AI기반 태양광 발전량 예측

AI 기반 태양광 발전량 예측 **:** 선형회귀 및 비선형회귀 모델 탐구

저자 : 강승연 고다은 나예서 임승미 임정현

전라남도 광양시 마로니에길 61 광양제철고등학교

요약 **(ABSTRACT)**

재생에너지원은 출력의 변동성이 거의 없는 전통전원과 달리 간헐성과 확률성 문제로 인해 전력 계통을 운영하는 데 많은 어려움이 발생한다. 정확한 발전량의 예측은 전력 가격을 결정하는 중요한 요인이 되며, 이를 바탕으로 태양광 발전을 담당하는 사업자의 수익과 직결되는 태양광 발전 시스템의 가동 정도를 정할 수 있다. 이번 탐구에서는 태양광 발전량 예측 모델을 제작하기 위한 핵심적인 선형회귀와 비선형회귀를 이해하고, 다양한 머신러닝 모델의 탐구를 통해 국내 신인천 태양광 발전소의 발전량을 예측해보았으며, 속도와 정확성을 높이기 위한 방법을 제시해보았다.

서론 **(INTRODUCTION)**

기후위기로 인해 세계적으로 전통전원의 축소와 신재생에너지의 확대가 진행 중이다. 수요의 증가로 인한 기술 발전과 가격 하락, 인센티브 등으로 풍력 및 태양광 점유율이 급격하게 증가하고 있다. 한국도 세계적인 움직임에 동참해 재생에너지의 보급을 확대하고 있다. 정부는 '제 10차 전력수급기본계획'을 발표하면서 2030년까지 신재생에너지 발전량 비중을 21.6%까지 확대하는 것으로' 제 9차 전력수급기본계획'의 20.8%보다 상향된 목표를 내세웠다.

		원자력	석탄	LNG	신재생	수소 암모니아	기타	계
18년	발전량 (TWh)	133.5	239.0	152.9	35.6	-	9.7	570.7
	비중 (%)	23.4	41.9	26.8	6.2	-	1.7	100
30년	발전량	201.7	122.5	142.4	134.1	13.0	8.1	621.8
	비중	32.4	19.7	22.9	21.6	2.1	1.3	100
36년	발전량	230.7	95.9	62.3	204.4	47.4	26.6	667.3
	비중	34.6	14.4	9.3	30.6	7.1	4.0	100

<표**1>** 전원별 발전량 비중 전망

하지만, 재생에너지원은 출력의 변동성이 거의 없는 전통전원과 달리 간헐성과 확률성 문제로 인해 전력 계통을 운영하는 데 많은 어려움이 발생한다. 정확한 발전량의 예측은 전력 가격을

결정하는 중요한 요인이 되며, 이를 바탕으로 태양광 발전을 담당하는 사업자의 수익과 직결되는 태양광 발전 시스템의 가동 정도를 정할 수 있다. 정확한 발전량의 예측을 통해 재생에너지원의 공급량 조절이 가능하다면 재생에너지의 활용도가 개선될 수 있다. 정확한 발전량의 예측을 통해 안정적인 그리드 운영이 가능하며, 에너지 출력 장치의 활용을 최적화하여 에너지의 낭비를 최소화할 수 있다. 또한, 예측 발전량과 실제 발전량의 비교를 통한 발전소의 성능 모니터링도 가능하다.

