MNIST 이미지 분류

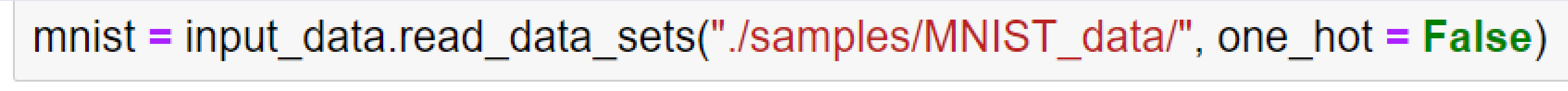
2017010698
수학과 오서영

목차

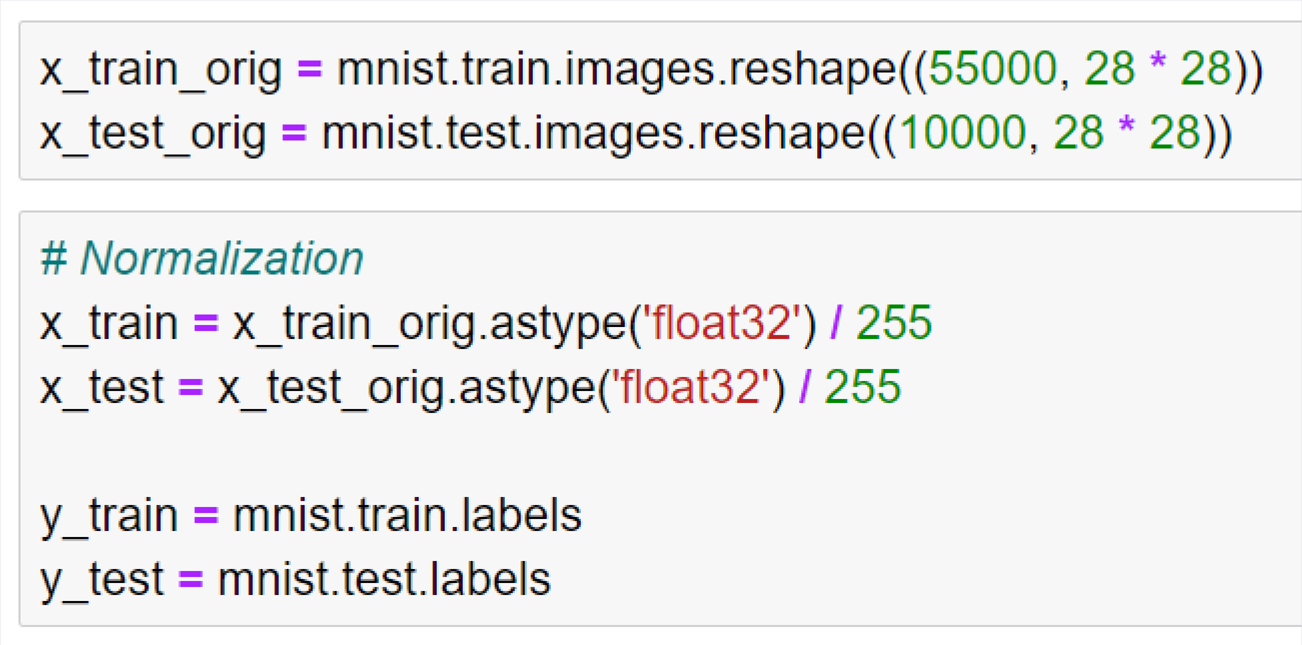
1 SLP

2 MLP

1 SLP : Single-Layer Perceptron



```
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot = False)
```



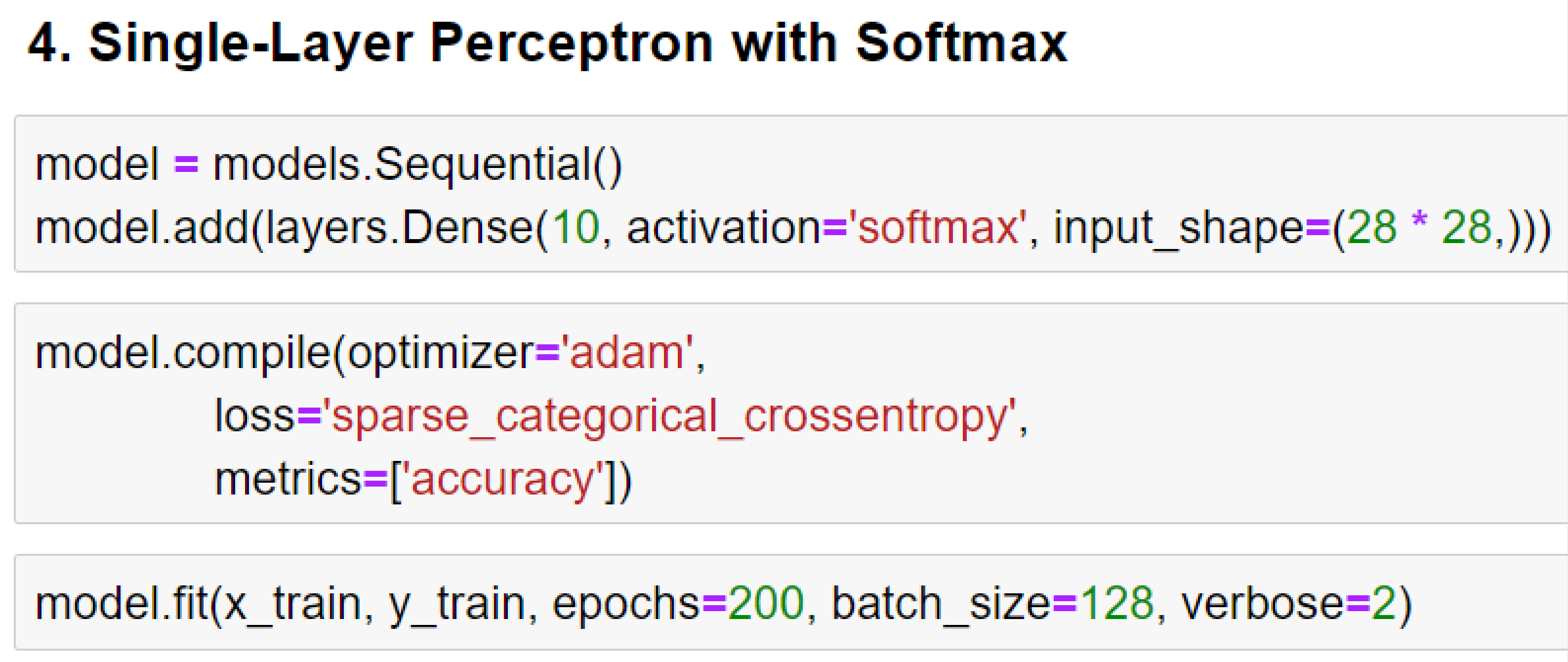
```
x_train_orig = mnist.train.images.reshape((55000, 28 * 28))
x_test_orig = mnist.test.images.reshape((10000, 28 * 28))
```

```
# Normalization
x_train = x_train_orig.astype('float32') / 255
x_test = x_test_orig.astype('float32') / 255

y_train = mnist.train.labels
y_test = mnist.test.labels
```

1

SLP : Single-Layer Perceptron



4. Single-Layer Perceptron with Softmax

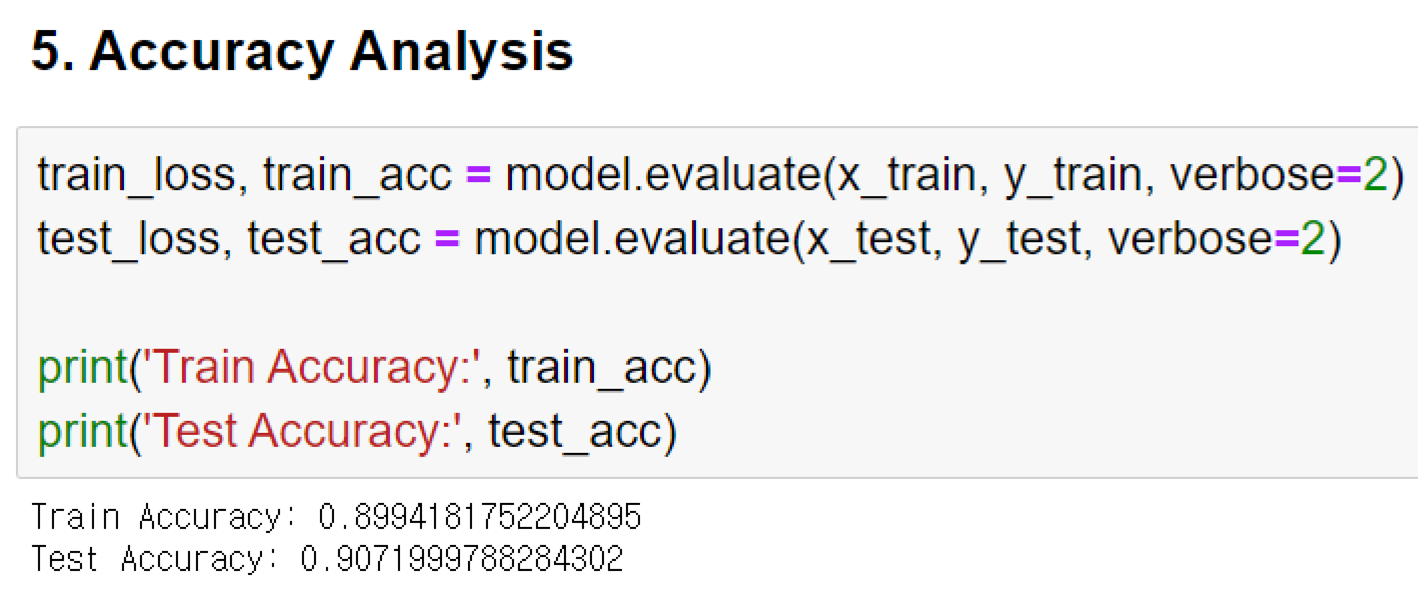
```
model = models.Sequential()
model.add(layers.Dense(10, activation='softmax', input_shape=(28 * 28,)))
```

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.fit(x_train, y_train, epochs=200, batch_size=128, verbose=2)
```

1

SLP : Single-Layer Perceptron



5. Accuracy Analysis

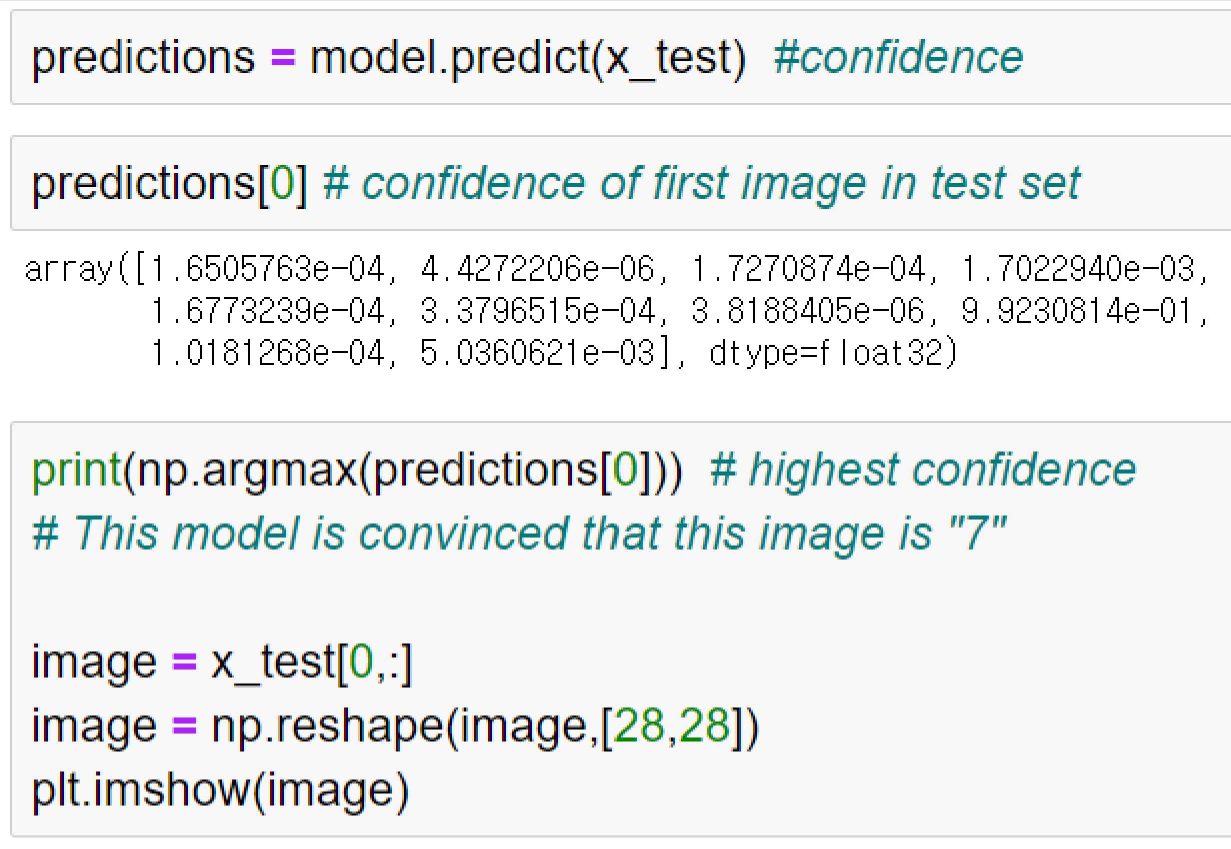
```
train_loss, train_acc = model.evaluate(x_train, y_train, verbose=2)
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)

print('Train Accuracy:', train_acc)
print('Test Accuracy:', test_acc)
```

```
Train Accuracy: 0.8994181752204895
Test Accuracy: 0.9071999788284302
```

Underfitting (과소적합)

1 SLP : Single-Layer Perceptron



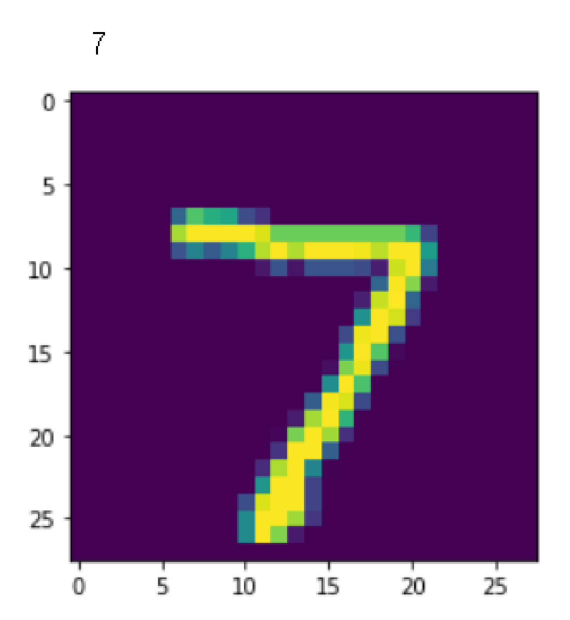
```
predictions = model.predict(x_test)  #confidence
```

