In [1]:

```
import pandas as pd
from scipy import stats
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Load the data from the Excel file
df = pd.read_excel('animelist2clean.xlsx')

# Filter for high scores (e.g., scores greater than 8)
high_score_df = df[df['score'] > 8]

# Sort by popularity (ascending)
hidden_gems_df = high_score_df.sort_values(by='popularity')

# Define the null and alternative hypotheses
# H0: The average score of high-scoring anime is equal to 8.5
# H1: The average score of high-scoring anime is not equal to 8.5

# Perform one-sample t-test
t_stat, p_value = stats.ttest_1samp(hidden_gems_df['score'], 8.5)

# Print the test statistic and p-value
print(f'Test Statistic: {t_stat}')
print(f'P-Value: {p_value}')

# Determine if we reject or fail to reject the null hypothesis
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis: The average score of high-scoring anime is significantly different from 8.5.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference between the average score of high-scoring anime ar

# Predict if hidden gems will become popular in the future
# For simplicity, we will use a linear regression model to predict future popularity based on current scores

# Prepare the data for linear regression
X = hidden_gems_df[['score']]
y = hidden_gems_df['popularity']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict future popularity
```

```
y_pred = model.predict(X_test)

# Calculate the mean squared error (MSE) to evaluate the model's performance
mse = np.mean((y_test - y_pred) ** 2)
print(f'Mean Squared Error: {mse}')

# Display the predicted vs actual popularity for the test set
results_df = pd.DataFrame({'Actual Popularity': y_test, 'Predicted Popularity': y_pred})
print(results_df.head(10))
```

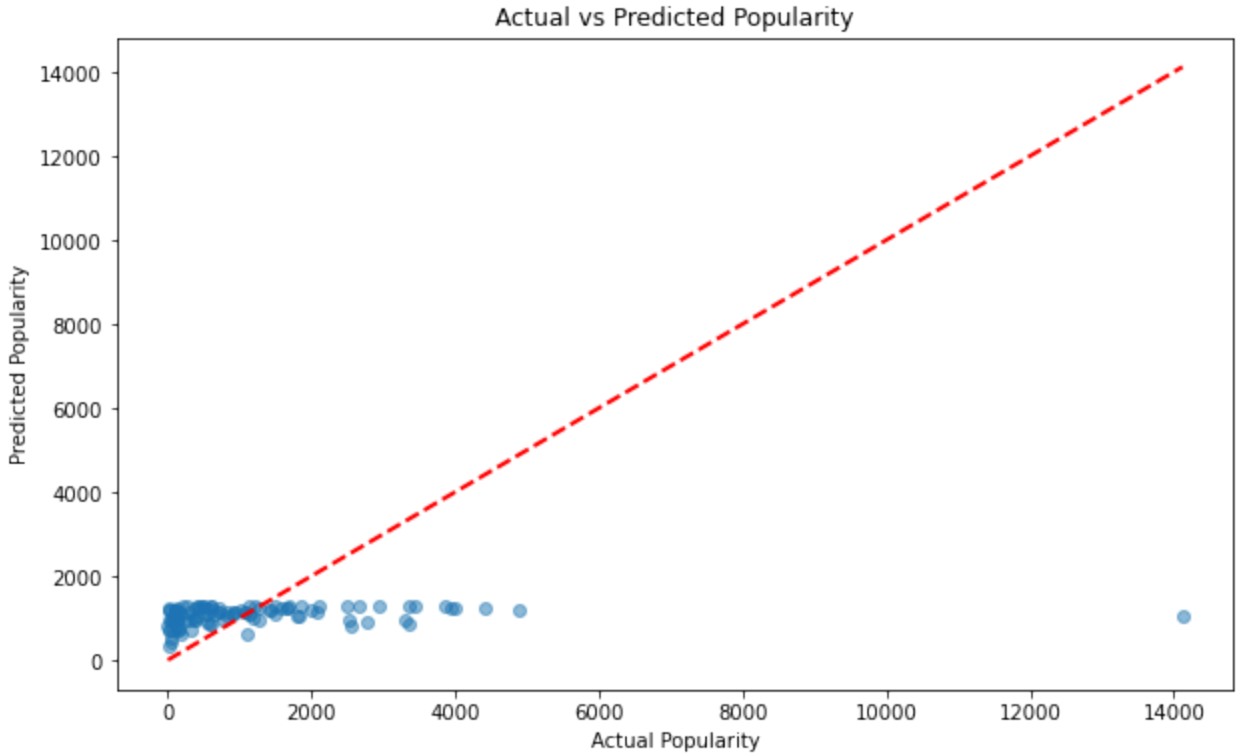
```
Test Statistic: -16.380498391622492
P-Value: 3.3056914615334438e-49
Reject the null hypothesis: The average score of high-scoring anime is significantly different from 8.5.
Mean Squared Error: 2729953.273992895
      Actual Popularity  Predicted Popularity
859                2663           1287.535666
16                  111           1210.073765
2573               1082           1155.850434
2603                138           1070.642342
6124               2075           1155.850434
4615                115            737.556165
3112               3364            846.002827
2921               1257           1248.804715
3832                146           1155.850434
1717               1691           1295.281856
```

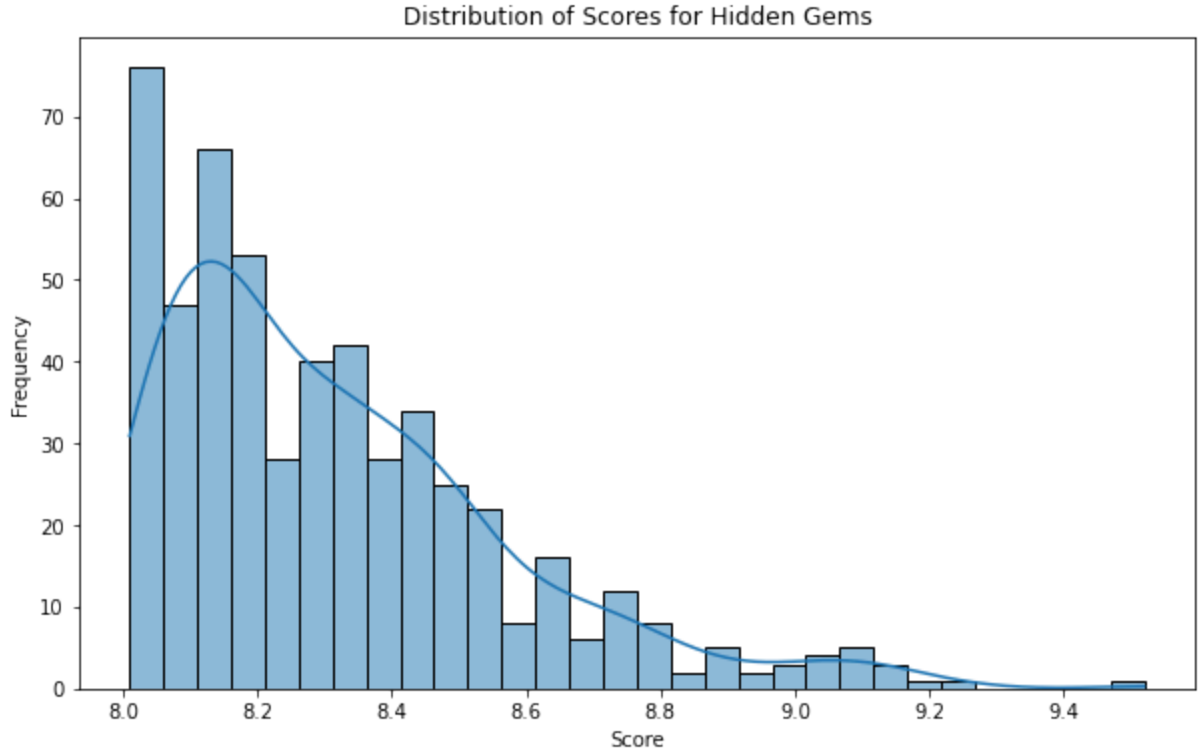
In [5]:

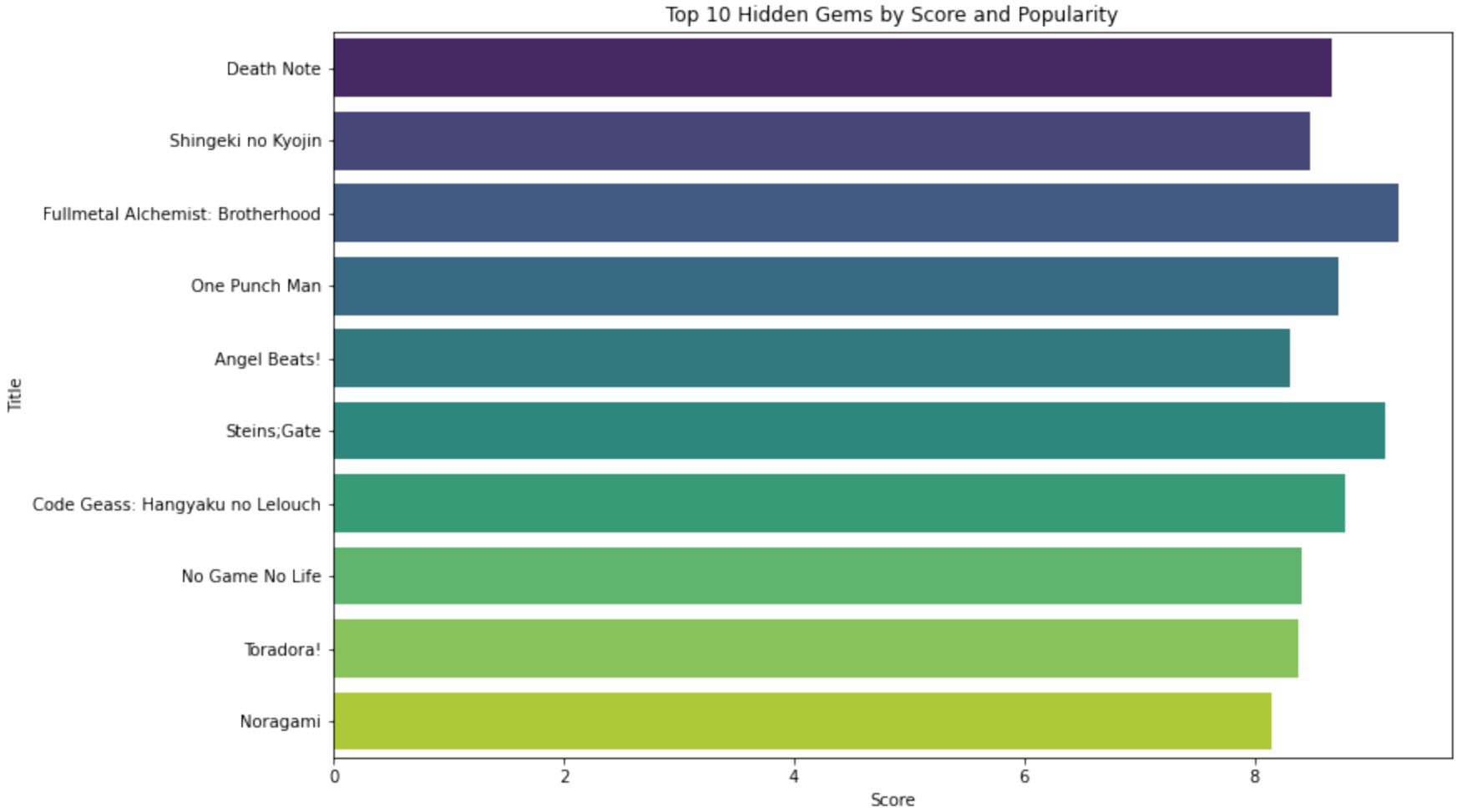
```
#pip install pandas
#pip install matplotlib
#pip install seaborn

import pandas as pd
from scipy import stats
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_excel('animelist2clean.xlsx')

high_score_df = df[df['score'] > 8]
hidden_gems_df = high_score_df.sort_values(by='popularity')

t_stat, p_value = stats.ttest_1samp(hidden_gems_df['score'], 8.5)
print(f'Test Statistic: {t_stat}')
print(f'P-Value: {p_value}')

X = hidden_gems_df[['score']]
y = hidden_gems_df['popularity']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = np.mean((y_test - y_pred) ** 2)
print(f'Mean Squared Error: {mse}')
results_df = pd.DataFrame({'Actual Popularity': y_test, 'Predicted Popularity': y_pred})
print(results_df.head(10))

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Popularity')
plt.ylabel('Predicted Popularity')
plt.title('Actual vs Predicted Popularity')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(hidden_gems_df['score'], kde=True, bins=30)
plt.xlabel('Score')
plt.ylabel('Frequency')
plt.title('Distribution of Scores for Hidden Gems')
plt.show()

top_10_hidden_gems = hidden_gems_df.head(10)
plt.figure(figsize=(12, 8))
sns.barplot(x='score', y='title', data=top_10_hidden_gems, palette='viridis')
plt.xlabel('Score')
plt.ylabel('Title')
plt.title('Top 10 Hidden Gems by Score and Popularity')
plt.show()
```

```
Test Statistic: -16.380498391622492
P-Value: 3.3056914615334438e-49
Mean Squared Error: 2729953.273992895
      Actual Popularity  Predicted Popularity
859                2663           1287.535666
16                  111           1210.073765
2573               1082           1155.850434
2603                138           1070.642342
6124               2075           1155.850434
4615                115            737.556165
3112               3364            846.002827
2921               1257           1248.804715
3832                146           1155.850434
1717               1691           1295.281856
```


Actual vs Predicted Popularity
Predicted Popularity
Actual Popularity
0
2000
4000
6000
8000
10000
12000
14000


Distribution of Scores for Hidden Gems
Frequency
Score
0
10
20
30
40
50
60
70
8.0
8.2
8.4
8.6
8.8
9.0
9.2
9.4


Top 10 Hidden Gems by Score and Popularity
Death Note
Shingeki no Kyojin
Fullmetal Alchemist: Brotherhood
One Punch Man
Angel Beats!
Steins;Gate
Code Geass: Hangyaku no Lelouch
No Game No Life
Toradora!
Noragami
Title
0
2
4
6
8
Score

In [ ]:

In [ ]: