생존 분석: Kaplan-Meier 생존 곡선, Log-rank test

- 특정 사건(Event)이 발생하기까지의 시간(Time-to-event)을 분석하는 통계적 방법론
- 사건은 사망, 질병 재발, 고객 이탈, 기계 고장 등 다양하게 정의 가능
- 시간은 사건이 발생하기까지의 기간을 의미
- **핵심 특징**: '중도 절단(Censoring)' 데이터를 다룰 수 있다.
    - 연구 기간 동안 사건이 발생하지 않았거나 추적 관찰이 중단된 경우를 의미

적용 가능한 상황

- **의학/생물학**: 환자의 생존 기간 분석, 특정 치료법의 효과 비교, 질병 재발까지의 시간 예측.
- **산업 공학**: 기계나 부품의 수명 예측, 고장 발생까지의 시간 분석.
- **경제학/마케팅**: 고객의 서비스 이탈(Churn) 시점 예측, 대출 상환 기간 분석.
- **사회학**: 특정 사회 현상(예: 결혼, 이직) 발생까지의 시간 분석.

1. Kaplan-Meier 생존 곡선

- 시간에 따른 생존 확률을 추정하고 시각화하는 비모수적 방법
- 특정 시점까지 사건이 발생하지 않고 생존할 확률을 계단 함수 형태로 보여주며, 그룹별 생존율을 비교하는 데 유용

