

도시가스 사용량 예측

오서영, 허지혜, 이수빈, 강수연

목차

1

도시가스 사용량 예측

2

단순 회귀 분석

3

불량 데이터 전처리

4

RNN 모델

목표) 1만 세대의 2009-01부터 2018-05까지 사용량으로부터
2018-06부터 2019-05까지 12개월 도시가스 사용량을 예측

train.csv

2009-01부터 2019-05까지 약 7만8천 세대의 월별 도시가스 사용량
데이터입니다.
불량데이터가 포함되어 있습니다.

test.csv

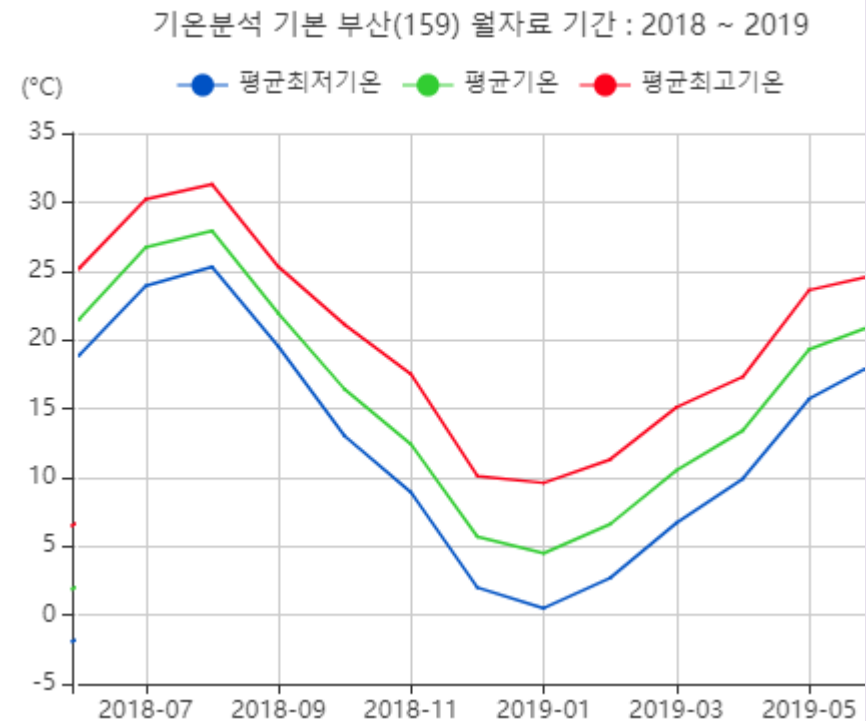
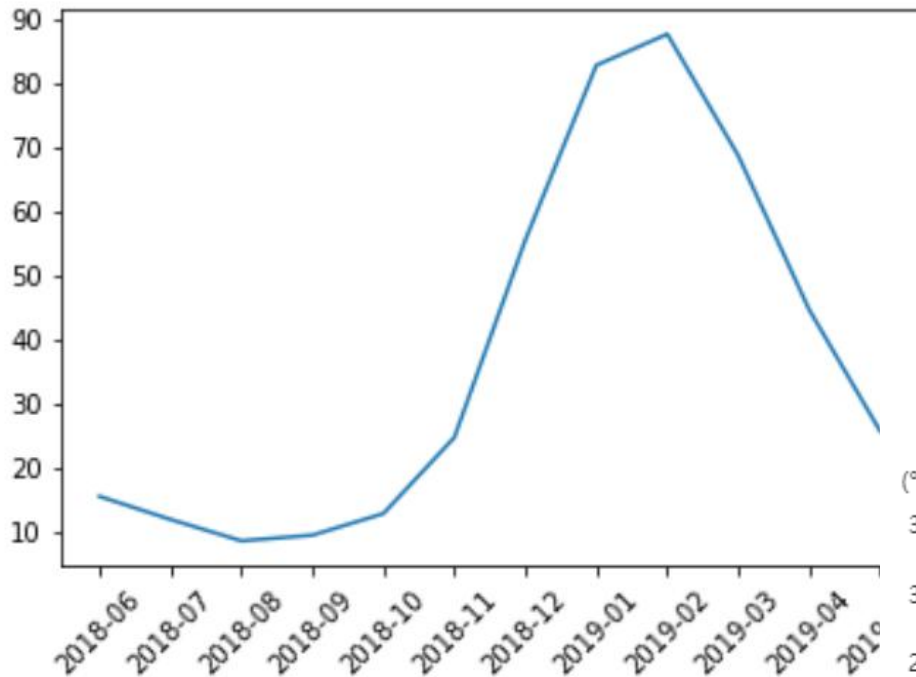
2009-01부터 2018-05까지 1만 세대의 월별 도시가스 사용량 데이터
입니다.
불량 데이터를 포함하지 않고 정상 데이터만 있습니다.

