Earthquake prediction

학번: 2018066

이름: 배정훈

Github address:

1. 안전 관련 머신러닝 모델 개발의 목적

a. 학습 모델 활용 대상: 지진 예측

b. 시간, 위도, 경도를 독립 변수로 사용하여 지진의 크기와 깊이를 예측합니다.

c. 개발의 의의: 우리나라도 점점 지진의 발생 빈도가 증가하고 있고 혹여 갑자기 강도 높은 지진이 오게 된다면 속수무책으로 무너져 내릴 수 있기 때문에 지진 예측 머신 러닝의 개발은 중요하다.

2. 안전 관련 머신러닝 모델의 네이밍의 의미

a. Earthquake prediction 은 말 그대로 지진 예측의 의미를 가지고 있고 의미 그대로 이름을 따왔다.

3. 개발 계획

1. 데이터 로드 및 전처리
2. Timestamp 생성 및 데이터 정제
3. 데이터 분할
4. 랜덤 포레스트 모델 학습
5. 모델 평가
6. 그리드 서치를 통한 하이퍼파라미터 튜닝
7. 최적 모델 평가
8. 결과 출력

4. 개발 과정

```python
import pandas as pd
import datetime
import time
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV

filename = "./data/database.csv"
data = pd.read_csv("./data/database.csv")

# 필요한 열만 추출합니다.
data = data[['Date', 'Time', 'Latitude', 'Longitude', 'Depth', 'Magnitude']]
```

‘Padas ‘라이브러리로 데이터를 로드하고 불러옵니다.

```python
# Timestamp 열을 생성합니다.
timestamp = []
for d, t in zip(data['Date'], data['Time']):
    try:
        ts = datetime.datetime.strptime(d + ' ' + t, '%m/%d/%Y %H:%M:%S')
        timestamp.append(time.mktime(ts.timetuple()))
    except ValueError:
        timestamp.append('ValueError')

data['Timestamp'] = pd.Series(timestamp).values

# 최종 데이터셋을 생성합니다.
final_data = data[data.Timestamp != 'ValueError'].drop(['Date', 'Time'], axis=1)
```

‘Date’와 ‘Time’을 합쳐 Unix timestamp 로 변환하고. 변환 중에 발생하는 오류는 ‘ValueError’로 표시한다.

```python
# 데이터 분할
X = final_data[['Timestamp', 'Latitude', 'Longitude']]
y = final_data[['Magnitude', 'Depth']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

독립 변수와 종속 변수를 작성하고 데이터를 학습용과 테스트용으로 나눈다.

```python
# 랜덤 포레스트 회귀 모델 생성 및 학습
reg = RandomForestRegressor(random_state=42)
reg.fit(X_train, y_train)
```

랜덤 포레스트 회귀 모델을 생성하고 학습시킨다.

```
# 모델 평가
score = reg.score(X_test, y_test)

# 그리드 서치를 사용한 하이퍼파라미터 튜닝
parameters = {'n_estimators': [10, 20, 50, 100, 200, 500]}
grid_obj = GridSearchCV(reg, parameters)
grid_fit = grid_obj.fit(X_train, y_train)
best_fit = grid_fit.best_estimator_
```

모델을 평가하고 랜덤 포레스트 모델의 하이퍼파라미터 중 'n_estimators'에 대해 그리드 서치를 수행한다

```
# 최적화된 모델을 사용한 모델 평가
best_score = best_fit.score(X_test, y_test)

# 결과 출력
print(f"초기 모델 점수: {score}")
print(f"최적 모델 점수: {best_score}")
```

최적화된 모델 평가하고 초기 모델과 최적 모델의 점수를 출력해 비교해본다.

5. **개발 후기**

쉽지 않았다. 확실히 개발하는 과정까지 오니 하나부터 열까지 만드는 게 쉽지 않고 기초도 중요하지만 기초를 기반으로 얼마나 잘 활용하는지 혹은 적용하는지가 가장 중요하다고 생각한다.

코드 이해하는 것도 힘들었다. 분명히 배운 코드지만 흉측하게 쓰여져 있어서 이해하기 쉽지 않았다. 더욱 많은 코드를 직접 써보고 다른 사람들이 쓴 코드를 많이 보면 어느 순간을 기점으로 보이지 않을 까라는 생각이 들었다.