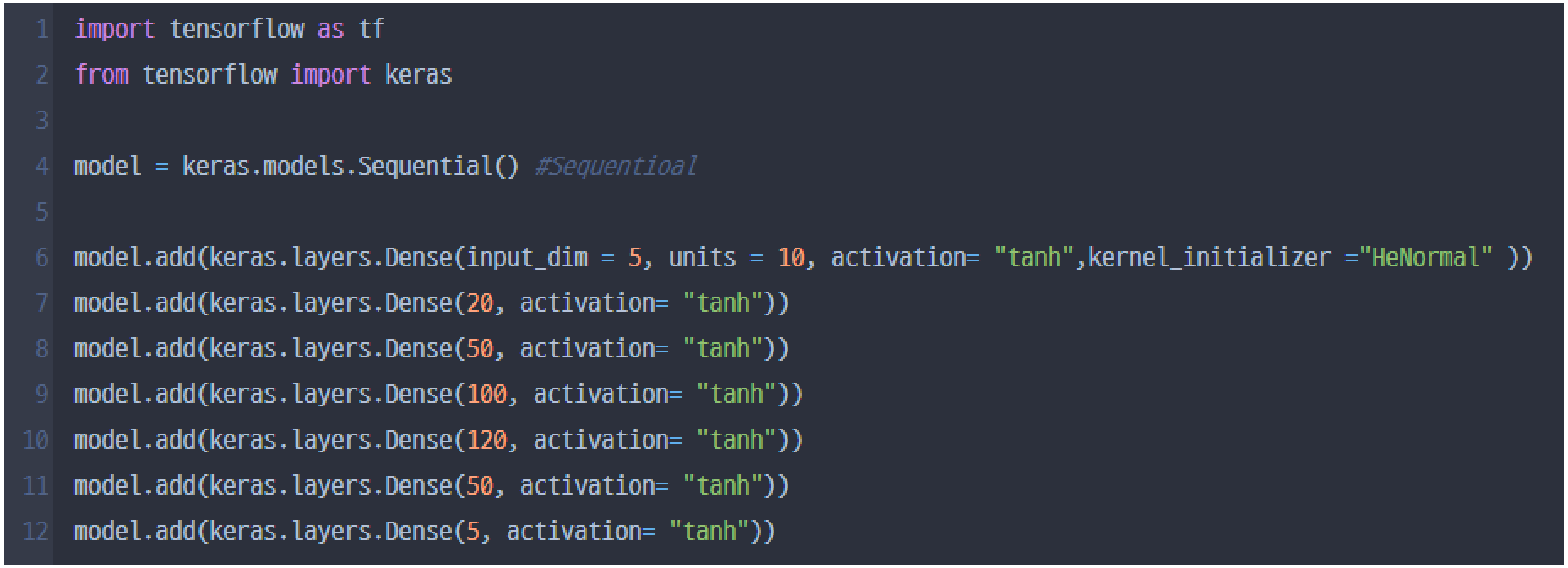
Training Result

2021.10.02

Model

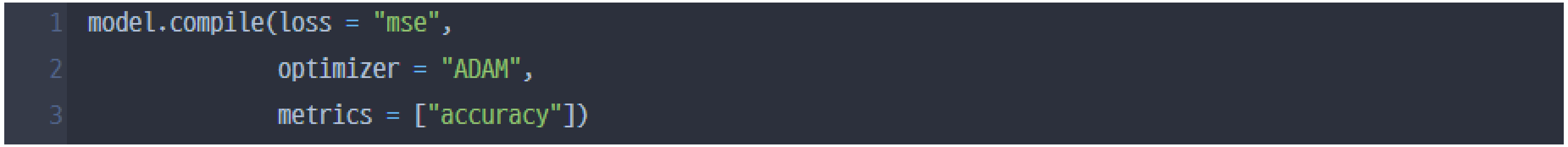
Data = 100000
Train : 60000, Validation : 20000, Test : 20000 (6:2:2)
N값의 변화에 따라 input_dim과 마지막 레이어의 노드 수만 N에 맞추면 됩니다.



```
import tensorflow as tf
from tensorflow import keras

model = keras.models.Sequential() #Sequentioal

model.add(keras.layers.Dense(input_dim = 5, units = 10, activation= "tanh",kernel_initializer ="HeNormal" ))
model.add(keras.layers.Dense(20, activation= "tanh"))
model.add(keras.layers.Dense(50, activation= "tanh"))
model.add(keras.layers.Dense(100, activation= "tanh"))
model.add(keras.layers.Dense(120, activation= "tanh"))
model.add(keras.layers.Dense(50, activation= "tanh"))
model.add(keras.layers.Dense(5, activation= "tanh"))
```



```
model.compile(loss = "mse",
              optimizer = "ADAM",
              metrics = ["accuracy"])
```

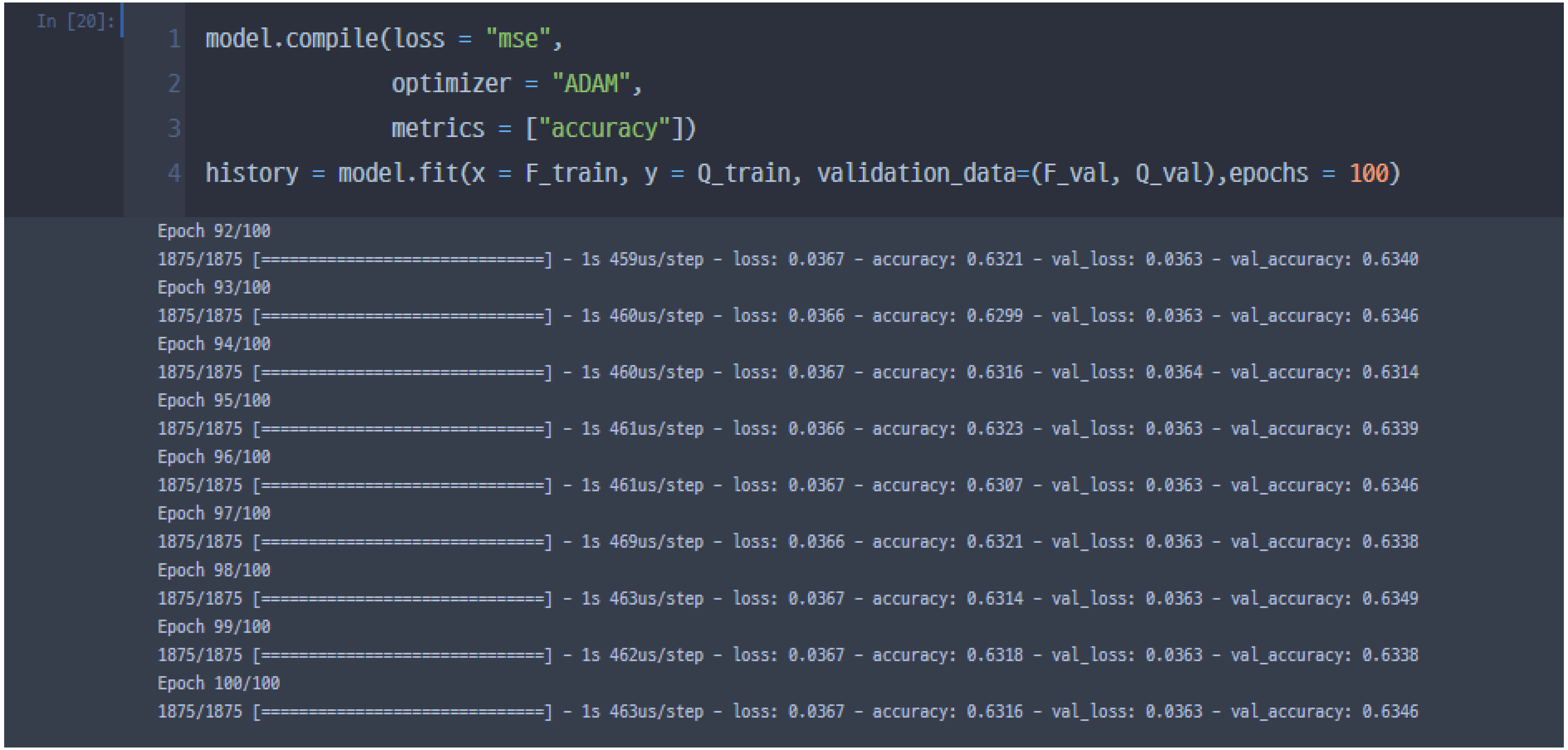
슬라이드 순서

- N = 5, epoch = 100
- N = 5, epoch = 1000
- N = 2, epoch = 50
- N = 3, epoch = 50
- N = 7, epoch = 50
- N = 7, epoch = 1000

각각
F train 시 F-validation accuracy와 F-test accuracy, nF-test accuracy
nF train 시 nF-validation accuracy, nF-test accuracy의 결과를 순서대로 넣었습니다.

F_train : Epoch = 30 정도부터 63% 수렴

N=5, epoch = 100 → Val_accuracy = 63%

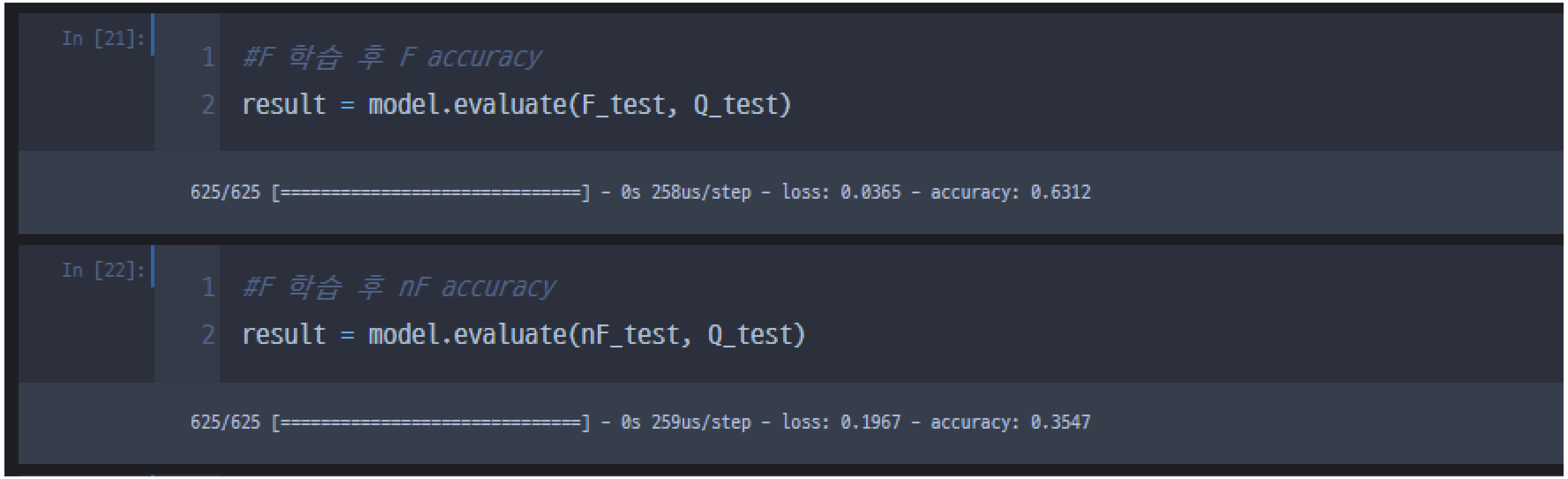


In [20]:

```
model.compile(loss = "mse",
              optimizer = "ADAM",
              metrics = ["accuracy"])
history = model.fit(x = F_train, y = Q_train, validation_data=(F_val, Q_val),epochs = 100)
```

```
Epoch 92/100
1875/1875 [==============================] - 1s 459us/step - loss: 0.0367 - accuracy: 0.6321 - val_loss: 0.0363 - val_accuracy: 0.6340
Epoch 93/100
1875/1875 [==============================] - 1s 460us/step - loss: 0.0366 - accuracy: 0.6299 - val_loss: 0.0363 - val_accuracy: 0.6346
Epoch 94/100
1875/1875 [==============================] - 1s 460us/step - loss: 0.0367 - accuracy: 0.6316 - val_loss: 0.0364 - val_accuracy: 0.6314
Epoch 95/100
1875/1875 [==============================] - 1s 461us/step - loss: 0.0366 - accuracy: 0.6323 - val_loss: 0.0363 - val_accuracy: 0.6339
Epoch 96/100
1875/1875 [==============================] - 1s 461us/step - loss: 0.0367 - accuracy: 0.6307 - val_loss: 0.0363 - val_accuracy: 0.6346
Epoch 97/100
1875/1875 [==============================] - 1s 469us/step - loss: 0.0366 - accuracy: 0.6321 - val_loss: 0.0363 - val_accuracy: 0.6338
Epoch 98/100
1875/1875 [==============================] - 1s 463us/step - loss: 0.0367 - accuracy: 0.6314 - val_loss: 0.0363 - val_accuracy: 0.6349
Epoch 99/100
1875/1875 [==============================] - 1s 462us/step - loss: 0.0367 - accuracy: 0.6318 - val_loss: 0.0363 - val_accuracy: 0.6338
Epoch 100/100
1875/1875 [==============================] - 1s 463us/step - loss: 0.0367 - accuracy: 0.6316 - val_loss: 0.0363 - val_accuracy: 0.6346
```

