기온에 따른 지면 온도 예측하기

한현민 장범수 김선준

팀명: lad

역할

김선준 - ppt 제작, 모델 설계

한현민 - 데이터 학습

장범수 - 파라미터 변경

목차

1. 데이터 수집 및 확인
2. 결측치 제거, data 나누기
3. 정규화
4. 선형회귀 모델 설계
5. 학습
6. 예측값 도출
7. 파라미터 변경
8. 실제값과 예측값을 시각화

데이터 수집 및 확인

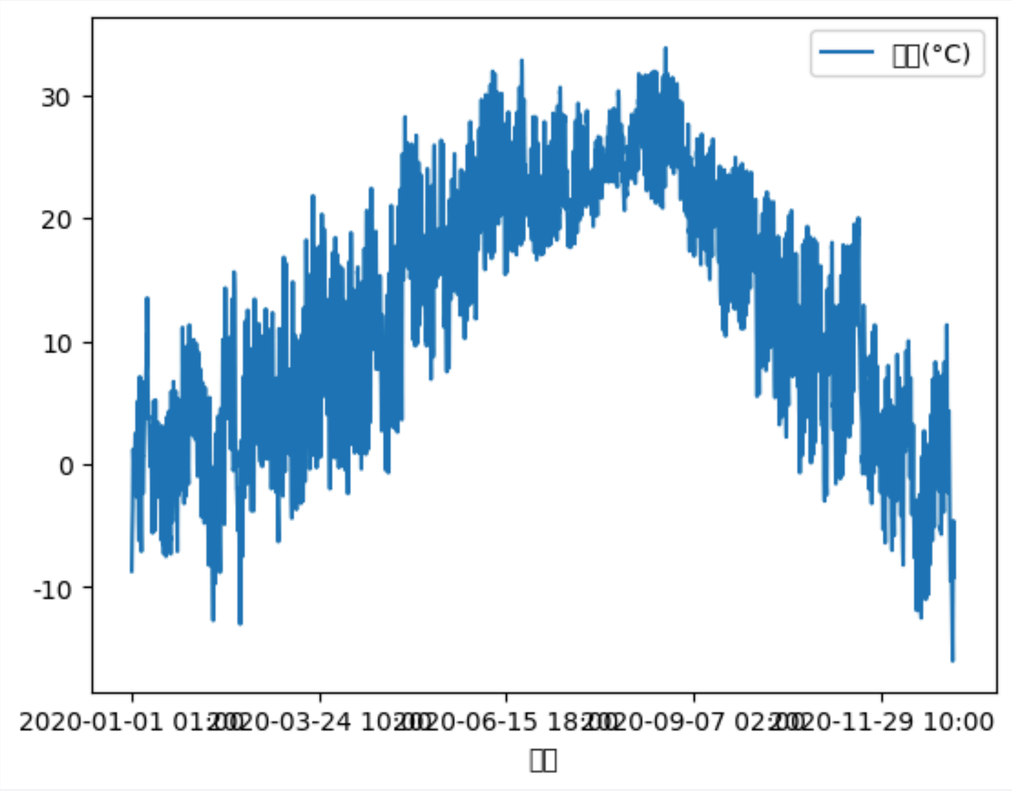
천안 지역의
2020.01.01~2020.12.31
01:00~24:00
공공데이터 수집

	지점	지점명	일시	기온(°C)	지면온도(°C)
0	232	천안	2020-01-01 01:00	-8.7	-2.9
1	232	천안	2020-01-01 02:00	-7.3	-2.4
2	232	천안	2020-01-01 03:00	-6.7	-2.2
3	232	천안	2020-01-01 04:00	-6.2	-2.0
4	232	천안	2020-01-01 05:00	-5.9	-1.9

dtype: float64
totla data: 8782개

```
 3   기온(°C)     8779 non-null   float64
 4   지면온도(°C)  8782 non-null   float64
dtypes: float64(2), int64(1), object(2)
memory usage: 343.2+ KB
```

기초 데이터 시각화



▲날짜에 따른 온도

데이터 변환 및 분할

```
1 data = df[[ '기온(° C)']].to_numpy()
2 target = df['지면온도(° C)'].to_numpy()
```

csv 데이터 파일 --> numpy

데이터셋 분할

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
1 print(X_train.shape)
2 print(y_train.shape)
3
4 print(X_test.shape)
5 print(y_test.shape)
```