test

	2009-01	2009-02	2009-03	2009-04	2009-05	2009-06	2009-07	2009-08	2009-09	2009-10	...
id											
78587	74.13	93.89	79.09	59.32	29.66	19.77	14.83	14.83	14.83	10.87	...
78588	41.51	53.37	43.50	34.60	25.70	17.79	11.86	11.86	9.88	10.87	...
78589	82.03	86.97	81.07	58.33	41.52	19.77	9.88	12.85	9.88	14.83	...
78590	89.94	80.06	107.76	62.28	25.70	20.76	19.77	18.78	30.64	0.00	...
78591	17.79	19.76	20.76	26.69	28.67	32.62	20.76	9.88	10.87	11.86	...
...
88582	197.68	142.32	69.20	108.75	32.62	19.77	19.77	18.78	10.87	17.79	...
88583	134.42	132.44	76.12	66.24	33.61	21.75	15.81	15.81	12.85	16.80	...
88584	74.13	69.18	34.60	21.75	18.78	8.89	20.76	0.00	0.00	0.00	...
88585	24.71	25.69	19.77	13.84	8.89	8.89	8.89	8.89	10.87	6.92	...
88586	190.76	136.39	113.70	75.14	43.50	30.64	23.72	25.70	19.77	21.75	...

10000 rows × 113 columns

2018.06 ~ 2019.05 월평균 사용량, 월평균 기온 비교하기



월평균 사용량, 월평균 기온 상관관계 구하기

	temp	consumption
0	21.5	15.462132
1	26.7	11.865392
2	27.9	8.522864
3	21.9	9.410376
4	16.4	12.777364
5	12.4	24.660091
6	5.7	55.451120
7	4.5	82.680122
8	6.6	87.545635
9	10.5	68.595784
10	13.4	44.560283
11	19.3	25.623164

```
print(df.corr(method='pearson'))
```

```
          temp  consumption
temp      1.000000    -0.874495
consumption -0.874495     1.000000
```

