Sri Lanka Institute of Information Technology

Kandy Uni

IT2011 - Artificial Intelligence and Machine Learning

Lab Submission

Worksheet No: 03

<IT 24610823>

<Shaahidh M.W.M>

<Batch 02>

<2025-08-19>

01

```
[7]  import pandas as pd
     import numpy as np
     import seaborn as sns                #visualisation
     import matplotlib.pyplot as plt      #visualisation
     %matplotlib inline
     sns.set(color_codes = True)
```

02

```
[9]  from google.colab import drive
     drive.mount('/content/drive')

     Mounted at /content/drive

[40] df = pd.read_csv("/content/drive/MyDrive/cars_data.csv")
```

03

```
#displaying first 05 rows
display(df.head())
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSRP
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	3916	46135
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	3916	40650
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3916	36350
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3916	29450
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	3916	34500

```
[50] #displaying last 05 rows
     df.tail()
```

	Make	Model	Year	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	highway MPG	city mpg	MSRP
11909	Acura	ZDX	2012	300.0	6.0	AUTOMATIC	all wheel drive	23	16	46120
11910	Acura	ZDX	2012	300.0	6.0	AUTOMATIC	all wheel drive	23	16	56670
11911	Acura	ZDX	2012	300.0	6.0	AUTOMATIC	all wheel drive	23	16	50620
11912	Acura	ZDX	2013	300.0	6.0	AUTOMATIC	all wheel drive	23	16	50920
11913	Lincoln	Zephyr	2006	221.0	6.0	AUTOMATIC	front wheel drive	26	17	28995

04

```
#Check data type
df.dtypes
```

	0
Make	object
Model	object
Year	int64
Engine HP	float64
Engine Cylinders	float64
Transmission Type	object
Driven_Wheels	object
highway MPG	int64
city mpg	int64
MSRP	int64

dtype: object

05

```
#Remove the irrelevent columns
df=df.drop(['Engine Fuel Type','Market Category','Vehicle Style','Popularity','Number of Doors','Vehicle Size'],axis=1)
df.head(5)
```

06

```
#Renaming the Column
df = df.rename(columns={"Engine HP": "HP", "Engine Cylinders": "Total Cylinders", "Transmission Type": "Mechanism", "Driven_Wheels": "Drive Mode", "Highway MPG": "MPG-H", "

df.head(5)
```

	Make	Model	Year	HP	Total Cylinders	Mechanism	Drive Mode	highway MPG	city mpg	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

07

```
#Check for Duplicate Rows

df.shape      #Check the Dimension
df.count()    #Used to count the number of rows
duplicate_rows_df = df[df.duplicated()]   #Find the duplicate rows
print("number of duplicate rows: ", duplicate_rows_df.shape)

df = df.drop_duplicates()
df.head(5)
```

```
number of duplicate rows:  (989, 10)
```

	Make	Model	Year	HP	Total Cylinders	Mechanism	Drive Mode	highway MPG	city mpg	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

08

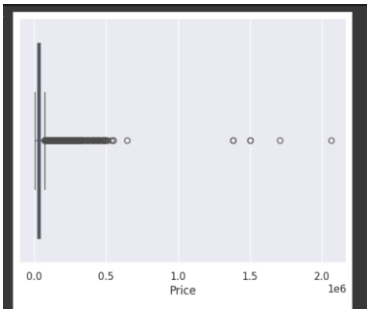
```
#Drop the MIssing or Null Values
print(df.isnull().sum())

df = df.dropna()    #Dropping the missing values
df.count()
```

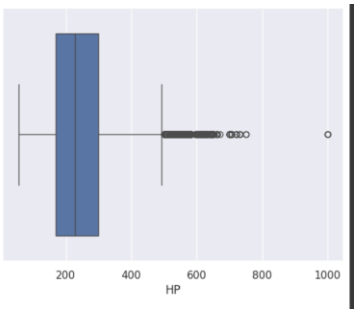
```
Make                0
Model               0
Year                0
Engine HP          69
Engine Cylinders   30
Transmission Type   0
Driven_Wheels       0
highway MPG         0
city mpg            0
MSRP                0
dtype: int64
```

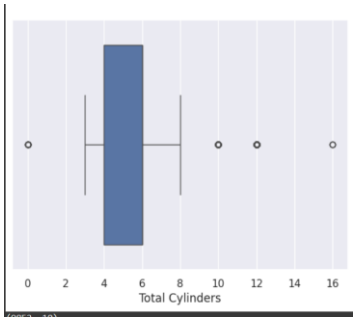
	0
Make	11816
Model	11816
Year	11816
Engine HP	11816
Engine Cylinders	11816
Transmission Type	11816
Driven_Wheels	11816
highway MPG	11816
city mpg	11816
MSRP	11816

09

```python
#Outliers Detection
sns.boxplot(x = df['Price'])
plt.show()

sns.boxplot(x = df['HP'])
plt.show()

sns.boxplot(x = df['Total Cylinders'])
plt.show()

# Step 1: Identify numeric columns
numeric_cols = df.select_dtypes(include='number').columns

# Step 2: Compute Q1, Q3, and IQR for only numeric columns
Q1 = df[numeric_cols].quantile(0.25)
Q3 = df[numeric_cols].quantile(0.75)
IQR = Q3 - Q1

# Step 3: Filter rows that are NOT outliers in any numeric column
# Keep only rows where all numeric values are within the IQR bounds
condition = ~((df[numeric_cols] < (Q1 - 1.5 * IQR)) | (df[numeric_cols] > (Q3 + 1.5 * IQR))).any(axis=1)

# Step 4: Apply this condition to the full DataFrame (all columns)
df_cleaned = df[condition]

# Step 5: Check the new shape
print(df_cleaned.shape)
```







10

```python
#Scatter Plot
fig, ax=plt.subplots (figsize=(10,6))
ax.scatter(df['HP'], df ['Price'])
ax.set_xlabel('HP')
ax.set_ylabel('Price')
plt.show()

#Histogram
df.Make.value_counts().nlargest (40).plot(kind='bar', figsize=(10,5))
plt.title("Number of cars Vs Make")
plt.ylabel('Number of cars')
plt.xlabel('Make');
plt.show()

#Heat Maps
# First, filter only numeric columns for correlation
numeric_df=df.select_dtypes (include=['float64', 'int64'])
plt.figure(figsize=(10,5))
c = numeric_df.corr() # Calculate correlation only on numeric columns
sns.heatmap(c, cmap="BrBG", annot=True)
# Uncomment to display the plot
plt.show()
```