5. RNN(순환신경망)

3강. 시계열 데이터를 예측하는 LSTM 구현

학습목표

- 세계 항공 여행객 수를 예측하는 LSTM을 구현할 수 있다.
- 아파트 지수를 예측하는 LSTM을 구현할 수 있다.

학습내용

- 세계 항공 여행객 수를 예측하는 LSTM 구현
- 아파트 지수를 예측하는 LSTM 구현

1. 세계 항공 여행객 수를 예측하는 LSTM 구현

(1) 시계열 데이터를 예측하는 LSTM 구현

- 세계 항공 여행 승객 수의 증가에 대한 데이터를 활용하여 이전 12개월치 승객 수를 이용해 다음 달의 승객 수를 예측하는 LSTM 구현하기

① 데이터를 불러오고 기본적인 처리를 하는데 필요한 패키지를 임포트

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import model_selection
from keras import models, layers
import seaborn as sns

from keraspp import skeras
```

```
Using TensorFlow backend.
```

② 데이터 불러오기

- Dataset 클래스를 구성해 데이터를 불러오기

In [2]:
```
class Dataset:
    def __init__(self, fname='international-airline-passengers.csv', D=12):
        data_dn = load_data(fname=fname)
        X, y = get_Xy(data_dn, D=D)
        X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.2, random_state=42)

        self.X, self.y = X, y
        self.X_train, self.X_test, self.y_train, self.y_test = X_train, X_test, y_train, y_test


def load_data(fname='international-airline-passengers.csv'):
    dataset = pd.read_csv(fname, usecols=[1], engine='python', skipfooter=3)
    data = dataset.values.reshape(-1)
    plt.plot(data)
    plt.xlabel('Time'); plt.ylabel('#Passengers')
    plt.title('Original Data')
    plt.show()

    # data normalize
    data_dn = (data - np.mean(data)) / np.std(data) / 5
    plt.plot(data_dn)
    plt.xlabel('Time'); plt.ylabel('Normalized #Passengers')
    plt.title('Normalized data by $E[]$ and $5#sigma$')
    plt.show()

    return data_dn

def get_Xy(data, D=12):
    # make X and y
    X_l = []
    y_l = []
    N = len(data)
    assert N > D, "N should be larger than D, where N is len(data)"
    for ii in range(N-D-1):
        X_l.append(data[ii:ii+D])
        y_l.append(data[ii+D])
    X = np.array(X_l)
    X = X.reshape(X.shape[0], X.shape[1], 1)
    y = np.array(y_l)
    print(X.shape, y.shape)
    return X, y
```

③ LSTM 시계열 회귀 모델링

In [3]:
```
def rnn_model(shape):
    m_x = layers.Input(shape=shape) #X.shape[1:]
    m_h = layers.LSTM(10)(m_x)
    m_y = layers.Dense(1)(m_h)
    m = models.Model(m_x, m_y)

    m.compile('adam', 'mean_squared_error')

    m.summary()

    return m
```

- 12개의 시간열과 1개의 특징점으로 구성
- LSTM은 10개의 노드로 구성되어 있고, 내부 파라미터는 480개
- 최종 출력 계층의 노드는 하나이며, LSTM과 출력 계층 사이에 적용되는 가중치 수는 11개
- 가중치 11개 중 10개는 입력 값에 대한 가중치이고, 1개는 평균값을 조절하는 가중치

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 12, 1)             0
_________________________________________________________________
lstm_1 (LSTM)                (None, 10)                480
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 11
=================================================================
Total params: 491
Trainable params: 491
Non-trainable params: 0
_________________________________________________________________
```

④ 학습하고 평가하기

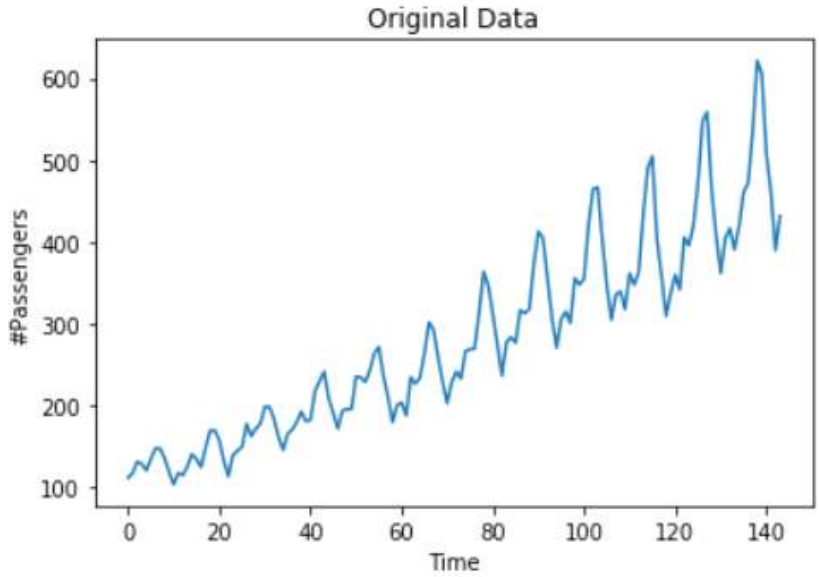
- 시계열 LSTM을 학습하고 평가하는 머신 클래스

In [4]:
```
class Machine():
    def __init__(self):
        self.data = Dataset()
        shape = self.data.X.shape[1:]
        self.model = rnn_model(shape)

    def run(self, epochs=400):
        d = self.data
        X_train, X_test, y_train, y_test = d.X_train, d.X_test, d.y_train, d.y_test
        X, y = d.X, d.y
        m = self.model
        h = m.fit(X_train, y_train, epochs=epochs, validation_data=[X_test, y_test], verbose=0)

        skeras.plot_loss(h)
        plt.title('History of training')
        plt.show()

        yp = m.predict(X_test)
        print('Loss:', m.evaluate(X_test, y_test))
        plt.plot(yp, label='Origial')
        plt.plot(y_test, label='Prediction')
        plt.legend(loc=0)
        plt.title('Validation Results')
        plt.show()

        yp = m.predict(X_test).reshape(-1)

        plt.figure(figsize=(7, 5))
        sns.barplot(x="Sample", y="Normalized #Passengers",
                    hue="Type", data=df)
        plt.ylabel('Normalized #Passengers')
        plt.show()

        yp = m.predict(X)

        plt.plot(y, label='Origial')
        plt.plot(yp, label='Prediction')
        plt.legend(loc=0)
        plt.title('All Results')
        plt.show()
```

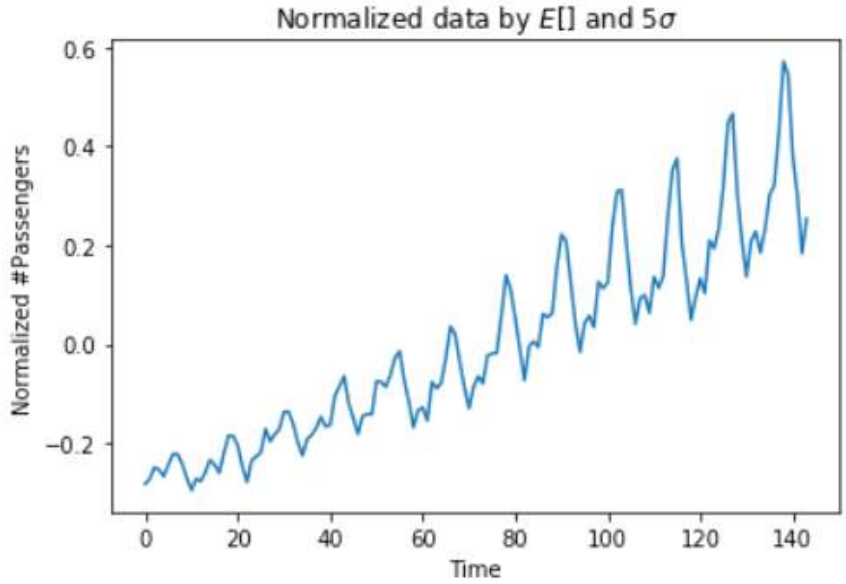
⑤ 코드 실행

```
In [5]: machine = Machine()
        machine.run(epochs=400)
```

