```python
# 🔹 Step 1: Install & Import Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files


# 🔹 Step 2: Upload Dataset Manually
print("Please upload your Automobile.csv file")
uploaded = files.upload()

# 🔹 Step 3: Load the Dataset
df = pd.read_csv("Automobile.csv")
```

```
Please upload your Automobile.csv file
```

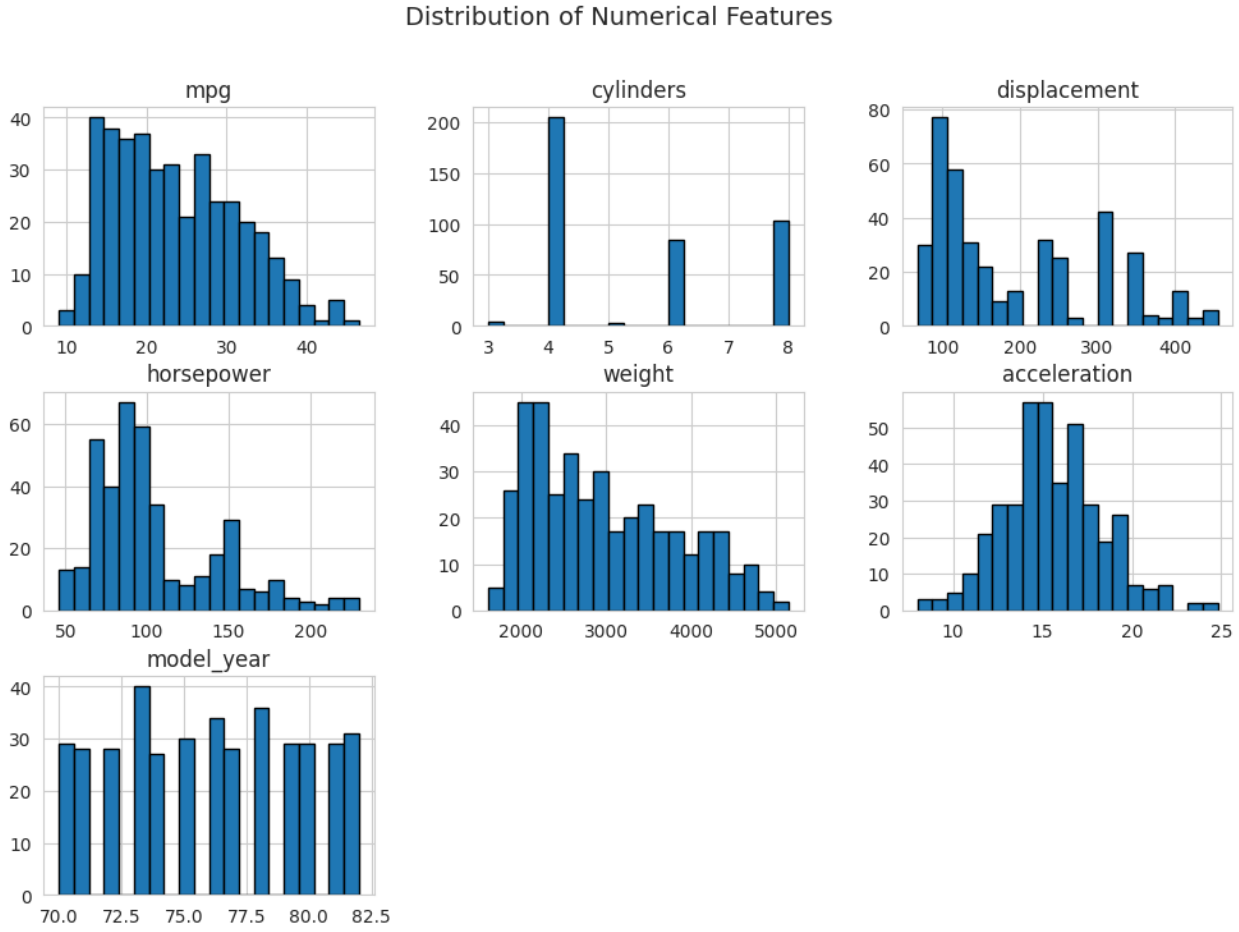
```
<IPython.core.display.HTML object>
```

```
Saving Automobile.csv to Automobile (5).csv
```

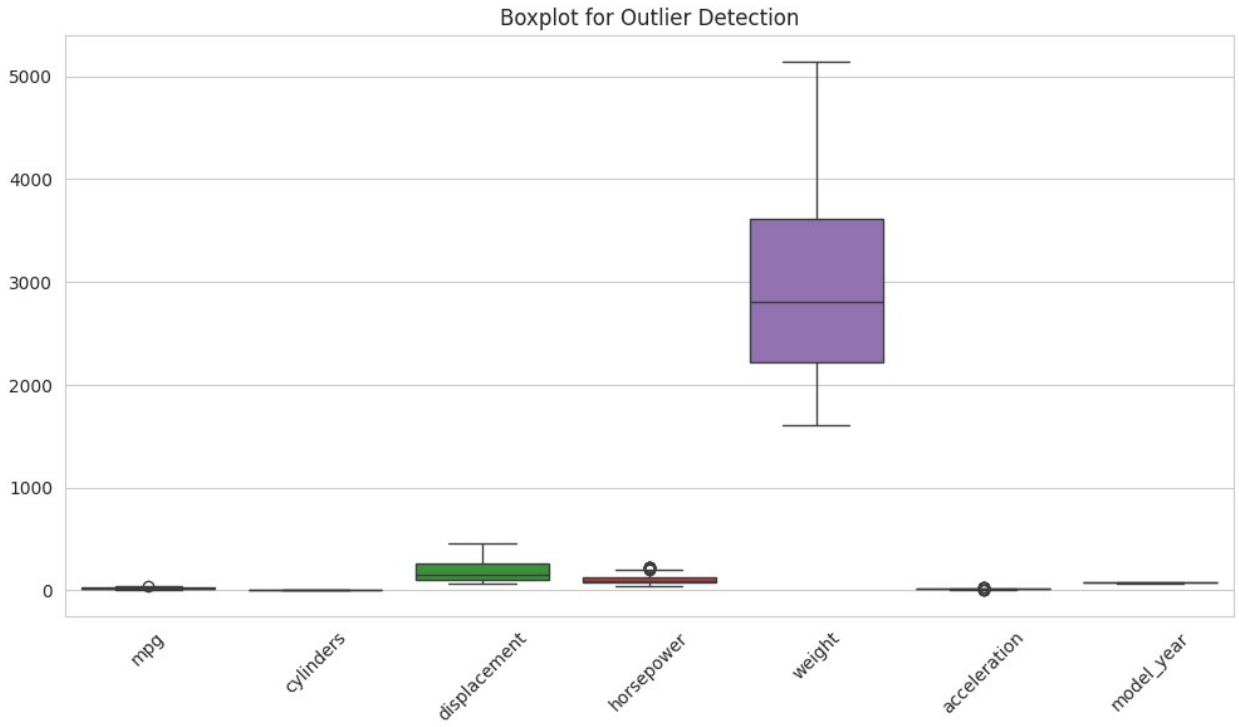
```python
# 🔹 Step 4: Display Dataset Info & First Few Rows
print("\nDataset Info:")
print(df.info())
print("\nFirst 5 Rows:")
print(df.head())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   name          398 non-null    object
 1   mpg           398 non-null    float64
 2   cylinders     398 non-null    int64
 3   displacement  398 non-null    float64
 4   horsepower    392 non-null    float64
 5   weight        398 non-null    int64
 6   acceleration  398 non-null    float64
 7   model_year    398 non-null    int64
 8   origin        398 non-null    object
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
None

First 5 Rows:
                       name   mpg  cylinders  displacement
horsepower  \
0  chevrolet chevelle malibu  18.0          8         307.0
```

```
130.0
1           buick skylark 320  15.0          8         350.0
165.0
2          plymouth satellite  18.0          8         318.0
150.0
3              amc rebel sst  16.0          8         304.0
150.0
4                ford torino  17.0          8         302.0
140.0

   weight  acceleration  model_year origin
0    3504          12.0          70    usa
1    3693          11.5          70    usa
2    3436          11.0          70    usa
3    3433          12.0          70    usa
4    3449          10.5          70    usa

# □ Step 5: Handle Missing Values (Fill with Median)
df.fillna(df.median(numeric_only=True), inplace=True)


# □ Step 6: Set Seaborn Style for Better Visualization
sns.set_style("whitegrid")

# □ Step 7: Histogram of Numerical Features
df.hist(figsize=(12, 8), bins=20, edgecolor="black")
plt.suptitle("Distribution of Numerical Features", fontsize=14)
plt.show()
```