-0.874495

-1.0 ~ -0.7 이므로,

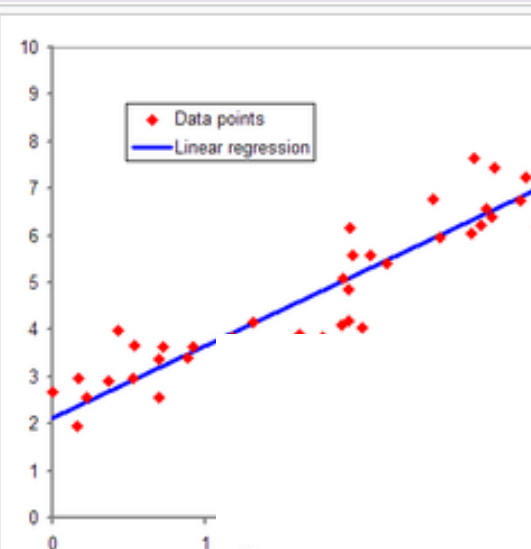
강한 음의 상관관계

단순 회귀 분석 : 종속 변수(y) 와 독립변수(x) 사이의 선형 관계를 파악하고 이를 예측에 활용하는 방법

$$y = Wx + b$$

- 하나의 독립변수에 기인하므로 단순회귀라고 한다
 - 독립변수(x) : 월 평균 온도
 - 종속 변수(y) : 월 평균 가스 사용량

단순 회귀 분석 한눈에 보기



독립변수 1개와

전자의 제곱
↑

"최소 제곱법"

$$\text{목적 함수 } E = \sum_{i=1}^n (y_i - \omega x_i - b)^2$$

$$\hookrightarrow \sum_{k=1}^n x_k y_k - \sum_{k=1}^n x_k \sum_{k=1}^n y_k$$

$$\omega =$$

$$\frac{\sum_{k=1}^n x_k^2 - \left(\sum_{k=1}^n x_k \right)^2}{\sum_{k=1}^n x_k^2 - \left(\sum_{k=1}^n x_k \right)^2} = 0$$

$$b =$$

$$\frac{\sum_{k=1}^n x_k^2 \sum_{k=1}^n y_k - \sum_{k=1}^n x_k y_k \sum_{k=1}^n x_k}{\sum_{k=1}^n x_k^2 - \left(\sum_{k=1}^n x_k \right)^2} = 0$$

단순 회귀 구현

1) 패키지 불러오기 & 2) 데이터셋 만들기

1) import packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

2) load datasets

```
# y_train : BUSAN_gas_consumption from 2009-01 to 2018-05 (113 months)
# temp_train_orig : BUSAN_average_temperature from 2009-01 to 2018_12
# temp_test_orig : BUSAN_average_temperature from 2018-01 to 2019_12

test = pd.read_csv("test.csv", encoding="utf-8", index_col=0)
temp_train_orig = pd.read_csv("ta_20200329144537.csv", encoding = "cp949" , skiprows =7, usecols = ['평균기온(℃)'])
temp_test_orig = pd.read_csv("ta_20200329180212.csv" , encoding = "cp949" , skiprows = 7 , usecols = ['평균기온(℃)'])

y_train = test.values[:,:]
sample_submission = pd.read_csv("sample_submission.csv", encoding="utf-8", index_col=0)
```

-> 2009.01 ~ 2018.05 , 10000세대의 사용량 데이터

3) 데이터 확인 & 인덱싱

```
test.head()
```

	2009-01	2009-02	2009-03	2009-04	2009-05	2009-06	2009-07	2009-08	2009-09	2009-10	...	2017-08	2017-09	2017-10	2017-11	2017-12	2018-01	2018-02	2018-03	2018-04	2018-05
id																					
78587	74.13	93.89	79.09	59.32	29.66	19.77	14.83	14.83	14.83	10.87	...	12.84	9.88	14.82	27.66	66.20	83.98	93.86	79.06	53.36	33.60
78588	41.51	53.37	43.50	34.60	25.70	17.79	11.86	11.86	9.88	10.87	...	10.86	8.89	9.88	23.71	44.46	82.01	92.88	79.06	55.34	26.68
78589	82.03	86.97	81.07	58.33	41.52	19.77	9.88	12.85	9.88	14.83	...	7.90	7.90	10.86	20.75	59.28	81.02	92.88	69.18	38.54	14.82
78590	89.94	80.06	107.76	62.28	25.70	20.76	19.77	18.78	30.64	0.00	...	7.90	9.88	12.84	8.89	34.58	64.22	64.22	64.23	49.41	39.53
78591	17.79	19.76	20.76	26.69	28.67	32.62	20.76	9.88	10.87	11.86	...	14.82	13.83	15.80	23.71	60.27	83.98	82.01	88.94	57.32	22.73

```
temp_train_orig.head()
```

