In [1]:

```
import tensorflow as tf
import numpy as np
from tensorflow import keras
from keras.utils import np_utils
```

In [2]:

```
import os
import matplotlib.pyplot as plt
import random
```

In [3]:

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
```

In [4]:

```
from pandas.io.parsers import read_csv
```

model = tf.compat.v1.global_variables_initializer()

In [5]:

```
model = tf.compat.v1.global_variables_initializer()
```

In [6]:

```
data = read_csv('InDummy.csv',sep=',')
```

In [7]:

```
xy = np.array(data)
```

In [8]:

```
print(xy)
```

```
[['강중위' 1 '7:15:58' '2021.3.21' 0.30275463 44276]
 ['강중위' 1 '6:09:33' '2021.3.22' 0.256631944 44277]
 ['강중위' 1 '6:48:10' '2021.3.23' 0.283449074 44278]
 ['강중위' 1 '7:50:46' '2021.3.24' 0.326921296 44279]
 ['강중위' 1 '6:44:25' '2021.3.25' 0.280844907 44280]
 ['강중위' 1 '7:21:01' '2021.3.26' 0.306261574 44281]
 ['강중위' 1 '6:25:55' '2021.3.27' 0.267997685 44282]
 ['강중위' 1 '6:17:57' '2021.3.28' 0.262465278 44283]
 ['강중위' 1 '6:18:14' '2021.3.29' 0.262662037 44284]
 ['강중위' 1 '6:23:06' '2021.3.30' 0.266041667 44285]
 ['강중위' 1 '6:40:53' '2021.3.31' 0.278391204 44286]
 ['강중위' 1 '6:56:28' '2021.4.1' 0.289212963 44287]
 ['강중위' 1 '7:09:22' '2021.4.2' 0.298171296 44288]
 ['강중위' 1 '7:45:19' '2021.4.3' 0.323136574 44289]
 ['강중위' 1 '7:28:16' '2021.4.4' 0.311296296 44290]
 ['강중위' 1 '6:17:45' '2021.4.5' 0.262326389 44291]
 ['강중위' 1 '6:04:26' '2021.4.6' 0.253078704 44292]
 ['강중위' 1 '7:07:38' '2021.4.7' 0.296967593 44293]
 ['강중위' 1 '6:13:01' '2021.4.8' 0.259039352 44294]
 ['강중위' 1 '7:16:47' '2021.4.9' 0.303321759 44295]
 ['강중위' 1 '6:43:20' '2021.4.10' 0.280092593 44296]
 ['강중위' 1 '7:59:59' '2021.4.11' 0.333321759 44297]
 ['강중위' 1 '7:25:43' '2021.4.12' 0.309525463 44298]
 ['강중위' 1 '7:56:57' '2021.4.13' 0.331215278 44299]
 ['강중위' 1 '7:22:22' '2021.4.14' 0.307199074 44300]
 ['강중위' 1 '7:22:39' '2021.4.15' 0.307395833 44301]
 ['강중위' 1 '6:33:35' '2021.4.16' 0.273321759 44302]
 ['강중위' 1 '6:48:35' '2021.4.17' 0.283738426 44303]
 ['강중위' 1 '7:02:11' '2021.4.18' 0.29318287 44304]
 ['강중위' 1 '7:08:03' '2021.4.19' 0.297256944 44305]
 ['강중위' 1 '6:37:53' '2021.4.20' 0.27630787 44306]
 ['강중위' 1 '7:55:56' '2021.4.21' 0.330509259 44307]
 ['강중위' 1 '7:15:04' '2021.4.22' 0.30212963 44308]
 ['강중위' 1 '7:42:11' '2021.4.23' 0.320960648 44309]
 ['강중위' 1 '7:05:29' '2021.4.24' 0.295474537 44310]
 ['강중위' 1 '6:26:10' '2021.4.25' 0.268171296 44311]
 ['강중위' 1 '7:45:49' '2021.4.26' 0.323483796 44312]
 ['강중위' 1 '6:50:29' '2021.4.27' 0.28505787 44313]
 ['강중위' 1 '6:16:53' '2021.4.28' 0.261724537 44314]
 ['강중위' 1 '7:49:22' '2021.4.29' 0.325949074 44315]
 ['강중위' 1 '7:49:00' '2021.4.30' 0.325694444 44316]
 ['강중위' 1 '6:49:48' '2021.5.1' 0.284583333 44317]
 ['강중위' 1 '6:56:49' '2021.5.2' 0.289456019 44318]
 ['강중위' 1 '6:02:14' '2021.5.3' 0.251550926 44319]
 ['강중위' 1 '7:12:07' '2021.5.4' 0.300081019 44320]
 ['강중위' 1 '6:20:10' '2021.5.5' 0.26400463 44321]
 ['강중위' 1 '6:55:16' '2021.5.6' 0.28837963 44322]
 ['강중위' 1 '6:33:52' '2021.5.7' 0.273518519 44323]
 ['강중위' 1 '7:36:38' '2021.5.8' 0.317106481 44324]]
```

In [9]:

```
x_data = xy[:, 5]
```

In [10]:

```
x_data
```

Out[10]:

```
array([44276, 44277, 44278, 44279, 44280, 44281, 44282, 44283, 44284,
       44285, 44286, 44287, 44288, 44289, 44290, 44291, 44292, 44293,
       44294, 44295, 44296, 44297, 44298, 44299, 44300, 44301, 44302,
       44303, 44304, 44305, 44306, 44307, 44308, 44309, 44310, 44311,
       44312, 44313, 44314, 44315, 44316, 44317, 44318, 44319, 44320,
       44321, 44322, 44323, 44324], dtype=object)
```

In [11]:

```
y_data = xy[:, 4]
```

In [12]:

```
y_data
```

Out[12]:

```
array([0.30275463, 0.256631944, 0.283449074, 0.326921296, 0.280844907,
       0.306261574, 0.267997685, 0.262465278, 0.262662037, 0.26604166
7,
       0.278391204, 0.289212963, 0.298171296, 0.323136574, 0.31129629
6,
       0.262326389, 0.253078704, 0.296967593, 0.259039352, 0.30332175
9,
       0.280092593, 0.333321759, 0.309525463, 0.331215278, 0.30719907
4,
       0.307395833, 0.273321759, 0.283738426, 0.29318287, 0.297256944,
       0.27630787, 0.330509259, 0.30212963, 0.320960648, 0.295474537,
       0.268171296, 0.323483796, 0.28505787, 0.261724537, 0.325949074,
       0.325694444, 0.284583333, 0.289456019, 0.251550926, 0.30008101
9,
       0.26400463, 0.28837963, 0.273518519, 0.317106481], dtype=objec
t)
```

In [45]:

```
X = tf.constant(x_data)
```

```
---------------------------------------------------------------------
------
ValueError                                Traceback (most recent call
last)
<ipython-input-45-254479e71c7f> in <module>
----> 1 X = tf.constant(x_data)

~/Google 드라이브(kyc3492@hanyang.ac.kr)/공부/GateDummy/GM/lib/python3.
8/site-packages/tensorflow/python/framework/constant_op.py in constan
t(value, dtype, shape, name)
    262     ValueError: if called on a symbolic tensor.
    263   """
--> 264   return _constant_impl(value, dtype, shape, name, verify_sha
pe=False,
    265                         allow_broadcast=True)
    266

~/Google 드라이브(kyc3492@hanyang.ac.kr)/공부/GateDummy/GM/lib/python3.
8/site-packages/tensorflow/python/framework/constant_op.py in _consta
```

In [18]:

```
Y = tf.constant(y_data)
```

In [ ]:

```
@tf.function
def forward():
    return X + Y
```

In [15]:

```
print(forward())
```

```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
 last)
<ipython-input-15-8682086ec0e2> in <module>
----> 1 print(forward())

NameError: name 'forward' is not defined
```

In [40]:

```
for step in range(10001):
    time = forward()
    if step % 500 == 0:
        print(time)
```

