Assignment 1: Exploratory Data Analysis (EDA) and Data Transformation

Yu-Chen Su

1. Download the automotive dataset, Auto-3.csv

   Import pandas to read file and show data head

```
[3]: import pandas as pd
     df = pd.read_csv('Auto-3.csv')
     df.head()
```

[3]:

	Make	Model	Type	Origin	DriveTrain	MSRP	EngineSize	Cylinders	Horsepower	MPG_Highway	Weight	Length
0	Acura	MDX	SUV	Asia	All	36945	3.5	6.0	265	23	4451	189
1	Acura	RSX Type S 2dr	Sedan	Asia	Front	23820	2.0	4.0	200	31	2778	172
2	Acura	TSX 4dr	Sedan	Asia	Front	26990	2.4	4.0	200	29	3230	183
3	Acura	TL 4dr	Sedan	Asia	Front	33195	3.2	6.0	270	28	3575	186
4	Acura	3.5 RL 4dr	Sedan	Asia	Front	43755	3.5	6.0	225	24	3880	197

2. Determine the data dimensionality by finding the following (5pts):

   A. Total number of vehicles.

```
[5]: # Display the count
     num_rows = len(df)
     print(f"Total number of rows: {num_rows}")
```

```
Total number of rows: 428
```

   B. Number of attributes (categories).

```
[7]: # Display the column (categories)
     num_column = df.shape[1]
     print(f"Total number of categories: {num_column}")
```

```
Total number of categories: 12
```

   C. Data types.

```
[11]: # Check data types of each column
      print(df.dtypes)
```

```
Make            object
Model           object
Type            object
Origin          object
DriveTrain      object
MSRP             int64
EngineSize     float64
Cylinders      float64
Horsepower       int64
MPG_Highway      int64
Weight           int64
Length           int64
dtype: object
```

D. Missing values.

```
[13]: missing_values = df.isnull().sum()
      print(missing_values)
```

```
Make          0
Model         0
Type          0
Origin        0
DriveTrain    0
MSRP          0
EngineSize    0
Cylinders     2
Horsepower    0
MPG_Highway   0
Weight        0
Length        0
dtype: int64
```

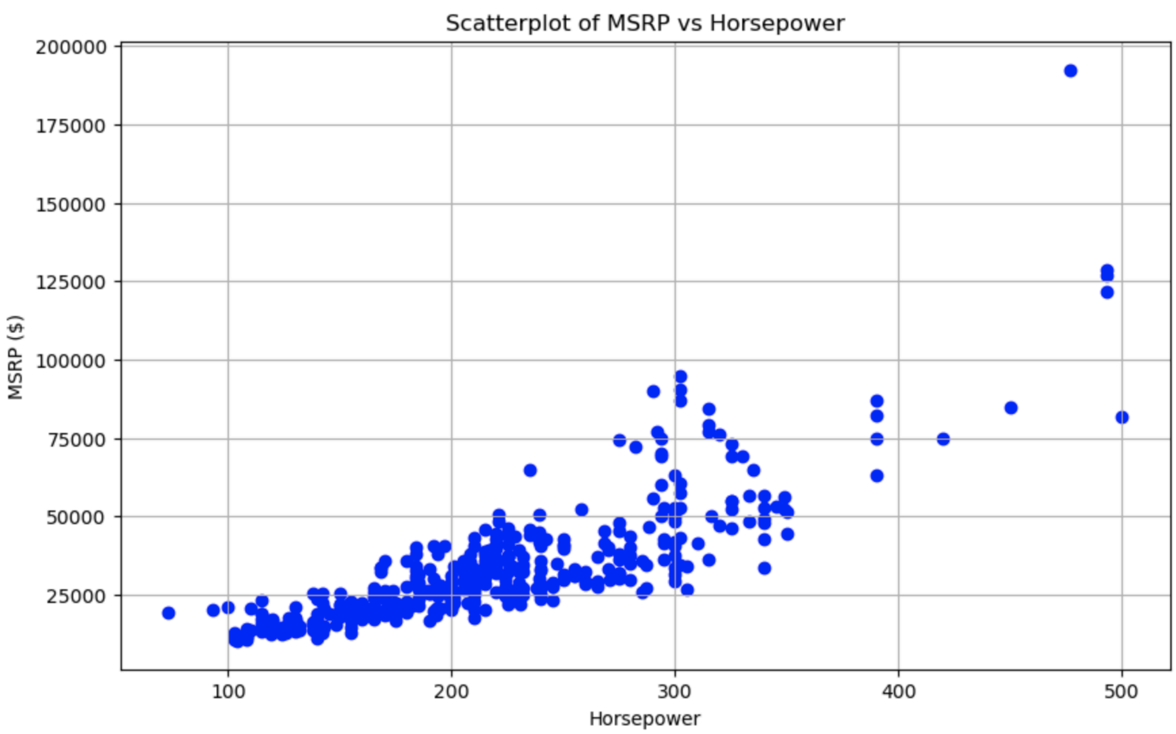
Find two missing values in Cylinders by checking null data

