01 6.000e+00 0.000e+00
2.500e+01 1.300e+01 8.260e+02]]


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[9.000e+00 1.100e+01 3.100e+01 9.000e+00 1.300e+01 4.800e+01 1.200e+01
4.000e+00 7.420e+02 1.700e+01]
[1.000e+00 2.000e+00 4.000e+00 8.000e+00 4.900e+01 6.000e+00 0.000e+00
2.500e+01 1.300e+01 8.260e+02]]
no_correct = 0
for i in range(10):
no_correct += conf_matrix[i][i]
accuracy = no_correct / len(y_pred)
print(accuracy)
0.8897

[6] 14-2. 22 28 쪽


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
6
x_train = x_train.astype(np.float32) / 255.
x_test = x_test.astype(np.float32) / 255.
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
print(x_train.shape, ':', y_train.shape, ':', x_test.shape, ':', y_test.shape)
(60000, 784) : (60000, 10) : (10000, 784) : (10000, 10)
n_input = 784
n_hidden = 1024
n_output = 10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
mlp = Sequential()
mlp.add(Dense(units=n_hidden, activation='tanh', input_shape=(n_input,), kernel_initializer='random_
mlp.add(Dense(units=n_output, activation='tanh', kernel_initializer='random_uniform', bias_initialize
mlp.compile(loss='mean_squared_error', optimizer=Adam(learning_rate=.001), metrics=['accuracy'])
hist = mlp.fit(x_train, y_train, batch_size=128, epochs=30, validation_data=(x_test, y_test), verbose


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
hist = mlp.fit(x_train, y_train, batch_size=128, epochs=30, validation_data=(x_test, y_test), verbose
469/469 - 4s - loss: 0.0427 - accuracy: 0.8465 - val_loss: 0.0286 - val_accuracy: 0.9108 - 4s/epoch -
Epoch 2/30
469/469 - 1s - loss: 0.0222 - accuracy: 0.9294 - val_loss: 0.0181 - val_accuracy: 0.9453 - 1s/epoch -
Epoch 3/30
469/469 - 2s - loss: 0.0162 - accuracy: 0.9495 - val_loss: 0.0143 - val_accuracy: 0.9542 - 2s/epoch -
Epoch 4/30
469/469 - 2s - loss: 0.0136 - accuracy: 0.9578 - val_loss: 0.0128 - val_accuracy: 0.9589 - 2s/epoch -
Epoch 5/30
469/469 - 3s - loss: 0.0120 - accuracy: 0.9638 - val_loss: 0.0117 - val_accuracy: 0.9622 - 3s/epoch -
Epoch 6/30
469/469 - 2s - loss: 0.0109 - accuracy: 0.9673 - val_loss: 0.0114 - val_accuracy: 0.9649 - 2s/epoch -
Epoch 7/30
469/469 - 2s - loss: 0.0100 - accuracy: 0.9709 - val_loss: 0.0105 - val_accuracy: 0.9657 - 2s/epoch -
Epoch 8/30
469/469 - 2s - loss: 0.0093 - accuracy: 0.9735 - val_loss: 0.0096 - val_accuracy: 0.9670 - 2s/epoch -
Epoch 9/30
469/469 - 2s - loss: 0.0088 - accuracy: 0.9752 - val_loss: 0.0093 - val_accuracy: 0.9694 - 2s/epoch -
Epoch 10/30
469/469 - 2s - loss: 0.0084 - accuracy: 0.9767 - val_loss: 0.0091 - val_accuracy: 0.9702 - 2s/epoch -
Epoch 11/30
469/469 - 3s - loss: 0.0079 - accuracy: 0.9788 - val_loss: 0.0087 - val_accuracy: 0.9720 - 3s/epoch -
Epoch 12/30
469/469 - 1s - loss: 0.0076 - accuracy: 0.9804 - val_loss: 0.0085 - val_accuracy: 0.9744 - 1s/epoch -
Epoch 13/30
469/469 - 1s - loss: 0.0073 - accuracy: 0.9818 - val_loss: 0.0088 - val_accuracy: 0.9739 - 1s/epoch -
Epoch 14/30
469/469 - 1s - loss: 0.0070 - accuracy: 0.9836 - val_loss: 0.0083 - val_accuracy: 0.9738 - 1s/epoch -
Epoch 15/30
469/469 - 1s - loss: 0.0068 - accuracy: 0.9841 - val_loss: 0.0083 - val_accuracy: 0.9743 - 1s/epoch -
Epoch 16/30
469/469 - 1s - loss: 0.0066 - accuracy: 0.9855 - val_loss: 0.0082 - val_accuracy: 0.9739 - 1s/epoch -
Epoch 17/30
469/469 - 1s - loss: 0.0063 - accuracy: 0.9859 - val_loss: 0.0079 - val_accuracy: 0.9752 - 1s/epoch -
Epoch 18/30
469/469 - 1s - loss: 0.0062 - accuracy: 0.9866 - val_loss: 0.0078 - val_accuracy: 0.9774 - 1s/epoch -
Epoch 19/30
469/469 - 1s - loss: 0.0060 - accuracy: 0.9876 - val_loss: 0.0076 - val_accuracy: 0.9754 - 1s/epoch -
Epoch 20/30
469/469 - 2s - loss: 0.0058 - accuracy: 0.9879 - val_loss: 0.0085 - val_accuracy: 0.9760 - 2s/epoch -


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] y_pred = mlp.evaluate(x_test, y_test, verbose=0)
print('accuracy : ', y_pred[1])
accuracy : 0.9768000245094299
[ ] plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'])
plt.grid()
plt.show()
Train
Validation
Accuracy
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0
5
10
15
20
25
30

colab.research.google.com

박종현, 214823, task 07.ipynb

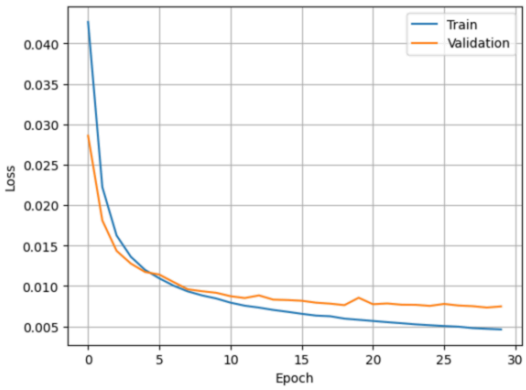
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

+ 코드 + 텍스트 연결 T4 Gemini

Epoch

```
[ ] plt.plot(hist.history['loss'])
