화재 예측 모델

학번: 2118031

이름: 이웅섭

Github address: https://github.com/2118031Leewoongsub/Burning_Prediction

1. 화재(산불) 예측 모델 개발의 목적

https://www.kaggle.com/에서 포르투갈 몬테지뉴 자연공원에서 발생한 517 건의 화재가 기록되어 있는 데이터를 바탕으로 강수량, 온도, 상대습도, 풍속을 고려하여 FFMC(미세연료지수)를 구한 뒤 예측값과 실제값을 비교해보며 앞으로 일어날 화재를 예방하여 막을 수 있을 것이다.

2. 화재(산불) 예측 모델의 네이밍의 의미

이름 그대로 데이터의 강수량, 온도, 상대습도, 풍속을 고려하여 계산한 뒤 화재확률을 예측하는 모델이다.

3. 개발 계획

데이터 파일 이름은 forestfires 으로 temp, RH, rain, wind 등과 같은 데이터를 가지고 있다.

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.00
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.00
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.00
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.00
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.00
..	..	..	...	...	...	...	...	...	...	..	...	...	...
512	4	3	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44
513	2	4	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	54.29
514	7	4	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	11.16
515	1	4	aug	sat	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	0.00
516	6	3	nov	tue	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00

선형 회귀 모델을 사용하여 데이터를 계산할 것이다. 여기서 선형 회귀 모델은 패턴을 직선으로 모델링하는 방법으로 주어진 입력에 대해 축력을 예측하는데 사용한다.

사용된 성능 지표는 평균 제곱 오차, 결정계수이다. 평균 제곱 오차의 경우 예측한 값과 실제 값 간의 평균 제곱오차를 나타내며 결정 계수의 경우 종속 변수의 분산을 얼마나 잘 설명하는지를 나타내는 지표이다.
이 머신 러닝 모델이 예측이 정확하다면 그래프는 모여 있을 것이다.

4. 개발 과정

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
filename = './archive/forestfires.csv'
data = pd.read_csv(filename)
print(data)
```

우선 파일을 불러오고 data 라는 변수로 저장한 뒤 데이터의 값이 정확히 불러오는지 확인했다.

```
X = data[['temp', 'RH', 'rain', 'wind']]
y = data['FFMC']
```

이후 사용할 특성과 예측할 변수를 선택했다.

```
model = LinearRegression()
model.fit(X, y)
predictions = model.predict(X)
mse = mean_squared_error(y, predictions)
r2 = r2_score(y, predictions)
```

변수와 특성을 선택한 뒤 선형회귀 모델을 선택하고 모델에 학습시켰으며 평균 제곱 오차와 결정 계수를 사용하여 모델의 성능을 평가하려고 했다.

```
plt.scatter(y, predictions, c=['blue' if actual > predicted else 'red' for actual, predicted in zip(y, predictions)])
plt.xlabel('Actual FFMC')
plt.ylabel('Predicted FFMC')
plt.title('Actual vs Predicted FFMC')
plt.show()
```

FFMC 의 값과 모델의 예측값을 비교하는 산점도를 그려 시각적으로 확인했다.

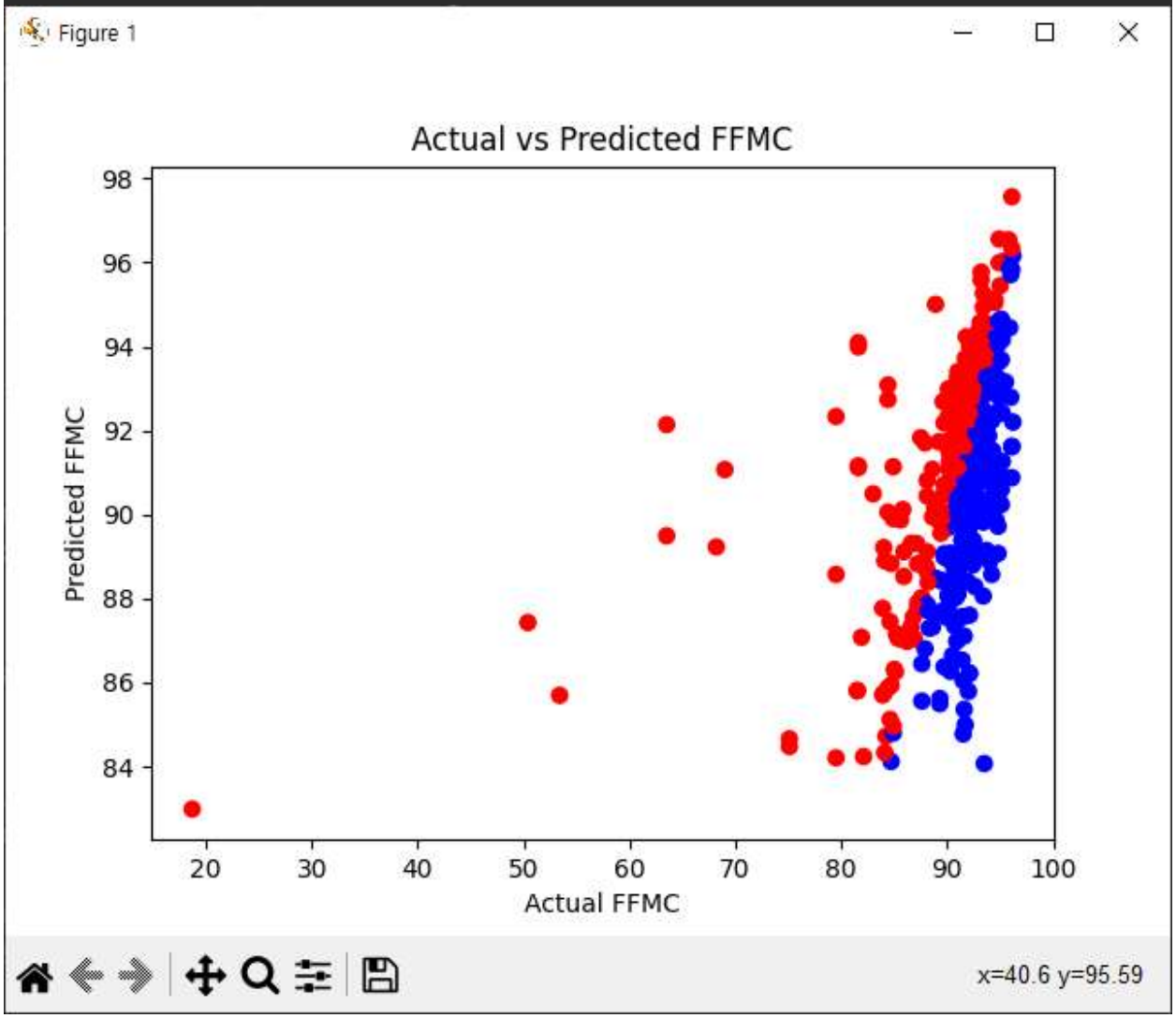
```
print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')
```

마지막으로 성능지표를 출력하여 결과를 확인했다.

```
plt.scatter(y, predictions)
plt.xlabel('Actual FFMC')
plt.ylabel('Predicted FFMC')
plt.title('Actual vs Predicted FFMC')
plt.show()
```

위의 그림과 같이 처음에는 Actual FFMC 와 Predicted FFMC 를 구별할 수 없었으나 밑에 그림처럼 코드를 바꾸어 구별하게 하였다.

```
plt.scatter(y, predictions, c=['blue' if actual > predicted else 'red' for actual, predicted in zip(y, predictions)])
plt.xlabel('Actual FFMC')
plt.ylabel('Predicted FFMC')
plt.title('Actual vs Predicted FFMC')
plt.show()
```



5. 개발 후기

우리가 생각하는 프로그래밍의 머신러닝 모델은 많이 복잡하고 어려운 것이라고 생각하고 있었는데 방대한 양의 정확한 데이터만 가지고 있다면 어렵지 않게 코드들을 참고하며 만들 수 있는 것이라고 생각하게 되었다.