빅데이터 프로젝트 보고서

인천 지역 온도에 따른 모기밀집도

201844089 이기원

GITHUB:
HTTPS://GITHUB.COM/LKW0820/BIG_DATA_FINAL.GIT

설명

주제: 인천 지역 모기밀집도 확인 및 예측

사용 데이터: 2021 인천지역 모기 밀집도, 강화지역 온도, 인천지역 온도

데이터 수집 방법: data.go.kr, 기상청

분석할 내용: 온도, 지역, 모기 종류에 따른 밀집도 분석 및 온도에 따른 모기 밀집도 예측

차례:

- 1 모기 종류 별 밀집도(인천 지역, 연중주수 입력)
- 2 모기 종류 별 밀집도 애니메이션 그래프
- 3 데이터 정보
- 4 선형회귀를 이용한 밀집도 예측
- 5 학습셋과 테스트셋을 이용한 선형회귀
- 6 딥러닝을 이용하여 예측(교차 검증X) + 그래프
- 7 딥러닝을 사용한 예측(교차검증 O) + 그래프
- 8 부족한점

1. 모기 종류 별 밀집도(인천 지역, 연중주수 입력)

코드

```
1. import pandas as pd
2. import matplotlib.pyplot as plt
3. import seaborn as sns
4.
5. plt.rc('font', family='NanumBarunGothic')
6.
7. df = pd.read_csv('/content/인천광역시_보건환경연구원-
   모기밀도조사_20211231.csv', encoding = 'CP949', header=0)
8. Text=input('연중주수를 입력해주세요(14주~44주): ')
9. #정렬
10.
11.dfd=df.groupby(['연중주수'])
12.df4=dfd.get_group(Text)
13.
14.name=list(df4.columns.values)[4:] #모기 종류
15.
16.fig, ax = plt.subplots(figsize=(20,8))
17.for i in name:
18.   sns.scatterplot(data=df4, x='장소',y=i, s=200)
19.
20.ax.set_ylabel('밀집도')
```

코드 설명

1~3 필요한 라이브러리 추가

5 글꼴 변환

7 CSV불러오기

8 밀집도를 보고 싶은 연중주수 입력

11 데이터프레임을 연중주수로 그룹화

12 입력된 연중주수 그룹선택

14 모기종류 리스트에 저장

16 그래프 사이즈 설정

17~18 모기종류를 종류 별로 그래프에 표시

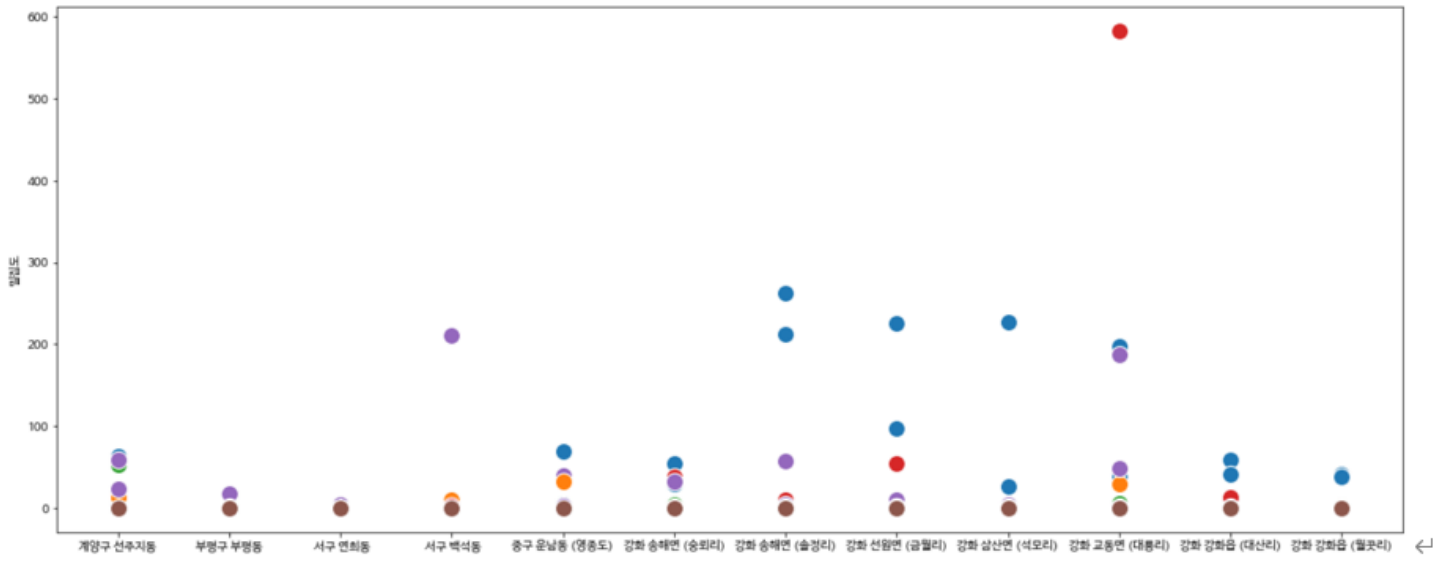
20 y축 이름을 밀집도로 설정

1. 모기 종류 별 밀집도(인천 지역, 연중주수 입력)

실행결과

```
... 연중주수를 입력해주세요(14주~44주):
```

```
연중주수를 입력해주세요(14주~44주): 33주
Text(0, 0.5, '밀집도')
```



2 모기 종류 별 밀집도 애니메이션 그래프

코드

```
1. import plotly.express as px
2. import numpy as np
3. 
4. 
5. 
6. 
7. plt.rc('font', family='NanumBarunGothic') 
8. 
9. df = pd.read_csv('/content/인천광역시_보건환경연구원-
   모기밀도조사_20211231.csv', encoding = 'CP949', header=0)
10. df1 = pd.read_csv('/content/인천 ta_20221201172941.csv', encoding
   = 'CP949', header=0)
11. df2 = pd.read_csv('/content/강화 ta_20221204010051.csv', encoding
   = 'CP949', header=0)
12. 
13. print('모기 종류: 얼룩날개모기류, 이나토미집모기, 반점날개집모기, 동양집모
   기, 빨간집모기, 작은빨간집모기')
14. print('            줄다리집모기, 노랑늪모기,반점날개늪모기, 흰줄숲모기, 금
   빛숲모기, 한국숲모기, 토고숲모기')
15. print('            등줄숲모기, 큰검정등모기, 금빛어깨숲모기기')
16. Text1 = input("모기 종류를 입력하세요: ")
17. Text2 = int(input("밀집도를 예측하고 싶은 온도를 입력하세요: "))
18. 
19. df['온도']=0
20. for i in np.arange(0,31): #30 은 4 월부터 10 월까지 30 주
21.   df['온도'].iloc[12*i:(12*i)+5] = df1['평균기온(℃)'].iloc[7*i:7*(
   i+1)].mean() #인천지역에 평균 온도
22.   df['온도'].iloc[5+(12*i):12+(12*i)] = df2['평균기온(℃)'].iloc[7*
   i:7*(i+1)].mean()#강화지역에 평균 온도
23. 
24. fig = px.scatter(df, x='장소', y='온도', animation_frame='연중주순'
   ,range_y=[df['온도'].min(),df['온도'].max()], size=Text1)
25. 
26. fig.show()
```

코드 설명

1~2 필요한 라이브러리 추가

7 글꼴 설정

9~11 필요한 CSV추가 9 2021년 인천지역 모기 밀집도, 10 2021년 4~10월 인천지역 온도, 11 2021년 4~10월 강화지역 온도