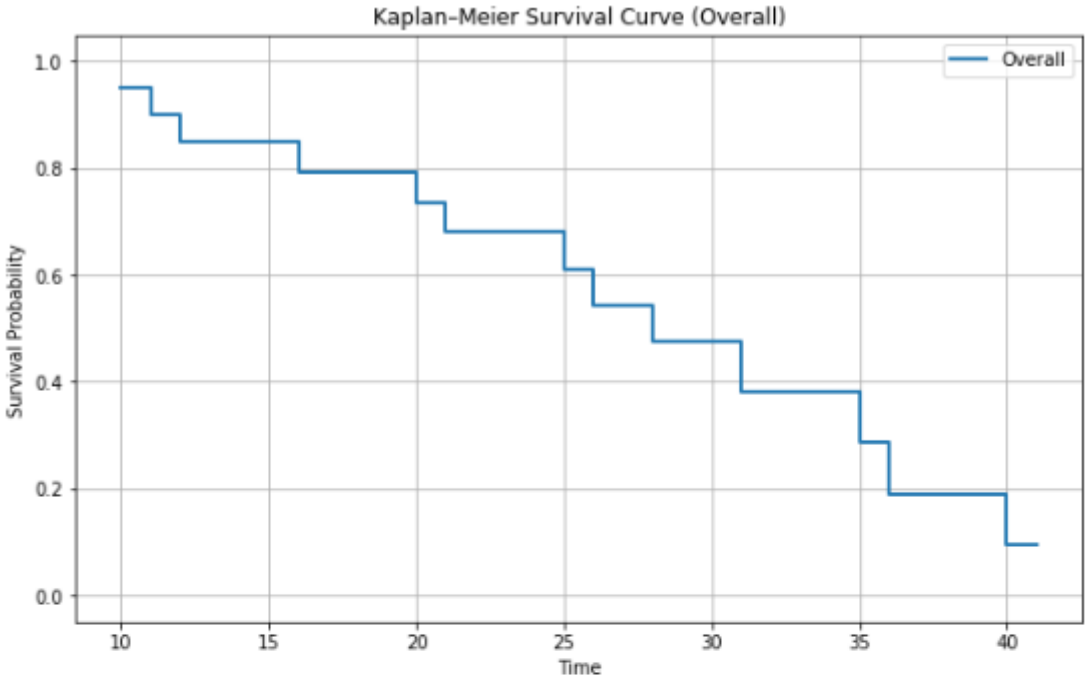
주의사항

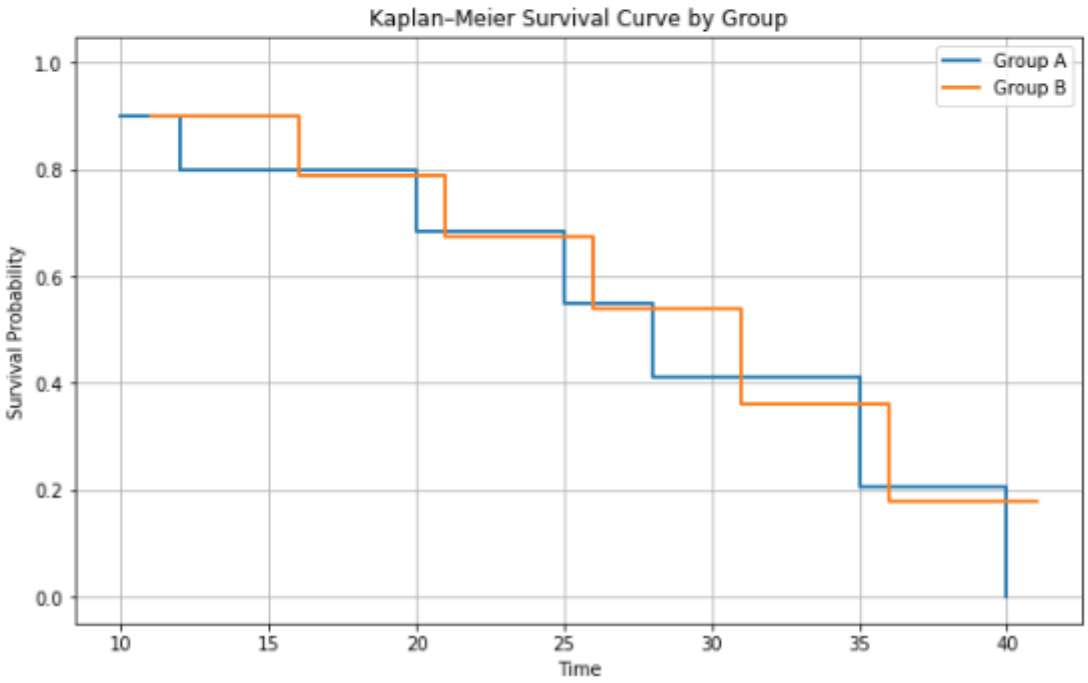
- Kaplan-Meier 곡선은 공변량(Covariates)의 영향을 직접적으로 모델링하지 않습니다. 즉, 다른 변수들의 영향을 통제한 상태에서의 생존율을 보여주지는 않습니다.
- 그룹 간 비교 시에는 통계적 유의성을 판단하기 위해 Log-rank test와 같은 추가적인 검정이 필요합니다.
- 중도 절단된 데이터가 많을수록 곡선의 신뢰도가 떨어질 수 있습니다.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# ----------------------------
# 1. 데이터 준비
# ----------------------------
data = {
    'id': range(1, 21),
    'duration': [10, 12, 15, 20, 22, 25, 28, 30, 35, 40,
                 11, 13, 16, 21, 23, 26, 29, 31, 36, 41],
    'event': [1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
              1, 0, 1, 1, 0, 1, 0, 1, 1, 0],
    'group': ['A'] * 10 + ['B'] * 10
}
df = pd.DataFrame(data)

# ----------------------------
# 2. Kaplan-Meier 함수 정의
# ----------------------------
```

```
def kaplan_meier_estimator(durations, events):
    """
    durations: iterable of times
    events: iterable of event indicators (1=event, 0=censor)
    return: pd.DataFrame with columns ['time','n_risk','n_event','survival']
    """
    durations = np.asarray(durations)
    events = np.asarray(events).astype(int)

    order = np.argsort(durations)
    durations = durations[order]
    events = events[order]

    unique_times = np.unique(durations)
    n_at_risk = []
    n_events = []
    survival_probs = []

    surv_prob = 1.0

    for t in unique_times:
        at_risk = np.sum(durations >= t)
        d = np.sum((durations == t) & (events == 1))
        if at_risk > 0:
            surv_prob *= (1 - d / at_risk)
        n_at_risk.append(int(at_risk))
        n_events.append(int(d))
        survival_probs.append(float(surv_prob))

    return pd.DataFrame({
        'time': unique_times,
        'n_risk': n_at_risk,
        'n_event': n_events,
        'survival': survival_probs
    })

# ------------------------------
# 3. 전체 생존 곡선 계산 및 시각화
# ------------------------------
overall_km = kaplan_meier_estimator(df['duration'], df['event'])

plt.figure(figsize=(10, 6))
plt.step(overall_km['time'], overall_km['survival'], where='post',
label='Overall', linewidth=2)
plt.title('Kaplan-Meier Survival Curve (Overall)')
plt.xlabel('Time')
plt.ylabel('Survival Probability')
plt.ylim(-0.05, 1.05)
plt.grid(True)
plt.legend()
plt.show()

