task2

April 30, 2024

[2]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
```

1 Load and Read the dataset

[4]:
```
# Use raw string literal or double backslashes for file path
data = pd.read_csv(r"C:\Users\aakas\OneDrive\Desktop\Rprog\Automobile_data.csv")

# Check the first few rows of the DataFrame
print(data.head())
```

```
   symboling normalized-losses         make fuel-type aspiration num-of-doors  \
0          3                 ?  alfa-romero       gas        std          two
1          3                 ?  alfa-romero       gas        std          two
2          1                 ?  alfa-romero       gas        std          two
3          2               164         audi       gas        std         four
4          2               164         audi       gas        std         four

    body-style drive-wheels engine-location  wheel-base  …  engine-size  \
0  convertible          rwd           front        88.6  …          130
1  convertible          rwd           front        88.6  …          130
2    hatchback          rwd           front        94.5  …          152
3        sedan          fwd           front        99.8  …          109
4        sedan          4wd           front        99.4  …          136

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm city-mpg  \
0         mpfi  3.47    2.68               9.0        111      5000       21
1         mpfi  3.47    2.68               9.0        111      5000       21
2         mpfi  2.68    3.47               9.0        154      5000       19
3         mpfi  3.19     3.4              10.0        102      5500       24
4         mpfi  3.19     3.4               8.0        115      5500       18

  highway-mpg  price
```

```
0          27  13495
1          27  16500
2          26  16500
3          30  13950
4          22  17450

[5 rows x 26 columns]
```

2 Display the Datatypes of Data Columns

[5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  205 non-null    object
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   aspiration         205 non-null    object
 5   num-of-doors       205 non-null    object
 6   body-style         205 non-null    object
 7   drive-wheels       205 non-null    object
 8   engine-location    205 non-null    object
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  curb-weight        205 non-null    int64
 14  engine-type        205 non-null    object
 15  num-of-cylinders   205 non-null    object
 16  engine-size        205 non-null    int64
 17  fuel-system        205 non-null    object
 18  bore               205 non-null    object
 19  stroke             205 non-null    object
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    object
 22  peak-rpm           205 non-null    object
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              205 non-null    object
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
```

3 Display the Statistics of Data

[6]:
```
data.describe()
```

[6]:
```
         symboling  wheel-base      length       width      height  \
count  205.000000  205.000000  205.000000  205.000000  205.000000
mean     0.834146   98.756585  174.049268   65.907805   53.724878
std      1.245307    6.021776   12.337289    2.145204    2.443522
min     -2.000000   86.600000  141.100000   60.300000   47.800000
25%      0.000000   94.500000  166.300000   64.100000   52.000000
50%      1.000000   97.000000  173.200000   65.500000   54.100000
75%      2.000000  102.400000  183.100000   66.900000   55.500000
max      3.000000  120.900000  208.100000   72.300000   59.800000

       curb-weight  engine-size  compression-ratio    city-mpg  highway-mpg
count   205.000000   205.000000         205.000000  205.000000   205.000000
mean   2555.565854   126.907317          10.142537   25.219512    30.751220
std     520.680204    41.642693           3.972040    6.542142     6.886443
min    1488.000000    61.000000           7.000000   13.000000    16.000000
25%    2145.000000    97.000000           8.600000   19.000000    25.000000
50%    2414.000000   120.000000           9.000000   24.000000    30.000000
75%    2935.000000   141.000000           9.400000   30.000000    34.000000
max    4066.000000   326.000000          23.000000   49.000000    54.000000
```

4 Cleaning The Dataset

[7]:
```
data.isnull().sum()
```

[7]:
```
symboling            0
normalized-losses    0
make                 0
fuel-type            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
dtype: int64
```

Cleaning the normalized losses field

[8]:
```
#Cleaning the normalized losses field
data['normalized-losses'].loc[data['normalized-losses'] == '?'].count()
```

[8]:
```
41
```

[9]:
```
nl = data['normalized-losses'].loc[data['normalized-losses'] != '?']
nlmean = nl.astype(str).astype(int).mean()
data['normalized-losses'] = data['normalized-losses'].replace('?',nlmean).
 ↪astype(int)
data['normalized-losses'].head()
```

[9]:
```
0    122
1    122
2    122
3    164
4    164
Name: normalized-losses, dtype: int32
```

5 Cleaning the horsepower

[10]:
```
data['horsepower'].str.isnumeric().value_counts()
```

[10]:
```
True     203
False      2
Name: horsepower, dtype: int64
```

[11]:
```
data['horsepower'].loc[data['horsepower'] == '?']
```

[11]:
```
130    ?
131    ?
Name: horsepower, dtype: object
```

[15]:
```
# Replace '?' in 'horsepower' column with NaN and convert to numeric
data['horsepower'] = pd.to_numeric(data['horsepower'], errors='coerce')
```

```
# Calculate mean horsepower excluding NaN values
hp_mean = data['horsepower'].mean()

# Replace NaN values with the mean horsepower
data['horsepower'] = data['horsepower'].fillna(hp_mean).astype(int)

# Check the first few rows of the 'horsepower' column
print(data['horsepower'].head())
```

```
0    111
1    111
2    154
3    102
4    115
Name: horsepower, dtype: int32
```