13~15 모기 종류 프린트

16 모기 종류 입력

17 16에서 입력한 모기 종류의 밀집도를 예측하고 싶은 온도

19 데이터프레임에 열(온도) 추가

20~22 데이터프레임에 평균 온도 추가

24~26 그래프 표시

2 모기 종류 별 밀집도 애니메이션 그래프

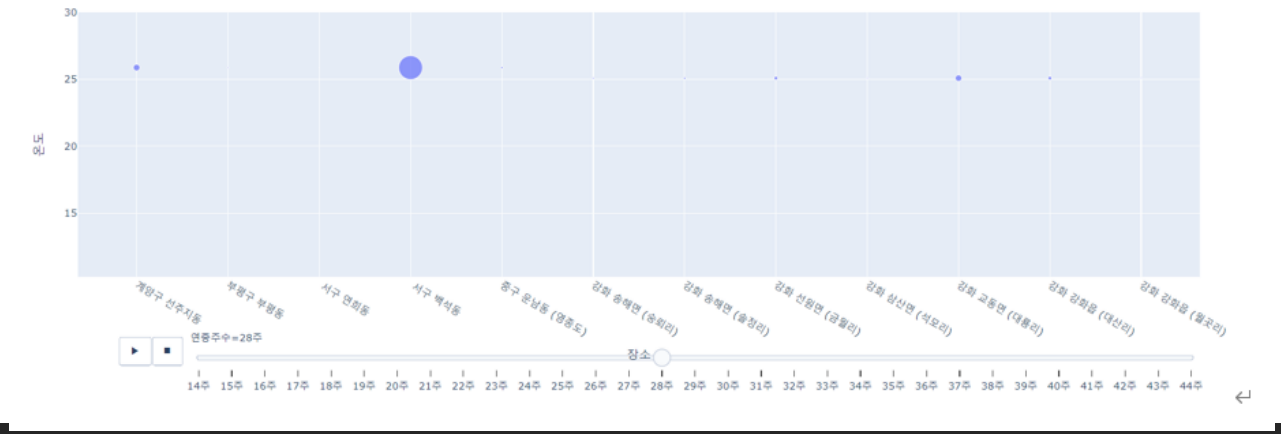
실행결과

```
모기 종류: 얼룩날개모기류, 이나토미집모기, 반점날개집모기, 동양집모기, 빨간집모기, 작은빨간집모기
        줄다리집모기, 노랑늪모기,반점날개늪모기, 흰줄숲모기, 금빛숲모기, 한국숲모기, 토고숲모기
        등줄숲모기, 큰검정들모기, 금빛어깨숲모기기
모기 종류를 입력하세요: 빨간집모
```

```
모기 종류: 얼룩날개모기류, 이나토미집모기, 반점날개집모기, 동양집모기, 빨간집모기, 작은빨간집모기
        줄다리집모기, 노랑늪모기,반점날개늪모기, 흰줄숲모기, 금빛숲모기, 한국숲모기, 토고숲모기
        등줄숲모기, 큰검정들모기, 금빛어깨숲모기기
모기 종류를 입력하세요: 빨간집모기
밀집도를 예측하고 싶은 온도를 입력하세요:
```

```
모기 종류를 입력하세요: 빨간집모기
밀집도를 예측하고 싶은 온도를 입력하세요: 26
/usr/local/lib/python3.8/dist-packages/pandas/core/indexing.py:1732: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```



3 데이터 정보

코드

```
1. df.info() #df 의 정보
2. df.dtypes #df 의 타입
3. df.head() #df 의 데이터확인
```

실행결과

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 371 entries, 0 to 370
Data columns (total 21 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   월        371 non-null    int64
 1   주        371 non-null    object
 2   연중주수     371 non-null    object
 3   장소       371 non-null    object
 4   얼룩날개모기류  371 non-null    int64
 5   이나토미집모기  371 non-null    int64
 6   반점날개집모기  371 non-null    int64
 7   동양집모기    371 non-null    int64
 8   빨간집모기    371 non-null    int64
 9   작은빨간집모기  371 non-null    int64
 10  줄다리집모기   371 non-null    int64
 11  노랑늪모기    371 non-null    int64
 12  반점날개늪모기  371 non-null    int64
 13  흰줄숲모기    371 non-null    int64
 14  금빛숲모기    371 non-null    int64
 15  한국숲모기    371 non-null    int64
 16  토고숲모기    371 non-null    int64
 17  등줄숲모기    371 non-null    int64
 18  큰검정등모기   371 non-null    int64
 19  금빛어깨숲모기  371 non-null    int64
 20  온도       371 non-null    float64
dtypes: float64(1), int64(17), object(3)
memory usage: 61.0+ KB
```

df.dtypes

```
월             int64
주            object
연중주수         object
장소           object
얼룩날개모기류       int64
이나토미집모기       int64
반점날개집모기       int64
동양집모기         int64
빨간집모기         int64
작은빨간집모기       int64
줄다리집모기        int64
노랑늪모기         int64
반점날개늪모기       int64
흰줄숲모기         int64
금빛숲모기         int64
한국숲모기         int64
토고숲모기         int64
등줄숲모기         int64
큰검정등모기        int64
금빛어깨숲모기       int64
온도          float64
dtype: object
```

3 데이터 정보

실행결과

df.head()↵

	월	주	연중주수	장소	얼룩날개모기류	에니토미 집모기	반점날개 집모기	동양집모기	빨간집모기	작은빨간 집모기	...	노랑늪모기	반점날개 늪모기	흰줄숲모기	금빛숲모기	한국숲모기	토고숲모기	등줄숲모기	큰검정등 모기	금빛어깨 숲모기	온도
0	4	1주	14주	계양구 선주지동	0	0	0	0	16	0	...	0	0	0	0	0	0	0	0	0	13.6
1	4	1주	14주	부평구 부평동	0	0	0	0	7	0	...	0	0	0	0	0	0	0	0	0	13.6
2	4	1주	14주	서구 연희동	0	0	0	0	2	0	...	0	0	0	0	0	0	0	0	0	13.6
3	4	1주	14주	서구 백석동	0	0	0	0	11	0	...	0	0	0	0	0	0	0	0	0	13.6
4	4	1주	14주	중구 운남동(영종도)	1	0	0	0	15	0	...	0	0	0	0	0	0	0	0	0	13.6

5 rows × 21 columns ↵

4 선형회귀를 이용한 밀집도 예측

코드

```
1. import tensorflow as tf
2. from sklearn import linear_model
3.
4. reg = linear_model.LinearRegression()
5. print(Text1)
6. print("계양구 선주지동, 부평구 부평동, 서구 연희동, 서구 백석동, 중구 운남동(영종도)")
7. print("강화 송해면(숭뢰리), 강화 송해면(솔정리), 강화 선원면(금월리), 강화 삼산면(석모리)")
8. print("강화 교동면(대룡리), 강화 강화읍(대산리), 강화 강화읍(월곶리)")
9. Text3 = input("지역을 입력하세요: ")
10.#y= 밀집도 x = 온도
11.grouped=df.groupby(['장소'])
12.g1=grouped.get_group(Text3)[Text1]
13.g2=grouped.get_group(Text3)['온도']
14.x=np.array(g2).reshape(31,1) #온도 6
15.y=np.array(g1) #밀집도
16.
17.#학습
18.reg.fit(x,y)
19.#예측값
20.print('선형 회귀모델: ',Text2, '℃ 일때 밀집도 예측: ',reg.predict([[Text2]]))
21.
22.# 학습 데이터와 y 값을 산포도로 그린다.
23.plt.scatter(x, y, color='black')
24.plt.xlabel('온도(℃)')
25.plt.ylabel('밀집도')
26.
27.# 학습 데이터를 입력으로 하여 예측값을 계산한다.
28.y_pred = reg.predict(x)
29.
30.# 학습 데이터와 예측값으로 선그래프로 그린다.
31.# 계산된 기울기와 y 절편을 가지는 직선이 그려진다.
32.plt.plot(x, y_pred, color='blue', linewidth=3)
33.plt.show()
```