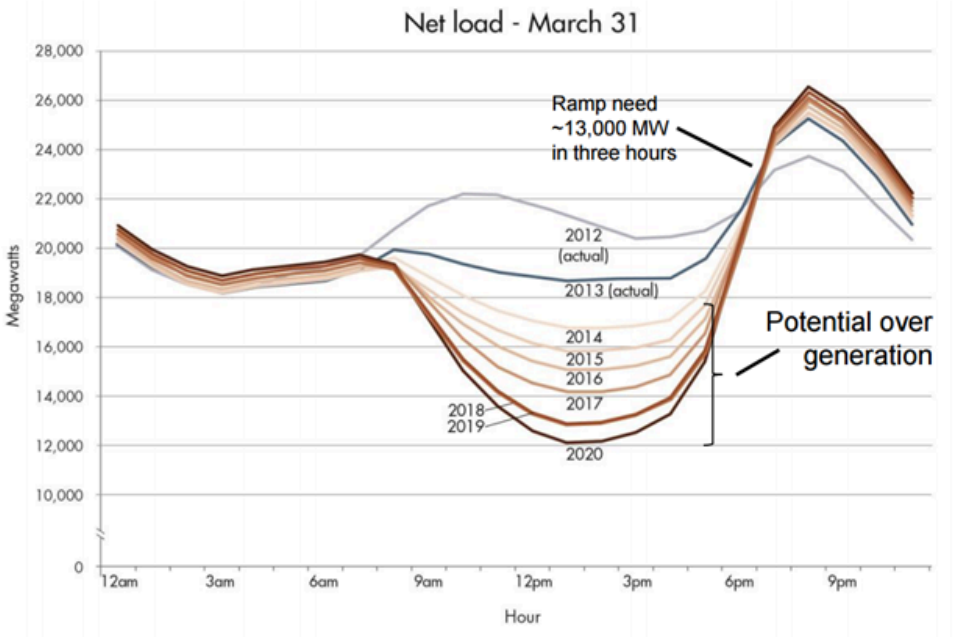
이번 탐구 활동을 진행하는 것은 선형 회귀와 비선형 회귀 모델 및 다양한 머신러닝 모델의 탐구를 바탕으로 국내 태양광 발전소의 발전량을 예측하기 위해서이다. 따라서, 주 탐구 목적은 다음과 같다.

- 선형 회귀와 비선형 회귀 모델 알아보기
- 다양한 머신러닝 모델 알아보기
- 태양광 발전량에 영향을 미치는 요소들을 알아보고 국내 태양광 발전소 (인천)의 발전량 예측해보기

재료 및 방법 **(MATERIALS AND METHODS)**

(1) 재생에너지의 간헐성과 확률성 문제

태양광 발전은 기상 및 기후의 변화에 큰 영향을 받는다. 이에 따라 발전의 출력이 간헐적이다. 주요한 재생에너지 자원을 활용한 전기 생산이 날씨 등 외부요인에 따라 좌우되는 특성을 말하는 간헐성의 영향에 따라 출력이 시시각각 변하기 때문에 전통전원(원자력, 석탄화력, 천연가스)에 비해 변동성이 비교적 높다. 그렇기 때문에 태양광 발전은 정확한 예측 또한 어려우며, 발전의 출력은 불확실성을 띈다.



<그림**1>** 덕커브 현상

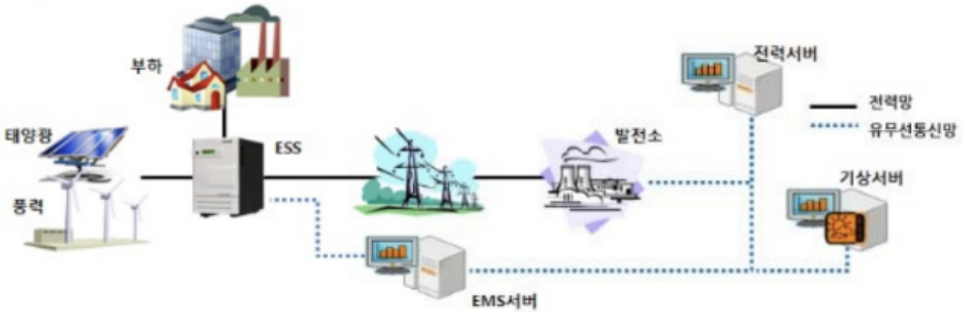
덕커브(Duck Curve)는 2013년도 CAISO(미국의 캘리포니아 주 전력계통운영회사)에서 발견한 현상으로, 신재생, 특히 태양광 발전량의 급증으로 일출 후 전력수요의 대부분을 태양광이 담당하여 일출에서 일몰 사이에 순부하 (전체부하 - 태양광, 풍력 발전량)가 급격히 떨어지는 현상이다. 덕커브 현상은 태양광 보급 확대에 따라 심화될 것으로 예상된다.

출력 변동성이 심한 신재생 발전의 증가로 순부하량 예측의 정확성이 떨어짐에 따라, 전력망의 안정적 운영이 곤란해진다. 태양광 발전량이 늘어나 석탄, 원자력 등 기저전원 담당 부하까지 점유하게 될 경우 하루 단위로 기저전원을 가동 정지 및 재가동해야 한다. 석탄, 원자력 대비 단시간에 기동, 정지가 가능한 가스발전으로 예비력을 충당할 수 있지만, 일몰 후 저녁 피크 부하만을 위해 가스발전을 운전하기에는 운영유지비가 부담된다.