N=5, epoch = 100



```
In [21]:
#F 학습 후 F accuracy
result = model.evaluate(F_test, Q_test)
```

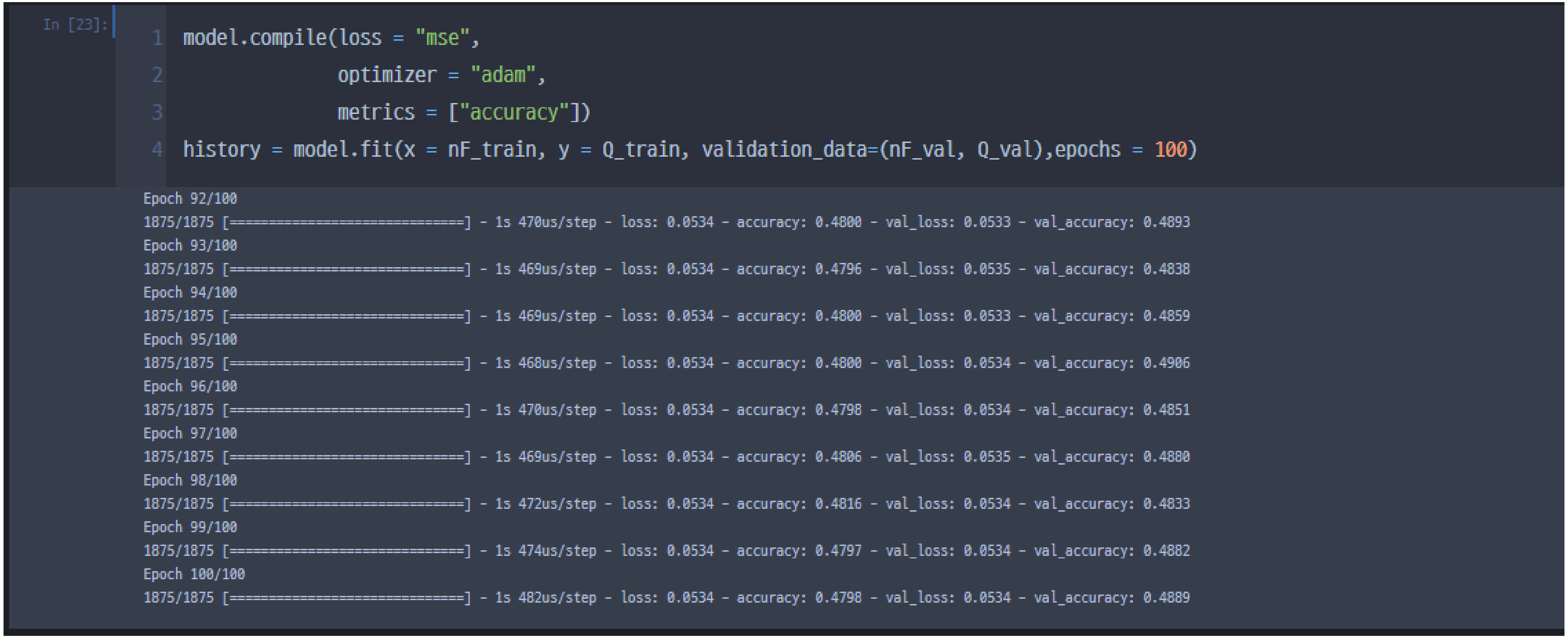
```
625/625 [==============================] - 0s 258us/step - loss: 0.0365 - accuracy: 0.6312
```

```
In [22]:
#F 학습 후 nF accuracy
result = model.evaluate(nF_test, Q_test)
```

```
625/625 [==============================] - 0s 259us/step - loss: 0.1967 - accuracy: 0.3547
```

nF_train : Epoch = 30 정도부터 48% 수렴

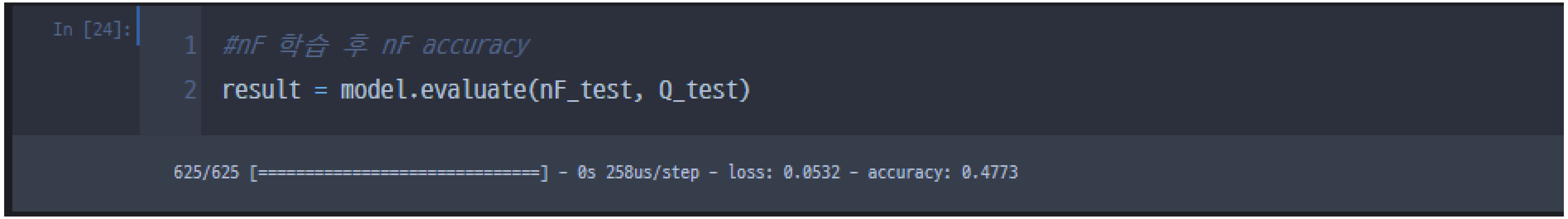
N=5, epoch = 100



```
model.compile(loss = "mse",
              optimizer = "adam",
              metrics = ["accuracy"])
history = model.fit(x = nF_train, y = Q_train, validation_data=(nF_val, Q_val),epochs = 100)
```

```
Epoch 92/100
1875/1875 [==============================] - 1s 470us/step - loss: 0.0534 - accuracy: 0.4800 - val_loss: 0.0533 - val_accuracy: 0.4893
Epoch 93/100
1875/1875 [==============================] - 1s 469us/step - loss: 0.0534 - accuracy: 0.4796 - val_loss: 0.0535 - val_accuracy: 0.4838
Epoch 94/100
1875/1875 [==============================] - 1s 469us/step - loss: 0.0534 - accuracy: 0.4800 - val_loss: 0.0533 - val_accuracy: 0.4859
Epoch 95/100
1875/1875 [==============================] - 1s 468us/step - loss: 0.0534 - accuracy: 0.4800 - val_loss: 0.0534 - val_accuracy: 0.4906
Epoch 96/100
1875/1875 [==============================] - 1s 470us/step - loss: 0.0534 - accuracy: 0.4798 - val_loss: 0.0534 - val_accuracy: 0.4851
Epoch 97/100
1875/1875 [==============================] - 1s 469us/step - loss: 0.0534 - accuracy: 0.4806 - val_loss: 0.0535 - val_accuracy: 0.4880
Epoch 98/100
1875/1875 [==============================] - 1s 472us/step - loss: 0.0534 - accuracy: 0.4816 - val_loss: 0.0534 - val_accuracy: 0.4833
Epoch 99/100
1875/1875 [==============================] - 1s 474us/step - loss: 0.0534 - accuracy: 0.4797 - val_loss: 0.0534 - val_accuracy: 0.4882
Epoch 100/100
1875/1875 [==============================] - 1s 482us/step - loss: 0.0534 - accuracy: 0.4798 - val_loss: 0.0534 - val_accuracy: 0.4889
```

N=5, epoch = 100

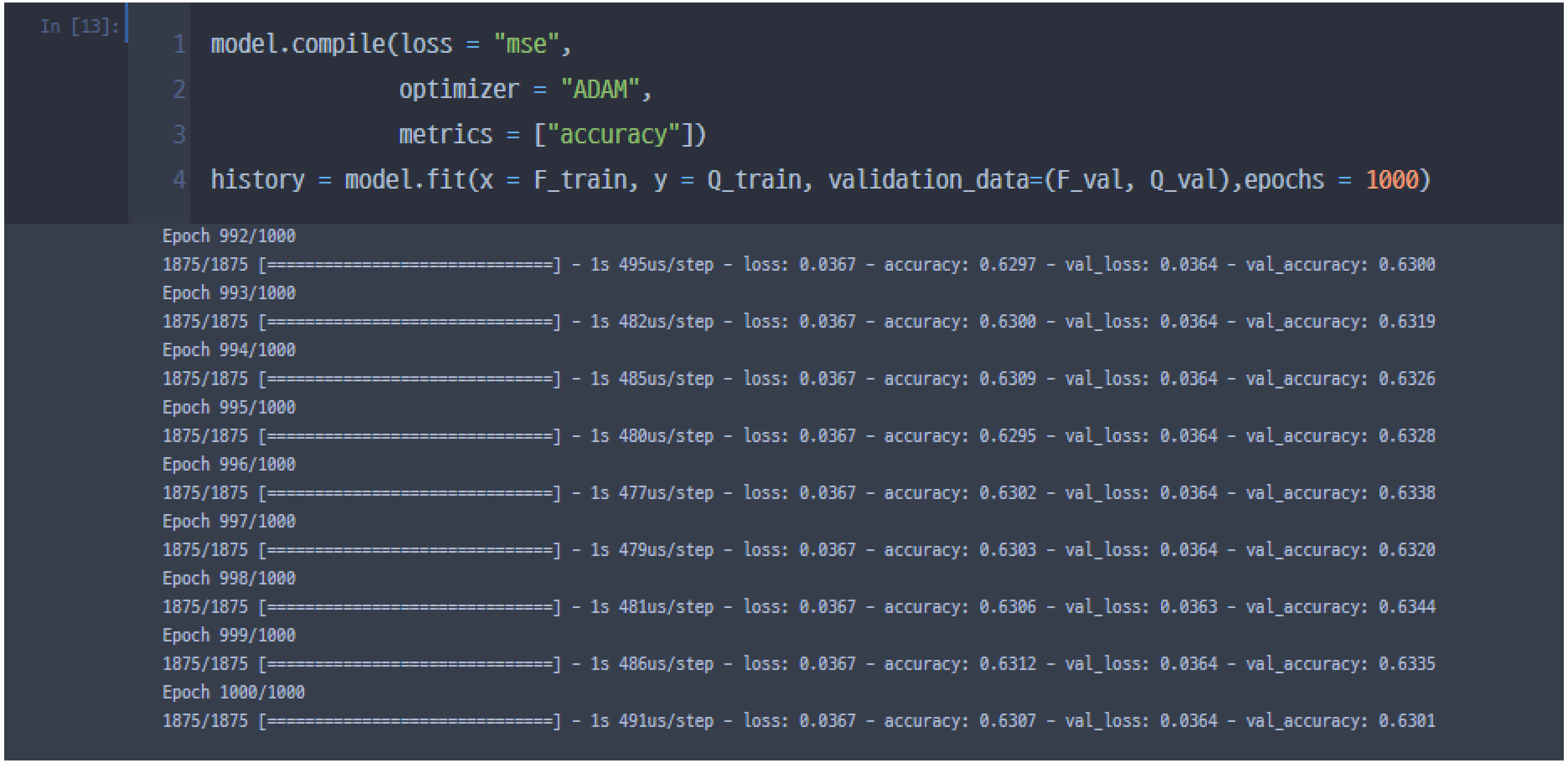


```
#nF 학습 후 nF accuracy
result = model.evaluate(nF_test, Q_test)
```

```
625/625 [==============================] - 0s 258us/step - loss: 0.0532 - accuracy: 0.4773
```

F_train : Epoch = 30 정도부터 63% 수렴

N=5, epoch = 1000 → F_train = 63%

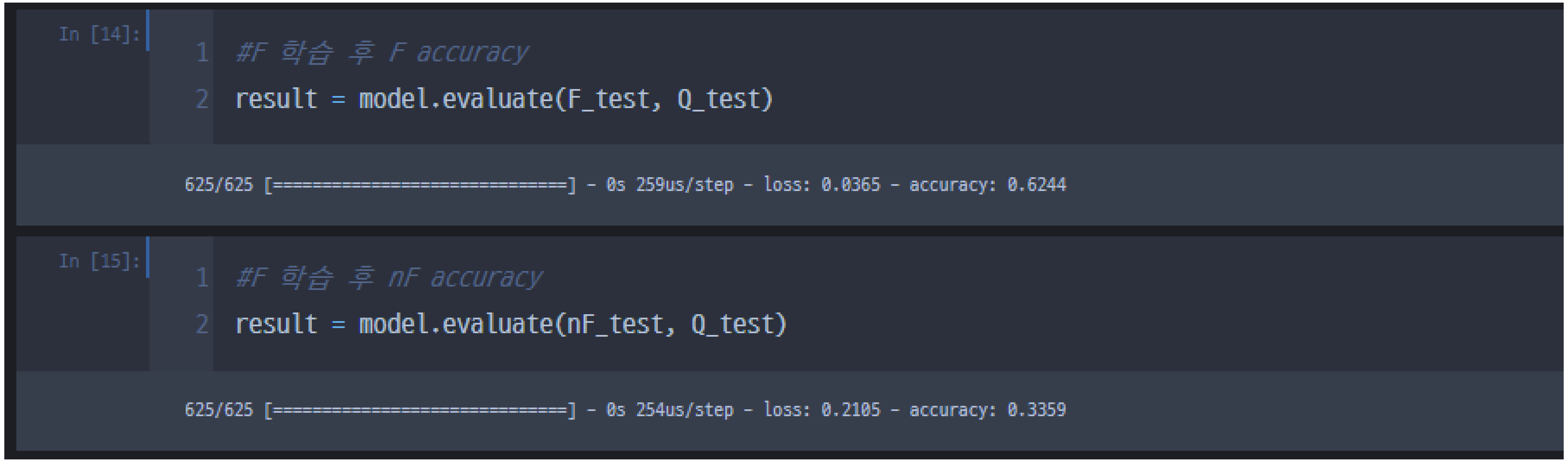


In [13]:

```
model.compile(loss = "mse",
              optimizer = "ADAM",
              metrics = ["accuracy"])
history = model.fit(x = F_train, y = Q_train, validation_data=(F_val, Q_val),epochs = 1000)
```

```
Epoch 992/1000
1875/1875 [==============================] - 1s 495us/step - loss: 0.0367 - accuracy: 0.6297 - val_loss: 0.0364 - val_accuracy: 0.6300
Epoch 993/1000
1875/1875 [==============================] - 1s 482us/step - loss: 0.0367 - accuracy: 0.6300 - val_loss: 0.0364 - val_accuracy: 0.6319
Epoch 994/1000
1875/1875 [==============================] - 1s 485us/step - loss: 0.0367 - accuracy: 0.6309 - val_loss: 0.0364 - val_accuracy: 0.6326
Epoch 995/1000
1875/1875 [==============================] - 1s 480us/step - loss: 0.0367 - accuracy: 0.6295 - val_loss: 0.0364 - val_accuracy: 0.6328
Epoch 996/1000
1875/1875 [==============================] - 1s 477us/step - loss: 0.0367 - accuracy: 0.6302 - val_loss: 0.0364 - val_accuracy: 0.6338
Epoch 997/1000
1875/1875 [==============================] - 1s 479us/step - loss: 0.0367 - accuracy: 0.6303 - val_loss: 0.0364 - val_accuracy: 0.6320
Epoch 998/1000
1875/1875 [==============================] - 1s 481us/step - loss: 0.0367 - accuracy: 0.6306 - val_loss: 0.0363 - val_accuracy: 0.6344
Epoch 999/1000
1875/1875 [==============================] - 1s 486us/step - loss: 0.0367 - accuracy: 0.6312 - val_loss: 0.0364 - val_accuracy: 0.6335
Epoch 1000/1000
1875/1875 [==============================] - 1s 491us/step - loss: 0.0367 - accuracy: 0.6307 - val_loss: 0.0364 - val_accuracy: 0.6301
```

N=5, epoch = 1000



```
#F 학습 후 F accuracy
result = model.evaluate(F_test, Q_test)
```

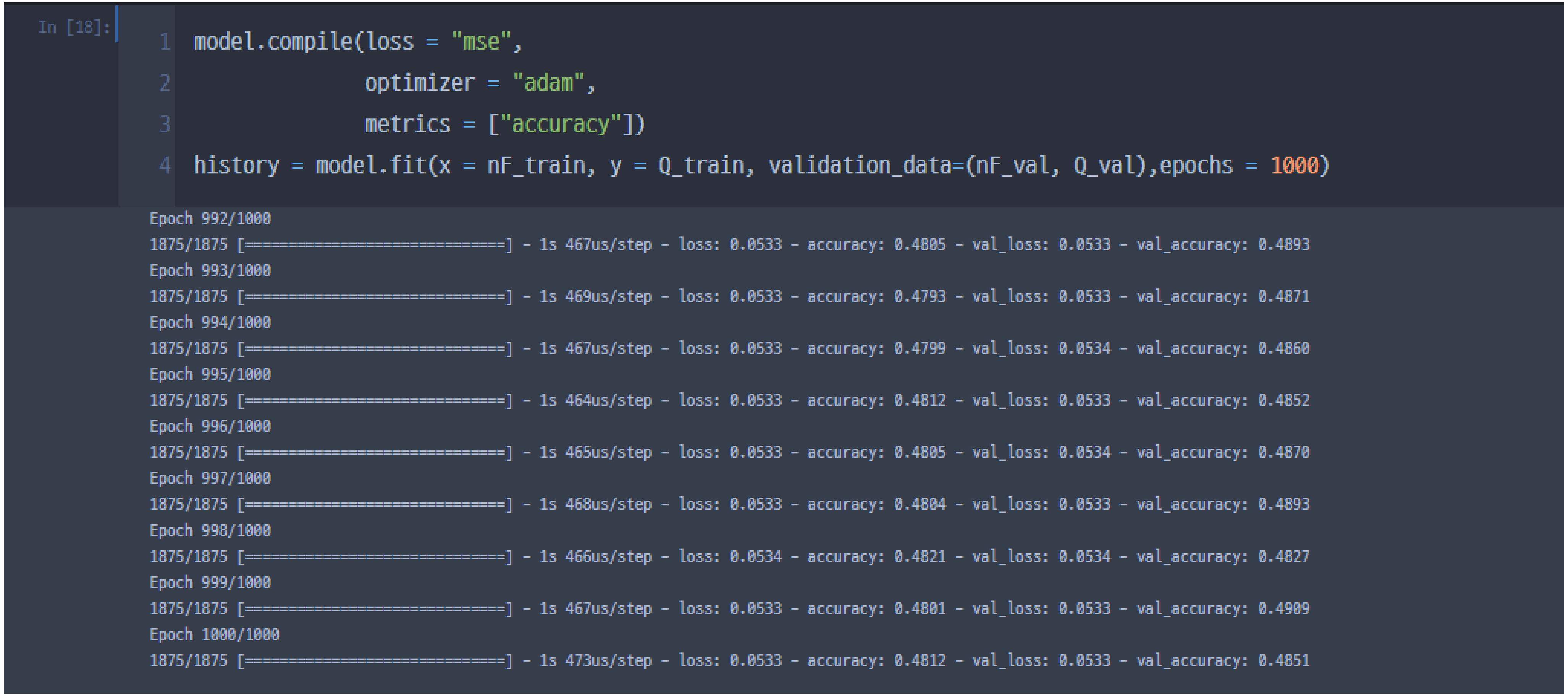
```
625/625 [==============================] - 0s 259us/step - loss: 0.0365 - accuracy: 0.6244
```

```
#F 학습 후 nF accuracy
result = model.evaluate(nF_test, Q_test)
```

```
625/625 [==============================] - 0s 254us/step - loss: 0.2105 - accuracy: 0.3359
```

nF_train : Epoch = 30 정도부터 48% 수렴

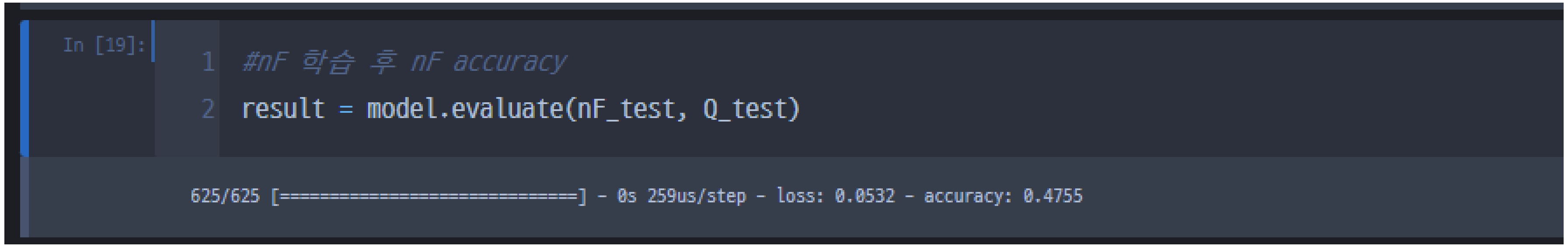
N=5, epoch = 1000→nF_train : 48%



```
model.compile(loss = "mse",
              optimizer = "adam",
              metrics = ["accuracy"])
history = model.fit(x = nF_train, y = Q_train, validation_data=(nF_val, Q_val),epochs = 1000)
```

```
Epoch 992/1000
1875/1875 [==============================] - 1s 467us/step - loss: 0.0533 - accuracy: 0.4805 - val_loss: 0.0533 - val_accuracy: 0.4893
Epoch 993/1000
1875/1875 [==============================] - 1s 469us/step - loss: 0.0533 - accuracy: 0.4793 - val_loss: 0.0533 - val_accuracy: 0.4871
Epoch 994/1000
1875/1875 [==============================] - 1s 467us/step - loss: 0.0533 - accuracy: 0.4799 - val_loss: 0.0534 - val_accuracy: 0.4860
Epoch 995/1000
1875/1875 [==============================] - 1s 464us/step - loss: 0.0533 - accuracy: 0.4812 - val_loss: 0.0533 - val_accuracy: 0.4852
Epoch 996/1000
1875/1875 [==============================] - 1s 465us/step - loss: 0.0533 - accuracy: 0.4805 - val_loss: 0.0534 - val_accuracy: 0.4870
Epoch 997/1000
1875/1875 [==============================] - 1s 468us/step - loss: 0.0533 - accuracy: 0.4804 - val_loss: 0.0533 - val_accuracy: 0.4893
Epoch 998/1000
1875/1875 [==============================] - 1s 466us/step - loss: 0.0534 - accuracy: 0.4821 - val_loss: 0.0534 - val_accuracy: 0.4827
Epoch 999/1000
1875/1875 [==============================] - 1s 467us/step - loss: 0.0533 - accuracy: 0.4801 - val_loss: 0.0533 - val_accuracy: 0.4909
Epoch 1000/1000
1875/1875 [==============================] - 1s 473us/step - loss: 0.0533 - accuracy: 0.4812 - val_loss: 0.0533 - val_accuracy: 0.4851
```

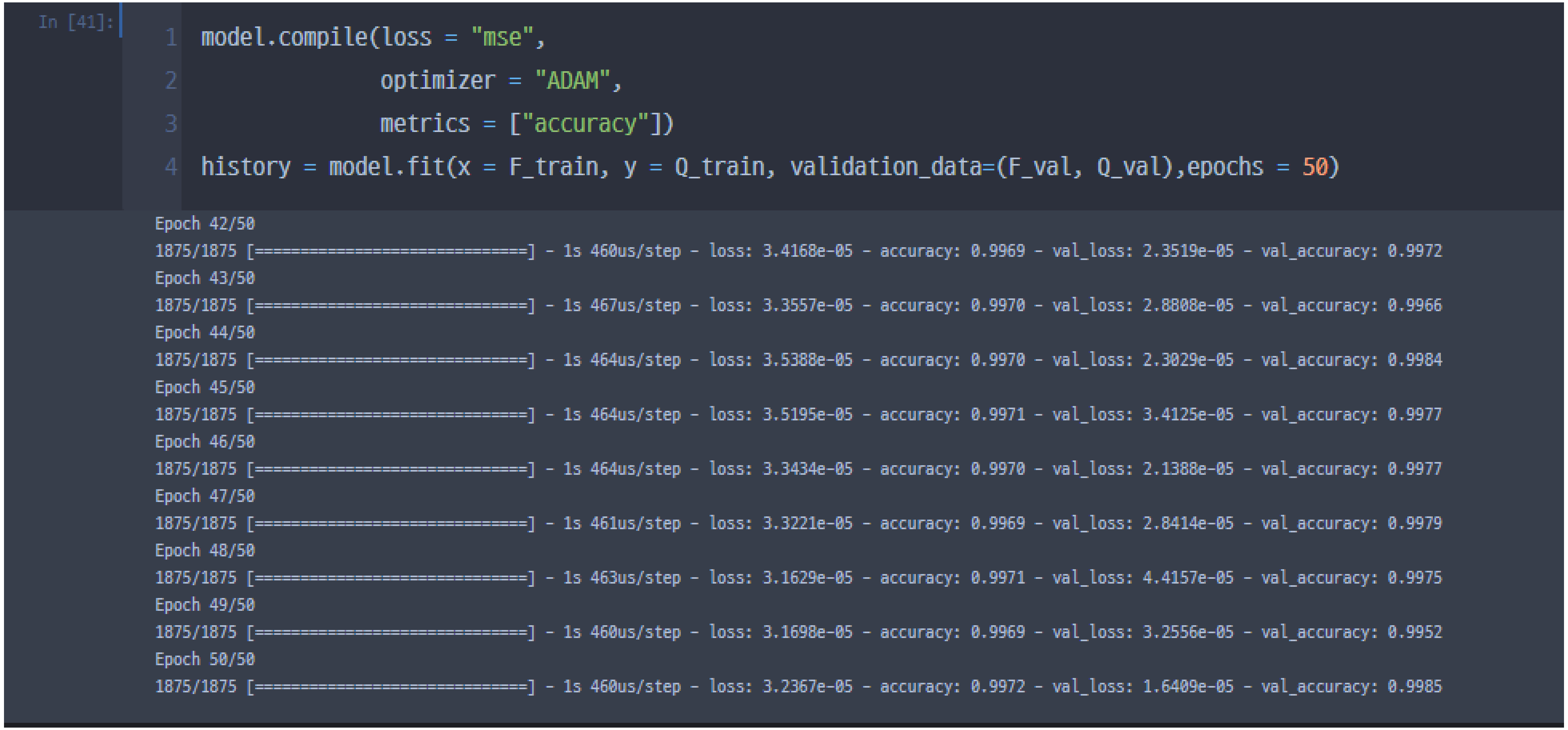
N=5, epoch = 1000→Val_accuracy : 48%



```
#nF 학습 후 nF accuracy
result = model.evaluate(nF_test, Q_test)
```

```
625/625 [==============================] - 0s 259us/step - loss: 0.0532 - accuracy: 0.4755
```

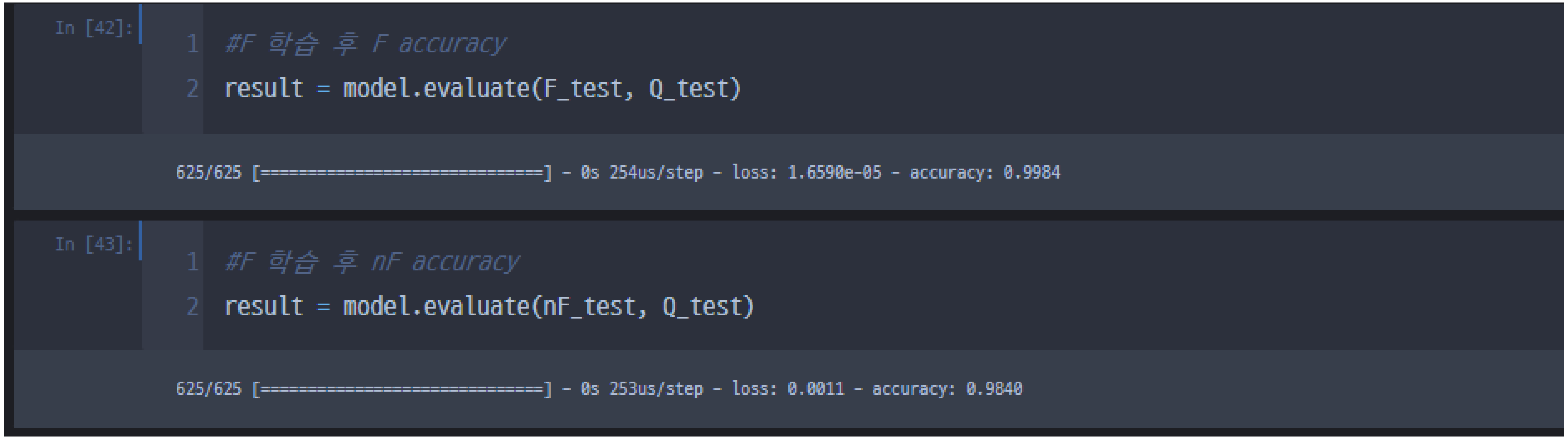
N=2, epoch = 50 → F_train : 99.8%



```
model.compile(loss = "mse",
              optimizer = "ADAM",
              metrics = ["accuracy"])
history = model.fit(x = F_train, y = Q_train, validation_data=(F_val, Q_val),epochs = 50)
```

```
Epoch 42/50
1875/1875 [==============================] - 1s 460us/step - loss: 3.4168e-05 - accuracy: 0.9969 - val_loss: 2.3519e-05 - val_accuracy: 0.9972
Epoch 43/50
1875/1875 [==============================] - 1s 467us/step - loss: 3.3557e-05 - accuracy: 0.9970 - val_loss: 2.8808e-05 - val_accuracy: 0.9966
Epoch 44/50
1875/1875 [==============================] - 1s 464us/step - loss: 3.5388e-05 - accuracy: 0.9970 - val_loss: 2.3029e-05 - val_accuracy: 0.9984
Epoch 45/50
1875/1875 [==============================] - 1s 464us/step - loss: 3.5195e-05 - accuracy: 0.9971 - val_loss: 3.4125e-05 - val_accuracy: 0.9977
Epoch 46/50
1875/1875 [==============================] - 1s 464us/step - loss: 3.3434e-05 - accuracy: 0.9970 - val_loss: 2.1388e-05 - val_accuracy: 0.9977
Epoch 47/50
1875/1875 [==============================] - 1s 461us/step - loss: 3.3221e-05 - accuracy: 0.9969 - val_loss: 2.8414e-05 - val_accuracy: 0.9979
Epoch 48/50
1875/1875 [==============================] - 1s 463us/step - loss: 3.1629e-05 - accuracy: 0.9971 - val_loss: 4.4157e-05 - val_accuracy: 0.9975
Epoch 49/50
1875/1875 [==============================] - 1s 460us/step - loss: 3.1698e-05 - accuracy: 0.9969 - val_loss: 3.2556e-05 - val_accuracy: 0.9952
Epoch 50/50
1875/1875 [==============================] - 1s 460us/step - loss: 3.2367e-05 - accuracy: 0.9972 - val_loss: 1.6409e-05 - val_accuracy: 0.9985
```

N=2, epoch = 50



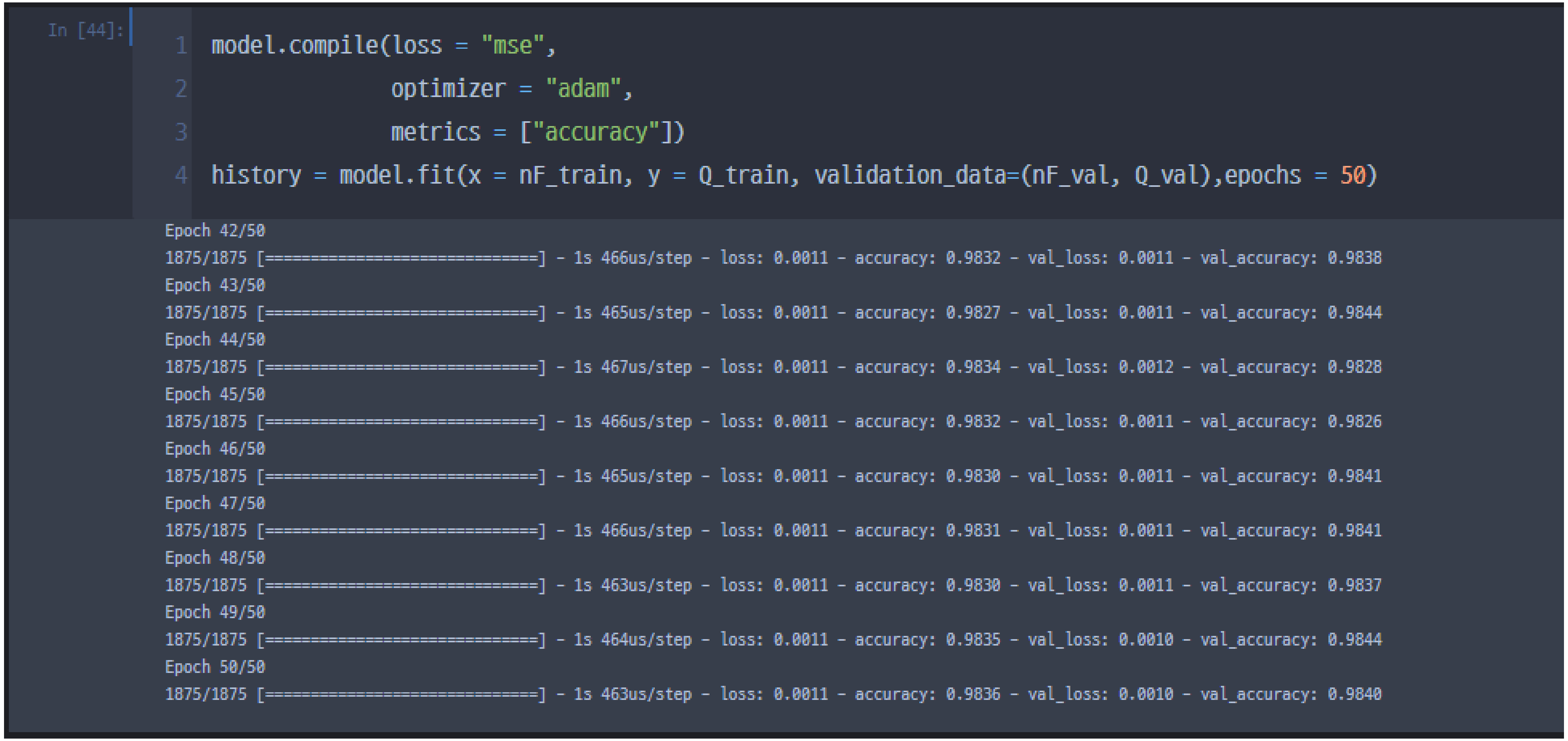
```
In [42]:
#F 학습 후 F accuracy
result = model.evaluate(F_test, Q_test)

625/625 [==============================] - 0s 254us/step - loss: 1.6590e-05 - accuracy: 0.9984
```

```
In [43]:
#F 학습 후 nF accuracy
result = model.evaluate(nF_test, Q_test)

625/625 [==============================] - 0s 253us/step - loss: 0.0011 - accuracy: 0.9840
```

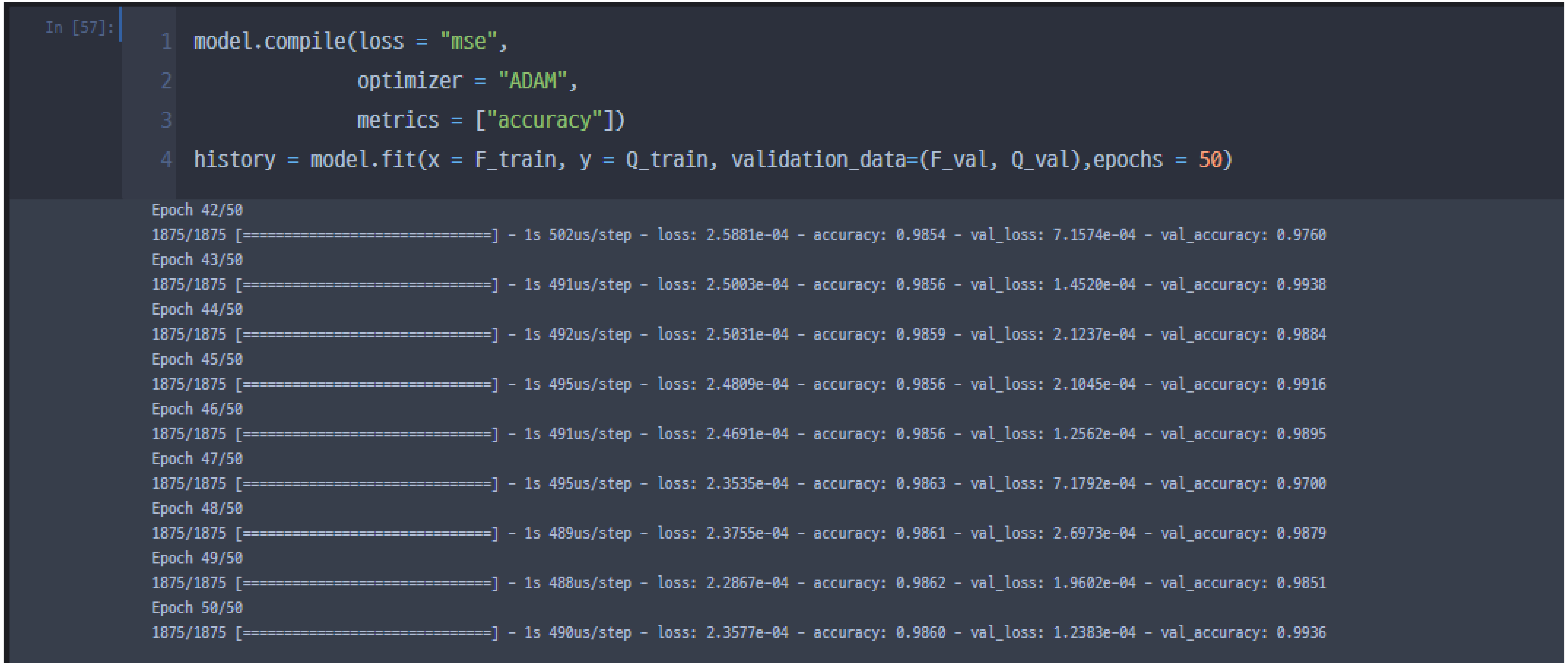
N=2, epoch = 50 → nF_train : 98%



```
In [44]:
model.compile(loss = "mse",
              optimizer = "adam",
              metrics = ["accuracy"])
history = model.fit(x = nF_train, y = Q_train, validation_data=(nF_val, Q_val),epochs = 50)
```

```
Epoch 42/50
1875/1875 [==============================] - 1s 466us/step - loss: 0.0011 - accuracy: 0.9832 - val_loss: 0.0011 - val_accuracy: 0.9838
Epoch 43/50
1875/1875 [==============================] - 1s 465us/step - loss: 0.0011 - accuracy: 0.9827 - val_loss: 0.0011 - val_accuracy: 0.9844
Epoch 44/50
1875/1875 [==============================] - 1s 467us/step - loss: 0.0011 - accuracy: 0.9834 - val_loss: 0.0012 - val_accuracy: 0.9828
Epoch 45/50
1875/1875 [==============================] - 1s 466us/step - loss: 0.0011 - accuracy: 0.9832 - val_loss: 0.0011 - val_accuracy: 0.9826
Epoch 46/50
1875/1875 [==============================] - 1s 465us/step - loss: 0.0011 - accuracy: 0.9830 - val_loss: 0.0011 - val_accuracy: 0.9841
Epoch 47/50
1875/1875 [==============================] - 1s 466us/step - loss: 0.0011 - accuracy: 0.9831 - val_loss: 0.0011 - val_accuracy: 0.9841
Epoch 48/50
1875/1875 [==============================] - 1s 463us/step - loss: 0.0011 - accuracy: 0.9830 - val_loss: 0.0011 - val_accuracy: 0.9837
Epoch 49/50
1875/1875 [==============================] - 1s 464us/step - loss: 0.0011 - accuracy: 0.9835 - val_loss: 0.0010 - val_accuracy: 0.9844
Epoch 50/50
1875/1875 [==============================] - 1s 463us/step - loss: 0.0011 - accuracy: 0.9836 - val_loss: 0.0010 - val_accuracy: 0.9840
```

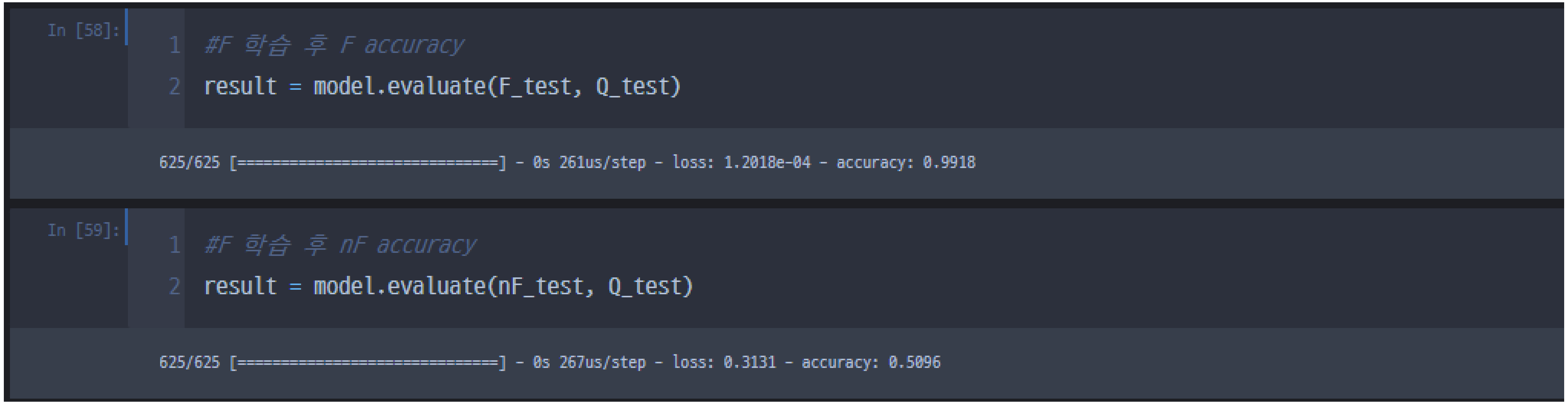
N=3, epoch =50 → F_train : 99.3%



```
model.compile(loss = "mse",
              optimizer = "ADAM",
              metrics = ["accuracy"])
history = model.fit(x = F_train, y = Q_train, validation_data=(F_val, Q_val),epochs = 50)
```

```
Epoch 42/50
1875/1875 [==============================] - 1s 502us/step - loss: 2.5881e-04 - accuracy: 0.9854 - val_loss: 7.1574e-04 - val_accuracy: 0.9760
Epoch 43/50
1875/1875 [==============================] - 1s 491us/step - loss: 2.5003e-04 - accuracy: 0.9856 - val_loss: 1.4520e-04 - val_accuracy: 0.9938
Epoch 44/50
1875/1875 [==============================] - 1s 492us/step - loss: 2.5031e-04 - accuracy: 0.9859 - val_loss: 2.1237e-04 - val_accuracy: 0.9884
Epoch 45/50
1875/1875 [==============================] - 1s 495us/step - loss: 2.4809e-04 - accuracy: 0.9856 - val_loss: 2.1045e-04 - val_accuracy: 0.9916
Epoch 46/50
1875/1875 [==============================] - 1s 491us/step - loss: 2.4691e-04 - accuracy: 0.9856 - val_loss: 1.2562e-04 - val_accuracy: 0.9895
Epoch 47/50
1875/1875 [==============================] - 1s 495us/step - loss: 2.3535e-04 - accuracy: 0.9863 - val_loss: 7.1792e-04 - val_accuracy: 0.9700
Epoch 48/50
1875/1875 [==============================] - 1s 489us/step - loss: 2.3755e-04 - accuracy: 0.9861 - val_loss: 2.6973e-04 - val_accuracy: 0.9879
Epoch 49/50
1875/1875 [==============================] - 1s 488us/step - loss: 2.2867e-04 - accuracy: 0.9862 - val_loss: 1.9602e-04 - val_accuracy: 0.9851
Epoch 50/50
1875/1875 [==============================] - 1s 490us/step - loss: 2.3577e-04 - accuracy: 0.9860 - val_loss: 1.2383e-04 - val_accuracy: 0.9936
```

N=3, epoch =50



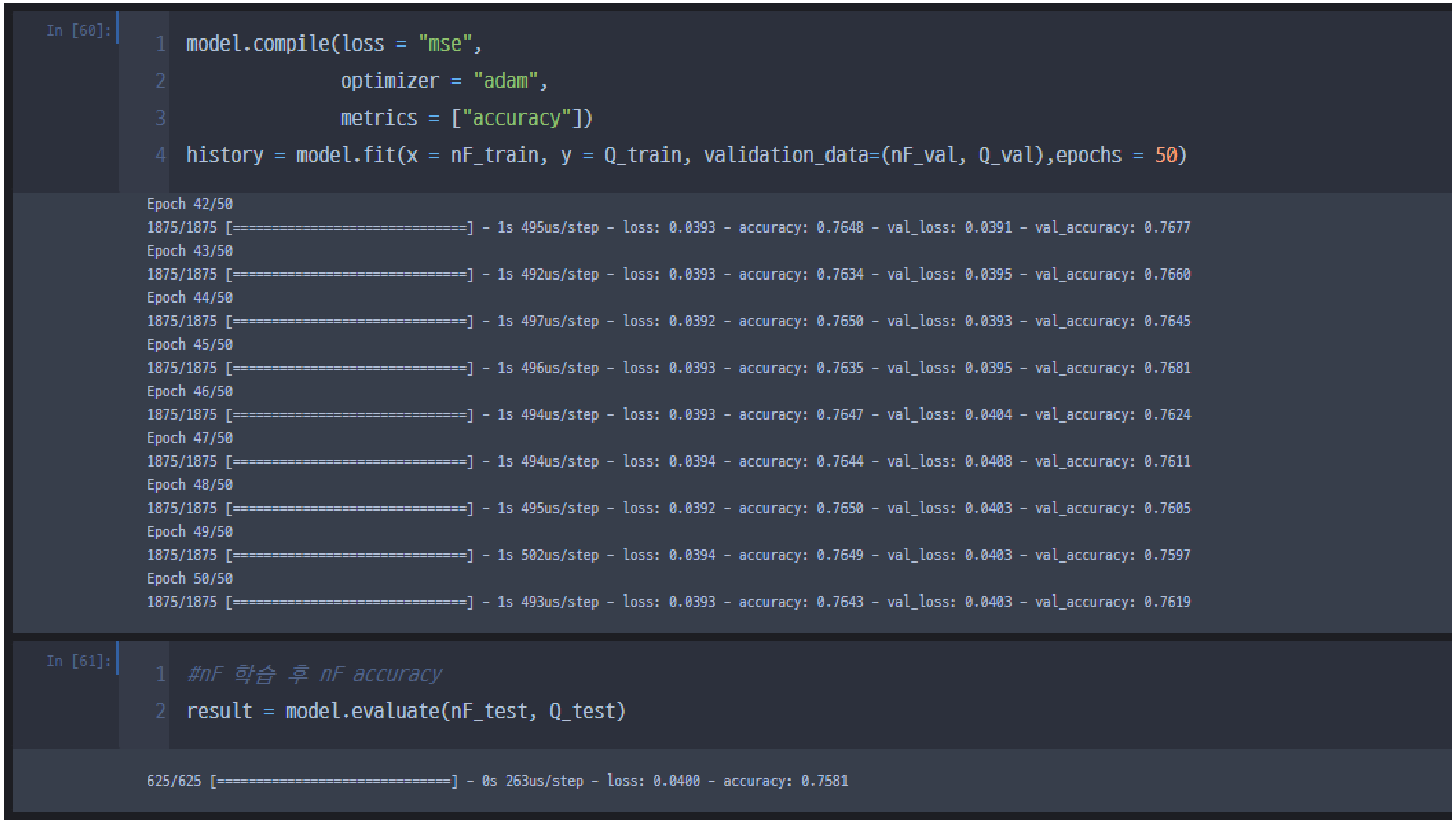
```
In [58]:
#F 학습 후 F accuracy
result = model.evaluate(F_test, Q_test)
```

```
625/625 [==============================] - 0s 261us/step - loss: 1.2018e-04 - accuracy: 0.9918
```

```
In [59]:
#F 학습 후 nF accuracy
result = model.evaluate(nF_test, Q_test)
```

```
625/625 [==============================] - 0s 267us/step - loss: 0.3131 - accuracy: 0.5096
```

N=3, epoch =50 → nF_train : 76.1%



In [60]:

```
model.compile(loss = "mse",
              optimizer = "adam",
              metrics = ["accuracy"])
history = model.fit(x = nF_train, y = Q_train, validation_data=(nF_val, Q_val),epochs = 50)
```

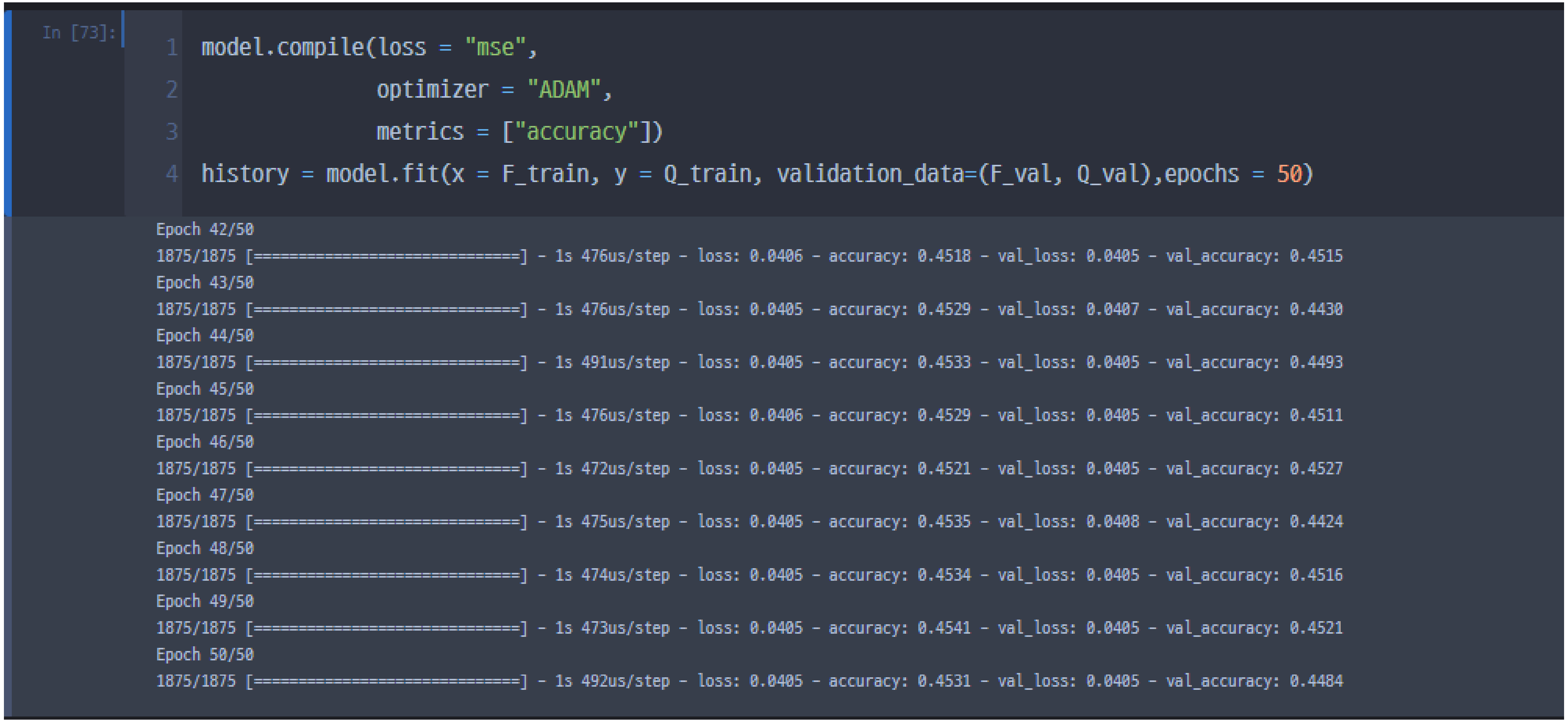
```
Epoch 42/50
1875/1875 [==============================] - 1s 495us/step - loss: 0.0393 - accuracy: 0.7648 - val_loss: 0.0391 - val_accuracy: 0.7677
Epoch 43/50
1875/1875 [==============================] - 1s 492us/step - loss: 0.0393 - accuracy: 0.7634 - val_loss: 0.0395 - val_accuracy: 0.7660
Epoch 44/50
1875/1875 [==============================] - 1s 497us/step - loss: 0.0392 - accuracy: 0.7650 - val_loss: 0.0393 - val_accuracy: 0.7645
Epoch 45/50
1875/1875 [==============================] - 1s 496us/step - loss: 0.0393 - accuracy: 0.7635 - val_loss: 0.0395 - val_accuracy: 0.7681
Epoch 46/50
1875/1875 [==============================] - 1s 494us/step - loss: 0.0393 - accuracy: 0.7647 - val_loss: 0.0404 - val_accuracy: 0.7624
Epoch 47/50
1875/1875 [==============================] - 1s 494us/step - loss: 0.0394 - accuracy: 0.7644 - val_loss: 0.0408 - val_accuracy: 0.7611
Epoch 48/50
1875/1875 [==============================] - 1s 495us/step - loss: 0.0392 - accuracy: 0.7650 - val_loss: 0.0403 - val_accuracy: 0.7605
Epoch 49/50
1875/1875 [==============================] - 1s 502us/step - loss: 0.0394 - accuracy: 0.7649 - val_loss: 0.0403 - val_accuracy: 0.7597
Epoch 50/50
1875/1875 [==============================] - 1s 493us/step - loss: 0.0393 - accuracy: 0.7643 - val_loss: 0.0403 - val_accuracy: 0.7619
```

In [61]:

```
#nF 학습 후 nF accuracy
result = model.evaluate(nF_test, Q_test)
```

```
625/625 [==============================] - 0s 263us/step - loss: 0.0400 - accuracy: 0.7581
```

N = 7, epoch = 50 → F_train : 44.8%

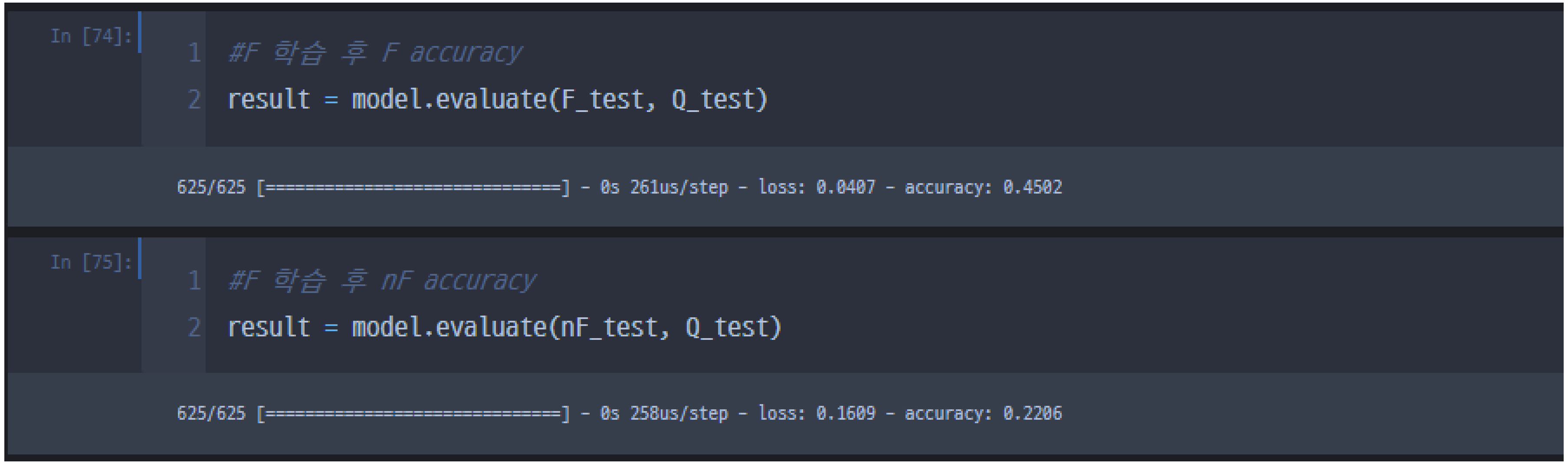


In [73]:

```
model.compile(loss = "mse",
              optimizer = "ADAM",
              metrics = ["accuracy"])
history = model.fit(x = F_train, y = Q_train, validation_data=(F_val, Q_val),epochs = 50)
```

```
Epoch 42/50
1875/1875 [==============================] - 1s 476us/step - loss: 0.0406 - accuracy: 0.4518 - val_loss: 0.0405 - val_accuracy: 0.4515
Epoch 43/50
1875/1875 [==============================] - 1s 476us/step - loss: 0.0405 - accuracy: 0.4529 - val_loss: 0.0407 - val_accuracy: 0.4430
Epoch 44/50
1875/1875 [==============================] - 1s 491us/step - loss: 0.0405 - accuracy: 0.4533 - val_loss: 0.0405 - val_accuracy: 0.4493
Epoch 45/50
1875/1875 [==============================] - 1s 476us/step - loss: 0.0406 - accuracy: 0.4529 - val_loss: 0.0405 - val_accuracy: 0.4511
Epoch 46/50
1875/1875 [==============================] - 1s 472us/step - loss: 0.0405 - accuracy: 0.4521 - val_loss: 0.0405 - val_accuracy: 0.4527
Epoch 47/50
1875/1875 [==============================] - 1s 475us/step - loss: 0.0405 - accuracy: 0.4535 - val_loss: 0.0408 - val_accuracy: 0.4424
Epoch 48/50
1875/1875 [==============================] - 1s 474us/step - loss: 0.0405 - accuracy: 0.4534 - val_loss: 0.0405 - val_accuracy: 0.4516
Epoch 49/50
1875/1875 [==============================] - 1s 473us/step - loss: 0.0405 - accuracy: 0.4541 - val_loss: 0.0405 - val_accuracy: 0.4521
Epoch 50/50
1875/1875 [==============================] - 1s 492us/step - loss: 0.0405 - accuracy: 0.4531 - val_loss: 0.0405 - val_accuracy: 0.4484
```

N = 7, epoch = 50



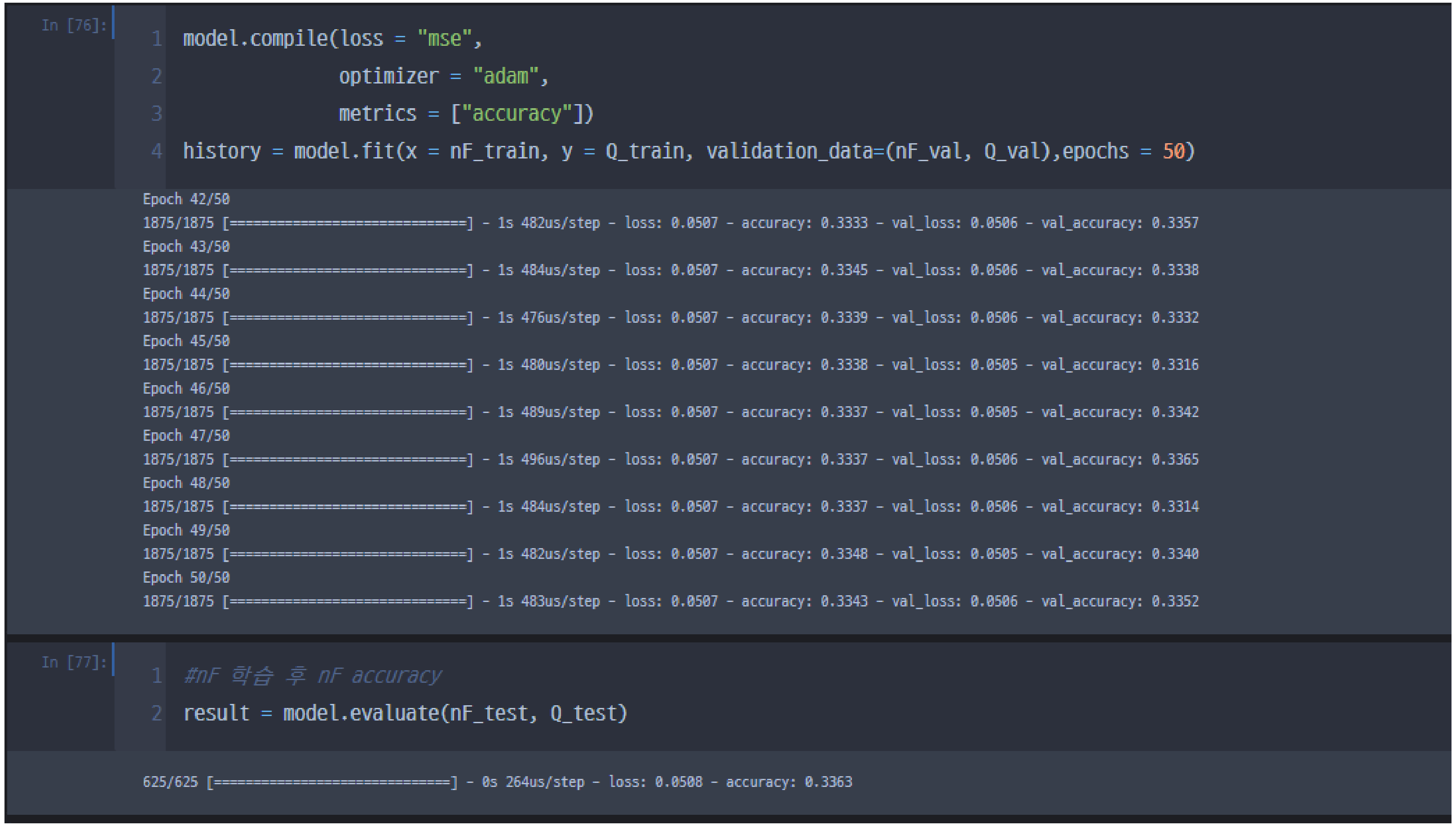
```
In [74]:
#F 학습 후 F accuracy
result = model.evaluate(F_test, Q_test)

625/625 [==============================] - 0s 261us/step - loss: 0.0407 - accuracy: 0.4502
```

```
In [75]:
#F 학습 후 nF accuracy
result = model.evaluate(nF_test, Q_test)

625/625 [==============================] - 0s 258us/step - loss: 0.1609 - accuracy: 0.2206
```

N = 7, epoch = 50 → nF_train : 33.5%



In [76]:

```
model.compile(loss = "mse",
              optimizer = "adam",
              metrics = ["accuracy"])
history = model.fit(x = nF_train, y = Q_train, validation_data=(nF_val, Q_val),epochs = 50)
```

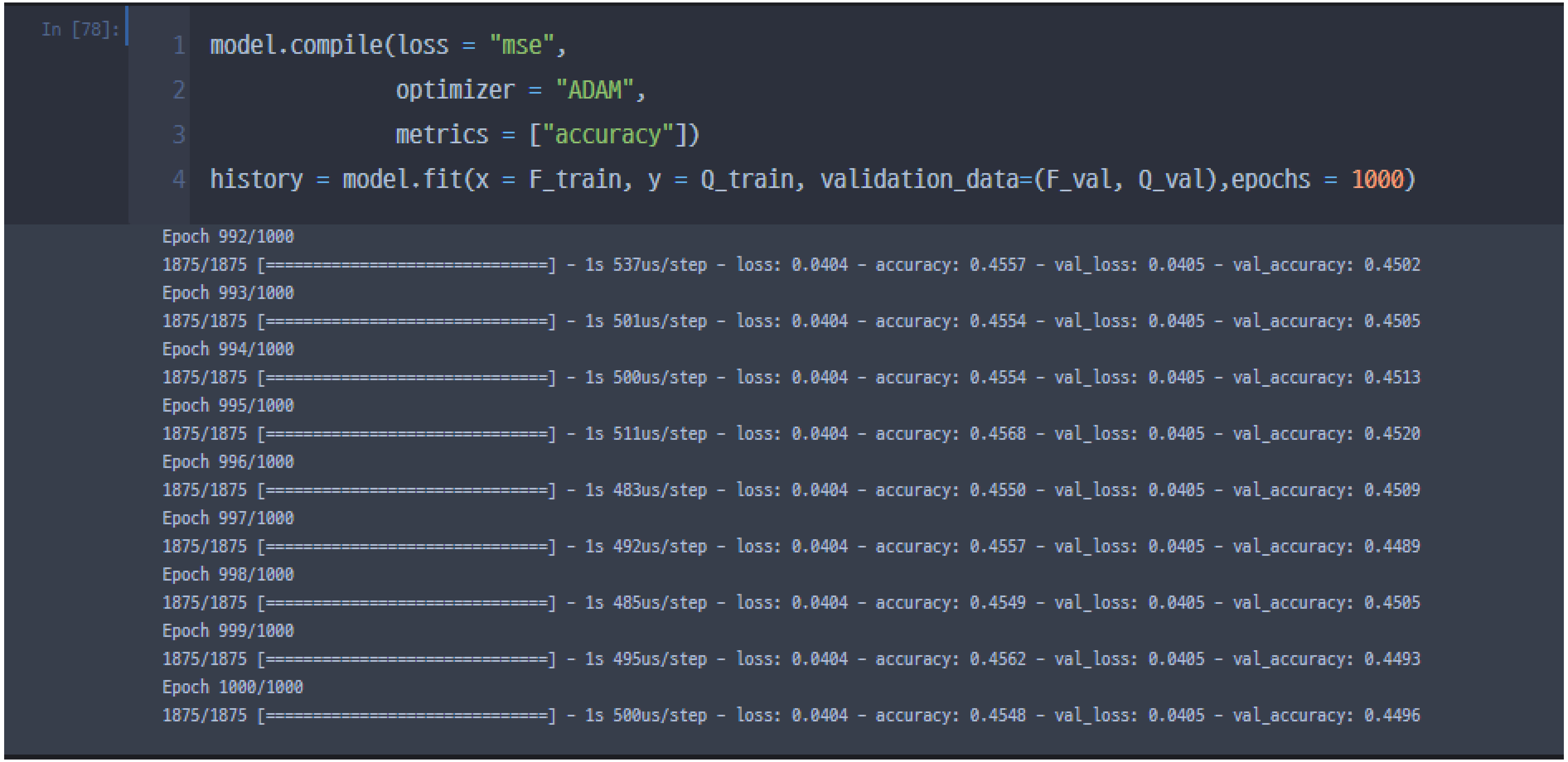
```
Epoch 42/50
1875/1875 [==============================] - 1s 482us/step - loss: 0.0507 - accuracy: 0.3333 - val_loss: 0.0506 - val_accuracy: 0.3357
Epoch 43/50
1875/1875 [==============================] - 1s 484us/step - loss: 0.0507 - accuracy: 0.3345 - val_loss: 0.0506 - val_accuracy: 0.3338
Epoch 44/50
1875/1875 [==============================] - 1s 476us/step - loss: 0.0507 - accuracy: 0.3339 - val_loss: 0.0506 - val_accuracy: 0.3332
Epoch 45/50
1875/1875 [==============================] - 1s 480us/step - loss: 0.0507 - accuracy: 0.3338 - val_loss: 0.0505 - val_accuracy: 0.3316
Epoch 46/50
1875/1875 [==============================] - 1s 489us/step - loss: 0.0507 - accuracy: 0.3337 - val_loss: 0.0505 - val_accuracy: 0.3342
Epoch 47/50
1875/1875 [==============================] - 1s 496us/step - loss: 0.0507 - accuracy: 0.3337 - val_loss: 0.0506 - val_accuracy: 0.3365
Epoch 48/50
1875/1875 [==============================] - 1s 484us/step - loss: 0.0507 - accuracy: 0.3337 - val_loss: 0.0506 - val_accuracy: 0.3314
Epoch 49/50
1875/1875 [==============================] - 1s 482us/step - loss: 0.0507 - accuracy: 0.3348 - val_loss: 0.0505 - val_accuracy: 0.3340
Epoch 50/50
1875/1875 [==============================] - 1s 483us/step - loss: 0.0507 - accuracy: 0.3343 - val_loss: 0.0506 - val_accuracy: 0.3352
```