코드설명

1~2 필요한 라이브러리 추가

4 선형회귀 모델 생성

5 예측할 모기 종류 출력

6~9 지역을 출력하고 예측할 지역 입력

11 데이터프레임 지역별로 그룹화

12 y축 설정

13 x축 설정

14 x축 데이터

15 y축 데이터

18 선형회귀 모델 학습

20 예측값 출력

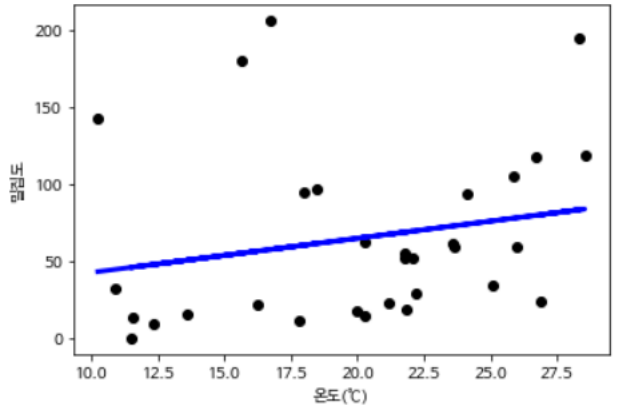
23~33 실제값(점)과 예측값(선)을 그래프로 표현

4 선형회귀를 이용한 밀집도 예측

실행결과

```
빨간집모기
계양구 선주지동, 부평구 부평동, 서구 연희동, 서구 백석동, 중구 운남동(영종도)
강화 송해면(숭뢰리), 강화 송해면(솔정리), 강화 선원면(금월리), 강화 삼산면(석모리)
강화 교동면(대룡리), 강화 강화읍(대산리), 강화 강화읍(월곳리)
지역을 입력하세요:
```

```
빨간집모기
계양구 선주지동, 부평구 부평동, 서구 연희동, 서구 백석동, 중구 운남동(영종도)
강화 송해면(숭뢰리), 강화 송해면(솔정리), 강화 선원면(금월리), 강화 삼산면(석모리)
강화 교동면(대룡리), 강화 강화읍(대산리), 강화 강화읍(월곳리)
지역을 입력하세요: 계양구 선주지동
선형 회귀모델: 26 ℃ 일때 밀집도 예측: [78.18490953]
```



5 학습셋과 테스트셋을 이용한 선형회귀

코드

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split


reg = linear_model.LinearRegression()
print(Text1)

#x= 온도 y = 밀집도
grouped=df.groupby(['장소'])
g1=grouped.get_group(Text3)[Text1]
#.drop(['온도','월','주','연중주수', '장소'], axis=1)
g2=grouped.get_group(Text3)['온도']
y=np.array(g1)
x=np.array(g2).reshape(31,1)

#학습셋과 테스트셋 구분
X_train, X_test, Y_train, Y_test = train_test_split(x,y,test_size
=0.3)

#학습
reg.fit(X_train,Y_train)
#예측값
print('선형 회귀모델, 학습셋 테스트셋 사용용: ',Text2, '℃ 일때 밀집도 예
측: ',reg.predict([[Text2]]))


# 학습 데이터와 y 값을 산포도로 그린다.
plt.scatter(X_test, Y_test, color='black')
plt.xlabel('온도(℃)')
plt.ylabel('밀집도')

# 학습 데이터를 입력으로 하여 예측값을 계산한다.
y_pred = reg.predict(X_test)

#학습 데이터와 예측값으로 선그래프로 그린다.
#계산된 기울기와 y 절편을 가지는 직선이 그려진다.
plt.plot(X_test, y_pred, color='blue', linewidth=3)
plt.show()
```

5 학습셋과 테스트셋을 이용한 선형회귀

코드 설명

1~3 필요한 라이브러리 추가↵

8 선형회귀 모델 생성↵

9 예측할 모기 종류 출력↵

12 데이터프레임 장소별로 그룹화↵

13 y축 설정 (입력한 지역의 입력한 모기 밀집도)↵

15 x축 설정 (입력한 지역의 온도)↵

16 x축 데이터↵

17 y축 데이터↵

20 학습셋과 테스트셋 설정↵

23 학습↵

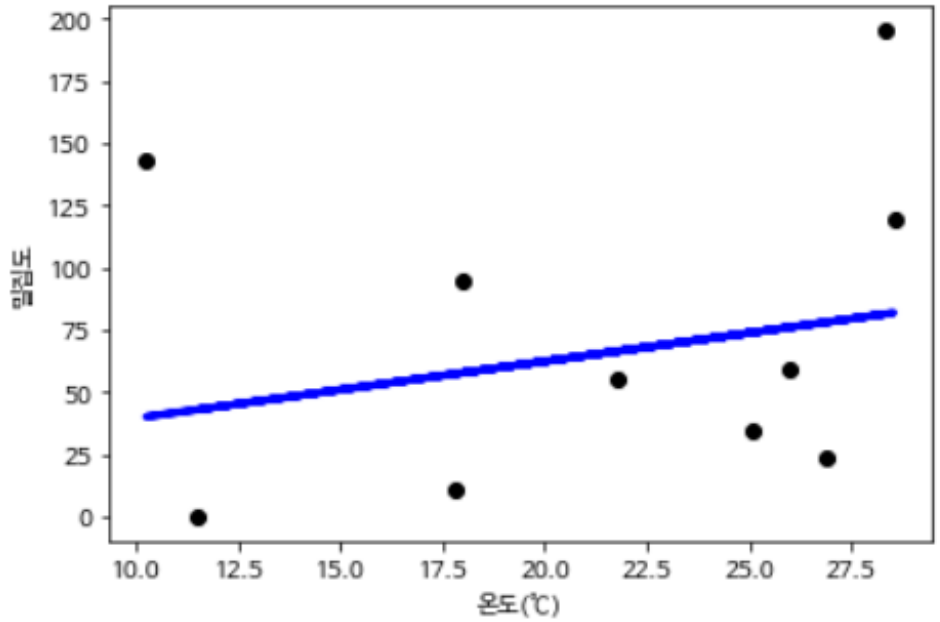
25 예측값 출력↵

28~38 실제값(점)과 예측값(선) 그래프로 표시↵

5 학습셋과 테스트셋을 이용한 선형회귀

실행결과

```
빨간집모기
선형 회귀모델, 학습셋 테스트셋 사용용:  26 ℃ 일때 밀집도 예측:  [75.94620628]
```



6 딥러닝을 이용하여 예측(교차검증X) + 그래프

코드

```
1. print('교차검증 X')
2. #딥러닝모델
3. model1 = Sequential()
4. model1.add(Dense(2,input_dim=1,activation='relu'))
5. model1.add(Dense(1,activation='relu'))
6. model1.compile(loss='mean_squared_error',
7.                optimizer='adam',
8.                metrics=['accuracy'])
9.
10.model1.fit(X_train,Y_train, epochs=30, batch_size=1)
```

코드설명

3 딥러닝 모델 생성

4~5 딥러닝 모델 설정

6 딥러닝 모델 컴파일

10 딥러닝 모델 학습

6 딥러닝을 이용하여 예측(교차검증X) + 그래프

실행결과

```
21/21 [==============================] - 0s 2ms/step - loss: 4561.4385 - accuracy: 0.0000e+00
Epoch 20/30
21/21 [==============================] - 0s 2ms/step - loss: 4507.4829 - accuracy: 0.0000e+00
Epoch 21/30
21/21 [==============================] - 0s 2ms/step - loss: 4450.6660 - accuracy: 0.0000e+00
Epoch 22/30
21/21 [==============================] - 0s 2ms/step - loss: 4398.3149 - accuracy: 0.0000e+00
Epoch 23/30
21/21 [==============================] - 0s 1ms/step - loss: 4343.0312 - accuracy: 0.0000e+00
Epoch 24/30
21/21 [==============================] - 0s 1ms/step - loss: 4291.7749 - accuracy: 0.0000e+00
Epoch 25/30
21/21 [==============================] - 0s 1ms/step - loss: 4239.9683 - accuracy: 0.0000e+00
Epoch 26/30
21/21 [==============================] - 0s 1ms/step - loss: 4187.6821 - accuracy: 0.0000e+00
Epoch 27/30
21/21 [==============================] - 0s 1ms/step - loss: 4132.7056 - accuracy: 0.0000e+00
Epoch 28/30
21/21 [==============================] - 0s 1ms/step - loss: 4086.0457 - accuracy: 0.0000e+00
Epoch 29/30
21/21 [==============================] - 0s 1ms/step - loss: 4036.3979 - accuracy: 0.0000e+00
Epoch 30/30
21/21 [==============================] - 0s 1ms/step - loss: 3989.6101 - accuracy: 0.0000e+00
<keras.callbacks.History at 0x7f773cc03af0>
```

6 딥러닝을 이용하여 예측(교차검증X) + 그래프

그래프 코드

```
1. #예측값
2. print('딥러닝 모델, 교차검증 X, 학습셋 테스트셋 나눔놈: ',Text2, '℃ 일때
    밀집도 예측: ', model1.predict([[Text2]]))
3.
4. y_pred = model1.predict(x)
5. plt.scatter(x, y, color='black')
6. # 학습 데이터와 예측값으로 선그래프로 그린다.
7. # 계산된 기울기와 y 절편을 가지는 직선이 그려진다.
8. plt.plot(x, y_pred, color='blue', linewidth=3)
9. plt.xlabel('온도(℃)')
10.plt.ylabel('밀집도')
11.plt.show()
```

코드 설명

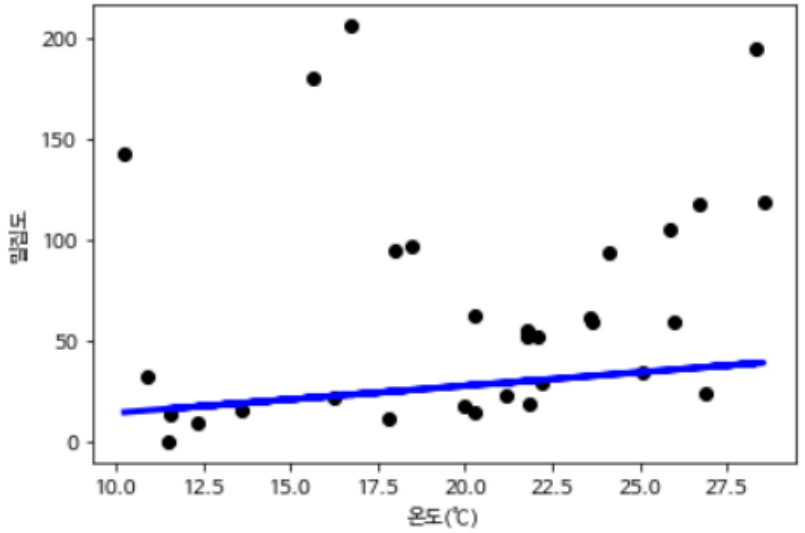
2 예측값 출력

4~11 예측값(선) 실제값(점) 그래프로 표시

6 딥러닝을 이용하여 예측(교차검증X) + 그래프

실행결과

```
1/1 [==============================] - 0s 36ms/step
딥러닝 모델, 교차검증X, 학습셋 테스트셋 나눔눔:  26 ℃ 일때 밀집도 예측:  [[35.623276]]
1/1 [==============================] - 0s 22ms/step
```



7 딥러닝을 사용한 예측(교차검증 O) + 그래프

코드

```
1. from sklearn.model_selection import StratifiedKFold
2.
3. print('교차검증 O')
4. skf = StratifiedKFold(n_splits=10, shuffle=True)
5. accuracy=[]
6. for train, test in skf.split(x,y):
7.   model2 = Sequential()
8.   model2.add(Dense(2,input_dim=1,activation='relu'))
9.   model2.add(Dense(1,activation='relu'))
10.  model2.compile(loss='mean_squared_error',
11.                 optimizer='adam',
12.                 metrics=['accuracy'])
13.
14.  model2.fit(x[train], y[train],  epochs=30, batch_size=1)
15.  k_accuracy = "%.4f"%(model2.evaluate(x[test],y[test])[1])
16.  accuracy.append(k_accuracy)
17.