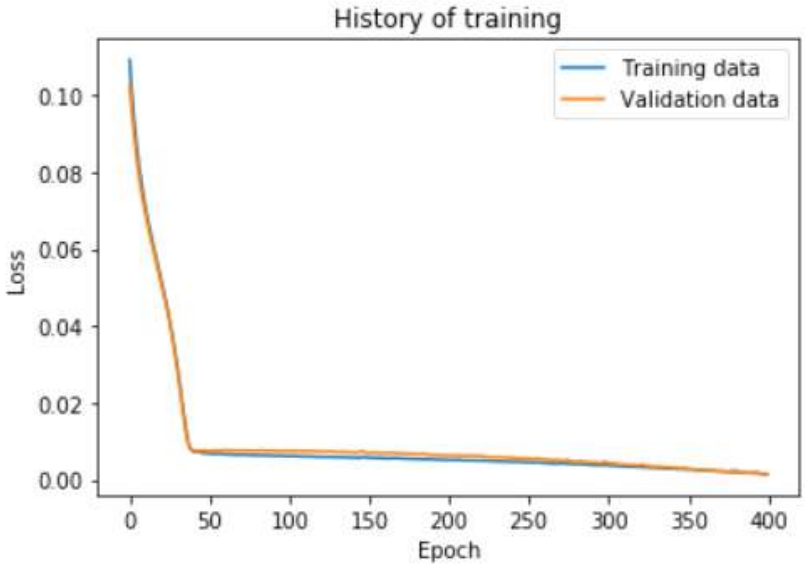
- 학습에 사용한 데이터
  - 월간 항공기 여행자 수의 추이



- 평준화된 월간 항공기 여행자 수



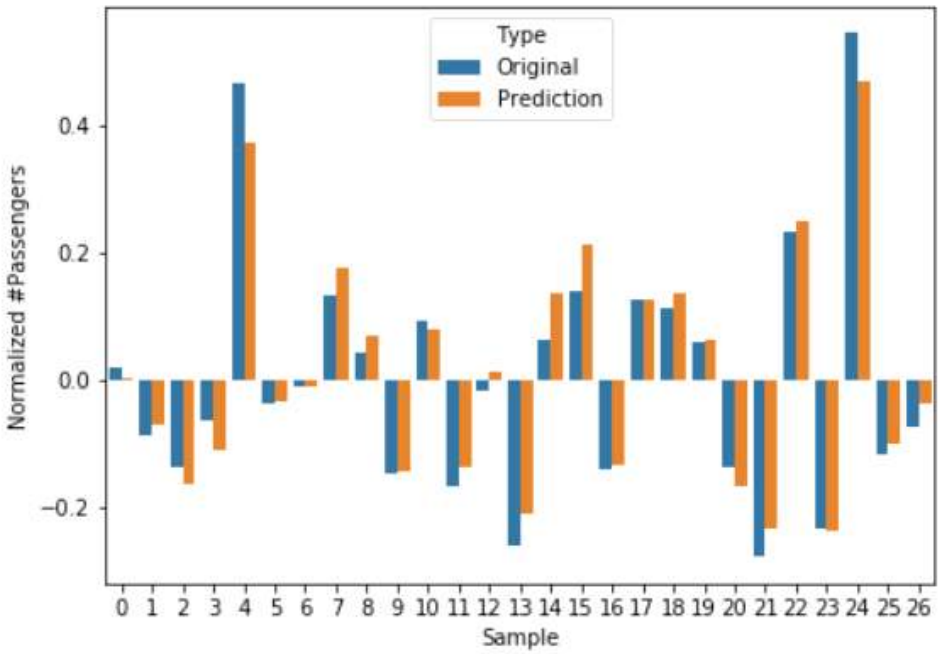
- LSTM 시계열 예측의 학습 곡선

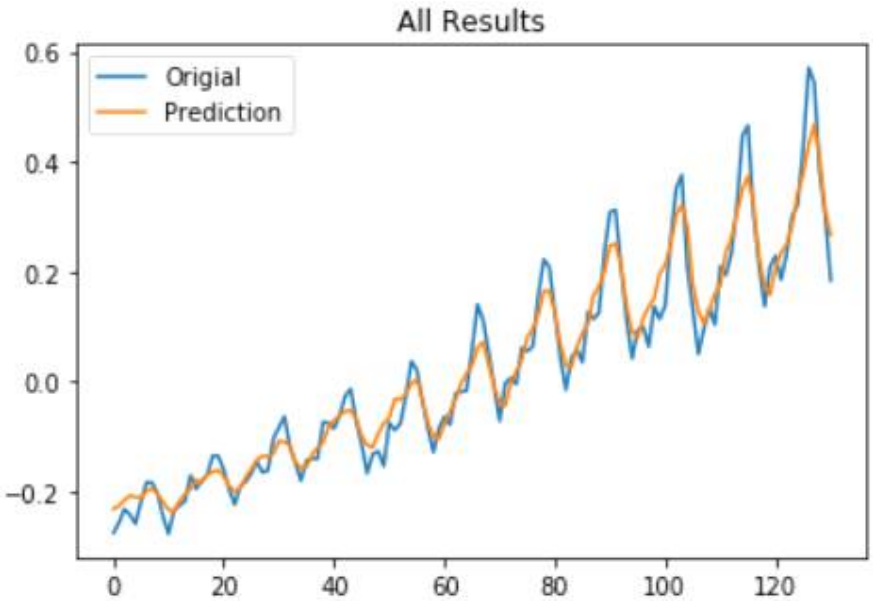


- LSTM 시계열 예측의 검증 데이터를 이용한 손실 측정과 예측값과 실제값 비교 곡선


```
27/27 [==============================] - 0s 37us/step
Loss: 0.0015149743994697928
(27,) (27,)
```

• LSTM 시계열 예측 신경망의 학습 후 예측 결과와 실제값 비교





2. 아파트 지수를 예측하는 LSTM 구현

(1) 아파트 지수 시계열을 예측하는 LSTM 구현하기

• 한국감정원에서 배포하는 전국주택가격지수 중에서 강남구의 아파트 거래가격 지수 시계열을 예측하는 LSTM 구현

① 데이터를 불러오고 기본적인 처리를 하는데 필요한 패키지를 임포트

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import model_selection
from keras import models, layers
import seaborn as sns

from keraspp import skeras
```

```
Using TensorFlow backend.
```

② 데이터 불러오기

- Dataset 클래스를 구성해 데이터를 불러오기

In [2]:
```
class Dataset:
    def __init__(self, fname='apt_price.csv', D=12):
        data_dn = load_data(fname=fname)
        X, y = get_Xy(data_dn, D=D)
        X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y,
                                                test_size=0.2, random_state=42)

        self.X, self.y = X, y
        self.X_train, self.X_test, self.y_train, self.y_test = X_train, X_test,
                                                               y_train, y_test


def load_data(fname='apt_price.csv'):
    dataset = pd.read_csv(fname, usecols=[1], engine='python', skipfooter=3)
    data = dataset.values.reshape(-1)
    plt.plot(data)
    plt.xlabel('Time'); plt.ylabel('#Passengers')
    plt.title('Original Data')
    plt.show()

    # data normalize
    data_dn = (data - np.mean(data)) / np.std(data) / 5
    plt.plot(data_dn)
    plt.xlabel('Time'); plt.ylabel('Normalized #Passengers')
    plt.title('Normalized data by $E[]$ and $5\Wsigma$')
    plt.show()

    return data_dn
```

```
def get_Xy(data, D=12):
    # make X and y
    X_l = []
    y_l = []
    N = len(data)
    assert N > D, "N should be larger than D, where N is len(data)"
    for ii in range(N-D-1):
        X_l.append(data[ii:ii+D])
        y_l.append(data[ii+D])
    X = np.array(X_l)
    X = X.reshape(X.shape[0], X.shape[1], 1)
    y = np.array(y_l)
    print(X.shape, y.shape)
    return X, y
```

③ LSTM 시계열 회귀 모델링

In [3]:
```
def rnn_model(shape):
    m_x = layers.Input(shape=shape) #X.shape[1:]
    m_h = layers.LSTM(10)(m_x)
    m_y = layers.Dense(1)(m_h)
    m = models.Model(m_x, m_y)

    m.compile('adam', 'mean_squared_error')

    m.summary()

