# Linear_regression_exercise

September 27, 2023

[ ]:
```python
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

[ ]:
```python
# Load the diabetes dataset
auto_df = pd.read_csv("/Users/alexanderdelriscomorales/Downloads/AI_ML_Files/
↪Auto.csv")
auto_df.info(show_counts=True)
auto_df['horsepower'].replace('?', np.nan, inplace=True)
# Drop rows with NaN values in the 'horsepower' column
auto_df = auto_df.dropna(subset=['horsepower'])
print(auto_df.shape)
display(auto_df.head(n=10))
print(auto_df.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 397 entries, 0 to 396
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           397 non-null    float64
 1   cylinders     397 non-null    int64
 2   displacement  397 non-null    float64
 3   horsepower    397 non-null    object
 4   weight        397 non-null    int64
 5   acceleration  397 non-null    float64
 6   year          397 non-null    int64
 7   origin        397 non-null    int64
 8   name          397 non-null    object
dtypes: float64(3), int64(4), object(2)
memory usage: 28.0+ KB
(392, 9)
```

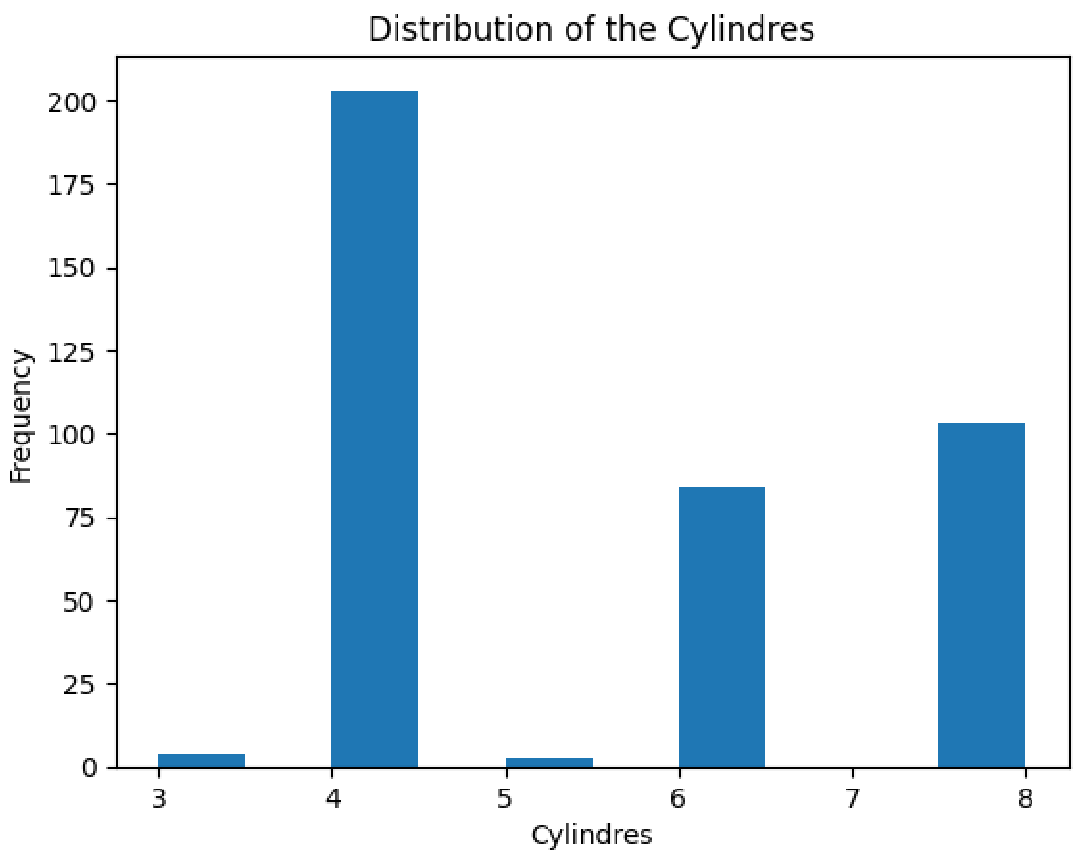
```
    mpg  cylinders  displacement horsepower  weight  acceleration  year  \
0  18.0          8         307.0        130    3504          12.0    70
1  15.0          8         350.0        165    3693          11.5    70
```

```
2  18.0          8         318.0        150     3436           11.0     70
3  16.0          8         304.0        150     3433           12.0     70
4  17.0          8         302.0        140     3449           10.5     70
5  15.0          8         429.0        198     4341           10.0     70
6  14.0          8         454.0        220     4354            9.0     70
7  14.0          8         440.0        215     4312            8.5     70
8  14.0          8         455.0        225     4425           10.0     70
9  15.0          8         390.0        190     3850            8.5     70

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
5       1           ford galaxie 500
6       1           chevrolet impala
7       1          plymouth fury iii
8       1           pontiac catalina
9       1         amc ambassador dpl
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
year            0
origin          0
name            0
dtype: int64
```

# 1 Cylinders

[ ]:
```
# Reshape the data
mpg = auto_df['mpg'].values.reshape(-1, 1)
cylinders = auto_df['cylinders'].values.reshape(-1, 1)
```

[ ]:
```
# 2. Data Exploration
# Explore and visualize your single feature.
plt.hist(cylinders)
plt.xlabel('Cylindres')
plt.ylabel('Frequency')
plt.title('Distribution of the Cylindres')
plt.show()
```




[ ]:
```
# 3. Data Splitting
X = auto_df[['cylinders']]  # Feature(s)
y = auto_df['mpg']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[ ]:
```
# 4. Model Selection
model = LinearRegression()
```

[ ]:
```
# 5. Model Training
model.fit(X_train, y_train)
```

[ ]: LinearRegression()

[ ]:
```
# 6. Model Evaluation
# For regression tasks (adjust as needed):
y_pred = model.predict(X_test)

# Calculate metrics (e.g., RMSE and R-squared)
mse = mean_squared_error(y_test, y_pred)
```

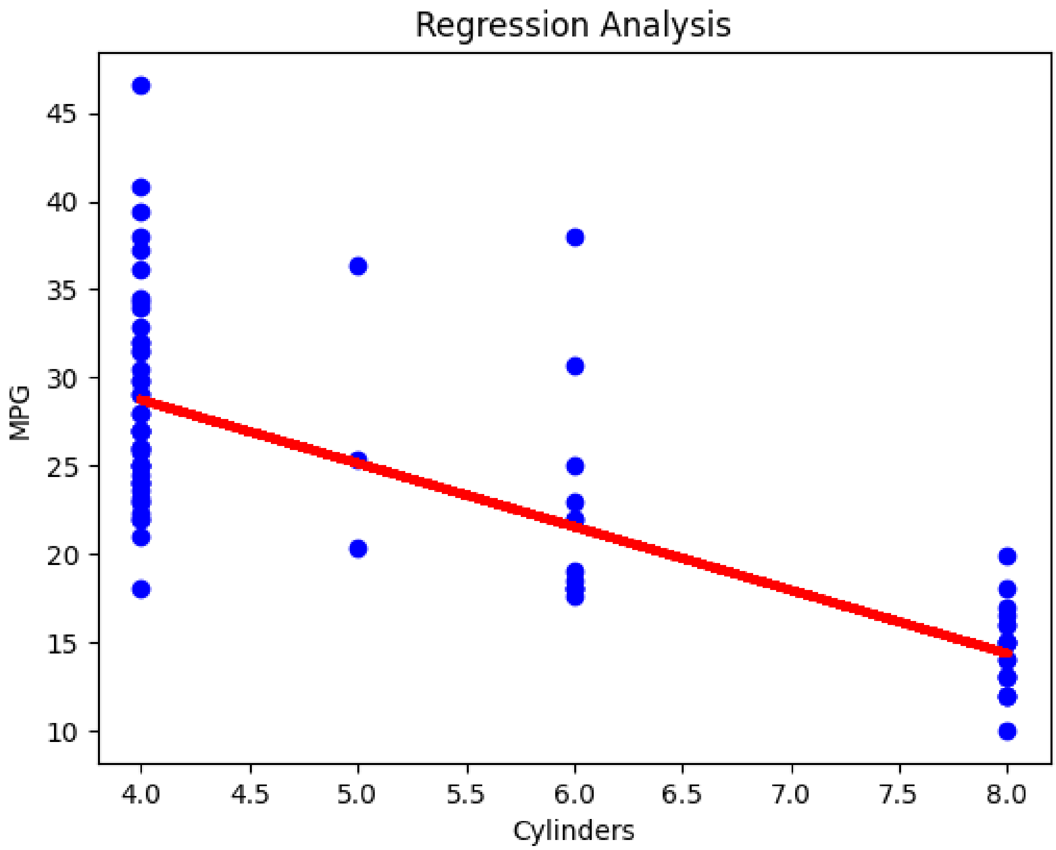
```python
print(f'Mean Squared Error: {mse}')
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')
r2 = r2_score(y_test, y_pred)
print(f'R2 score: {r2}')
```

```
Mean Squared Error: 28.1139504893218
Root Mean Squared Error: 5.3022589987025155
R2 score: 0.5418904610188053
```

[ ]:
```python
# Print the model coefficients
print('Intercept:', model.intercept_)
print('Coefficient:', model.coef_)
```

```
Intercept: 43.103277678094486
Coefficient: [-3.59388706]
```

[ ]:
```python
# 7. Visualization and Interpretation (for regression)
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=3)
plt.xlabel('Cylinders')
plt.ylabel('MPG')
plt.title('Regression Analysis')
plt.show()
```

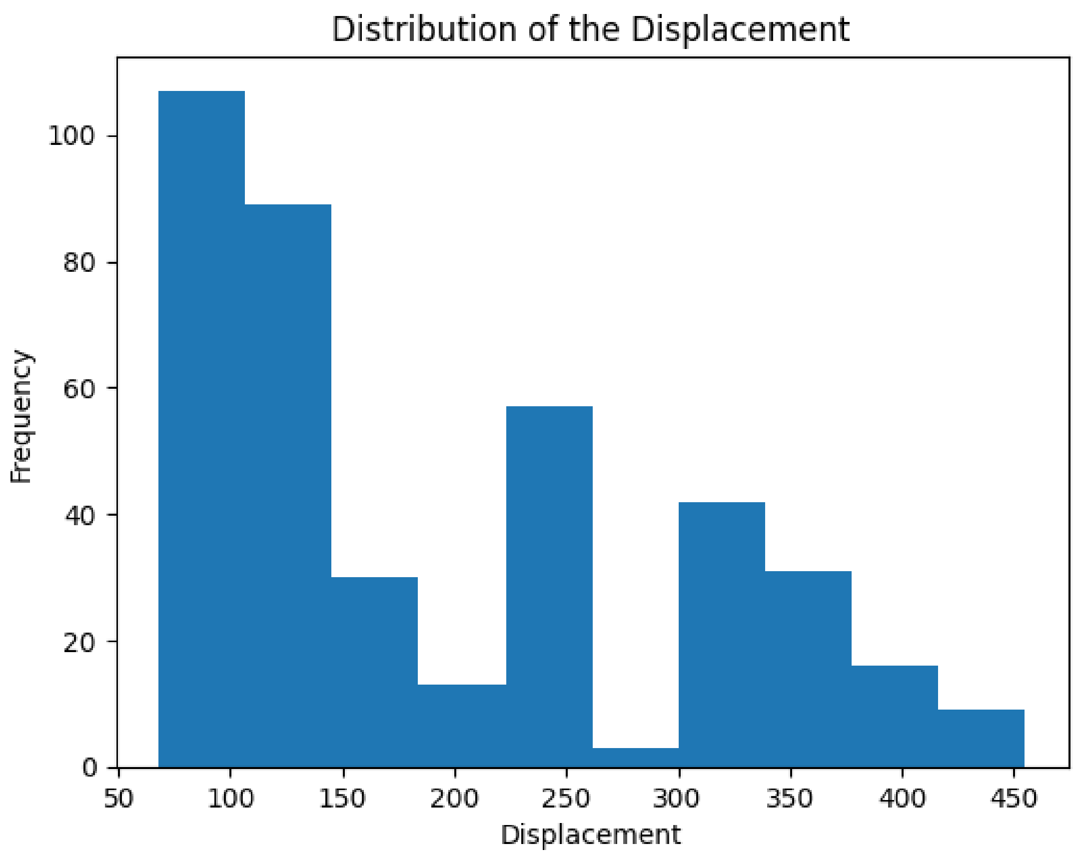



## 2 Displacement

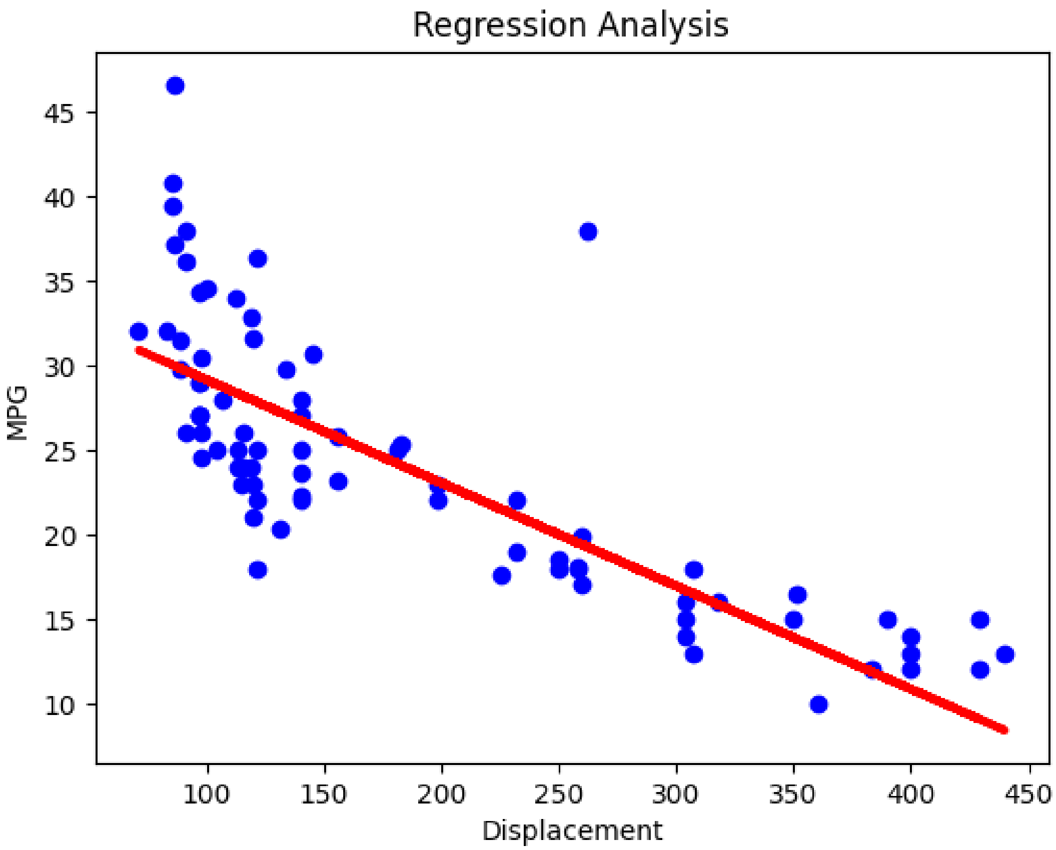
[ ]:
```
# Reshape the data
mpg = auto_df['mpg'].values.reshape(-1, 1)
displacement = auto_df['displacement'].values.reshape(-1, 1)

# 2. Data Exploration
# Explore and visualize your single feature.
plt.hist(displacement)
plt.xlabel('Displacement')
plt.ylabel('Frequency')
plt.title('Distribution of the Displacement')
plt.show()

# 3. Data Splitting
X = auto_df[['displacement']]  # Feature(s)
y = auto_df['mpg']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

```python
# 4. Model Selection
model = LinearRegression()

# 5. Model Training
model.fit(X_train, y_train)

# 6. Model Evaluation
# For regression tasks (adjust as needed):
y_pred = model.predict(X_test)

# Calculate metrics (e.g., RMSE and R-squared)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')
r2 = r2_score(y_test, y_pred)
print(f'R2 score: {r2}')

# Print the model coefficients
print('Intercept:', model.intercept_)
print('Coefficient:', model.coef_)

# 7. Visualization and Interpretation (for regression)
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=3)
plt.xlabel('Displacement')
plt.ylabel('MPG')
plt.title('Regression Analysis')
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 397 entries, 0 to 396
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           397 non-null    float64
 1   cylinders     397 non-null    int64  
 2   displacement  397 non-null    float64
 3   horsepower    397 non-null    object 
 4   weight        397 non-null    int64  
 5   acceleration  397 non-null    float64
 6   year          397 non-null    int64  
 7   origin        397 non-null    int64  
 8   name          397 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.0+ KB
None
```



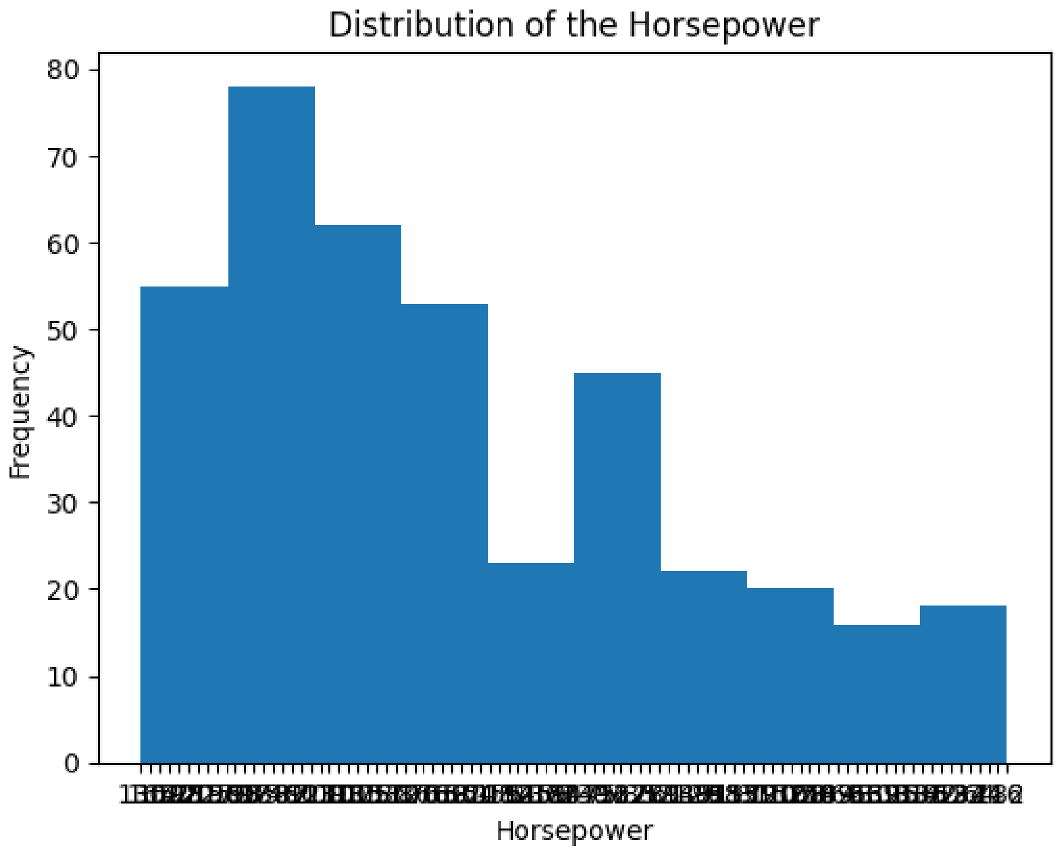

```
Mean Squared Error: 23.701871713473047
Root Mean Squared Error: 4.868456810270894
R2 score: 0.6137841415145607
Intercept: 35.23650478615161
Coefficient: [-0.06092309]
```




## 3 Horsepower

[ ]:
```python
# Reshape the data

mpg = auto_df['mpg'].values.reshape(-1, 1)
horsepower = auto_df['horsepower'].values.reshape(-1, 1)

# 2. Data Exploration
# Explore and visualize your single feature.
plt.hist(horsepower)
plt.xlabel('Horsepower')
plt.ylabel('Frequency')
plt.title('Distribution of the Horsepower')
plt.show()

# 3. Data Splitting
X = auto_df[['horsepower']]  # Feature(s)
y = auto_df['mpg']  # Target variable
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# 4. Model Selection
model = LinearRegression()

# 5. Model Training
model.fit(X_train, y_train)

# 6. Model Evaluation
# For regression tasks (adjust as needed):
y_pred = model.predict(X_test)

# Calculate metrics (e.g., RMSE and R-squared)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')
r2 = r2_score(y_test, y_pred)
print(f'R2 score: {r2}')

# Print the model coefficients
print('Intercept:', model.intercept_)
print('Coefficient:', model.coef_)



# 7. Visualization and Interpretation (for regression)
plt.scatter(X_test['horsepower'].values, y_test, color='blue')
plt.plot(X_test['horsepower'].values, y_pred, color='red', linewidth=3)
plt.xlabel('Horsepower')
plt.ylabel('MPG')
plt.title('Regression Analysis')
plt.show()
```

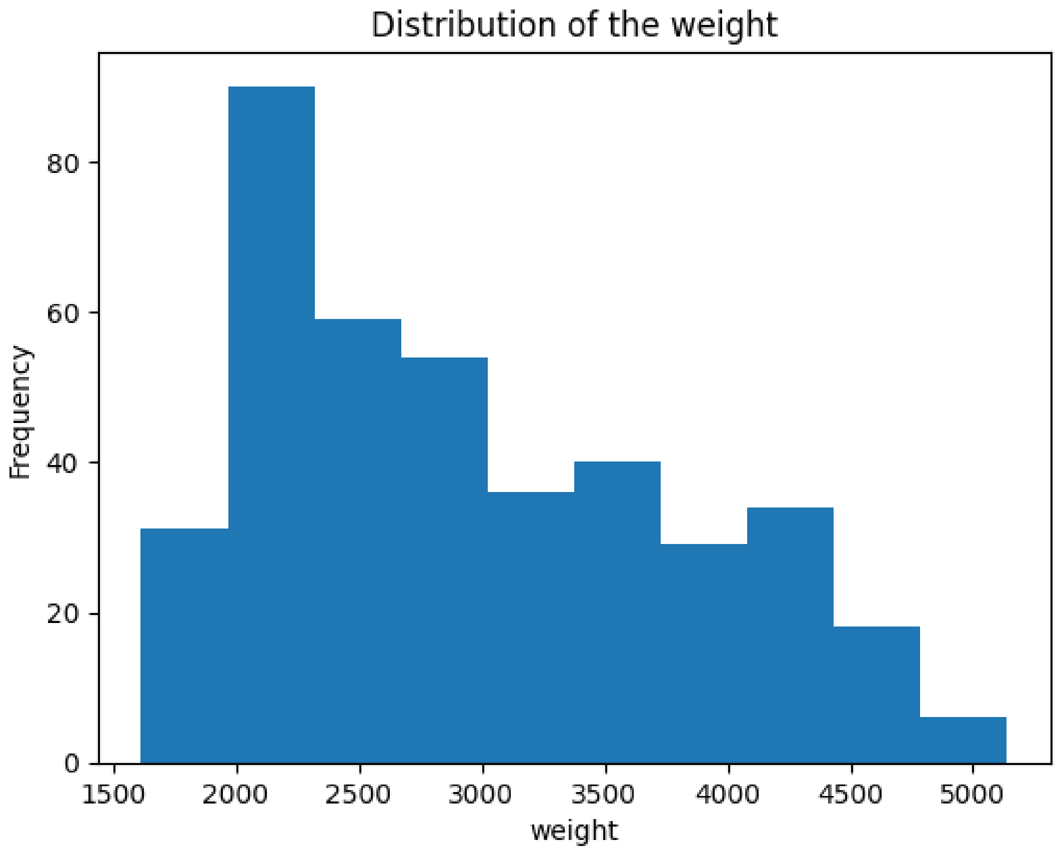



```
Mean Squared Error: 22.153237123863413
Root Mean Squared Error: 4.706722545876633
R2 score: 0.5659681822256185
Intercept: 40.606097600118346
Coefficient: [-0.16259724]
```

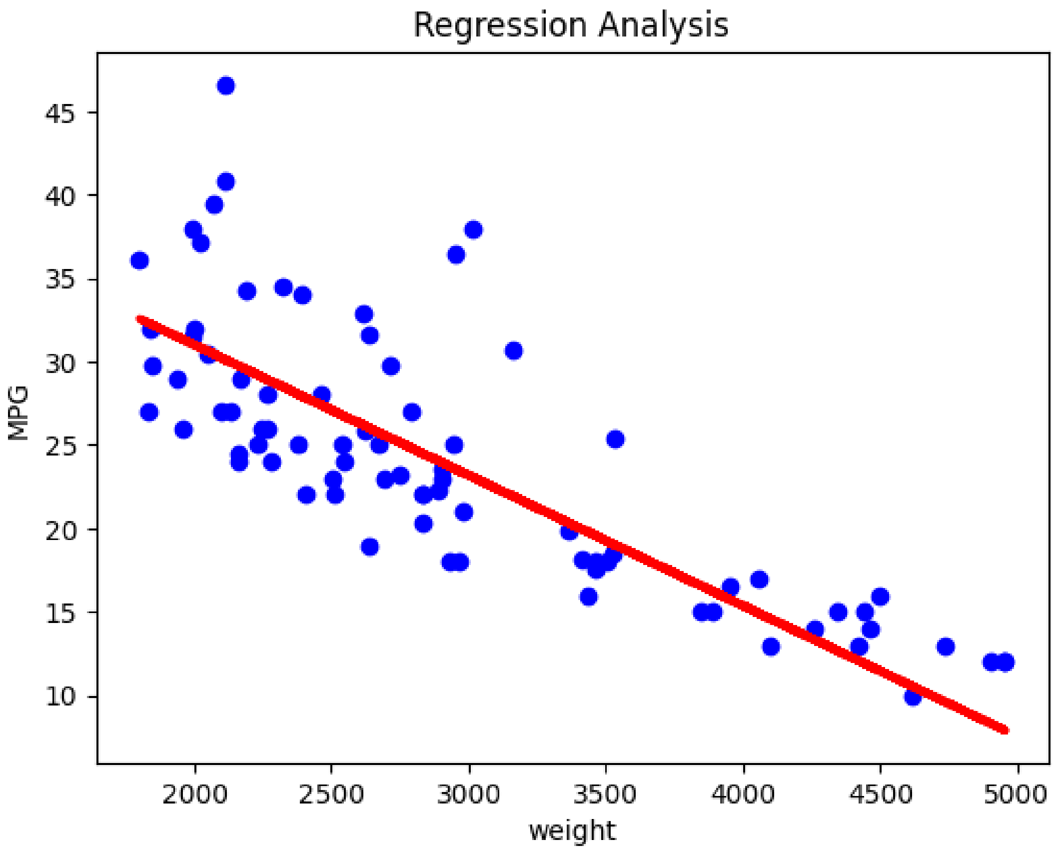

# 4 weight

```
[ ]: # Reshape the data
mpg = auto_df['mpg'].values.reshape(-1, 1)
weight = auto_df['weight'].values.reshape(-1, 1)

# 2. Data Exploration
# Explore and visualize your single feature.
plt.hist(weight)
plt.xlabel('weight')
plt.ylabel('Frequency')
plt.title('Distribution of the weight')
plt.show()

# 3. Data Splitting
X = auto_df[['weight']]  # Feature(s)
y = auto_df['mpg']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

```python
# 4. Model Selection
model = LinearRegression()

# 5. Model Training
model.fit(X_train, y_train)

# 6. Model Evaluation
# For regression tasks (adjust as needed):
y_pred = model.predict(X_test)

# Calculate metrics (e.g., RMSE and R-squared)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')
r2 = r2_score(y_test, y_pred)
print(f'R2 score: {r2}')

# Print the model coefficients
print('Intercept:', model.intercept_)
print('Coefficient:', model.coef_)

# 7. Visualization and Interpretation (for regression)
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=3)
plt.xlabel('weight')
plt.ylabel('MPG')
plt.title('Regression Analysis')
plt.show()
```



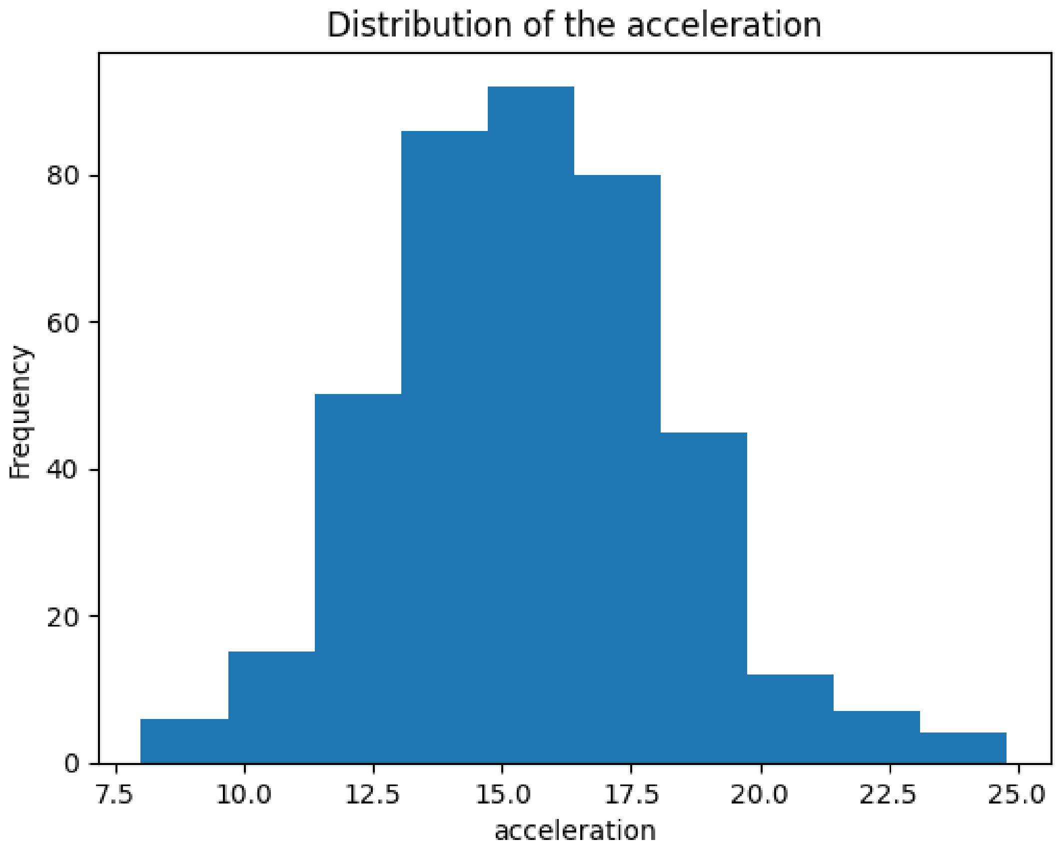

```
Mean Squared Error: 23.443428465276174
Root Mean Squared Error: 4.841841433305739
R2 score: 0.6179954072820395
Intercept: 46.64235444447036
Coefficient: [-0.00781978]
```



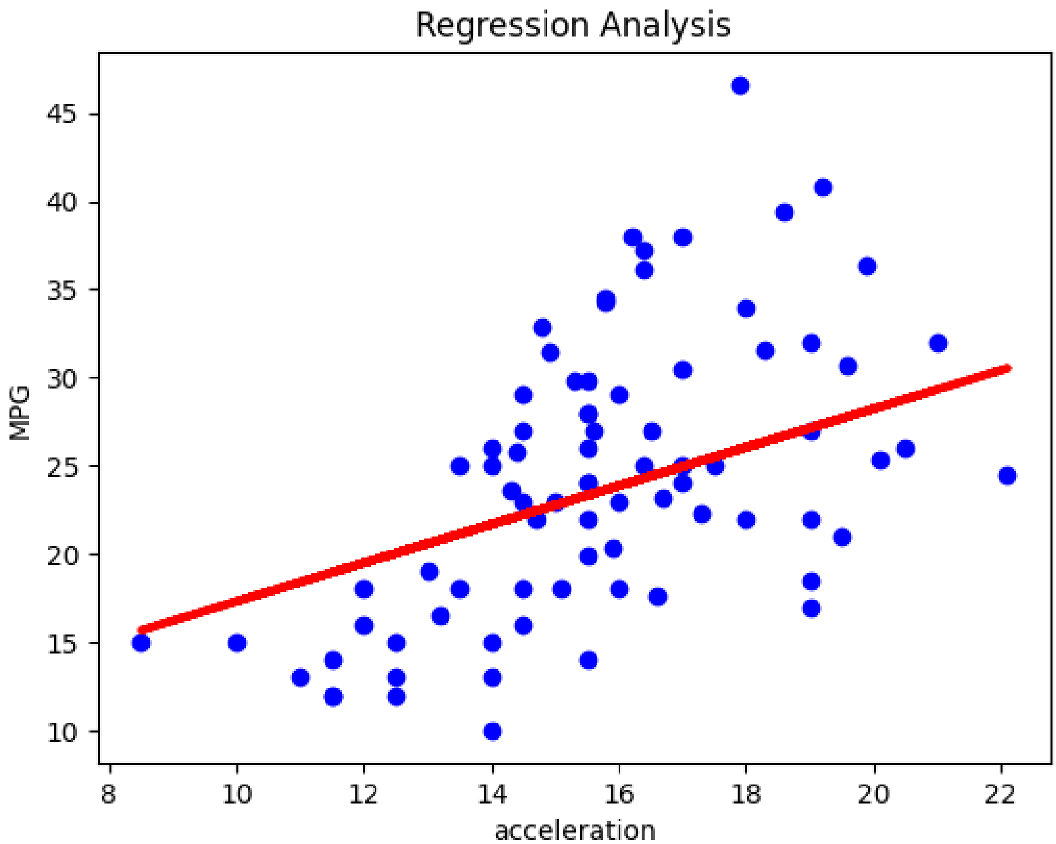

## 5 acceleration

```
[ ]: # Reshape the data
mpg = auto_df['mpg'].values.reshape(-1, 1)
acceleration = auto_df['acceleration'].values.reshape(-1, 1)

# 2. Data Exploration
# Explore and visualize your single feature.
plt.hist(acceleration)
plt.xlabel('acceleration')
plt.ylabel('Frequency')
plt.title('Distribution of the acceleration')
plt.show()

# 3. Data Splitting
X = auto_df[['acceleration']]  # Feature(s)
y = auto_df['mpg']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

```python
# 4. Model Selection
model = LinearRegression()

# 5. Model Training
model.fit(X_train, y_train)

# 6. Model Evaluation
# For regression tasks (adjust as needed):
y_pred = model.predict(X_test)

# Calculate metrics (e.g., RMSE and R-squared)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')
r2 = r2_score(y_test, y_pred)
print(f'R2 score: {r2}')

# Print the model coefficients
print('Intercept:', model.intercept_)
print('Coefficient:', model.coef_)

# 7. Visualization and Interpretation (for regression)
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=3)
plt.xlabel('acceleration')
plt.ylabel('MPG')
plt.title('Regression Analysis')
plt.show()
```



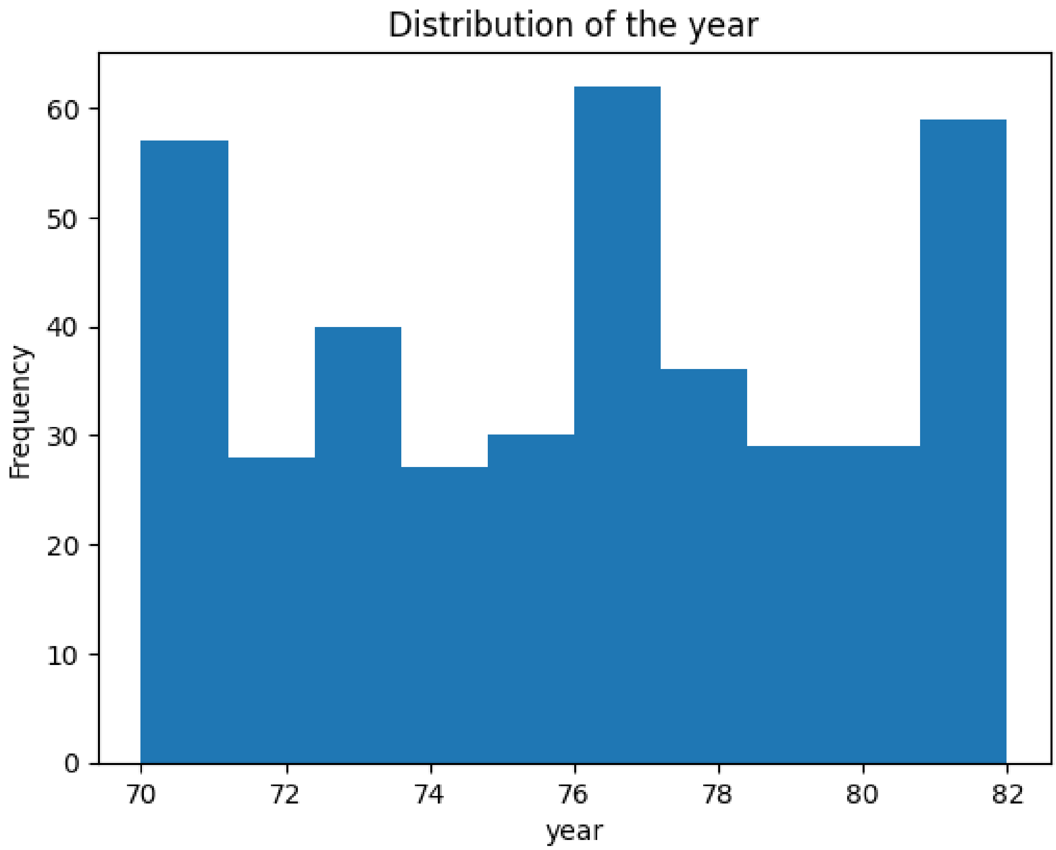

```
Mean Squared Error: 45.02835033044602
Root Mean Squared Error: 6.71031670865437
R2 score: 0.2662746980791898
Intercept: 6.3893623243885145
Coefficient: [1.09166846]
```



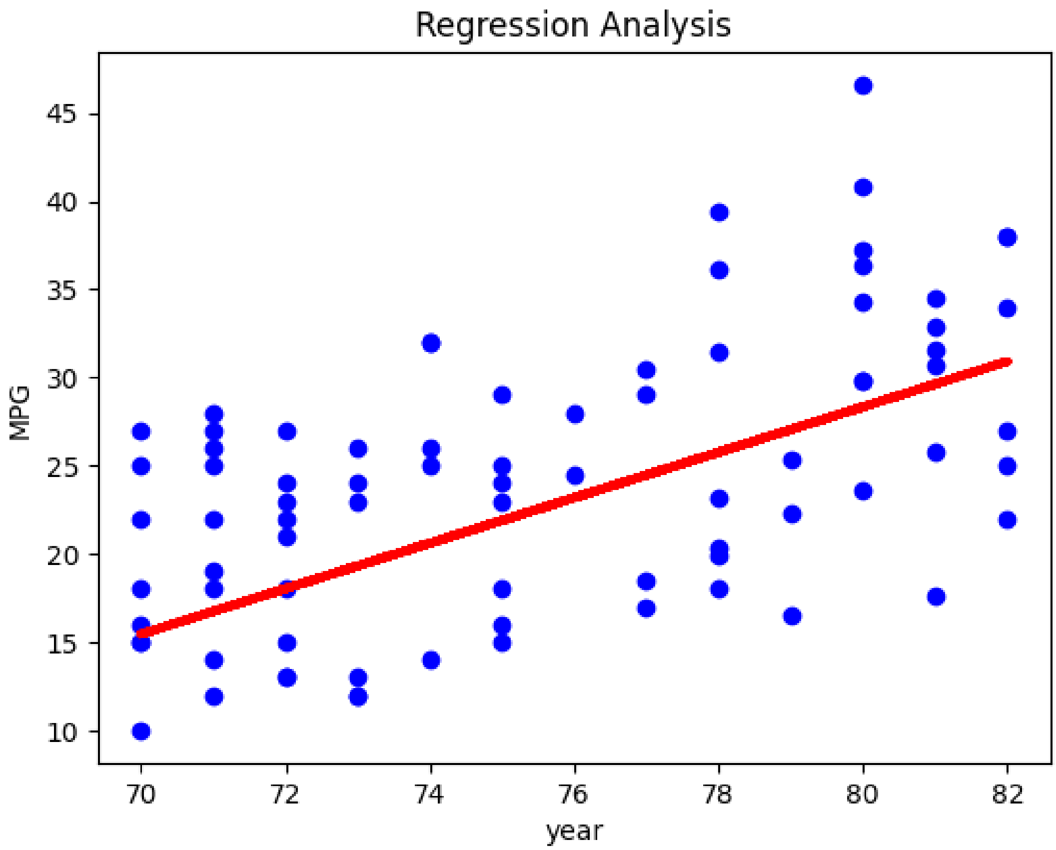

## 6 year

[ ]:
```
# Reshape the data
mpg = auto_df['mpg'].values.reshape(-1, 1)
year = auto_df['year'].values.reshape(-1, 1)

# 2. Data Exploration
# Explore and visualize your single feature.
plt.hist(year)
plt.xlabel('year')
plt.ylabel('Frequency')
plt.title('Distribution of the year')
plt.show()

# 3. Data Splitting
X = auto_df[['year']]  # Feature(s)
y = auto_df['mpg']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

```
# 4. Model Selection
model = LinearRegression()

# 5. Model Training
model.fit(X_train, y_train)

# 6. Model Evaluation
# For regression tasks (adjust as needed):
y_pred = model.predict(X_test)

# Calculate metrics (e.g., RMSE and R-squared)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')
r2 = r2_score(y_test, y_pred)
print(f'R2 score: {r2}')

# Print the model coefficients
print('Intercept:', model.intercept_)
print('Coefficient:', model.coef_)

# 7. Visualization and Interpretation (for regression)
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=3)
plt.xlabel('year')
plt.ylabel('MPG')
plt.title('Regression Analysis')
plt.show()
```



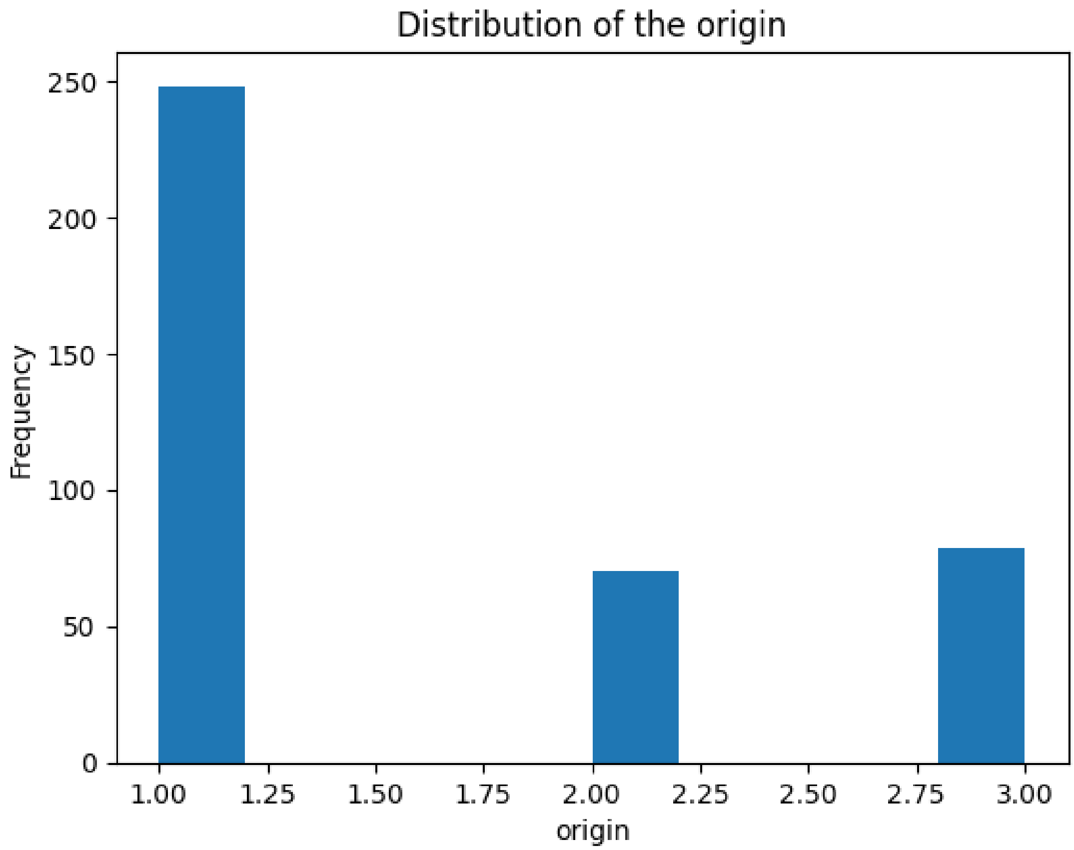

```
Mean Squared Error: 45.25485348058966
Root Mean Squared Error: 6.7271727702348825
R2 score: 0.2625838879339025
Intercept: -74.69630631241105
Coefficient: [1.28785814]
```



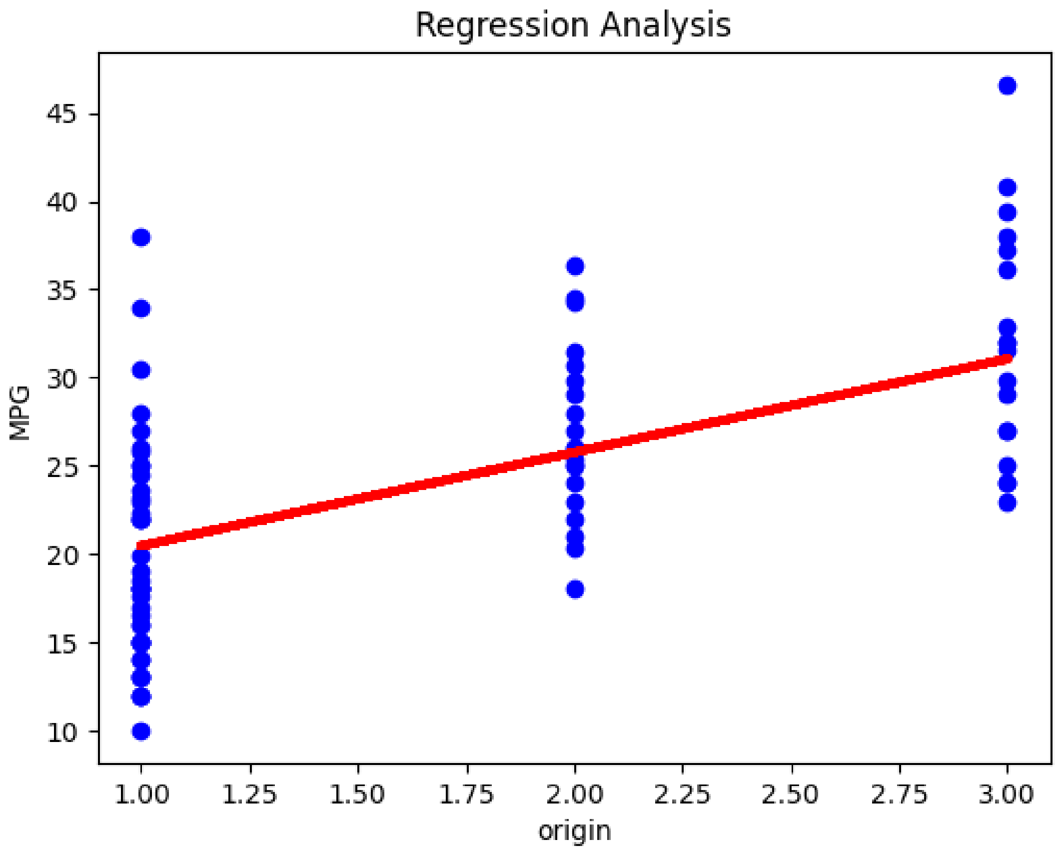

## 7 origin

```
[ ]: # Reshape the data
mpg = auto_df['mpg'].values.reshape(-1, 1)
origin = auto_df['origin'].values.reshape(-1, 1)

# 2. Data Exploration
# Explore and visualize your single feature.
plt.hist(origin)
plt.xlabel('origin')
plt.ylabel('Frequency')
plt.title('Distribution of the origin')
plt.show()

# 3. Data Splitting
X = auto_df[['origin']]  # Feature(s)
y = auto_df['mpg']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

```python
# 4. Model Selection
model = LinearRegression()

# 5. Model Training
model.fit(X_train, y_train)

# 6. Model Evaluation
# For regression tasks (adjust as needed):
y_pred = model.predict(X_test)

# Calculate metrics (e.g., RMSE and R-squared)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')
r2 = r2_score(y_test, y_pred)
print(f'R2 score: {r2}')

# Print the model coefficients
print('Intercept:', model.intercept_)
print('Coefficient:', model.coef_)

# 7. Visualization and Interpretation (for regression)
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=3)
plt.xlabel('origin')
plt.ylabel('MPG')
plt.title('Regression Analysis')
plt.show()
```




```
Mean Squared Error: 36.215950984956486
Root Mean Squared Error: 6.017969008308076
R2 score: 0.40987046214705836
Intercept: 15.16051340899802
Coefficient: [5.29676329]
```



Regression Analysis
MPG
45
40
35
30
25
20
15
10
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
origin