In [77]:

```
#nF 학습 후 nF accuracy
result = model.evaluate(nF_test, Q_test)
```

```
625/625 [==============================] - 0s 264us/step - loss: 0.0508 - accuracy: 0.3363
```

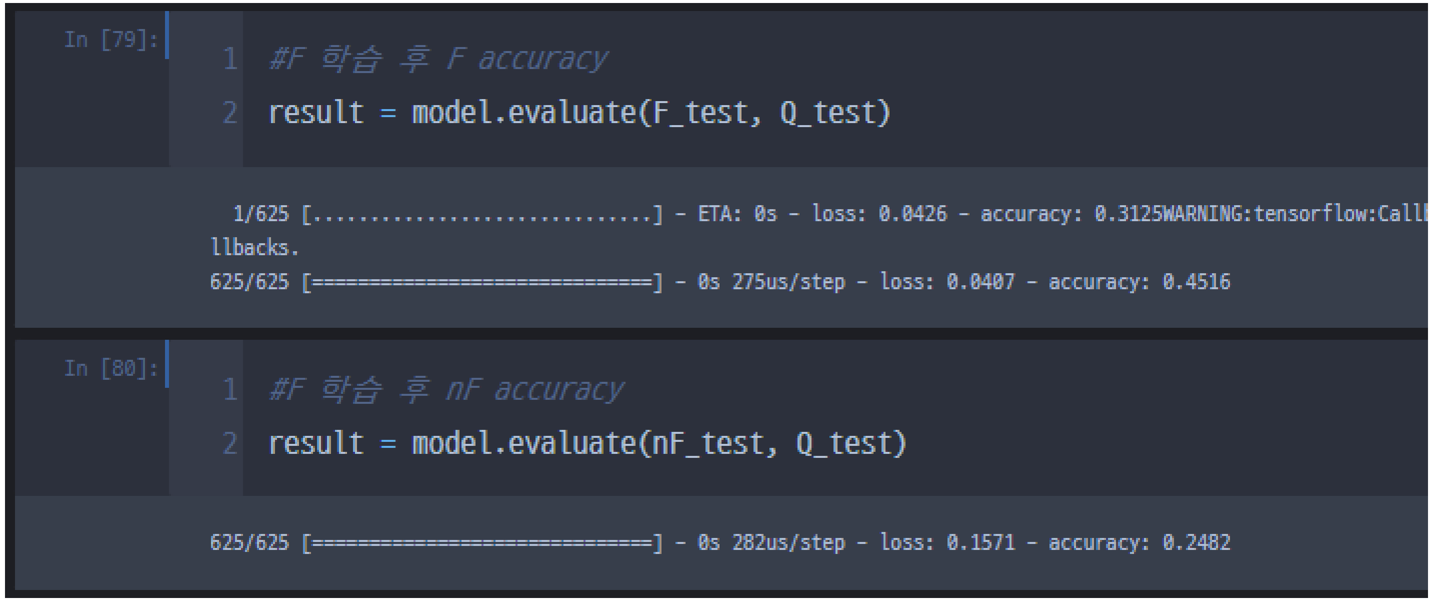
N = 7, epoch = 1000 → F_train : 44.9%



```python
model.compile(loss = "mse",
              optimizer = "ADAM",
              metrics = ["accuracy"])
history = model.fit(x = F_train, y = Q_train, validation_data=(F_val, Q_val),epochs = 1000)
```

```
Epoch 992/1000
1875/1875 [==============================] - 1s 537us/step - loss: 0.0404 - accuracy: 0.4557 - val_loss: 0.0405 - val_accuracy: 0.4502
Epoch 993/1000
1875/1875 [==============================] - 1s 501us/step - loss: 0.0404 - accuracy: 0.4554 - val_loss: 0.0405 - val_accuracy: 0.4505
Epoch 994/1000
1875/1875 [==============================] - 1s 500us/step - loss: 0.0404 - accuracy: 0.4554 - val_loss: 0.0405 - val_accuracy: 0.4513
Epoch 995/1000
1875/1875 [==============================] - 1s 511us/step - loss: 0.0404 - accuracy: 0.4568 - val_loss: 0.0405 - val_accuracy: 0.4520
Epoch 996/1000
1875/1875 [==============================] - 1s 483us/step - loss: 0.0404 - accuracy: 0.4550 - val_loss: 0.0405 - val_accuracy: 0.4509
Epoch 997/1000
1875/1875 [==============================] - 1s 492us/step - loss: 0.0404 - accuracy: 0.4557 - val_loss: 0.0405 - val_accuracy: 0.4489
Epoch 998/1000
1875/1875 [==============================] - 1s 485us/step - loss: 0.0404 - accuracy: 0.4549 - val_loss: 0.0405 - val_accuracy: 0.4505
Epoch 999/1000
1875/1875 [==============================] - 1s 495us/step - loss: 0.0404 - accuracy: 0.4562 - val_loss: 0.0405 - val_accuracy: 0.4493
Epoch 1000/1000
1875/1875 [==============================] - 1s 500us/step - loss: 0.0404 - accuracy: 0.4548 - val_loss: 0.0405 - val_accuracy: 0.4496
```

N = 7, epoch = 1000



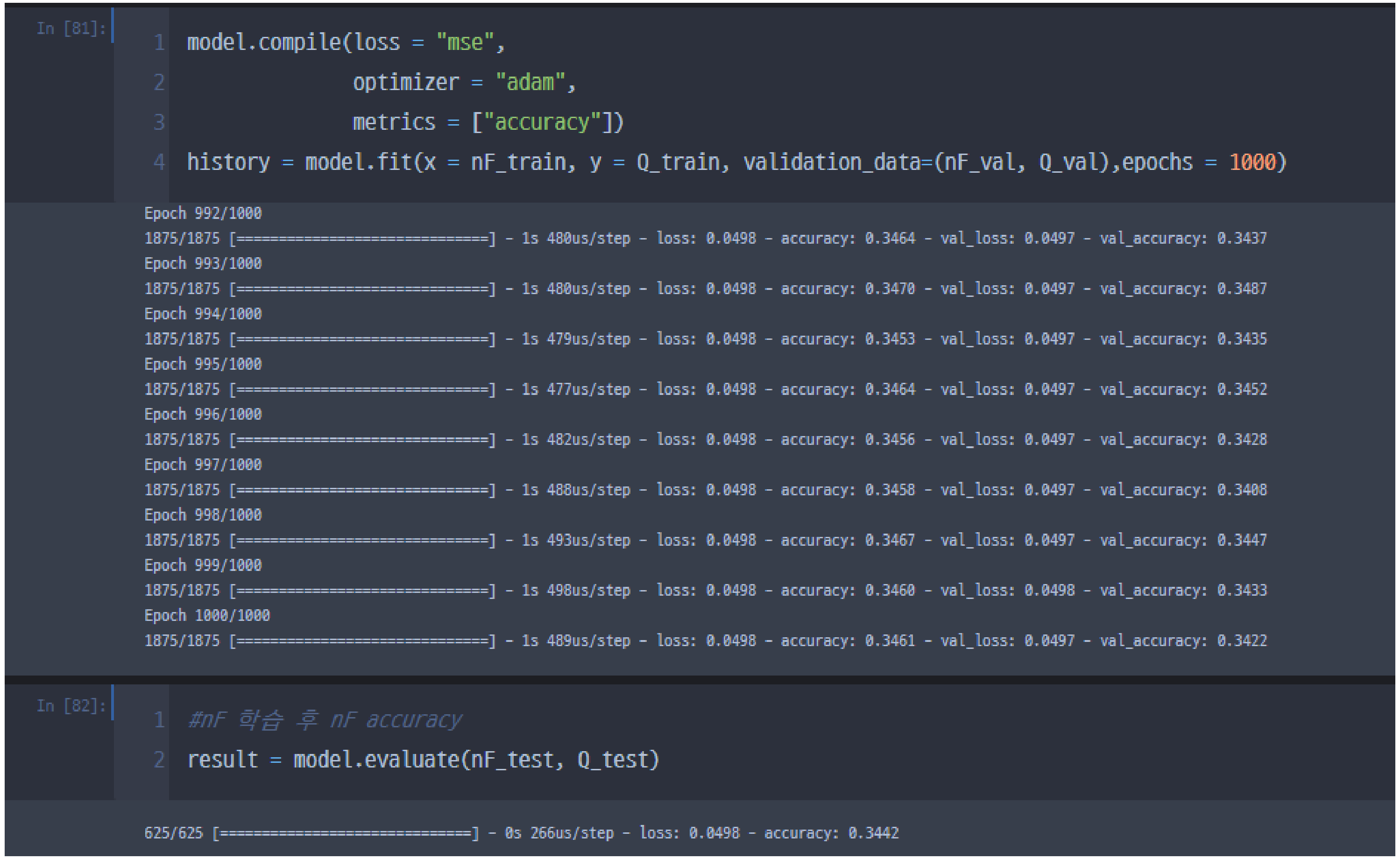
In [79]:
```
#F 학습 후 F accuracy
result = model.evaluate(F_test, Q_test)
```

```
  1/625 [..............................] - ETA: 0s - loss: 0.0426 - accuracy: 0.3125WARNING:tensorflow:Call
llbacks.
625/625 [==============================] - 0s 275us/step - loss: 0.0407 - accuracy: 0.4516
```

In [80]:
```
#F 학습 후 nF accuracy
result = model.evaluate(nF_test, Q_test)
```

```
625/625 [==============================] - 0s 282us/step - loss: 0.1571 - accuracy: 0.2482
```

N = 7, epoch = 1000 → nF_train : 34.2%



```
model.compile(loss = "mse",
              optimizer = "adam",
              metrics = ["accuracy"])
history = model.fit(x = nF_train, y = Q_train, validation_data=(nF_val, Q_val),epochs = 1000)
```

```
Epoch 992/1000
1875/1875 [==============================] - 1s 480us/step - loss: 0.0498 - accuracy: 0.3464 - val_loss: 0.0497 - val_accuracy: 0.3437
Epoch 993/1000
1875/1875 [==============================] - 1s 480us/step - loss: 0.0498 - accuracy: 0.3470 - val_loss: 0.0497 - val_accuracy: 0.3487
Epoch 994/1000
1875/1875 [==============================] - 1s 479us/step - loss: 0.0498 - accuracy: 0.3453 - val_loss: 0.0497 - val_accuracy: 0.3435
Epoch 995/1000
1875/1875 [==============================] - 1s 477us/step - loss: 0.0498 - accuracy: 0.3464 - val_loss: 0.0497 - val_accuracy: 0.3452
Epoch 996/1000
1875/1875 [==============================] - 1s 482us/step - loss: 0.0498 - accuracy: 0.3456 - val_loss: 0.0497 - val_accuracy: 0.3428
Epoch 997/1000
1875/1875 [==============================] - 1s 488us/step - loss: 0.0498 - accuracy: 0.3458 - val_loss: 0.0497 - val_accuracy: 0.3408
Epoch 998/1000
1875/1875 [==============================] - 1s 493us/step - loss: 0.0498 - accuracy: 0.3467 - val_loss: 0.0497 - val_accuracy: 0.3447
Epoch 999/1000
1875/1875 [==============================] - 1s 498us/step - loss: 0.0498 - accuracy: 0.3460 - val_loss: 0.0498 - val_accuracy: 0.3433
Epoch 1000/1000
1875/1875 [==============================] - 1s 489us/step - loss: 0.0498 - accuracy: 0.3461 - val_loss: 0.0497 - val_accuracy: 0.3422
```

```
#nF 학습 후 nF accuracy
result = model.evaluate(nF_test, Q_test)
```

```
625/625 [==============================] - 0s 266us/step - loss: 0.0498 - accuracy: 0.3442
```