머신 러닝을 통한 평균 기온 예측하기

시스템제어공학과
20210710 맹지우

Content

Introduction

서론_배경 및 목적

Main Body

본론_코드 구현

Conclusion

결론_결과 분석

01

서론

Introduction

배경 및 목적

기상 데이터는 일상 생활에서 많은 영향을 미치는 중요한 정보입니다. 평균기온은 농업, 에너지 생산, 건축 등 다양한 분야에서 활용되며, 이를 정확하게 예측하는 것은 매우 중요합니다. 정확한 평균기온 예측은 농작물의 수확 시기, 에너지 수급 계획, 건물 내부 온도 관리 등 다양한 결정에 영향을 미치는 요소입니다.

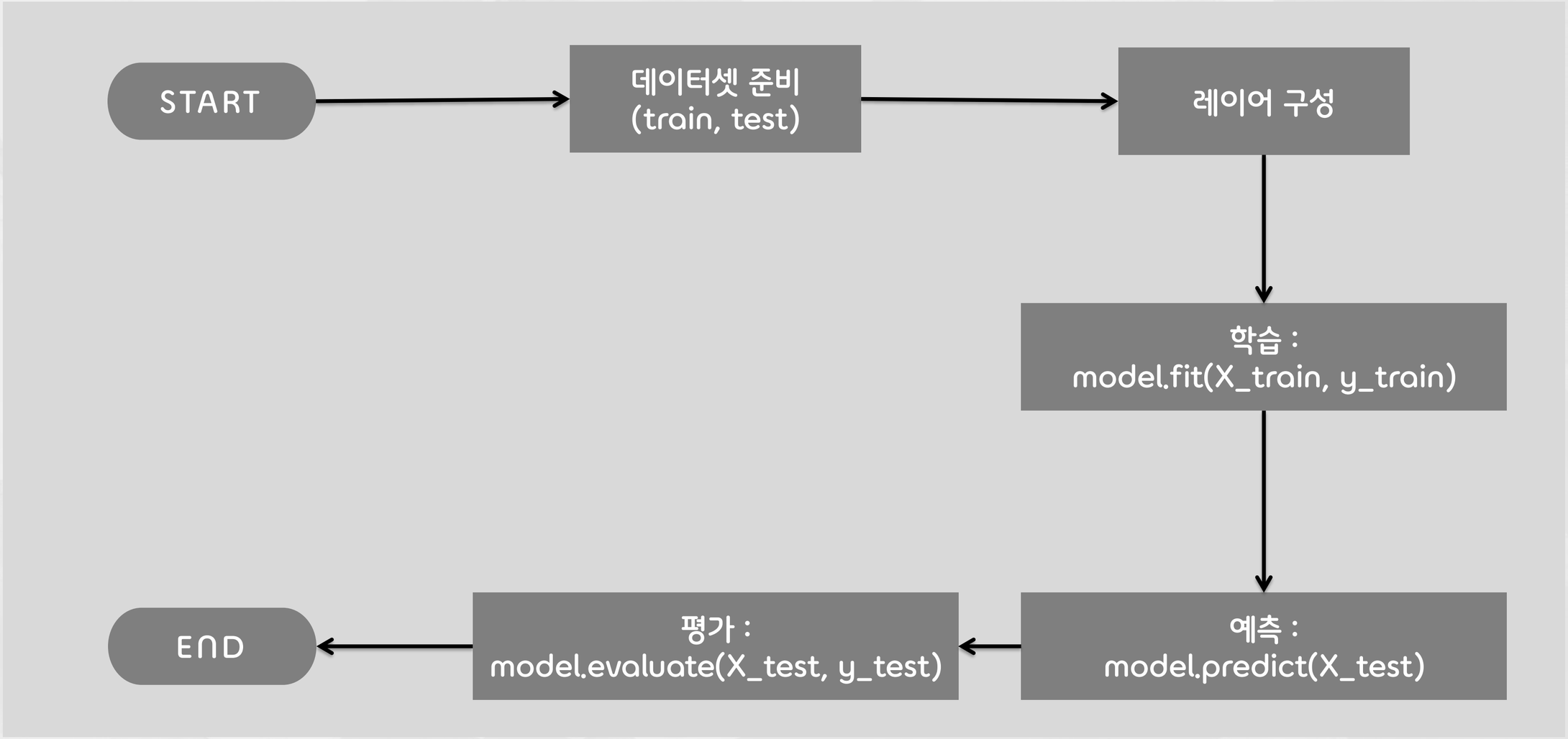
머신러닝은 기상 데이터 분석과 예측에 널리 활용되는 방법 중 하나입니다. 머신러닝 알고리즘을 활용하여 기상 데이터의 패턴과 관계를 학습하고, 이를 기반으로 평균기온을 예측하는 모델을 구축할 수 있습니다. 머신러닝은 데이터의 복잡한 패턴을 탐지하고 예측할 수 있는 강력한 도구로서, 정확한 평균기온 예측을 위해 적절히 활용될 수 있습니다. 따라서, 머신러닝을 활용하여 평균기온을 예측하는 프로그램을 작성해보려고 합니다.

