### 2. 회귀 분석 (20분)

안녕하세요, 여러분. 오늘은 회귀 분석에 대해 이야기해보겠습니다. 회귀 분석은 우리가 데이터를 바탕으로 예측을 할 때 사용하는 아주 기본적이면서도 중요한 방법이에요. 먼저, 선형 회귀부터 시작해볼게요.

\*\*선형 회귀란?\*\*

- 쉽게 말해서, 선형 회귀는 두 변수 간의 관계를 직선으로 나타내는 거예요. 예를 들어, 우리가 어떤 제품의 광고비와 판매량 데이터를 가지고 있다고 해봅시다. 광고비가 많으면 판매량이 증가할 거라고 예상할 수 있겠죠? 이 관계를 수식으로 이렇게 됩니다.

- 여기서 는 예측하고자 하는 값, 즉 판매량이에요.

- 는 설명 변수, 즉 광고비죠.

- 는 회귀 계수라고 해서, 이 직선의 기울기와 절편을 의미합니다.

- 마지막으로 은 예측이 완벽하지 않기 때문에 발생하는 오차입니다.

\*\*다중 선형 회귀\*\*

- 다중 선형 회귀는 이름 그대로 설명 변수가 여러 개인 경우에요. 예를 들어, 광고비뿐만 아니라, 날씨나 계절 같은 요인도 판매량에 영향을 줄 수 있겠죠? 이때는 이런 식으로 표현합니다. 여기서 은 각각의 설명 변수들입니다.

\*\*모델 평가 지표\*\*

- 우리가 만든 모델이 얼마나 잘 작동하는지 평가하는 방법도 여러 가지가 있어요.

- \*\*MAE (Mean Absolute Error)\*\*: 실제 값과 예측 값의 차이의 절대값을 평균낸 것. 쉽게 말해, 평균적으로 얼마나 예측이 빗나갔는지 알려줍니다.

- \*\*MSE (Mean Squared Error)\*\*: 차이의 제곱을 평균낸 것. 제곱을 하다 보니, 큰 오차에 더 큰 페널티를 주는 효과가 있어요.

- \*\*RMSE (Root Mean Squared Error)\*\*: MSE의 제곱근을 씌운 것. 오차의 단위가 원래 데이터의 단위와 같아지므로 해석이 쉬워요.

- \*\*R² (Coefficient of Determination)\*\*: 이 값은 0에서 1 사이의 값을 가지는데, 1에 가까울수록 모델이 데이터를 잘 설명한다는 뜻이에요.

이제 회귀 분석에 대해 기본적인 이해가 되었을 거라 생각합니다. 다음으로는 분류 기법에 대해 이야기해볼게요.

---

### 3. 분류 기법 (20분)

자, 이제 분류 기법에 대해 이야기해봅시다. 분류 기법은 우리가 데이터를 여러 가지 범주로 나눌 때 사용하는 방법이에요. 여기서 가장 기본적인 기법 두 가지를 설명드릴게요: 로지스틱 회귀와 결정 트리입니다.

\*\*로지스틱 회귀\*\*

- 이름에 회귀가 들어가지만, 이건 분류 기법이에요. 로지스틱 회귀는 주로 이항 분류 문제, 즉 두 가지 범주로 나누는 문제에 사용됩니다. 예를 들어, 어떤 학생이 시험에 합격할지 불합격할지를 예측한다고 해볼게요.

- 선형 회귀처럼 보이지만, 결과값이 0과 1 사이의 확률로 나와요.

- 수식으로는이렇게 표현되는데, 이걸 통해 일 확률, 즉 합격할 확률을 계산할 수 있어요.

\*\*결정 트리\*\*

- 결정 트리는 데이터를 나무 구조로 분류하는 방법이에요. 각 노드는 하나의 질문을 나타내고, 이 질문에 따라 데이터를 왼쪽 가지 또는 오른쪽 가지로 나눠갑니다.

- 예를 들어, 주식이 상승할지 하락할지를 예측할 때, 첫 번째 노드는 "오늘의 거래량이 평균 이상인가?" 같은 질문이 될 수 있어요.

- 이렇게 계속 분기점을 만들어가며 최종 예측에 도달합니다.

\*\*과적합 방지\*\*

- 결정 트리에서 중요한 점 중 하나는 과적합을 방지하는 거예요. 과적합은 모델이 학습 데이터에 너무 잘 맞아 떨어져서 새로운 데이터에는 잘 맞지 않는 현상이에요. 이를 방지하기 위해 Pruning, 즉 가지치기를 합니다. 이는 불필요하게 복잡한 가지를 제거하여 모델의 일반화를 도와줍니다.

\*\*모델 평가 지표\*\*

- 분류 모델의 성능을 평가하는 지표들도 여러 가지가 있어요.

- \*\*정확도 (Accuracy)\*\*: 전체 예측 중 맞춘 비율을 의미해요.

- \*\*정밀도 (Precision)\*\*: 양성으로 예측한 것 중에서 실제 양성의 비율이에요.

- \*\*재현율 (Recall)\*\*: 실제 양성 중에서 맞춘 비율이죠.

- \*\*F1 스코어\*\*: 정밀도와 재현율의 조화 평균으로, 두 지표의 균형을 봅니다.

이제 회귀 분석과 분류 기법의 기본 개념을 다뤘습니다. 다음 시간에는 Kaggle 주식 시장 데이터를 활용해 실제로 모델을 구축해보는 실습을 진행하겠습니다. 질문이 있으면 언제든지 물어보세요!

좋습니다. 이제 릿지(Ridge) 회귀와 랏소(Lasso) 회귀에 대해 설명드리겠습니다. 이 두 가지 방법은 모두 회귀 분석에서 정규화를 통해 모델의 성능을 향상시키고 과적합을 방지하는 기법이에요.

### 릿지 회귀 (Ridge Regression)

\*\*릿지 회귀란?\*\*

- 릿지 회귀는 선형 회귀에 L2 정규화를 추가한 모델입니다. L2 정규화는 회귀 계수의 제곱합을 최소화하는 제약을 두어 모델이 너무 복잡해지지 않도록 합니다.

- 기본 아이디어는 회귀 계수가 너무 크지 않도록 제어해서 모델이 데이터를 지나치게 잘 맞추는 것을 방지하는 거예요.

\*\*수식\*\*

- 선형 회귀의 손실 함수에 페널티 항을 추가한 형태로, 다음과 같이 표현됩니다:

- 여기서 는 정규화 강도를 조절하는 하이퍼파라미터입니다.

- 가 클수록 페널티가 커져서 회귀 계수들이 작아지게 되고, 과적합을 방지할 수 있습니다.

\*\*장점\*\*

- 변수 간의 상관관계가 높을 때 특히 효과적입니다.

- 모든 변수를 모델에 포함시켜 과적합을 방지하면서 예측력을 유지합니다.

### 랏소 회귀 (Lasso Regression)

\*\*랏소 회귀란?\*\*

- 랏소 회귀는 선형 회귀에 L1 정규화를 추가한 모델입니다. L1 정규화는 회귀 계수의 절대값 합을 최소화하는 제약을 두어 모델을 간결하게 합니다.

- 이 방법은 일부 회귀 계수를 정확히 0으로 만들어서 불필요한 변수를 제거하는 효과가 있습니다.

\*\*수식\*\*

- 선형 회귀의 손실 함수에 페널티 항을 추가한 형태로, 다음과 같이 표현됩니다:

- 여기서도 는 정규화 강도를 조절하는 하이퍼파라미터입니다.

- 가 클수록 페널티가 커져서 많은 회귀 계수들이 0이 되게 합니다.

\*\*장점\*\*

- 자동으로 변수 선택을 수행하여 모델을 간결하게 만듭니다.

- 과적합을 방지하고 해석 가능성을 높입니다.

### 비교와 실습

\*\*비교\*\*

- 릿지 회귀는 모든 변수를 포함하되, 회귀 계수의 크기를 작게 만들어 과적합을 방지합니다.

- 랏소 회귀는 일부 회귀 계수를 0으로 만들어 변수 선택을 수행하고 모델을 간결하게 만듭니다.

이제 간단한 Python 코드로 릿지 회귀와 랏소 회귀를 실습해보겠습니다.

\*\*실습\*\*

```python

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import Ridge, Lasso

from sklearn.metrics import mean\_squared\_error, r2\_score

# 데이터 로드 및 전처리 (앞에서 설명한 Kaggle 주식 시장 데이터 예제 사용)

data = pd.read\_csv('path/to/stock\_market\_data.csv')

data = data.dropna()

data['Date'] = pd.to\_datetime(data['Date'])

data['Year'] = data['Date'].dt.year

data['Month'] = data['Date'].dt.month

data['Day'] = data['Date'].dt.day

X = data[['Open', 'High', 'Low', 'Volume', 'Year', 'Month', 'Day']]

y = data['Close']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 릿지 회귀 모델 훈련

ridge = Ridge(alpha=1.0)

ridge.fit(X\_train, y\_train)

y\_pred\_ridge = ridge.predict(X\_test)

print("Ridge Regression")

print("MSE:", mean\_squared\_error(y\_test, y\_pred\_ridge))

print("R²:", r2\_score(y\_test, y\_pred\_ridge))

# 랏소 회귀 모델 훈련

lasso = Lasso(alpha=0.1)

lasso.fit(X\_train, y\_train)

y\_pred\_lasso = lasso.predict(X\_test)

print("Lasso Regression")

print("MSE:", mean\_squared\_error(y\_test, y\_pred\_lasso))

print("R²:", r2\_score(y\_test, y\_pred\_lasso))

```

이 코드에서는 릿지 회귀와 랏소 회귀를 각각 훈련시키고, 테스트 데이터에 대한 예측 값을 계산하여 모델의 성능을 평가합니다. `alpha` 값은 정규화 강도를 조절하는 하이퍼파라미터로, 필요에 따라 조정할 수 있습니다.