    plt.plot(hist.history['val_loss'])
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Validation'])
    plt.grid()
    plt.show()
```




```
[ ] mlp.summary()
```

Model: "sequential"

colab.research.google.com

박종현, 214823, task 07.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

+ 코드 + 텍스트 연결 T4 Gemini

```
[ ] mlp.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1024)	803840
dense_1 (Dense)	(None, 10)	10250

Total params: 814090 (3.11 MB)
Trainable params: 814090 (3.11 MB)
Non-trainable params: 0 (0.00 Byte)

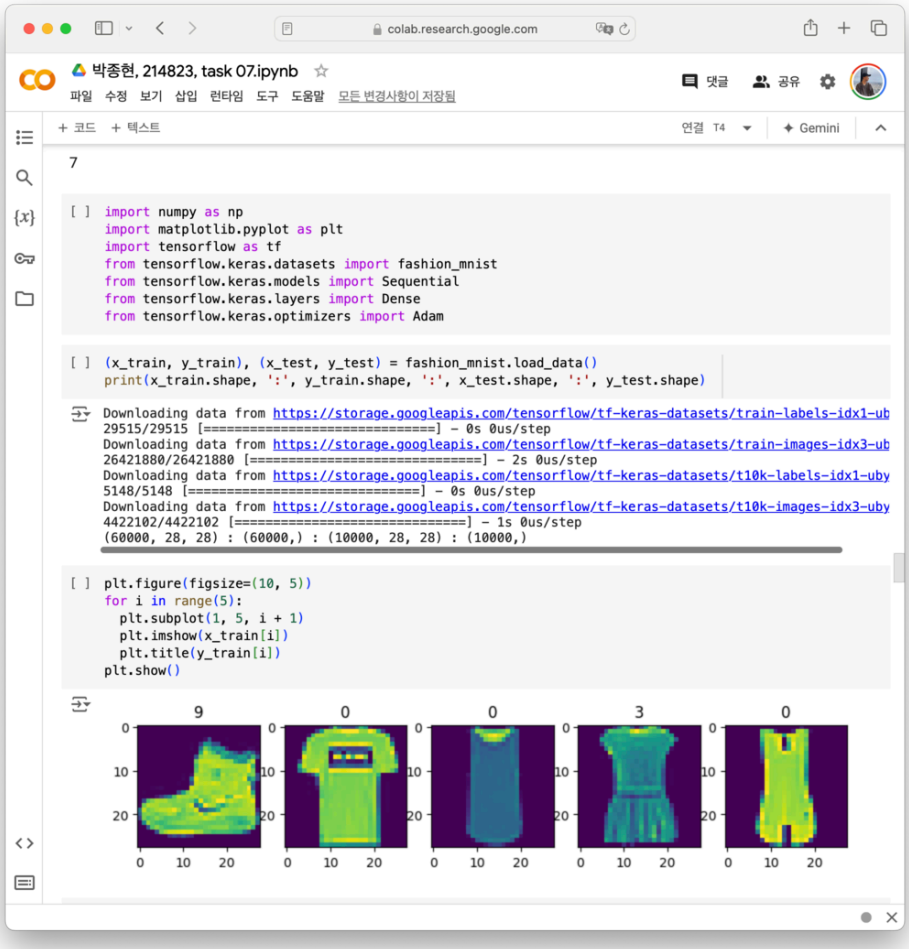
```
[ ] mlp.save('my_mnist_mlp_keras.h5')
```

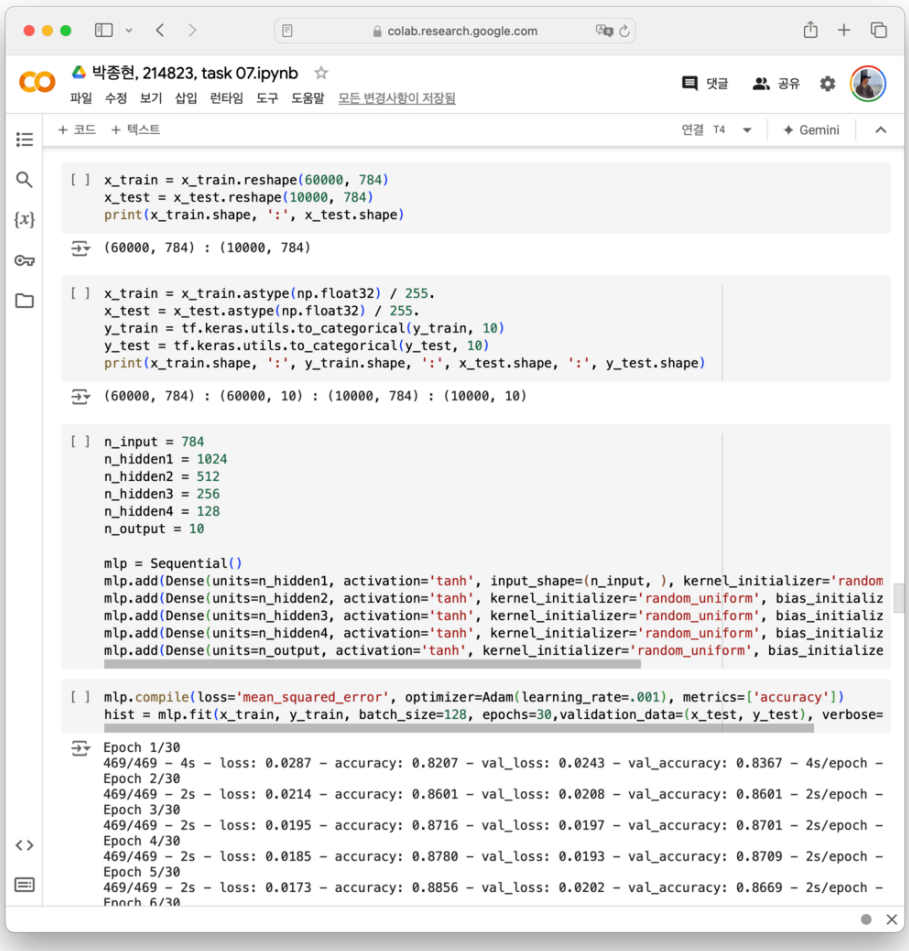
```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving
  saving_api.save_model(
```

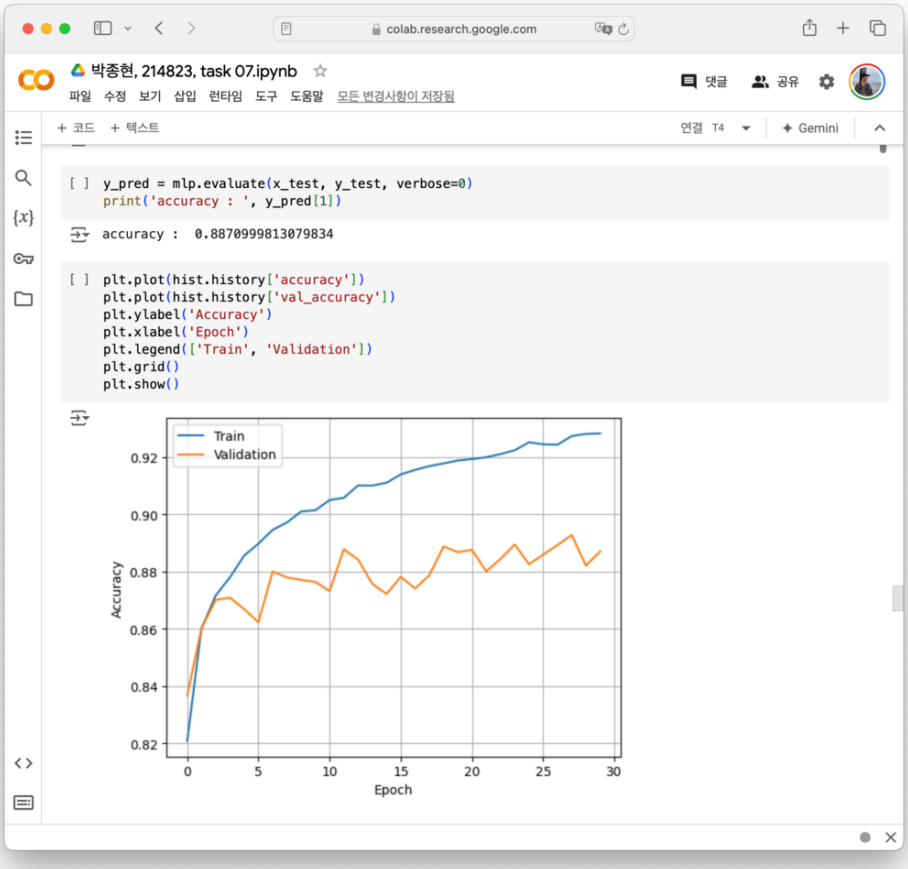
```
[ ] mlp_load = tf.keras.models.load_model('my_mnist_mlp_keras.h5')
    y_pred_load = mlp_load.evaluate(x_test, y_test, verbose=0)
    print('accuracy : ', y_pred_load[1])
```

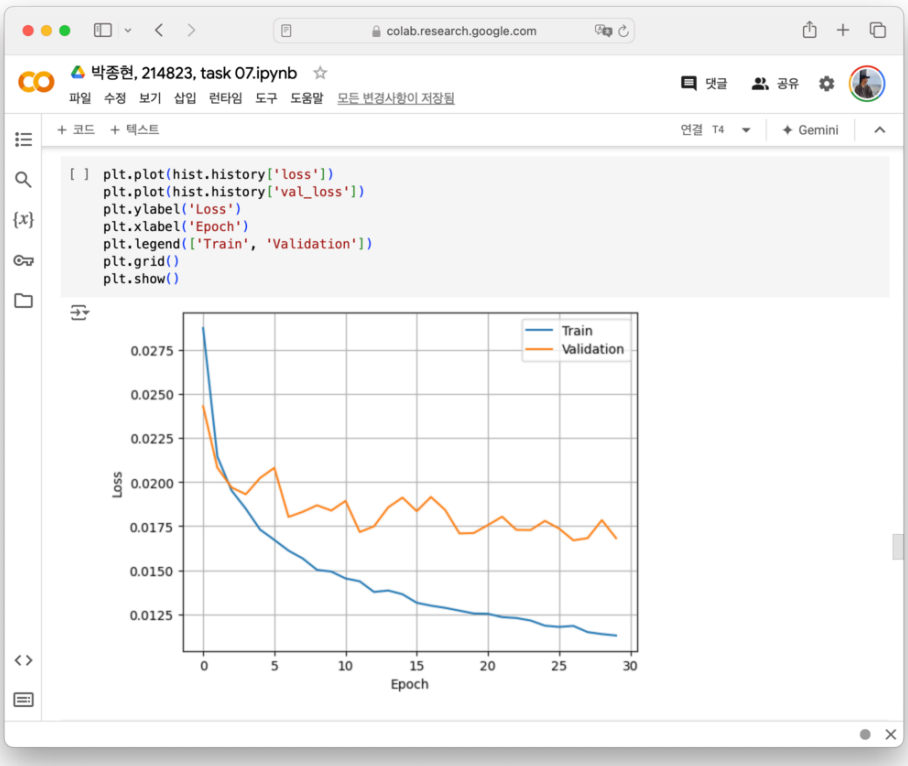
accuracy : 0.9768000245094299

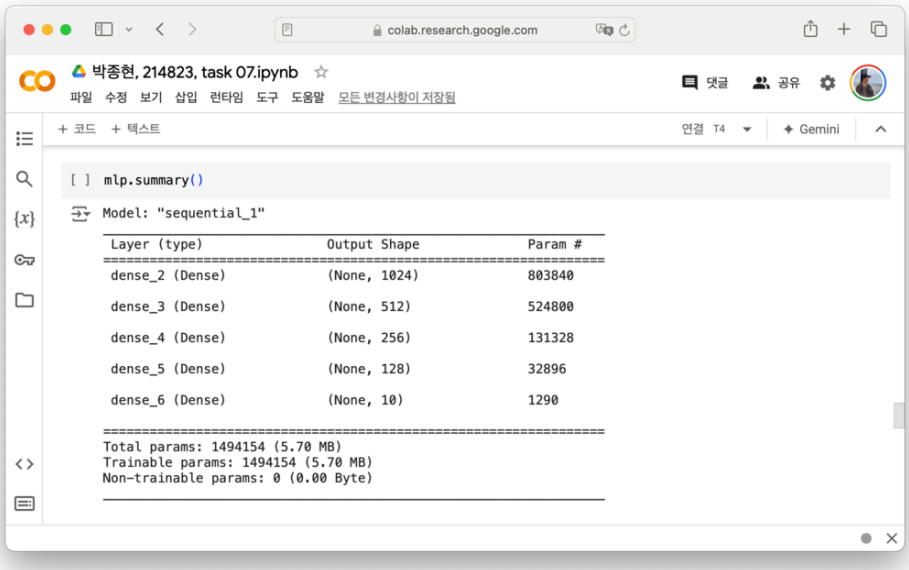
[7] 14-3. 9 15 쪽


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
7
[ ] import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
[ ] (x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
print(x_train.shape, ':', y_train.shape, ':', x_test.shape, ':', y_test.shape)
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ub
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ub
26421880/26421880 [==============================] - 2s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-uby
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-uby
4422102/4422102 [==============================] - 1s 0us/step
(60000, 28, 28) : (60000,) : (10000, 28, 28) : (10000,)
[ ] plt.figure(figsize=(10, 5))
for i in range(5):
plt.subplot(1, 5, i + 1)
plt.imshow(x_train[i])
plt.title(y_train[i])
plt.show()
9
0
0
3
0

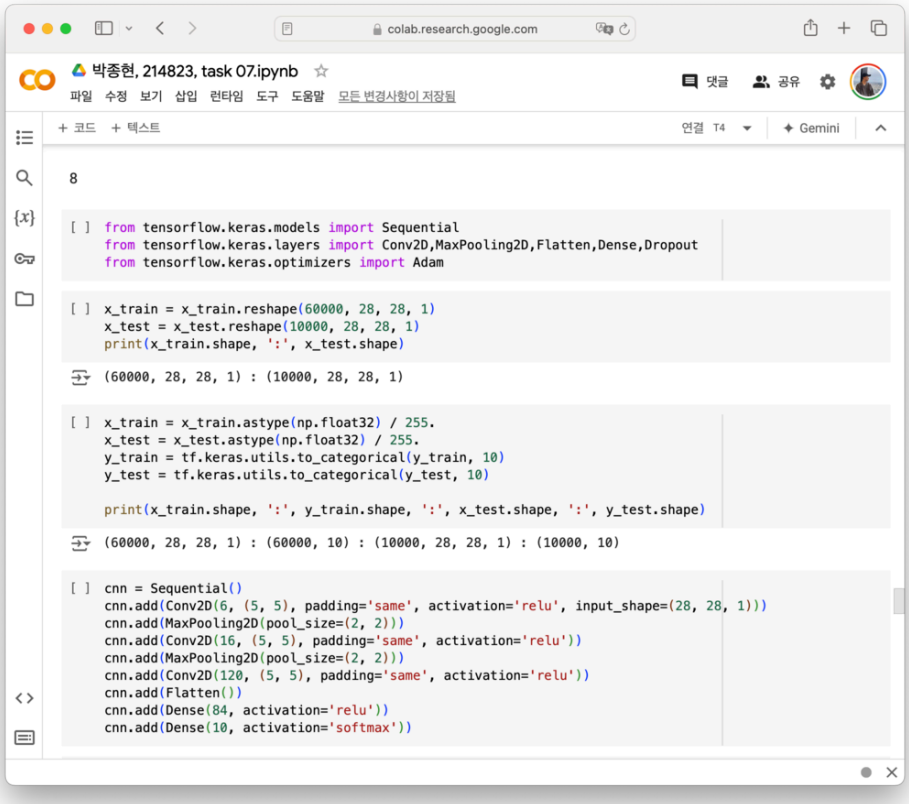

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
print(x_train.shape, ':', x_test.shape)
(60000, 784) : (10000, 784)
[ ] x_train = x_train.astype(np.float32) / 255.
x_test = x_test.astype(np.float32) / 255.