```
predictions[0] # confidence of first image in test set
```

```
array([1.6505763e-04, 4.4272206e-06, 1.7270874e-04, 1.7022940e-03,
       1.6773239e-04, 3.3796515e-04, 3.8188405e-06, 9.9230814e-01,
       1.0181268e-04, 5.0360621e-03], dtype=float32)
```

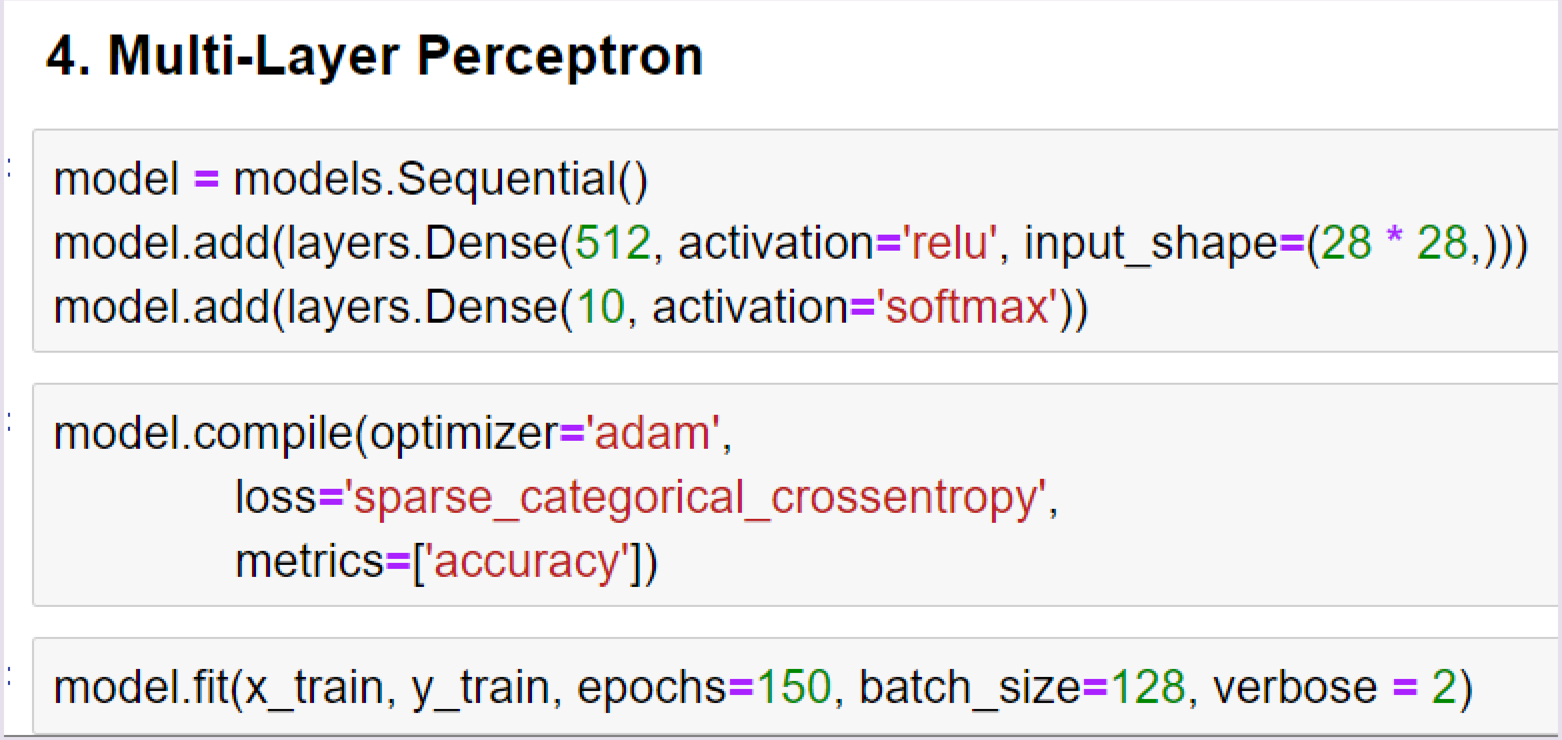
```
print(np.argmax(predictions[0]))  # highest confidence
# This model is convinced that this image is "7"

