math_hw_7

January 17, 2025

[1]:
```
import pandas as pd
import numpy as np
from scipy import stats
import statsmodels.api as sm
import itertools
import pandas as pd
import numpy as np
n_rows = 50000
np.random.seed(42)
ages = np.random.choice(
    ['Under 18', '18-25', '26-35', '36-50', '50+'],
    size=n_rows,
    p=[0.15, 0.35, 0.25, 0.15, 0.1]
)
#Длительность сессии
duration_means = {'Under 18': 20, '18-25': 20, '26-35': 25, '36-50':
  ↪22, '50+': 18}
duration_stds = {'Under 18': 7, '18-25': 7, '26-35': 8, '36-50': 6,
  ↪'50+': 5}
session_durations = [
    max(0, np.random.normal(duration_means[age], duration_stds[age]))
    for age in ages
]
#Количество прослушанных треков за сессию
tracks_listened = [
    max(1, int(duration / np.random.uniform(3, 5)))
    for duration in session_durations
]
#Количество уникальных исполнителей
unique_artists = [
    max(1, int(tracks / np.random.uniform(1.5, 3)))
    for tracks in tracks_listened
]

dataset = pd.DataFrame({
    'age_category': ages,
    'session_duration': session_durations,
```

```
    'tracks_listened': tracks_listened,
    'unique_artists': unique_artists
})
file_path = 'synthetic_music_sessions.csv'
dataset.to_csv(file_path, index=False)
dataset.info
```

[1]:
```
<bound method DataFrame.info of       age_category  session_duration
tracks_listened  unique_artists
0             18-25         19.866556                4                1
1               50+         24.036442                6                3
2             26-35         30.002340                7                2
3             26-35         18.562488                5                2
4             18-25         32.509794                6                2
…               …               …                …              …
49995         26-35         17.733346                5                1
49996         18-25         30.494072                8                3
49997         26-35         34.111409                9                3
49998         26-35         23.129974                6                2
49999         36-50         24.741292                6                2

[50000 rows x 4 columns]>
```

1. Гипотеза о медиане (дискретный случай)

H0 - медиана уникальных исполнителей одинаково для возвраствных групп 18-25 и 26-35 H1 - медиана уникальных исполнителей отличается для возвраствных групп 18-25 и 26-35

Проверим критерием Манна-Уитни. Используем тест Манна-Уитни для проверки различий между медианами, так как распределение данных не обязательно нормальное, а тест Манна-Уитни работает для независимых выборок и не требует нормальности. Или можно использовать t - test если данные нормальны.

[4]:
```
from scipy.stats import mannwhitneyu, ks_2samp, chi2_contingency
group_18_25 = dataset[dataset['age_category'] ==
  ↪'18-25']['unique_artists']
group_26_35 = dataset[dataset['age_category'] ==
  ↪'26-35']['unique_artists']
```

[5]:
```
stat1, pval1 = mannwhitneyu(group_18_25, group_26_35,
  ↪alternative='two-sided')
print(f"1. Гипотеза о медиане (дискретный случай): p-value = {pval1:.
  ↪4f}")
```

```
1. Гипотеза о медиане (дискретный случай): p-value = 0.0000
```

H0 отвергается

2. Гипотеза о медиане (непрерывный случай)

H0 - медиана длительности сессии для возрастной группы “36-50” равна медианной длительности сессии для группы “50+”. H1 - медиана длительности сессии для возрастной группы “36-50” отличается от медианной длительности сессии для группы “50+”.

Проверим критерием Манна-Уитни. Используем тест Манна-Уитни для проверки различий между медианами, так как распределение данных не обязательно нормальное, а тест Манна-Уитни работает для независимых выборок и не требует нормальности. Или можно использовать t - test если данные нормальны.

[7]:
```
group_36_50 = dataset[dataset['age_category'] ==
  ↪'36-50']['session_duration']
group_50_plus = dataset[dataset['age_category'] ==
  ↪'50+']['session_duration']
stat2, pval2 = mannwhitneyu(group_36_50, group_50_plus,
  ↪alternative='two-sided')
print(f"2. Гипотеза о медиане (непрерывный случай): p-value = {pval2:.
  ↪4f}")
```

```
2. Гипотеза о медиане (непрерывный случай): p-value = 0.0000
```

H0 отвергается

3. Гипотеза о распределении (дискретный случай)

H□: Распределение количества уникальных исполнителей одинаково для возрастных групп Under 18 и 18-25. H□: Распределение количества уникальных исполнителей различается для возрастных групп Under 18 и 18-25.

Проверка: Критерий хи-квадрат. Применяется для дискретных данных или категориальных распределений.

[27]:
```
group_under_18 = dataset[dataset['age_category'] == 'Under
  ↪18']['unique_artists']
group_18_25 = dataset[dataset['age_category'] ==
  ↪'18-25']['unique_artists']

chi2_stat, pval3, dof, expected  = chi2_contingency(group_under_18,
  ↪group_18_25)
print(f"3. Гипотеза о распределении (дискретный случай): p-value =
  ↪{pval3:.4f}")
```

```
3. Гипотеза о распределении (дискретный случай): p-value = 1.0000
```

H0 принимается, т.к. p-value > 0.05

4. Гипотеза о распределении (непрерывный случай):

H□: Распределение длительности сессий одинаково для возрастных групп 26-35 и 36-50. H□: Распределение длительности сессий различается для возрастных групп 26-35 и 36-50.