평균기온(°C)

0 3.5

1 8.3

2 9.8

3 14.3

4 18.6

```
# temp_train : indexing tempurature from 2009-01 to 2018-05 (113 months)  
# temp_train : indexing tempurature from 2018_06 to 2019-05 (113 months)  
temp_train = temp_train_orig.values[:-7,:].T  
temp_test = temp_test_orig.values[5:-7].T
```

```
# explore datasets  
print("Shape of y_train :", y_train.shape)  
print("Shape of temp_train :", temp_train.shape)  
print("Shape of temp_test :", temp_test.shape)  
print("Type of y_train :", type(y_train))  
print("Type of temp_train :", type(temp_train))  
print("Type of temp_test :", type(temp_test))
```

```
Shape of y_train : (10000, 113)  
Shape of temp_train : (1, 113)  
Shape of temp_test : (1, 12)  
Type of y_train : <class 'numpy.ndarray'>  
Type of temp_train : <class 'numpy.ndarray'>  
Type of temp_test : <class 'numpy.ndarray'>
```

4) 모델 구현하기

4) Linear Regression Model

```
model = LinearRegression(fit_intercept=True)  
model = model.fit(temp_train.T, y_train.T)
```

5) 결과 확인

5) Result

```
# explore W & b
W = model.coef_
b = model.intercept_
b = b.reshape(10000,1)

print("W =", W)
print("b =", b)
print()
print("Shape of W :", W.shape)
print("Shape of b :", b.shape)
```

```
W = [[-3.06201993]
      [-2.32832629]
      [-3.08658326]
      ...
      [-3.03092111]
      [-1.11216643]
      [-5.91888552]]
b = [[ 84.66048907]
      [ 74.10395583]
      [ 83.46083752]
      ...
      [ 75.99082708]
      [ 33.03855055]
      [149.90076898]]
```

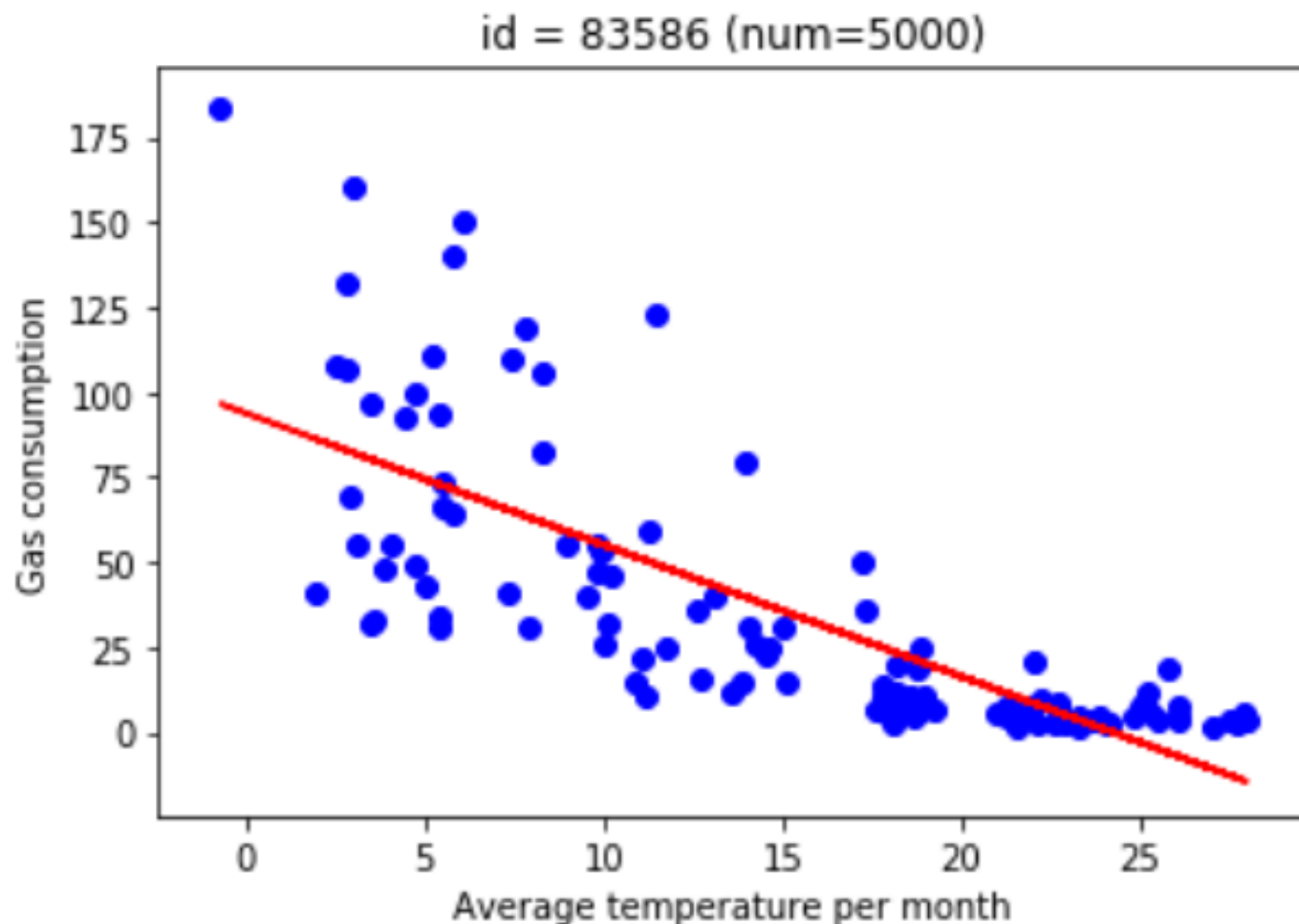
```
Shape of W : (10000, 1)
Shape of b : (10000, 1)
```