3. Visualize the data by plotting the following (5pts):
   A. Scatterplot of MSRP (Y) and Horsepower (X).

```python
import matplotlib.pyplot as plt
# Create a scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(df['Horsepower'], df['MSRP'], color='blue')

# Add titles and labels
plt.title('Scatterplot of MSRP vs Horsepower')
plt.xlabel('Horsepower')
plt.ylabel('MSRP ($)')
plt.grid(True)

# Show the plot
plt.show()
```



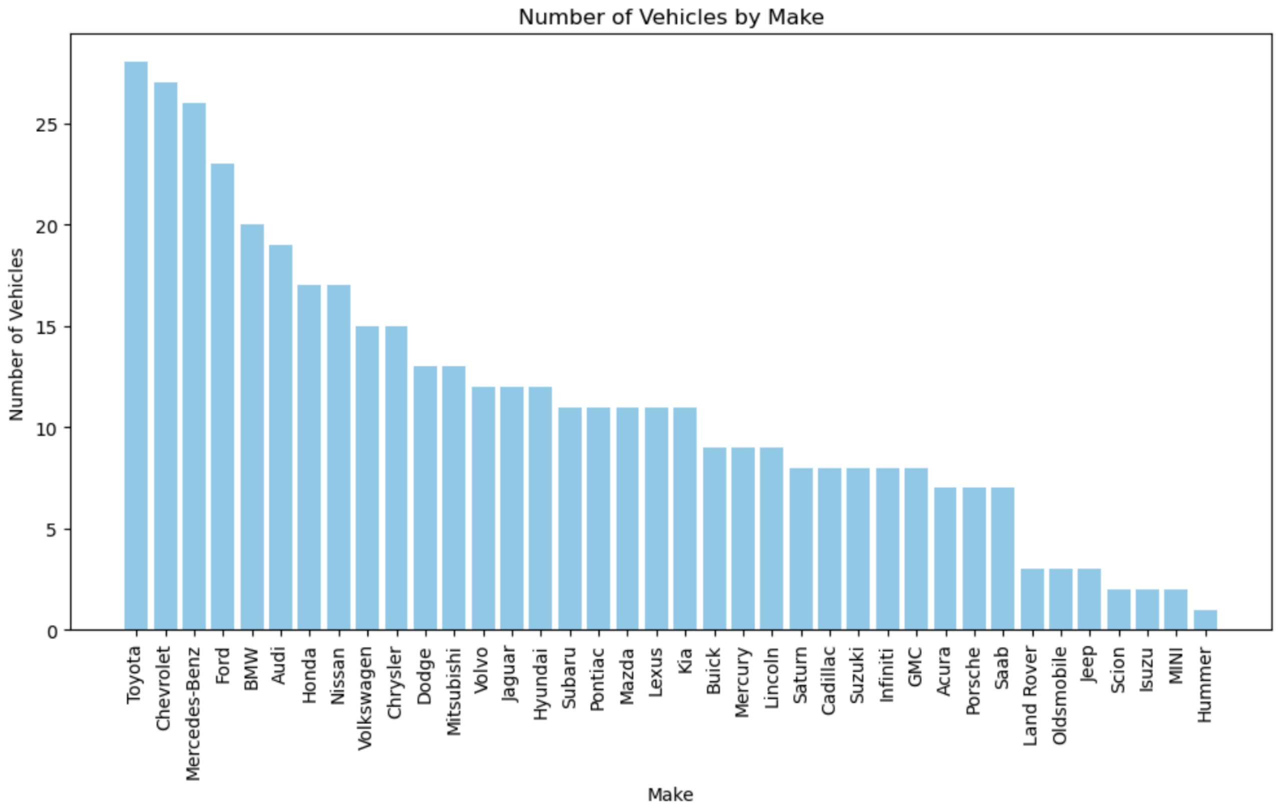
B. Bar plot of Number of vehicles (Y) and Make (X).

```
import matplotlib.pyplot as plt2
# Count the number of vehicles for each make
make_counts = df['Make'].value_counts()

# Create a bar plot
plt2.figure(figsize=(12, 6))
plt2.bar(make_counts.index, make_counts.values, color='skyblue')

# Add titles and labels
plt2.title('Number of Vehicles by Make')
plt2.xlabel('Make')
plt2.ylabel('Number of Vehicles')
plt2.xticks(rotation=90)  # Rotate x labels if they overlap

# Show the plot
plt2.show()
```



C. Answer the following based on the bar plot: Which Make has the greatest number of vehicles?

```
count = df['Make'].str.count('Toyota').sum()
print(f"Toyota number of vehicles:{count}")
```

```
Toyota number of vehicles:28
```

Based on the bar plot, Toyota has the highest number of vehicles, with a total of 28 cars.

4. Normalize and Standardize the Horsepower variable. Make sure the entire results print out. Answer the following questions (5pts):
   A. How do the transformation results differ?

```python
# Normalize the Horsepower column
from sklearn.preprocessing import MinMaxScaler
# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Normalize the 'Horsepower' column
df['Horsepower_Normalized'] = scaler.fit_transform(df[['Horsepower']])

# Print the entire DataFrame
print(df)
```

Normalization

```python
from sklearn.preprocessing import StandardScaler
# Initialize the StandardScaler
scaler = StandardScaler()

# Standardize the 'Horsepower' column
df['Horsepower_Standardized'] = scaler.fit_transform(df[['Horsepower']])

# Print the entire DataFrame
print(df)
```

Standardization

```python
# Print specific columns
columns_to_print = ['Horsepower','Horsepower_Normalized', 'Horsepower_Standardized']
print(df[columns_to_print])
```

```
     Horsepower  Horsepower_Normalized  Horsepower_Standardized
0           265               0.449649                 0.684503
1           200               0.297424                -0.221395
2           200               0.297424                -0.221395
3           270               0.461358                 0.754187
4           225               0.355972                 0.127028
..          ...                    ...                      ...
423         197               0.290398                -0.263205
424         242               0.395785                 0.363955
425         268               0.456674                 0.726313
426         170               0.227166                -0.639501
427         208               0.316159                -0.109899

[428 rows x 3 columns]
```

Show the result

   B. Which transformation method do you prefer and why?
   I prefer normalization in this case because it offers clear interpretability within a bounded range, scaling all values between 0 and 1. While some might argue that normalization is vulnerable to outliers, this dataset falls within an acceptable range, making normalization not only suitable but also an effective method for presenting the data.

   (Comparison between normalized & standardized horsepower are shown below)

```python
# Plot normalized Horsepower
sns.histplot(df['Horsepower_Normalized'], kde=True, color='green', label='Normalized Horsepower', bins=5, alpha=0.5)

# Plot standardized Horsepower
sns.histplot(df['Horsepower_Standardized'], kde=True, color='red', label='Standardized Horsepower', bins=5, alpha=0.5)

# Add titles and labels
plt.title('Comparison of Horsepower Transformations')
plt.xlabel('Horsepower')
plt.ylabel('Frequency')
plt.legend()

# Adjust layout
plt.tight_layout()
plt.show()
```