정확한 발전량을 예측하는 것 뿐만 아니라 간헐성과 확률성을 대비하기 위해 발전량이 안정적일 때 전력을 충전했다가 나머지 시간대에 방전해 에너지의 출력을 안정화하는, 즉 효율적인 에너지 사용을 위해서는 ESS의 역할 역시 중요하다.

(2) ESS (에너지저장장치**)**

- 전력 수요가 적을 때 잉여에너지를 그리드에 저장해 과부하 혹은 비상시에 전력을 공급하는 장치를 말한다.



<그림 **2> ESS**의 구성요소

- PCS(전력변환시스템), BMS(배터리관리시스템), EMS(에너지관리시스템)으로 구성된다.

(3) 신인천 태양광 발전량을 선택한 이유

인천은 에너지 자급률이 300%에 육박할 만큼 수도권 시민들을 위한 보급도시로서의 역할을 한다. 지리적 위치 때문에 인천 관내에 소재하는 석탄, LNG 발전소에서는 24시간 미세먼지와 온실가스가 배출되고 있고, 이로 인해 미세먼지의 고통과 더불어 온실가스 다배출도시이다. 이러한 추세는 감소하기는 커녕 노후 석탄발전소의 일부 가동 중지에 따른 인천의 LNG 화력

발전소의 가동량 증가로 이어진다. 이에 과도한 화석연료를 통한 발전을 대체할 신재생에너지의 확대가 불가피하다. 신재생에너지의 보급이 확대됨에 따라 인천의 태양광 발전량을 예측하는 것은 몹시 중요할 것으로 예상된다.

그렇기 때문에 우리는 이번 탐구에서 신인천의 기상정보 데이터셋을 쓰기로 했다.

(4) 영향 요소

일사량은 단위면적이 단위시간에 받는 태양 복사 에너지[MJ/m^2]의 양으로 지표면에 도달한 태양 에너지의 양을 말한다. 일사량은 공기 분자, 먼지, 수증기, 오염물질, 구름,습도 등에 의해서 감소하게 된다. 또한, 태양의 고도가 낮으면 태양빛이 대기를 통과하는 거리가 멀어지기 때문에 태양의 고도에도 영향을 받는다.

일조시간은 일조량이라고 불리기도 하지만 이는 시간적 개념에 해당하기 때문에 의미의 혼동을 피하기 위해 일조시간이라고 부르는 것이 더 적절하다. 일조시간은 태양광선이 지표를 내리쬔 시간으로, 태양광선이 구름이나 안개 혹은 그림자로 가려지지 않고 지표면에 도달한 지속시간을 의미한다.

태양광 발전은 pn 접합 반도체 소자로 구성된 태양전지에 태양광이 입사하면 반도체에서 빛 에너지를 전기 에너지로 바꾸는 식으로 작동한다. 이 반도체의 특성 상 고온에서는 효율이 떨어진다. 여름철 반도체 소자인 태양전지가 햇빛을 받으면 온도가 상승해 태양전지의 효율이 떨어지는데, 이때 태양전지 모듈의 온도는 일사량과 풍속 등의 기상 요소에 의해 영향을 받게 된다.

모듈 온도에 영향을 미치는 요소에는 일사량과 대기온도 그리고 풍속이 있다. 일사량과 대기온도의 상승은 모듈 온도를 상승, 풍속의 증가는 모듈온도를 낮춰준다. 일사량이 상승하면 발전량을 증가시키지만, 대기온도와 모듈온도 또한 같이 높아진다. 어느 정도 이상 모듈온도가 상승하면 발전효율이 떨어지게 된다. 한편 풍속이 증가하면 모듈온도를 낮춰주는 효과로 인해 발전효율을 높일 수 있다.

우리는 이번 탐구에서 일조량과 일사량 중 어떠한 요소가 태양광 발전량에 더 큰 영향을 미칠 지 알아보기로 했다.

(5) 회귀모델의 평가지표

1. **MAE (Mean Absolute Error) :** 평균 절대 오차

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left| x_i - x \right|$$

- MAE는 모델의 예측값과 실제값의 차이의 절대값의 평균으로, 절대값을 취하기 때문에 가장 직관적으로 알 수 있는 지표이다 즉, 해석에 용이하다.

- Ourlier에 민감하지 않다. Outlier는 이상치인데, 이는 보통 관측된 데이터의 범위에서 많이 벗어난 아주 작은 값이나 큰 값을 말한다.

- 절대값을 취하기 때문에 모델이 Underperformance인지 Overperformance인지 알 수 없다.

- 함수 값에 미분 불가능한 지점이 있다.

2. MSE (Mean Squared Error) : 평균 제곱 오차

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- 모델의 예측값과 실제값 간의 평균 제곱 차이를 말한다.

- 제곱을 하기 때문에 Outlier에 민감하다. 예를 들자면, 절댓값이 1 미만인 값은 더 작아지고, 1 초과인 값은 더 커지는 왜곡이 발생할 수 있다.

- Outlier에 민감하다.

- 모든 함수 값에서 미분이 가능하다.

3. RMSE (Root Mean Sqaured Error) : 평균 제곱근 오차

$$RMES = \sqrt{MSE} = \sqrt{\frac{\sum(\hat{y}-y)^2}{n}}$$

- MSE에 루트를 씌워 사용하는 것으로, 오류 지표를 실제값과 유사한 단위로 다시 변환하여 해석이 용이하다.

- 예측 대상의 크기에 영향을 바로 받는다.

- Outlier에 대한 민감도가 MSE보다 작고 MAE보다 크기 때문에 이를 적절히 잘 다룬다고 간주된다.

- MSE는 부드러운 곡선형으로 오차함수가 그려지지만, MSE에 루트를 취한 값이기 때문에 미분 불가능한 지점이 존재한다.

우리는 이번 탐구에서 회귀 모델의 최적화를 위해 Outlier에 대한 민감도가 MSE보다는 작고 MAE보다 큰 **RMSE**를 사용하기로 했다.

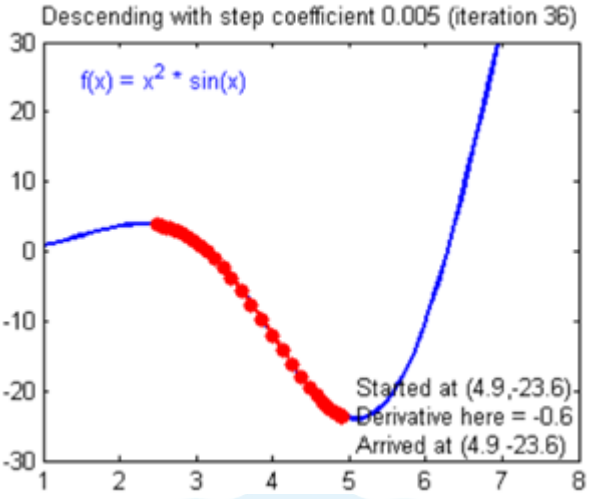
(6) 경사하강법

- 함수의 값이 낮아지는 방향으로 각 독립변수들의 값을 변형시키며 함수가 최솟값을 갖도록 하는 독립변수의 값을 탐색하는 방법을 말한다.

- 인공지능의 경우 최적의 학습 패턴을 위해 피라미터의 검증이 필요하며, 그 과정에서 '손실 함수'를 사용한다. 이 손실 함수의 값이 가장 낮은 피라미터를

찾아야 최적의 피라미터임이 증명된다.