6) 세대 별 결과 확인하기 (함수 사용)

6) Visualization of an id

```
def plot(W,b,temp,y,n) :  
    # propagation  
    Z = np.dot(W,temp) + b  
  
    nth_temp = temp[0]  
  
    # indexing n_th data  
    nth_Z = Z[n-1]  
    nth_y = y[n-1]  
  
    # visualization  
    plt.plot(nth_temp, nth_Z, color = "red")  
    plt.scatter(nth_temp, nth_y, color = "blue")  
    # plt.plot(nth_temp, nth_y, 'o', color = "blue") # same as above  
    plt.title("id = {0} (num={1})".format(78586 + n, n))  
    plt.xlabel("Average temperature per month")  
    plt.ylabel("Gas consumption")  
    plt.show()
```

```
plot(W, b, temp_train, y_train, n = 5000) # you can change "n"
```



월 평균 가스 사용료와 온도는 **반비례** 관계로 보인다

7) 구현한 모델로 예측하여 제출물 만들기

7) Predict & Make a submission

```
# predict gas consumption from 2018-06 to 2019-05 by using linear model  
y_test = np.dot(W,temp_test) + b
```

```
# make submission  
submission = pd.DataFrame(data=y_test, columns=sample_submission.columns, index=sample_submission.index)  
submission.to_csv('submission.csv')  
print(submission)
```

	2018-06	2018-07	2018-08	2018-09	2018-10	2018-11	₩
id							
78587	18.827061	2.904557	-0.769867	17.602253	34.443362	46.691442	
78588	24.044941	11.937644	9.143652	23.113610	35.919405	45.232710	
78589	17.099297	1.049064	-2.654836	15.864664	32.840872	45.187205	
78590	22.180159	7.373932	3.957110	21.041219	36.701652	48.091058	
78591	20.055745	5.416693	2.038451	18.929664	34.413277	45.674086	
...	
88582	25.036166	-2.949153	-9.407304	22.883449	52.483306	74.010475	
88583	23.899870	-3.330538	-9.614479	21.805223	50.606616	71.553084	
88584	10.826023	-4.934767	-8.571872	9.613655	26.283721	38.407405	
88585	9.126972	3.343707	2.009107	8.682106	14.799021	19.247687	
88586	22.644730	-8.133474	-15.236137	20.277176	52.831046	76.506589	











