EDA project of Cars Dataset:

It is one of the primary steps in the journey towards data science, and specifically associated with preparing the data set for the further analysis and machine learning modelling. It has been an issue of performing EDA via spreadsheets due to limitations towards process oversized data and to solve such problems there are certain open source tools and softwares available market wide to bridge this gap. Here, python is the commonly expected solving machine for EDA.

The EDA concept speaks about understanding the data, cleaning the data and analyzing relations among variables with the dataset. Performing EDA has seven steps involved: 1. variable identification, 2.Univariate Analysis, 3.Bivariate Analysis, 4.Outlier Treatment, 5.Missing Value Treatment, 6. Variable Transformation, 7. Variable Creation. Post performing these line of operations, our data set looks fair for further analysis using machine learning models.

The actual use of these seven steps will be explained later after below example. Here below in first example we are considering the example of car data set borrowed from the Kaggle website which makes explanation easier (https://www.kaggle.com/datasets/CooperUnion/cardataset).

Example 1: EDA on car data set.

Step 1: Import the following python libraries:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import kagglehub

# Download latest version
path = kagglehub.dataset_download("CooperUnion/cardataset")

print("Path to dataset files:", path)
df=pd.read_csv(f"/Users/divyadeepverma/.cache/kagglehub/datasets/CooperUnion/
cardataset/versions/1/data.csv")
print(df.head(5))
```

Be mindful when running the above code because there might be some python packages that you need to install before making port 80 requests because this is the modern way of getting dataset bypassing the manual download process. In this case I had to install urllib3 and requests as additional libraries in order to get the following data as a result of last statement in the above code:

```
  Make       Model  Year  ... city mpg  Popularity  MSRP
0  BMW  1 Series M  2011  ...       19        3916  46135
1  BMW    1 Series  2011  ...       19        3916  40650
2  BMW    1 Series  2011  ...       20        3916  36350
3  BMW    1 Series  2011  ...       18        3916  29450
4  BMW    1 Series  2011  ...       18        3916  34500

[5 rows x 16 columns]
```

Similarly, you can try print(df.tail(5)) to get the last 5 rows of the dataset.

Step2: This step involves checking the type of the data, for which following code is required:

print(df.dtype)

This gives the list of attributes and associated data types of elements. Now it becomes a business requirement to find out the variables required and which are not required. Below is the list of elements with their data types:

```
Make                 object
Model                object
Year                 int64
Engine Fuel Type     object
Engine HP            float64
Engine Cylinders     float64
Transmission Type    object
Driven_Wheels        object
Number of Doors      float64
Market Category      object
Vehicle Size         object
Vehicle Style        object
highway MPG          int64
city mpg             int64
Popularity           int64
MSRP                 int64
dtype: object
```

Step3: This involves the deleting the columns which are not required. Columns such as engine type fule, market category, vehicle style, popularity, number of doors, vehicle size are not required in this analysis,
so we have drop those out.

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

Step 4: This step involves renaming of the column as per business requirements. Following is the code for same:

```
df=df.rename(columns={"Engine HP": "HP", "Engine Cylinders": "Cylinders",
"Transmission Type": "Transmission", "Driven_Wheels": "Drive Mode","highway MPG":
"MPG-H", "city mpg": "MPG-C", "MSRP": "Price" })
print(df.head(5))
```

Step 5: Until now, the metadata operations are concluded, next is the stepwise work on the actual data. Now we have delete the duplicate data.

```
print(df.shape)
```

(11914, 10)

```
duplicate_rows_df=df[df.duplicated()]
print("no of duplicate rows: ", duplicate_rows_df)
print(df.count())
df=df.drop_duplicates()
print(df.head(5))
print(df.count())
```

The above code show the difference in count of data points as below:

```
Make            10925
Model           10925

Year            10925

HP              10856

Cylinders       10895

Transmission    10925

Drive Mode      10925

MPG-H           10925

MPG-C           10925

Price           10925

dtype: int64
```

Here you will observe that duplicate rows are eliminated, but you will observe some difference in counts which is due to the NULL values across dataset.

Step 6: In this step below we are going to replace null values from the above count to the average of the values. However, it is forcible method because we have to consider the decision based on descriptive statistics.

```
print(df.isnull().sum())
df=df.dropna() #dropping the null values
print(df.count())
```

Now we get the following output of the attribute counts:

```
Make            10827
Model           10827

Year            10827

HP              10827

Cylinders       10827

Transmission    10827

Drive Mode      10827

MPG-H           10827

MPG-C           10827

Price           10827

dtype: int64
```

You can observe that all the counts of each column are uniform now

```
print(df.isnull().sum()) #again checking for null values
```

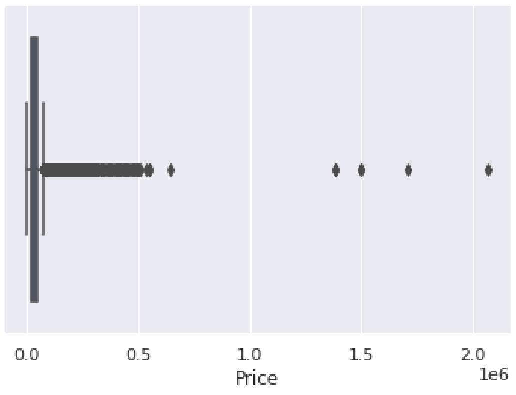
The above code gives all the counts as zero, means there is no null or duplicate values in the dataset.

Step 7: This step involves detecting and deleting the outliers in the above dataset. This is subjective to the situation of the business where the analysts decide the degree of confidence. Usually, there are two ways to remove outliers- one is using z-score while the other by finding inter quartile range (IQR). For the sake of simplicity, we will move forward using IQR method, however, code for outlier treatment using z-score will also be provided because it resonates as more professional style of outlier treatment.

First detect the outlier of each numeric attributes in consideration using box plot method for the columns- MSRP, Cylinders, HorsePower, and EngineSize

```
sns.boxplot(x=df['Price'])
```

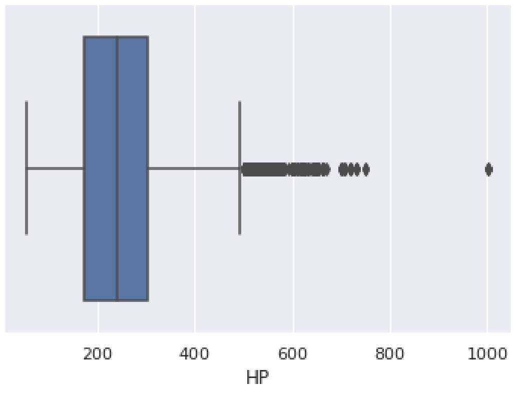
Below is the output we get to visualize outliers:



The same we can perform for Horse Power using the following code:

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

Below is the visualization output of the Horse Power dataset:



Considering the IQR method, we are below calculating the IQR over the range of data set.

```
q1=df.quantile(0.25)
q2=df.quantile(0.75)
iqr=q3-q1
print(iqr)
```

Output:

```
Year              9.0
HP              130.0

Cylinders         2.0

MPG-H             8.0

MPG-C             6.0

Price         21327.5

dtype: float64
```

Below is the main code line to clearly segregate the outliers using the boolean expression code:

```
df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
```

Upon finishing the segregation over dataset, find out the shape of the dataset for more clarity resulting from above code:

```
print(df.shape)
```

now you got the results without outlier and the numbers are as follows:

```
(9191, 10)
```

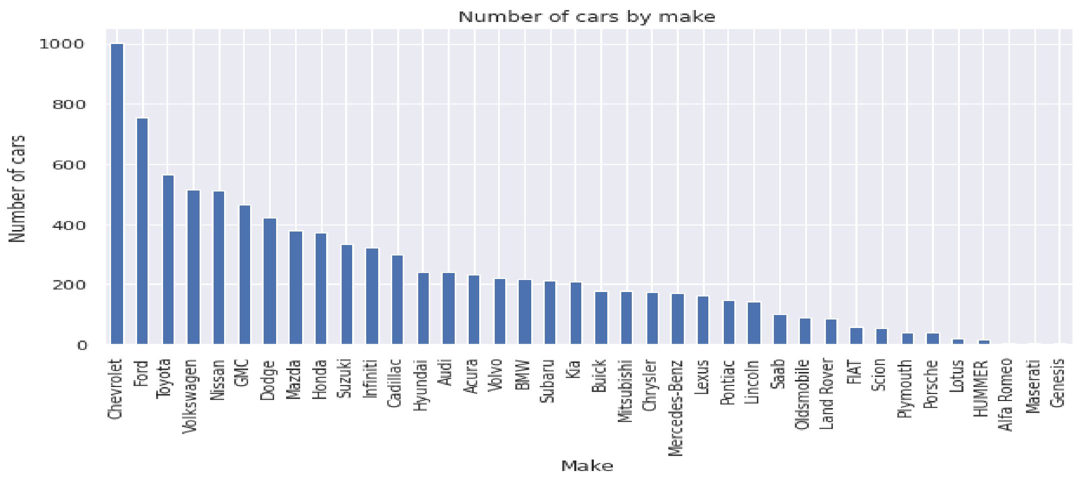
Step 8:

This step helps in generating the relationship plots like scatter plot and histogram plot to analyze the relationships among variables:

Histogram plot explains the relationship among variables. For example in this example finding out the relationship between the ‘no of cars’ and ‘make’ of cars in this data set

Below is the python code for histogram plotting with the plot below:

```
df.Make.value_counts().nlargest(40).plot(kind='bar', figsize=(10,5))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make');
```

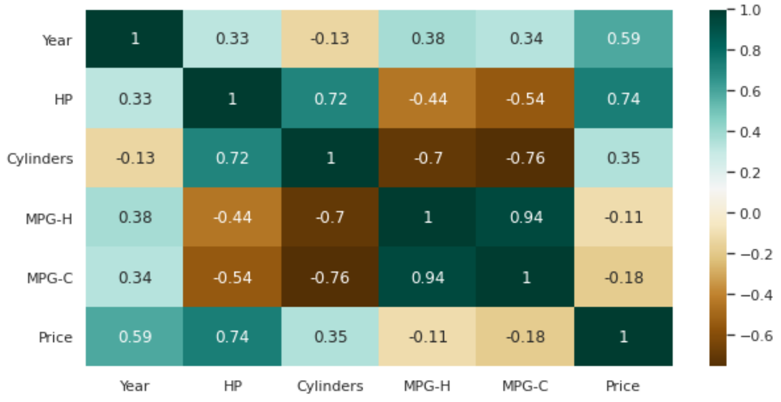


Next is the heat maps which suggests the correlation among the variables and also helps to find out the dependent variable. In this code, we can understand from the heat map that that the price of the car depends on HP, Cylinders and Engine Size.

Code:

```
plt.figure(figsize=(10,5))
c= df.corr()
sns.heatmap(c,cmap="BrBG",annot=True)
print(c)
```

Output:

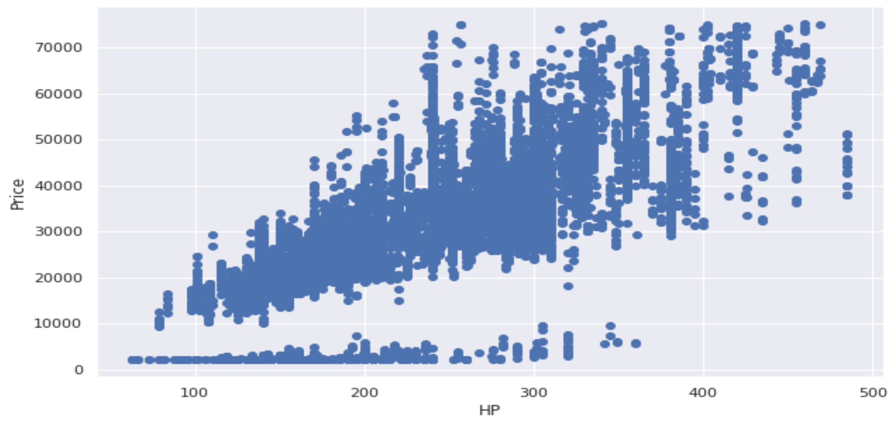


	Year	HP	Cylinders	MPG-H	MPG-C	Price
Year	1	0.33	-0.13	0.38	0.34	0.59
HP	0.33	1	0.72	-0.44	-0.54	0.74
Cylinders	-0.13	0.72	1	-0.7	-0.76	0.35
MPG-H	0.38	-0.44	-0.7	1	0.94	-0.11
MPG-C	0.34	-0.54	-0.76	0.94	1	-0.18
Price	0.59	0.74	0.35	-0.11	-0.18	1

Another style of plot is scattered plot for which the code and output is shown below:

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

Output:



The complete code for the example above is given below:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import kagglehub

# Download latest version
path = kagglehub.dataset_download("CooperUnion/cardataset")

# adding this to control timeout session
import requests

# Example with a longer timeout setting, this is optional
response = requests.get("https://path_to_your_resource", timeout=60)

print("Path to dataset files:", path)

df=pd.read_csv(f"/Users/divyadeepverma/.cache/kagglehub/datasets/CooperUnion/
cardataset/versions/1/data.csv")
#print(df.head(5))
#print (df.dtypes)

# dropping the elements which are not required:
df=df.drop(['Engine Fuel Type', 'Market Category', 'Vehicle Style', 'Popularity',
'Number of Doors', 'Vehicle Size'], axis=1)
#print(df.head(5))

#renaming of columns
df=df.rename(columns={"Engine HP": "HP", "Engine Cylinders": "Cylinders",
"Transmission Type": "Transmission", "Driven_Wheels": "Drive Mode","highway MPG":
"MPG-H", "city mpg": "MPG-C", "MSRP": "Price" })
#print(df.head(5))

#deleting duplicate data
print(df.shape)
duplicate_rows_df=df[df.duplicated()]
print("no of duplicate rows: ", duplicate_rows_df)
print(df.count())
df=df.drop_duplicates()
print(df.head(5))
print(df.count())

#deleting null values

print(df.isnull().sum())
df=df.dropna() #dropping the null values
print(df.count())
print(df.isnull().sum()) #again checking for null values

# detecting and deleting outliers
sns.boxplot(x=df['Price'])
sns.boxplot(x=df['HP'])
sns.boxplot(x=df['Cylinders'])

q1=df.quantile(0.25)
q2=df.quantile(0.75)
```

```
iqr=q3-q1
print(iqr)

df = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
print(df.shape)

# histogram plot for variable relationship

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


#heat map
plt.figure(figsize=(10,5))
c= df.corr()
sns.heatmap(c,cmap="BrBG",annot=True)
print(c)

#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()
```