```
tf.Tensor(
[b'\xea\xb0\x95\xec\xa4\x91\xec\x9c\x84\xec\x98\xa4\xec\xa0\x84 7:42:
41'
 b'\xea\xb0\x95\xec\xa4\x91\xec\x9c\x84\xec\x98\xa4\xec\xa0\x84 6:14:
04'
 b'\xea\xb0\x95\xec\xa4\x91\xec\x9c\x84\xec\x98\xa4\xec\xa0\x84 6:23:
28'
 b'\xea\xb0\x95\xec\xa4\x91\xec\x9c\x84\xec\x98\xa4\xec\xa0\x84 6:56:
01'
 b'\xea\xb0\x95\xec\xa4\x91\xec\x9c\x84\xec\x98\xa4\xec\xa0\x84 6:20:
57'
 b'\xea\xb0\x95\xec\xa4\x91\xec\x9c\x84\xec\x98\xa4\xec\xa0\x84 7:20:
56'
 b'\xea\xb0\x95\xec\xa4\x91\xec\x9c\x84\xec\x98\xa4\xec\xa0\x84 6:33:
16'
 b'\xea\xb0\x95\xec\xa4\x91\xec\x9c\x84\xec\x98\xa4\xec\xa0\x84 6:45:
49'
 b'\xea\xb0\x95\xec\xa4\x91\xec\x9c\x84\xec\x98\xa4\xec\xa0\x84 7:50:
26'
```

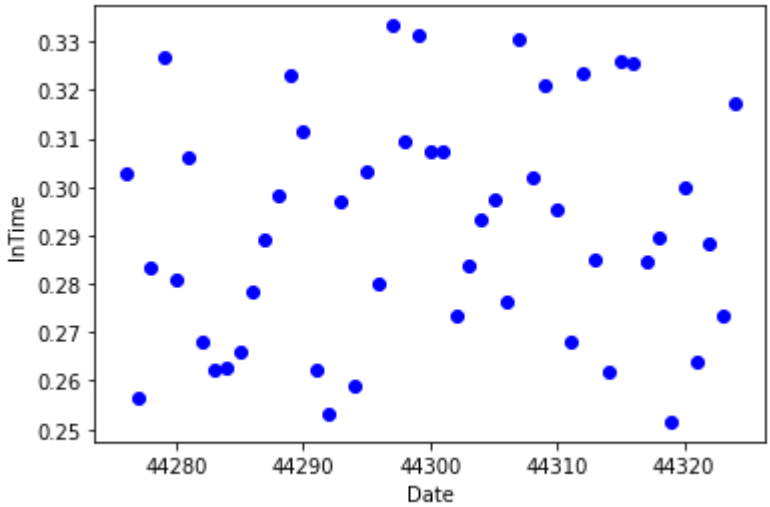
In [20]:

```
x = tf.reduce_mean(y_data)
```

```
-> 2370       gen_math_ops.mean(
   2371           input_tensor, _ReductionDims(input_tensor, axis), k
eepdims,
   2372           name=name))

~/Google 드라이브(kyc3492@hanyang.ac.kr)/공부/GateDummy/GM/lib/python3.
8/site-packages/tensorflow/python/ops/gen_math_ops.py in mean(input,
 axis, keep_dims, name)
   5766       return _result
   5767     except _core._NotOkStatusException as e:
-> 5768       _ops.raise_from_not_ok_status(e, name)
   5769     except _core._FallbackException:
   5770       pass

~/Google 드라이브(kyc3492@hanyang.ac.kr)/공부/GateDummy/GM/lib/python3.
8/site-packages/tensorflow/python/framework/ops.py in raise_from_not_
ok_status(e, name)
   6860   message = e.message + (" name: " + name if name is not None
else "")
   6861   # pylint: disable=protected-access
```

In [52]:

```
plt.plot(x_data, y_data, 'bo')
plt.xlabel('Date')
plt.ylabel('InTime')
plt.show()
```



In [54]:

```
x_bar = sum(x_data) / len(x_data)
y_bar = sum(y_data) / len(y_data)
```

In [55]:

```
a = sum([(y - y_bar) * (x - x_bar) for y, x in list(zip(y_data, x_data))])
a /= sum([(x - x_bar) ** 2 for x in x_data])
b = y_bar - a * x_bar
```

In [56]:

```
print('a:', a , 'b:', b)
```

```
a: 0.000186102845102041 b: -7.952695512734703
```

In [ ]:

```
# 그래프를 그리기 위해 회귀선의 x, y 데이터를 구합니다.
line_x = np.arange(min(x_data), max(x_data), 0.01)
line_y = a * line_x + b

# 붉은색 실선
plt.show()
```

In [13]:

```
a = tf.Variable(random.random())
b = tf.Variable(random.random())
```

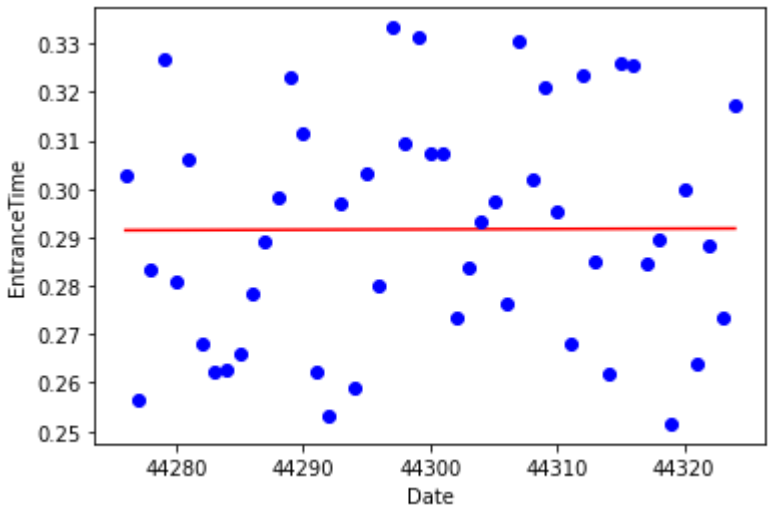
In [14]:

```
# 잔차의 제곱의 평균을 반환하는 함수입니다.

def compute_loss():
  y_pred = a * x_data + b
  loss = tf.reduce_mean((y_data - y_pred) ** 2)
  return loss

optimizer = tf.optimizers.Adam(lr=0.07)

for i in range(1000):
  # 잔차의 제곱의 평균을 최소화(minimize) 합니다.
  optimizer.minimize(compute_loss, var_list=[a,b])

  if i % 100 == 0:
    print(i, 'a:', a.numpy(), 'b:', b.numpy(), 'loss:', compute_loss().numpy())

line_x = np.arange(min(x_data), max(x_data), 0.01)
line_y = a * line_x + b
```

```
0 a: 0.18963021 b: 0.13807535 loss: 70567816.0
100 a: -0.00095389935 b: -0.052508842 loss: 1814.9236
200 a: 2.2646323e-06 b: -0.051552664 loss: 0.059542436
300 a: 7.713847e-06 b: -0.051547207 loss: 0.0005487077
400 a: 7.747187e-06 b: -0.05154718 loss: 0.0005465022
500 a: 7.74737e-06 b: -0.05154718 loss: 0.0005465021
600 a: 7.747371e-06 b: -0.05154718 loss: 0.0005465021
700 a: 7.747371e-06 b: -0.05154718 loss: 0.0005465021
800 a: 7.747371e-06 b: -0.05154718 loss: 0.0005465021
900 a: 7.74737e-06 b: -0.05154718 loss: 0.0005465021
```

In [22]:

```
line_x = np.arange(min(x_data), max(x_data), 0.01)
line_y = a * line_x + b

# 그래프를 그립니다.
plt.plot(line_x, line_y, 'r-')
plt.plot(x_data, y_data, 'bo')
plt.xlabel('Date')
plt.ylabel('EntranceTime')
plt.show()
```



In [34]:

```
line_x = line_x.reshape(4800, ).astype('float32') / 255.0
line_y = np_utils.to_categorical(line_y) / 255.0
```

In [28]:

```
# Define a simple sequential model
def create_model():
    model = tf.keras.models.Sequential([
        keras.layers.Dense(32, activation='relu', input_shape=(1, 1)),
        keras.layers.Dropout(0.2),
        keras.layers.Dense(10, activation='softmax')
      ])
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

In [29]:

```
model = create_model()
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 1, 32)             64
_________________________________________________________________
dropout_1 (Dropout)          (None, 1, 32)             0
_________________________________________________________________
dense_3 (Dense)              (None, 1, 10)             330
=================================================================
Total params: 394
Trainable params: 394
Non-trainable params: 0
_________________________________________________________________
```

In [35]:

```
line_y
```

Out[35]:

```
array([[0.00392157],
       [0.00392157],
       [0.00392157],
       ...,
       [0.00392157],
       [0.00392157],
       [0.00392157]], dtype=float32)
```

In [41]:

```
model.fit(line_x, line_y, epochs=5)
```

```
Epoch 1/5
150/150 [==============================] - 0s 1ms/step - loss: 0.0760
- accuracy: 0.0000e+00
Epoch 2/5
150/150 [==============================] - 0s 1ms/step - loss: 0.0842
- accuracy: 0.0000e+00
Epoch 3/5
150/150 [==============================] - 0s 1ms/step - loss: 0.0533
- accuracy: 0.0000e+00
Epoch 4/5
150/150 [==============================] - 0s 1ms/step - loss: 0.0497
- accuracy: 0.0000e+00
Epoch 5/5
150/150 [==============================] - 0s 1ms/step - loss: 0.0576
- accuracy: 0.0000e+00
```

Out[41]:

```
<tensorflow.python.keras.callbacks.History at 0x7f8d43c69b50>
```

In [42]:

```
loss, acc = model.evaluate(line_x, line_y)
print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

```
150/150 [==============================] - 0s 978us/step - loss: 0.000
0e+00 - accuracy: 0.0000e+00
Restored model, accuracy:  0.00%
```

In [87]:

```
model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_22 (Dense)             (None, 32, 512)           1024
_________________________________________________________________
dropout_11 (Dropout)         (None, 32, 512)           0
_________________________________________________________________
dense_23 (Dense)             (None, 32, 10)            5130
=================================================================
Total params: 6,154
Trainable params: 6,154
Non-trainable params: 0
_________________________________________________________________
```

In [20]:

```

```

```
---------------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
 last)
<ipython-input-20-c193d79f7d5a> in <module>
----> 1 tf.reset_default_graph()

AttributeError: module 'tensorflow' has no attribute 'reset_default_gr
aph'
```

In [ ]:

```

```