```
(7025, 1)
(7025,)
(1757, 1)
(1757,)
```

train data 7025개

test data 1757개

totla data 8772개

결측치 확인 및 제거

결측치 확인

```
1 print(df.isnull().sum())
```

```
지점           0
지점명          0
일시           0
기온(°C)       3
지면온도(°C)     0
dtype: int64
```

결측치 제거

```
[ ]  1 df = df.dropna()
     2
     3 print(df.isnull().sum())
```

```
지점           0
지점명          0
일시           0
기온(°C)       0
지면온도(°C)     0
dtype: int64
```

정규화

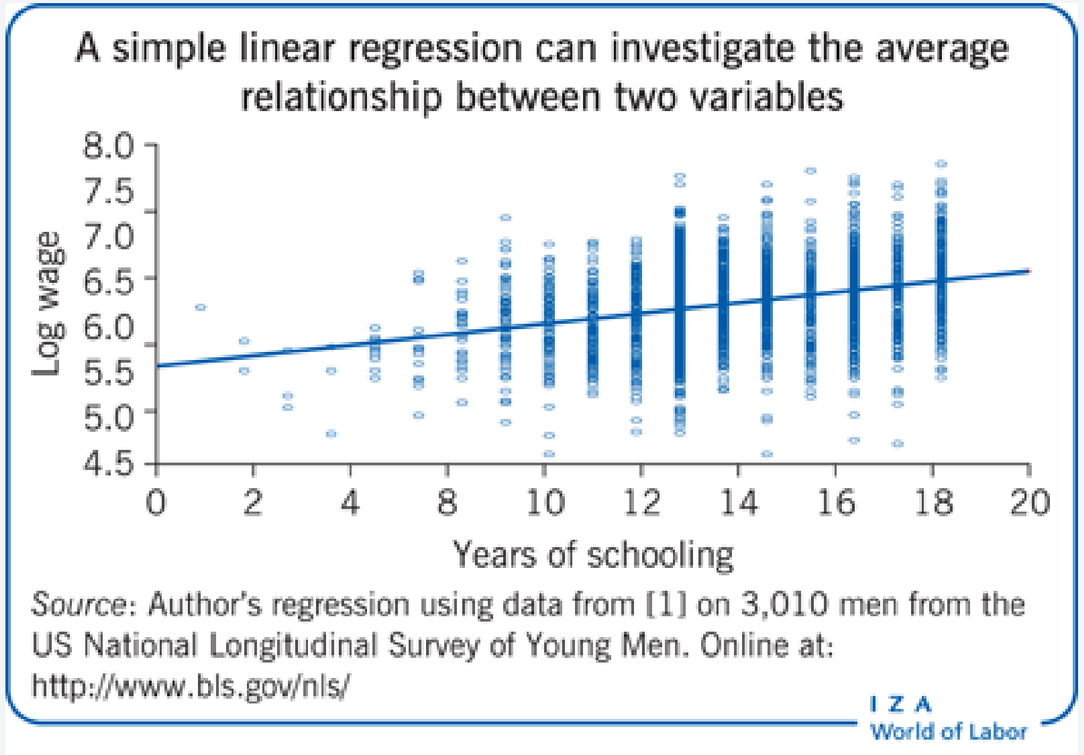
```
from sklearn.preprocessing import StandardScaler # 정규화 하는 거
scaler = StandardScaler()
scaler.fit(X_train)

X_train_scaled = scaler.transform(X_train)
# y_train_scaled = scaler.transform(y_train)  # Remove this line - don't scale the target variable

X_test_scaled = scaler.transform(X_test)
# y_test_scaled = scaler.transform(y_test)  # Remove this line - don't scale the target variable
```

```
print(model.score(X_train_scaled_imputed, y_train))
print(model.score(X_test_scaled_imputed, y_test))
```

```
0.9071990834932966
0.9013958529500761
```

통계적 기법 선정

지면 온도와 기온연관성을 분석하기 위해 ‘선형회귀 ‘ 사용



특성(X)와 목표 변수(y) 분리

```
df = pd.read_csv('https://github.com/kairess/toy-datasets/raw/master/temps.csv', encoding='euc-kr')

X = df.drop(columns=['기온(°C)', '지점', '지점명', '일시'])
y = df['기온(°C)']
```

선형 회귀 모델 초기화 및 학습

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
```

학습전 score

```
[ ]  1 from sklearn.model_selection import train_test_split
     2 X_train, X_test, y_train, y_test = train_test_split(data, target, test_size=0.2, random_state=42)
```

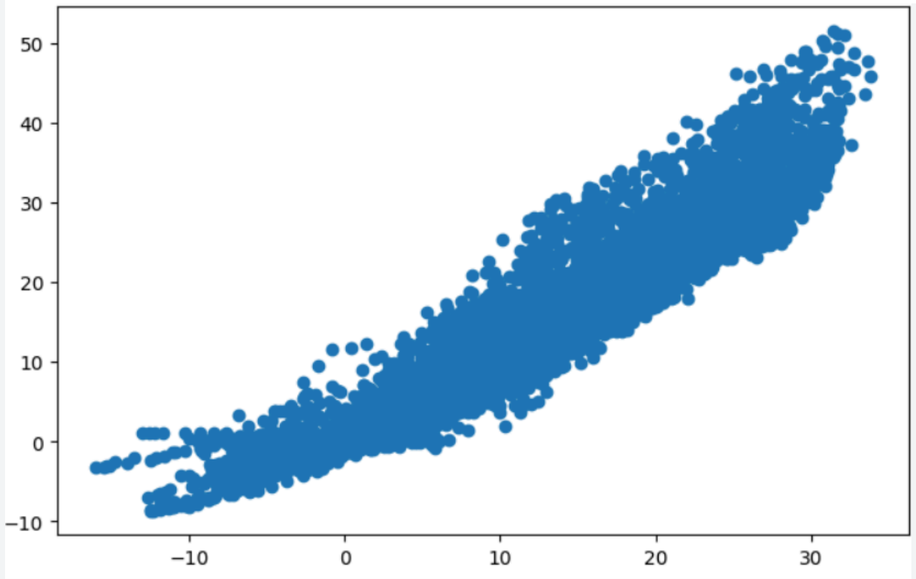
```
11 # Now create and fit the model using the imputed data
12 model = LinearRegression()
13 model.fit(X_train_scaled_imputed, y_train)  # Use y_train, not X
14
15 print(model.score(X_train_scaled_imputed, y_train))
16 print(model.score(X_test_scaled_imputed, y_test))
```

```
0.9019612718885828
0.905363712525228
```

optimizer 생성

```
model = nn.Linear(1, 1)

optimizer = torch.optim.SGD(model.parameters(), lr=0.001)

print(list(model.parameters()))
```

```
[Parameter containing:
tensor([[-0.2057]], requires_grad=True), Parameter containing:
tensor([0.5087], requires_grad=True)]
```



-> 입력 특징이 1개, 출력 특징이 1개인 선형 계층 생성

-> SGD 이용

-> 옵티마이저의 학습률 0.001로 설정

모델 데이터 학습

1.입력 데이터 x_data를 모델에 통과시켜 예측값 y_pred를 계산 (x-기온, y-지면온도)

2. loss 함수 MSE 사용하여 기울기를 계산

3. 계산된 기울기를 사용하여 모델의 가중치를 업데이트

4. 기계 학습 모델을 반복하여 학습시키고, 손실 값을 출력하여 학습 과정을 확인 (반복과정)

```
epochs = 1000

for epoch in range(epochs + 1):
    y_pred = model(x_data)
    loss = nn.MSELoss()(y_pred, y_data)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

예측값 도출

1.입력 데이터 x_data에 대한 예측값을 계산

2.계산 결과를 NumPy 배열로 변환 후 예측값 출력

```
array([[ -9.547958],
       [ -7.984181],
       [ -7.31399 ],
       ...,
       [ -7.984181],
       [ -9.883054],
       [-10.10645 ]], dtype=float32)
```

학습률, epoch에 따른 loss 차이

```
lr=0.002)
```

```
1000/1000 Loss: 12.9576
```

```
100/100 Loss: 12.9573
```

```
lr=0.001)
```

```
1000/1000 Loss: 12.9613
```

```
100/100 Loss: 13.2327
```

test size에 따른 정확도 차이

test size 0.8

```
1 print(model.score(X_train_scaled_imputed, y_train))
2 print(model.score(X_test_scaled_imputed, y_test))
```

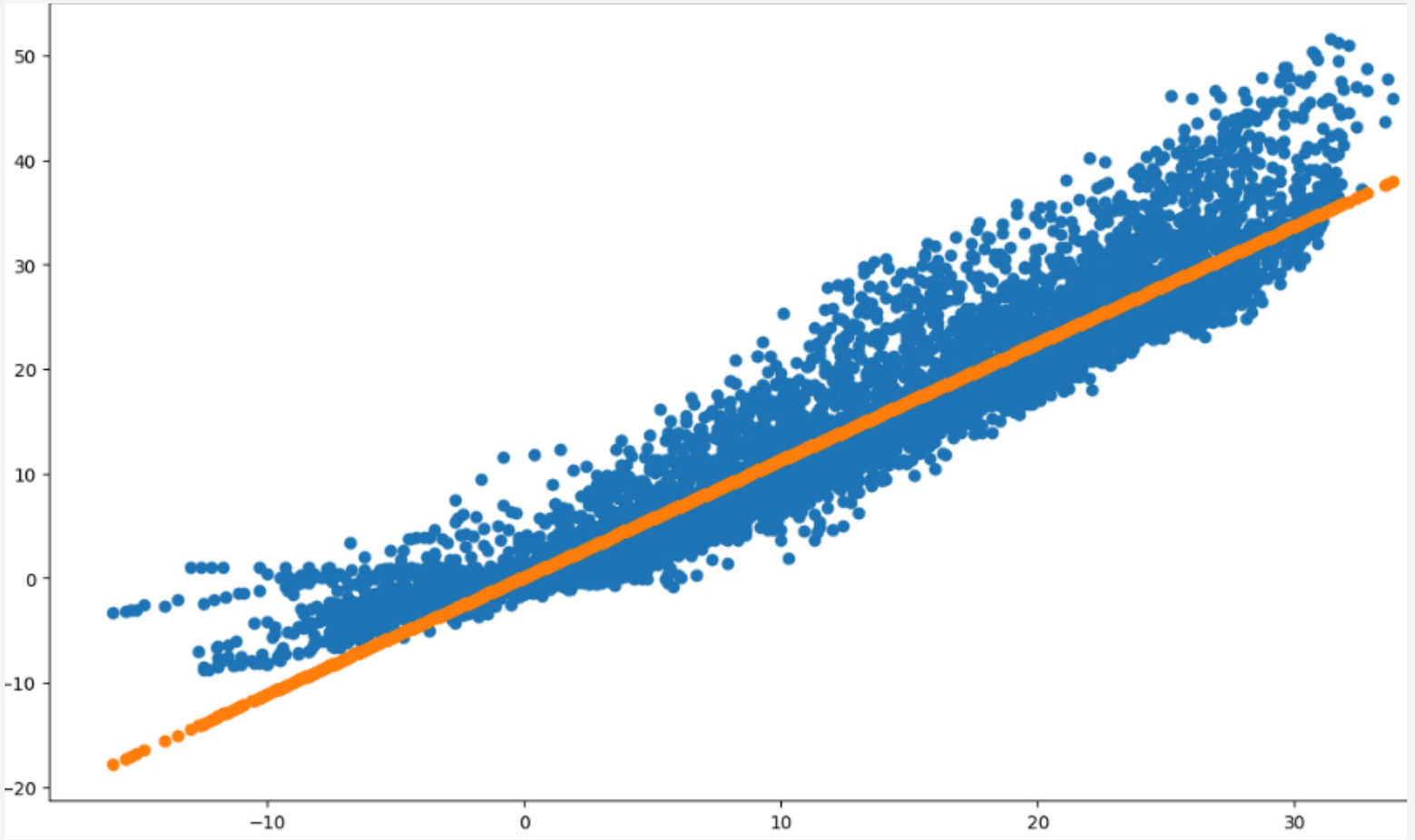
```
0.90719908
0.90139585
```

```
0.901961
0.905363
```

test size 0.2

```
15 print(model.score(X_train_scaled_imputed, y_train))
16 print(model.score(X_test_scaled_imputed, y_test))
```

실제값과 예측값을 시각화



기온 상승 → 지면온도 상승

개선할 점

시간별 바람 데이터를 통한 예측
최저기온, 최고기온 시간 예측