Проверка: Тест Колмогорова-Смирнова. Применяется для непрерывнх распределений. В случае непрерывных данных этот тест чувствителен к различиям в форме распределений.

[9]:
```
group_26_35 = dataset[dataset['age_category'] ==
  ↪'26-35']['session_duration']
group_36_50 = dataset[dataset['age_category'] ==
  ↪'36-50']['session_duration']

stat4, pval4 = ks_2samp(group_26_35, group_36_50)
print(f"4. Гипотеза о распределении (непрерывный случай): p-value =
  ↪{pval4:.4f}")
```

```
4. Гипотеза о распределении (непрерывный случай): p-value = 0.0000
```

H0 отвергается

5. Бустрап

[17]:
```
def bootstrap_p_value(data1, data2, n_iterations=1000):
    observed_diff = np.mean(data1) - np.mean(data2)
    bootstrapped_diffs = []
    for _ in range(n_iterations):
        sample1 = np.random.choice(data1, size=len(data1),
  ↪replace=True)
        sample2 = np.random.choice(data2, size=len(data2),
  ↪replace=True)
        bootstrapped_diffs.append(np.mean(sample1) - np.mean(sample2))
    bootstrapped_diffs = np.array(bootstrapped_diffs)
    p_value = np.mean(np.abs(bootstrapped_diffs) >= np.
  ↪abs(observed_diff))
    return p_value
```

[32]:
```
group1 = dataset[dataset['age_category'] == '18-25']['unique_artists']
group2 = dataset[dataset['age_category'] == '26-35']['unique_artists']

bootstrap_conf_interval = bootstrap_p_value(group1, group2)
print(f'Бутстрап для гипотезы 1 и 2: {bootstrap_conf_interval}')
```

```
Бутстрап для гипотезы 1 и 2: 0.463
```

[33]:
```
def bootstrap_ks_p_value(data1, data2, n_iterations=10000):
    observed_stat, _ = stats.ks_2samp(data1, data2)
    bootstrapped_stats = []
    for _ in range(n_iterations):
        sample1 = np.random.choice(data1, size=len(data1),
  ↪replace=True)
        sample2 = np.random.choice(data2, size=len(data2),
  ↪replace=True)
```

```
        stat, _ = stats.ks_2samp(sample1, sample2)
        bootstrapped_stats.append(stat)
    bootstrapped_stats = np.array(bootstrapped_stats)
    p_value = np.mean(bootstrapped_stats >= observed_stat)
    return p_value


group3 = dataset[dataset['age_category'] ==
  ↪'26-35']['session_duration']
group4 = dataset[dataset['age_category'] ==
  ↪'36-50']['session_duration']

bootstrap_p_value_3 = bootstrap_ks_p_value(group3, group4)
print(f'Бутстрап p-value для гипотезы 3: {bootstrap_p_value_3}')
```

```
Бутстрап p-value для гипотезы 3: 0.5815
```

[28]:
```
def bootstrap_chi2_p_value(data1, data2, n_iterations=1000):
    observed_stat, p_value, dof, expected = chi2_contingency(data1,
  ↪data2)
    bootstrapped_stats = []
    for _ in range(n_iterations):
        sample1 = np.random.choice(data1, size=len(data1),
  ↪replace=True)
        sample2 = np.random.choice(data2, size=len(data2),
  ↪replace=True)
        stat, p_value, dof, expected = chi2_contingency(sample1,
  ↪sample2)
        bootstrapped_stats.append(stat)
    bootstrapped_stats = np.array(bootstrapped_stats)
    p_value = np.mean(bootstrapped_stats >= observed_stat)
    return p_value

group5 = dataset[dataset['age_category'] == 'Under
  ↪18']['unique_artists']
group6 = dataset[dataset['age_category'] == '18-25']['unique_artists']
bootstrap_p_value_4 = bootstrap_chi2_p_value(group5, group6)
print(f'Бутстрап p-value для гипотезы 4: {bootstrap_p_value_4}')
```

```
Бутстрап p-value для гипотезы 4: 1.0
```

В 1,2 и 4 случае бустрап дал результат выше, чем первый способ. Вероятно, мои данные плохо сгенерированы и бустрап лучше решает эту проблему, и следовательно лучше находит какие-то паттерны в них. В 3 случае бустрап дал результат ниже, чем первый способ. Вероятно, есть какие-то отклонения от предполагаемого распределения, которые бустрап фиксирует лучше. Проблемы могут возникнуть из-за того, что например предположения не выполняются (например, если данные не нормально распределены), то стандартные тесты могут

давать ошибочные или менее точные результаты. Также в данных могут быть выбросы, которые классические тесты могут игнорировать, однако бустрап может учитывать эту проблему. На мой взгляд данные малы и имеют довольно сложную структуру и традиционные стат тесты могут быть менее мощными.

Я считаю, что для сгенерированных данных мощнее является бустрап, потому что он создает более стабильные данные. Потому что при увеличении итераций в бустрапе, p-value увеличивается.