    return m
```

④ 학습 및 성능 평가를 위한 머신 클래스 구현

- 클래스를 선언한 후 머신을 초기화

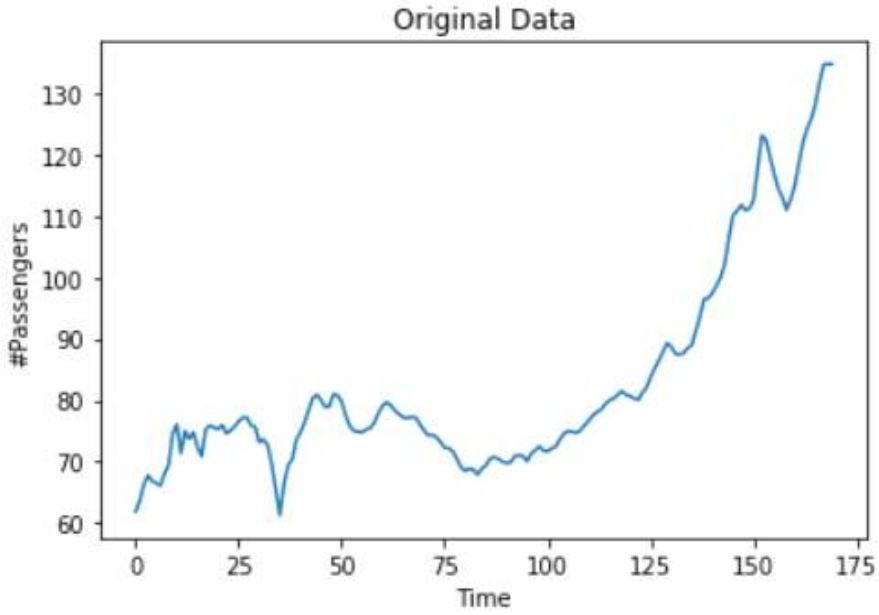
In [4]:
```
class Machine():
    def __init__(self):
        self.data = Dataset()
        shape = self.data.X.shape[1:]
        self.model = rnn_model(shape)
```

```
    def run(self, epochs=400):
        d = self.data
        X_train, X_test, y_train, y_test = d.X_train, d.X_test,
                                           d.y_train, d.y_test
        X, y = d.X, d.y
        m = self.model
        h = m.fit(X_train, y_train, epochs=epochs,
                  validation_data=[X_test, y_test], verbose=0)

        skeras.plot_loss(h)
        plt.title('History of training')
        plt.show()

        df = pd.DataFrame()
        df['Sample'] = list(range(len(y_test))) * 2
        df['Normalized #Passengers'] = np.concatenate([y_test, yp], axis=0)
        df['Type'] = ['Original'] * len(y_test) + ['Prediction'] * len(yp)

        plt.figure(figsize=(7, 5))
        sns.barplot(x="Sample", y="Normalized #Passengers",
                    hue="Type", data=df)
        plt.ylabel('Normalized #Passengers')
        plt.show()

        yp = m.predict(X)

        plt.plot(y, label='Origial')
        plt.plot(yp, label='Prediction')
        plt.legend(loc=0)
        plt.title('All Results')
        plt.show()
```

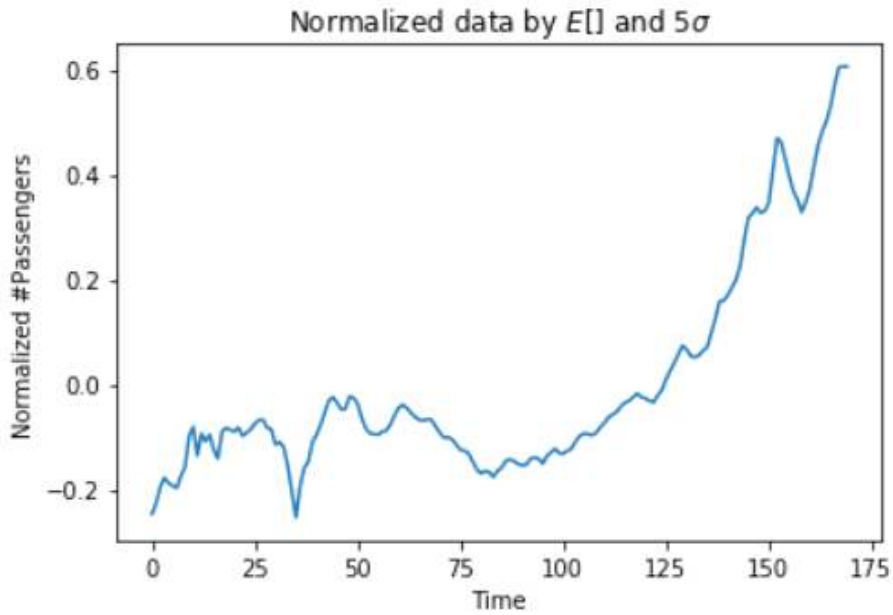
⑤ 코드 실행