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
print(x_train.shape, ':', y_train.shape, ':', x_test.shape, ':', y_test.shape)
(60000, 784) : (60000, 10) : (10000, 784) : (10000, 10)
[ ] n_input = 784
n_hidden1 = 1024
n_hidden2 = 512
n_hidden3 = 256
n_hidden4 = 128
n_output = 10
mlp = Sequential()
mlp.add(Dense(units=n_hidden1, activation='tanh', input_shape=(n_input,), kernel_initializer='random
mlp.add(Dense(units=n_hidden2, activation='tanh', kernel_initializer='random_uniform', bias_initializ
mlp.add(Dense(units=n_hidden3, activation='tanh', kernel_initializer='random_uniform', bias_initializ
mlp.add(Dense(units=n_hidden4, activation='tanh', kernel_initializer='random_uniform', bias_initializ
mlp.add(Dense(units=n_output, activation='tanh', kernel_initializer='random_uniform', bias_initialize
[ ] mlp.compile(loss='mean_squared_error', optimizer=Adam(learning_rate=.001), metrics=['accuracy'])
hist = mlp.fit(x_train, y_train, batch_size=128, epochs=30,validation_data=(x_test, y_test), verbose=
Epoch 1/30
469/469 - 4s - loss: 0.0287 - accuracy: 0.8207 - val_loss: 0.0243 - val_accuracy: 0.8367 - 4s/epoch -
Epoch 2/30
469/469 - 2s - loss: 0.0214 - accuracy: 0.8601 - val_loss: 0.0208 - val_accuracy: 0.8601 - 2s/epoch -
Epoch 3/30
469/469 - 2s - loss: 0.0195 - accuracy: 0.8716 - val_loss: 0.0197 - val_accuracy: 0.8701 - 2s/epoch -
Epoch 4/30
469/469 - 2s - loss: 0.0185 - accuracy: 0.8780 - val_loss: 0.0193 - val_accuracy: 0.8709 - 2s/epoch -
Epoch 5/30
469/469 - 2s - loss: 0.0173 - accuracy: 0.8856 - val_loss: 0.0202 - val_accuracy: 0.8669 - 2s/epoch -
Epoch 6/30

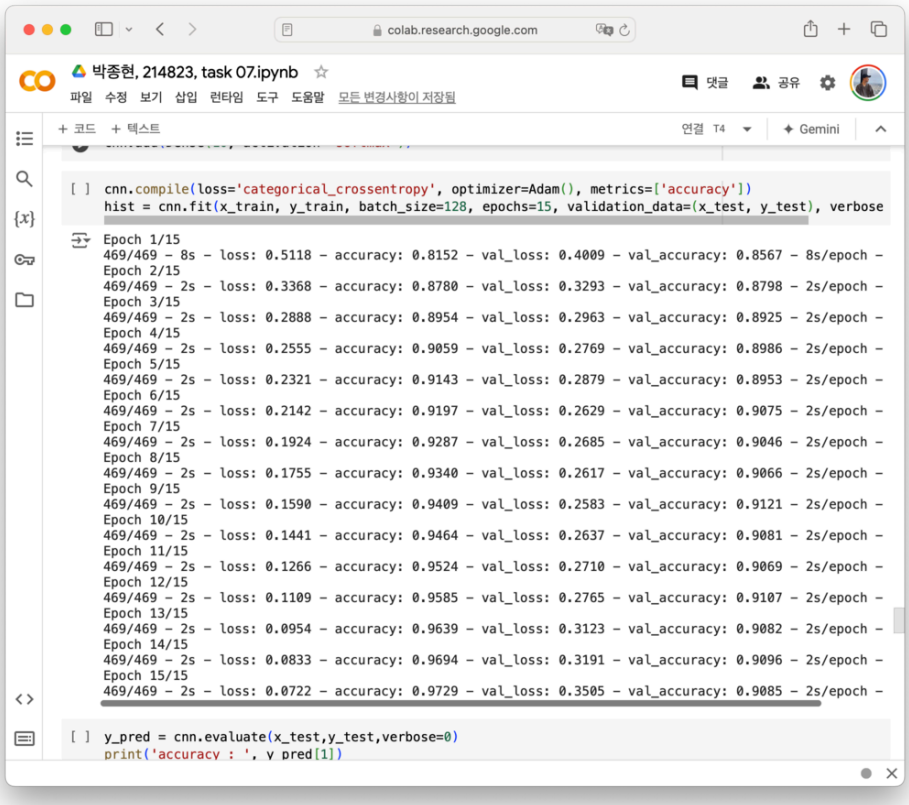

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] y_pred = mlp.evaluate(x_test, y_test, verbose=0)
print('accuracy : ', y_pred[1])
accuracy : 0.8870999813079834
[ ] plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'])
plt.grid()
plt.show()
Train
Validation
Accuracy
Epoch

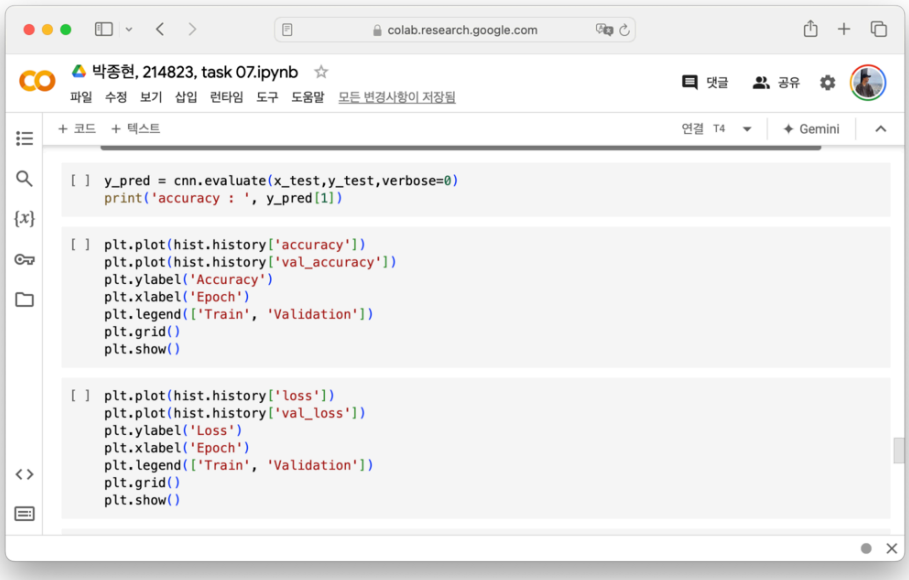

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'])
plt.grid()
plt.show()
Train
Validation
Loss
Epoch

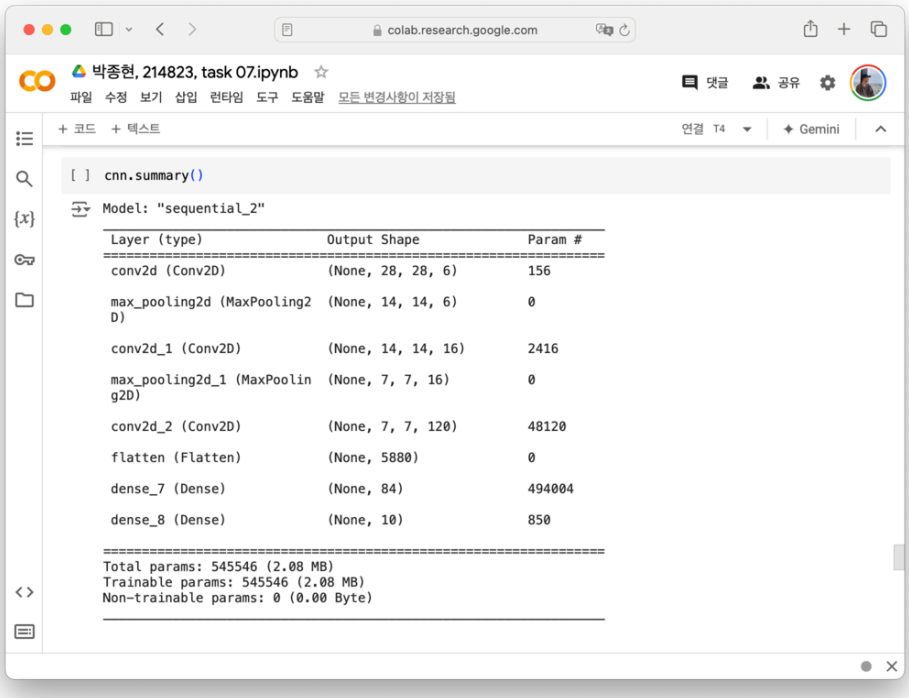

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] mlp.summary()
Model: "sequential_1"
Layer (type) Output Shape Param #
dense_2 (Dense) (None, 1024) 803840
dense_3 (Dense) (None, 512) 524800
dense_4 (Dense) (None, 256) 131328
dense_5 (Dense) (None, 128) 32896
dense_6 (Dense) (None, 10) 1290
Total params: 1494154 (5.70 MB)
Trainable params: 1494154 (5.70 MB)
Non-trainable params: 0 (0.00 Byte)

[8] 14-3. 24 29 쪽

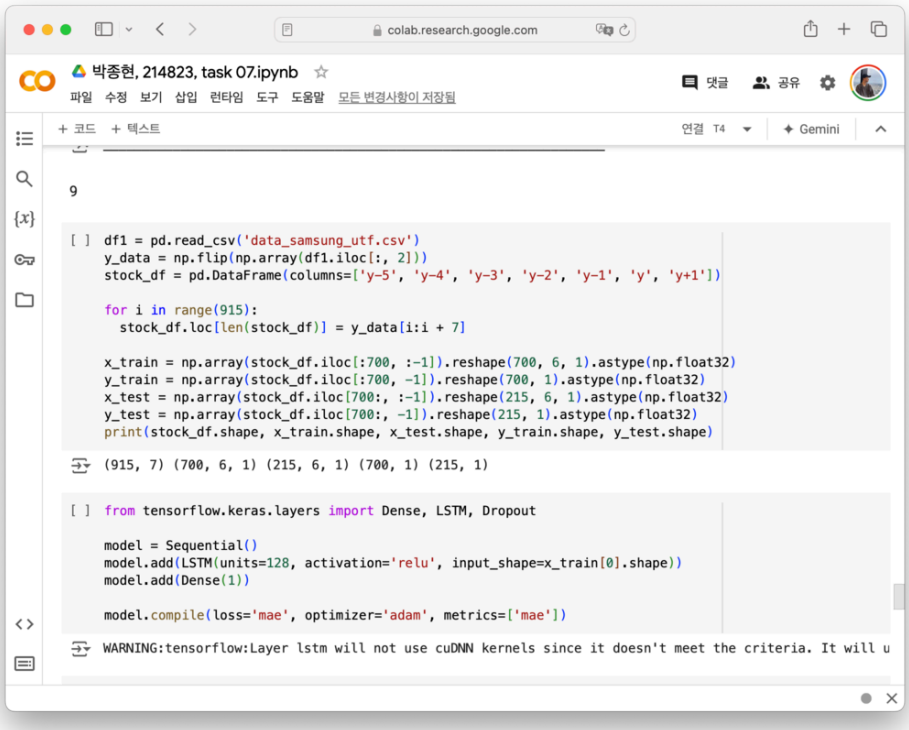

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
8
[ ] from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Dropout
from tensorflow.keras.optimizers import Adam
[ ] x_train = x_train.reshape(60000, 28, 28, 1)
x_test = x_test.reshape(10000, 28, 28, 1)
print(x_train.shape, ':', x_test.shape)
(60000, 28, 28, 1) : (10000, 28, 28, 1)
[ ] x_train = x_train.astype(np.float32) / 255.
x_test = x_test.astype(np.float32) / 255.
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)
print(x_train.shape, ':', y_train.shape, ':', x_test.shape, ':', y_test.shape)
(60000, 28, 28, 1) : (60000, 10) : (10000, 28, 28, 1) : (10000, 10)
[ ] cnn = Sequential()
cnn.add(Conv2D(6, (5, 5), padding='same', activation='relu', input_shape=(28, 28, 1)))
cnn.add(MaxPooling2D(pool_size=(2, 2)))
cnn.add(Conv2D(16, (5, 5), padding='same', activation='relu'))
cnn.add(MaxPooling2D(pool_size=(2, 2)))
cnn.add(Conv2D(120, (5, 5), padding='same', activation='relu'))
cnn.add(Flatten())
cnn.add(Dense(84, activation='relu'))
cnn.add(Dense(10, activation='softmax'))

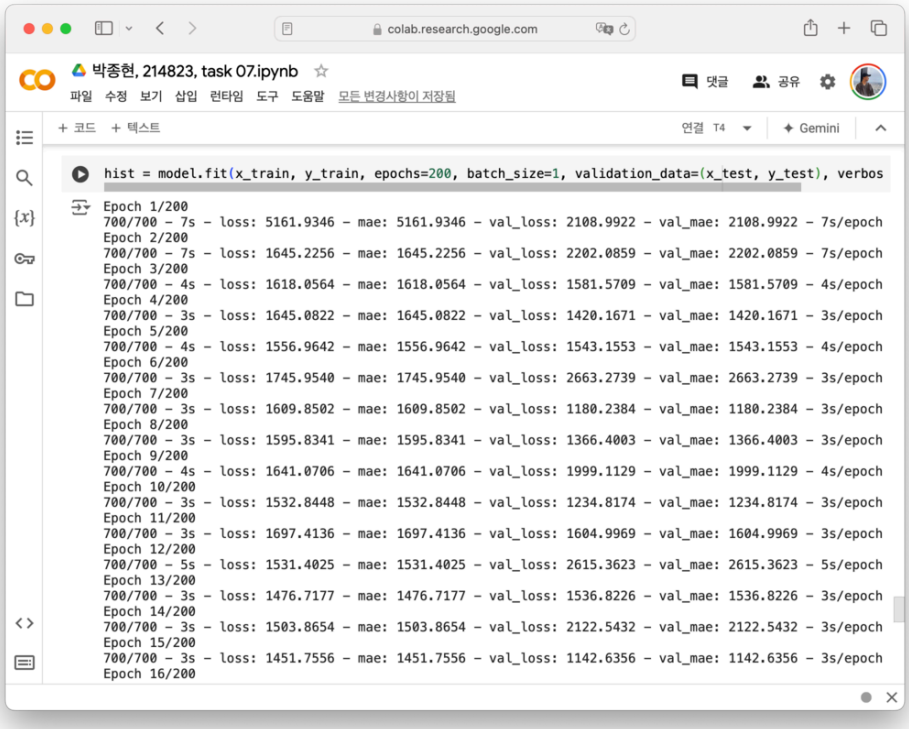

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] cnn.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])
hist = cnn.fit(x_train, y_train, batch_size=128, epochs=15, validation_data=(x_test, y_test), verbose
Epoch 1/15
469/469 - 8s - loss: 0.5118 - accuracy: 0.8152 - val_loss: 0.4009 - val_accuracy: 0.8567 - 8s/epoch -
Epoch 2/15
469/469 - 2s - loss: 0.3368 - accuracy: 0.8780 - val_loss: 0.3293 - val_accuracy: 0.8798 - 2s/epoch -
Epoch 3/15
469/469 - 2s - loss: 0.2888 - accuracy: 0.8954 - val_loss: 0.2963 - val_accuracy: 0.8925 - 2s/epoch -
Epoch 4/15
469/469 - 2s - loss: 0.2555 - accuracy: 0.9059 - val_loss: 0.2769 - val_accuracy: 0.8986 - 2s/epoch -
Epoch 5/15
469/469 - 2s - loss: 0.2321 - accuracy: 0.9143 - val_loss: 0.2879 - val_accuracy: 0.8953 - 2s/epoch -
Epoch 6/15
469/469 - 2s - loss: 0.2142 - accuracy: 0.9197 - val_loss: 0.2629 - val_accuracy: 0.9075 - 2s/epoch -
Epoch 7/15
469/469 - 2s - loss: 0.1924 - accuracy: 0.9287 - val_loss: 0.2685 - val_accuracy: 0.9046 - 2s/epoch -
Epoch 8/15
469/469 - 2s - loss: 0.1755 - accuracy: 0.9340 - val_loss: 0.2617 - val_accuracy: 0.9066 - 2s/epoch -
Epoch 9/15
469/469 - 2s - loss: 0.1590 - accuracy: 0.9409 - val_loss: 0.2583 - val_accuracy: 0.9121 - 2s/epoch -
Epoch 10/15
469/469 - 2s - loss: 0.1441 - accuracy: 0.9464 - val_loss: 0.2637 - val_accuracy: 0.9081 - 2s/epoch -
Epoch 11/15
469/469 - 2s - loss: 0.1266 - accuracy: 0.9524 - val_loss: 0.2710 - val_accuracy: 0.9069 - 2s/epoch -
Epoch 12/15
469/469 - 2s - loss: 0.1109 - accuracy: 0.9585 - val_loss: 0.2765 - val_accuracy: 0.9107 - 2s/epoch -
Epoch 13/15
469/469 - 2s - loss: 0.0954 - accuracy: 0.9639 - val_loss: 0.3123 - val_accuracy: 0.9082 - 2s/epoch -
Epoch 14/15
469/469 - 2s - loss: 0.0833 - accuracy: 0.9694 - val_loss: 0.3191 - val_accuracy: 0.9096 - 2s/epoch -
Epoch 15/15
469/469 - 2s - loss: 0.0722 - accuracy: 0.9729 - val_loss: 0.3505 - val_accuracy: 0.9085 - 2s/epoch -
[ ] y_pred = cnn.evaluate(x_test,y_test,verbose=0)
print('accuracy : ', y_pred[1])

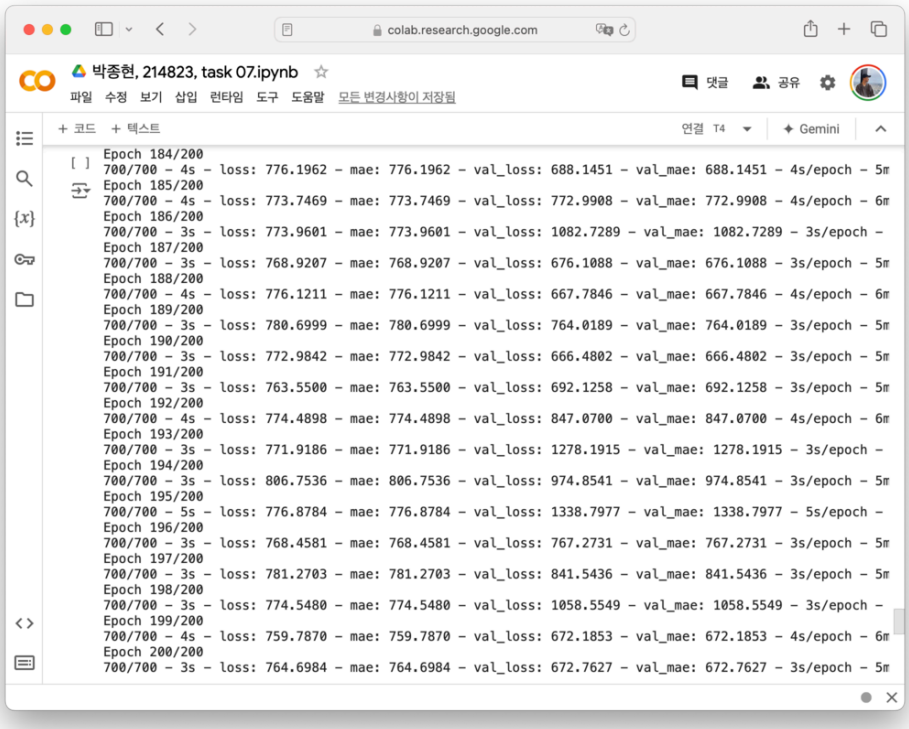

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] y_pred = cnn.evaluate(x_test,y_test,verbose=0)
print('accuracy : ', y_pred[1])
[ ] plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'])
plt.grid()
plt.show()
[ ] plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'])
plt.grid()
plt.show()

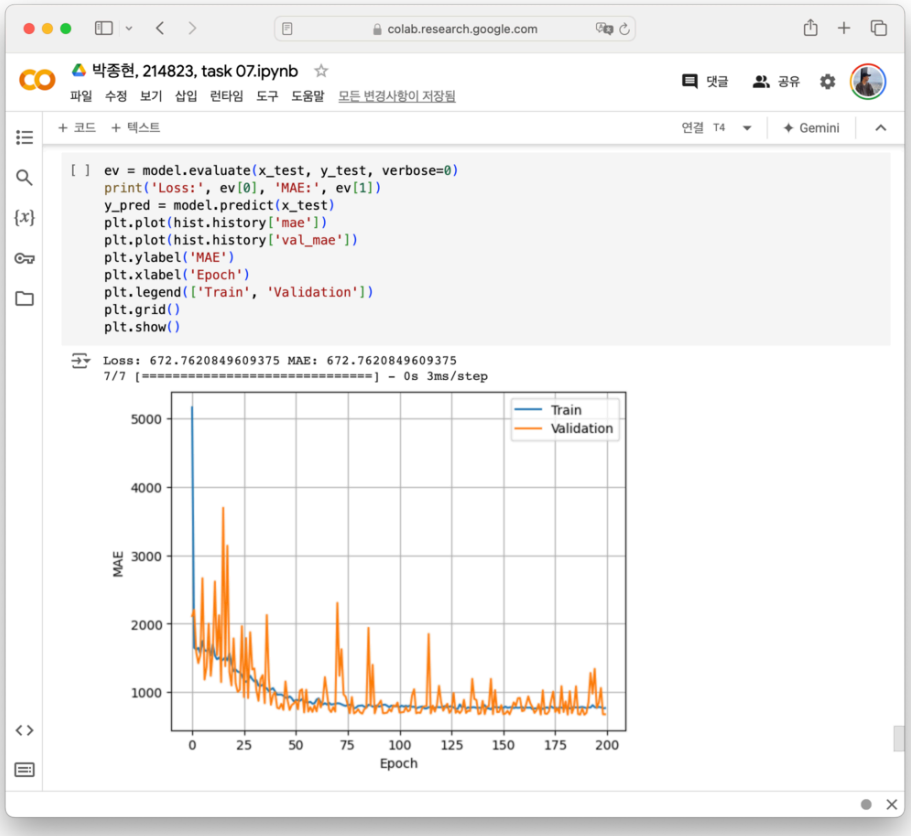

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] cnn.summary()
Model: "sequential_2"
Layer (type) Output Shape Param #
conv2d (Conv2D) (None, 28, 28, 6) 156
max_pooling2d (MaxPooling2D) (None, 14, 14, 6) 0
