Multiple_Linear_Regression

Multiple Linear Regression

핵심 개념

- **다중 선형 회귀(Multiple Linear Regression)**: 여러 개의 독립 변수를 사용하여 종속 변수를 예측하는 모델
- **최소 제곱법(Least Square Method)**:

$$\theta = (X^T \cdot X)^{-1} \cdot (X^T \cdot Y)$$

  를 통해 최적의 회귀 계수를 찾는 방법
- **가우시안 정규화**: 데이터를 평균 0, 표준편차 1로 변환하여 스케일을 통일

데이터 분석 결과

- 주택 가격에 가장 큰 영향을 미치는 요인:
  1. 실내 면적 (상관계수: 0.702)
  2. 건축 품질 등급 (상관계수: 0.667)
  3. 지상 층 면적 (상관계수: 0.606)
- MSE가 0.33~0.46 범위로, 모델의 예측 성능이 양호함

주의사항

1. 데이터 전처리 시 결측치 제거와 정규화가 필수
2. 새로운 데이터 예측 시 반드시 정규화 후 입력
3. 예측 결과 해석 시 역정규화 필요

모델 사용 예시

```
# 입력 데이터 정규화
sqft_living = (실제면적 - mean_sqft_living) / std_sqft_living
grade = (등급 - mean_grade) / std_grade

# 예측
price = model.predict([[sqft_living, grade]])

# 결과 역정규화
real_price = (price * std_price) + mean_price
```

Import packages

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
np.set_printoptions(precision=6, suppress=True)
```

데이터셋 로딩

```
- kc_house_data: 미국 워싱턴주 시애틀 지역의 주택 가격 데이터를 포함한 공개 데이터셋
1. id: 각 주택 거래에 대한 고유 식별자.
2. date: 주택이 판매된 날짜.
3. price: 주택의 판매 가격 (종속 변수, 목표 값).
4. bedrooms: 침실의 개수.
5. bathrooms: 욕실의 개수 (부분 욕실도 포함).
6. sqft_living: 주택의 실내 면적 (평방 피트).
7. sqft_lot: 주택의 대지 면적 (평방 피트).
8. floors: 주택의 층수.
9. waterfront: 주택이 해안가에 위치해 있는지 여부 (1: 해안가, 0: 해안가 아님).
...
```

```
# Download dataset file

# Load dataset file
data = pd.read_csv('kc_house_data.csv')
data
```

데이터셋 전처리

```
# 예측 수행에 불필요한 열 삭제
data = data.drop(['id', 'date'], axis=1)

# 값이 존재하지 않는 행 삭제
data = data.dropna()

# 데이터셋 가우시안 정규화
data_normalized = (data - data.mean()) / data.std()
data_normalized
```

데이터셋 특성 파악 (시각화, 상관계수)

```
# 상관계수 행렬 계산
correlation_matrix = data_normalized.corr()

# 상관계수 히트맵 시각화
plt.figure(figsize=(15, 12))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Feature Correlation Heatmap')
plt.tight_layout()
plt.show()

# 데이터의 각 열에 대한 상관관계 확인
correlation_matrix = data_normalized.corr()
correlation_matrix
```

상관계수 행렬 분석 결과:

- 주택 가격(price)과 가장 높은 상관관계를 보이는 변수들:
  - sqft_living (0.702): 주택의 실내 면적
  - grade (0.667): 주택의 건축 품질과 디자인 등급
  - sqft_above (0.606): 지상 층의 면적
  - sqft_living15 (0.585): 인근 주택들의 평균 실내 면적

- 주택 가격과 낮은 상관관계를 보이는 변수들:
  - zipcode (-0.053): 우편번호
  - condition (0.036): 주택의 상태
  - long (0.022): 경도
  - sqft_lot (0.090): 대지 면적

집 가격 예측에 사용할 데이터 지정

```
# 예측에 사용할 데이터들에 대한 2차원 행렬 변환

# MSE : 0.33
X = np.array(data_normalized[['sqft_living', 'grade', 'sqft_above', 'bathrooms', 'sqft_living15',
'bedrooms', 'floors', 'waterfront', 'view', 'condition', 'sqft_basement', 'yr_built',
'yr_renovated', 'zipcode', 'lat', 'long', 'sqft_lot', 'sqft_lot15']])

# MSE : 0.45
X = np.array(data_normalized[['sqft_living', 'grade', 'sqft_above']])

X = np.array(data_normalized[['sqft_living', 'grade']])
Y = np.array(data_normalized[['price']])

# Train dataset / Test dataset 분할
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=1234)

# Train dataset 형상 확인
print(X_train.shape)
print(Y_train.shape)
```

Least Square Method 기반 선형 회귀 모델 작성

- Least Square Method:

$$\theta = (X^T \cdot X)^{-1} \cdot (X^T \cdot Y)$$

```
class LinearRegression_LSM():

    def __init__(self):
        self.theta = None

    def fit(self, X, Y):
        N = X.shape[0] # N = 입력 데이터 개수

        # 입력 X에 대해 bias 차원 추가
        bias = np.ones((N, 1))    # N x 1
        X = np.hstack([X, bias])  # N x 2

        # theta (W, b) 저장을 위한 배열 초기화
        self.theta = np.zeros(X.shape[1])

        # Least Square Method 수행
        XT = X.T
        XTX = np.dot(XT, X)
        XTX_inverse = np.linalg.inv(XTX)
        XTY = np.dot(XT, Y)

        self.theta = np.dot(XTX_inverse, XTY)

        return self.theta

    def predict(self, X):
```

```
        N = X.shape[0] # N = 입력 데이터 개수

        # 입력 X에 대해 bias 차원 추가
        bias = np.ones((N, 1)) # N x 1
        X = np.hstack([X, bias]) # N x 2

        Y_hat = np.dot(X, self.theta)
        return Y_hat
```

모델 학습 수행

```
model_LSM = LinearRegression_LSM()
theta = model_LSM.fit(X_train, Y_train)

print("theta = ", theta)
```

모델 성능 평가

Mean Squared Error (MSE) 계산:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

```
# 정답 데이터와 예측 데이터 간 차이 계산
def MSE(Y, Y_hat):
    error = Y - Y_hat
    mse = np.mean(error ** 2)
    return mse

Y_hat = model_LSM.predict(X_test)
print(MSE(Y_test, Y_hat))
```

새로운 데이터에 대한 예측

가우시안 정규화와 역정규화:

- 정규화:

$$x' = \frac{x - \mu}{\sigma}$$

- 역정규화:

$$x = x' \cdot \sigma + \mu$$

```
# 가우시안 정규화/역정규화를 위한 평균, 표준편차 저장
mean_array = data.mean()
std_array = data.std()

mean_sqft_living = mean_array['sqft_living']
mean_grade = mean_array['grade']
mean_price = mean_array['price']

std_sqft_living = std_array['sqft_living']
std_grade = std_array['grade']
std_price = std_array['price']

# 임의 데이터 X 생성
sqft_living = 1000
grade = 8
```

```python
# 각 입력 변수 X에 대한 정규화 수행
sqft_living = (sqft_living - mean_sqft_living) / std_sqft_living
grade = (grade - mean_grade) / std_grade

X_new = np.array([[sqft_living, grade]])

# 학습한 모델 theta를 이용해 Y_hat 예측
Y_hat = model_LSM.predict(X_new)

# 출력 변수 Y에 대한 역정규화 수행
Y_hat = (Y_hat * std_price) + mean_price

print(f"Y_hat = {Y_hat}")
```