image = x_test[0,:]
image = np.reshape(image,[28,28])
plt.imshow(image)
```



```
7
```

2 MLP : Multi-Layer Perceptron



4. Multi-Layer Perceptron

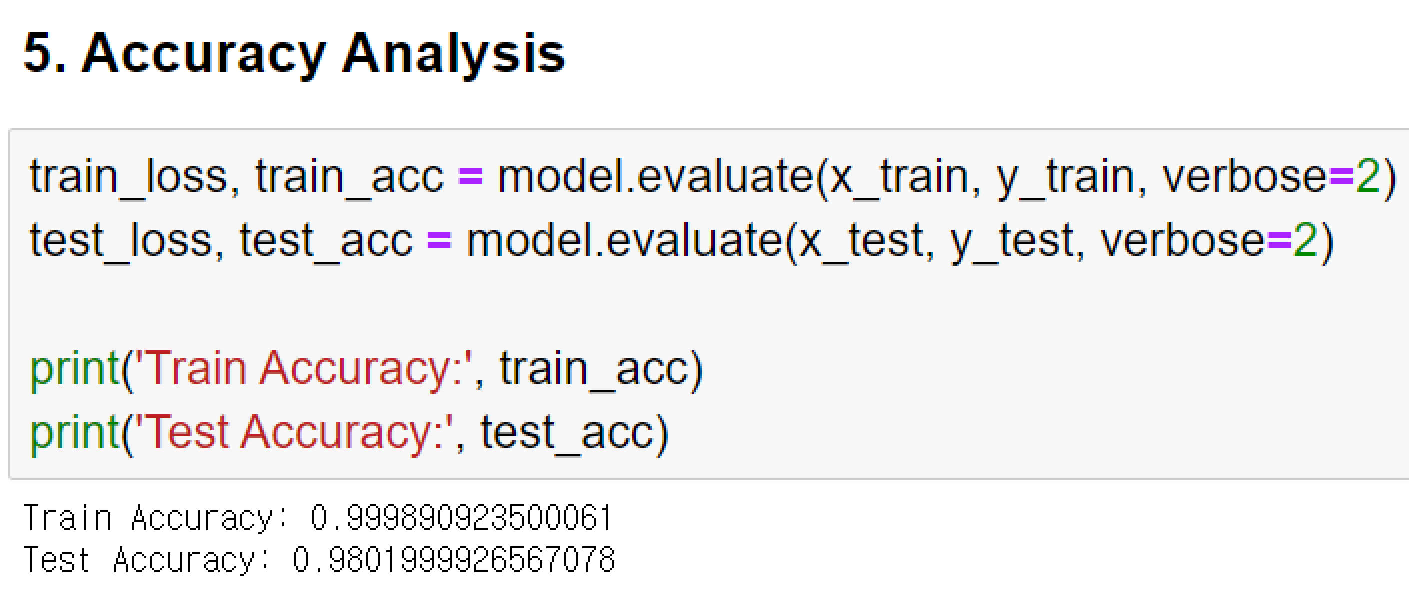
```
model = models.Sequential()
model.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
model.add(layers.Dense(10, activation='softmax'))
```

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.fit(x_train, y_train, epochs=150, batch_size=128, verbose = 2)
```

2

MLP : Multi-Layer Perceptron

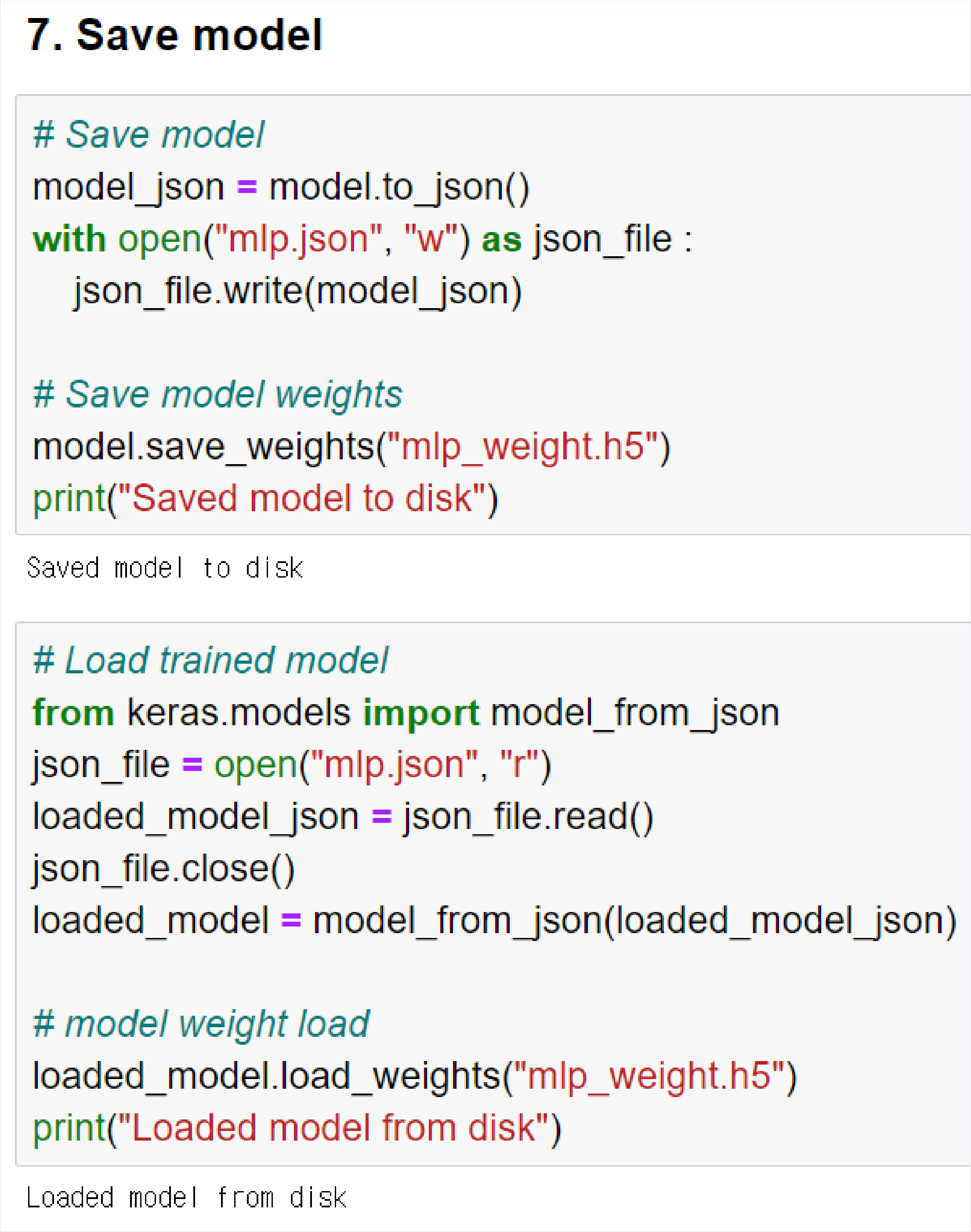


5. Accuracy Analysis

```
train_loss, train_acc = model.evaluate(x_train, y_train, verbose=2)
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)

print('Train Accuracy:', train_acc)
print('Test Accuracy:', test_acc)
```

```
Train Accuracy: 0.999890923500061
Test Accuracy: 0.9801999926567078
```



7. Save model

```python
# Save model
model_json = model.to_json()
with open("mlp.json", "w") as json_file :
    json_file.write(model_json)

# Save model weights
model.save_weights("mlp_weight.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

```python
# Load trained model
from keras.models import model_from_json
json_file = open("mlp.json", "r")
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)

# model weight load
loaded_model.load_weights("mlp_weight.h5")
print("Loaded model from disk")
```

```
Loaded model from disk
```

끝