# ------------------------------
# 4. 그룹별 생존 곡선
```

```
# ----------------------------
plt.figure(figsize=(10, 6))
for name, grouped_df in df.groupby('group'):
    km = kaplan_meier_estimator(grouped_df['duration'], grouped_df['event'])
    plt.step(km['time'], km['survival'], where='post', label=f'Group {name}',
linewidth=2)

plt.title('Kaplan-Meier Survival Curve by Group')
plt.xlabel('Time')
plt.ylabel('Survival Probability')
plt.ylim(-0.05, 1.05)
plt.grid(True)
plt.legend()
plt.show()

# ----------------------------
# 5. 중앙 생존시간 계산 (pandas 비교 연산을 피함)
# ----------------------------
def median_survival_time(km_df):
    """
    km_df: DataFrame returned by kaplan_meier_estimator
    returns: median time (float) where survival <= 0.5; np.nan if never drops to
0.5
    NOTE: 비교는 NumPy 배열로 수행하여 pandas 내부 optional import를 피합니다.
    """
    surv = np.asarray(km_df['survival'], dtype=float)   # numpy array
    times = np.asarray(km_df['time'], dtype=float)     # numpy array
    idx = np.where(surv <= 0.5)[0]
    if idx.size == 0:
        return np.nan
    return float(times[idx[0]])

print("--- 중앙 생존 시간 ---")
print(f"전체: {median_survival_time(overall_km):.2f}")

for name, grouped_df in df.groupby('group'):
    km = kaplan_meier_estimator(grouped_df['duration'], grouped_df['event'])
    print(f"그룹 {name}: {median_survival_time(km):.2f}")
'''
전체: 28.00
그룹 A: 28.00
그룹 B: 31.00
'''
```





결과 해석 방법

- **생존 곡선**: 곡선이 가파르게 떨어질수록 사건 발생률이 높다는 것을 의미합니다. 곡선이 높은 위치에 있을수록 생존 확률이 높습니다.
- **중앙 생존 시간**: 생존 확률이 0.5(50%)가 되는 시점의 시간입니다. 이는 전체 관측치의 절반이 사건을 경험하는 데 걸리는 시간을 나타냅니다.
- **위험군 수 (At-risk counts)**: 각 시점별로 아직 사건을 경험하지 않고 추적 관찰 중인 관측치의 수를 나타냅니다. 이 수가 너무 적어지면 생존 곡선의 신뢰도가 낮아질 수 있습니다.

---

2. Log-rank test

- 두 개 이상의 그룹 간 생존 곡선에 통계적으로 유의미한 차이가 있는지를 검정하는 비모수적 가설 검정 방법

- 새로운 치료법을 받은 환자 그룹과 기존 치료법을 받은 환자 그룹의 생존율에 차이가 있는지 비교할 때 사용

주의사항

- **비례 위험 가정 (Proportional Hazards Assumption)**: Log-rank test는 그룹 간의 위험률(Hazard Rate)이 시간에 따라 일정하게 비례한다는 가정을 전제로 합니다. 이 가정이 위배될 경우 검정 결과의 신뢰도가 떨어질 수 있습니다.
- **공변량 통제 불가**: Kaplan-Meier와 마찬가지로 다른 공변량의 영향을 통제하지 않고 단순히 그룹 간의 차이만을 비교합니다.

```
import numpy as np
import pandas as pd
from scipy.stats import chi2

# ------------------------------------------------------------
# 1. 데이터 준비 (Kaplan-Meier 예시와 동일)
# ------------------------------------------------------------
data = {
    'id': range(1, 21),
    'duration': [10, 12, 15, 20, 22, 25, 28, 30, 35, 40,
                 11, 13, 16, 21, 23, 26, 29, 31, 36, 41],
    'event': [1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
              1, 0, 1, 1, 0, 1, 0, 1, 1, 0],
    'group': ['A']*10 + ['B']*10
}
df = pd.DataFrame(data)

# 그룹 A와 B로 분리
df_A = df[df['group'] == 'A']
df_B = df[df['group'] == 'B']

# ------------------------------------------------------------
# 2. Log-rank test 직접 구현
# ------------------------------------------------------------
def logrank_test_manual(durations_A, durations_B, events_A, events_B):
    """
    lifelines.statistics.logrank_test()의 간단한 수동 구현 버전
    durations_X: 생존 시간
    events_X: 사건 발생 여부 (1=발생, 0=중도절단)
    return: test_statistic, p_value
    """