모델의
성능은
RMSE 로
측정된다

**Root
Mean
Square
Error**

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

실시간 랭킹



순위	이름	제출횟수	최종제출일	점수
1	 민수	1	2일 전	19,48722
2	 김병천	2	하루 전	19,51332
3	 gyuwonable	1	3일 전	19,53804
4	 김민중	2	35분 전	22,39489
5	 1등하겠어요	6	한 시간 전	24,23303
6	 관리자	3	한 시간 전	24,23303
7	 오서영	1	10시간 전	26,07178
8	 민원	1	4일 전	49,07211
9	 양승환	1	4일 전	49,07211
10	 saturn	1	5일 전	49,07211

RMSE 오차 : 26.07178

train.csv

2009-01부터 2019-05까지 약 7만8천 세대의 월별 도시가스 사용량 데이터입니다.

불량데이터가 포함되어 있습니다.

- 불량 데이터 예

- 1) 사용량이 있다가 계량기 고장으로 0으로 떨어지는 경우**
- 2) 거주지 이전으로 인해 사용량이 0인 경우**
- 3) 고객이 사용량을 전화나 문자로 알려줄 때 사용량을 속여 매월 일정한 값을 제공하는 경우**
- 4) 도시가스 용도가 잘못 기재되어 월별 사용량이 비정상적으로 큰 경우**

1) 0 값 비율 확인

```
# train1 : numpy array of training set  
# num_bad_zero : the number of bad zero data  
  
train1 = train_orig.values[:, :]  
num_bad_zero = sum(sum(train1==0))  
print("The number of zero value is" , num_bad_zero)  
print("Its rate is" , num_bad_zero * 100 / (train.shape[0] * train.shape[1]) , "%")
```

The number of zero value is 672038
Its rate is 7.510063139073588 %

2) 결측치가 20% 이상인 세대 데이터 제거

결측치 비율에 따른 변수 제거

- 결측치가 10%이하인 경우 : 해당 표본을 제거하거나 imputation
- 결측치가 20%이상인 경우 : 해당 변수 제거 or imputation

```
# Convert zero-values to NAN  
# n2 : the number of zero-values  
train=train_orig[train_orig>0]  
n1 = train.T.isnull().sum()  
n2 = np.array(n1).reshape(78587,1)
```

```
# Delete id-data when the id has zero-values over 20%.  
train["num"] = n2  
train = train[train['num']<25]  
del train['num']
```

```
print("The shape of preprocessed data is", train.shape)
```

The shape of preprocessed data is (71588, 125)

약 7000개
세대 데이터
제거

3) 0을 평균값으로 바꾸기

```
# Calculate the average of monthly consumption
```

```
mean=train.mean()
```

```
mean
```

```
# Convert 'NA' to 'mean'
```

```
train_nonzero=train.fillna(mean)
```

```
train_nonzero
```

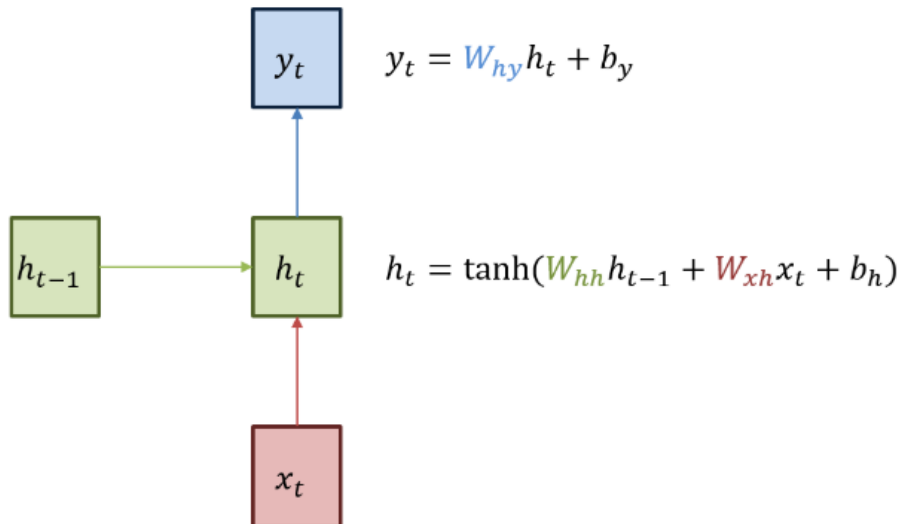
	2009-01	2009-02	2009-03	2009-04	2009-05	2009-06	2009-07	2009-08	2009-09	2009-10
id										
0	26.680000	11.860000	19.770000	14.830000	7.900000	2.960000	5.930000	4.940000	4.94	4.94
1	107.730000	77.090000	79.090000	4.940000	48.440000	22.740000	33.610000	23.720000	23.72	25.70
2	94.880000	90.930000	78.100000	66.240000	38.550000	25.700000	27.680000	26.690000	18.78	15.81
3	115.487984	114.181544	149.290000	51.410000	6.920000	8.890000	2.960000	2.960000	1.97	3.95
4	97.850000	97.850000	70.190000	60.310000	31.630000	25.700000	20.760000	22.740000	20.76	20.76

RMSE 오차 : 약 44

Test 데이터를 사용하여 간단한 RNN 구현
 2018-06부터 2019-05까지 데이터가 없으므로
 2009-01부터 2018-05까지 데이터로만 학습

RNN 이란?

히든 노드가 방향을 가진 엣지로 연결돼 순환구조 이룸
 -> 순차적으로 등장하는 데이터 처리에 적합한 모델



녹색은 히든 state, 빨강은 인풋,
 파랑은 아웃풋
 현재 상태의 히든 state H_t 는
 직전 시점의 히든 state H_{t-1} 를
 받아 갱신됩니다.

RNN 모델 구현 & 오차 확인

```
model = Sequential()  
model.add(LSTM(128, input_shape = (1, 101), return_sequences=True))  
model.add(LSTM(128, return_sequences=False))
```

```
model.add(Dense(128, activation = "relu"))  
model.add(Dropout(0.5))  
model.add(Dense(64, activation = "relu"))  
model.add(Dropout(0.5))  
model.add(Dense(12, activation = "linear"))
```

```
losses=tf.compat.v1.losses.log_loss  
model.compile(loss = 'mse', optimizer='adam')
```

```
hist_model = model.fit(x,y,  
                        epochs = 100,  
                        batch_size = 256,  
                        verbose = 2,  
                        validation_split=0.2)
```


17	 동규	8	3시간 전	21,96490
18	 수학천재	1	5일 전	22,77777
19	 최진규	13	12시간 전	23,10527
20	 <u>오서영</u>	2	3분 전	<u>23,38848</u>
21	 saturn	3	5일 전	23,67816
22	 안녕세상아	5	5일 전	24,28170

RMSE 오차 : 23.38848

RNN 모델이
타 데이터(온도)를 이용한 선형모델보다
성능이 좋다

참고자료

1) RNN과 LSTM을 이해해보자

<https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/>

2) 데이터의 탐색 - 결측치와 이상치

<https://hodubab.tistory.com/297>

3) NIMS 풀림 - 도시가스 사용량 예측

<https://icim.nims.re.kr/platform/question/16>

4) 기상자료개방포털

<https://data.kma.go.kr/stcs/grnd/grndTaList.do?pgmNo=70>