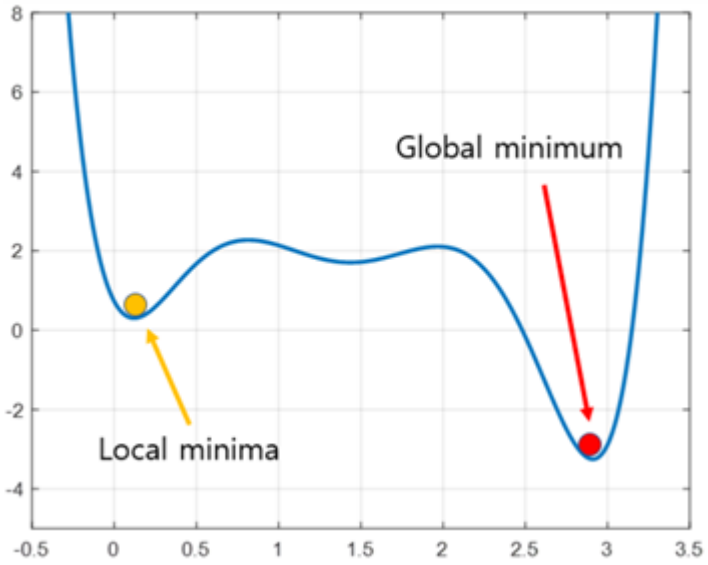
(7) 경사하강법의 과정 및 단점



- 특정 지점에서 x 값이 증가함에 따라 함수의 값이 증가한다면 음의 방향으로, 반대로 x 값이 증가함에 따라 함수의 값이 감소한다면 양의 방향으로 옮겨가면 된다.

- 값의 이동 사이즈 (step size)를 너무 크게 잡으면, 빠르게 수렴점을 찾을 수 있다는 장점이 있지만, 함수의 값이 커지는 방향으로 최적화가 진행될 수도 있다는 단점이 있다.

- 반대로, step size를 너무 작게 설정한다면 최적화가 올바른 방향으로 진행되지만 최적의 파라미터를 탐색할 때 소요되는 시간이 굉장히 오래 걸릴 수 있다는 단점이 있다.



- 또한, 극솟값 중 지역 극솟값(Local minima)가 존재할 수 있는데, 이 경우에서는 경사 하강법의 특성상 알고리즘이 시작되는 파라미터 위치가 랜덤이기에 완전한 최적 파라미터를 찾지 못할 수 있다.

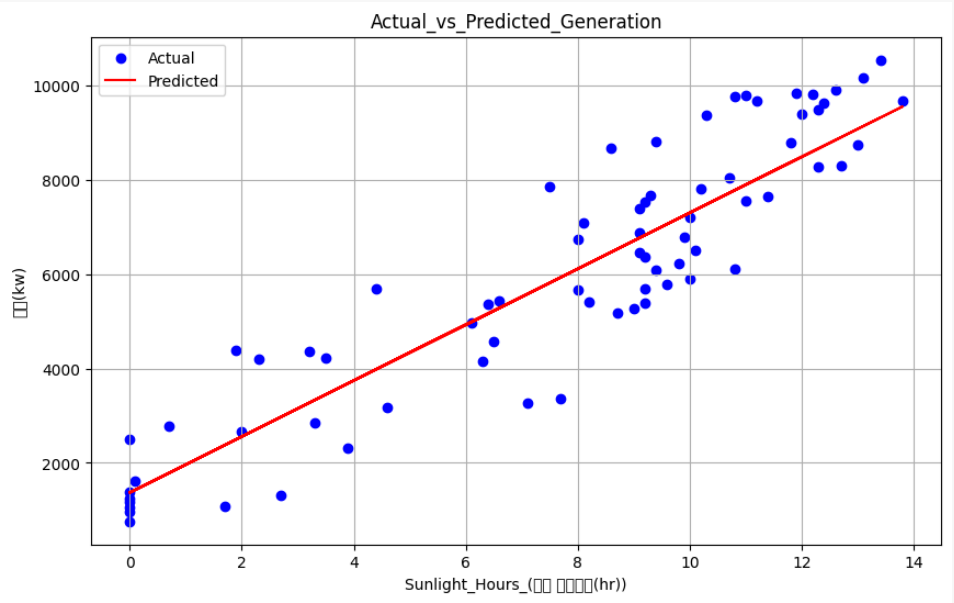
결과 **(RESULTS)**

이번 탐구에서 우리는 구글 코랩을 활용했다.

(1) 평균 일조시간에 따른 신인천 태양광 발전량 예측 선형회귀 모델

```
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
import matplotlib .pyplot as plt
from sklearn. model_selection import train_test_split
from sklearn. linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

data = pd.read_csv ('/content/drive/MyDrive/에너지공학동아리/신인천전망대 태양광발전량(일조량) 2019년.csv', encoding='cp949')
data

data['년월일'] = pd. to_datetime (data['년월일'])
X = data [['평균 일조시간(hr)']]
y = data['총량(kw)']
```

```
X_train , X_test , y_train , y_test = train_test_split (X, y, test_size
=0.2 , random_state =0)
model = LinearRegression ()
model.fit(X_train , y_train)
y_pred = model.predict(X_test)

coef = model.coef_ [0]
intercept = model. intercept_
print(f'y={coef:.2f}x+{intercept:.2f}')

rmse = mean_squared_error (y_test , y_pred , squared=False)
print('Root_Mean_Squared_Error␣(RMSE):', rmse)

correlation = data['총량(kw)']. corr(data['평균 일조시간(hr)'])
print('Correlation_coefficient:', correlation )

plt.figure(figsize =(10 , 6))
plt.scatter(X_test , y_test , color='blue', label='Actual')
plt.plot(X_test , y_pred , color='red', label='Predicted')
plt.title('Actual_vs_Predicted_Generation')
plt.xlabel('Sunlight_Hours_(평균 일조시간(hr))')
plt.ylabel('총량(kw)')
plt.legend ()
plt.grid(True)
plt.show ()
```

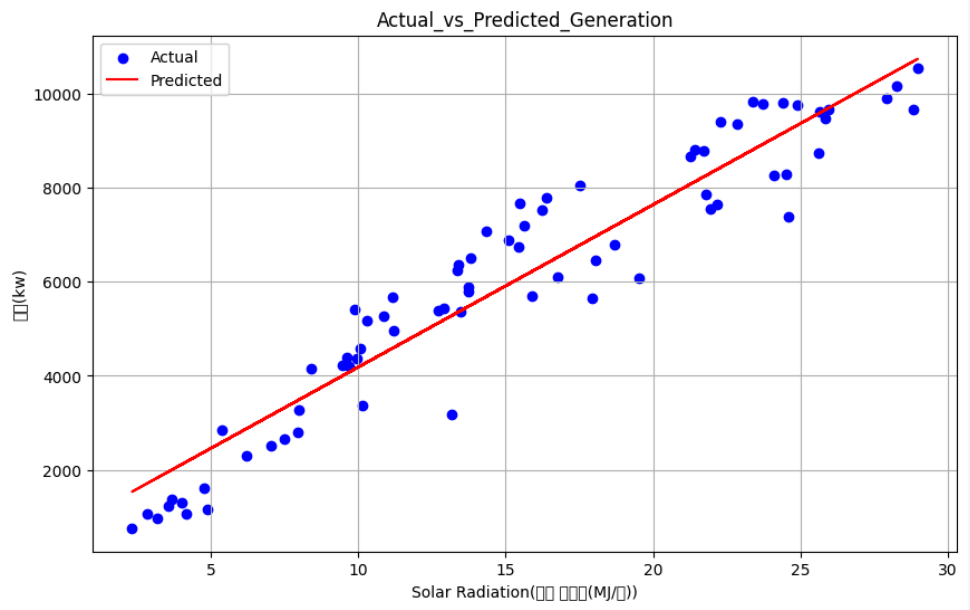



```
Root_Mean_Squared_Error␣(RMSE): 1121.643994131298
Correlation_coefficient: 0.9034855379719439
```

(2) 평균 일사량에 따른 신인천 태양광 발전량 예측 선형회귀 모델

```
from google.colab import drive
drive.mount('/content/drive')

import pandas as pd
import matplotlib .pyplot as plt
from sklearn. model_selection import train_test_split
from sklearn. linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

data = pd.read_csv ('/content/drive/MyDrive/에너지공학동아리/신인천전망대
태양광발전량(일사량) 2019년.csv', encoding='cp949')
data

data['년월일'] = pd. to_datetime (data['년월일'])
X = data [['평균 일사량(MJ/㎡)']]
y = data['총량(kw)']

X_train , X_test , y_train , y_test = train_test_split (X, y, test_size
=0.2 , random_state =0)
model = LinearRegression ()
model.fit(X_train , y_train)
y_pred = model.predict(X_test)

coef = model.coef_ [0]
intercept = model. intercept_
print(f'y={coef:.2f}x+{intercept:.2f}')

rmse = mean_squared_error (y_test , y_pred , squared=False)
print('Root␣Mean␣Squared␣Error␣(RMSE):', rmse)

correlation = data['총량(kw)']. corr(data['평균 일사량(MJ/㎡)'])
print('Correlation_coefficient:', correlation )

plt.figure(figsize =(10 , 6))
plt.scatter(X_test , y_test , color='blue', label='Actual')
plt.plot(X_test , y_pred , color='red', label='Predicted')
plt.title('Actual_vs_Predicted_Generation')
plt.xlabel('Solar Radiation_(평균 일사량(MJ/㎡))')
plt.ylabel('총량(kw)')
plt.legend ()
```

```
plt.grid(True)
plt.show ()
```




```
Root_Mean_Squared_Error␣(RMSE): 840.8004945147413
Correlation_coefficient: 0.9496164994330775
```

토의 **(DISCUSSION),** 결론 **(CONCLUSION)**

질문 ***1:*** 태양광 발전량에 가장 큰 영향을 미치는 요인은 무엇인가**?**

이번 탐구의 결과를 정리하면 다음과 같다.

	RMSE	R(상관계수)
일조시간 (Sunlight Hour)	1121.644	0.9035
일사량 (Solar radiation)	840.800	0.9496

- 일조시간보다 일사량이 태양광 발전량에 더 큰 영향을 미치는 것으로 해석할 수 있었다.

- 일조시간의 RMSE 값이 일사량의 RMSE 값에 비해 큰 것을 확인해 일사량의 직선이 더 최적화된 직선이며 오차가 적다는 것을 알 수 있었다.

질문 ***2:*** 선형 회귀 분석의 예측 정확도와 속도를 높이기 위한 방법

(1) 정확도를 높이는 방법

- 다중 공신성은 종속변수와 독립변수가 아니라 독립변수들 간에 상관관계가 강할 때 각 변수들이 종속변수에 어느 정도 영향을 미치는 지 정확히 알 수 없는 문제를 말한다. 단순 선형 회귀는 상관 없지만, 독립변수가 여러 개인 다중 선형회귀 모델을 사용하고자 한다면 이를 신경써야 한다.

- 다중 공선성을 확인하고 필요한 경우 변수를 제거하거나 다른 방법으로 처리하여 정확도를 높일 수 있다. 그리고 데이터의 이상치가 존재할 경우 모델의 성능을 저하시킬 수 있으므로 이상치를 식별하고 처리함으로써 모델의 예측 정확도를 높일 수 있다. 또한 교차 검증을 사용하여 모델의 일반화 성능을 평가하고 모델 선택을 지원하여 과적합을 방지하고 더 정확한 모델을 개발할 수 있다.

(2) 속도를 높이는 방법

- 불필요하거나 상관관계가 낮은 특성을 제거하여 데이터의 차원을 줄이면 모델의 예측 속도가 빨라질 수 있다. 모든 변수를 모델에 포함시키지 않고 중요한 변수만 선택하여 모델의 복잡성을 줄이고 계산 속도를 높일 수 있다. 변수 선택 알고리즘을 사용하여 중요한 변수를 식별하고 선택하여 속도를 줄인다. 또한 데이터를 정규화하거나 표준화하면 모델의 수렴 속도가 빨라질 수 있다. 특정 스케일링을 통해 데이터의 스케일을 맞추면 예측 성능과 속도가 향상될 수 있다.

<Gradient Descent를 미소한 단위로 감소하도록 설정 즉, Learning rate 증가시킴>

- 선형 회귀 모델의 학습 속도를 높이기 위해 학습률을 증가시킬 수 있다. 모델이 더 빠르게 수렴함으로써 모델의 훈련 시간이 단축된다. 그러나 단점으로는, 너무 큰 학습률은 학습을 불안정하게 만들어 모델이 최적의 해에 수렴하지 못하고 발산할 수 있다. 또한 큰 학습률은 최적값을 자주 넘기게 되고 진동이 커진다.

- **Learning Rate Schedule(**최적의 학습률 설정 방법**)**: 학습 초기에는 큰 학습률을 사용하고, 시간이 지남에 따라 학습률을 점차 줄여나가는 방법이다. 이 방법은 초기 학습 속도를 높이고 안정적인 수렴을 돕는다.

- **Exponential Decay (**지수 감소**)**: 초기 학습률을 설정한 후, 매 에포크마다 학습률을 지수적으로 감소시킨다.

- **Polynomial Decay (**다항식 감소**)**: 학습률이 다항식 형태로 감소한다.

- **Inverse Scaling Decay (**역비례 감소**)**: 학습률이 시간에 반비례하여 감소한다.

- **Cosine Annealing**: 학습률을 주기적으로 감소시켜 초기값으로 돌아오도록 설정한다.

- **Warm Restarts**: 학습률을 주기적으로 크게 설정하고 다시 감소시키는 방법이다.