    # 모든 고유 시간 (이벤트 발생한 시점만 고려)
    all_times = np.unique(np.concatenate((durations_A[events_A == 1],
                                          durations_B[events_B == 1])))

    # 누적 기대값과 분산 초기화
    O_A = E_A = V_A = 0.0

    for t in all_times:
        # 각 그룹에서 위험군과 사건 수 계산
```

```
        nA = np.sum(durations_A >= t)
        nB = np.sum(durations_B >= t)
        n = nA + nB

        dA = np.sum((durations_A == t) & (events_A == 1))
        dB = np.sum((durations_B == t) & (events_B == 1))
        d = dA + dB

        # 기대값과 분산 (Greenwood's formula 기반)
        if n > 1:
            eA = d * (nA / n)
            vA = (nA * nB * d * (n - d)) / (n**2 * (n - 1)) if (n - 1) > 0 else 0
        else:
            eA, vA = 0, 0

        O_A += dA
        E_A += eA
        V_A += vA

    # 통계량 (Chi-square)
    Z = (O_A - E_A)
    chi2_stat = Z**2 / V_A if V_A > 0 else 0
    p_value = 1 - chi2.cdf(chi2_stat, df=1)

    return chi2_stat, p_value

# ------------------------------------------------------------
# 3. 검정 수행
# ------------------------------------------------------------
chi2_stat, p_value = logrank_test_manual(
    durations_A=np.array(df_A['duration']),
    durations_B=np.array(df_B['duration']),
    events_A=np.array(df_A['event']),
    events_B=np.array(df_B['event'])
)

# ------------------------------------------------------------
# 4. 결과 출력
# ------------------------------------------------------------
print("--- Log-rank test 결과 ---")
print(f"Chi-square 통계량: {chi2_stat:.4f}")
print(f"p-value: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:
    print(f"유의수준 {alpha}에서 귀무가설(두 그룹의 생존곡선이 동일하다)을 기각합니
다.")
    print("→ 두 그룹의 생존곡선에는 통계적으로 유의미한 차이가 있습니다.")
else:
    print(f"유의수준 {alpha}에서 귀무가설을 기각할 수 없습니다.")
    print("→ 두 그룹의 생존곡선에 통계적으로 유의미한 차이가 있다고 보기 어렵습니다.")
'''
Chi-square 통계량: 0.2642
p-value: 0.6072
```

```
유의수준 0.05에서 귀무가설을 기각할 수 없습니다.
→ 두 그룹의 생존곡선에 통계적으로 유의미한 차이가 있다고 보기 어렵습니다.
'''
```

결과 해석 방법

- **p-value**: Log-rank test의 핵심 결과입니다. 일반적으로 p-value가 유의수준(예: 0.05)보다 작으면 귀무가설(두 그룹의 생존 곡선에 차이가 없다)을 기각하고, 두 그룹 간 생존 곡선에 통계적으로 유의미한 차이가 있다고 결론 내립니다.
- **test_statistic**: 검정 통계량 값입니다. 이 값이 클수록 두 그룹 간 차이가 크다는 것을 의미합니다.

장단점 및 대안

기법	장점	단점
Kaplan-Meier 생존 곡선	- 직관적인 생존 확률 시각화 - 중도 절단 데이터 처리 가능 - 비모수적 방법으로 분포 가정 불필요	- 공변량의 영향 모델링 불가 - 그룹 간 통계적 비교에 한계
Log-rank test	- 두 그룹 이상 간 생존 곡선 비교의 통계적 유의성 검정 - 비모수적 방법	- 비례 위험 가정 필요 - 공변량의 영향 통제 불가

대안

- **Cox 비례 위험 모형 (Cox Proportional Hazards Model)**: 여러 공변량(나이, 성별, 치료 방법 등)의 영향을 동시에 고려하여 생존 시간에 미치는 영향을 분석할 수 있는 회귀 모형입니다. 비례 위험 가정을 전제로 합니다.
    - `lifelines.CoxPHFitter`를 사용합니다.
- **가속 수명 모형 (Accelerated Failure Time, AFT Model)**: Cox 모형과 유사하게 공변량을 포함하지만, 위험률이 아닌 생존 시간에 직접적으로 영향을 미치는 요인을 모델링합니다. 특정 분포(예: Weibull, Exponential)를 가정합니다.
- ** parametric survival models**: 생존 시간에 대한 특정 확률 분포(예: 지수 분포, 와이블 분포)를 가정하고 모형을 구축하는 방법입니다.