In [1]:
```
### DL_HW06_곽용하_Kwak Yongha_2014121047

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

(a)

In [2]:
```
df = pd.read_csv("C:/Users/David/Downloads/case.csv",header=None)
from datetime import datetime
def date_range(start, end):
    start = datetime.strptime(start, "%Y-%m-%d")
    end = datetime.strptime(end, "%Y-%m-%d")
    dates = [date.strftime("%Y-%m-%d") for date in pd.date_range(start, periods=(e
    return dates
dates = date_range("2020-02-08", "2021-03-07")
```

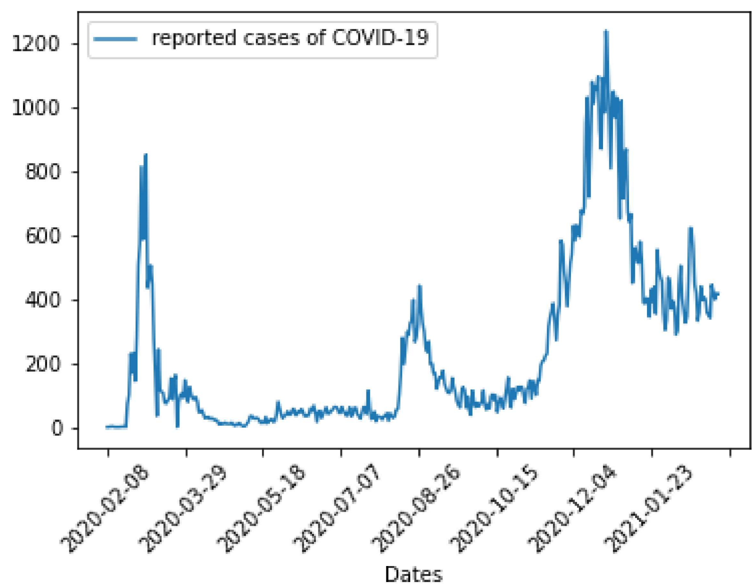
In [3]:
```
df.head()
```

Out[3]:

	0
0	0
1	0
2	1
3	2
4	1

In [4]:
```
df['Time']= list(dates)
df.columns = ['reported cases of COVID-19','Dates']
df.plot(x='Dates', y='reported cases of COVID-19')
plt.xticks(rotation = 45)
```

Out[4]:
```
(array([-50.,   0.,  50., 100., 150., 200., 250., 300., 350., 400., 450.]),
 [Text(-50.0, 0, '2021-01-17'),
  Text(0.0, 0, '2020-02-08'),
  Text(50.0, 0, '2020-03-29'),
  Text(100.0, 0, '2020-05-18'),
  Text(150.0, 0, '2020-07-07'),
  Text(200.0, 0, '2020-08-26'),
  Text(250.0, 0, '2020-10-15'),
  Text(300.0, 0, '2020-12-04'),
  Text(350.0, 0, '2021-01-23'),
  Text(400.0, 0, ''),
  Text(450.0, 0, '')])
```



In [33]:
```
## Explain the pattern of the plot
# 특별한 경향성이 있다기보다는 랜덤워크 적인 모습이 나타나고 있다. 또한 변동성이 ㄱ
# 특히 R에서 auto.arima()를 돌려보면 ARIMA(2,1,1)이 나온다.
```

In [5]:
```
# Divide the dataset into training (first 300 days) and test (remaining 94 days) s
train_data = df.loc[0:299,'reported cases of COVID-19':'Time'].values
test_data = df.loc[300:394,'reported cases of COVID-19':'Time'].values
```

In [6]:
```
len(train_data), len(test_data)
```

Out[6]:
```
(300, 94)
```

In [7]:
```
test_data
```

Out[7]:
```
array([[ 629],
       [ 583],
       [ 631],
       [ 615],
       [ 594],
       [ 677],
       [ 666],
       [ 688],
       [ 950],
       [1030],
       [ 718],
       [ 880],
       [1078],
       [1011],
       [1062],
       [1055],
       [1095],
       [ 926],
       [ 869],
```

(b)

In [8]:
```
# Feature Scaling
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
train_data_scaled = sc.fit_transform(train_data)
```

In [9]:
```
# Creating a data structure with 30 timesteps and 1 output
X_train = []
y_train = []
for i in range(30, 300):
    X_train.append(train_data_scaled[i-30:i, 0])
    y_train.append(train_data_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

In [10]:
```
# Reshaping
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

In [11]:
```
# Building the RNN
# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

In [12]:
```
regressor = Sequential()
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.sha
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))
```

In [13]:
```
print(X_train.shape)
```

```
(270, 30, 1)
```

In [14]:
```
# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 100, batch_size = 5)
```

```
Epoch 92/100
54/54 [==============================] - 1s 21ms/step - loss: 0.0024
Epoch 93/100
54/54 [==============================] - 1s 21ms/step - loss: 0.0028
Epoch 94/100
54/54 [==============================] - 1s 21ms/step - loss: 0.0027
Epoch 95/100
54/54 [==============================] - 1s 21ms/step - loss: 0.0029
Epoch 96/100
54/54 [==============================] - 1s 22ms/step - loss: 0.0023
Epoch 97/100
54/54 [==============================] - 1s 21ms/step - loss: 0.0021
Epoch 98/100
54/54 [==============================] - 1s 21ms/step - loss: 0.0027
Epoch 99/100
54/54 [==============================] - 1s 21ms/step - loss: 0.0027
Epoch 100/100
54/54 [==============================] - 1s 21ms/step - loss: 0.0027
```

Out[14]:
```
<keras.callbacks.History at 0x258082951f0>
```

In [15]:
```
real_data = test_data
dataset_total = df.loc[:,'reported cases of COVID-19']
inputs = dataset_total[len(dataset_total) - len(test_data) - 30:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(30, 124):
    X_test.append(inputs[i-30:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_data = regressor.predict(X_test)
predicted_data = sc.inverse_transform(predicted_data)
```

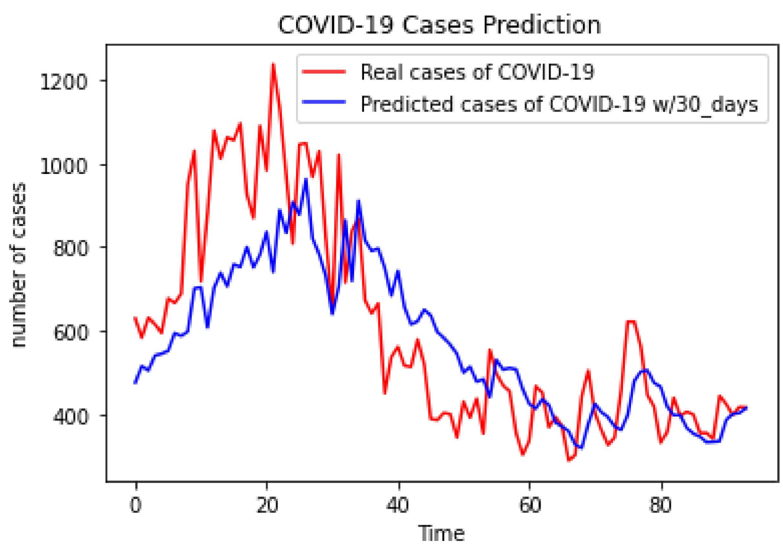
In [16]:
```
len(real_data)
```

Out[16]: 94

In [17]:
```
X_test.shape
```

Out[17]: (94, 30, 1)

In [18]:
```
# Visualising the results
plt.plot(real_data, color = 'red', label = 'Real cases of COVID-19')
plt.plot(predicted_data, color = 'blue', label = 'Predicted cases of COVID-19 w/30
plt.title('COVID-19 Cases Prediction')
plt.xlabel('Time')
plt.ylabel('number of cases')
plt.legend()
plt.show()
```



초반에 학습을 하면서 중반에는 초과예측을 하고 있으나, 후반에 갈수록 예측을 잘 하고 있는 것으로 보입니다.

(c)

In [19]:
```
# Creating a data structure with 60 timesteps and 1 output
X_train = []
y_train = []
for i in range(60, 300):
    X_train.append(train_data_scaled[i-60:i, 0])
    y_train.append(train_data_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

In [20]:
```
# Reshaping
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

In [21]:
```
regressor = Sequential()
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.sha
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))
```

In [22]:
```
print(X_train.shape)
```

```
(240, 60, 1)
```

In [23]:

```
# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 100, batch_size = 5)
```

```
Epoch 92/100
48/48 [==============================] - 2s 39ms/step - loss: 0.0025
Epoch 93/100
48/48 [==============================] - 2s 39ms/step - loss: 0.0026
Epoch 94/100
48/48 [==============================] - 2s 39ms/step - loss: 0.0027
Epoch 95/100
48/48 [==============================] - 2s 40ms/step - loss: 0.0028
Epoch 96/100
48/48 [==============================] - 2s 39ms/step - loss: 0.0020
Epoch 97/100
48/48 [==============================] - 2s 39ms/step - loss: 0.0023
Epoch 98/100
48/48 [==============================] - 2s 39ms/step - loss: 0.0024
Epoch 99/100
48/48 [==============================] - 2s 39ms/step - loss: 0.0025
Epoch 100/100
48/48 [==============================] - 2s 39ms/step - loss: 0.0024
```

Out[23]:

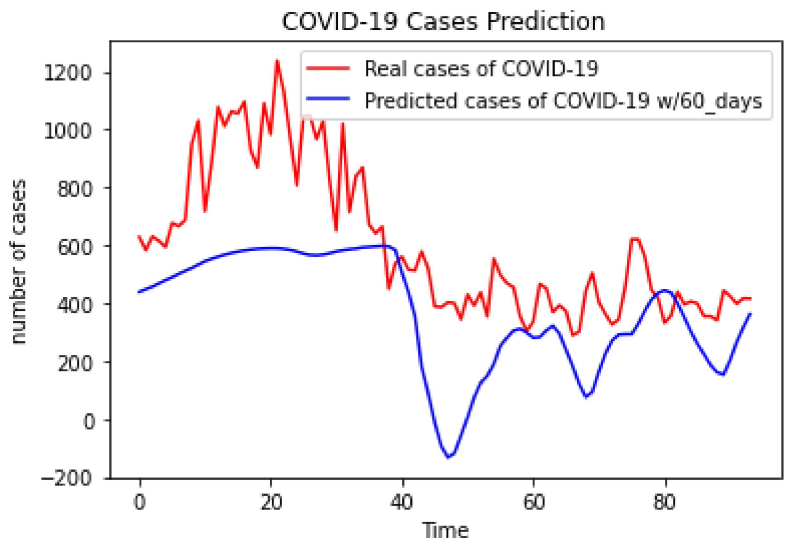
```
<keras.callbacks.History at 0x25812064640>
```

In [24]:

```
real_data = test_data
dataset_total = df.loc[:,'reported cases of COVID-19']
inputs = dataset_total[len(dataset_total) - len(test_data) - 60:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(60, 154):
    X_test.append(inputs[i-60:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_data = regressor.predict(X_test)
predicted_data = sc.inverse_transform(predicted_data)
```

In [25]:

```
# Visualising the results
plt.plot(real_data, color = 'red', label = 'Real cases of COVID-19')
plt.plot(predicted_data, color = 'blue', label = 'Predicted cases of COVID-19 w/60
plt.title('COVID-19 Cases Prediction')
plt.xlabel('Time')
plt.ylabel('number of cases')
plt.legend()
plt.show()
```



(b)에 비해 초반 학습이 효과적이지 않고 후반에 가도 예측력이 좋지 않아 보입니다.

(d)

In [26]:

```
# Creating a data structure with 90 timesteps and 1 output
X_train = []
y_train = []
for i in range(90, 300):
    X_train.append(train_data_scaled[i-90:i, 0])
    y_train.append(train_data_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

In [27]:

```
# Reshaping
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

In [28]:

```
regressor = Sequential()
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.sha
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))
```

In [29]:

```
# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 100, batch_size = 5)
```

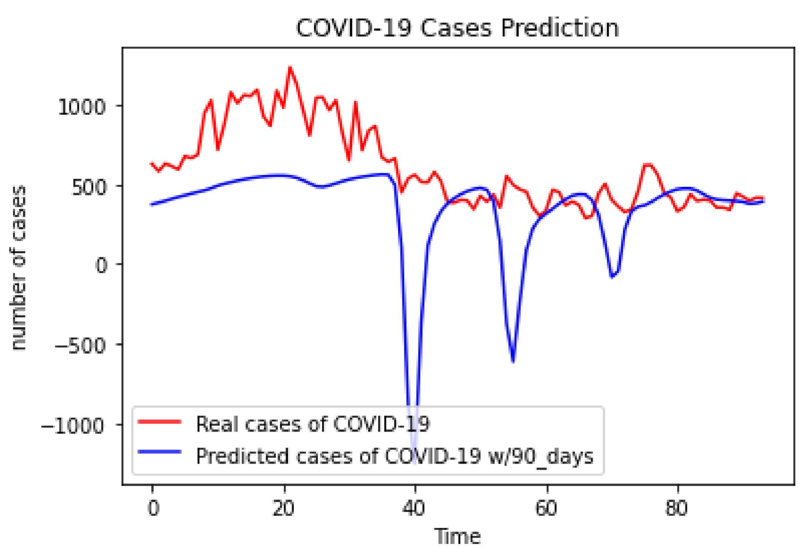
```
Epoch 92/100
42/42 [==============================] - 2s 58ms/step - loss: 0.0038
Epoch 93/100
42/42 [==============================] - 2s 59ms/step - loss: 0.0029
Epoch 94/100
42/42 [==============================] - 2s 58ms/step - loss: 0.0028
Epoch 95/100
42/42 [==============================] - 2s 59ms/step - loss: 0.0028
Epoch 96/100
42/42 [==============================] - 2s 58ms/step - loss: 0.0026
Epoch 97/100
42/42 [==============================] - 3s 60ms/step - loss: 0.0025
Epoch 98/100
42/42 [==============================] - 2s 58ms/step - loss: 0.0025
Epoch 99/100
42/42 [==============================] - 2s 58ms/step - loss: 0.0027
Epoch 100/100
42/42 [==============================] - 2s 59ms/step - loss: 0.0030
```

Out[29]:

```
<keras.callbacks.History at 0x2581bd2da00>
```

In [30]:
```
real_data = test_data
dataset_total = df.loc[:,'reported cases of COVID-19']
inputs = dataset_total[len(dataset_total) - len(test_data) - 90:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(90, 184):
    X_test.append(inputs[i-90:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_data = regressor.predict(X_test)
predicted_data = sc.inverse_transform(predicted_data)
```

In [31]:
```
# Visualising the results
plt.plot(real_data, color = 'red', label = 'Real cases of COVID-19')
plt.plot(predicted_data, color = 'blue', label = 'Predicted cases of COVID-19 w/90
plt.title('COVID-19 Cases Prediction')
plt.xlabel('Time')
plt.ylabel('number of cases')
plt.legend()
plt.show()
```



(e)

In [32]:
```
# window 기간이 길어짐에 따라 단기 파동을 세세하게 예측하지 못하고 더 둔하게 반응
# 또한 지나치게 기간설정을 길게하면 변동폭이 지나치게 커지면서 음의 값을 예측하는

# 따라서 30일 기준을 사용하는 (b)가 가장 좋다.
# 이는 데이터의 양 자체가 적은 것뿐만 아니라, 기존 데이터가 arima(2,1,1) 등 시계열
# 1차 차분이 필요한 특성을 지닌 것에 따른 것일 수 있다.
```