Linear Regression Analysis Report

This report outlines the process of performing linear regression analysis on a dataset to predict trip duration based on various features. The analysis includes checks for linearity, multicollinearity, outliers, and residual analysis, as well as training and evaluating both linear and ridge regression models. The following steps are implemented:

1. **Linearity Check**: A scatter plot is created to visualize the relationship between 'Distance (km)' and 'Trip Duration (min)'.
2. **Independence Check**: A heatmap displays the correlation matrix to identify multicollinearity among the features.
3. **Outlier Check**: A boxplot and histogram visualize the distribution of trip durations and identify potential outliers.
4. **Model Training - Linear Regression**: A linear regression model is trained on the training data, and predictions are made on the validation set.
5. **Residual Analysis**: The residuals are analyzed through scatter plots, histograms, and Q-Q plots to check for patterns and normality.
6. **Model Evaluation**: The root mean square error (RMSE) and R^2 score are calculated to evaluate the performance of the linear regression model.
7. **Model Training - Ridge Regression**: A ridge regression model is trained and evaluated, and predictions are made on the validation set.
8. **Prediction Visualization**: A scatter plot compares the actual trip durations with the predicted values from the linear regression model.

Now, let's see the code implementation.

libraries

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

from sklearn.linear_model import LinearRegression, Ridge

from sklearn.metrics import mean_squared_error, r2_score
```

- Load training and validation data

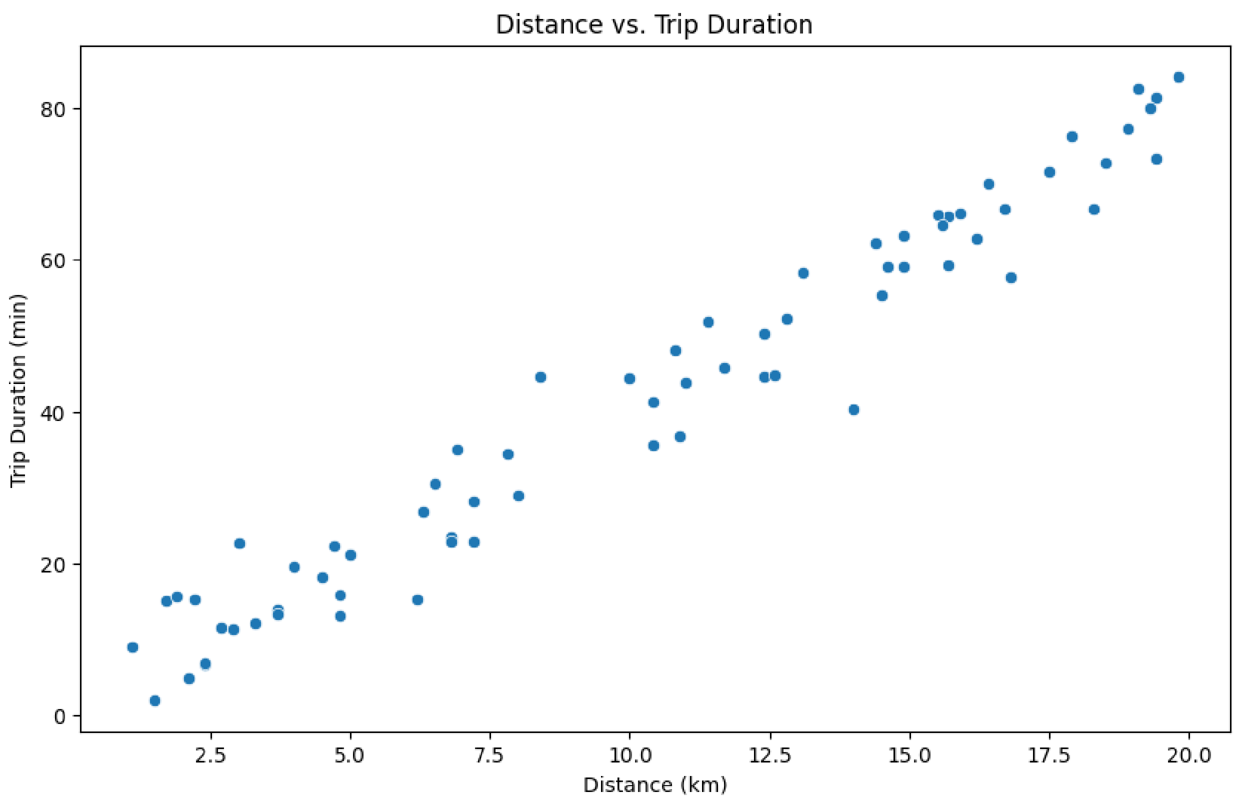
  ```
  X_train = pd.read_csv('../data/splits/X_train.csv')

  y_train = pd.read_csv('../data/splits/y_train.csv')
  ```

```
X_val = pd.read_csv('../data/splits/X_val.csv')
y_val = pd.read_csv('../data/splits/y_val.csv')
```

1. Linearity Check

- Check linear relationship between 'Distance (km)' and 'Trip Duration'

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_train['Distance (km)'], y=y_train.squeeze())
plt.xlabel('Distance (km)')
plt.ylabel('Trip Duration (min)')
plt.title('Distance vs. Trip Duration')
plt.show()
```

