Keras를 활용한 MNIST 데이터셋 분류 모델

데이터셋 로드 및 전처리

```
from keras.utils import to_categorical
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense

# MNIST data
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
print(train_images.shape, train_labels.shape, test_images.shape,
test_labels.shape)

train_images = train_images.reshape(train_images.shape[0], 784)
.astype('float32')/255.0
test_images = test_images.reshape(test_images.shape[0], 784)
.astype('float32')/255.0
train_labels = to_categorical(train_labels) # One-Hot Encoding
test_labels = to_categorical(test_labels) # One-Hot Encoding
```

Model 구축

```
model = Sequential ()
model.add(Dense(256, activation='relu')) # units=256, activation='relu'
model.add(Dense(256, activation='relu')) # units=256, activation='relu'
model.add(Dense(256, activation='relu')) # units=256, activation='relu'
model.add(Dense(10, activation='softmax')) # units=10, activation='softmax'
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=
['accuracy'])
```

Model Training & Testing

```
model.fit(train_images, train_labels, epochs=5, batch_size=32, verbose=1)

_, accuracy = model.evaluate(test_images, test_labels)
print('Accuracy: ', accuracy)
model. summary ()
```

기존의 sentiment_imdb 데이터의 코드에서 keras 딥러닝 모델을 추가해보자!

```
list_preprocessing_reviews=[] # 전처리한 모든 리뷰글들을 저장하는 리스트
for review in list_reviews:
```

```
    list_preprocessing_reviews.append(preprocessing(review))
# review들의 전처리가 완성되어 리스트로 만들었으니, 다시 sentiment와 결합(데이터프레임으로 만들어줌)
list_preprocessing_data = pd.DataFrame({'review':list_preprocessing_reviews, 'sentiment':imdb_pd['sentiment']})
# 각각의 review와 sentiment를 가지고 각각의 리스트를 만들어줌
list_reviews = list(list_preprocessing_data['review'])
list_sentiments = list(list_preprocessing_data['sentiment'])
```

```
from sklearn.feature_extraction.text import TfidfVectorizer # 벡터화
from sklearn.model_selection import train_test_split
import numpy as np
vector = TfidfVectorizer(max_features=5000) # 주어진 문장을 벡터로 만드는 객체 생성
X = vector.fit_transform(list_reviews).toarray() # fit_transform함수를 이용하여 5000사이즈 벡터 생성
y = np.array(list_sentiments) # 0~1로 표현되는 감정값을 가지고 있음
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=42)
# 위의 코드는 80:20으로 주어진 데이터를 학습용/테스트용으로 나눠줌(랜덤)
print(X.shape) # (50000, 5000) 5만개의 데이터ㅏ 있고, 각각의 데이터는 5000사이즈의 벡터
print(x_train.shape) # (40000, 5000) 그중 80%가 학습용이라 4만개의 데이터가 x_train
print(x_test.shape) # (1000, 5000) 그중 20%가 테스트용이니, 1만개의 데이터가 x_test

print(y_train.shape) # (40000,) 그 중에서 80%가 학습용이니, 4만개의 감정수치(0~1)가 y_train(1이라는 값은 생략해서 나오지 않음)
print(y_test.shape) # (10000,) 그 중에서 20%가 학습용이니, 1만개의 감정수치(0~1)가 y_test(1이라는 값은 생략해서 나오지 않음)
```

```
# Keras로 이진분류
# 딥러닝 1단계 : 모델을 만든다.
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='sgd', metrics=['accuracy'])

# 딥러닝 2단계 : 학습을 한다.
from keras.callbacks import EarlyStopping
es = EarlyStopping(monitor='loss', min_delta=1e-2, patience=5, verbose=1, restore_best_weights=True)

model.fit(x_train, y_train, epochs=100, verbose=1, callbacks=[es])

# 딥러닝 3단계 : 추론/테스트을 한다.
_, accuracy = model.evaluate(x_test, y_test)
print('Accuracy: %6.2f%%'%(100*accuracy))
model. summary ()
```

직접 입력한 문장이 긍정인지 부정인지 분리해주도록 코드 작성해보자!

```
query = input().rstrip()
query = vector.transform([query]).toarray()
result = 1 if model.predict(query).item() > 0.5 else 0
print(result)
```