이렇게 릿지와 랏소 회귀는 모델의 복잡성을 조절하고 과적합을 방지하는 데 유용한 방법입니다. 질문이 있으면 언제든지 물어보세요!

좋은 질문입니다! "alpha"는 릿지 회귀와 랏소 회귀에서 매우 중요한 역할을 하는 하이퍼파라미터입니다. 이를 좀 더 자세히 설명드리겠습니다.

### Alpha (Regularization Parameter)란?

\*\*정규화의 필요성\*\*

- 회귀 모델이 데이터에 너무 잘 맞추려고 하면(과적합), 새로운 데이터에 대한 예측 성능이 떨어질 수 있습니다.

- 정규화는 이런 과적합을 방지하기 위해 회귀 계수의 크기를 제어하여 모델이 너무 복잡해지지 않도록 하는 방법입니다.

\*\*Alpha의 역할\*\*

- alpha 값은 정규화 강도를 조절합니다. 릿지 회귀와 랏소 회귀에서 이 값은 페널티 항의 앞에 곱해지는 값입니다.

- 쉽게 말해서, alpha가 클수록 모델에 더 강한 페널티를 주어 회귀 계수들을 더 작게 만듭니다.

#### 릿지 회귀의 alpha

릿지 회귀에서는 L2 정규화를 사용하여 회귀 계수의 제곱합을 최소화합니다. 수식으로 보면 다음과 같습니다:

- 여기서 가 크면 회귀 계수 들이 더 작아져 모델이 더 단순해집니다.

- 반대로 가 작으면 페널티가 줄어들어 회귀 계수들이 커질 수 있습니다.

#### 랏소 회귀의 alpha

랏소 회귀에서는 L1 정규화를 사용하여 회귀 계수의 절대값 합을 최소화합니다. 수식으로 보면 다음과 같습니다:

- 여기서 가 크면 회귀 계수 중 일부가 0이 되어 변수 선택이 이루어집니다.

- 반대로 가 작으면 페널티가 줄어들어 회귀 계수들이 모두 포함될 수 있습니다.

### Alpha 값 설정의 중요성

- 적절한 alpha 값을 설정하는 것이 중요합니다. 너무 크면 모델이 과소적합(underfitting) 되어 성능이 떨어질 수 있고, 너무 작으면 과적합(overfitting)될 위험이 있습니다.

- 일반적으로 교차 검증(cross-validation)을 통해 최적의 alpha 값을 찾습니다.

### 실습 예시

이제 alpha 값을 설정하는 예제를 코드로 보여드리겠습니다.

```python

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import Ridge, Lasso

from sklearn.metrics import mean\_squared\_error, r2\_score

# 데이터 로드 및 전처리

data = pd.read\_csv('path/to/stock\_market\_data.csv')

data = data.dropna()

data['Date'] = pd.to\_datetime(data['Date'])

data['Year'] = data['Date'].dt.year

data['Month'] = data['Date'].dt.month

data['Day'] = data['Date'].dt.day

X = data[['Open', 'High', 'Low', 'Volume', 'Year', 'Month', 'Day']]

y = data['Close']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 릿지 회귀 모델 훈련 (alpha = 1.0)

ridge = Ridge(alpha=1.0)

ridge.fit(X\_train, y\_train)

y\_pred\_ridge = ridge.predict(X\_test)

print("Ridge Regression (alpha=1.0)")

print("MSE:", mean\_squared\_error(y\_test, y\_pred\_ridge))

print("R²:", r2\_score(y\_test, y\_pred\_ridge))

# 랏소 회귀 모델 훈련 (alpha = 0.1)

lasso = Lasso(alpha=0.1)

lasso.fit(X\_train, y\_train)

y\_pred\_lasso = lasso.predict(X\_test)

print("Lasso Regression (alpha=0.1)")

print("MSE:", mean\_squared\_error(y\_test, y\_pred\_lasso))

print("R²:", r2\_score(y\_test, y\_pred\_lasso))

```

이 코드는 주식 시장 데이터를 사용하여 릿지 회귀와 랏소 회귀 모델을 훈련시키고, alpha 값을 각각 1.0과 0.1로 설정하여 모델의 성능을 평가합니다. 실제 프로젝트에서는 교차 검증을 통해 최적의 alpha 값을 찾는 것이 좋습니다.

이제 alpha 값의 의미와 역할이 조금 더 명확해졌기를 바랍니다. 질문이 있으면 언제든지 물어보세요!