### Data Wrangling I

Perform the following operations using Python on any open source dataset (e.g., data.csv) the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the

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

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

In [1]:
```
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).

In [2]:
```
url_link="/home/mca01/Downloads/autodata.csv"
df = pd.read_csv(url_link)
```

In [3]:
```
df.head(10)
```

Out[3]:

| | Unnamed: 0 | symboling | normalized-losses | make | aspiration | num-of-doors | body-style | w |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | 122 | alfa-romero | std | two | convertible | |
| **1** | 1 | 3 | 122 | alfa-romero | std | two | convertible | |
| **2** | 2 | 1 | 122 | alfa-romero | std | two | hatchback | |
| **3** | 3 | 2 | 164 | audi | std | four | sedan | |
| **4** | 4 | 2 | 164 | audi | std | four | sedan | |
| **5** | 5 | 2 | 122 | audi | std | two | sedan | |
| **6** | 6 | 1 | 158 | audi | std | four | sedan | |
| **7** | 7 | 1 | 122 | audi | std | four | wagon | |
| **8** | 8 | 1 | 158 | audi | turbo | four | sedan | |
| **9** | 9 | 2 | 192 | bmw | std | two | sedan | |

10 rows × 30 columns

In [4]:

```
df.tail()
```

Out[4]:

| | Unnamed: 0 | symboling | normalized-losses | make | aspiration | num-of-doors | body-style | dri whe |
|---|---|---|---|---|---|---|---|---|
| **196** | 196 | -1 | 95 | volvo | std | four | sedan | |
| **197** | 197 | -1 | 95 | volvo | turbo | four | sedan | |
| **198** | 198 | -1 | 95 | volvo | std | four | sedan | |
| **199** | 199 | -1 | 95 | volvo | turbo | four | sedan | |
| **200** | 200 | -1 | 95 | volvo | turbo | four | sedan | |

5 rows × 30 columns

In [5]:

```
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
```

In [6]:

```
df.describe()
```

Out[6]:

| | Unnamed: 0 | symboling | normalized-losses | wheel-base | length | widtl |
|---|---|---|---|---|---|---|
| **count** | 201.000000 | 201.000000 | 201.00000 | 201.000000 | 201.000000 | 201.00000 |
| **mean** | 100.000000 | 0.840796 | 122.00000 | 98.797015 | 0.837102 | 0.91512 |
| **std** | 58.167861 | 1.254802 | 31.99625 | 6.066366 | 0.059213 | 0.02918 |
| **min** | 0.000000 | -2.000000 | 65.00000 | 86.600000 | 0.678039 | 0.83750 |
| **25%** | 50.000000 | 0.000000 | 101.00000 | 94.500000 | 0.801538 | 0.89027 |
| **50%** | 100.000000 | 1.000000 | 122.00000 | 97.000000 | 0.832292 | 0.90972 |
| **75%** | 150.000000 | 2.000000 | 137.00000 | 102.400000 | 0.881788 | 0.92500 |
| **max** | 200.000000 | 3.000000 | 256.00000 | 120.900000 | 1.000000 | 1.00000 |

Loading [MathJax]/extensions/Safe.js

In [7]:
```
df.isnull()
```

Out[7]:

| | Unnamed: 0 | symboling | normalized-losses | make | aspiration | num-of-doors | body-style | dri whe |
|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | Fa |
| **1** | False | False | False | False | False | False | False | Fa |
| **2** | False | False | False | False | False | False | False | Fa |
| **3** | False | False | False | False | False | False | False | Fa |
| **4** | False | False | False | False | False | False | False | Fa |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **196** | False | False | False | False | False | False | False | Fa |
| **197** | False | False | False | False | False | False | False | Fa |
| **198** | False | False | False | False | False | False | False | Fa |
| **199** | False | False | False | False | False | False | False | Fa |
| **200** | False | False | False | False | False | False | False | Fa |

201 rows × 30 columns

In [8]:
```
df.isnull().sum()
```

Loading [MathJax]/extensions/Safe.js

Out[8]:

```
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
```

In [9]:

```
df.notnull()
```

Out[9]:

| | Unnamed: 0 | symboling | normalized-losses | make | aspiration | num-of-doors | body-style | dri whe |
|---|---|---|---|---|---|---|---|---|
| **0** | True | True | True | True | True | True | True | T |
| **1** | True | True | True | True | True | True | True | T |
| **2** | True | True | True | True | True | True | True | T |
| **3** | True | True | True | True | True | True | True | T |
| **4** | True | True | True | True | True | True | True | T |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **196** | True | True | True | True | True | True | True | T |
| **197** | True | True | True | True | True | True | True | T |
| **198** | True | True | True | True | True | True | True | T |
| **199** | True | True | True | True | True | True | True | T |
| **200** | True | True | True | True | True | True | True | T |

201 rows × 30 columns

In [10]:

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

```
Out[10]: 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
```

```
In [11]: #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 :

```
In [12]: avg_hp=df["horsepower"].astype("float").mean(axis = 0)
         print("Average of stroke :",avg_hp)
```

```
Average of stroke : 103.39698492462311
```

```
In [13]: df['horsepower'].replace(np.nan,avg_hp,inplace = True)
```

```
In [14]: from contextlib import nullcontext
         df['num-of-doors'].value_counts()
```

```
Out[14]: four    115
         two      86
         Name: num-of-doors, dtype: int64
```

In [15]:
```
df['num-of-doors'].value_counts().idxmax()
```

Out[15]:
```
'four'
```

In [16]:
```
# 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)
```

In [17]:
```
df.isnull().sum()
```

Out[17]:
```
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             0
city-mpg             0
highway-mpg          0
price                0
city-L/100km         0
horsepower-binned    0
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

In [18]:
```
df['city-L/100km']=235/df['city-mpg']
df.head()
```

Loading [MathJax]/extensions/Safe.js

Out[18]:

|   | Unnamed: 0 | symboling | normalized-losses | make | aspiration | num-of-doors | body-style |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | 122 | alfa-romero | std | two | convertible |
| **1** | 1 | 3 | 122 | alfa-romero | std | two | convertible |
| **2** | 2 | 1 | 122 | alfa-romero | std | two | hatchback |
| **3** | 3 | 2 | 164 | audi | std | four | sedan |
| **4** | 4 | 2 | 164 | audi | std | four | sedan |

5 rows × 30 columns

In [19]:

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

Out[19]:

|   | Unnamed: 0 | symboling | normalized-losses | make | aspiration | num-of-doors | body-style |
|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | 122 | alfa-romero | std | two | convertible |
| **1** | 1 | 3 | 122 | alfa-romero | std | two | convertible |
| **2** | 2 | 1 | 122 | alfa-romero | std | two | hatchback |
| **3** | 3 | 2 | 164 | audi | std | four | sedan |
| **4** | 4 | 2 | 164 | audi | std | four | sedan |

5 rows × 31 columns

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

In [20]:

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

In [21]:

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

Out[21]:

|   | 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

In [22]:

```
df.columns
```

Out[22]:

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

In [23]:

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

Out[23]:

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

In [24]:

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

Out[24]:

|   | std | turbo |
|---|---|---|
| **0** | 1 | 0 |
| **1** | 1 | 0 |
| **2** | 1 | 0 |
| **3** | 1 | 0 |
| **4** | 1 | 0 |

In [25]:

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

In [26]:

```
df.head()
```

Out[26]:

| | Unnamed: 0 | symboling | normalized-losses | make | num-of-doors | body-style | drive-wheels | eng loca |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | 122 | alfa-romero | two | convertible | rwd | f |
| **1** | 1 | 3 | 122 | alfa-romero | two | convertible | rwd | f |
| **2** | 2 | 1 | 122 | alfa-romero | two | hatchback | rwd | f |
| **3** | 3 | 2 | 164 | audi | four | sedan | fwd | f |
| **4** | 4 | 2 | 164 | audi | four | sedan | 4wd | f |

5 rows × 32 columns

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

In [27]:

```
df.columns
```

Out[27]:

```
Index(['Unnamed: 0', 'symboling', 'normalized-losses', 'make', 'num-of-door
s',
       'body-style', 'drive-wheels', 'engine-location', 'wheel-base', 'leng
th',
       '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/100
km',
       'std', 'turbo'],
      dtype='object')
```

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

In [28]:

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

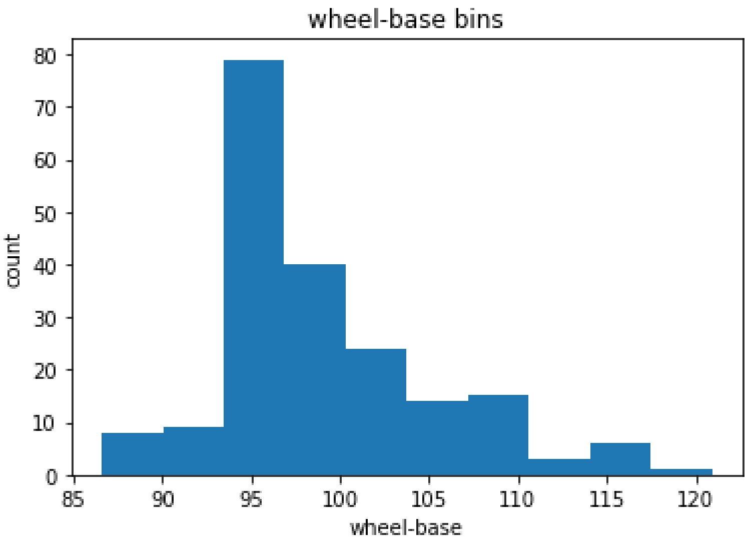
In [29]:

```
%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')
```

Out[29]:

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



```
In [30]: bins = np.linspace(min(df['wheel-base']),max(df['wheel-base']),4)
         bins
```

```
Out[30]: array([ 86.6       ,  98.03333333, 109.46666667, 120.9       ])
```

```
In [ ]:
```