1) 다음은 지수 감소 방식을 사용하여 학습률을 미소한 단위로 감소시키는 코드식이다.

```
# 인천 태양광 발전량(일사량) 예측을 사용함. 위 선형회귀모델에 추가하여 활동.

import numpy as np
import time
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.callbacks import LearningRateScheduler

initial_lr = 0.1
decay_rate = 0.01

def learning_rate_schedule(epoch, lr):
    return initial_lr * np.exp(-decay_rate * epoch)
lr_scheduler = LearningRateScheduler(learning_rate_schedule)
```

```
start_time=time.time()
model.fit(X_train,y_train)
end_time=time.time()
training_time=end_time-start_time
print(f'학습에 걸린 시간: {training_time:.4f}초')
```

처음 학습에 걸린 시간: **0.00183**초
학습율 감소 후 학습에 걸린 시간: **0.00163**초
시간이 줄어들었음을 확인할 수 있음.

2) 다음은 IQR을 사용한 이상치 제거 함수를 사용하여 이상치를 제거하는 함수 코드식이다.

```
# 인천 태양광 발전량(일사량) 예측을 사용함.

import pandas as pd
from sklearn.model_selection import train_test_split

def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <=
upper_bound)]

for column in data.columns:
    df = remove_outliers(data, column)

X = df.drop('평균 일사량(MJ/㎡)', axis=1).values
y = df['평균 일사량(MJ/㎡)'].values
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

print(f'이상치 제거 후 남은 데이터 샘플 수: {len(df)}')
```

이상치 제거 후 남은 데이터 샘플 수: **365 /** 이상치가 없음을 확인할 수 있음.

질문 ***3***: X와 Y가 비선형 관계를 가질 때, 회귀분석에 대한 아이디어를 제시한다.

(1) Y=wX^2 +b 의 관계를 가지고 있다고 유추한 후, X 데이터셋을 Z = X^2 로 치환하여 Y= wZ + b 형태의 선형 회귀를 진행한다.

(2) 다항 회귀 **(Polynomial Regression)**: 변수 XXX를 다항식 형태로 변환하여 선형 회귀 모델에 적용한다. 예를 들어, Y=aX2+bX+cY = aX^2 + bX + cY=aX2+bX+c 형태의 관계를 다룰 때, XXX를 [X,X2][X, X^2][X,X2]로 변환하여 선형 회귀를 수행한다.

(3) 로그 변환 **(Log Transformation)**: 데이터가 기하급수적으로 증가하는 경우, 로그 변환을 통해 선형 관계로 변환할 수 있다. 예를 들어, Y=a⋅ln(X)+bY = a \cdot \ln(X) + bY=a⋅ln(X)+b 형태의 관계를 ln(X)\ln(X)ln(X)로 변환하여 선형 회귀를 수행한다.

(4) 지수 변환 **(Exponential Transformation)**: 데이터가 지수 함수 형태를 띄는 경우, 지수 변환을 통해 선형 관계로 변환할 수 있다. ln(Y)=bX+ln(a)\ln(Y) = bX + \ln(a)ln(Y)=bX+ln(a)로 변환하여 선형 회귀를 수행한다.

참고문헌 **(REFERENCES)**

출처표기법

최동배,『스마트그리드(smartgrid)』, 인포더북스 (2016), 페이지.

김종욱, 김문겸,『스마트 그리드 개론』, 도서출판 홍릉(홍릉과학출판사) (2012), 페이지.

김지효, 김현제,『에너지전환 정책의 성과 및 향후 추친방향 연구』, 에너지경제연구원, 2021.
게재면.

조강의,『기후변화 대응을 위한 인천의 신재생에너지 사업 연구에 관한 연구』, 환경브릿지연구소, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville,『Deep Learning』

(인천뉴스) 황남건,「인천 신재생에너지 보급률 '저조'」,『경기일보』, 2024 June 30, https://www.kyeonggi.com/article/20240626580359 (2024.07.19)

전력통계정보시스템(EPSIS)

제 10차 전력수급기본계획

2023년도판 한국전력통계(제91호)

차왕철,『태양광발전에 영향을 미치는 요소 분석을 통한 연간 발전량 예측에 관한 연구』, 2015.

한국남부발전(주)_신인천전망대 태양광발전실적

https://www.data.go.kr/data/15043395/fileData.do