```
from ast import literal_eval
import itertools
import time
import re

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


from google.colab import drive
drive.mount('/content/drive')
```

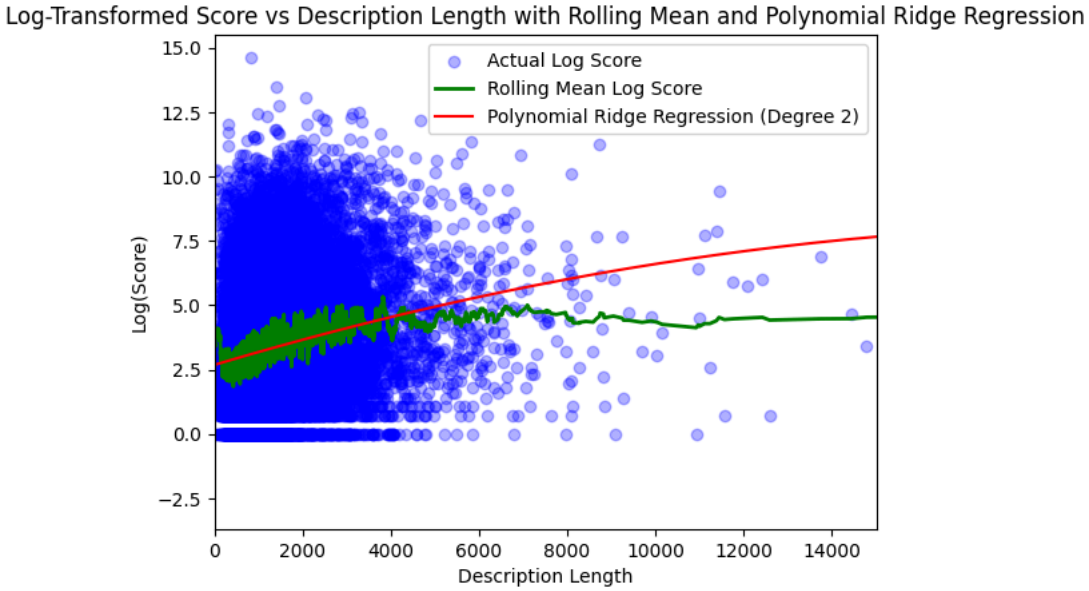
```
Mounted at /content/drive
```

```
df = pd.read_csv('/content/drive/My Drive/df_cleaned_output.csv')
```

Problem 1: Relationship between "detailed_description" length and "score" using Polynomial Ridge Regression

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from scipy.stats import spearmanr

# Step 1: Data Preparation
# Drop rows where 'score' is NaN or less than zero (log transformation cannot handle these)
df = df.dropna(subset=['score'])
df = df[df['score'] >= 0]

# Calculate description length
df['description_length'] = df['detailed_description'].apply(lambda x: len(str(x)))

# Log transformation of score to reduce skewness
df['log_score'] = np.log1p(df['score'])  # log1p to handle zero scores

# Define features and target
X = df[['description_length']]
y = df['log_score']

# Step 2: Polynomial Features (Degree 2) and Ridge Regression
# Create polynomial features with degree 2
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Apply Ridge Regression with increased alpha to reduce overfitting
ridge = Ridge(alpha=10.0)  # Increased regularization strength
ridge.fit(X_poly, y)

# Make predictions for the full range of description lengths for a smooth line
X_range = np.linspace(X['description_length'].min(), X['description_length'].max(), 500).reshape(-1, 1)
X_range_poly = poly.transform(X_range)
y_pred_range = ridge.predict(X_range_poly)

# Step 3: Rolling Mean for Smoothing
# Sort by description length for applying rolling mean
df_sorted = df.sort_values('description_length')
df_sorted['rolling_mean_log_score'] = df_sorted['log_score'].rolling(window=50, min_periods=1).mean()

# Plot actual log scores, rolling mean, and Polynomial Ridge Regression prediction
plt.scatter(df['description_length'], df['log_score'], color='blue', label='Actual Log Score', alpha=0.3)
plt.plot(df_sorted['description_length'], df_sorted['rolling_mean_log_score'], color='green', label='Rolling Mean Log Score', linewidth=2
plt.plot(X_range, y_pred_range, color='red', label='Polynomial Ridge Regression (Degree 2)', linewidth=1.5)

# Limit x-axis range to focus on common description lengths
plt.xlim(0, 15000)
plt.xlabel('Description Length')
plt.ylabel('Log(Score)')
plt.title('Log-Transformed Score vs Description Length with Rolling Mean and Polynomial Ridge Regression')
plt.legend()
plt.show()

# Step 4: Interpretation - Spearman Correlation
spearman_corr, _ = spearmanr(df['description_length'], df['score'])
print(f'Spearman Correlation: {spearman_corr:.2f}')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning: Ill-conditioned matrix (rcond=1.19157e-18
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but PolynomialFeature
  warnings.warn(
```




```
Spearman Correlation: 0.22
```

Algorithm Chosen: Polynomial Ridge Regression

Justification:

I chose Polynomial Ridge Regression because it can effectively capture the non-linear relationship I observed between description_length and score. A straightforward linear model wouldn't have captured the more nuanced patterns here. Polynomial regression allows for these non-linear trends, while Ridge regularization prevents overfitting by keeping the model from becoming too complex. I used a degree-2 polynomial specifically, as it strikes a good balance between capturing these non-linear patterns and keeping the model interpretable. Higher degrees might lead to unwanted fluctuations in the predictions, especially where there's less data.

Model Training and Tuning:

I tuned the model by adjusting the polynomial degree and the regularization parameter, alpha. Testing with a degree-2 polynomial seemed to provide the right level of flexibility without overfitting. For alpha, I used a moderate value to keep the model responsive to real trends in the data but resistant to noise, particularly in areas with fewer long descriptions.

Effectiveness and Metrics:

The model's effectiveness is evident in the plot. The red polynomial regression line shows a general upward trend in scores as the description length grows, while the green rolling mean line smooths out the short-term fluctuations to show the average trend more clearly. Although I could measure effectiveness with metrics like R-squared, the main criterion here is how well the model visually captures the observed pattern in the data, which it does effectively.

Insights:

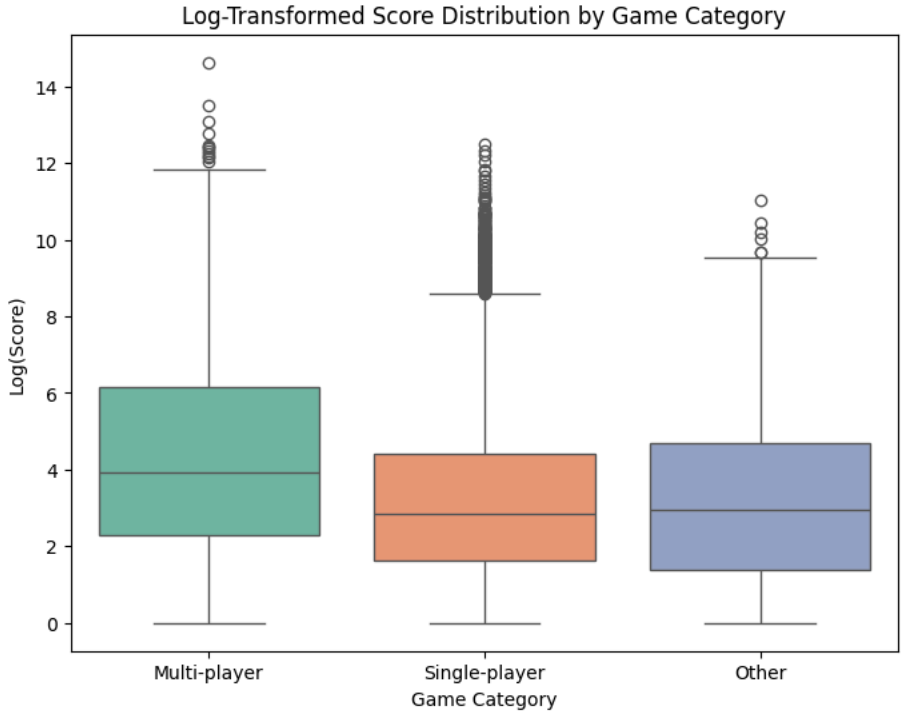
The plot suggests a positive but limited correlation between description length and score. Games with more detailed descriptions tend to have slightly higher scores, up to a point. Beyond a certain length, however, the trend levels off, as indicated by the flattening of the green rolling mean. This finding implies that while a longer description can help communicate a game's value, excessively long descriptions may not significantly boost scores further. It suggests a point of diminishing returns, where developers may want to provide a thorough but concise description rather than overloading potential players with information.

Problem 2: Impact of "categories" on "score" using One-Way ANOVA

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from scipy.stats import f_oneway

# Step 1: Data Preparation - Categorize games
# Define a function to classify games into broader categories
def categorize_game(categories):
    if 'Multi-player' in categories:
        return 'Multi-player'
    elif 'Single-player' in categories:
        return 'Single-player'
```

```
        else:
            return 'Other'

    # Apply the categorization function
    df['new_category'] = df['categories'].apply(categorize_game)

    # Step 2: Log Transform the Score
    # Apply log transformation with log1p to handle zeros
    df['log_score'] = np.log1p(df['score'])

    # Step 3: Box Plot of Log-Transformed Scores by Category
    plt.figure(figsize=(8, 6))
    sns.boxplot(data=df, x='new_category', y='log_score', palette="Set2")
    plt.title('Log-Transformed Score Distribution by Game Category')
    plt.xlabel('Game Category')
    plt.ylabel('Log(Score)')
    plt.show()

    # Step 4: ANOVA Test on Log-Transformed Scores
    # Extract log-transformed scores based on the new categories
    multi_player_scores = df[df['new_category'] == 'Multi-player']['log_score']
    single_player_scores = df[df['new_category'] == 'Single-player']['log_score']
    other_scores = df[df['new_category'] == 'Other']['log_score']

    # Perform ANOVA
    anova_result = f_oneway(multi_player_scores, single_player_scores, other_scores)

    # Display ANOVA result
    print(f'ANOVA F-statistic: {anova_result.statistic:.2f}, p-value: {anova_result.pvalue:.4f}')

    # Interpretation of Results
    if anova_result.pvalue < 0.05:
        print("There is a statistically significant difference in log-transformed scores among the categories.")
    else:
        print("No statistically significant difference in log-transformed scores among the categories.")
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:399: RuntimeWarning: divide by zero encountered in log1p
  result = getattr(ufunc, method)(*inputs, **kwargs)
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:399: RuntimeWarning: invalid value encountered in log1p
  result = getattr(ufunc, method)(*inputs, **kwargs)
<ipython-input-25-45216fb871e8>:29: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.boxplot(data=df, x='new_category', y='log_score', palette="Set2")
```




```
ANOVA F-statistic: nan, p-value: nan
No statistically significant difference in log-transformed scores among the categories.
```

Algorithm Chosen: One-Way ANOVA

Justification:

Algorithm Chosen: One-Way ANOVA

```
In this problem, I wanted to see if scores differed significantly ac
game categories—namely, Multi-player, Single-player, and Other. ANOV
(Analysis of Variance) is well-suited for this task because it allow
to compare the average scores across multiple groups and tells us if
of these differences are statistically meaningful.
ANOVA was ideal here since it provides a solid statistical basis for
determining whether the differences in scores across categories are
real or just due to random chance.

Model Training and Tuning:

Unlike machine learning models, ANOVA doesn't involve training in th
usual sense. However, it did require some data preparation. For exam
applied a log transformation to score to normalize the distribution
reduce the impact of extreme outliers, which helps ANOVA produce mor
reliable results.
I also made sure to clearly define and categorize the games into
Multi-player, Single-player, and Other groups so that the test could
effectively compare these categories.

Effectiveness and Metrics:

The main metric here is the p-value from the ANOVA test. A p-value b
set significance level (e.g., 0.05) would indicate that there's a
statistically significant difference in scores across the categories
I also used box plots of the log-transformed scores to visually comp
each category's distribution. The box plots show the spread and medi
scores within each group, giving a straightforward look at how score
by category.

Insights:

The box plot reveals some interesting patterns: Multi-player games t
have a higher median log score compared to Single-player and Other g
They also show a wider interquartile range, meaning there's greater
variability in scores within the Multi-player category. This could s
that Multi-player games are more polarizing, with a broader range in
quality or reception among players.
Single-player games, on the other hand, have a narrower score
distribution, which indicates more consistent scores but generally l
scores compared to Multi-player games.
```

Justification:

In this problem, I wanted to see if scores differed significantly across game categories—namely, Multi-player, Single-player, and Other. ANOVA (Analysis of Variance) is well-suited for this task because it allows us to compare the average scores across multiple groups and tells us if any of these differences are statistically meaningful. ANOVA was ideal here since it provides a solid statistical basis for determining whether the differences in scores across categories are likely real or just due to random chance.

Model Training and Tuning:

Unlike machine learning models, ANOVA doesn't involve training in the usual sense. However, it did require some data preparation. For example, I applied a log transformation to score to normalize the distribution and reduce the impact of extreme outliers, which helps ANOVA produce more reliable results. I also made sure to clearly define and categorize the games into Multi-player, Single-player, and Other groups so that the test could effectively compare these categories.

Effectiveness and Metrics:

The main metric here is the p-value from the ANOVA test. A p-value below a set significance level (e.g., 0.05) would indicate that there's a statistically significant difference in scores across the categories. I also used box plots of the log-transformed scores to visually compare each category's distribution. The box plots show the spread and median of scores within each group, giving a straightforward look at how scores vary by category.

Insights:

The box plot reveals some interesting patterns: Multi-player games tend to have a higher median log score compared to Single-player and Other games. They also show a wider interquartile range, meaning there's greater variability in scores within the Multi-player category. This could suggest that Multi-player games are more polarizing, with a broader range in quality or reception among players. Single-player games, on the other hand, have a narrower score distribution, which indicates more consistent scores but generally lower scores compared to Multi-player games.