Data Wrangling I

Perform the following operations using Python on any open source dataset (e.g., data.csv)

1. Import all the required Python Libraries.
2. Locate an open source data from the web (e.g., https://www.kaggle.com). Provide a clear description of the data and its source (i.e., URL of the web site).
3. Load the Dataset into pandas dataframe.
4. Data Preprocessing: check for missing values in the data using pandas isnull(), describe() function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame.
5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions.
6. Turn categorical variables into quantitative variables in Python. In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set.

1.1 import all the required Pyhton Libraries

```
import pandas as pd
import matplotlib.pylab as plt
import numpy as np
```

Locate an open source data from the web (e.g., https://www.kaggle.com). Provide a clear description of the data and its source (i.e., URL of the web site).

```
url_link="https://raw.githubusercontent.com/rohinidevkar/DSBDA/main/autodata.csv"
df = pd.read_csv(url_link)
```

```
df.head(10)
```

	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	e lo
0	0	3	122	alfa-romero	std	two	convertible	rwd	
1	1	3	122	alfa-romero	std	two	convertible	rwd	
2	2	1	122	alfa-romero	std	two	hatchback	rwd	
3	3	2	164	audi	std	four	sedan	fwd	
4	4	2	164	audi	std	four	sedan	4wd	

```
df.tail()
```

	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engi locat
196	196	-1	95	volvo	std	four	sedan	rwd	f
197	197	-1	95	volvo	turbo	four	sedan	rwd	f
198	198	-1	95	volvo	std	four	sedan	rwd	f
199	199	-1	95	volvo	turbo	four	sedan	rwd	f
200	200	-1	95	volvo	turbo	four	sedan	rwd	f

5 rows × 30 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 30 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         201 non-null    int64  
 1   symboling          201 non-null    int64  
 2   normalized-losses  201 non-null    int64  
 3   make               201 non-null    object 
 4   aspiration         201 non-null    object 
 5   num-of-doors       201 non-null    object 
 6   body-style         201 non-null    object 
 7   drive-wheels       201 non-null    object 
 8   engine-location    201 non-null    object 
 9   wheel-base         201 non-null    float64
 10  length             201 non-null    float64
 11  width              201 non-null    float64
 12  height             201 non-null    float64
 13  curb-weight        201 non-null    int64  
 14  engine-type        201 non-null    object 
 15  num-of-cylinders   201 non-null    object 
 16  engine-size        201 non-null    int64
 17  fuel-system        201 non-null    object
 18  bore               201 non-null    float64
 19  stroke             197 non-null    float64
 20  compression-ratio  201 non-null    float64
 21  horsepower         199 non-null    float64
 22  peak-rpm           199 non-null    float64
 23  city-mpg           201 non-null    int64
 24  highway-mpg        201 non-null    int64
 25  price              201 non-null    float64
 26  city-L/100km       201 non-null    float64
 27  horsepower-binned  199 non-null    object
 28  diesel             201 non-null    int64
 29  gas                201 non-null    int64
dtypes: float64(11), int64(9), object(10)
memory usage: 47.2+ KB
```

```
df.describe()
```

	Unnamed: 0	symboling	normalized-losses	wheel-base	length	width	hei
count	201.000000	201.000000	201.00000	201.000000	201.000000	201.000000	201.000
mean	100.000000	0.840796	122.00000	98.797015	0.837102	0.915126	53.766
std	58.167861	1.254802	31.99625	6.066366	0.059213	0.029187	2.447
min	0.000000	-2.000000	65.00000	86.600000	0.678039	0.837500	47.800
25%	50.000000	0.000000	101.00000	94.500000	0.801538	0.890278	52.000
50%	100.000000	1.000000	122.00000	97.000000	0.832292	0.909722	54.100
75%	150.000000	2.000000	137.00000	102.400000	0.881788	0.925000	55.500
max	200.000000	3.000000	256.00000	120.900000	1.000000	1.000000	59.800

```
df.isnull()
```

	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engi locat
0	False	False	False	False	False	False	False	False	Fa
1	False	False	False	False	False	False	False	False	Fa
2	False	False	False	False	False	False	False	False	Fa

```
df.isnull().sum()
```

```
Unnamed: 0           0
symboling            0
normalized-losses    0
make                 0
aspiration           0
num-of-doors         0
body-style           0
drive-wheels         0
engine-location      0
wheel-base           0
length               0
width                0
height               0
curb-weight          0
engine-type          0
num-of-cylinders     0
engine-size          0
fuel-system          0
bore                 0
stroke               4
compression-ratio    0
horsepower           2
peak-rpm             2
city-mpg             0
highway-mpg          0
price                0
city-L/100km         0
horsepower-binned    2
diesel               0
gas                  0
dtype: int64
```

```
df.notnull()
```

	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	engi locati
0	True	True	True	True	True	True	True	True	T
1	True	True	True	True	True	True	True	True	T
2	True	True	True	True	True	True	True	True	T
3	True	True	True	True	True	True	True	True	T
4	True	True	True	True	True	True	True	True	T
...	...	...	...	...	...	...	...	...	

```
df.notnull().sum()
```

```
Unnamed: 0           201
symboling            201
normalized-losses    201
make                 201
aspiration           201
num-of-doors         201
body-style           201
drive-wheels         201
engine-location      201
wheel-base           201
length               201
width                201
height               201
curb-weight          201
engine-type          201
num-of-cylinders     201
engine-size          201
fuel-system          201
bore                 201
stroke               197
compression-ratio    201
horsepower           199
peak-rpm             199
city-mpg             201
highway-mpg          201
price                201
city-L/100km         201
horsepower-binned    199
diesel               201
gas                  201
dtype: int64
```

```
#caLculate the mean value for "stroke" column
avg_stroke = df["stroke"].astype("float").mean(axis = 0)
print("Average of stroke :",avg_stroke)

#replace NaN by mean value in "stroke" column
df["stroke"].replace(np.nan, avg_stroke,inplace = True)
```

```
Average of stroke : 3.2569035532994857
```

Calculate the mean value for the 'horsepower' column :

```
avg_hp=df["horsepower"].astype("float").mean(axis = 0)
print("Average of stroke :",avg_hp)
```

```
Average of stroke : 103.39698492462311
```

```
df['horsepower'].replace(np.nan,avg_hp,inplace = True)
```

```
from contextlib import nullcontext
df['num-of-doors'].value_counts()
```

```
four    115
two      86
Name: num-of-doors, dtype: int64
```

```
df['num-of-doors'].value_counts().idxmax()
```

```
'four'
```

```
# replace the missing 'num-of-door' values by most frequent
df['num-of-doors'].replace(np.nan, "four" , inplace=True)

#simply drop whole row with nan in "Horsepower-banned" column
df.dropna(subset=['horsepower-binned'], axis=0 , inplace=True)

#reset index, because we dropped two rows
df.reset_index(drop=True, inplace=True)
```

```
df.isnull().sum()
```

```
Unnamed: 0           0
symboling            0
normalized-losses    0
make                 0
aspiration           0
num-of-doors         0
body-style           0
drive-wheels         0
engine-location      0
wheel-base           0
length               0
width                0
height               0
curb-weight          0
engine-type          0
num-of-cylinders     0
engine-size          0
fuel-system          0
bore                 0
stroke               0
compression-ratio    0
```

```
horsepower          0
peak-rpm            2
city-mpg            0
highway-mpg         0
price               0
city-L/100km        0
horsepower-binned   2
diesel              0
gas                 0
dtype: int64
```

DATA STANDARDIZATION : It is process of transforming data into common format which allows the researcher to make meaningful comparision

```
df['city-L/100km']=235/df['city-mpg']
df.head()
```

	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	e lo
0	0	3	122	alfa-romero	std	two	convertible	rwd	
1	1	3	122	alfa-romero	std	two	convertible	rwd	
2	2	1	122	alfa-romero	std	two	hatchback	rwd	
3	3	2	164	audi	std	four	sedan	fwd	
4	4	2	164	audi	std	four	sedan	4wd	

5 rows × 30 columns

```
df['highway-L/100km']=235/df["highway-mpg"]
df.head()
```

	Unnamed: 0	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels	e lo
0	0	3	122	alfa-romero	std	two	convertible	rwd	
1	1	3	122	alfa-	std	two	convertible	rwd	
2	2	1	122	alfa-	std	two	hatchback	rwd	

DATA NORMALIZATION : It is process of transforming several values into similar range

```
df['length']=df['length']/df['length'].max()
df['width']=df['width']/df['width'].max()
df['height']=df['height']/df['height'].max()
```

```
df[['length','width','height']].head()
```

	length	width	height
0	0.811148	0.890278	0.816054
1	0.811148	0.890278	0.816054
2	0.822681	0.909722	0.876254
3	0.848630	0.919444	0.908027
4	0.848630	0.922222	0.908027

INDIACTOR VARIABLE : Indicator variable or dummy variable are used to label numerical variable used to label categories

```
df.columns
```

```
Index(['Unnamed: 0', 'symboling', 'normalized-losses', 'make', 'aspiration',
       'num-of-doors', 'body-style', 'drive-wheels', 'engine-location',
       'wheel-base', 'length', 'width', 'height', 'curb-weight', 'engine-type',
       'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke',
       'compression-ratio', 'horsepower', 'peak-rpm', 'city-mpg',
       'highway-mpg', 'price', 'city-L/100km', 'horsepower-binned', 'diesel',
       'gas', 'highway-L/100km'],
      dtype='object')
```

```
df['aspiration'].value_counts()
```

```
std      163
turbo     36
Name: aspiration, dtype: int64
```

```
dummy_var_1=pd.get_dummies(df['aspiration'])
dummy_var_1.head()
```

	std	turbo
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0

```
df=pd.concat([df,dummy_var_1], axis=1)
df.drop('aspiration',axis = 1 , inplace = True)
```

```
df.head()
```

	Unnamed: 0	symboling	normalized-losses	make	num-of-doors	body-style	drive-wheels	engine-location	wheel-ba
0	0	3	122	alfa-romero	two	convertible	rwd	front	88
1	1	3	122	alfa-romero	two	convertible	rwd	front	88
2	2	1	122	alfa-romero	two	hatchback	rwd	front	94
3	3	2	164	audi	four	sedan	fwd	front	99
4	4	2	164	audi	four	sedan	4wd	front	99

5 rows × 52 columns

The last columns are indicator variable which are represented by 0's and 1's

```
df.columns
```

```
Index(['Unnamed: 0', 'symboling', 'normalized-losses', 'make', 'num-of-doors',
       'body-style', 'drive-wheels', 'engine-location', 'wheel-base', 'length',
       'width', 'height', 'curb-weight', 'engine-type', 'num-of-cylinders',
       'engine-size', 'fuel-system', 'bore', 'stroke', 'compression-ratio',
       'horsepower', 'peak-rpm', 'city-mpg', 'highway-mpg', 'price',
       'city-L/100km', 'horsepower-binned', 'diesel', 'gas', 'highway-L/100km',
       'std', 'turbo', 'std', 'turbo', 'std', 'turbo', 'std', 'turbo', 'std',
       'turbo', 'std', 'turbo', 'std', 'turbo', 'std', 'turbo', 'std', 'turbo',
       'std', 'turbo', 'std', 'turbo'],
      dtype='object')
```

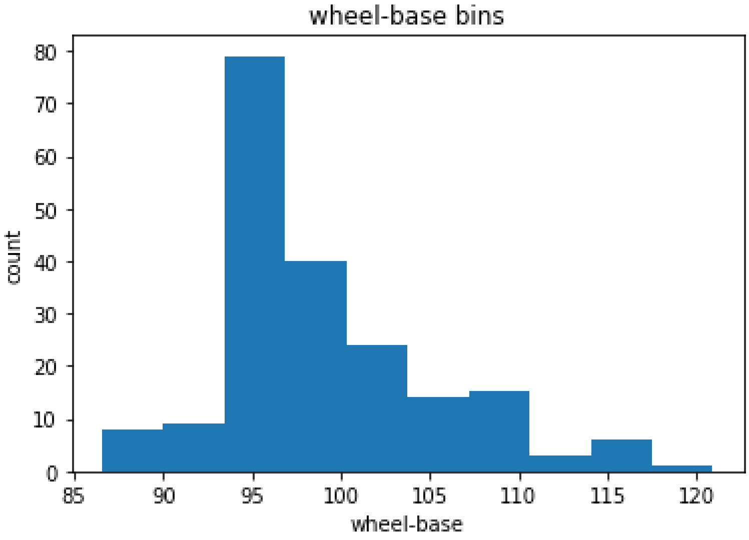
BINNING: It is process of transforming continous data into discrete categorical 'bins' for group analysis

```
df ["horsepower"]=df ["horsepower"].astype(float, copy=True)
```

```
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib import pyplot
import numpy as np

plt.matplotlib.pyplot.hist(df['wheel-base'])
plt.matplotlib.pyplot.xlabel('wheel-base')
plt.matplotlib.pyplot.ylabel('count')
plt.matplotlib.pyplot.title('wheel-base bins')
```

```
Text(0.5, 1.0, 'wheel-base bins')
```



```
bins = np.linspace(min(df['wheel-base']),max(df['wheel-base']),4)
bins
```

```
array([ 86.6       ,  98.03333333, 109.46666667, 120.9       ])
```

● ✕