conv2d_1 (Conv2D) (None, 14, 14, 16) 2416
max_pooling2d_1 (MaxPooling2D) (None, 7, 7, 16) 0
conv2d_2 (Conv2D) (None, 7, 7, 120) 48120
flatten (Flatten) (None, 5880) 0
dense_7 (Dense) (None, 84) 494004
dense_8 (Dense) (None, 10) 850
Total params: 545546 (2.08 MB)
Trainable params: 545546 (2.08 MB)
Non-trainable params: 0 (0.00 Byte)

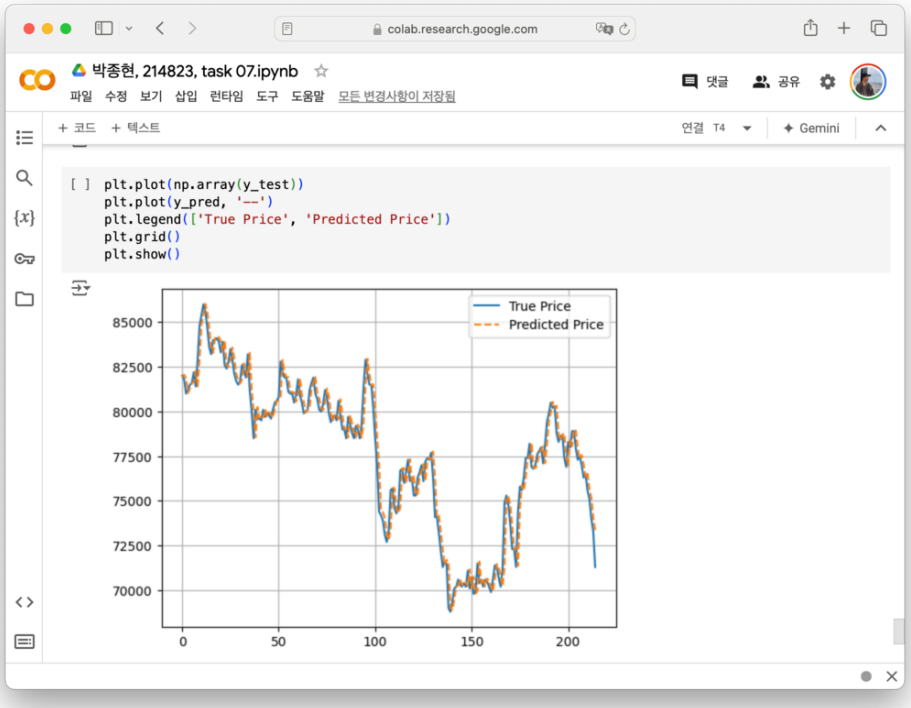
[9] 14-3. 34 38 쪽

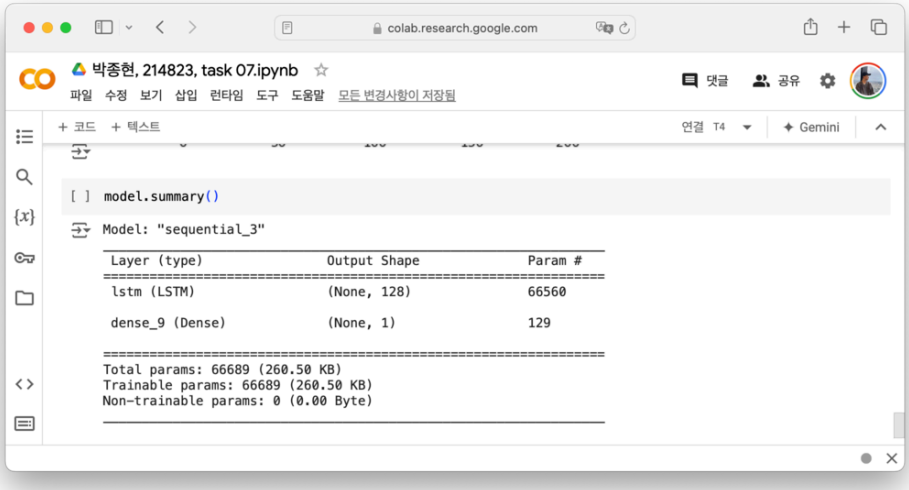

colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
9
[ ] df1 = pd.read_csv('data_samsung_utf.csv')
y_data = np.flip(np.array(df1.iloc[:, 2]))
stock_df = pd.DataFrame(columns=['y-5', 'y-4', 'y-3', 'y-2', 'y-1', 'y', 'y+1'])
for i in range(915):
stock_df.loc[len(stock_df)] = y_data[i:i + 7]
x_train = np.array(stock_df.iloc[:700, :-1]).reshape(700, 6, 1).astype(np.float32)
y_train = np.array(stock_df.iloc[:700, -1]).reshape(700, 1).astype(np.float32)
x_test = np.array(stock_df.iloc[700:, :-1]).reshape(215, 6, 1).astype(np.float32)
y_test = np.array(stock_df.iloc[700:, -1]).reshape(215, 1).astype(np.float32)
print(stock_df.shape, x_train.shape, x_test.shape, y_train.shape, y_test.shape)
(915, 7) (700, 6, 1) (215, 6, 1) (700, 1) (215, 1)
[ ] from tensorflow.keras.layers import Dense, LSTM, Dropout
model = Sequential()
model.add(LSTM(units=128, activation='relu', input_shape=x_train[0].shape))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam', metrics=['mae'])
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will u


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
hist = model.fit(x_train, y_train, epochs=200, batch_size=1, validation_data=(x_test, y_test), verbos
Epoch 1/200
700/700 - 7s - loss: 5161.9346 - mae: 5161.9346 - val_loss: 2108.9922 - val_mae: 2108.9922 - 7s/epoch
Epoch 2/200
700/700 - 7s - loss: 1645.2256 - mae: 1645.2256 - val_loss: 2202.0859 - val_mae: 2202.0859 - 7s/epoch
Epoch 3/200
700/700 - 4s - loss: 1618.0564 - mae: 1618.0564 - val_loss: 1581.5709 - val_mae: 1581.5709 - 4s/epoch
Epoch 4/200
700/700 - 3s - loss: 1645.0822 - mae: 1645.0822 - val_loss: 1420.1671 - val_mae: 1420.1671 - 3s/epoch
Epoch 5/200
700/700 - 4s - loss: 1556.9642 - mae: 1556.9642 - val_loss: 1543.1553 - val_mae: 1543.1553 - 4s/epoch
Epoch 6/200
700/700 - 3s - loss: 1745.9540 - mae: 1745.9540 - val_loss: 2663.2739 - val_mae: 2663.2739 - 3s/epoch
Epoch 7/200
700/700 - 3s - loss: 1609.8502 - mae: 1609.8502 - val_loss: 1180.2384 - val_mae: 1180.2384 - 3s/epoch
Epoch 8/200
700/700 - 3s - loss: 1595.8341 - mae: 1595.8341 - val_loss: 1366.4003 - val_mae: 1366.4003 - 3s/epoch
Epoch 9/200
700/700 - 4s - loss: 1641.0706 - mae: 1641.0706 - val_loss: 1999.1129 - val_mae: 1999.1129 - 4s/epoch
Epoch 10/200
700/700 - 3s - loss: 1532.8448 - mae: 1532.8448 - val_loss: 1234.8174 - val_mae: 1234.8174 - 3s/epoch
Epoch 11/200
700/700 - 3s - loss: 1697.4136 - mae: 1697.4136 - val_loss: 1604.9969 - val_mae: 1604.9969 - 3s/epoch
Epoch 12/200
700/700 - 5s - loss: 1531.4025 - mae: 1531.4025 - val_loss: 2615.3623 - val_mae: 2615.3623 - 5s/epoch
Epoch 13/200
700/700 - 3s - loss: 1476.7177 - mae: 1476.7177 - val_loss: 1536.8226 - val_mae: 1536.8226 - 3s/epoch
Epoch 14/200
700/700 - 3s - loss: 1503.8654 - mae: 1503.8654 - val_loss: 2122.5432 - val_mae: 2122.5432 - 3s/epoch
