Day 7

RNN

Exercise #7

- Complete the Pima-Indian Diabetes tutorial on https://www.kaggle.com/atulnet/pima-diabetes-keras-implementation
  - 난수를 고정 (np.random.seed)
  - Replace null values with average values (fillna)
  - x, y split and train, test split
  - Build a model (12 Dense with RELU, 8 Dense nodes with RELU activation, 1 Dense output with sigmoid activation)
  - Compile the model (loss=‘binary_crossentropy’, optimizer=‘adam’, metrics=[‘accuracy’]

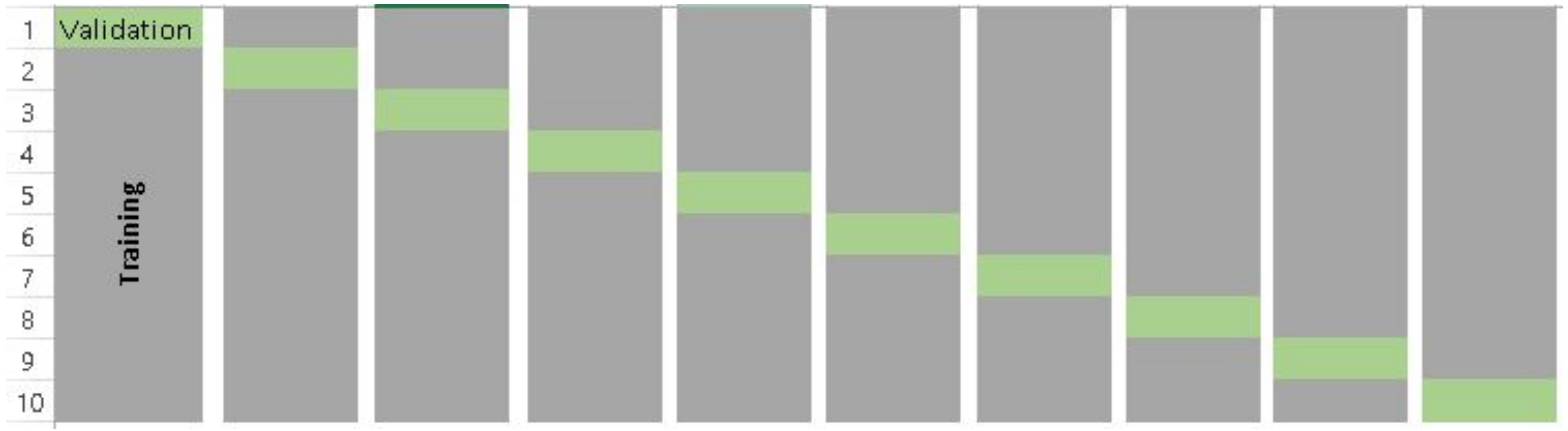
Exercise #7

- Fit the model (epochs = 1000, batch_size=16)
- Predict the diabetes
- Evaluate the model
- Create History table and accuracy and loss chart

Cross Validation

- A techniques to build robust models which are not prone to overfitting 모델을 구축할때 오버피팅을 방지하기 위한 방법
- Iterate over various models to find a better performing model 가장 효과적인 모델을 찾기위해 다양한 모델들을 반복함
  - Leave one out cross validation (LOOCV)
  - K-fold cross validation

k-fold Cross Validation



- The data set is divided into *k* subsets. One of the *k* subsets is used as the test set and the other *k-1* subsets are put together to form a training set. **전체 데이터셋을 k개로 나눈뒤 한개는 테스트셋으로 나머지는 훈련셋으로 나눔. 이 작업을 k번 수행**

Leave One Out Cross Validation

- The extreme case of k-fold cross validation
  k교차검증의 가장 극단적인 예
- k equal to n, the number of data points in the set.
  k가 데이터셋 안에 있는 데이터포인트의 수가 되는 경우

Holdout Method

- The simplest kind of cross validation 가장 간단한 교차검증의 종류
- The dataset is divided into two groups: training set and test set. 데이터셋을 훈련과 테스트셋으로 나눔
- Training set is used to develop a prediction model and testing set to evaluate the model 훈련셋은 모델을 구축하기 위해 테스트셋은 평가하기 위해 사용

Model Evaluation

```
from sklearn.metrics import r2_score, accuracy_score

r2_score(y_train, lmfit.predict(x_train))
```

http://scikit-learn.org/stable/modules/model_evaluation.html

Cross Validation

```
from sklearn.model_selection import
cross_val_score

cross_val_score(lmfit, x_train,
y_train, cv=5, scoring ='r2')
```

R^2

- R Squared, also called the coefficient of determination 결정계수라고 불림
- Provides a “goodness of fit” measure for the predictions to the observations. 관측치에 대한 예측이 얼마나 적절한지를 보여주는 측정치

Accuracy

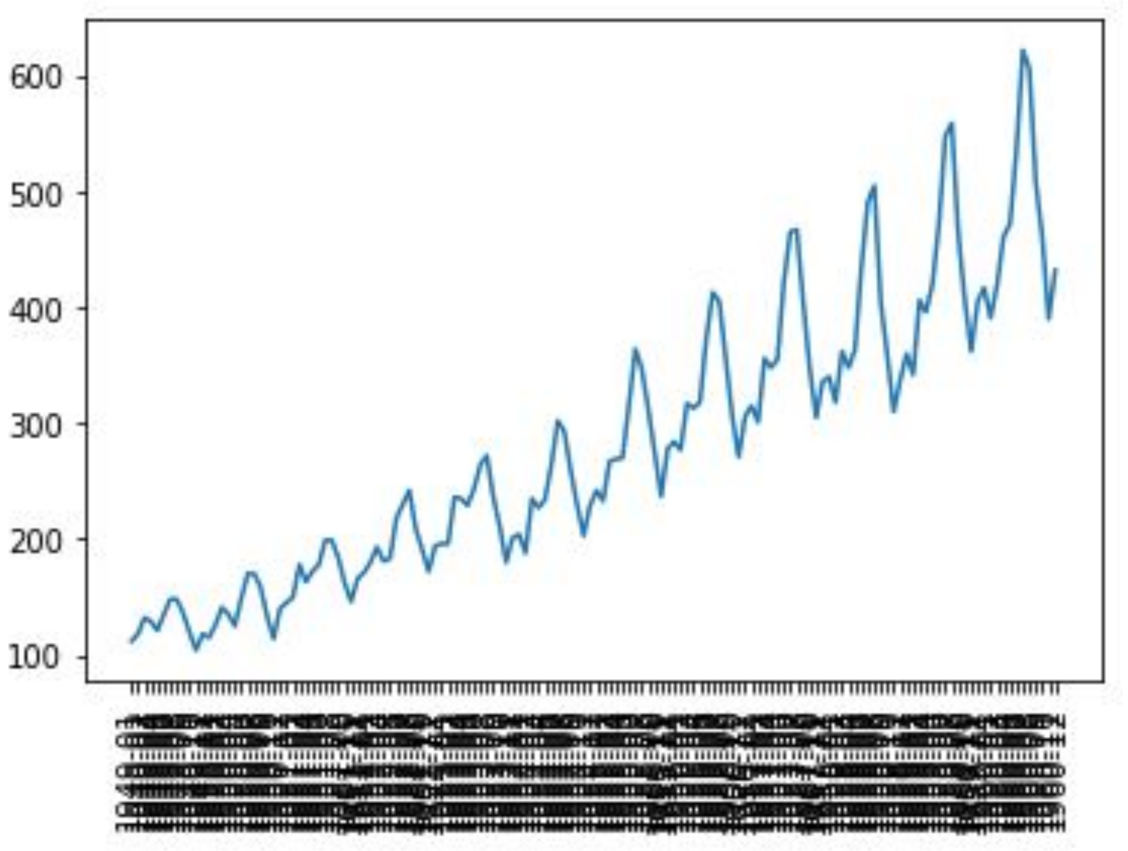
- The percentage of correctly classifies instances out of all instances. 모든 데이터셋중에 정확히 분류된 경우의 퍼센트
- It is more useful on a binary classification than multi-class classification problems 이진분류에 더 유용함

AirPassengers

AirPassengers dataset includes monthly Airline Passenger Numbers 1949-1960. Build a prediction model for Airline Passenger Number. 에어패씬저 데이터셋에는 1949년에서 1960년까지의 월별 승객수에 대한 데이터를 포함하고 있습니다. 승객수를 예측하는 모델을 만드시요.

Time Series Data

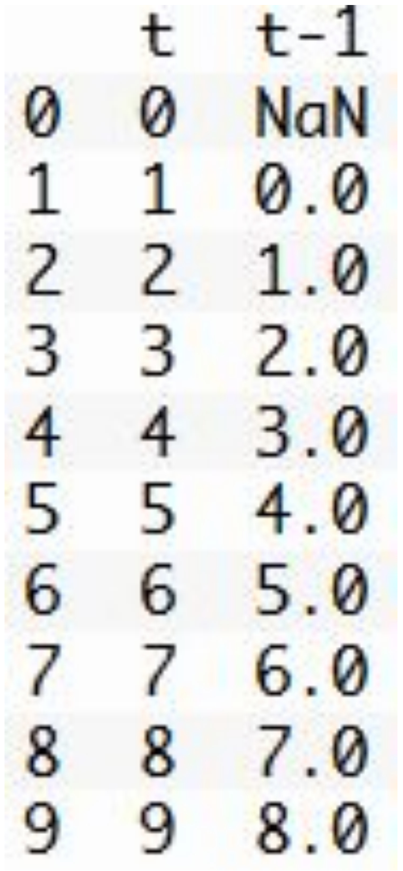
```
air =
pd.read_csv('AirPassenge
rs.csv')

air.set_index('Month',
inplace=True)

plt.plot(air)

plt.xticks(rotation=90)
```



shift()

- Used to create copies of columns that are pushed forward (rows of NaN values added to the front) or pulled back (rows of NaN values added to the end).
  데이터를 앞이나 뒤로 밀수있음

```
df = pd.DataFrame()
df['t'] = [x for x in range(10)]
df['t-1'] = df['t'].shift(1)
print(df)
```



```
   t  t-1
0  0  NaN
1  1  0.0
2  2  1.0
3  3  2.0
4  4  3.0
5  5  4.0
6  6  5.0
7  7  6.0
8  8  7.0
9  9  8.0
```

Null Values in DataFrame

```
df = pd.DataFrame()

df['t'] = [x for x in
range(10)]

df['t+1'] =
df['t'].shift(-1)

df['t+2'] =
df['t'].shift(-2)

df['t+3'] =
df['t'].shift(-3)

print(df)
```

```
df.dropna(inplace=True)
```

X and Y Split

```
y = df['t+3']
x = df.drop('t+3', axis=1)
or
x = df.iloc[:,:-1]
y = df.iloc[:, -1]
```

CNN Model

```
model = Sequential()
model.add(Dense(100, activation='relu', input_dim=3))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

Fit and Prediction

```
# fit model

history = model.fit(x,
y, epochs=2000,
verbose=1,
validation_split=.2)


# demonstrate prediction

y_pred =
model.predict(x)
```

```
# plot prediction

plt.plot(y)

plt.plot(y_pred)
```

Evaluation

```
#evaluate the model
model.evaluate(x,y)
```

```
history_table =
pd.DataFrame(history.histo
ry)
import matplotlib.pyplot
as plt
plt.plot(history.history['
loss'], label='loss')
plt.plot(history.history['
val_loss'], label='val
loss')
plt.title('LOSS CHART')
plt.legend()
```

Test Data