[16]:
```
#Checking the outlier of horsepower
data.loc[data['horsepower'] > 10000]
```

[16]:
```
Empty DataFrame
Columns: [symboling, normalized-losses, make, fuel-type, 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]
Index: []

[0 rows x 26 columns]
```

[17]:
```
#Excluding the outlier data for horsepower
data[np.abs(data.horsepower-data.horsepower.mean())<=(3*data.horsepower.std())]
```

[17]:
```
     symboling  normalized-losses         make fuel-type aspiration  \
0            3                122  alfa-romero       gas        std
1            3                122  alfa-romero       gas        std
2            1                122  alfa-romero       gas        std
3            2                164         audi       gas        std
4            2                164         audi       gas        std
..         …                  …          …       …         …
200         -1                 95        volvo       gas        std
201         -1                 95        volvo       gas      turbo
202         -1                 95        volvo       gas        std
203         -1                 95        volvo    diesel      turbo
204         -1                 95        volvo       gas      turbo

    num-of-doors   body-style drive-wheels engine-location  wheel-base  …  \
0            two  convertible          rwd           front        88.6  …
1            two  convertible          rwd           front        88.6  …
```

```
2            two    hatchback          rwd           front        94.5  …
3           four        sedan          fwd           front        99.8  …
4           four        sedan          4wd           front        99.4  …
..           …          …          …              …          …  …
200         four        sedan          rwd           front       109.1  …
201         four        sedan          rwd           front       109.1  …
202         four        sedan          rwd           front       109.1  …
203         four        sedan          rwd           front       109.1  …
204         four        sedan          rwd           front       109.1  …

     engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0            130         mpfi  3.47    2.68               9.0        111
1            130         mpfi  3.47    2.68               9.0        111
2            152         mpfi  2.68    3.47               9.0        154
3            109         mpfi  3.19     3.4              10.0        102
4            136         mpfi  3.19     3.4               8.0        115
..           …          …   …     …                …         …
200          141         mpfi  3.78    3.15               9.5        114
201          141         mpfi  3.78    3.15               8.7        160
202          173         mpfi  3.58    2.87               8.8        134
203          145          idi  3.01     3.4              23.0        106
204          141         mpfi  3.78    3.15               9.5        114

     peak-rpm city-mpg highway-mpg  price
0        5000       21          27  13495
1        5000       21          27  16500
2        5000       19          26  16500
3        5500       24          30  13950
4        5500       18          22  17450
..        …      …          …     …
200      5400       23          28  16845
201      5300       19          25  19045
202      5500       18          23  21485
203      4800       26          27  22470
204      5400       19          25  22625

[203 rows x 26 columns]
```

6 Cleaning bore

[18]:
```
data['bore'].loc[data['bore']=='?']
```

[18]:
```
55    ?
56    ?
57    ?
58    ?
```

```
Name: bore, dtype: object
```

[19]:
```
# Replace the non-numeric value to null and conver the datatype
data['bore'] = pd.to_numeric(data['bore'],errors='coerce')
data.dtypes
```

[19]:
```
symboling              int64
normalized-losses      int32
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                 float64
stroke                object
compression-ratio    float64
horsepower             int32
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

[20]:
```
# Cleaning the stroke
```

[21]:
```
# Replace the non-numeric value to null and conver the datatype
data['stroke'] = pd.to_numeric(data['stroke'],errors='coerce')
data.dtypes
```

[21]:
```
symboling              int64
normalized-losses      int32
make                  object
fuel-type             object
aspiration            object
num-of-doors          object
```

```
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                 float64
stroke               float64
compression-ratio    float64
horsepower             int32
peak-rpm              object
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

7 Cleaning the peak rpm data

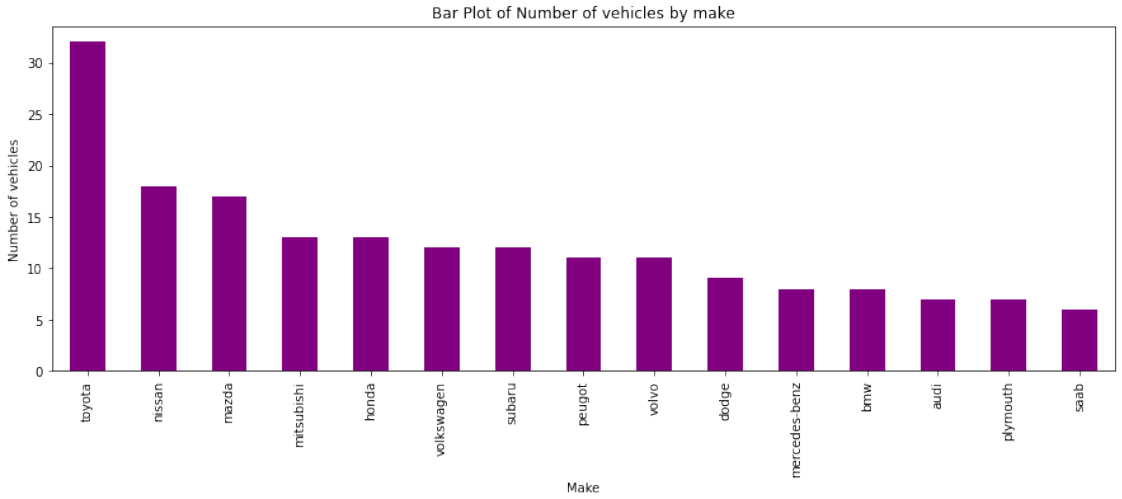
[22]:
```
# Convert the non-numeric data to null and convert the datatype
data['peak-rpm'] = pd.to_numeric(data['peak-rpm'],errors='coerce')
data.dtypes
```

[22]:
```
symboling             int64
normalized-losses     int32
make                 object
fuel-type            object
aspiration           object
num-of-doors         object
body-style           object
drive-wheels         object
engine-location      object
wheel-base          float64
length              float64
width               float64
height              float64
curb-weight           int64
engine-type          object
num-of-cylinders     object
engine-size           int64
```

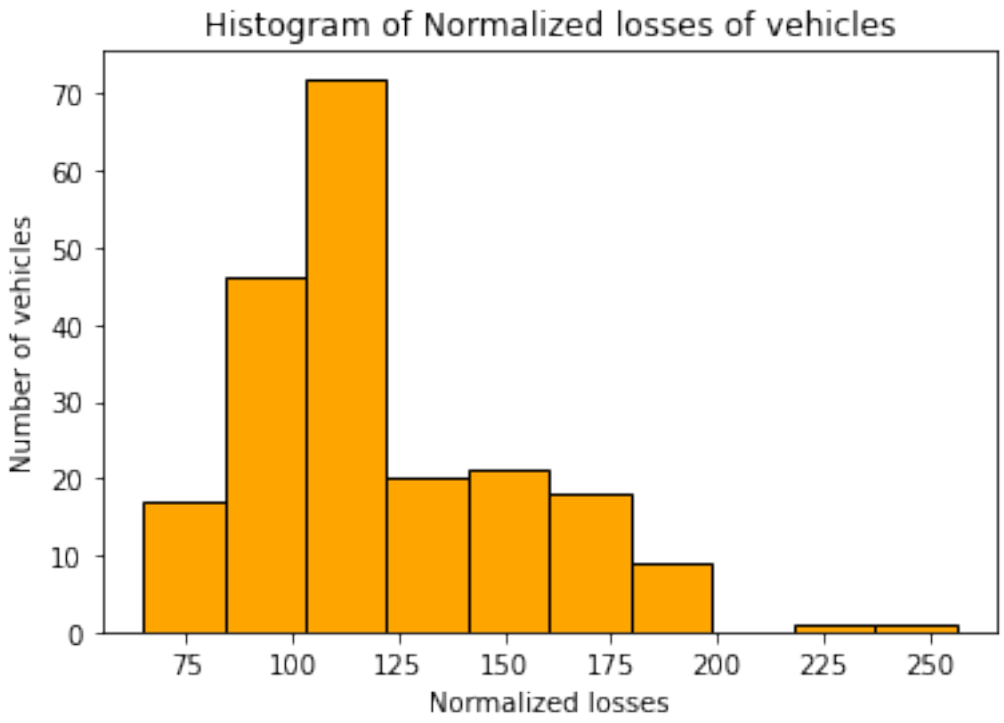
```
fuel-system           object
bore                 float64
stroke               float64
compression-ratio    float64
horsepower             int32
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                 object
dtype: object
```

8 Exploratory Data Analysis of Dataset

[23]:
```
#Bar Plot of Make of Vehicles
data.make.value_counts().nlargest(15).plot(kind='bar', figsize=(15,5),␣
 ↪color='purple')
plt.title("Bar Plot of Number of vehicles by make")
plt.ylabel('Number of vehicles')
plt.xlabel('Make');
```



[24]:
```
#Histogram of Normalized Loses Of Vehicles
plt.hist(data['normalized-losses'],color='orange', edgecolor='black');
plt.title("Histogram of Normalized losses of vehicles")
plt.ylabel('Number of vehicles')
plt.xlabel('Normalized losses');
```


Histogram of Normalized losses of vehicles
Number of vehicles
0
10
20
30
40
50
60
70
75
100
125
150
175
200
225
250
Normalized losses

[ ]:

[ ]:

[ ]: