Name-Ishan Kiran Joshi   Div:D15C Roll No-21 A.Y.-2024-25
DS LAB 4

Aim: Implementation of Statistical Hypothesis Test using Scipy and Sci-kit learn.

Problem Statement: Perform the following Tests:Correlation Tests:
a) Pearson’s Correlation Coefficient
b) Spearman’s Rank Correlation
c) Kendall’s Rank Correlation
d) Chi-Squared Test

a) Pearson’s Correlation Coefficient

```
from scipy.stats import pearsonr

excluded_columns = ['Unnamed: 0']
numeric_cols = [col for col in train_df.select_dtypes(include=['number']).columns if col not in excluded_columns]

# Compute Pearson correlation
print("\nPearson’s Correlation Coefficient:")
for i in range(len(numeric_cols)):
    for j in range(i + 1, len(numeric_cols)):
        col1, col2 = numeric_cols[i], numeric_cols[j]
        corr, _ = pearsonr(train_df[col1], train_df[col2])
        print(f"Pearson correlation between {col1} and {col2}: {corr:.4f}")
```

```
Pearson’s Correlation Coefficient:
Pearson correlation between id and popularity: 0.0736
Pearson correlation between id and vote_average: -0.5373
Pearson correlation between id and vote_count: 0.1149
Pearson correlation between popularity and vote_average: -0.2973
Pearson correlation between popularity and vote_count: 0.2058
Pearson correlation between vote_average and vote_count: -0.6040
```

This calculates the Pearson correlation coefficient between numeric columns in your dataset, excluding "Unnamed: 0". It filters numerical columns and iterates over each pair using pearsonr from scipy.stats. Pearson correlation formula:

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2}\sqrt{\sum(Y_i - \bar{Y})^2}}$$

It measures the linear relationship between two variables, ranging from -1 (strong negative) to 1 (strong positive). The output suggests id has little correlation, while popularity and vote_count show a positive correlation, meaning more votes generally indicate higher popularity. However, vote_average and vote_count have a negative correlation, implying that a higher number of votes does not always result in a higher average rating.

b) Spearman's Rank Correlation

```
from scipy.stats import spearmanr

# Compute Spearman correlation
print("\nSpearman's Rank Correlation:")
for i in range(len(numeric_cols)):
    for j in range(i + 1, len(numeric_cols)):
        col1, col2 = numeric_cols[i], numeric_cols[j]
        corr, _ = spearmanr(train_df[col1], train_df[col2])
        print(f"Spearman correlation between {col1} and {col2}: {corr:.4f}")
```

```
Spearman's Rank Correlation:
Spearman correlation between id and popularity: -0.1665
Spearman correlation between id and vote_average: -0.3710
Spearman correlation between id and vote_count: 0.0849
Spearman correlation between popularity and vote_average: -0.5039
Spearman correlation between popularity and vote_count: 0.6677
Spearman correlation between vote_average and vote_count: -0.8222
```

This calculates Spearman's rank correlation coefficient between numeric columns, measuring the monotonic relationship between variables. Unlike Pearson, it considers ranks rather than absolute values, making it robust against outliers. The formula is:

$$r_s = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where di is the difference between ranks and n is the number of observations. The output shows popularity and vote_count have a strong positive correlation, meaning as one increases, the other generally does too. However, vote_average and vote_count have a strong negative correlation, implying movies with more votes tend to have lower ratings. Popularity and vote_average are also negatively correlated, suggesting that more popular movies don't always have higher ratings.

c) Kendall's Rank Correlation

```python
from scipy.stats import kendalltau

# Compute Kendall correlation
print("\nKendall’s Rank Correlation:")
for i in range(len(numeric_cols)):
    for j in range(i + 1, len(numeric_cols)):
        col1, col2 = numeric_cols[i], numeric_cols[j]
        corr, _ = kendalltau(train_df[col1], train_df[col2])
        print(f"Kendall correlation between {col1} and {col2}: {corr:.4f}")
```

```
Kendall’s Rank Correlation:
Kendall correlation between id and popularity: -0.0943
Kendall correlation between id and vote_average: -0.2485
Kendall correlation between id and vote_count: -0.0014
Kendall correlation between popularity and vote_average: -0.3038
Kendall correlation between popularity and vote_count: 0.4383
Kendall correlation between vote_average and vote_count: -0.6656
```

This calculates Kendall’s rank correlation coefficient, measuring the ordinal association between two variables based on concordant and discordant pairs. It is more robust for small datasets and useful for ordinal data. The formula is:

$$\tau = \frac{(C - D)}{\frac{1}{2}n(n-1)}$$

where C and D are the counts of concordant and discordant pairs, respectively. The output shows popularity and vote_count have a moderate positive correlation, meaning as one increases, the other tends to as well. However, vote_average and vote_count have a strong negative correlation, suggesting movies with higher votes tend to have lower ratings. Popularity and vote_average are also negatively correlated, indicating that popular movies don’t always have higher ratings.

d) Chi-Squared Test

```python
from scipy.stats import chi2_contingency
import pandas as pd

categorical_columns = train_df.select_dtypes(include=['object']).columns

print("\nChi-Squared Test for Categorical Variables:")
for i in range(len(categorical_columns)):
    for j in range(i + 1, len(categorical_columns)):  # Avoid duplicate comparisons
        col1, col2 = categorical_columns[i], categorical_columns[j]
        contingency_table = pd.crosstab(train_df[col1], train_df[col2])
        chi2, p, _, _ = chi2_contingency(contingency_table)
        print(f"Chi-Squared test between {col1} and {col2}: Chi2 = {chi2:.4f}, p-value = {p:.4f}")
```

```
Chi-Squared Test for Categorical Variables:
Chi-Squared test between original_title and original_language: Chi2 = 12072.0000, p-value = 0.0000
Chi-Squared test between original_title and release_date: Chi2 = 54324.0000, p-value = 0.0000
Chi-Squared test between original_title and media_type: Chi2 = 0.0000, p-value = 1.0000
Chi-Squared test between original_language and release_date: Chi2 = 12072.0000, p-value = 0.0000
Chi-Squared test between original_language and media_type: Chi2 = 4888.1157, p-value = 0.0000
Chi-Squared test between release_date and media_type: Chi2 = 0.0000, p-value = 1.0000
```

The code performs a Chi-Squared test for independence between categorical variables to check if they are significantly associated. It compares observed and expected frequencies under the assumption of independence. The formula is:

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

where O is the observed frequency and E is the expected frequency. A low p-value (< 0.05) suggests a significant relationship, while a high p-value (≥ 0.05) indicates no association. The results show strong associations between original_title and original_language and release_date ($p = 0.0000$), meaning they are dependent. However, original_title and media_type, as well as release_date and media_type ($p = 1.0000$), indicate no relationship. This analysis helps in feature selection for machine learning.

Conclusion-The correlation analysis using four different techniques—Pearson, Spearman, Kendall, and Chi-Square—provides valuable insights into relationships between numerical and categorical variables. Pearson correlation measures linear relationships, showing how one variable changes proportionally with another. Spearman and Kendall correlations capture monotonic relationships, making them more robust for non-linear associations. The Chi-Square test evaluates categorical dependencies, determining whether two categorical variables are related. While Pearson is effective for continuous data with normal distribution, Spearman and Kendall are preferable for ordinal or non-linear data. The Chi-Square test helps identify categorical variable dependencies, guiding feature selection in machine learning models. Together, these techniques provide a comprehensive understanding of data relationships, ensuring better preprocessing and model accuracy.