~\OneDrive\Documents\regression.py

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Data Loading
data = pd.read_csv('books_data.csv')

# Data Printing
print("Dataset Head:")
print(data.head())

# Data Preprocessing
# Separate features and target
X = data.drop('Price', axis=1)  # Features (e.g., Title)
y = data['Price']  # Target (Price)

# Remove the currency symbol and convert Price to float
y = y.str.replace('£', '').astype(float)

# Encoding categorical data (Title)
preprocessor = ColumnTransformer(
    transformers=[
        ('title_encoder', OneHotEncoder(sparse_output=True), ['Title'])  # One-hot encode the Title column
    ]
)
X = preprocessor.fit_transform(X)

# Splitting Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature Scaling
scaler = StandardScaler(with_mean=False)  # Set with_mean=False for sparse matrices
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Model Initialization
model1 = LinearRegression()
model2 = LinearRegression()

# Model Training
model1.fit(X_train, y_train)
```

```
model2.fit(X_train, y_train)

# Model Evaluation
y_pred1 = model1.predict(X_test)
y_pred2 = model2.predict(X_test)

rmse1 = np.sqrt(mean_squared_error(y_test, y_pred1))
rmse2 = np.sqrt(mean_squared_error(y_test, y_pred2))
mae1 = mean_absolute_error(y_test, y_pred1)
mae2 = mean_absolute_error(y_test, y_pred2)
r2_1 = r2_score(y_test, y_pred1)
r2_2 = r2_score(y_test, y_pred2)

print("\nModel 1 Metrics:")
print("RMSE:", rmse1, "MAE:", mae1, "R^2:", r2_1)
print("\nModel 2 Metrics:")
print("RMSE:", rmse2, "MAE:", mae2, "R^2:", r2_2)

# Model Comparison
if r2_1 > r2_2:
    print("\nModel 1 performs better.")
else:
    print("\nModel 2 performs better.")

# Visualization: Actual vs Predicted
plt.figure(figsize=(12, 6))
plt.scatter(y_test, y_pred1, alpha=0.7, label='Model 1 Predictions', color='blue')
plt.scatter(y_test, y_pred2, alpha=0.7, label='Model 2 Predictions', color='red')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.title('Actual vs Predicted')
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.legend()
plt.show()

# Visualization: Residuals
plt.figure(figsize=(12, 6))
sns.histplot(y_test - y_pred1, kde=True, color='blue', label='Model 1 Residuals', bins=30)
sns.histplot(y_test - y_pred2, kde=True, color='red', label='Model 2 Residuals', bins=30)
plt.title('Residual Distribution')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```