Epoch 15/200
700/700 - 3s - loss: 1451.7556 - mae: 1451.7556 - val_loss: 1142.6356 - val_mae: 1142.6356 - 3s/epoch
Epoch 16/200


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
Epoch 184/200
700/700 - 4s - loss: 776.1962 - mae: 776.1962 - val_loss: 688.1451 - val_mae: 688.1451 - 4s/epoch - 5m
Epoch 185/200
700/700 - 4s - loss: 773.7469 - mae: 773.7469 - val_loss: 772.9908 - val_mae: 772.9908 - 4s/epoch - 6m
Epoch 186/200
700/700 - 3s - loss: 773.9601 - mae: 773.9601 - val_loss: 1082.7289 - val_mae: 1082.7289 - 3s/epoch -
Epoch 187/200
700/700 - 3s - loss: 768.9207 - mae: 768.9207 - val_loss: 676.1088 - val_mae: 676.1088 - 3s/epoch - 5m
Epoch 188/200
700/700 - 4s - loss: 776.1211 - mae: 776.1211 - val_loss: 667.7846 - val_mae: 667.7846 - 4s/epoch - 6m
Epoch 189/200
700/700 - 3s - loss: 780.6999 - mae: 780.6999 - val_loss: 764.0189 - val_mae: 764.0189 - 3s/epoch - 5m
Epoch 190/200
700/700 - 3s - loss: 772.9842 - mae: 772.9842 - val_loss: 666.4802 - val_mae: 666.4802 - 3s/epoch - 5m
Epoch 191/200
700/700 - 3s - loss: 763.5500 - mae: 763.5500 - val_loss: 692.1258 - val_mae: 692.1258 - 3s/epoch - 5m
Epoch 192/200
700/700 - 4s - loss: 774.4898 - mae: 774.4898 - val_loss: 847.0700 - val_mae: 847.0700 - 4s/epoch - 6m
Epoch 193/200
700/700 - 3s - loss: 771.9186 - mae: 771.9186 - val_loss: 1278.1915 - val_mae: 1278.1915 - 3s/epoch -
Epoch 194/200
700/700 - 3s - loss: 806.7536 - mae: 806.7536 - val_loss: 974.8541 - val_mae: 974.8541 - 3s/epoch - 5m
Epoch 195/200
700/700 - 5s - loss: 776.8784 - mae: 776.8784 - val_loss: 1338.7977 - val_mae: 1338.7977 - 5s/epoch -
Epoch 196/200
700/700 - 3s - loss: 768.4581 - mae: 768.4581 - val_loss: 767.2731 - val_mae: 767.2731 - 3s/epoch - 5m
Epoch 197/200
700/700 - 3s - loss: 781.2703 - mae: 781.2703 - val_loss: 841.5436 - val_mae: 841.5436 - 3s/epoch - 5m
Epoch 198/200
700/700 - 3s - loss: 774.5480 - mae: 774.5480 - val_loss: 1058.5549 - val_mae: 1058.5549 - 3s/epoch -
Epoch 199/200
700/700 - 4s - loss: 759.7870 - mae: 759.7870 - val_loss: 672.1853 - val_mae: 672.1853 - 4s/epoch - 6m
Epoch 200/200
700/700 - 3s - loss: 764.6984 - mae: 764.6984 - val_loss: 672.7627 - val_mae: 672.7627 - 3s/epoch - 5m


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] ev = model.evaluate(x_test, y_test, verbose=0)
print('Loss:', ev[0], 'MAE:', ev[1])
y_pred = model.predict(x_test)
plt.plot(hist.history['mae'])
plt.plot(hist.history['val_mae'])
plt.ylabel('MAE')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'])
plt.grid()
plt.show()
Loss: 672.7620849609375 MAE: 672.7620849609375
7/7 [==============================] - 0s 3ms/step
Train
Validation
MAE
Epoch


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
+ 코드 + 텍스트
연결 T4 Gemini
[ ] plt.plot(np.array(y_test))
plt.plot(y_pred, '--')
plt.legend(['True Price', 'Predicted Price'])
plt.grid()
plt.show()
True Price
Predicted Price


colab.research.google.com
박종현, 214823, task 07.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글
공유
+ 코드 + 텍스트
연결 T4
Gemini
[ ] model.summary()
Model: "sequential_3"
Layer (type) Output Shape Param #
lstm (LSTM) (None, 128) 66560
dense_9 (Dense) (None, 1) 129
Total params: 66689 (260.50 KB)
Trainable params: 66689 (260.50 KB)
Non-trainable params: 0 (0.00 Byte)