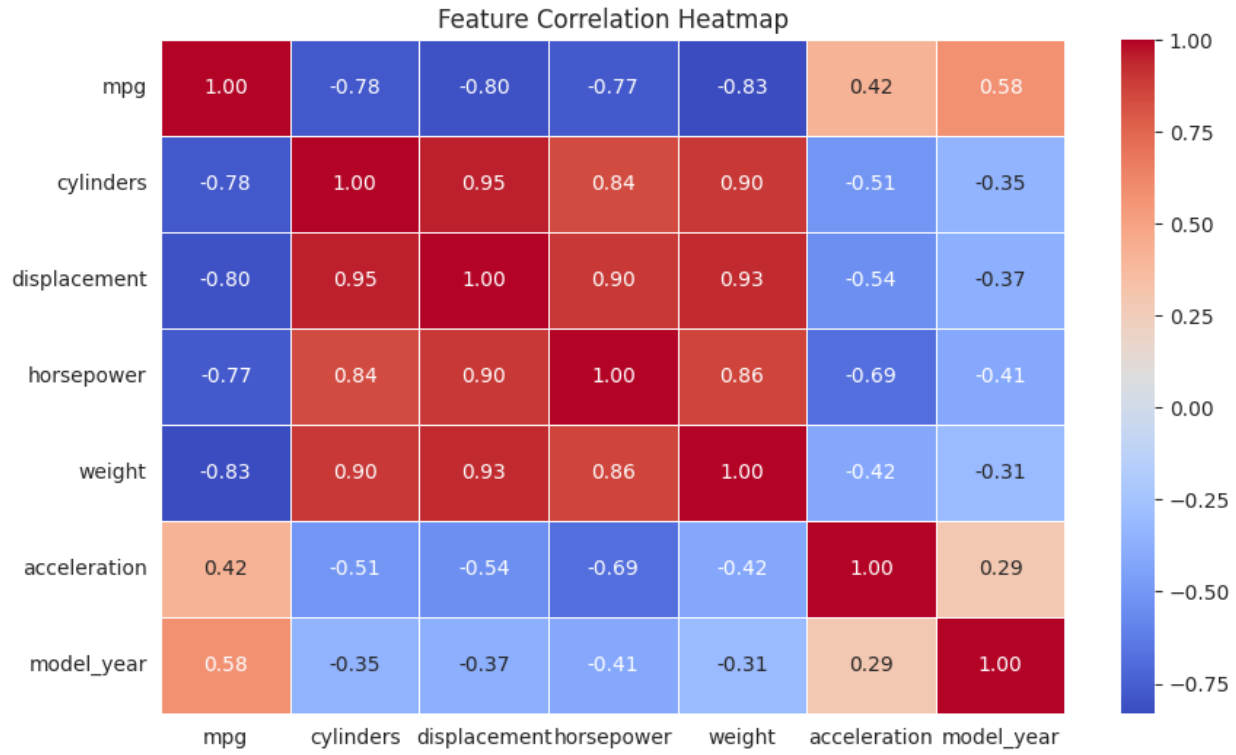

```
# 📦 Step 8: Boxplot to Detect Outliers
plt.figure(figsize=(12, 6))
sns.boxplot(data=df.select_dtypes(include=["float64", "int64"]))
plt.title("Boxplot for Outlier Detection")
plt.xticks(rotation=45)
plt.show()
```



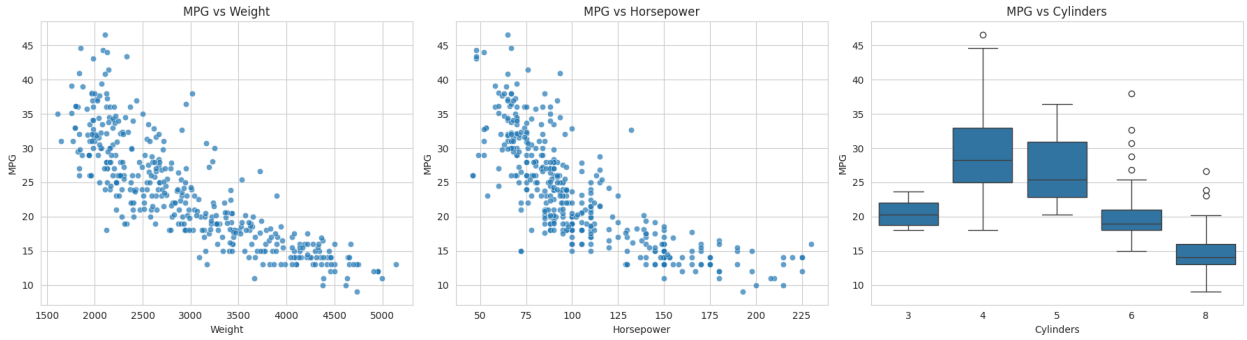
The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
# □ Step 9: Correlation Heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="coolwarm",
fmt=".2f", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```




```
# 🔟 Step 10: Scatter Plots for Relationship Analysis
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

# 🚗 MPG vs Weight
sns.scatterplot(x=df["weight"], y=df["mpg"], alpha=0.7, ax=axes[0])
axes[0].set_title("MPG vs Weight")
axes[0].set_xlabel("Weight")
axes[0].set_ylabel("MPG")

# 🚗 MPG vs Horsepower
sns.scatterplot(x=df["horsepower"], y=df["mpg"], alpha=0.7,
ax=axes[1])
axes[1].set_title("MPG vs Horsepower")
axes[1].set_xlabel("Horsepower")
axes[1].set_ylabel("MPG")

# 🚗 MPG vs Cylinders
sns.boxplot(x=df["cylinders"], y=df["mpg"], ax=axes[2])
axes[2].set_title("MPG vs Cylinders")
axes[2].set_xlabel("Cylinders")
axes[2].set_ylabel("MPG")

plt.tight_layout()
plt.show()
```




```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

```
86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

```
seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

```
604800
```

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To learn more, see Overview of Colab. To create a new Colab notebook you can use the File menu above, or use the following link: create a new Colab notebook.

Colab notebooks are Jupyter notebooks that are hosted by Colab. To learn more about the Jupyter project, see jupyter.org.

With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualize it. To edit the code, just click the cell and start editing.

You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from Github and many other sources. To learn more about importing data, and how Colab can be used for data science, see the links below under Working with Data.

```
import numpy as np
import IPython.display as display
```

```
from matplotlib import pyplot as plt
import io
import base64

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

fig = plt.figure(figsize=(4, 3), facecolor='w')
plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g',
alpha=0.6)
plt.title("Sample Visualization", fontsize=10)

data = io.BytesIO()
plt.savefig(data)
image = F"
{base64.b64encode(data.getvalue()).decode()}"
alt = "Sample Visualization"
display.display(display.Markdown(F"""![{alt}]({image})"""))
plt.close(fig)
```

```
<IPython.core.display.Markdown object>
```

Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

For example, if you find yourself waiting for **pandas** code to finish running and want to go faster, you can switch to a GPU Runtime and use libraries like RAPIDS cuDF that provide zero-code-change acceleration.

To learn more about accelerating pandas on Colab, see the 10 minute guide or US stock market data analysis demo.

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code.

Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow
- Developing and training neural networks
- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

To see sample Colab notebooks that demonstrate machine learning applications, see the machine learning examples below.

- Overview of Colab
- Guide to Markdown
- Importing libraries and installing dependencies

- Saving and loading notebooks in GitHub
- Interactive forms
- Interactive widgets
- Loading data: Drive, Sheets, and Google Cloud Storage
- Charts: visualizing data
- Getting started with BigQuery

Machine Learning Crash Course

These are a few of the notebooks from Google's online Machine Learning course. See the full course website for more.

- Intro to Pandas DataFrame
- Intro to RAPIDS cuDF to accelerate pandas
- Linear regression with tf.keras using synthetic data
- TensorFlow with GPUs
- TensorFlow with TPUs
- Retraining an Image Classifier: Build a Keras model on top of a pre-trained image classifier to distinguish flowers.
- Text Classification: Classify IMDB movie reviews as either *positive* or *negative*.
- Style Transfer: Use deep learning to transfer style between images.
- Multilingual Universal Sentence Encoder Q&A: Use a machine learning model to answer questions from the SQuAD dataset.
- Video Interpolation: Predict what happened in a video between the first and the last frame.