```
In [5]: machine = Machine()
        machine.run(epochs=400)
```

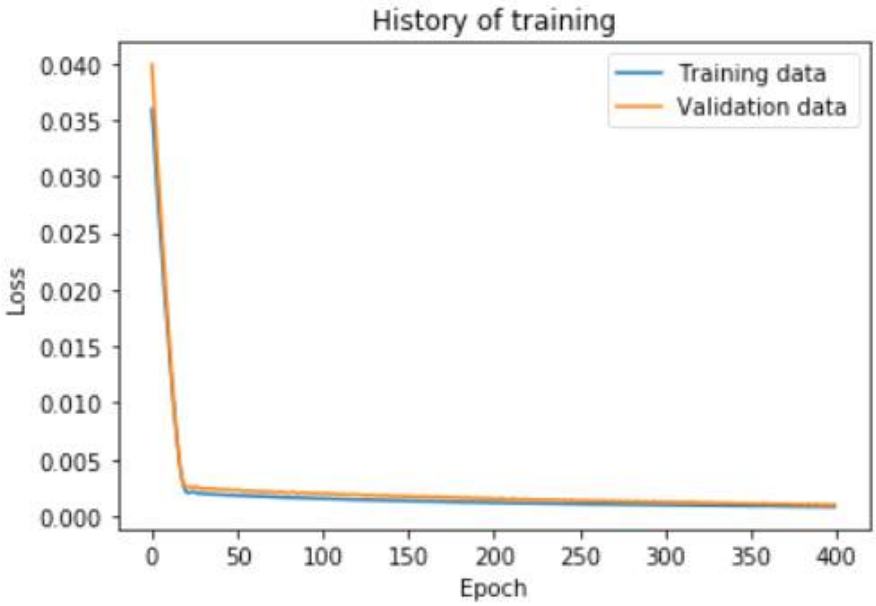
- 학습에 사용한 데이터
  - 월간 아파트 지수의 추이



- 평준화된 월간 아파트 지수

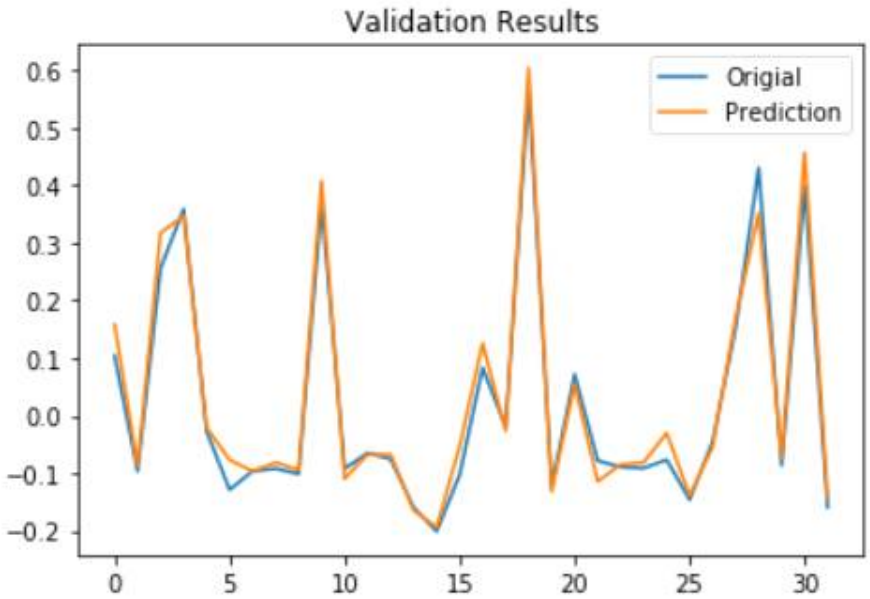


- LSTM 시계열 예측의 학습 곡선



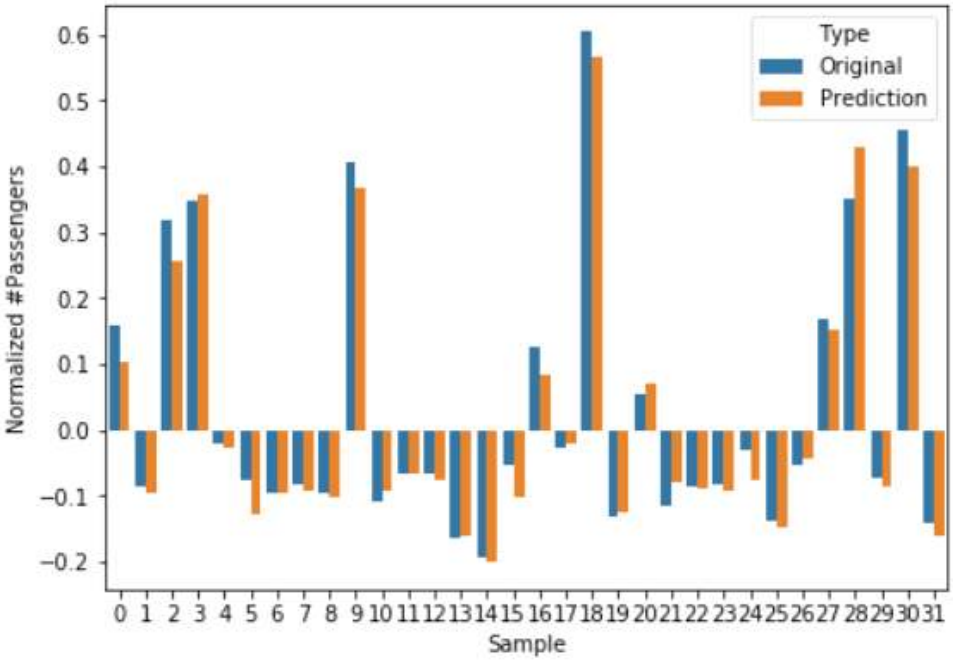

```
32/32 [==============================] - 0s 62us/step
Loss: 0.0010097066406160593
```

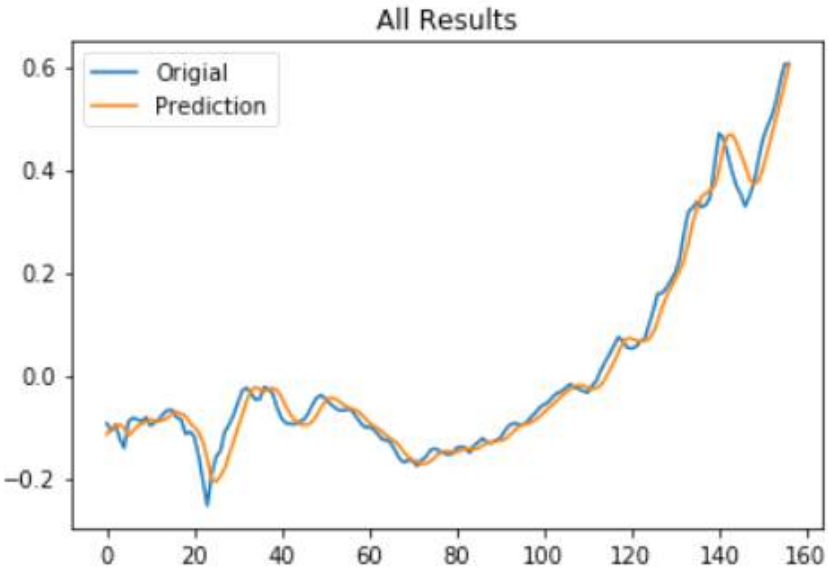
- LSTM 시계열 예측의 검증 데이터를 이용한 손실 측정과 예측값과 실제값 비교 곡선




```
32/32 [==============================] - 0s 31us/step
Loss: 0.0010097066406160593
(32,) (32,)
```

• LSTM 시계열 예측 신경망의 학습 후 예측 결과와 실제값 비교





평가하기

1. LSTM에서 어떤 값을 업데이트할지 결정하는 Gate를 고르시오.

① Forget Gate

② Output gate

③ Hidden Gate

④ Input Gate

- 정답 : ①번

해설 : LSTM의 Input Gate에서 어떤 값을 업데이트할지 결정하게 된다.

2. LSTM은 RNN의 한 종류이다.

① O

② X

- 정답 : O

해설 : LSTM은 RNN의 한 종류이며, 장기의존성 문제를 해결한 인공신경망이다.

학습정리

1. 세계 항공 여행객 수를 예측하는 LSTM 구현
   - 세계 항공 여행 승객 수에 증가를 예측하는 LSTM 구현

2. 아파트 지수를 예측하는 LSTM 구현
   - 아파트 매매 지수를 통해 증가/감소를 예측하는 LSTM 구현

다음 주 예고

"6. AE(오토인코더)" 에 대해 학습하겠습니다.