~/Downloads/정규세션_기초통계(1)과제 (1).ipynb

```
# ans :

# 원본 데이터 플롯
plt.figure(figsize=(12, 6))
plt.subplot(2, 2, 1)
plt.plot(data['t'], data['nt'])
plt.xlabel('t')
plt.ylabel('nt')
plt.title('Original Data')

# 로그 변환
data['log_nt'] = np.log(data['nt'])

# 로그 변환된 데이터로 회귀 모델 적용
model_log = LinearRegression()
model_log.fit(data[['t']], data['log_nt'])

# 예측값과 표준화된 잔차 계산 (로그 변환)
fitted_values_log = model_log.predict(data[['t']])
standardized_residuals_log = (data['log_nt'] - fitted_values_log) /
data['log_nt'].std()

# 로그 변환 잔차 플롯
plt.subplot(2, 2, 2)
plt.scatter(fitted_values_log, standardized_residuals_log)
plt.xlabel('Fitted values (log transformed)')
plt.ylabel('Standardized residuals')
plt.title('Residual plot (log transformed)')

# 제곱근 변환
data['sqrt_nt'] = np.sqrt(data['nt'])

# 제곱근 변환된 데이터로 회귀 모델 적용
model_sqrt = LinearRegression()
model_sqrt.fit(data[['t']], data['sqrt_nt'])

# 예측값과 표준화된 잔차 계산 (제곱근 변환)
fitted_values_sqrt = model_sqrt.predict(data[['t']])
standardized_residuals_sqrt = (data['sqrt_nt'] - fitted_values_sqrt) /
data['sqrt_nt'].std()

# 제곱근 변환 잔차 플롯
plt.subplot(2, 2, 3)
plt.scatter(fitted_values_sqrt, standardized_residuals_sqrt)
plt.xlabel('Fitted values (sqrt transformed)')
plt.ylabel('Standardized residuals')
plt.title('Residual plot (sqrt transformed)')

plt.tight_layout()
plt.show()
```

~/Downloads/정규세션_기초통계(1)과제 (1).ipynb

```
# ans :

# 선형 회귀 모델 적용
result1 = smf.ols('Y ~ X', data=data2).fit()
fitted_values = result1.predict()
standardized_residuals = result1.get_influence().resid_studentized_internal

# 요약 정보 출력
print(result1.summary())

# 잔차 플롯
plt.scatter(fitted_values, standardized_residuals)
plt.xlabel('Fitted values')
plt.ylabel('Standardized Residuals')
plt.title('Residual Plot')
plt.show()

# 로그 변환
data2['log_Y'] = np.log(data2['Y'])

# 로그 변환된 데이터로 회귀 모델 적용
model_log = smf.ols('log_Y ~ X', data=data2).fit()
fitted_values_log = model_log.predict()
standardized_residuals_log = model_log.get_influence().resid_studentized_internal

# 로그 변환 잔차 플롯
plt.scatter(fitted_values_log, standardized_residuals_log)
plt.xlabel('Fitted values (log transformed)')
plt.ylabel('Standardized residuals')
plt.title('Residual plot (log transformed)')
plt.show()

# 제곱근 변환
data2['sqrt_Y'] = np.sqrt(data2['Y'])

# 제곱근 변환된 데이터로 회귀 모델 적용
model_sqrt = smf.ols('sqrt_Y ~ X', data=data2).fit()
fitted_values_sqrt = model_sqrt.predict()
standardized_residuals_sqrt = model_sqrt.get_influence().resid_studentized_internal

# 제곱근 변환 잔차 플롯
plt.scatter(fitted_values_sqrt, standardized_residuals_sqrt)
plt.xlabel('Fitted values (sqrt transformed)')
plt.ylabel('Standardized residuals')
plt.title('Residual plot (sqrt transformed)')
plt.show()
```

~/Downloads/정규세션_기초통계(1)과제 (1).ipynb

ans : 독립변수 X의 값에 따라 종속변수 Y의 분산이 일정하다는 것을 등분산성이라고 한다. 이는 회귀모델이 다양한 X값에 대해 일관적인 예측 정확도를 유지해야 한다는 것을 의미한다.

등분산성이 위배되는 것에 대한 문제점으로

첫 번째는 모델의 예측 정확도 저하이다. 이는 등분산성이 위배되면 X값의 변화에 따라 Y의 분산이 달라지기 때문이다. 특정 X값에서 예측이 더 정확하고, 다른 X값에서는 덜 정확할 수 있다는 것을 의미한다. 결과적으로 모델의 전반적인 예측 성능이 떨어질 수 있다.

두 번째는 잔차 분석의 문제이다. 이때 잔차는 실제 값과 예측값의 차이를 의미한다. 등분산성이 충족되면 잔차가 무작위로 분포되며, 특정 패턴이 보이지 않아야 한다. 그런데 잔차가 넓어지는 경우, 잔차에 정보가 남아 있기 때문에 모델이 특정 X값에서 더 나은 예측을 제공하지 못하게 되는 것을 의미한다. 또한 잔차에 2차 함수 형태가 남아 있는 경우, 잔차에 정보가 남아 있어 모델이 비선형 관계를 제대로 반영하지 못하고 있음을 나타낸다.

마지막으로 추론의 정확성 저하이다. 회귀분석에서 등분산성이 충족되지 않으면 회귀 계수의 표준 오차가 부정확해질 수 있다. 이는 t-검정이나 F-검정 등의 통계적 검정이 잘못된 결과를 초래할 수 있음을 의미한다. 즉, 회귀계수에 대한 추론을 신뢰할 수 없게 된다.

결론적으로, 등분산성이 위배되면 모델의 예측 정확도와 통계적 추론의 신뢰성이 저하된다. 따라서 회귀 분석을 수행할 때 등분산성을 충족하는지 확인하는 것이 필요하다.