CSE474/524 HW2
Siqi Cheng, scheng42
Mukesh Thamilvanan, mthamilv

Assignment 2: Regression Methods

Part I: Data Analysis

Wine Quality Dataset

1. Dataset Overview

The Wine Quality dataset includes various physicochemical properties of wines and their quality ratings. It comprises samples of red and white wines. The dataset contains 1599 entries and 12 variables.

2. Main Statistics

The dataset's main statistics:

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

Mean:
fixed acidity: 8.319637
volatile acidity: 0.527981
citric acid: 0.270976
residual sugar: 2.538806
chlorides: 0.087467
free sulfur dioxide: 15.874922
total sulfur dioxide: 46.467792
density: 0.996747
pH: 3.311113
sulphates: 0.658149
alcohol: 10.422983
quality: 5.636023

	fixed…	volati…	citri…	residu…	chlor…	free …	total…	densi…	pH	sulph…	alcoh…	quali…
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000

Std:
fixed acidity: 1.741096
volatile acidity: 0.179060
citric acid: 0.194801
residual sugar: 1.409928
chlorides: 0.047065
free sulfur dioxide: 10.460157
total sulfur dioxide: 32.895324
density: 0.001887
pH: 0.154386
sulphates: 0.169507
alcohol: 1.065668
quality: 0.807569

	fixed…	volat…	citri…	resi…	chlo…	free…	tota…	dens…	pH	sulph…	alco…	quali…
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569

Missing Values: There are no missing values in this dataset.

```
Missing values in Wine Quality dataset:
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

Minimum Values:
fixed acidity: 4.60000
volatile acidity: 0.12000
citric acid: 0.00000
residual sugar: 0.90000
chlorides: 0.01200
free sulfur dioxide: 1.00000
total sulfur dioxide: 6.00000
density: 0.99007
pH: 2.74000
sulphates: 0.33000
alcohol: 8.40000
quality: 3.00000

	fixed…	volat…	citri…	resi…	chlo…	free…	tota…	dens…	pH	sulph…	alco…	quali…
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000

Maximum Values:
fixed acidity: 15.90000
volatile acidity: 1.58000
citric acid: 1.00000
residual sugar: 15.50000
chlorides: 0.61100
free sulfur dioxide: 72.00000
total sulfur dioxide: 289.00000
density: 1.00369
pH: 4.01000
sulphates: 2.00000
alcohol: 14.90000
quality: 8.00000

max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

Median Values:
fixed acidity: 7.900

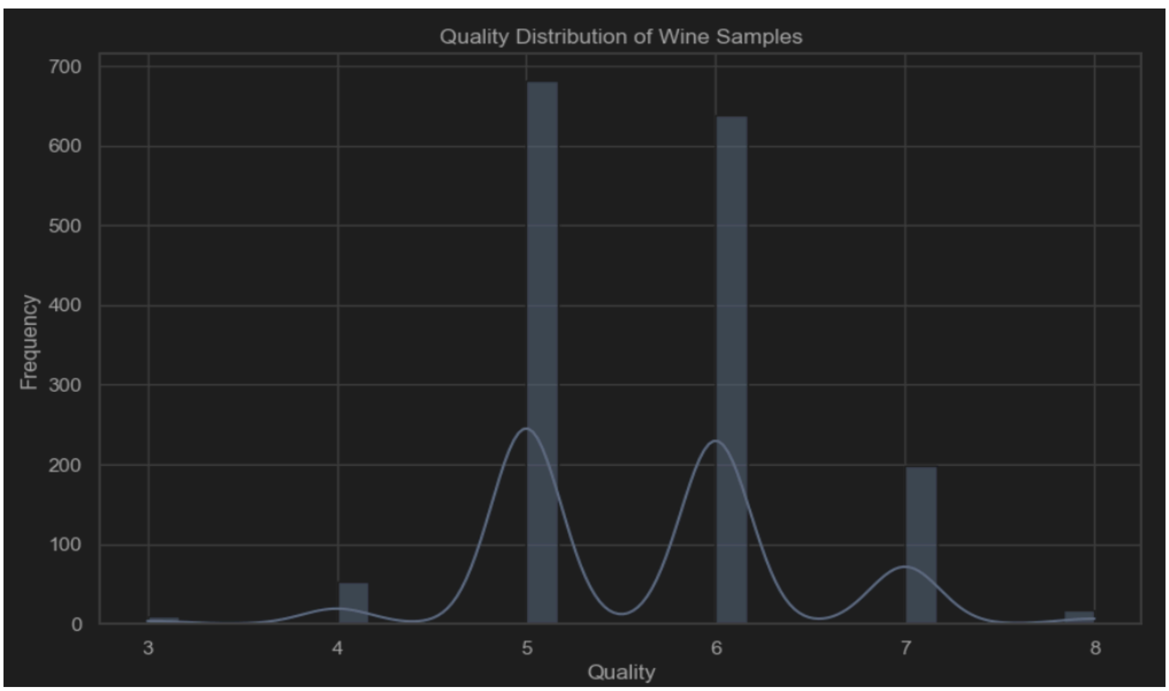
volatile acidity: 0.520
citric acid: 0.260
residual sugar: 2.200
chlorides: 0.079
free sulfur dioxide: 14.000
total sulfur dioxide: 38.000
density: 0.997
pH: 3.310
sulphates: 0.620
alcohol: 10.200
quality: 6.000

50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000

3. Data Visualizations

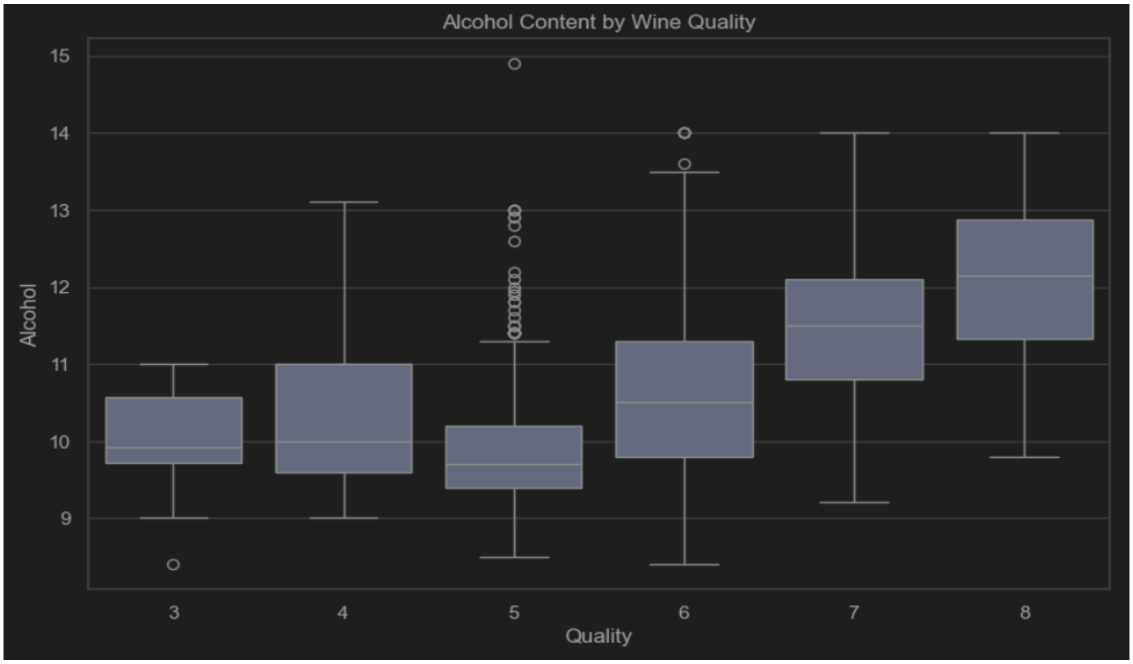
Quality Distribution

This histogram displays the distribution of wine quality ratings. It shows that the majority of wines are rated between 5 and 7.



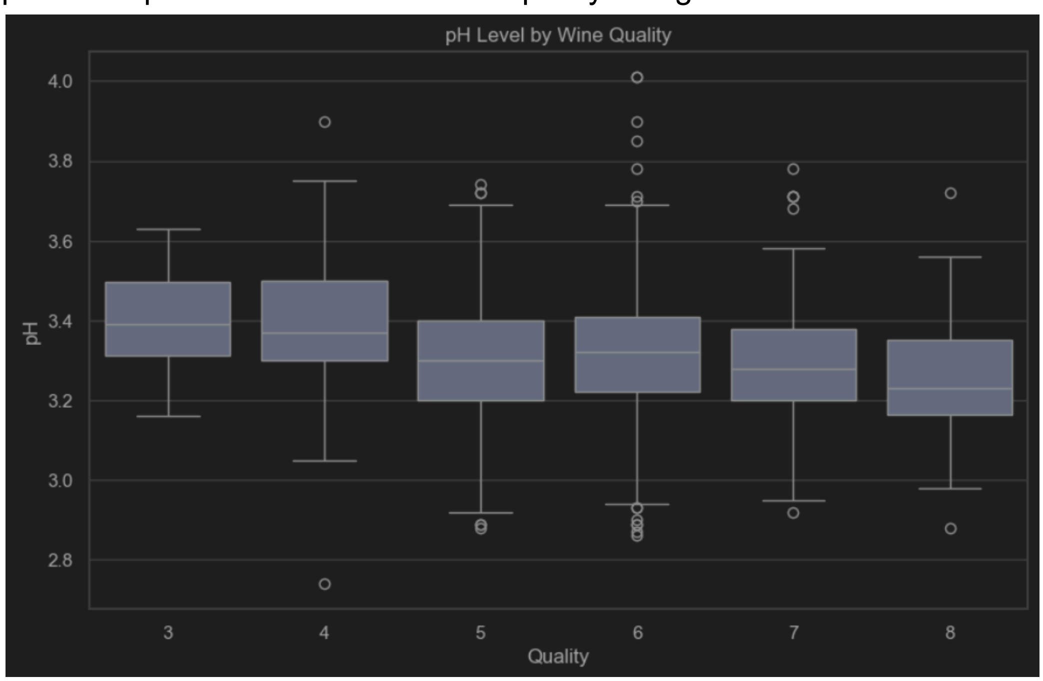
Alcohol vs Quality

This boxplot depicts the alcohol content across different wine quality ratings. Higher quality wines generally have higher alcohol content.

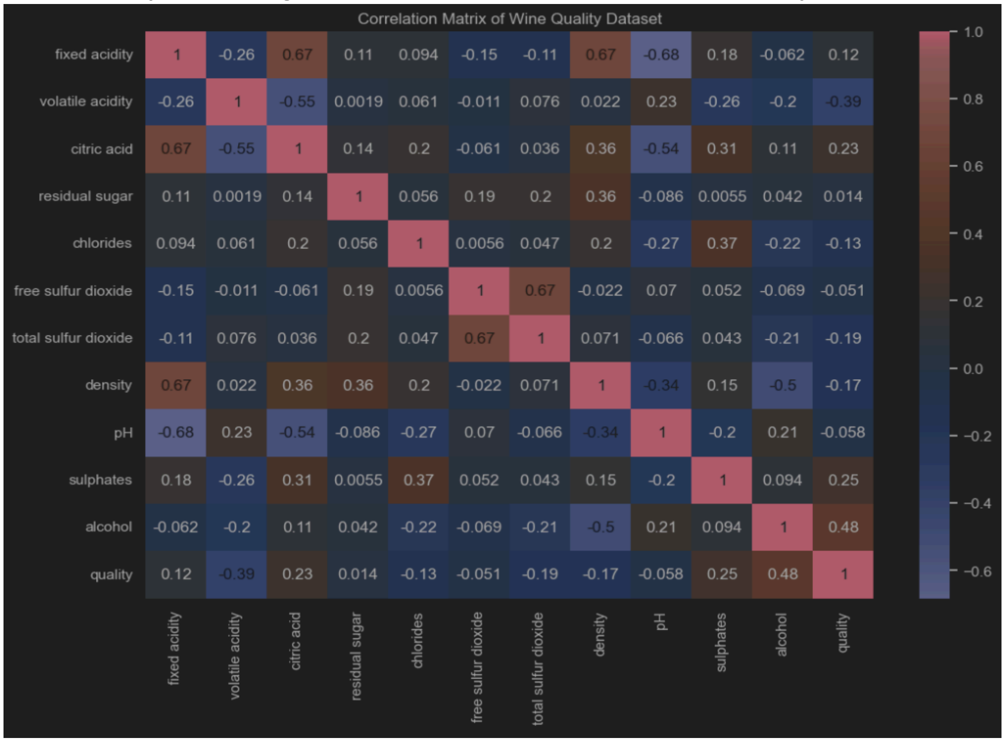


pH vs Quality

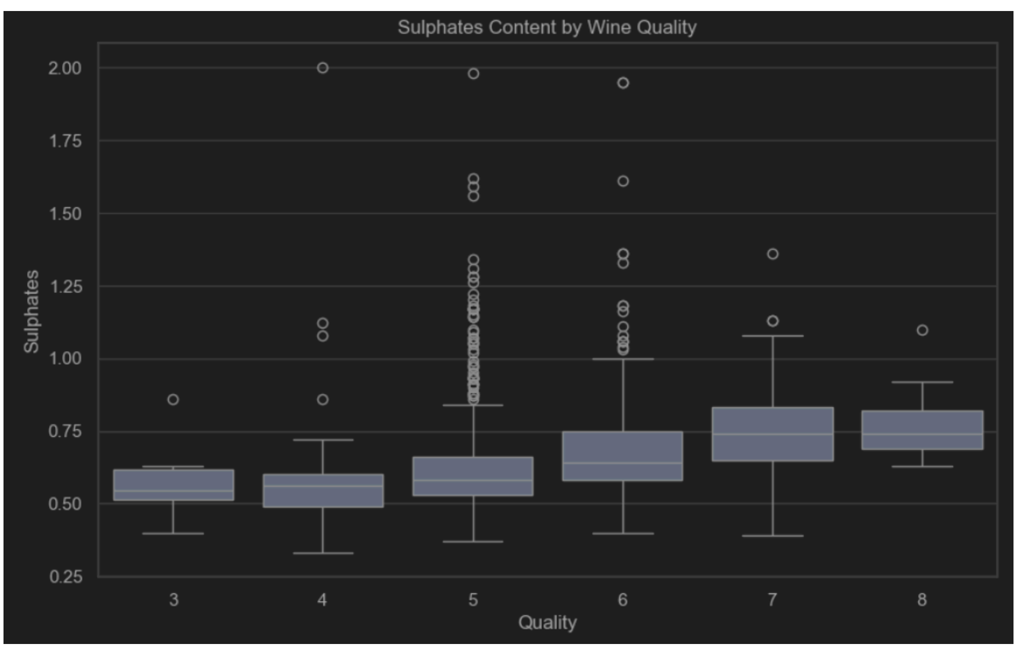
This boxplot shows the pH levels for wines of different quality ratings. There is no clear pattern in pH levels across different quality ratings.



Correlation Matrix
The heatmap represents the correlation matrix of the wine dataset. Features like alcohol and density show significant correlations with wine quality.



Sulphates vs Quality
This boxplot illustrates the sulphate content across different wine quality ratings. Wines with higher quality tend to have slightly higher sulphate content.



Netflix Titles Dataset

1. Dataset Overview

The Netflix Dataset includes information about their hosting titles in Netflix. There is a wide variety of collections with various features about every title. The dataset contains 8807 entries with 12 features.

2. Main Statistics

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

Since there is only ‘release_year’ as a numerical value, we could make a descriptive analysis.

Descriptive Statistics:

	release_year
count	8807.000000
mean	2014.180198
std	8.819312
min	1925.000000
25%	2013.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

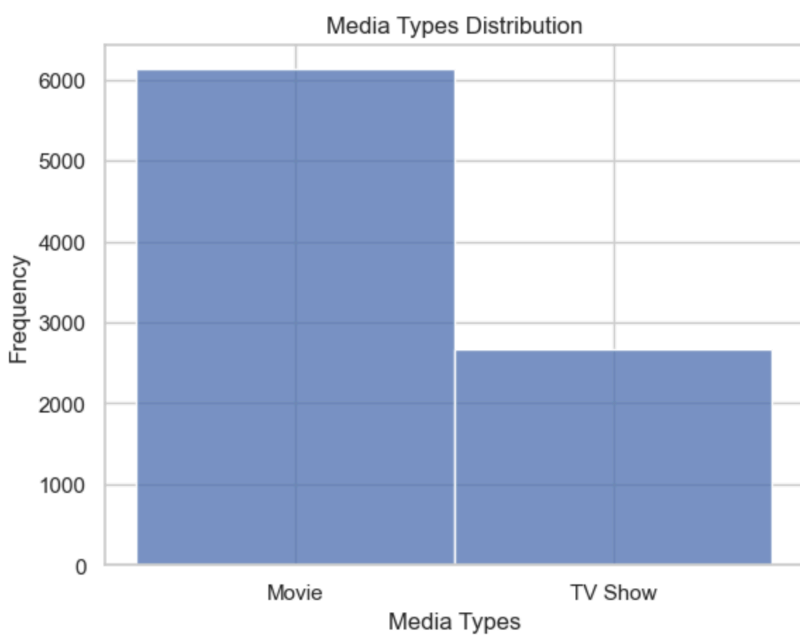
Missing Values: There is quite a bit of missing data only for certain features.

```
Missing values count in each column:

show_id            0
type               0
title              0
director        2634
cast             825
country          831
date_added        10
release_year       0
rating             4
duration           3
listed_in          0
description        0
dtype: int64
```

3. Data Visualizations

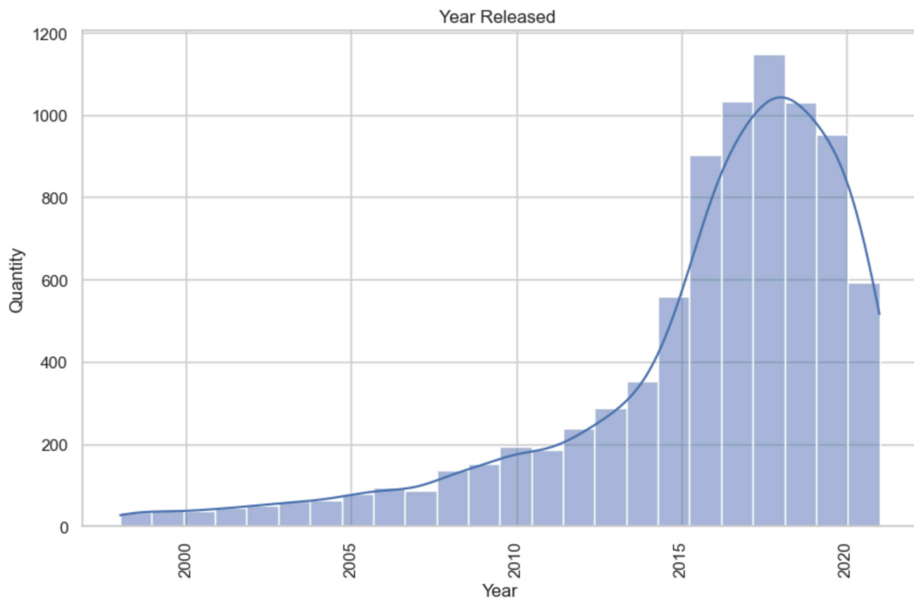
Media Type Distribution

- Using a bar graph to show the divide between the number of ‘TV show’ and ‘Movie’ in the dataset. There is clearly the significant difference between the number of movies vs tv shows collection in Netflix.



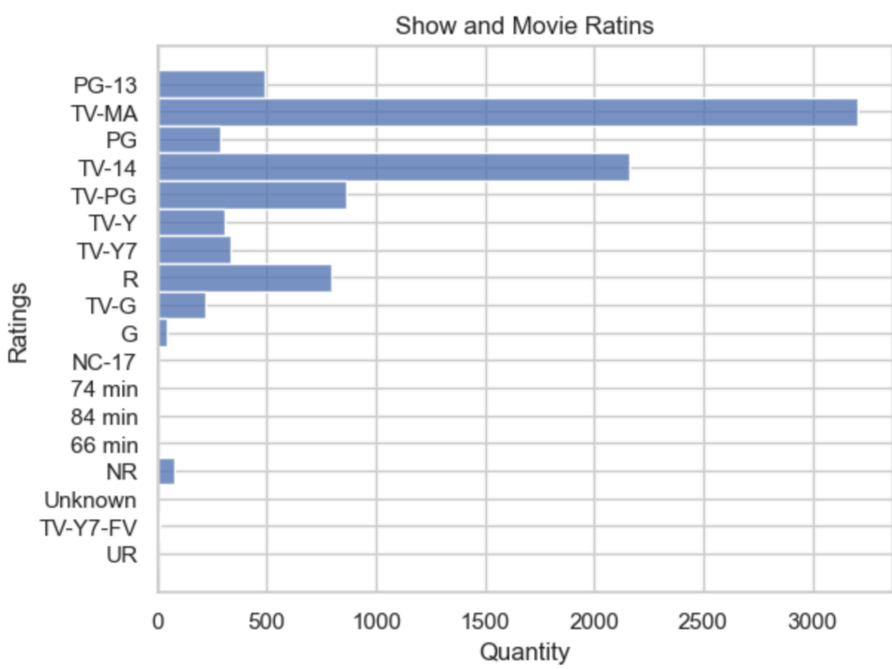
Release Year Distribution

- This graph shows the frequency of titles from their years released. We can see a boost in titles in recent years compared to the past. There is a dip in the number of titles around 2020, the COVID-19 pandemic could cause this.



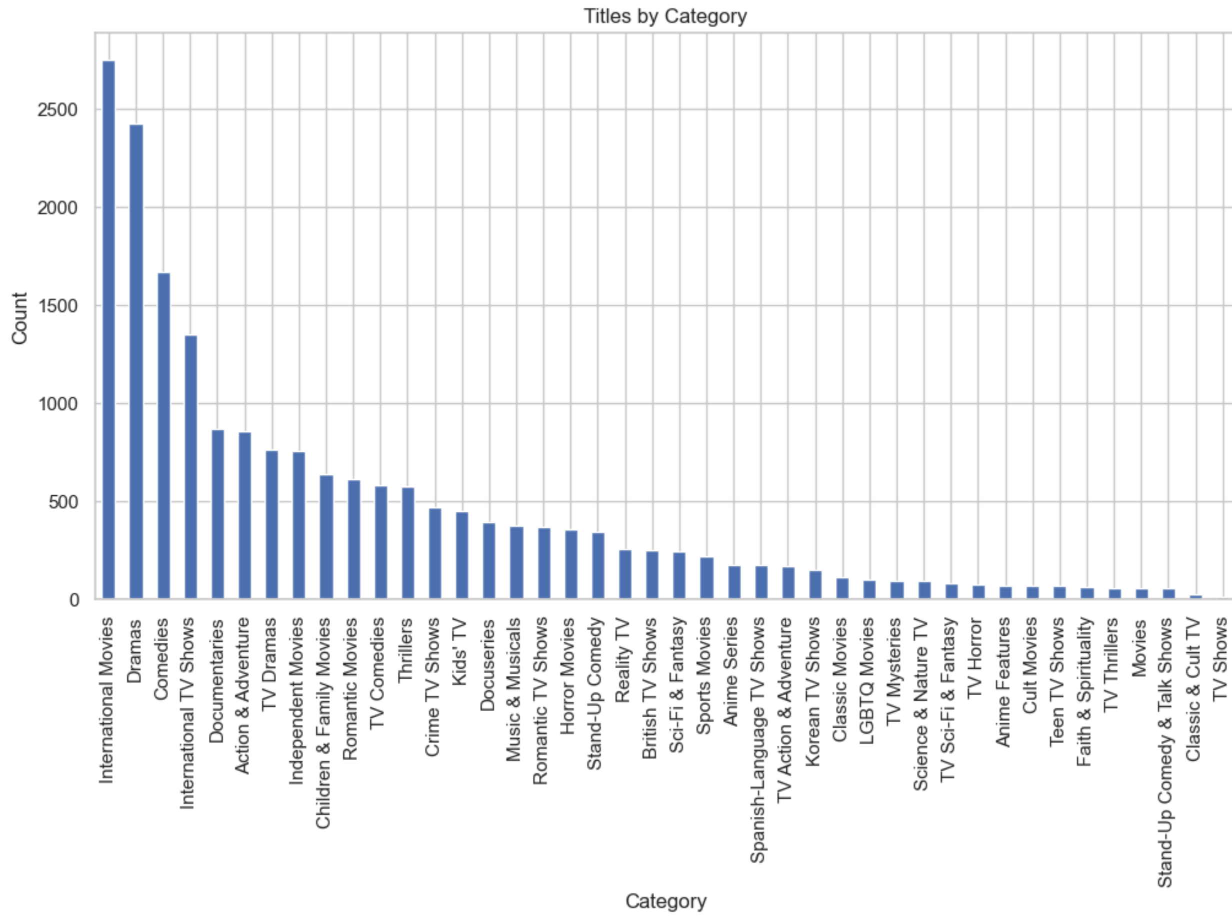
Rating Distribution

- This distribution helps us to understand the split of data amongst the title ratings. We can infer the type of distribution and the audience they cater to. Based on the data, we can see a large number of titles.



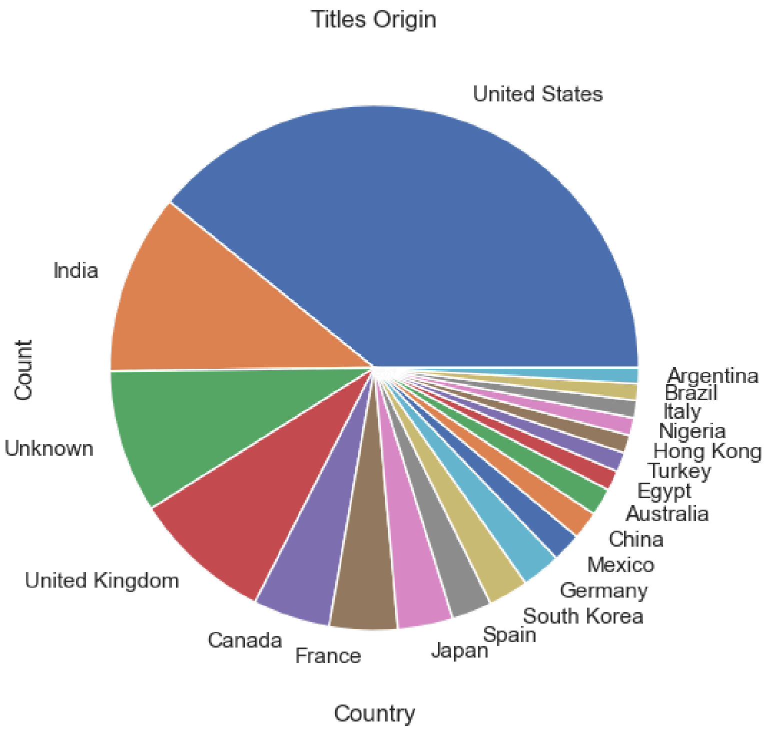
Genre Distribution

- This graph shows the distribution of the titles based on the genres they are in. There is a large collection of international titles as they started to expand their business to other countries outside the US.



Titles Origin Distribution

- This chart shows the origin of titles from a wide range of countries. Comparing and contrasting with the genre distribution and origin distribution helps to understand the numbers of titles clearly.



Part 2: Linear Regression Analysis

Wine Quality Dataset

Loss Value and Weight Vector

The loss value for the Wine Quality dataset linear regression model is calculated using the RMSE. The weight vector w\mathbf{w}w obtained from the OLS method is used for predictions.

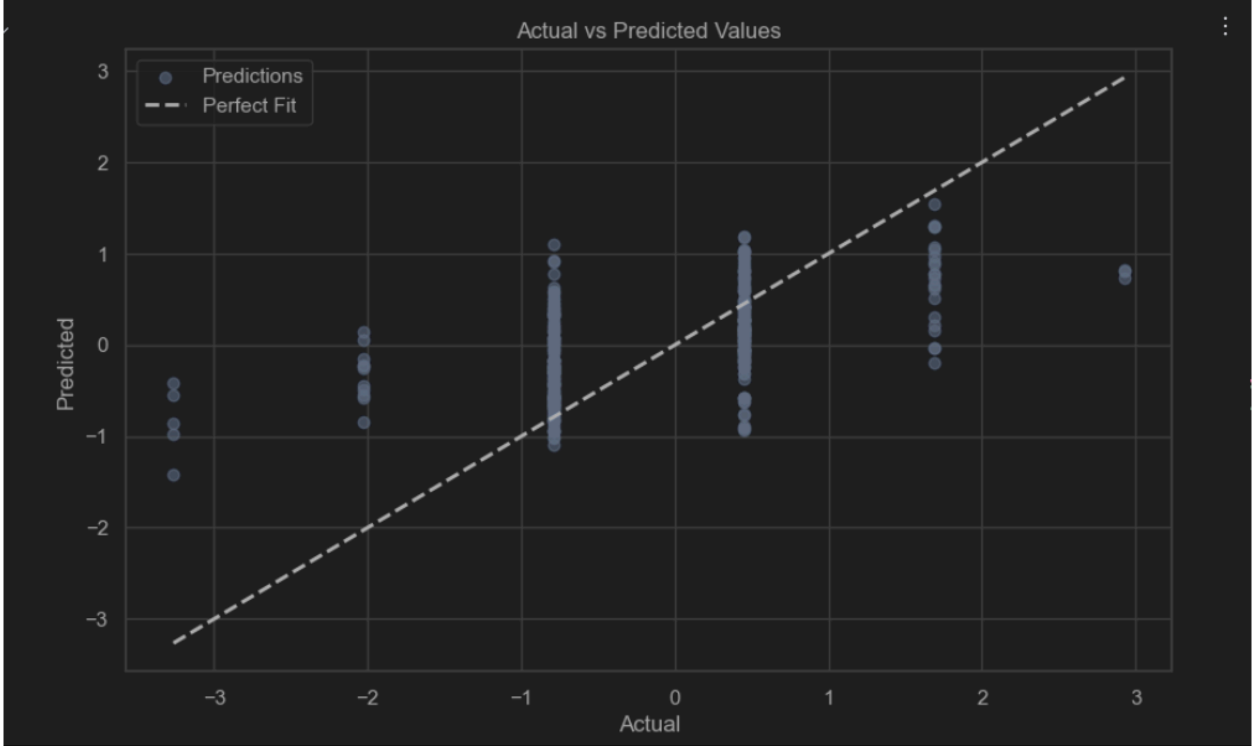
```
# 8. Calculate the weight vector w using the OLS equation
w = np.linalg.inv(X_train.T @ X_train) @ X_train.T @ y_train
print("Weights:", w)
Executed at 2024.06.15 22:45:08 in 5ms

 Weights: [[ 0.02920651]
  [ 0.04917558]
  [-0.23320567]
  [-0.04345051]
  [ 0.01849417]
  [-0.10248426]
  [ 0.04300297]
  [-0.15017197]
  [-0.04538778]
  [-0.06528552]
  [ 0.16943512]
  [ 0.37396083]]
```

```
# 9. Get predictions
y_pred_train = X_train @ w
y_pred_test = X_test @ w

# Calculate RMSE
rmse_train = np.sqrt(np.mean((y_train - y_pred_train) ** 2))
rmse_test = np.sqrt(np.mean((y_test - y_pred_test) ** 2))

print("RMSE on training set:", rmse_train)
print("RMSE on test set:", rmse_test)

Executed at 2024.06.15 22:45:08 in 3ms

 RMSE on training set: 0.798377913572987
 RMSE on test set: 0.8134330763107801
```

Linear regression Plot

```python
# 10. Plot predictions vs actual data
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_test, alpha=0.6, color='b', label='Predictions')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2, label='Perfect
  Fit')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted Values')
plt.legend()
plt.show()
```

Executed at 2024.06.15 22:45:08 in 132ms



Netflix Titles Dataset

The loss value for the Netflix Titles dataset linear regression model is calculated using the RMSE. The weight vector w\mathbf{w}w obtained from the OLS method is used for predictions.

Loss value and Weight Vector

Click to add a breakpoint

```
w = np.linalg.inv(X_train.T @ X_train) @ X_train.T @ y_train
print("Weights:", w)

# x and y predictions
y_pred_train = X_train @ w
y_pred_test = X_test @ w
```

✓ 0.0s

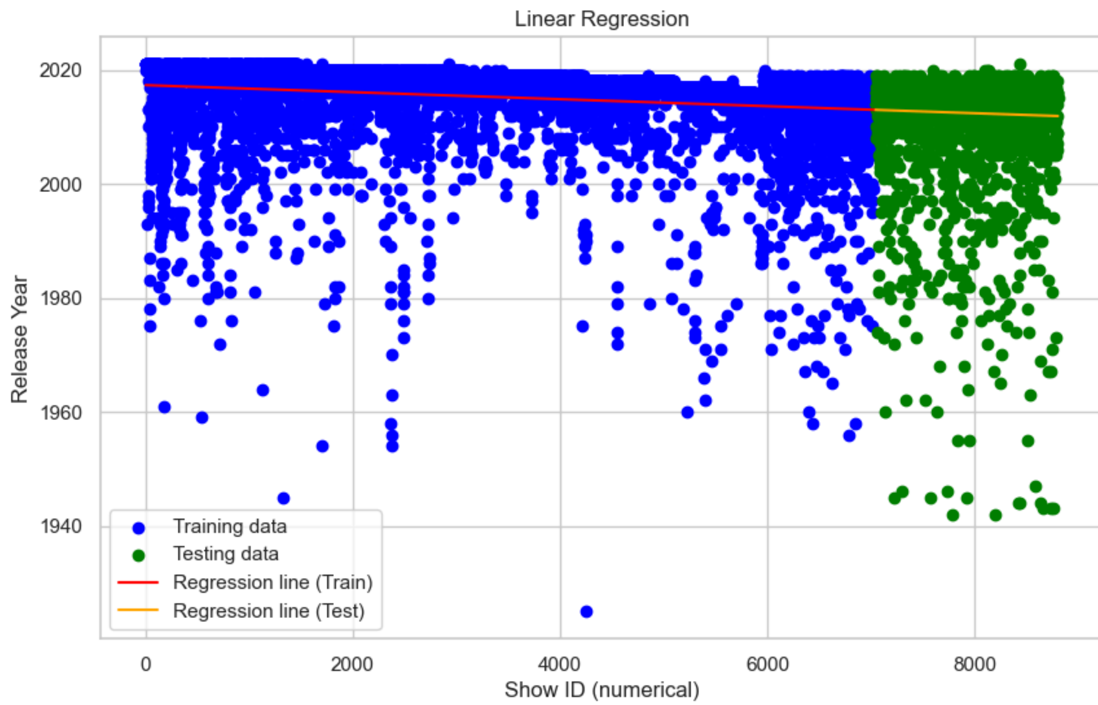
```
Weights: [ 2.01733587e+03 -6.15128407e-04]
```

```
# calc RMSE
rmse_train = np.sqrt(np.mean((y_train - y_pred_train) ** 2))
rmse_test = np.sqrt(np.mean((y_test - y_pred_test) ** 2))

print("RMSE Data")
print(f'Training RMSE: {rmse_train}')
print(f'Testing RMSE: {rmse_test}')
```

✓ 0.0s

```
RMSE Data
Training RMSE: 7.586220339614318
Testing RMSE: 11.739991225932693
```

Linear regression Plot



Benefits/Drawbacks of Using OLS Estimate for Computing Weights

- Benefits:
  - Simplicity: OLS is straightforward to implement and understand. It directly minimizes the sum of squared residuals, providing a simple solution to linear regression problems.
  - Interpretability: The results of OLS are easy to interpret. Each coefficient represents the change in the dependent variable for a one-unit change in the independent variable, holding all other variables constant.
- Drawbacks:
  - Overfitting: OLS can overfit the training data, especially when the number of features is large or when there is multicollinearity. This leads to poor generalization to unseen data.
  - Sensitivity: OLS is highly sensitive to outliers. Outliers can significantly affect the fit of the model, leading to unreliable estimates.
  - Multicollinearity: When independent variables are highly correlated, the variance of the OLS estimates increases, making the estimates unstable and less reliable.
  - OLS relies on several assumptions. Violations of these assumptions can lead to biased or inefficient estimates.

Part III: Ridge Regression Analysis

Wine Quality Dataset

Loss Value and Weight Vector

The loss value for the Wine Quality dataset ridge regression model is calculated using the RMSE. The weight vector w(ridge) obtained from the Ridge Regression method is used for predictions.

Weight Vector:

```
#6. Calculate the weights with the Ridge Regression equation.
# Set the regularization parameter lambda (can be adjusted as needed)
lambda_reg = 1.0

# Calculate the weight vector for Ridge Regression
I = np.eye(X_train.shape[1])  # Identity matrix
I[0, 0] = 0  # Do not regularize the bias term
w_ridge = np.linalg.inv(X_train.T @ X_train + lambda_reg * I) @ X_train.T @ y_train
print("Weights:", w_ridge)
```

Executed at 2024.06.20 20:19:55 in 6ms

```
Weights: [-7.97505965e+01 -2.12504056e-05  1.03833962e+00  2.91199323e-06
  1.45698438e-05 -1.07540298e-05 -1.37362237e-03  7.71511950e-05
  4.43539241e-02  2.69181831e-03  1.14985662e-03  2.77502213e-05]
```

```
#7. Predict training and test set values for Ridge Regression
y_train_pred_ridge = X_train @ w_ridge
y_test_pred_ridge = X_test @ w_ridge

# Calculate the RMSE for Ridge Regression
rmse_train_ridge = np.sqrt(np.mean((y_train - y_train_pred_ridge) ** 2))
rmse_test_ridge = np.sqrt(np.mean((y_test - y_test_pred_ridge) ** 2))

print("Ridge Regression RMSE on Training Set:", rmse_train_ridge)
print("Ridge Regression RMSE on Test Set:", rmse_test_ridge)
```

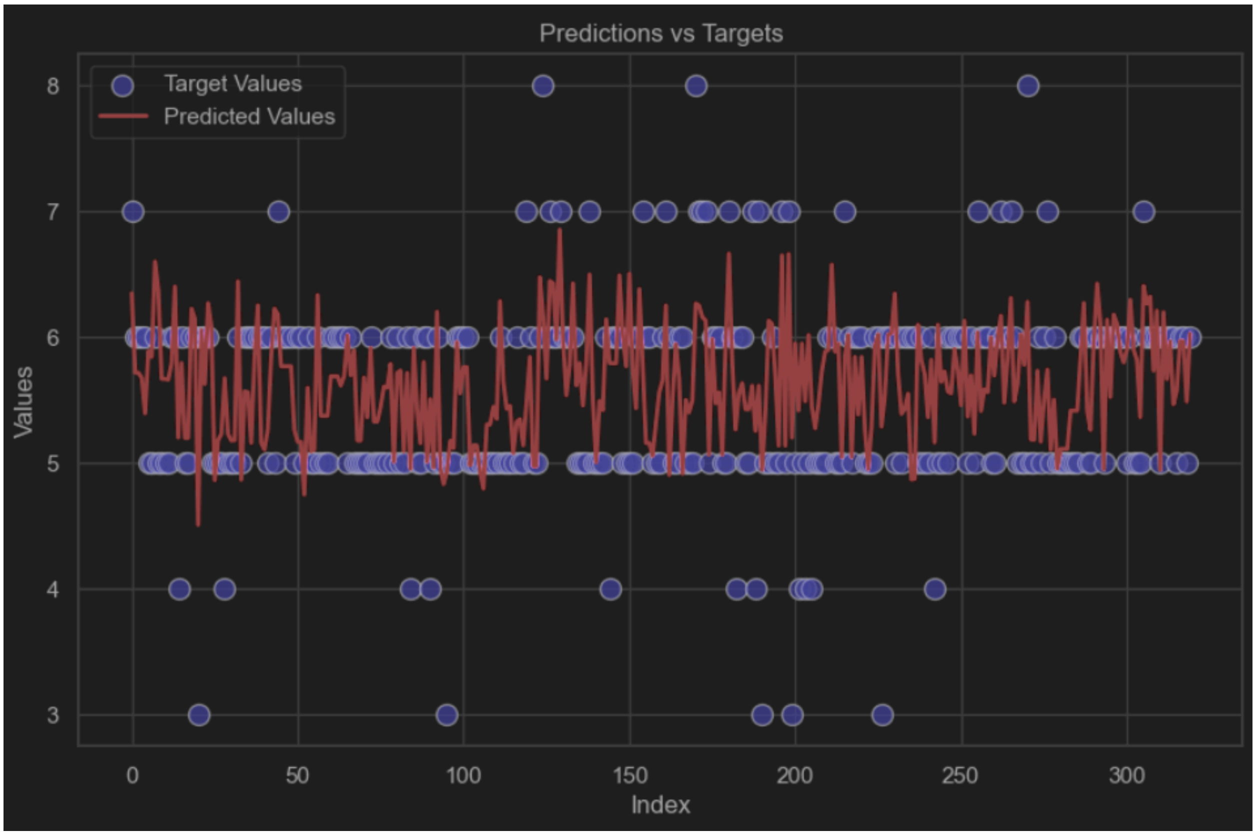
Executed at 2024.06.20 20:19:57 in 4ms

```
Ridge Regression RMSE on Training Set: 1.7003243392066736
Ridge Regression RMSE on Test Set: 2.1554309430122904
```

Ridge Regression Plot:

```
# Plot predictions vs actual values for Ridge Regression
plot_predictions_and_targets(y_test_pred_ridge, y_test)
```

Executed at 2024.06.20 19:33:47 in 144ms



Netflix Titles Analysis

Loss Value and Weight Vector

The loss value for the Netflix titles dataset ridge regression model is calculated using the RMSE. The RMSE values for the regression are below:

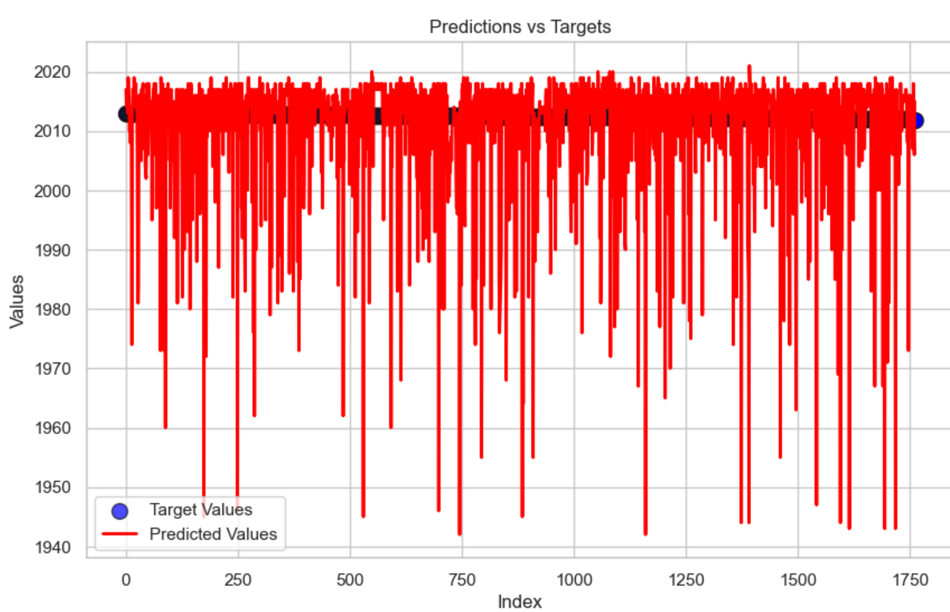
Ridge Regression RMSE on Training Set: 7.586220339614316

Ridge Regression RMSE on Test Set: 11.739991225950643

Weight Vector:

```
#6. Calculate the weights with the Ridge Regression equation.
# Set the regularization parameter lambda (can be adjusted as needed)
lambda_reg = 1.0

# Calculate the weight vector for Ridge Regression
I = np.eye(X_train.shape[1])  # Identity matrix
I[0, 0] = 0  # Do not regularize the bias term
w_ridge = np.linalg.inv(X_train.T @ X_train + lambda_reg * I) @ X_train.T @ y_train
print(w_ridge)

[ 2.01733587e+03 -6.15128407e-04]
```

Ridge Regression Plot:



Difference Between Linear and Ridge Regressions

- Linear Regression:
  - OLS (Ordinary Least Squares) estimates weights by minimizing the sum of squared residuals.
  - Susceptible to overfitting, especially with a high number of features.
  - Sensitive to multicollinearity and outliers.
- Ridge Regression:
  - Introduces L2 regularization (penalty term) to the loss function, which is the sum of squared residuals plus the squared magnitude of coefficients.
  - Helps prevent overfitting by shrinking the coefficients.
  - More robust to multicollinearity and less sensitive to outliers.

Motivation of Using L2 Regularization:

- The primary motivation is to prevent overfitting by adding a penalty term to the loss function, which discourages large coefficients.
- It improves the generalization performance of the model, especially when the number of features is large or when multicollinearity exists.

Bonus Points

Wine Quality Dataset:

```
Iteration 0, Gradient Norm: 266.5287393707014
Iteration 1000, Gradient Norm: 12.217312990852763
Iteration 2000, Gradient Norm: 6.08073985874553
Iteration 3000, Gradient Norm: 3.384831680680049
Iteration 4000, Gradient Norm: 2.0672754583615744
Iteration 5000, Gradient Norm: 1.3484970463340886
Iteration 6000, Gradient Norm: 0.931650333432515
Iteration 7000, Gradient Norm: 0.6888945648530576
Iteration 8000, Gradient Norm: 0.5521591699753913
Iteration 9000, Gradient Norm: 0.47795898343177895
Gradient Descent Ridge Regression RMSE on Training Set: 0.7468068618693352
Gradient Descent Ridge Regression RMSE on Test Set: 0.7484526618789604
Comparison of RMSE:
Ridge Regression (Analytical) RMSE on Training Set: 0.6454028731361967
Ridge Regression (Analytical) RMSE on Test Set: 0.6582388965491944
Ridge Regression (Gradient Descent) RMSE on Training Set: 0.7468068618693352
Ridge Regression (Gradient Descent) RMSE on Test Set: 0.7484526618789604
```

Netflix Dataset:

```
Iteration 0, Predictions: [-4.12113551e-05 -3.61341833e-04 -1.18681152e-04 -2.32321001e-04
 -1.61640344e-04], Errors: [-2020.00004121 -2021.00036134 -2021.00011868 -2021.00023232
 -2021.00016164], Gradient: [ 2.50397778e-09  1.27895305e+00 -1.44076001e+00 -6.02617596e-01
  1.25950086e+00], Weights: [1.76405235e-04 4.00029313e-05 9.78882060e-05 2.24095346e-04
 1.86743204e-04]
Iteration 1000, Predictions: [-7.80590736e-05 -2.93939792e-04 -8.45466831e-05 -1.72034624e-04
 -1.09985364e-04], Errors: [-2020.00007806 -2021.00029394 -2021.00008455 -2021.00017203
 -2021.00010999], Gradient: [ 2.50397778e-09  1.27893577e+00 -1.44072614e+00 -6.02606010e-01
  1.25947162e+00], Weights: [1.76405235e-04 2.72134873e-05 1.12295637e-04 2.30121464e-04
 1.74148342e-04]
Iteration 2000, Predictions: [-1.14905880e-04 -2.26539015e-04 -5.04129674e-05 -1.11749565e-04
 -5.83314736e-05], Errors: [-2020.00011491 -2021.00022654 -2021.00005041 -2021.00011175
 -2021.00005833], Gradient: [ 2.50397778e-09  1.27891848e+00 -1.44069227e+00 -6.02594424e-01
  1.25944239e+00], Weights: [1.76405235e-04 1.44242162e-05 1.26702729e-04 2.36147466e-04
 1.61553772e-04]
Gradient Descent Ridge Regression RMSE on Training Set: 2015.1834396579616
Gradient Descent Ridge Regression RMSE on Test Set: 2010.2604475342303
Comparison of RMSE:
Ridge Regression (Analytical) RMSE on Training Set: 7.58622033961432
Ridge Regression (Analytical) RMSE on Test Set: 11.739991225950522
Ridge Regression (Gradient Descent) RMSE on Training Set: 2015.1834396579616
Ridge Regression (Gradient Descent) RMSE on Test Set: 2010.2604475342303
```

Team Contribution

Team Member	Assignment Part	Contribution
Siqi Cheng	Wine Quality Dataset All Parts	50%
Mukesh Thamilvanan	Netflix Dataset All Parts	50%