Main Body

02

본론

실행 과정


START
데이터셋 준비
(train, test)
레이어 구성
학습 :
model.fit(X_train, y_train)
예측 :
model.predict(X_test)
평가 :
model.evaluate(X_test, y_test)
END

라이브러리 불러오기 & 폰트 설정

```python
import tensorflow as tf
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

TensorFlow, Pandas, NumPy, Seaborn, Matplotlib 라이브러리를 import합니다.

```python
# os에 따라 폰트 설정하기
import os

if os.name == 'nt':
    font_family = "Malgun Gothic"
else:
    font_family = "AppleGothic"

sns.set(font=font_family, rc={"axes.unicode_minus" : False })
```

Windows 운영체제인 경우 "Malgun Gothic" 폰트를 사용하고, 그 외의 경우에는 "AppleGothic" 폰트를 사용합니다.
Seaborn의 폰트 설정과 Matplotlib의 유니코드 마이너스 기호 설정을 변경합니다.
위의 코드를 통해 시각화 결과물의 한글 폰트가 정상적으로 표시되고, 마이너스 기호가 유니코드로 표시되도록 설정됩니다.

데이터프레임 읽어오기

```
df = pd.read_csv("weather.csv")
df.shape
```

"weather.csv" 파일을 DataFrame으로 읽어옵니다.

DataFrame의 크기를 확인하는 코드로, 출력 결과는 (행의 수, 열의 수) 형태로 나타납니다.

```
(7665, 62)
```

```
df.head()
```

DataFrame의 처음 5개의 행을 출력합니다.

	지점	지점명	일시	평균기온(°C)	최저기온(°C)	최저기온 시각(hhmi)	최고기온(°C)	최고기온 시각(hhmi)	강수계속시간(hr)	10분최다강수량(mm)	...	1.0m지중온도(°C)	1.5m지중온도(°C)	3.0m지중온도(°C)	5.0m지중온도(°C)	합계 대형증발량(mm)	합계 소형증발량(mm)	9-9강수(mm)	안개계속시간(hr)	합계 일조시간(hr)	합계 일사량(MJ/m2)
0	108	서울	2017-01-01	2.7	-1.6	540.0	6.9	1419.0	NaN	NaN	...	7.1	10.2	15.7	17.5	0.7	1.0	0.3	NaN	NaN	NaN
1	108	서울	2017-01-02	5.0	1.8	2355.0	9.2	1355.0	2.08	NaN	...	7.0	10.1	15.6	17.5	0.9	1.2	NaN	NaN	NaN	NaN
2	108	서울	2017-01-03	2.0	-2.3	731.0	7.7	1444.0	NaN	NaN	...	6.9	9.9	15.5	17.4	1.3	1.8	NaN	NaN	NaN	NaN
3	108	서울	2017-01-04	3.9	1.0	1.0	8.9	1519.0	NaN	NaN	...	6.9	9.8	15.4	17.4	1.5	2.2	NaN	NaN	NaN	NaN
4	108	서울	2017-01-05	3.8	-0.1	316.0	7.3	1444.0	NaN	NaN	...	6.9	9.7	15.4	17.4	1.6	2.3	NaN	NaN	NaN	NaN

5 rows × 62 columns

데이터 전처리

```
# 예측할 정답값
label_name = '평균기온(° C)'
```

```
# 결측치를 0값으로 임의로 채움
df = df.fillna(0)
```

label_name = '평균기온(°C)'

예측할 정답 값(평균기온)의 열 이름을 '평균기온(°C)'로 지정합니다.

df = df.fillna(0)

데이터 프레임(df)의 결측치를 0으로 채웁니다. fillna 함수를 사용하여 결측치를 지정한 값(여기서는 0)으로 대체합니다.

데이터 전처리

```
# 수치데이터만 feature 로 사용
num_desc = df.describe()
feature_name = num_desc.columns.tolist()
feature_name
```

```
['지점',
 '평균기온(°C)',
 '최저기온(°C)',
 '최저기온 시각(hhmi)',
 '최고기온(°C)',
 '최고기온 시각(hhmi)',
 '강수 계속시간(hr)'
```

데이터프레임(df)의 수치 데이터에 대한 요약 통계를 계산하여 num_desc에 저장합니다.

num_desc의 열 이름(수치 데이터의 feature)을 리스트로 추출하여 feature_name에 저장합니다.

```
# feature와 label 값으로 나눔
X = df[feature_name].drop([label_name], axis=1)
y = df[label_name]

X.shape, y.shape
```

```
((7665, 59), (7665,))
```

feature_name에 해당하는 열을 선택한 후, label_name 열을 제외하고 X 변수에 저장합니다

label_name 열을 y 변수에 저장합니다.

X와 y의 형태(크기)를 확인하여 출력합니다. 결과는 (X의 행 수, X의 열 수)와 (y의 행 수,) 형태로 나타납니다.

데이터 전처리

데이터를 학습(train) 세트와 테스트(test) 세트로 나누는 작업을 수행하는 코드입니다.

```
# train, test 세트로 나눔
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.33, random_state=42)
```

scikit-learn의 train_test_split 함수를 import합니다.

X와 Y 데이터를 주어진 비율로(train 데이터: 67%, test 데이터: 33%) 나누어 X_train, X_test, y_train, y_test 변수에 저장합니다.

random_state는 난수 생성 시드(seed)로, 같은 값을 사용하면 매번 같은 학습 및 테스트 세트가 생성됩니다.

```
# 텐서플로우 로드
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

print(tf.__version__)
```

```
2.12.0
```

TensorFlow를 import합니다.

TensorFlow에서 keras 모듈을 import합니다.

TensorFlow.keras에서 layers 모듈을 import합니다.

TensorFlow의 버전을 출력합니다.(2.12.0 verison)

결과 예측하기

```
y_pred = model.predict(X_test)
y_pred
```

```
80/80 [==============================] - 0s 2ms/step

array([[ 9.685562 ],
       [13.677121 ],
       [12.624794 ],
       ...,
       [23.833534 ],
       [ 7.4977975],
       [22.634932 ]], dtype=float32)
```

```
score = model.evaluate(X_test, y_test, verbose=2)
score
```

```
80/80 - 0s - loss: 25.3752 - mae: 4.0465 - mse: 25.3752 - 452ms/epoch - 6ms/step

[25.375177383422285, 4.046520233154297, 25.375177383422285]
```

요약

X_test 데이터에 대한 평균 기온 값들을 출력하고 예측을 수행한 결과와 해당 예측을 평가한 결과를 나타내고 있습니다. 이를 평가한 결과 손실 값은 약 25.38, 평균 절대 오차는 약 4.05, 평균 제곱 오차는 약 25.38입니다.

예측 결과 (y_pred)

X_test 데이터에 대한 예측값을 나타내는 배열로, 각 행은 X_test의 각 샘플에 대한 예측된 평균 기온을 나타냅니다. 주어진 결과에서는 X_test의 각 샘플에 대한 예측된 평균 기온이 출력되어 있습니다.

평가 결과 (score)

score는 모델의 성능 평가를 나타내는 배열로, 첫 번째 요소는 손실(loss) 값입니다. 이 값은 모델이 예측한 결과와 실제값 간의 차이를 나타내는 오차를 평균한 값입니다. 작을수록 모델의 예측이 정확하다는 것을 의미합니다. 이 모델의 손실 값은 약 25.38입니다.
두 번째 요소는 평균 절대 오차(MAE: Mean Absolute Error)입니다. MAE는 예측값과 실제값 간의 차이를 평균한 값입니다. 작을수록 모델의 예측이 정확하다는 것을 의미합니다. 이 모델의 MAE는 약 4.05입니다. 세 번째 요소는 평균 제곱 오차(MSE: Mean Squared Error)입니다. MSE는 예측값과 실제값 간의 차이를 제곱한 후 평균한 값입니다. 작을수록 모델의 예측이 정확하다는 것을 의미합니다. 이 모델의 MSE는 약 25.38입니다.

결과 예측하기

```
# MAE(Mean Absolute Error) : 평균 절대 오차
MAE = score[1]
MAE
```

```
4.046520233154297
```

```
# MSE(Mean Square Error) : 평균 제곱 오차
MSE = score[2]
MSE
```

```
25.37517738342285
```

```
# RMSE(Root Mean Square Error) : 평균 제곱근 오차
RMSE = np.sqrt(MSE)
RMSE
```

```
5.037378026654625
```

평균 절대 오차(MAE: Mean Absolute Error)

예측값과 실제값 간의 차이를 평균한 값으로, MAE 값이 작을수록 모델의 예측이 실제값과 가깝다는 것을 의미합니다.

평균적으로 약 4.05 정도의 절대적인 오차를 가지고 있습니다.

평균 제곱 오차(MSE: Mean Squared Error)

예측값과 실제값 간의 차이를 제곱한 후 평균한 값으로, MSE 값이 작을수록 모델의 예측이 실제값과 가깝다는 것을 의미합니다.

평균적으로 약 25.38 정도의 제곱 오차를 가지고 있습니다.

평균 제곱근 오차(RMSE: Root Mean Squared Error)

예측값과 실제값 간의 차이를 평균적으로 얼마나 크게 예측했는지 나타내는 지표로, RMSE 값이 작을수록 모델의 예측이 실제값과 가깝다는 것을 의미합니다.

평균적으로 약 5.04 정도의 제곱근 오차를 가지고 있습니다.

03

결론

Conclusion

결론

```
# MAE(Mean Absolute Error) : 평균 절대 오차
MAE = score[1]
MAE
```

```
4.046520233154297
```

```
# MSE(Mean Square Error) : 평균 제곱 오차
MSE = score[2]
MSE
```

```
25.37517738342285
```

```
# RMSE(Root Mean Square Error) : 평균 제곱근 오차
RMSE = np.sqrt(MSE)
RMSE
```

```
5.0373780266546255
```

주어진 모델은 평균 절대 오차(MAE) 약 4.05, 평균 제곱 오차(MSE) 약 25.38, 평균 제곱근 오차(RMSE) 약 5.04를 가지고 있습니다. 이 결과만 보면 모델의 예측은 상당히 정확하다고 볼 수 있습니다.

하지만 예측 모델의 평가는 단일한 지표만으로 결정되지 않으며, 다양한 요소를 고려해야 하기 때문에 추가적인 평가 지표와 데이터에 대한 분석을 진행하여야 합니다.

THANK YOU