18.print('정확도: ',accuracy)
```

코드 설명

1 필요한 라이브러리 추가

4 교차검증

5 정확도 배열 생성

6~12 학습셋 테스트셋 구분 후 교차검증 모델 생성

14 모델 학습

15~16 정확도 배열에 추가

18 정확도 출력

7 딥러닝을 사용한 예측(교차검증 O) + 그래프

실행결과

```
16/16 [==============================] - 0s 2ms/step - loss: 6973.8750 - accuracy: 0.0625
Epoch 25/30
16/16 [==============================] - 0s 1ms/step - loss: 6973.8750 - accuracy: 0.0625
Epoch 26/30
16/16 [==============================] - 0s 2ms/step - loss: 6973.8750 - accuracy: 0.0625
Epoch 27/30
16/16 [==============================] - 0s 2ms/step - loss: 6973.8750 - accuracy: 0.0625
Epoch 28/30
16/16 [==============================] - 0s 2ms/step - loss: 6973.8750 - accuracy: 0.0625
Epoch 29/30
16/16 [==============================] - 0s 2ms/step - loss: 6973.8750 - accuracy: 0.0625
Epoch 30/30
16/16 [==============================] - 0s 2ms/step - loss: 6973.8750 - accuracy: 0.0625
1/1 [==============================] - 0s 106ms/step - loss: 7814.3335 - accuracy: 0.0000e+00
정확도:  ['0.0625', '0.0000']
```

7 딥러닝을 사용한 예측(교차검증 O) + 그래프

그래프 코드

```
1. #예측값
2. print('딥러닝모델, 교차검증 o, 테스트셋 학습셋 나눔: ',Text2, '℃ 일때 밀
   집도 예측: ',model2.predict([[Text2]]))
3. y_pred = model2.predict(x)
4. plt.scatter(x, y, color='black')
5. # 학습 데이터와 예측값으로 선그래프로 그린다.
6. # 계산된 기울기와 y 절편을 가지는 직선이 그려진다.
7. plt.plot(x, y_pred, color='blue', linewidth=3)
8. plt.xlabel('온도(℃)')
9. plt.ylabel('밀집도')
10.plt.show()
```

코드 설명

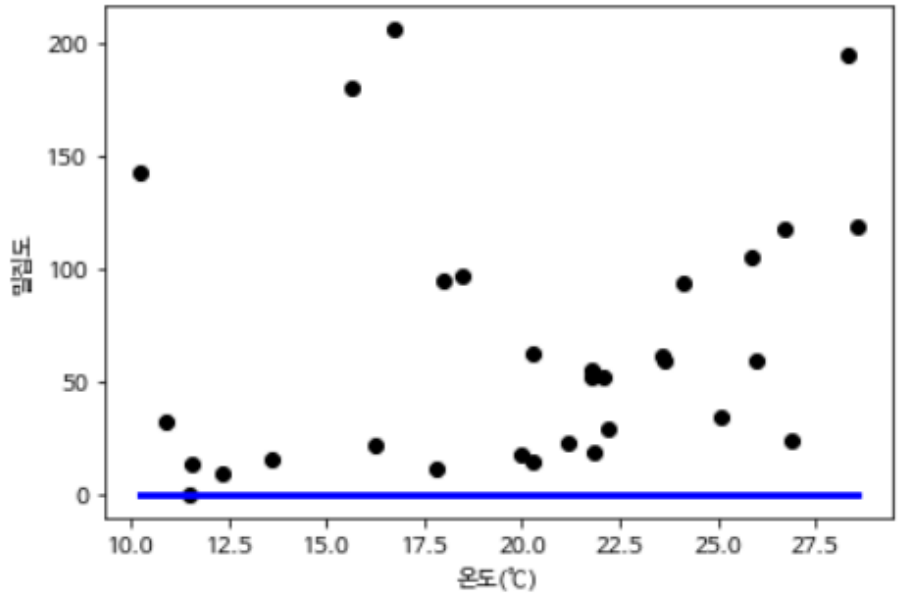
2 예측값 출력

3~10 예측값(선) 실제값(점) 그래프로 표시

7 딥러닝을 사용한 예측(교차검증 O) + 그래프

실행결과

```
1/1 [==============================] - 0s 65ms/step
딥러닝모델, 교차검증 O, 테스트셋 학습셋 나눔:  26 ℃ 일때 밀집도 예측:  [[0.]]
1/1 [==============================] - 0s 48ms/step
```



8 부족한점

실제 값들이 너무 튀어서 딥러닝 학습이 잘 안되어서 예측에 실패하였다