```
#test data
x_input = np.array([50, 60, 70])
x_input = x_input.reshape((1, 3))
test_pred = model.predict(x_input, verbose=0)
print(test_pred)
```

Exercise #7

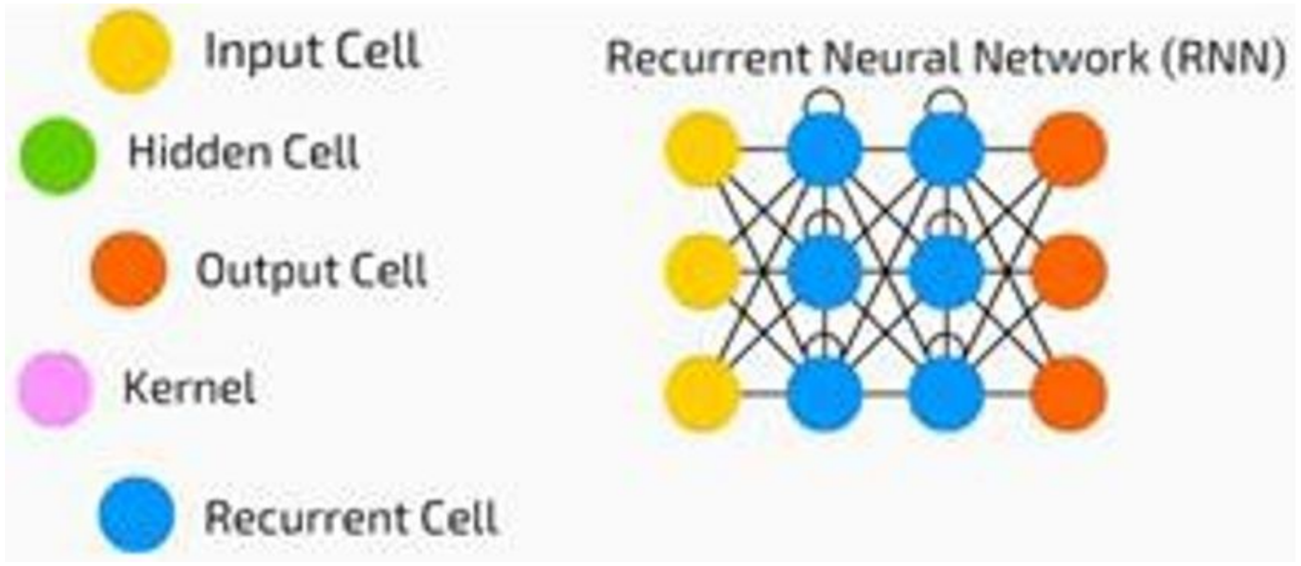
- Use the shift function and build the CNN model for Air Passenger data 여객승객데이터를 쉬프트로 이동한 후에 **CNN 모델을 만드시요**
  - Load air passengers csv file (set_index, plot)
  - Create a new dataframe (shift, dropna)
  - X and y split
  - CNN model (100 Dense with relu, 1 Dense)

Exercise #7

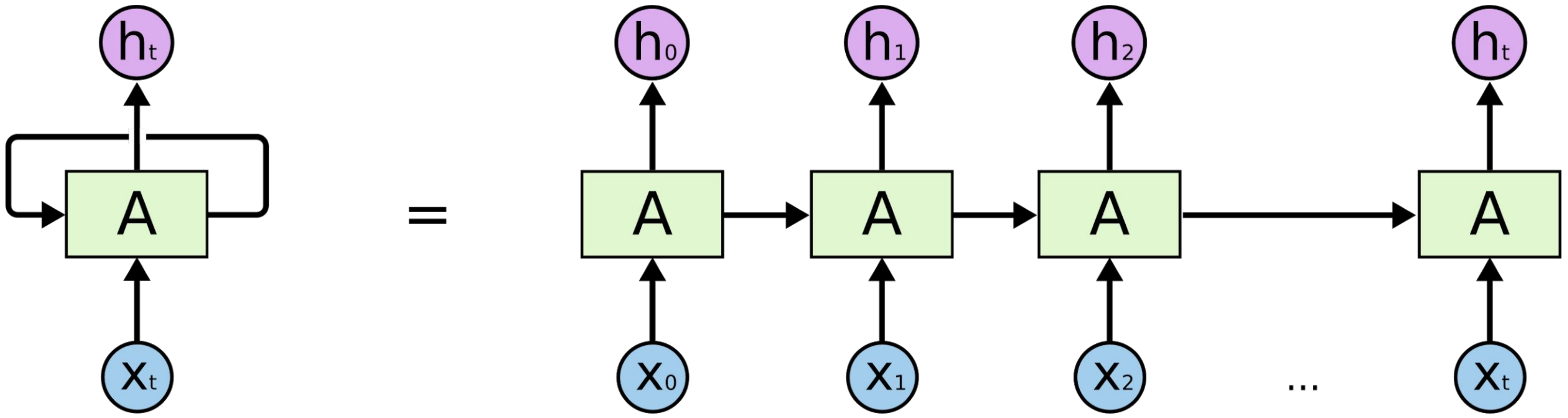
- Use the shift function and build the CNN model for Air Passenger data **여객승객데이터를 쉬프트로 이동한 후에 CNN 모델을 만드시요**
  - Compile (optimizer=‘adam’, loss=‘mse’)
  - Fit (epochs=2000, validation_split=.2)
  - Predict and plot
  - Evaluate (loss and val_loss chart)
  - Test the data (50, 60, 70, reshape)

RNN (Recurrent Neural Network) 순환 인공 신경망

- A type of Neural Network where the output from previous step are fed as input to the current step
  신경망의 종류로 이전스텝의 결과물이 현재스텝의 입력값이 됨
- Language modelling or Natural Language Processing (NLP)
  자연어처리



Recurrent Cells



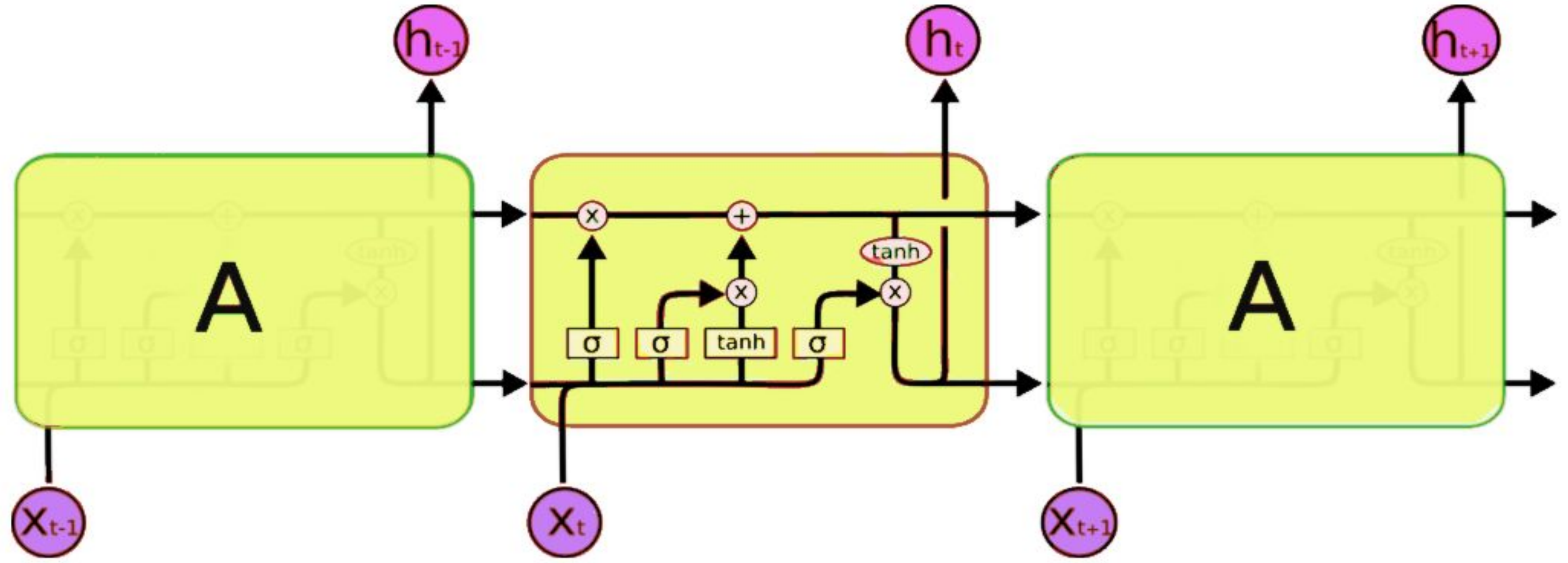
RNN Formula

$$Y_t = \tanh(wY_{t-1} + u\,x_t)$$

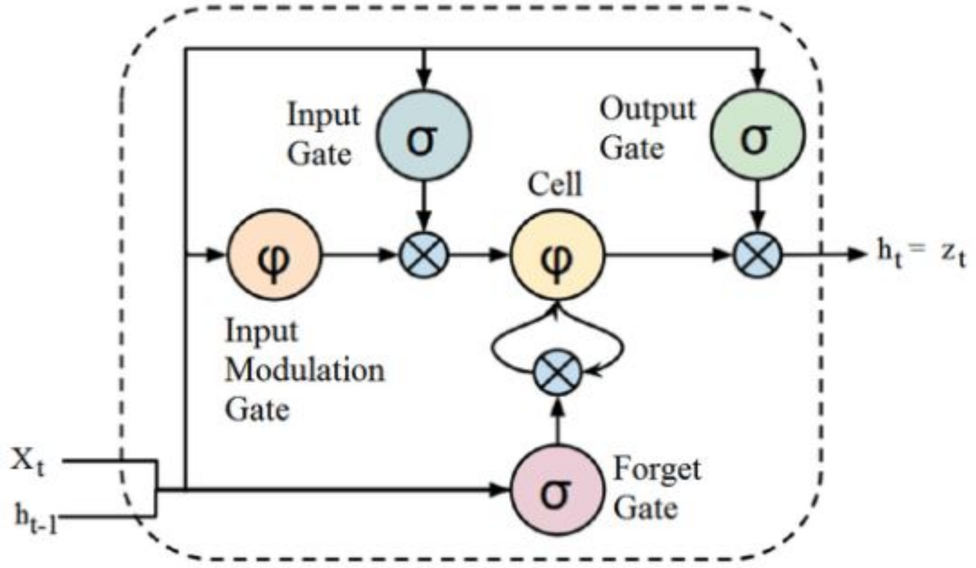
- The weight multiplying the current input *xt*, which is *u,* and the weight multiplying the previous output *yt-1*, which is *w*. **현재입력값 x와 가중치 u를 곱한 것을 이전결과값 y와 가중치 w를 곱한 것을 더한값을 이용하여 예측**
- This formula is like the exponential weighted moving average (EWMA) **지수가중이동평균과 유사함**

LSTM (Long Short Term Memory)

- A type of RNN that introduces the memory cell, a unit of computation that replaces traditional artificial neurons in the hidden layer of the network **RNN의 한종류로 은닉층안에 있는 뉴런 대신 기억셀을 소개**
- Solved the problem of long-term dependencies in a sequence. **연속수에서 장기의존성 문제를 해결**



Operation Gates



LSTM

- LSTM uses a set of gates to control the flow of information 게이트를 이용하여 정보의 흐름을 조절함
- LSTM is organized in cells which include several operations. 여러종류의 연산수행하는 셀들로 구성
  - Forget gate
  - Input gate
  - Output gate

Forget Gate (Remember Vector)

- Also called The output of the forget gate tells the cell state which information to forget by multiplying 0
  0을 곱해서 기억을 지우거나..
- If the output of the forget gate is 1, the information is kept in the cell state
  1을 곱해서 기억을 남기던가..

$$f_t = \sigma\left(W_f . [h_{t-1}, x_t] + b_f\right)$$

Input Gate

- Takes the previous output and the new input and passes them through another sigmoid layer. **이전 결과값과 새 입력값을 합쳐서 시그모이드 레이어로 보냄**
- Returns a value between 0 and 1. **0이나 1값을 반환함**

$$i_t = \sigma\left(W_i . [h_{t-1}, x_t] + b_i\right)$$

Internal State

- The previous state is multiplied by the forget gate and then added to the fraction of the new candidate allowed by the output gate. 이전 후보자는 포겟게이트와 곱해지고, 새로운 후보자는 입력값과 곱해져서 아웃풋 게이트로 보내짐

$$C_t = f_t * C_{t-1} + i_t * C_t$$

Candidate Vector

- Applying a hyperbolic tangent to the mix of input and previous output, this formula returns a candidate vector to be added to the internal state.
  이전 결과값과 현재 입력값을 합쳐서 탄젠트를 적용하여 후보벡터를 만듬

$$C_t = \tanh(W_C.[h_{t-1}, x_t] + \ b_C)$$

Output Gate

- Controls how much of the internal state is passed to the output 얼마나 많은 후보벡터를 출력으로 보낼지 조절
- Works in a similar way to the other gates 다른 게이트와 유사한 방법으로 작동

$$O_t = \sigma\left(W_o . [h_{t-1}, x_t] + b_o\right)$$

$$h_t = O_t * \tanh C_t$$

Libraries for RNN

```
from tensorflow.keras.models
import Sequential
```

```
from tensorflow.keras.layers
import Dense, Dropout, LSTM,
CuDNNLSTM,
```

CuDNNLSTM

- Implements kernels for large matrix operations on GPU using CUDA (Compute Unified Device Architecture) **GPU위에서 큰 행렬연산을 위한 커널을 제공**
- Designed for CUDA parallel processing. CUDA allows vector operations using GPU **병렬처리를 위해서 설계**
- Can’t run if there is no GPU. **GPU가 없으면 실행이 안됨**
- Faster time of execution **빠른 실행시간**

LSTM Model

```
model = Sequential()

model.add(LSTM(50, activation='relu',
input_shape=(timesteps,n_features)))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mse')
```

Scaling

```
#normalize dataset
scaler = MinMaxScaler(feature_range=(0,1))
pssngr_scaled = scaler.fit_transform(pssngr)
plt.plot(pssngr_scaled)
```

Train and Test Split

```
# split into train and test sets
train_size = int(len(pssngr)*(2/3)) #2/3
train, test = pssngr_scaled[:train_size], 
pssngr_scaled[train_size:]
len(train), len(test)
```

X and Y Split

```
x_train = []
y_train = []
for i in range(60, 1407):
    x_train.append(train[i-60:i, 0])
    y_train.append(train[i, 0])
x_train, y_train = np.array(x_train), np.array(y_train)
```

Reshape Input Values

```
x_train = np.reshape(x_train,
(x_train.shape[0], 1,
x_train.shape[1]))

x_test = np.reshape(x_test,
(x_test.shape[0], 1,
x_test.shape[1]))
```

- Reshape train x and test x to be [samples, time steps, features]
  훈련과 테스트셋의 모양을 바꿈

Prediction and Invert Predictions

- # make predictions
- y_pred = model.predict(x_train)
- prediction = model.predict(x_test)

- # invert predictions
- y_pred_inverse = scaler.inverse_transform(y_pred)
- y_train_inverse = scaler.inverse_transform([y_train]).T
- prediction_inverse = scaler.inverse_transform(prediction)
- y_test_inverse = scaler.inverse_transform([y_test]).T

Evaluation

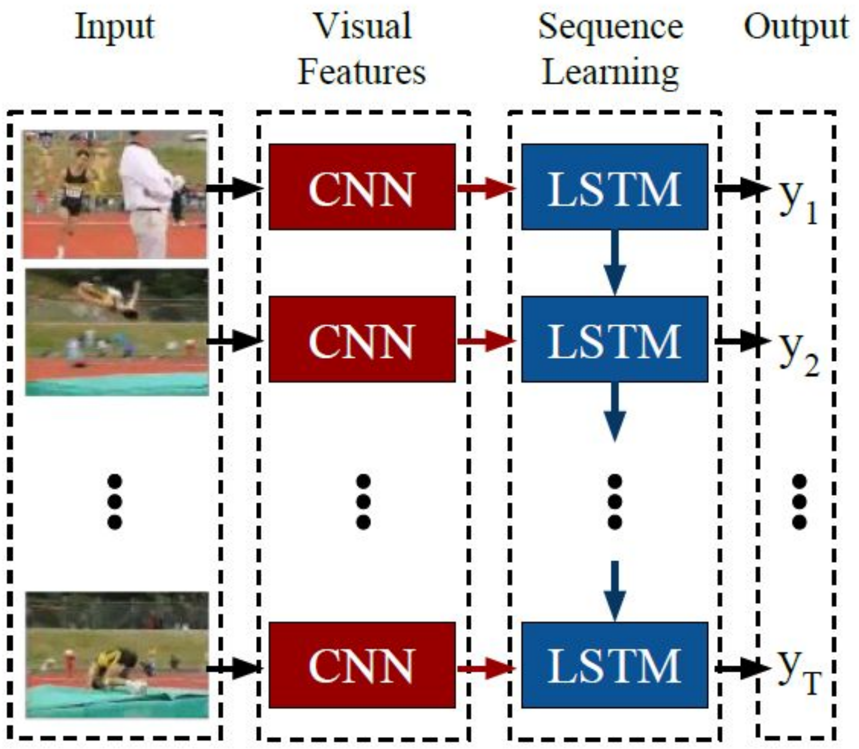
- `# calculate root mean squared error`
- `train_mse = np.sqrt(mean_squared_error(y_train_inverse[0], y_pred_inverse[0]))`
- `print('Train Score: %.2f RMSE' % (train_mse))`
- `test_mse = np.sqrt(mean_squared_error(y_test_inverse[0], prediction_inverse[0]))`
- `print('Test Score: %.2f RMSE' % (test_mse))`

Forecasting Plot

```
# plot baseline and predictions

plt.plot(scaler.inverse_transform(pssngr_scaled),
label='Original')

plt.plot(y_train_inverse, label='Predicted')

plt.plot(np.arange(96, 143), y_test_inverse)

plt.title('Air Passengers Prediction')

plt.legend()

plt.show()
```

CNN-LSTM Model

```
model = Sequential()

model.add(TimeDistributed(Conv1D(filters=64, kernel_size=1, activation='relu'), input_shape=(None, 2, 1)))

model.add(TimeDistributed(MaxPooling1D(pool_size=2)))

model.add(TimeDistributed(Flatten()))

model.add(LSTM(50, activation='relu'))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mse')
```



AirPassengers

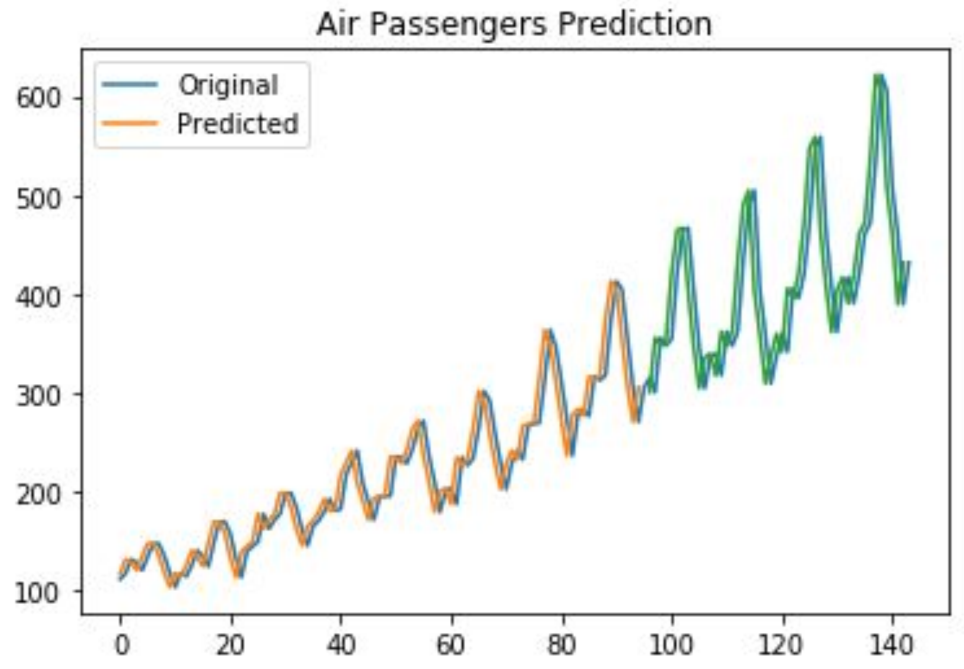
AirPassengers dataset includes monthly Airline Passenger Numbers 1949-1960.
Build a prediction model for Airline Passenger Number. **에어패씬저 데이터셋에는 1949년에서 1960년까지의 월별 승객수에 대한 데이터를 포함하고 있습니다. 승객수를 예측하는 모델을 만드시요.**

Exercise #7

- Use AirPassanger Dataset to build a prediction model using RNN
  - Scaling (MinMaxScaler between 0 and 1)
  - Split into train and test datasets
  - Prepare data using shift(1)
  - Reshape input data to be [samples, time steps, features]

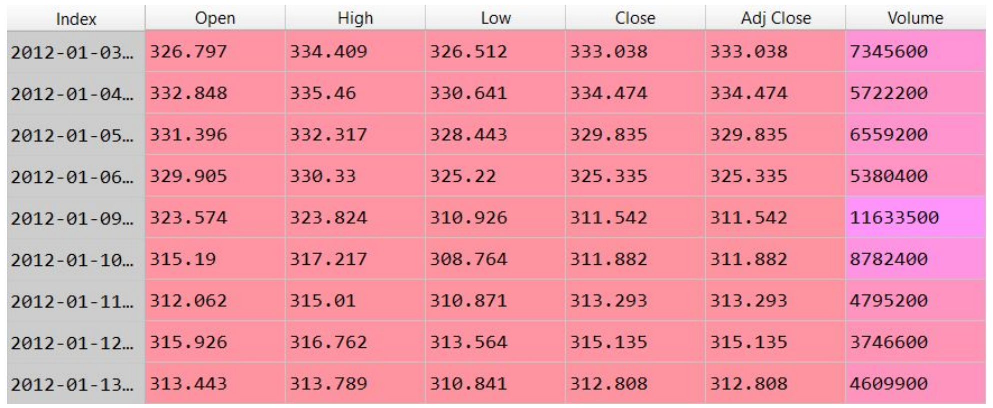
Exercise #7

- Build a model
  - 4 units of LSTM
  - 1 Dense layer
  - adam optimizer and mse loss
- Fit the model
  - 100 epochs and 1 batch size
- Prediction using test dataset
- Calculate the root mean squared error
- Visualize the result



Exercise #7

- Collect the Google stock price from yahoo finance.
  - January 2012 to 2019
- Check the null values df.isna().sum()
- Scaling (MinMaxScaler between 0 and 1)
- Preparing train and test sets with 60 timesteps
- Reshaping(x_train.shape[0], x_train.shape[1], 1)



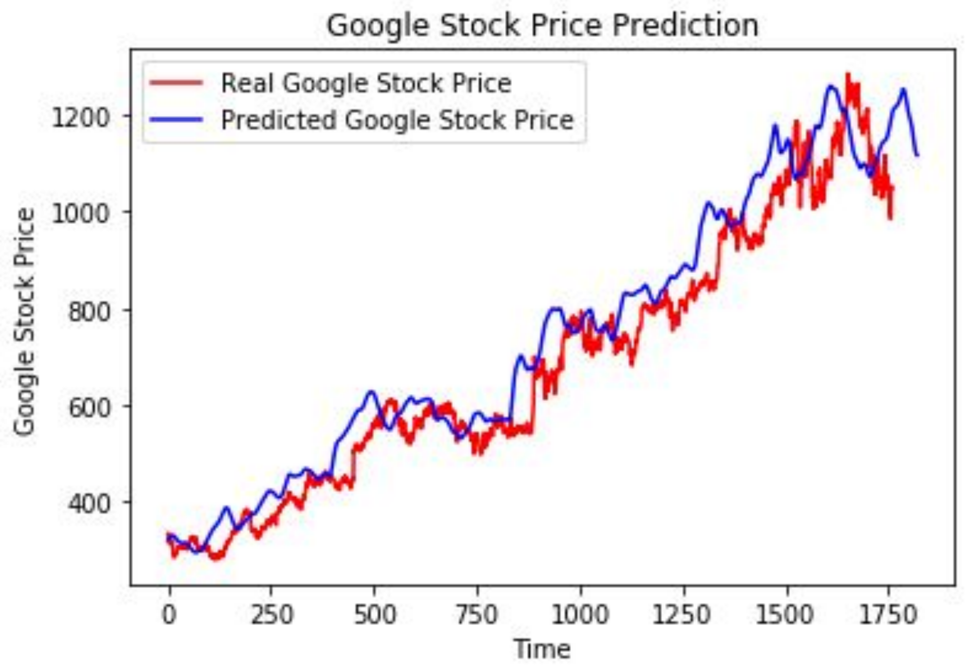
Index	Open	High	Low	Close	Adj Close	Volume
2012-01-03...	326.797	334.409	326.512	333.038	333.038	7345600
2012-01-04...	332.848	335.46	330.641	334.474	334.474	5722200
2012-01-05...	331.396	332.317	328.443	329.835	329.835	6559200
2012-01-06...	329.905	330.33	325.22	325.335	325.335	5380400
2012-01-09...	323.574	323.824	310.926	311.542	311.542	11633500
2012-01-10...	315.19	317.217	308.764	311.882	311.882	8782400
2012-01-11...	312.062	315.01	310.871	313.293	313.293	4795200
2012-01-12...	315.926	316.762	313.564	315.135	315.135	3746600
2012-01-13...	313.443	313.789	310.841	312.808	312.808	4609900

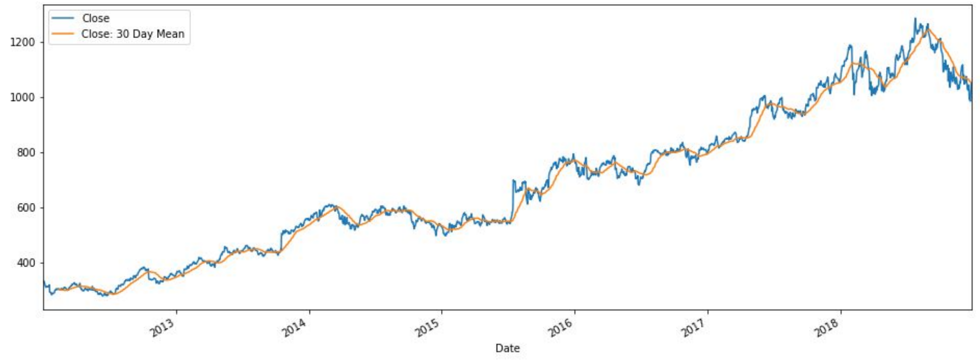
Exercise #7

- Building the RNN LSTM model using sequential model
  - 4 LSTM laysers with 50 units
  - Dropout right after LSTM layers - .2
  - Dense with 1 units
  - Compile with adam optimizer and mean_squared_error loss
  - Fit with 100 epochs and 32 batch size
- Building a RNN model without Dropout

Exercise #7

- Preparing test dataset with 2019 data
  - Concat with original dataset
  - Transform
  - Reshape
  - Predict
  - Inverse_transform
- Visualize the result
- Visualize the rolling mean (30 days)


Google Stock Price Prediction
Real Google Stock Price
Predicted Google Stock Price
Google Stock Price
Time


Close
Close: 30 Day Mean
Date