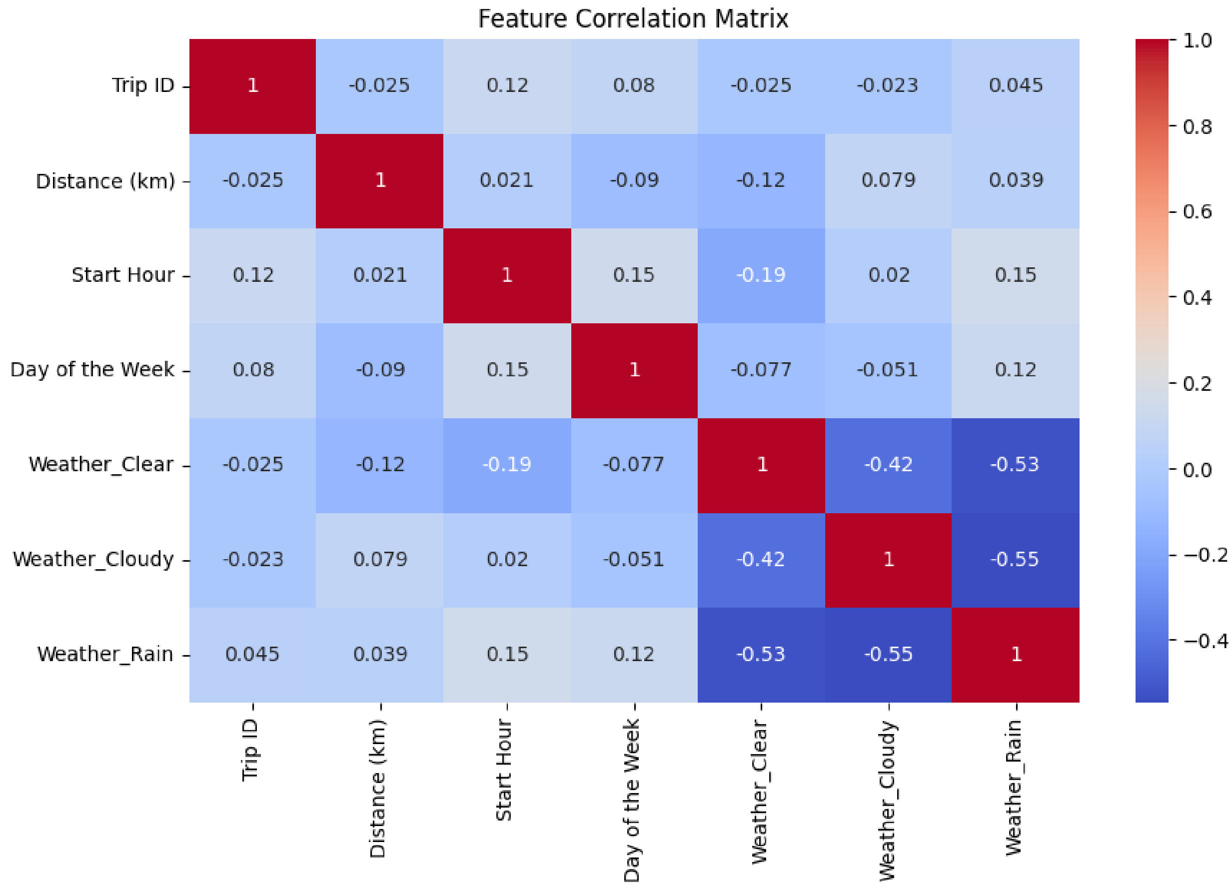


2. Independence Check

- Correlation matrix to check for multicollinearity among features

```
plt.figure(figsize=(10, 6))
sns.heatmap(X_train.corr(), annot=True, cmap="coolwarm")
plt.title('Feature Correlation Matrix')
```

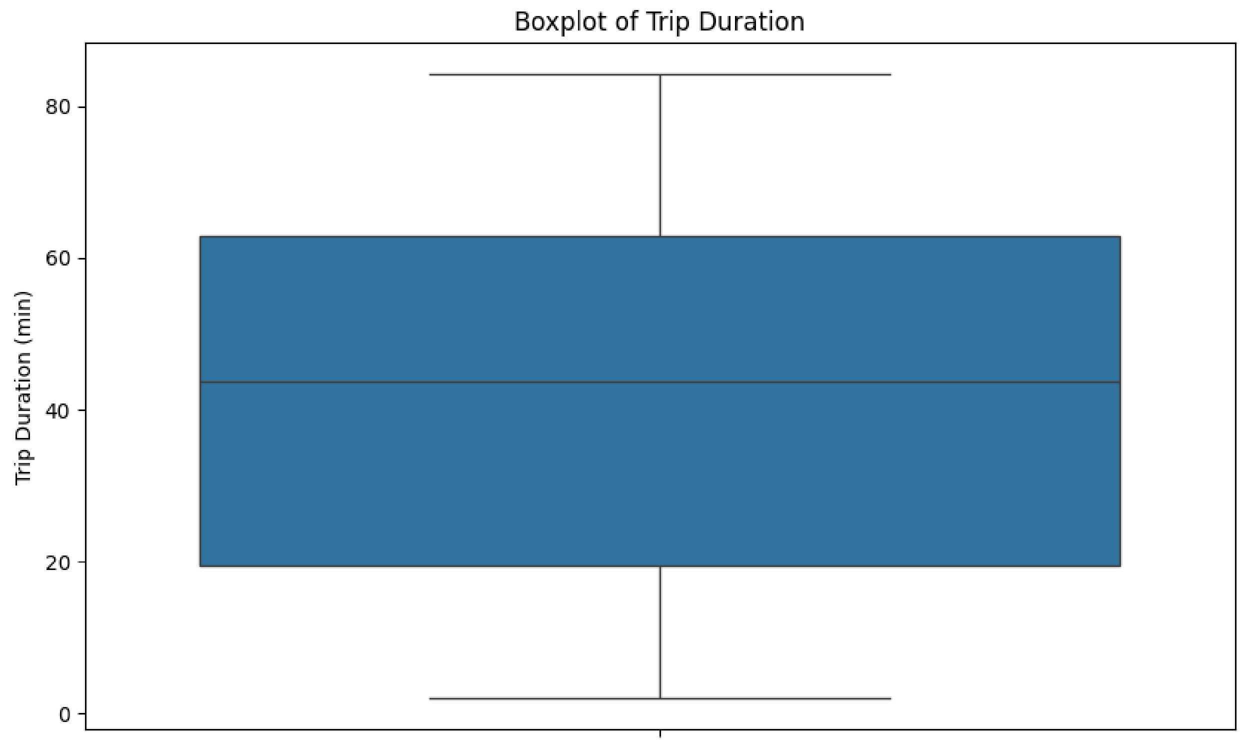
```
plt.show()
```



3. Outlier Check in Target Variable

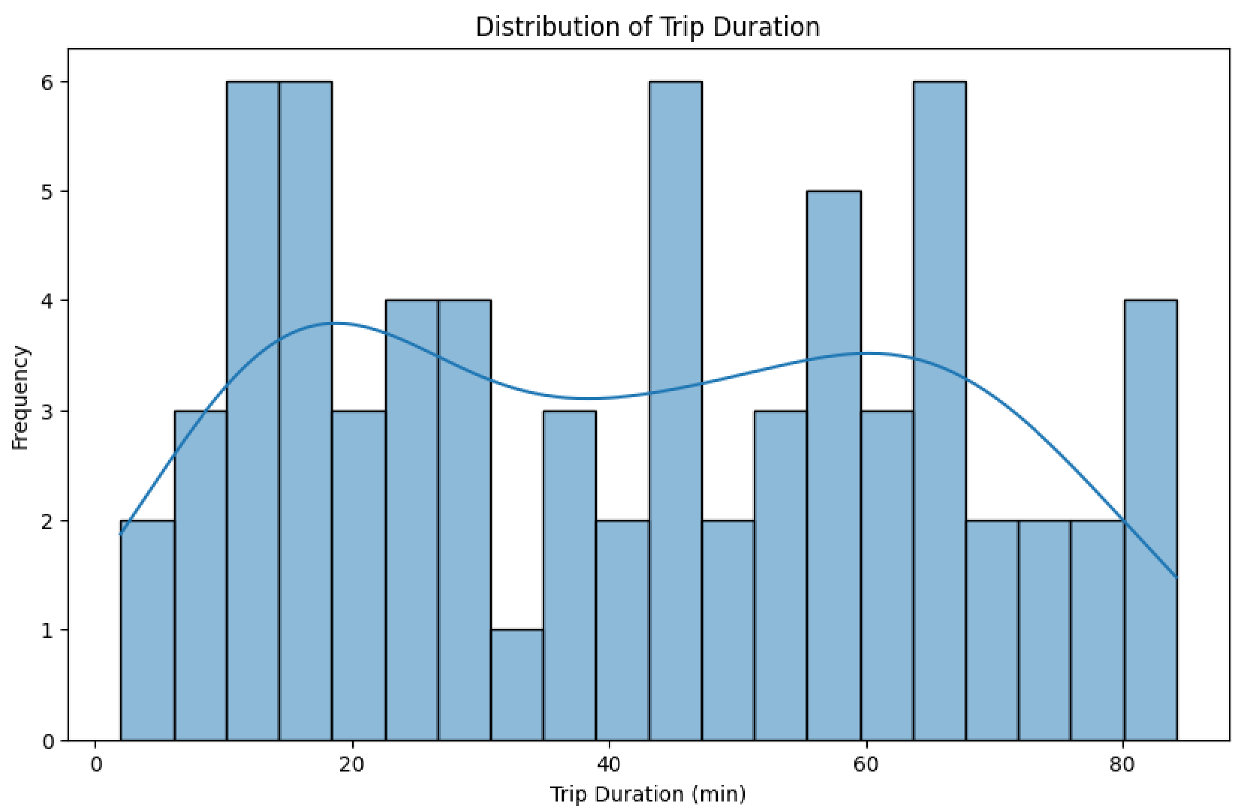
- Boxplot to identify outliers in 'Trip Duration'

```
plt.figure(figsize=(10, 6))
sns.boxplot(y=y_train.squeeze())
plt.title('Boxplot of Trip Duration')
plt.ylabel('Trip Duration (min)')
plt.show()
```



Histogram of Trip Duration

```
plt.figure(figsize=(10, 6))
sns.histplot(y_train.squeeze(), bins=20, kde=True)
plt.title('Distribution of Trip Duration')
plt.xlabel('Trip Duration (min)')
plt.ylabel('Frequency')
plt.show()
```



4. Model Training - Linear Regression

- Train a Linear Regression model

  model = LinearRegression()

  model.fit(X_train, y_train)

- Make predictions on the validation set

  y_pred = model.predict(X_val)

- Print shapes of y_val and y_pred

  print("Shape of y_val:", y_val.shape)

  print("Shape of y_pred:", y_pred.shape)

  Shape of y_val: (16, 1)

  Shape of y_pred: (16, 1)

5. Residual Analysis

- Calculate residuals by flattening the arrays to ensure they are 1D

  residuals = y_val.values.flatten() - y_pred.flatten()

- **Residuals vs. Fitted Values**

- Make predictions on the validation set

  y_pred = model.predict(X_val)

- Check shapes before calculating residuals

  print("Shape of y_val:", y_val.shape)

  print("Shape of y_pred:", y_pred.shape)

  Shape of y_val: (16, 1)

  Shape of y_pred: (16, 1)

- Calculate residuals

  residuals = y_val.values.flatten() - y_pred.flatten()

- Assuming y_pred and residuals are 1D numpy arrays, convert them to Series

  y_pred_series = pd.Series(y_pred.flatten())

  residuals_series = pd.Series(residuals.flatten())

- Now plot using the Series

  plt.figure(figsize=(10, 6))

  sns.scatterplot(x=y_pred_series, y=residuals_series)

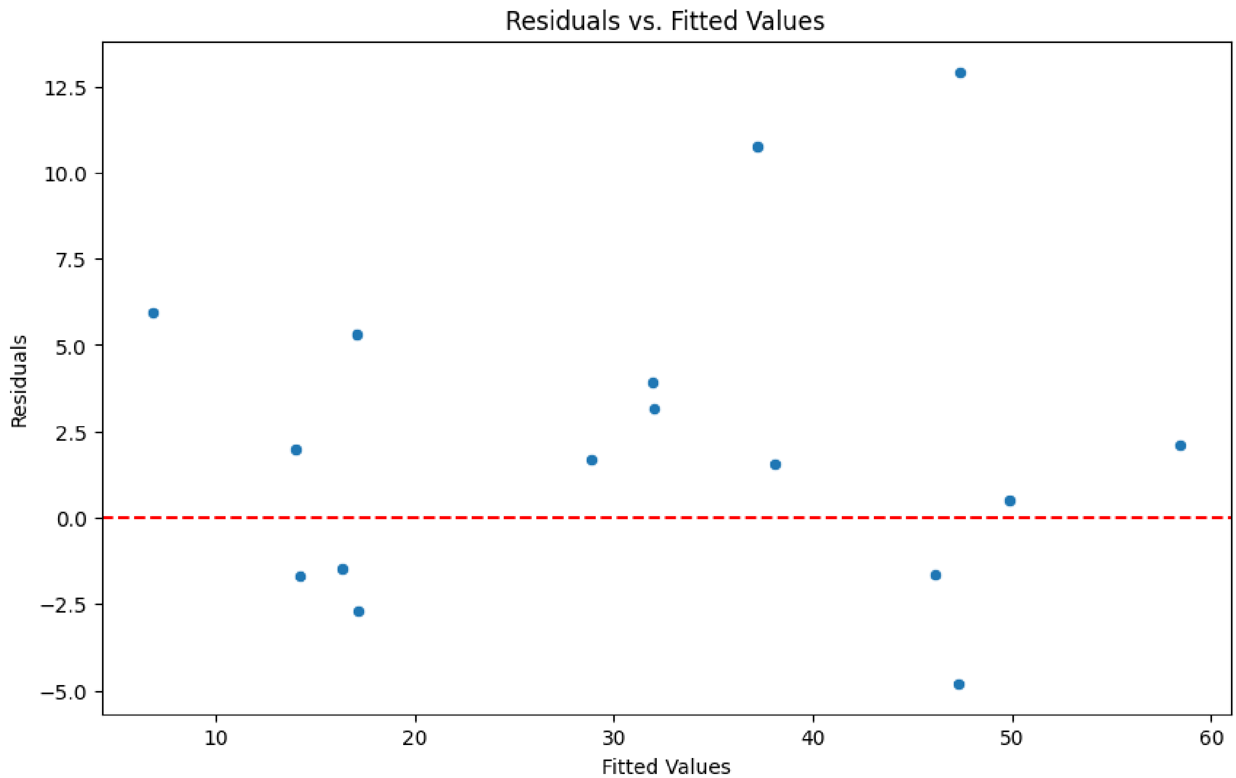
  plt.axhline(0, color='red', linestyle='--')

  plt.xlabel('Fitted Values')

  plt.ylabel('Residuals')

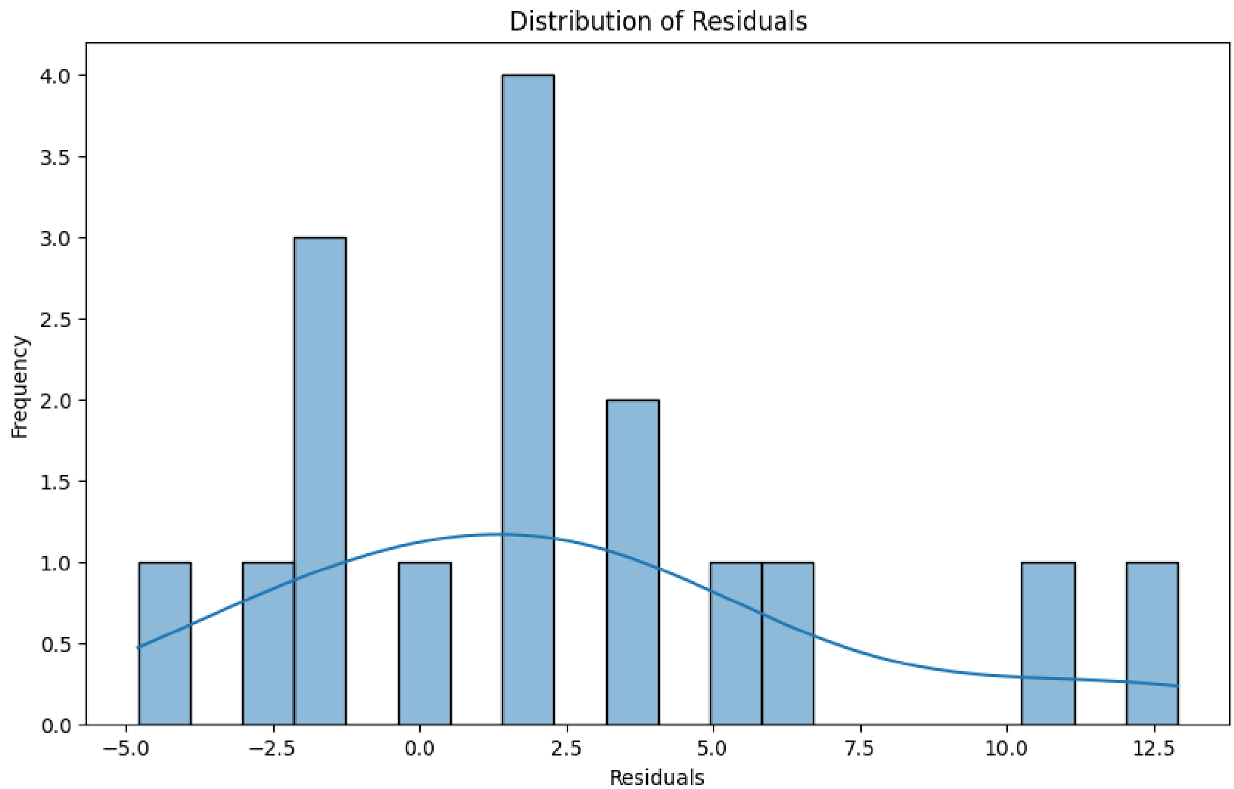
  plt.title('Residuals vs. Fitted Values')

  plt.show()



- Histogram of Residuals

```
plt.figure(figsize=(10, 6))

sns.histplot(residuals, bins=20, kde=True)

plt.title('Distribution of Residuals')

plt.xlabel('Residuals')

plt.ylabel('Frequency')

plt.show()
```



6. Model Evaluation

- Make predictions on the validation set

  y_pred = model.predict(X_val)

  y_pred = model.predict(X_val)

  y_pred = y_pred.flatten() *# Flatten the predictions to make them 1D*

- Check the shapes to ensure they match

  print("Shape of y_val:", y_val.shape)

  print("Shape of y_pred:", y_pred.shape)

  Shape of y_val: (16, 1)

  Shape of y_pred: (16,)

- Calculate RMSE and R^2 Score for the Linear Regression model

  mse = mean_squared_error(y_val, y_pred, squared=**False**) *# RMSE*

  r2 = r2_score(y_val, y_pred)

  print(f"Linear Regression RMSE: {mse:.2f}")

  print(f"Linear Regression R^2 Score: {r2:.2f}")

Linear Regression RMSE: 5.14

Linear Regression R^2 Score: 0.90

7. Model Training - Ridge Regression (with regularization)

- Train a Ridge Regression model

```
ridge_model = Ridge(alpha=1.0)  # Modify alpha as needed
ridge_model.fit(X_train, y_train)
```

- Make predictions and evaluate Ridge model

```
y_pred_ridge = ridge_model.predict(X_train)
```

- Generate predictions on the validation set (X_val) for Ridge

```
y_pred_ridge = ridge_model.predict(X_val)
```

- Check shapes

```
print("Shape of y_val:", y_val.shape)
print("Shape of y_pred_ridge:", y_pred_ridge.shape)
```

Shape of y_val: (16, 1)

Shape of y_pred_ridge: (16, 1)

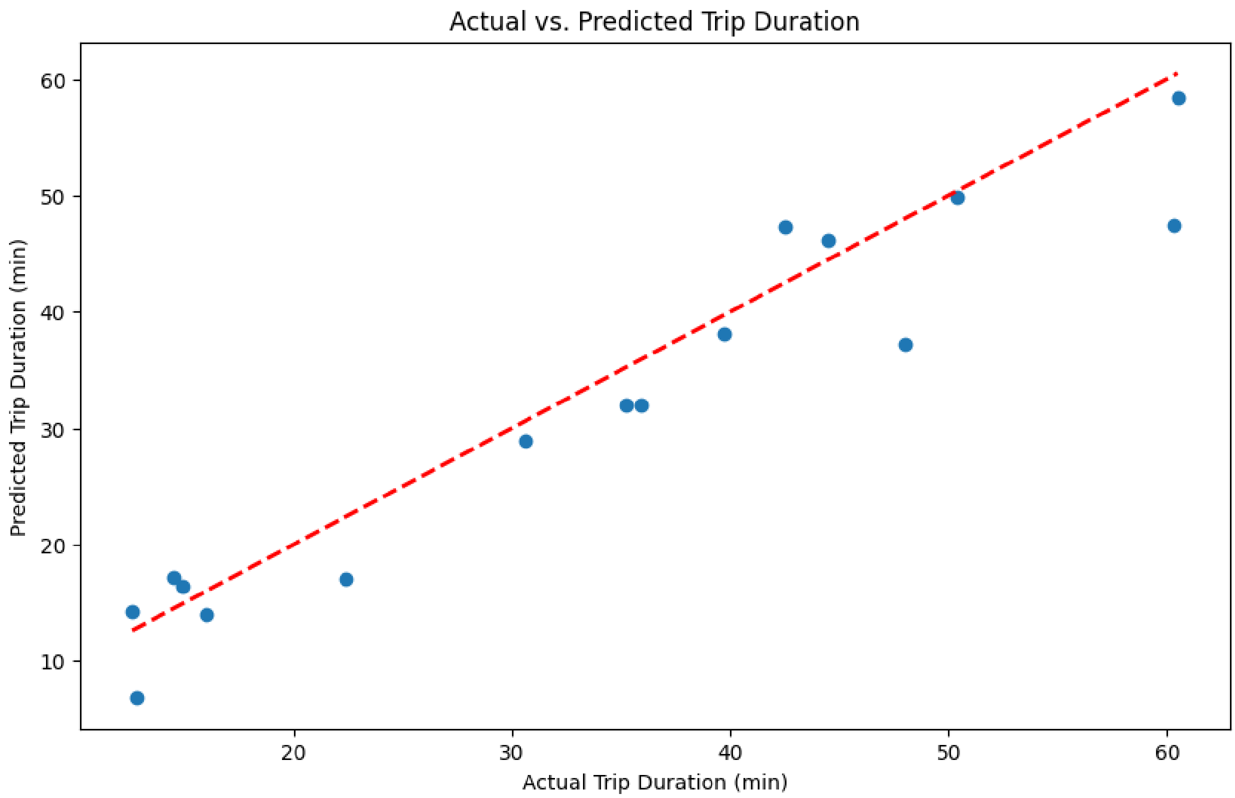
- Calculate RMSE and print

```
mse_ridge = mean_squared_error(y_val, y_pred_ridge, squared=False)
print(f"Ridge Root Mean Squared Error: {mse_ridge:.2f}")
```

Ridge Root Mean Squared Error: 5.13

8.Prediction Visualization

```
plt.figure(figsize=(10, 6))
plt.scatter(y_val, y_pred)
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--', lw=2)
plt.xlabel('Actual Trip Duration (min)')
plt.ylabel('Predicted Trip Duration (min)')
plt.title('Actual vs. Predicted Trip Duration')
plt.show()
```



- Calculate residuals

  residuals = y_val.squeeze() - y_pred

- Residuals vs. Fitted Values Plot

  plt.figure(figsize=(10, 6))

  sns.scatterplot(x=y_pred, y=residuals)

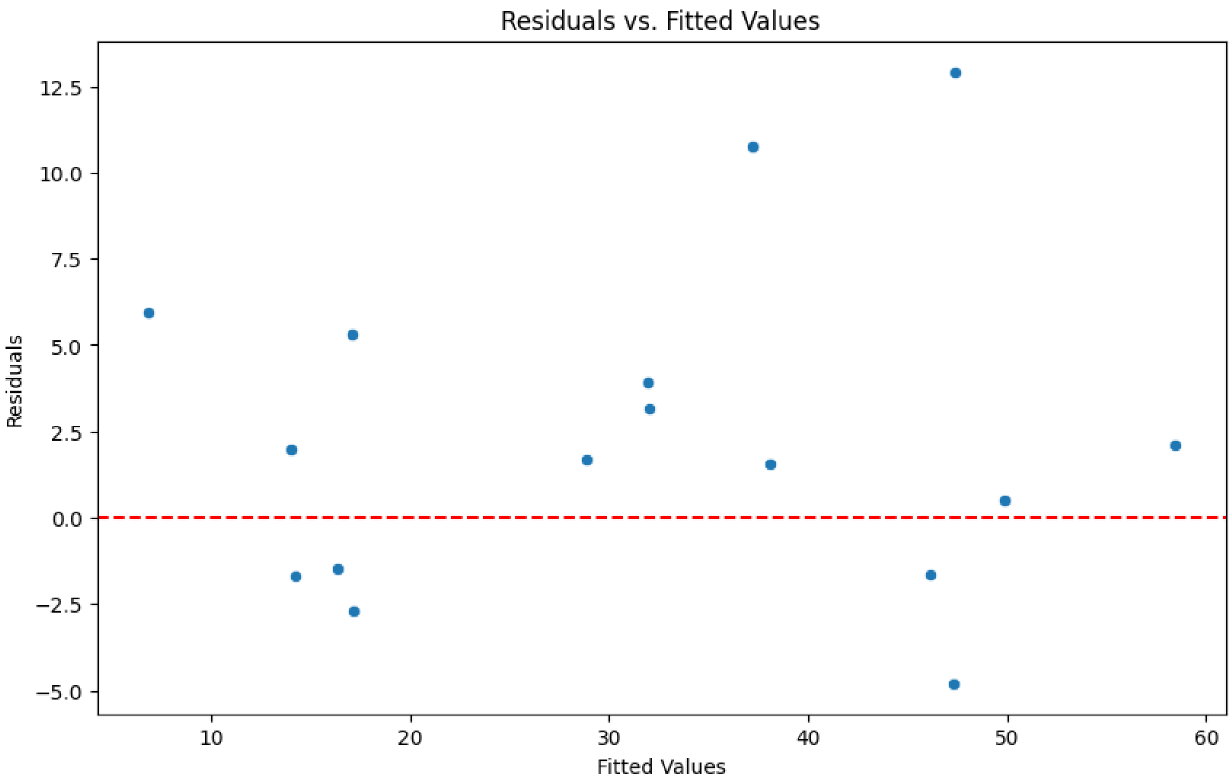
  plt.axhline(0, color='red', linestyle='--')

  plt.xlabel('Fitted Values')

  plt.ylabel('Residuals')

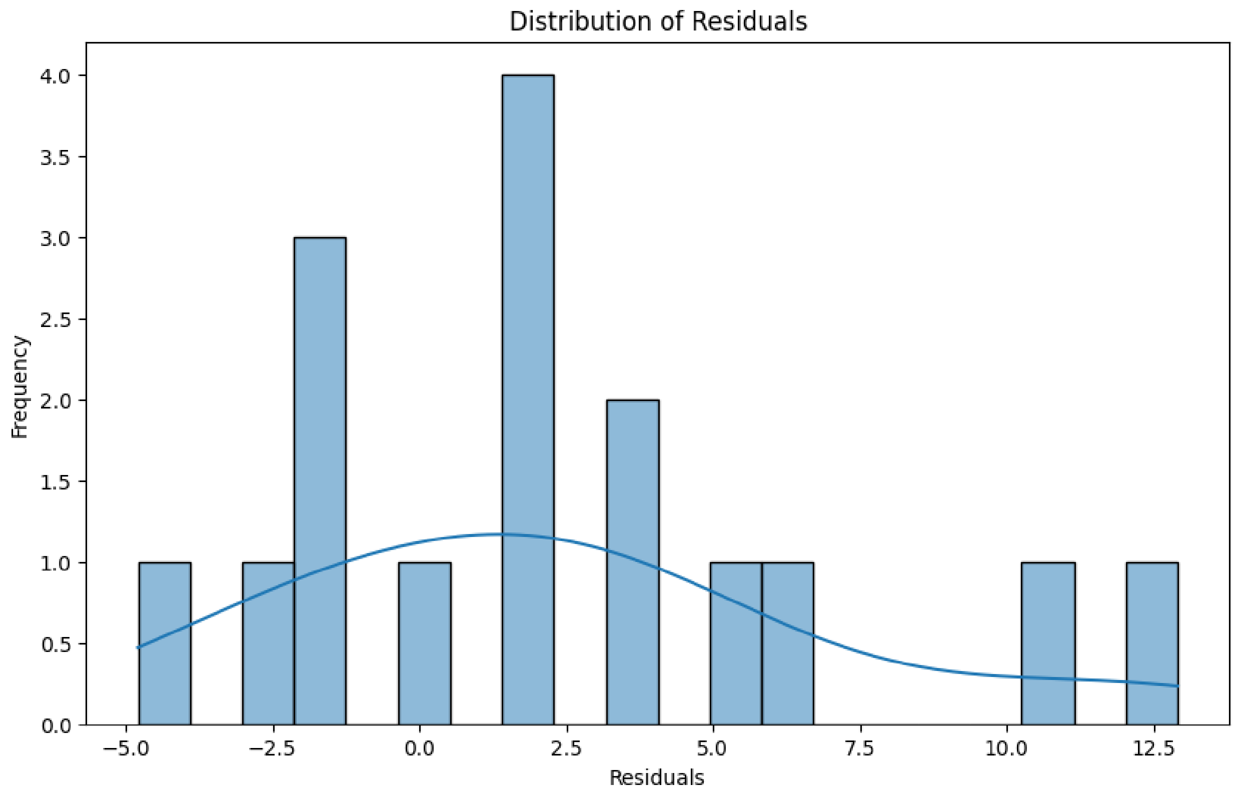
  plt.title('Residuals vs. Fitted Values')

  plt.show()



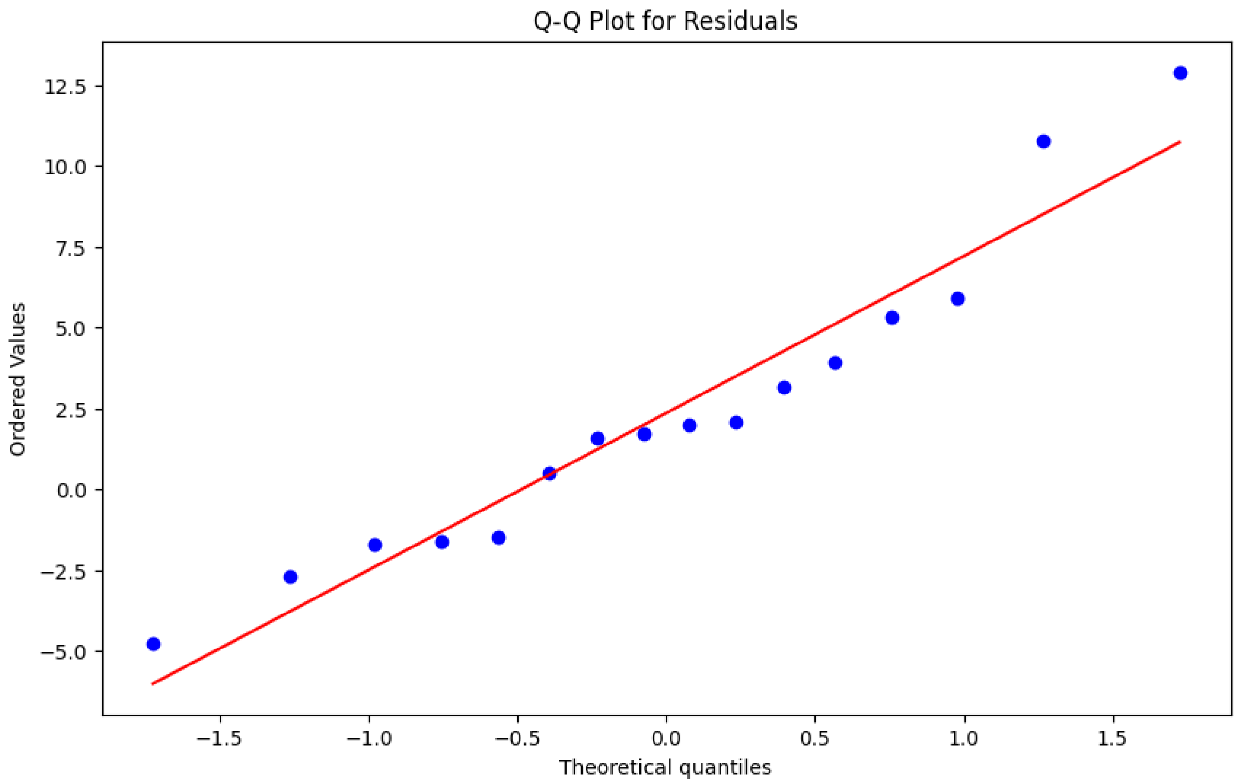
- Histogram of Residuals

```
plt.figure(figsize=(10, 6))

sns.histplot(residuals, bins=20, kde=True)

plt.title('Distribution of Residuals')

plt.xlabel('Residuals')

plt.ylabel('Frequency')

plt.show()
```



- Q-Q Plot for Residuals

```
plt.figure(figsize=(10, 6))
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot for Residuals')
plt.show()
```



Q-Q Plot for Residuals
Ordered Values
Theoretical quantiles
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5