Working.Assignment-Statistics for ML

May 11, 2024

0.1 Import Libraries

[91]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
import numpy as np
from scipy.stats import skewnorm
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

0.2 Read Car_Sales.csv Data

[4]:
```
df=pd.read_csv('Car_sales.csv')
```

0.2.1 Shape of Data

[6]:
```
df.shape
```

[6]:
```
(157, 16)
```

0.3 Let's Explore the data

[8]:
```
df.head()
```

[8]:
```
  Manufacturer    Model  Sales_in_thousands  __year_resale_value Vehicle_type  \
0        Acura  Integra              16.919               16.360    Passenger
1        Acura       TL              39.384               19.875    Passenger
2        Acura       CL              14.114               18.225    Passenger
3        Acura       RL               8.588               29.725    Passenger
4         Audi       A4              20.397               22.255    Passenger

   Price_in_thousands  Engine_size  Horsepower  Wheelbase  Width  Length  \
0               21.50          1.8       140.0      101.2   67.3   172.4
1               28.40          3.2       225.0      108.1   70.3   192.9
2                 NaN          3.2       225.0      106.9   70.6   192.0
3               42.00          3.5       210.0      114.6   71.4   196.6
4               23.99          1.8       150.0      102.6   68.2   178.0
```

```
   Curb_weight  Fuel_capacity  Fuel_efficiency Latest_Launch  \
0        2.639           13.2             28.0      2/2/2012
1        3.517           17.2             25.0      6/3/2011
2        3.470           17.2             26.0      1/4/2012
3        3.850           18.0             22.0     3/10/2011
4        2.998           16.4             27.0     10/8/2011

   Power_perf_factor
0          58.280150
1          91.370778
2                NaN
3          91.389779
4          62.777639
```

[9]:
```
df.tail()
```

[9]:
```
     Manufacturer Model  Sales_in_thousands  __year_resale_value Vehicle_type  \
152         Volvo   V40               3.545                  NaN    Passenger
153         Volvo   S70              15.245                  NaN    Passenger
154         Volvo   V70              17.531                  NaN    Passenger
155         Volvo   C70               3.493                  NaN    Passenger
156         Volvo   S80              18.969                  NaN    Passenger

     Price_in_thousands  Engine_size  Horsepower  Wheelbase  Width  Length  \
152                24.4          1.9       160.0      100.5   67.6   176.6
153                27.5          2.4       168.0      104.9   69.3   185.9
154                28.8          2.4       168.0      104.9   69.3   186.2
155                45.5          2.3       236.0      104.9   71.5   185.7
156                36.0          2.9       201.0      109.9   72.1   189.8

     Curb_weight  Fuel_capacity  Fuel_efficiency Latest_Launch  \
152        3.042           15.8             25.0     9/21/2011
153        3.208           17.9             25.0    11/24/2012
154        3.259           17.9             25.0     6/25/2011
155        3.601           18.5             23.0     4/26/2011
156        3.600           21.1             24.0    11/14/2011

     Power_perf_factor
152          66.498812
153          70.654495
154          71.155978
155         101.623357
156          85.735655
```

0.3.1 Basic information about the dataset and data types

[11]:
```
df.info
```

[11]:
```
<bound method DataFrame.info of     Manufacturer    Model  Sales_in_thousands
__year_resale_value  \
0          Acura  Integra              16.919               16.360
1          Acura       TL              39.384               19.875
2          Acura       CL              14.114               18.225
3          Acura       RL               8.588               29.725
4           Audi       A4              20.397               22.255
..           …       …                 …                  …
152        Volvo      V40               3.545                  NaN
153        Volvo      S70              15.245                  NaN
154        Volvo      V70              17.531                  NaN
155        Volvo      C70               3.493                  NaN
156        Volvo      S80              18.969                  NaN

    Vehicle_type  Price_in_thousands  Engine_size  Horsepower  Wheelbase  \
0      Passenger               21.50          1.8       140.0      101.2
1      Passenger               28.40          3.2       225.0      108.1
2      Passenger                 NaN          3.2       225.0      106.9
3      Passenger               42.00          3.5       210.0      114.6
4      Passenger               23.99          1.8       150.0      102.6
..           …                 …          …          …          …
152    Passenger               24.40          1.9       160.0      100.5
153    Passenger               27.50          2.4       168.0      104.9
154    Passenger               28.80          2.4       168.0      104.9
155    Passenger               45.50          2.3       236.0      104.9
156    Passenger               36.00          2.9       201.0      109.9

     Width  Length  Curb_weight  Fuel_capacity  Fuel_efficiency Latest_Launch  \
0     67.3   172.4        2.639           13.2             28.0      2/2/2012
1     70.3   192.9        3.517           17.2             25.0      6/3/2011
2     70.6   192.0        3.470           17.2             26.0      1/4/2012
3     71.4   196.6        3.850           18.0             22.0     3/10/2011
4     68.2   178.0        2.998           16.4             27.0     10/8/2011
..     …      …          …            …              …           …
152   67.6   176.6        3.042           15.8             25.0     9/21/2011
153   69.3   185.9        3.208           17.9             25.0    11/24/2012
154   69.3   186.2        3.259           17.9             25.0     6/25/2011
155   71.5   185.7        3.601           18.5             23.0     4/26/2011
156   72.1   189.8        3.600           21.1             24.0    11/14/2011

     Power_perf_factor
0            58.280150
1            91.370778
```

```
2                NaN
3          91.389779
4          62.777639
..               …
152        66.498812
153        70.654495
154        71.155978
155       101.623357
156        85.735655

[157 rows x 16 columns]>
```

[12]:
```
df.dtypes
```

[12]:
```
Manufacturer            object
Model                   object
Sales_in_thousands     float64
__year_resale_value    float64
Vehicle_type            object
Price_in_thousands     float64
Engine_size            float64
Horsepower             float64
Wheelbase              float64
Width                  float64
Length                 float64
Curb_weight            float64
Fuel_capacity          float64
Fuel_efficiency        float64
Latest_Launch           object
Power_perf_factor      float64
dtype: object
```

Converting 'Latest Launch' to datetime format

[14]:
```
df['Latest_Launch'] = pd.to_datetime(df['Latest_Launch'])
```

[15]:
```
df.dtypes
```

[15]:
```
Manufacturer                   object
Model                          object
Sales_in_thousands            float64
__year_resale_value           float64
Vehicle_type                   object
Price_in_thousands            float64
Engine_size                   float64
Horsepower                    float64
Wheelbase                     float64
```

```
Width                         float64
Length                        float64
Curb_weight                   float64
Fuel_capacity                 float64
Fuel_efficiency               float64
Latest_Launch          datetime64[ns]
Power_perf_factor             float64
dtype: object
```

[16]:
```
df.describe
```

[16]:
```
<bound method NDFrame.describe of     Manufacturer    Model  Sales_in_thousands
__year_resale_value  \
0          Acura  Integra              16.919               16.360
1          Acura       TL              39.384               19.875
2          Acura       CL              14.114               18.225
3          Acura       RL               8.588               29.725
4           Audi       A4              20.397               22.255
..           …        …                 …                  …
152        Volvo      V40               3.545                  NaN
153        Volvo      S70              15.245                  NaN
154        Volvo      V70              17.531                  NaN
155        Volvo      C70               3.493                  NaN
156        Volvo      S80              18.969                  NaN

    Vehicle_type  Price_in_thousands  Engine_size  Horsepower  Wheelbase  \
0      Passenger               21.50          1.8       140.0      101.2
1      Passenger               28.40          3.2       225.0      108.1
2      Passenger                 NaN          3.2       225.0      106.9
3      Passenger               42.00          3.5       210.0      114.6
4      Passenger               23.99          1.8       150.0      102.6
..           …                   …          …          …          …
152    Passenger               24.40          1.9       160.0      100.5
153    Passenger               27.50          2.4       168.0      104.9
154    Passenger               28.80          2.4       168.0      104.9
155    Passenger               45.50          2.3       236.0      104.9
156    Passenger               36.00          2.9       201.0      109.9

     Width  Length  Curb_weight  Fuel_capacity  Fuel_efficiency Latest_Launch  \
0     67.3   172.4        2.639           13.2             28.0    2012-02-02
1     70.3   192.9        3.517           17.2             25.0    2011-06-03
2     70.6   192.0        3.470           17.2             26.0    2012-01-04
3     71.4   196.6        3.850           18.0             22.0    2011-03-10
4     68.2   178.0        2.998           16.4             27.0    2011-10-08
..     …      …           …             …               …           …
152   67.6   176.6        3.042           15.8             25.0    2011-09-21
153   69.3   185.9        3.208           17.9             25.0    2012-11-24
```

```
154   69.3   186.2        3.259           17.9              25.0     2011-06-25
155   71.5   185.7        3.601           18.5              23.0     2011-04-26
156   72.1   189.8        3.600           21.1              24.0     2011-11-14

     Power_perf_factor
0            58.280150
1            91.370778
2                  NaN
3            91.389779
4            62.777639
..                 …
152          66.498812
153          70.654495
154          71.155978
155         101.623357
156          85.735655

[157 rows x 16 columns]>
```

Prior to summary statistics, identify missing values or duplicates and drop them.

0.3.2 Missing values

[19]:
```
df.isna().sum()
```

[19]:
```
Manufacturer            0
Model                   0
Sales_in_thousands      0
__year_resale_value    36
Vehicle_type            0
Price_in_thousands      2
Engine_size             1
Horsepower              1
Wheelbase               1
Width                   1
Length                  1
Curb_weight             2
Fuel_capacity           1
Fuel_efficiency         3
Latest_Launch           0
Power_perf_factor       2
dtype: int64
```

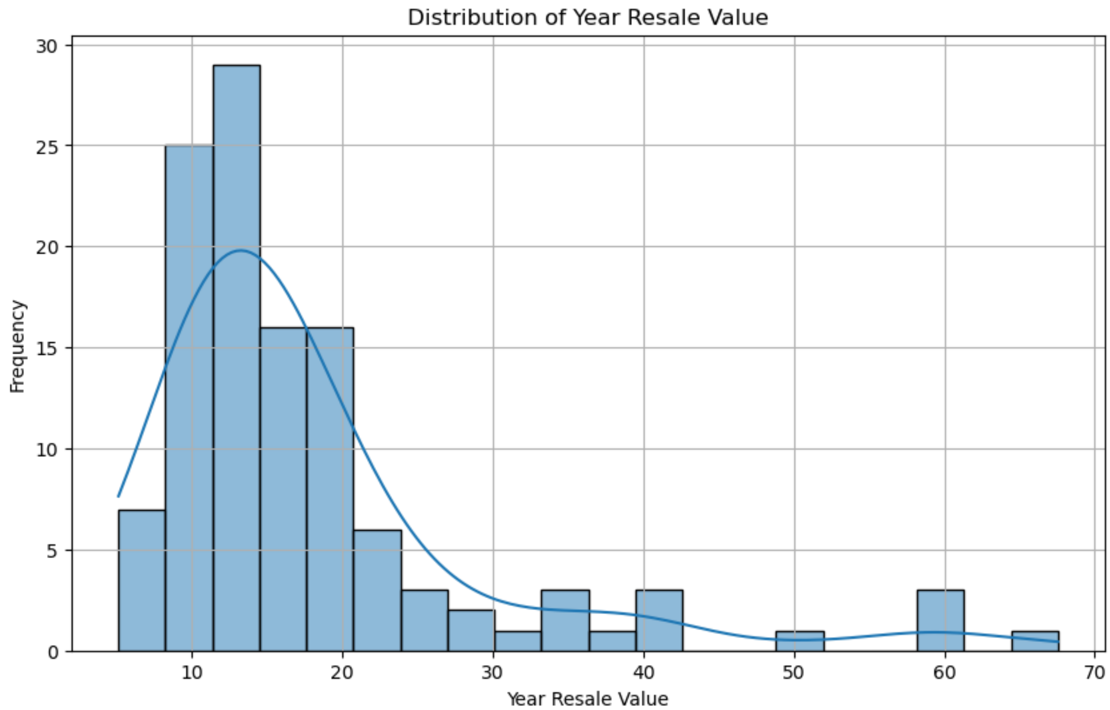
[20]:
```
columns_to_check = ['Price_in_thousands', 'Engine_size', 'Horsepower',␣
 ↪'Wheelbase', 'Width', 'Length',
                    'Curb_weight', 'Fuel_capacity', 'Fuel_efficiency',␣
 ↪'Power_perf_factor']
```

```
df1 = df.dropna(subset=columns_to_check)

df1.isnull().sum()
```

[20]:
```
Manufacturer            0
Model                   0
Sales_in_thousands      0
__year_resale_value    35
Vehicle_type            0
Price_in_thousands      0
Engine_size             0
Horsepower              0
Wheelbase               0
Width                   0
Length                  0
Curb_weight             0
Fuel_capacity           0
Fuel_efficiency         0
Latest_Launch           0
Power_perf_factor       0
dtype: int64
```

Distribution for '___year__resale__value'

[22]:
```
# Plotting the distribution of the __year_resale_value column
plt.figure(figsize=(10, 6))
sns.histplot(df1['__year_resale_value'].dropna(), kde=True, bins=20)
plt.title('Distribution of Year Resale Value')
plt.xlabel('Year Resale Value')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



Imputing with median values

[24]:
```
median_resale_value = df1['__year_resale_value'].median()

# Impute missing values with the median
df1['__year_resale_value'].fillna(median_resale_value, inplace=True)

# Verification
df1['__year_resale_value'].isnull().sum()
```

```
/tmp/ipykernel_155/1722989452.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['__year_resale_value'].fillna(median_resale_value, inplace=True)
```

[24]: 0

[25]:
```
df1.isnull().sum()
```

[25]:
```
Manufacturer          0
Model                 0
Sales_in_thousands    0
```

```
__year_resale_value    0
Vehicle_type           0
Price_in_thousands     0
Engine_size            0
Horsepower             0
Wheelbase              0
Width                  0
Length                 0
Curb_weight            0
Fuel_capacity          0
Fuel_efficiency        0
Latest_Launch          0
Power_perf_factor      0
dtype: int64
```

0.4 Check Duplicates

[27]:
```python
df1.duplicated().sum()
```

[27]:
```
0
```

0.4.1 Keep only numeric columns in a dataframe

hint: (use function select_dtype)

[29]:
```python
df2 = df1.select_dtypes(include=['number'])
```

0.4.2 Summary statistics

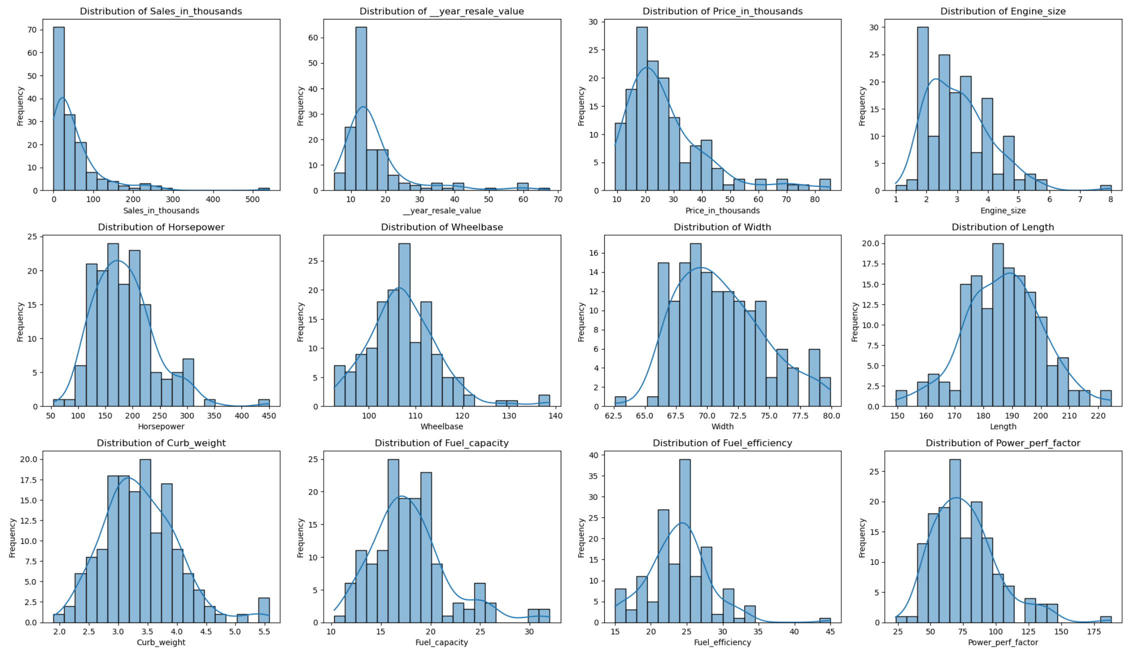
[31]:
```python
df2.describe
```

[31]:
```
<bound method NDFrame.describe of      Sales_in_thousands  __year_resale_value
Price_in_thousands  Engine_size  \
0                 16.919               16.360              21.50          1.8
1                 39.384               19.875              28.40          3.2
3                  8.588               29.725              42.00          3.5
4                 20.397               22.255              23.99          1.8
5                 18.780               23.555              33.95          2.8
..                   …                    …                  …            …
152                3.545               14.010              24.40          1.9
153               15.245               14.010              27.50          2.4
154               17.531               14.010              28.80          2.4
155                3.493               14.010              45.50          2.3
156               18.969               14.010              36.00          2.9

     Horsepower  Wheelbase  Width  Length  Curb_weight  Fuel_capacity  \
0         140.0      101.2   67.3   172.4        2.639           13.2
```

```
1          225.0      108.1   70.3   192.9          3.517            17.2
3          210.0      114.6   71.4   196.6          3.850            18.0
4          150.0      102.6   68.2   178.0          2.998            16.4
5          200.0      108.7   76.1   192.0          3.561            18.5
..           …          …      …       …            …            …
152        160.0      100.5   67.6   176.6          3.042            15.8
153        168.0      104.9   69.3   185.9          3.208            17.9
154        168.0      104.9   69.3   186.2          3.259            17.9
155        236.0      104.9   71.5   185.7          3.601            18.5
156        201.0      109.9   72.1   189.8          3.600            21.1

     Fuel_efficiency  Power_perf_factor
0               28.0          58.280150
1               25.0          91.370778
3               22.0          91.389779
4               27.0          62.777639
5               22.0          84.565105
..               …                …
152             25.0          66.498812
153             25.0          70.654495
154             25.0          71.155978
155             23.0         101.623357
156             24.0          85.735655

[152 rows x 12 columns]>
```

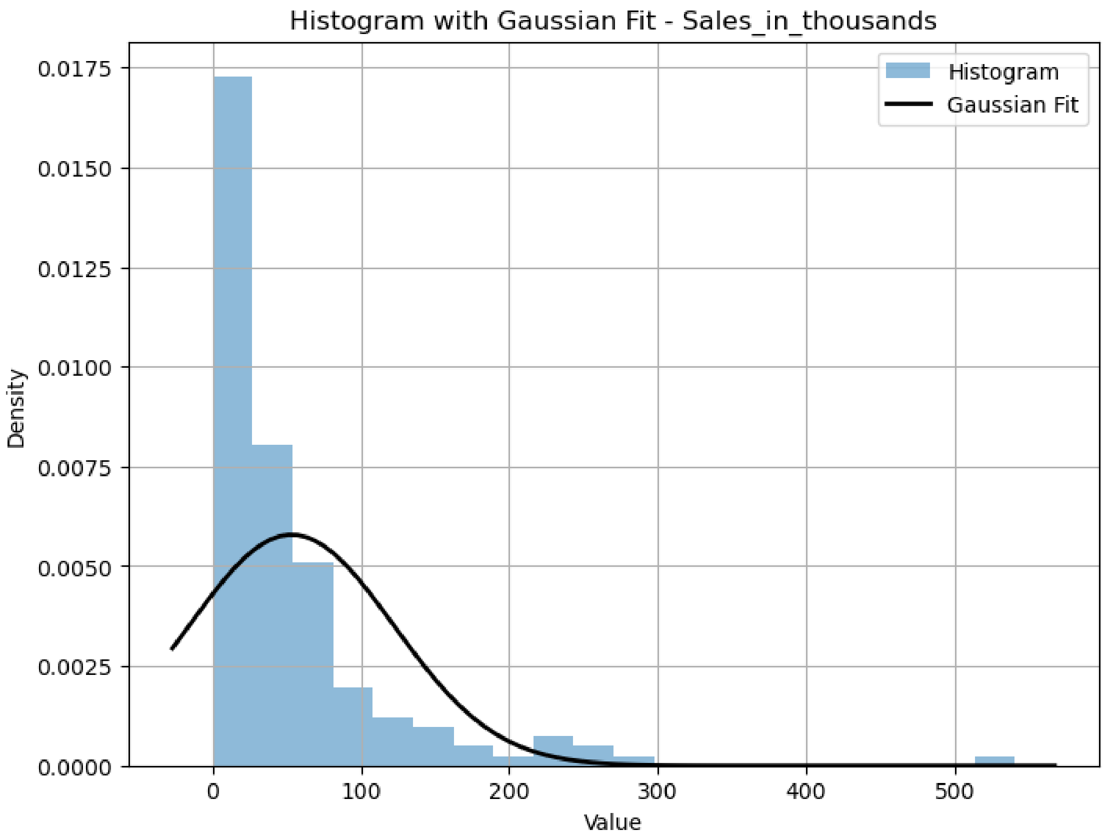
0.5 Distribution

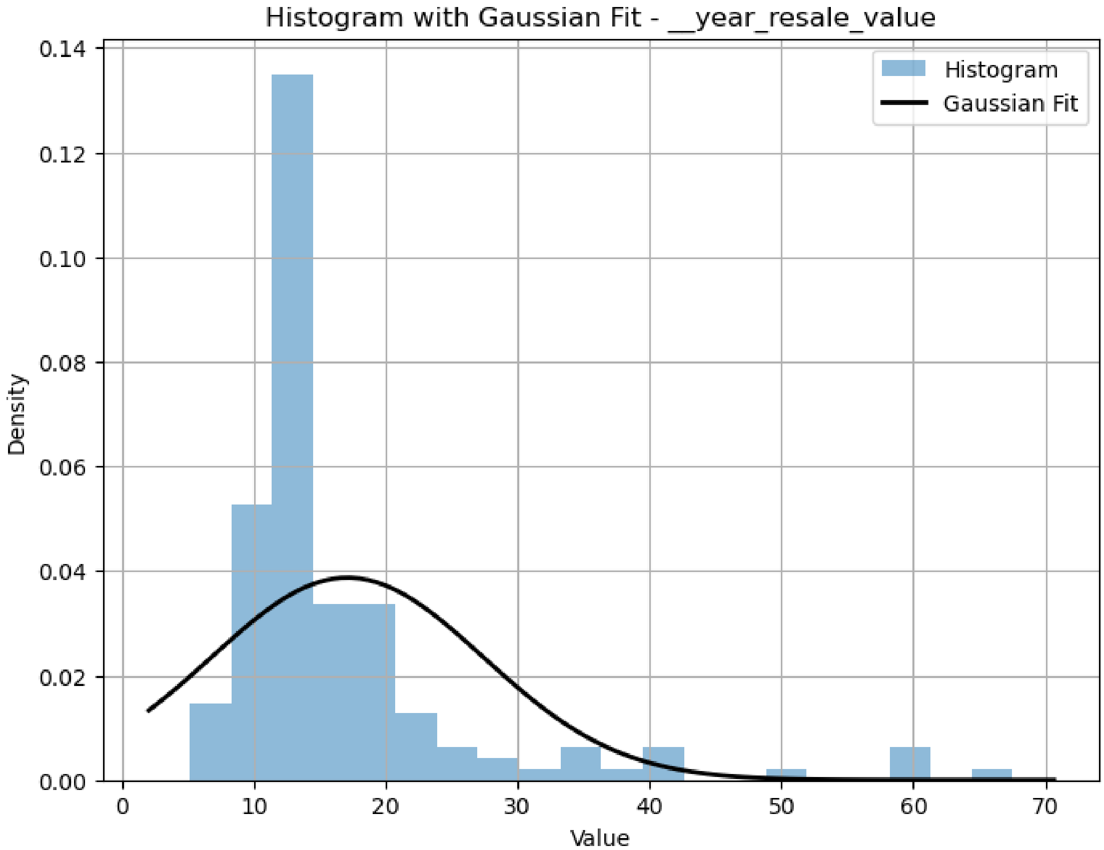
[33]:
```python
# Plotting histograms for each numeric column in df2
plt.figure(figsize=(20, 15))
for i, column in enumerate(df2.columns, 1):
    plt.subplot(4, 4, i)
    sns.histplot(df2[column], kde=True, bins=20)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



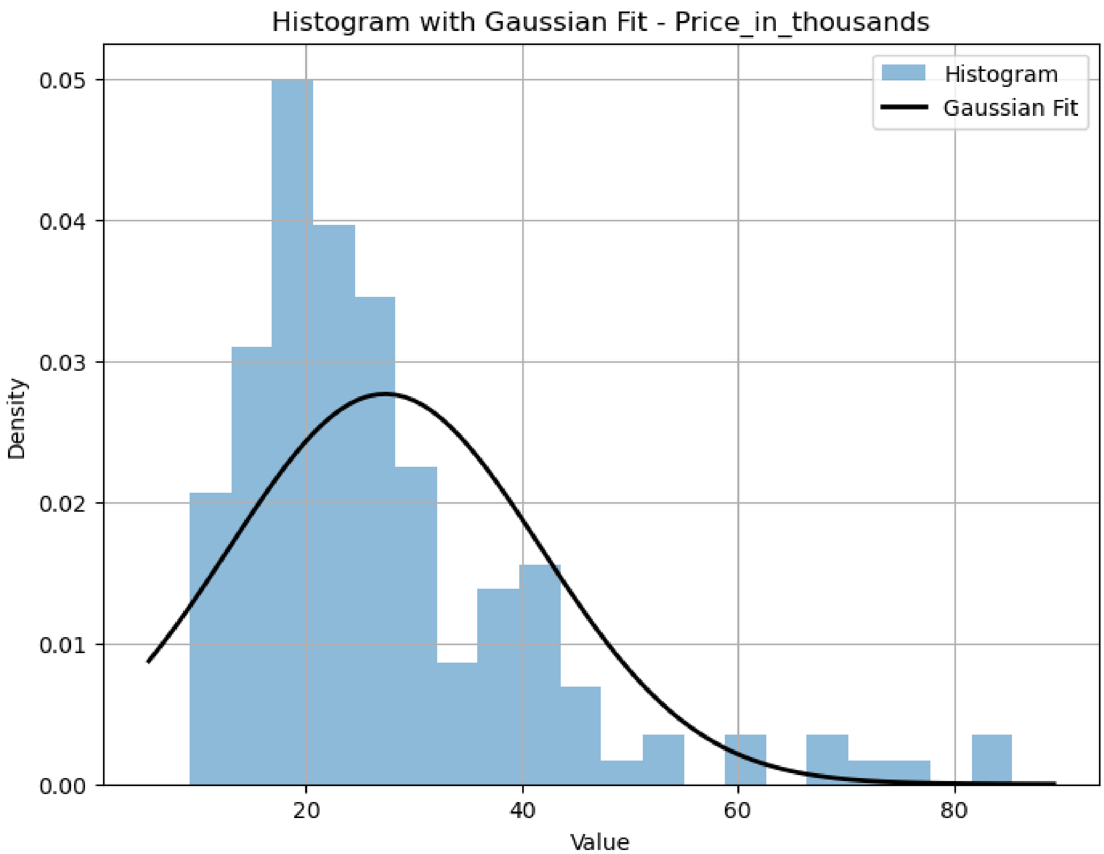
0.6 Normalized Distribution (Gaussian)

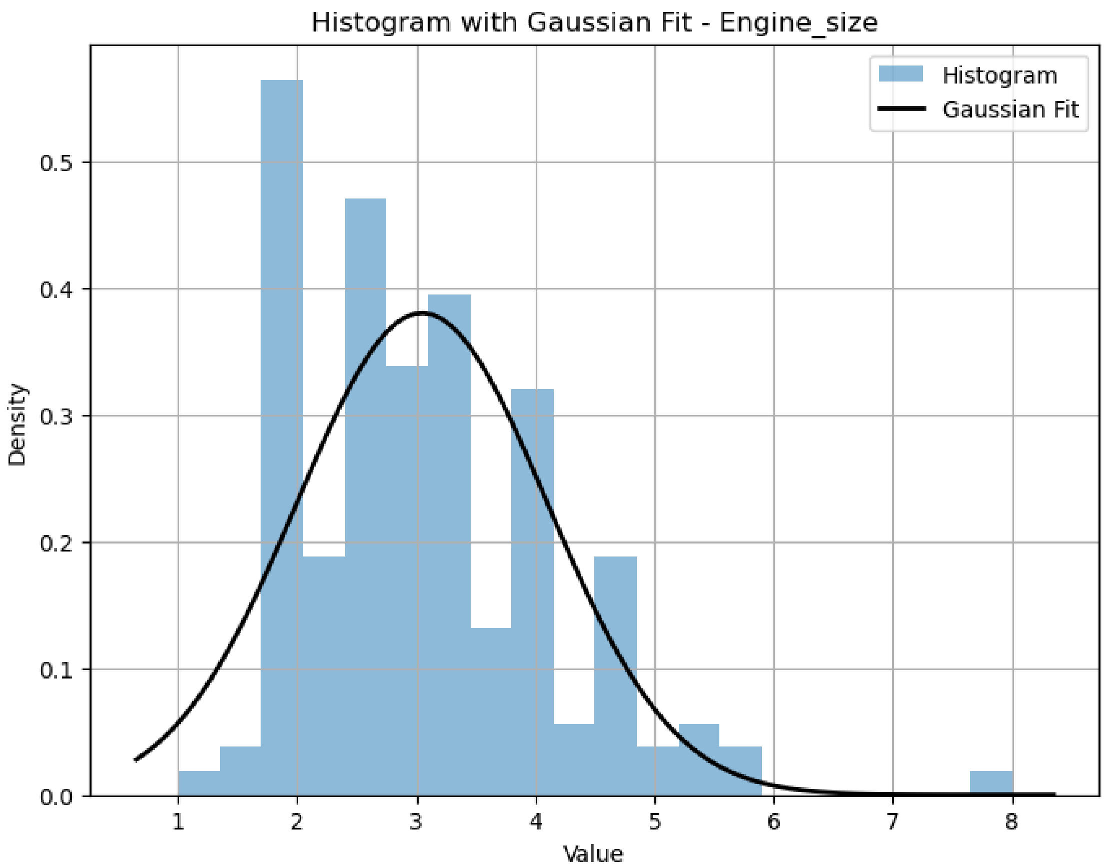
[35]:
```
means = df2.mean()
std_devs = df2.std()


for col in df2.columns:
    plt.figure(figsize=(8, 6))


    plt.hist(df2[col], bins=20, density=True, alpha=0.5, label='Histogram')


    mu, sigma = means[col], std_devs[col]
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    p = norm.pdf(x, mu, sigma)
    plt.plot(x, p, 'k', linewidth=2, label='Gaussian Fit')

    plt.title(f'Histogram with Gaussian Fit - {col}')
    plt.xlabel('Value')
    plt.ylabel('Density')
    plt.legend()
    plt.grid(True)
    plt.show()
```

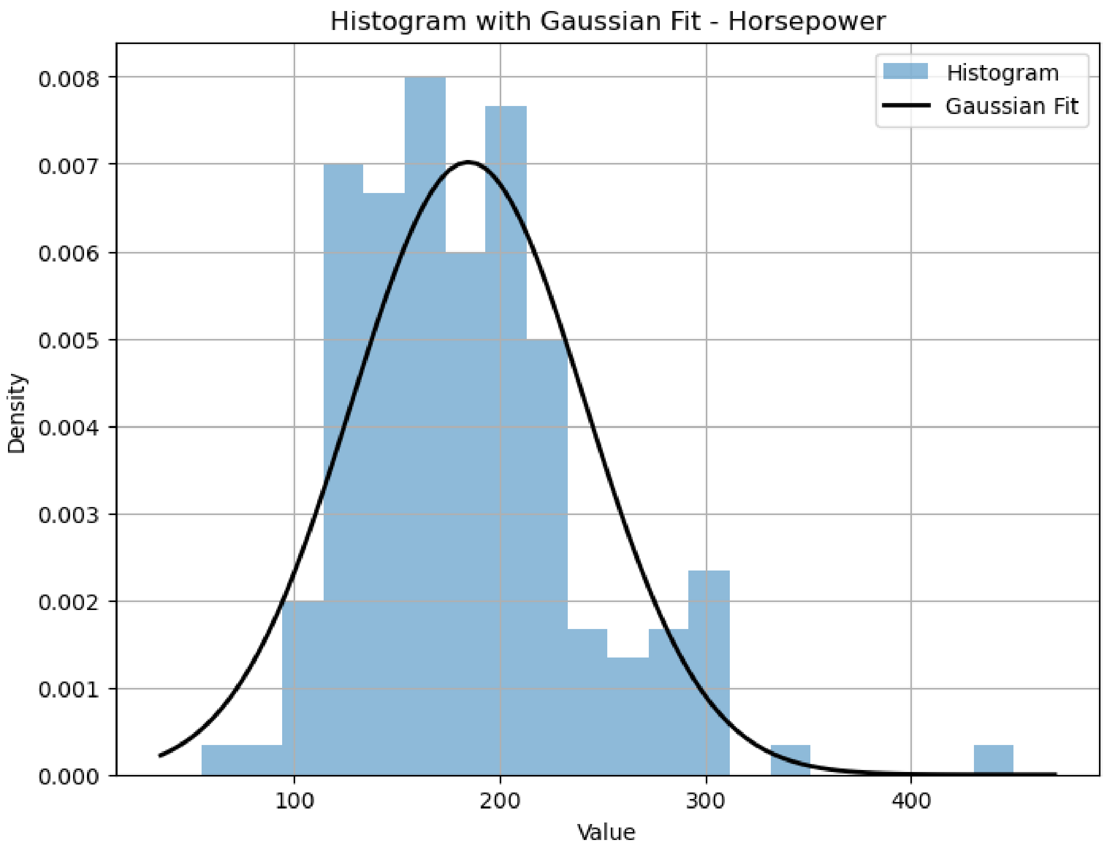

Histogram with Gaussian Fit - Sales_in_thousands
Histogram
Gaussian Fit
Density
Value
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0
100
200
300
400
500

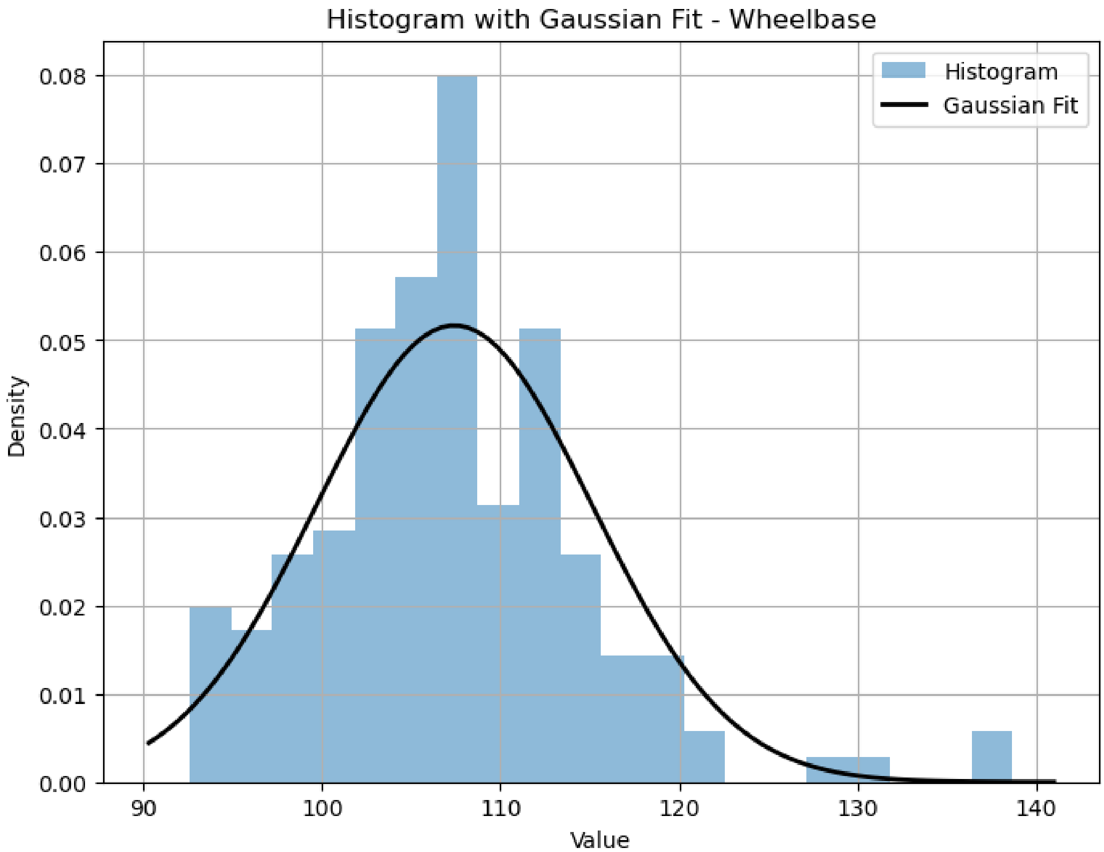


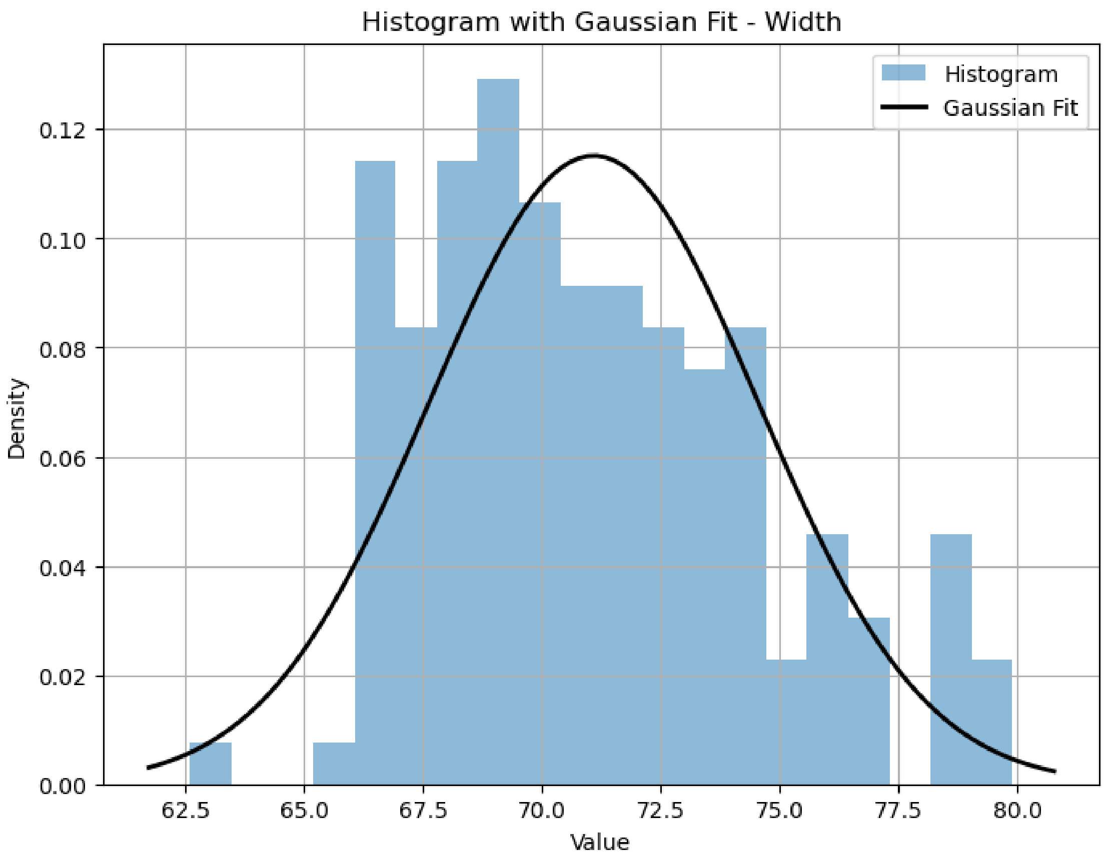
Histogram with Gaussian Fit - __year_resale_value
Histogram
Gaussian Fit
Density
Value
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0
10
20
30
40
50
60
70

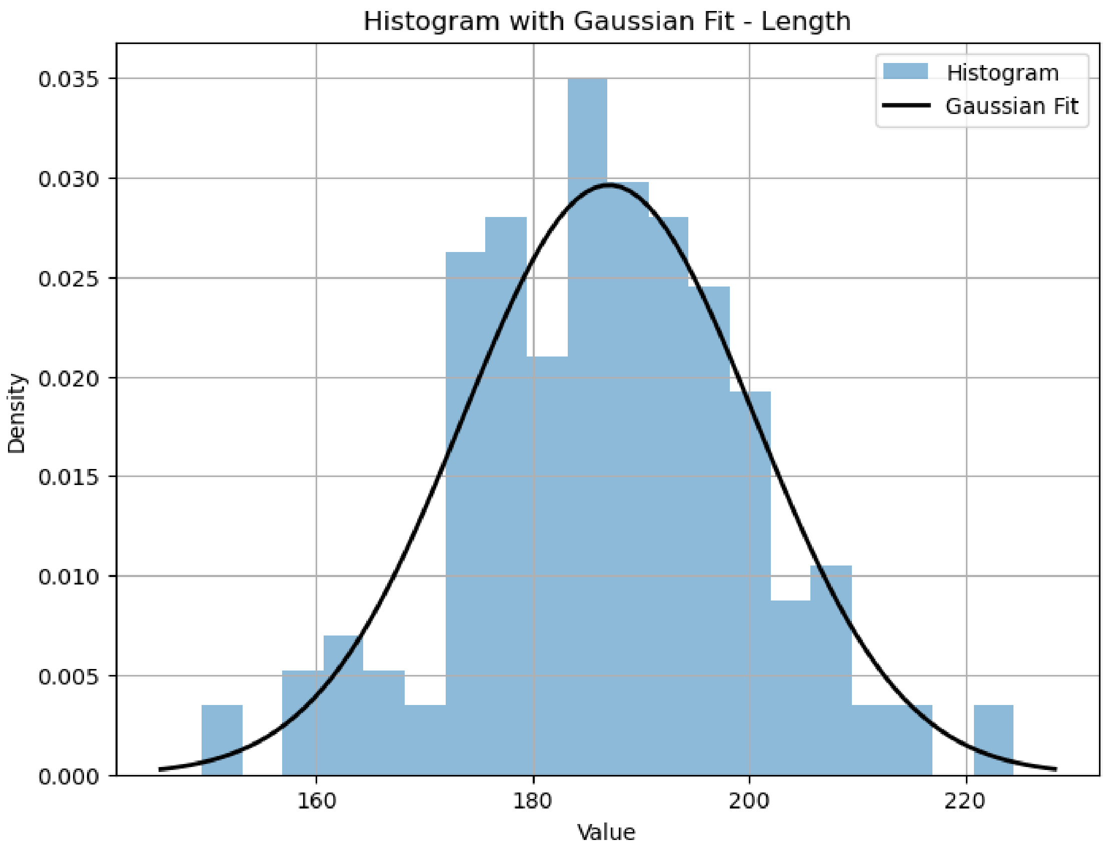

Histogram with Gaussian Fit - Price_in_thousands
Histogram
Gaussian Fit
Density
Value
0.00
0.01
0.02
0.03
0.04
0.05
20
40
60
80

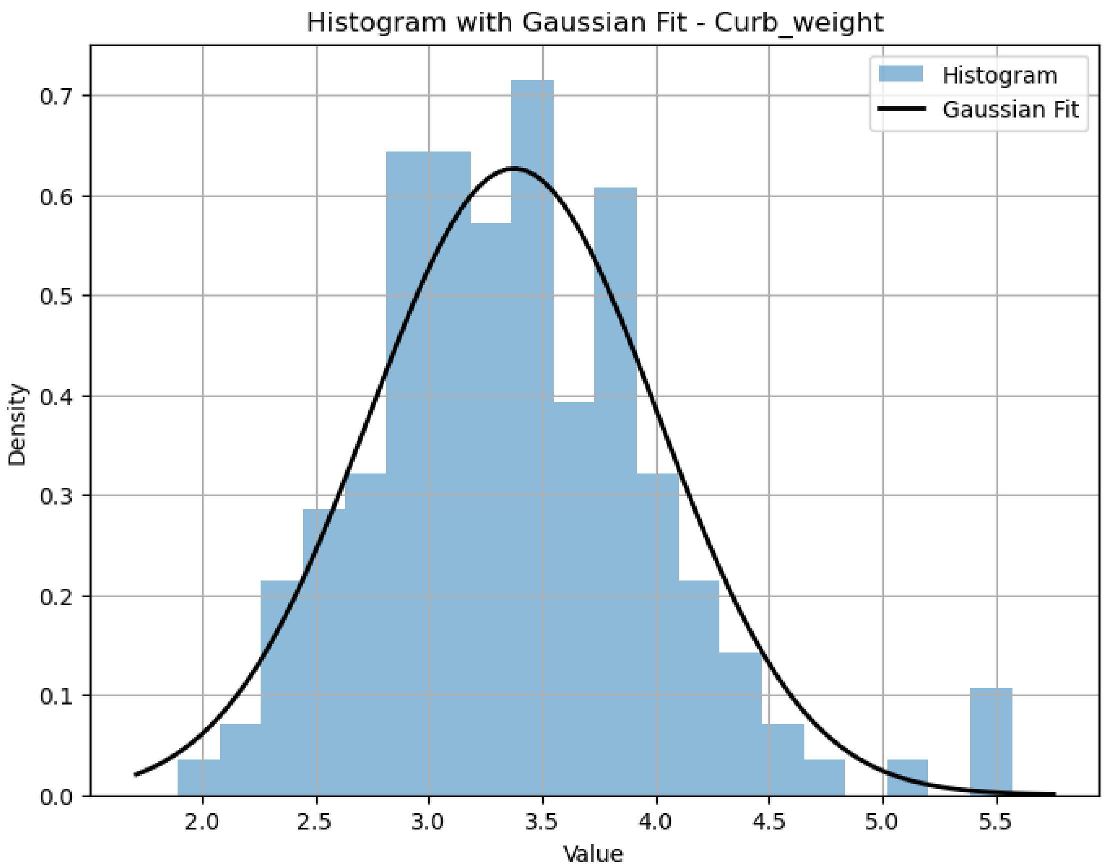

Histogram with Gaussian Fit - Engine_size
Histogram
Gaussian Fit
Density
Value
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
6
7
8

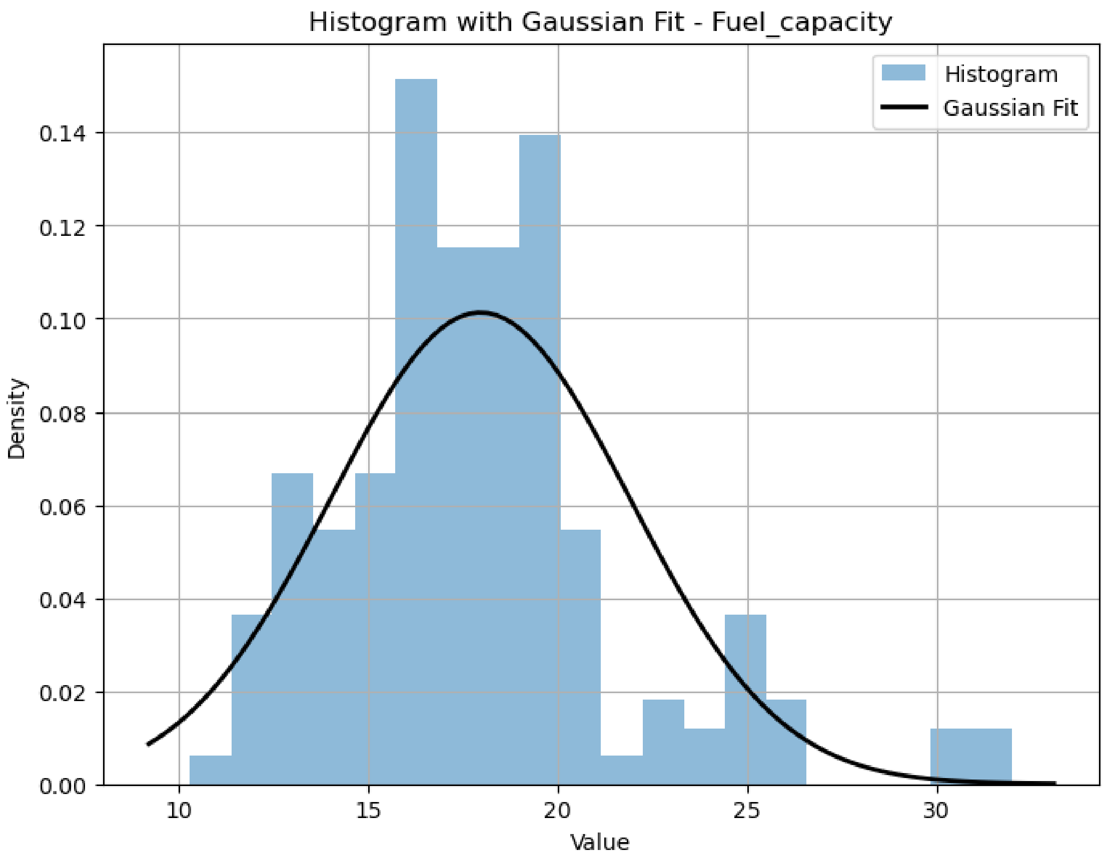

Histogram with Gaussian Fit - Horsepower
Histogram
Gaussian Fit
Density
0.000
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
100
200
300
400
Value

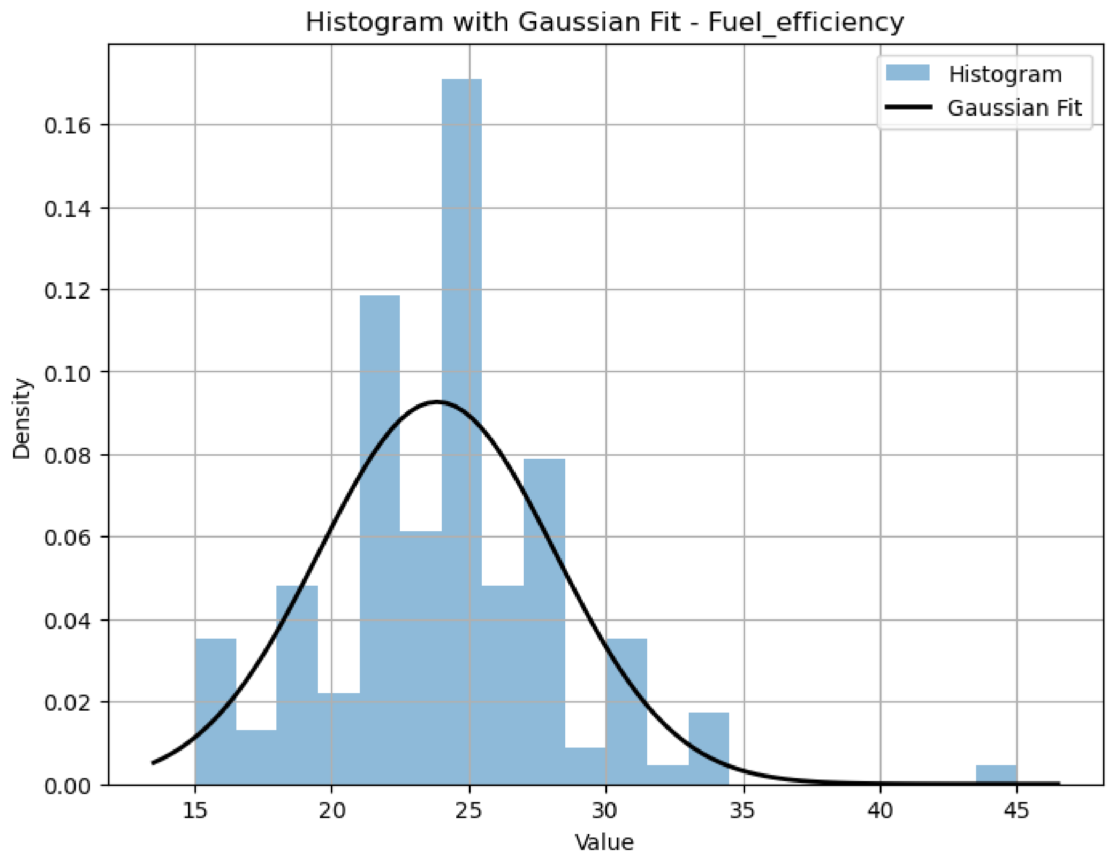

Histogram with Gaussian Fit - Wheelbase
Histogram
Gaussian Fit
Density
Value
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
90
100
110
120
130
140

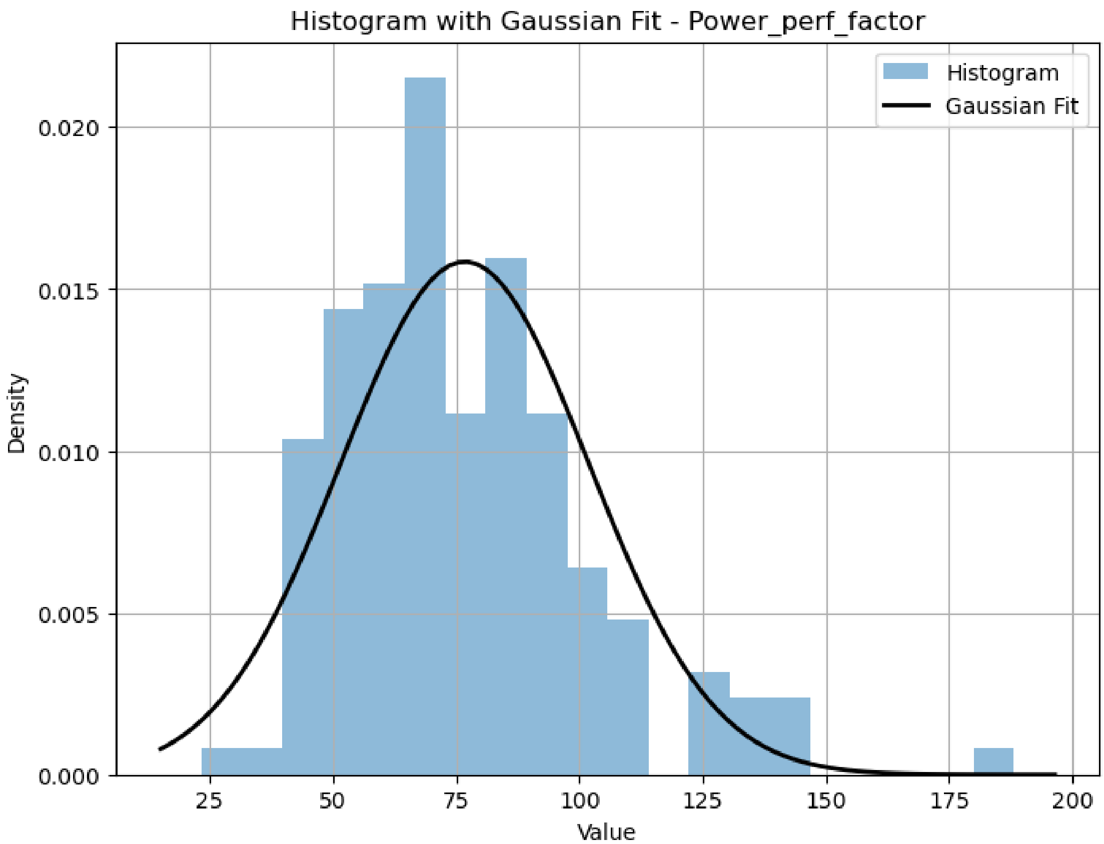

Histogram with Gaussian Fit - Width
Histogram
Gaussian Fit
Density
Value
0.00
0.02
0.04
0.06
0.08
0.10
0.12
62.5
65.0
67.5
70.0
72.5
75.0
77.5
80.0


Histogram with Gaussian Fit - Length
Histogram
Gaussian Fit
Density
Value
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
160
180
200
220


Histogram with Gaussian Fit - Curb_weight
Histogram
Gaussian Fit
Density
Value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5


Histogram with Gaussian Fit - Fuel_capacity
Histogram
Gaussian Fit
Density
Value
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
10
15
20
25
30


Histogram with Gaussian Fit - Fuel_efficiency
Histogram
Gaussian Fit
Density
Value
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
15
20
25
30
35
40
45

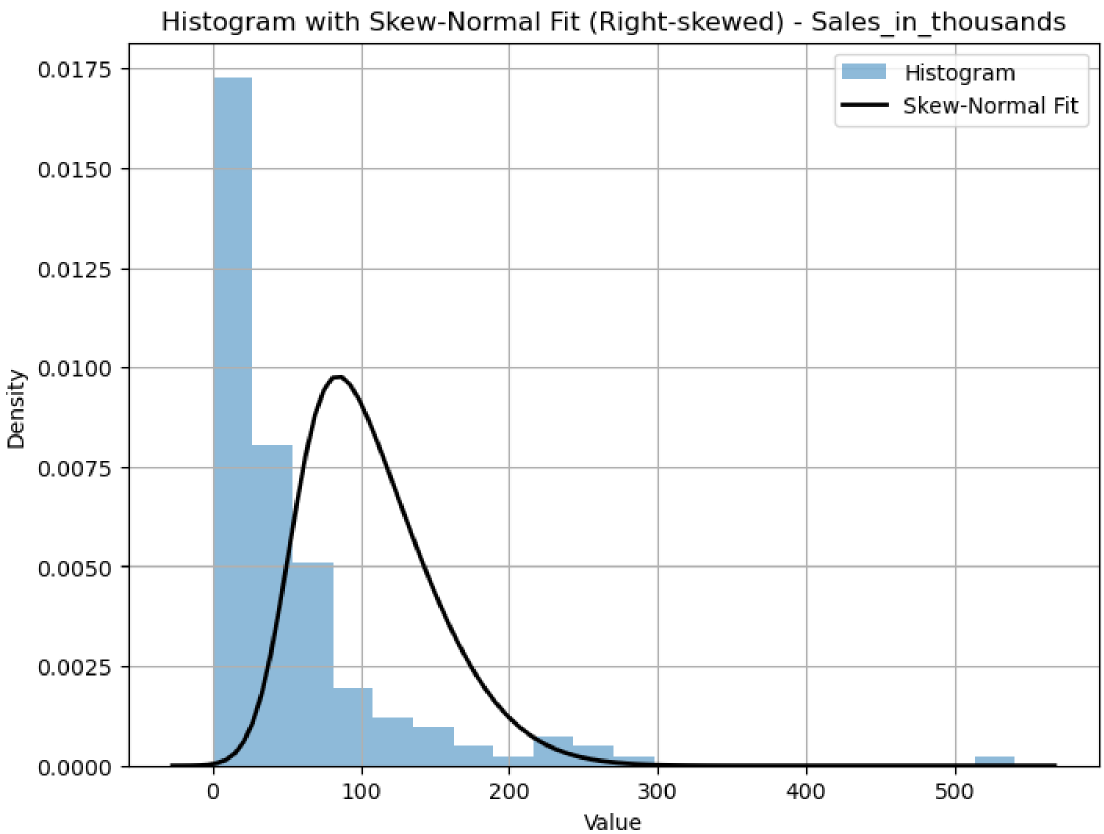


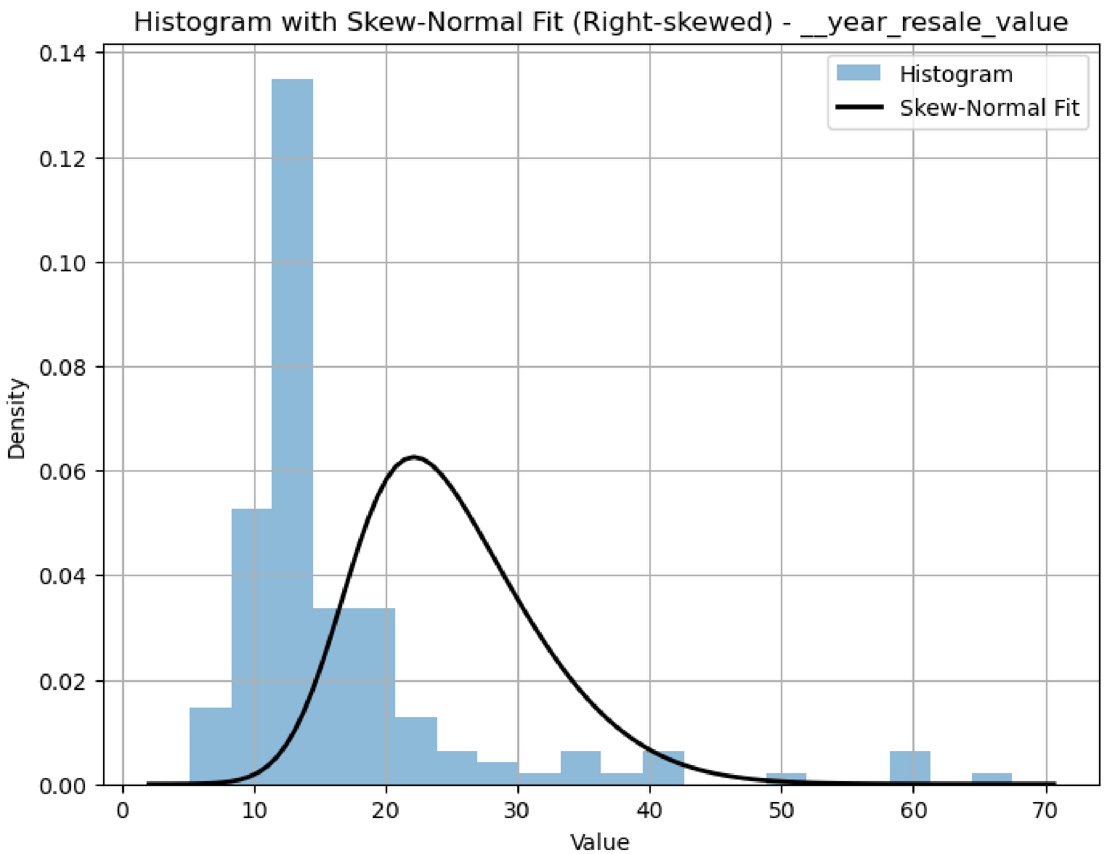
0.7 Skewed Distribution (negative and positive)

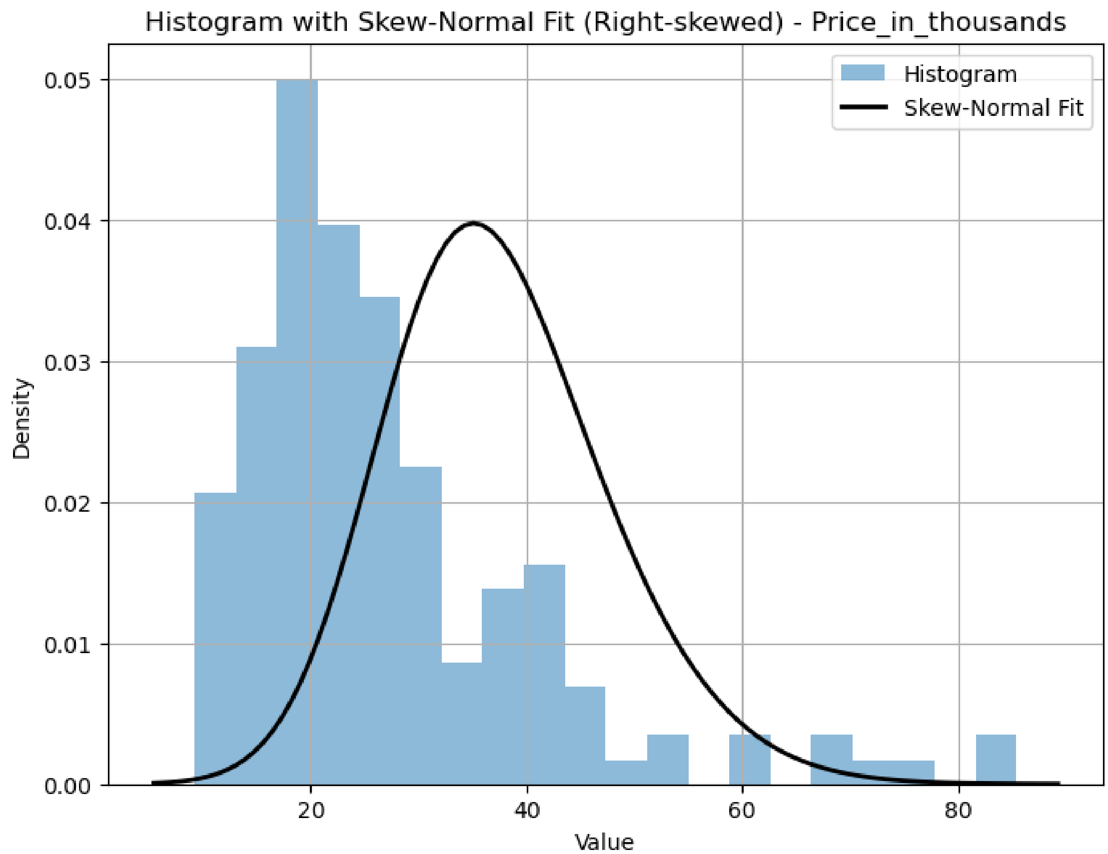
[37]:

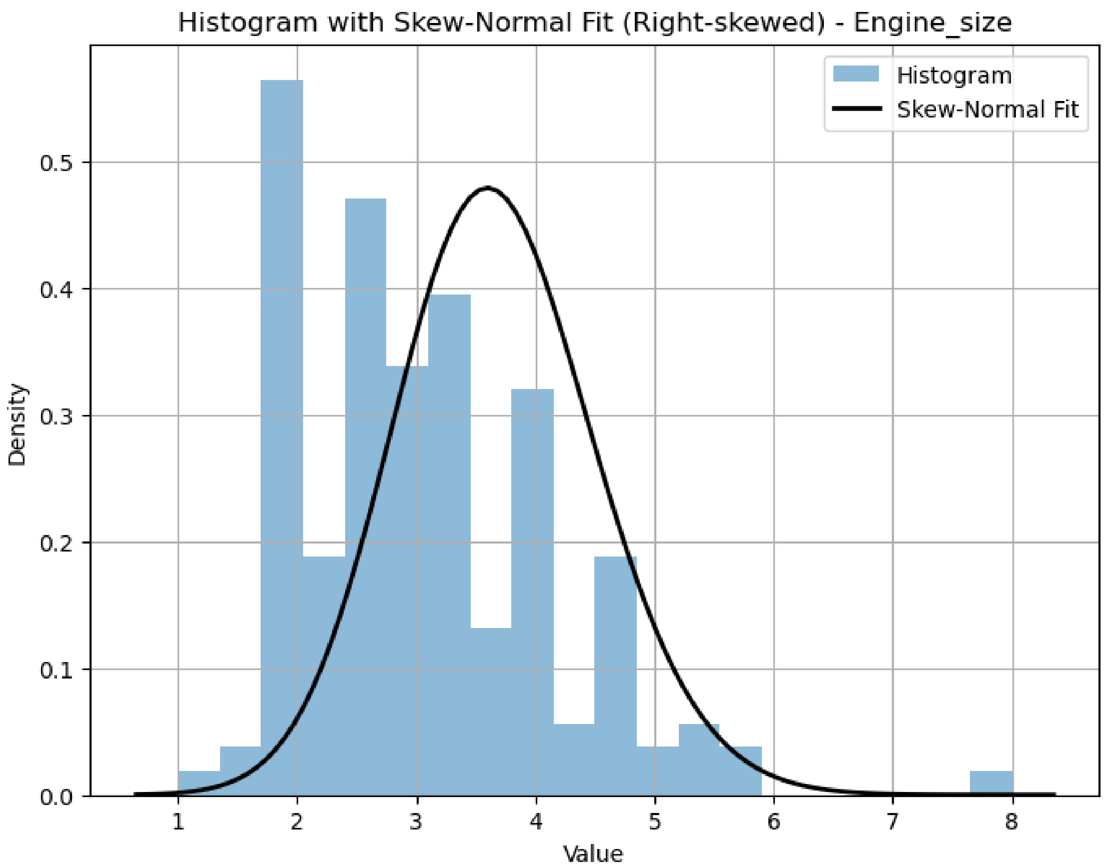
```
means = df2.mean()
std_devs =df2.std()
skews = df2.skew()


for col in df2.columns:
    plt.figure(figsize=(8, 6))

    # Plot histogram
    plt.hist(df2[col], bins=20, density=True, alpha=0.5, label='Histogram')

    # Fit skew-normal distribution
    alpha = skews[col]   # Skewness
    mu, sigma = means[col], std_devs[col]
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    p = skewnorm.pdf(x, alpha, loc=mu, scale=sigma)
    plt.plot(x, p, 'k', linewidth=2, label='Skew-Normal Fit')
```

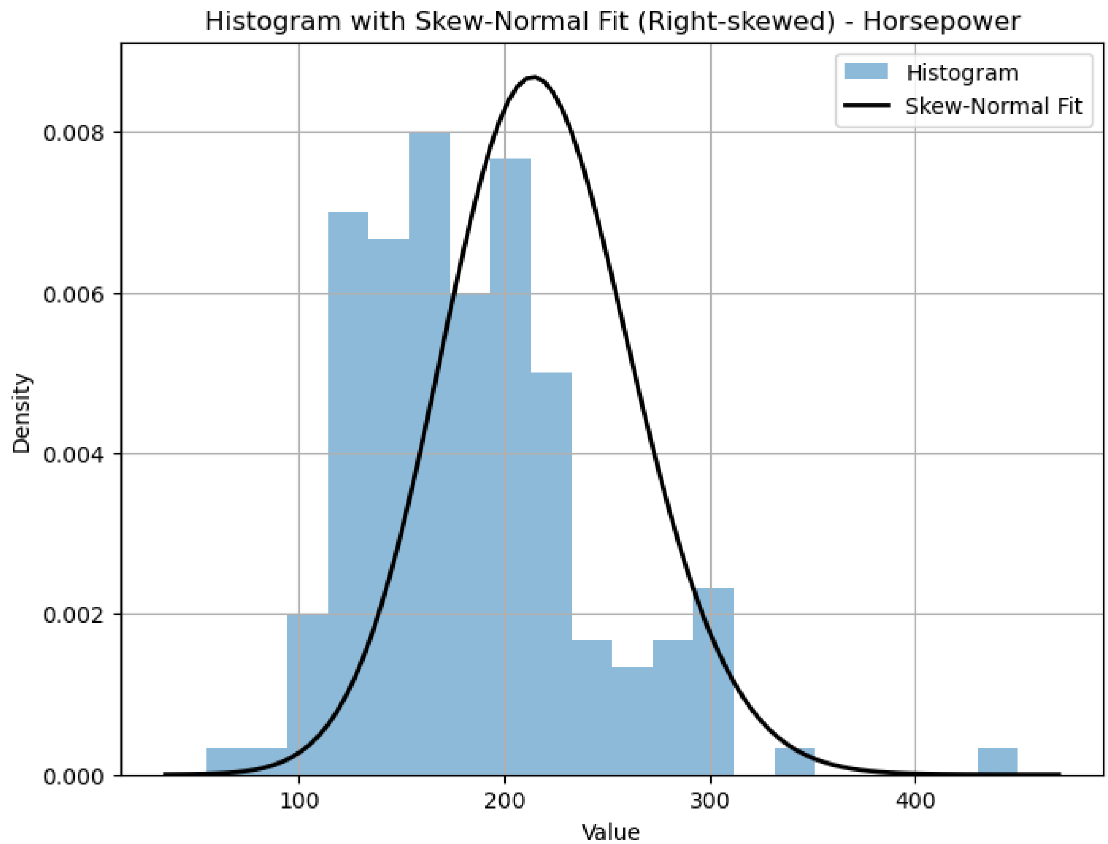
```python
if alpha > 0:
    skew_label = 'Right-skewed'
elif alpha < 0:
    skew_label = 'Left-skewed'
else:
    skew_label = 'Symmetric'

plt.title(f'Histogram with Skew-Normal Fit ({skew_label}) - {col}')
plt.xlabel('Value')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.show()
```

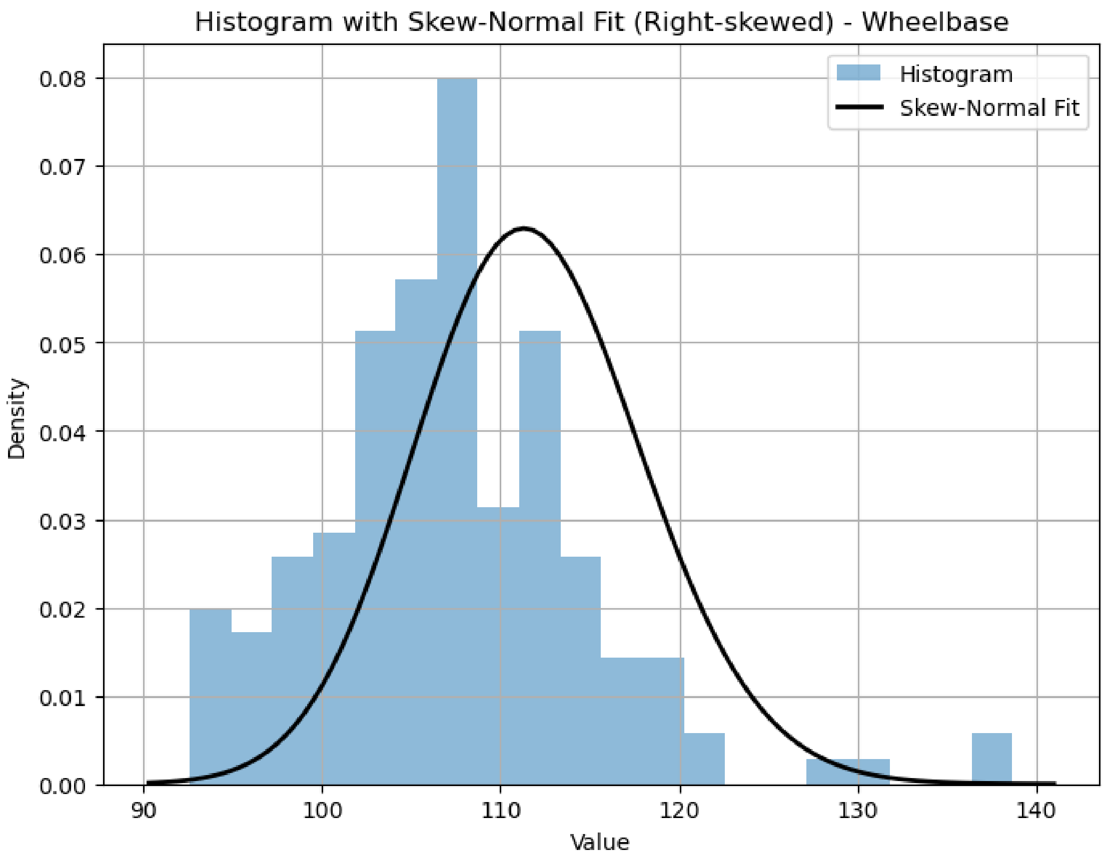


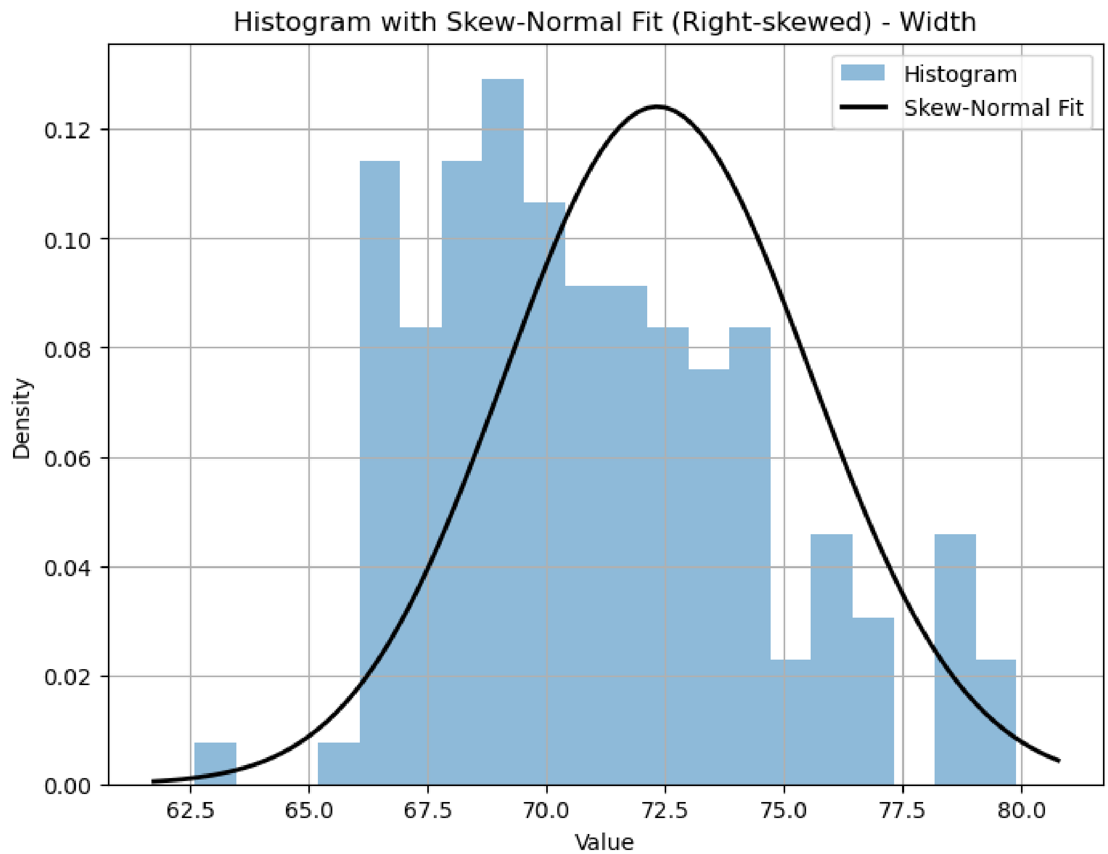

Histogram with Skew-Normal Fit (Right-skewed) - __year_resale_value
Histogram
Skew-Normal Fit
Density
Value
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0
10
20
30
40
50
60
70


Histogram with Skew-Normal Fit (Right-skewed) - Price_in_thousands
Histogram
Skew-Normal Fit
Density
Value
0.05
0.04
0.03
0.02
0.01
0.00
20
40
60
80

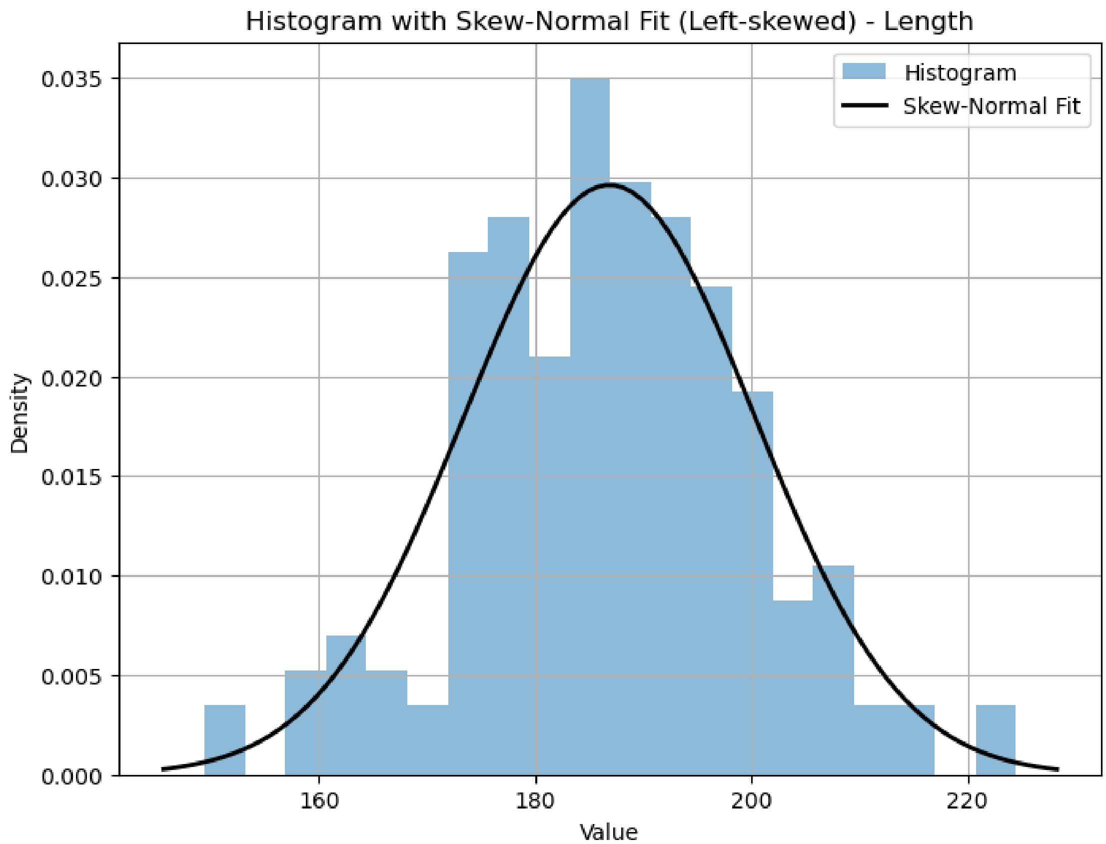

Histogram with Skew-Normal Fit (Right-skewed) - Engine_size
Histogram
Skew-Normal Fit
Density
Value
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
6
7
8

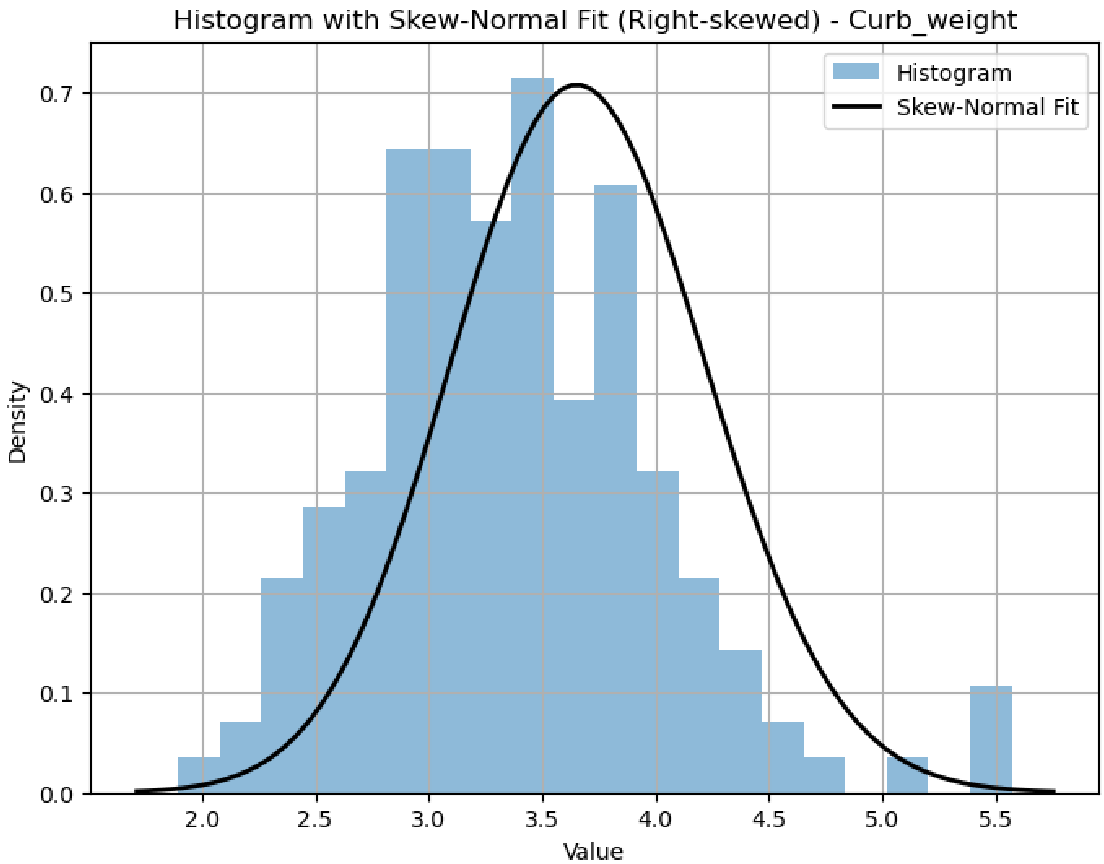

Histogram with Skew-Normal Fit (Right-skewed) - Horsepower
Histogram
Skew-Normal Fit
Density
0.000
0.002
0.004
0.006
0.008
100
200
300
400
Value

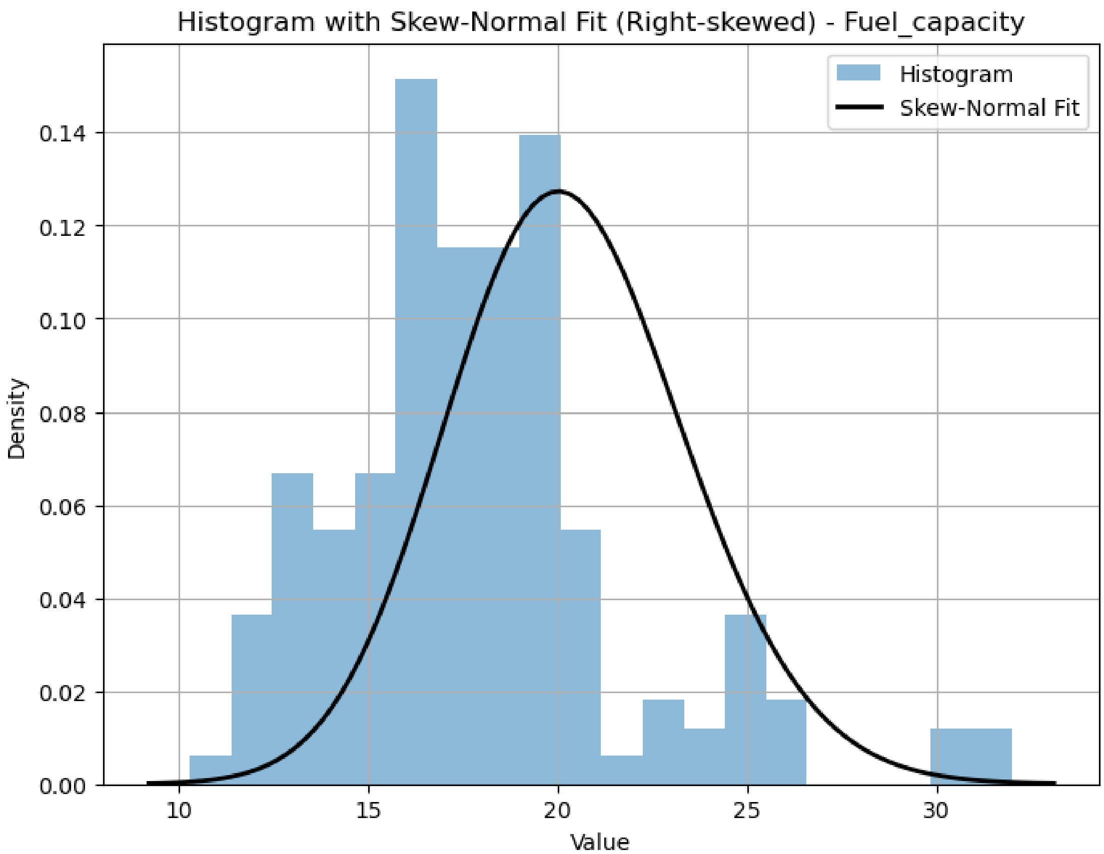

Histogram with Skew-Normal Fit (Right-skewed) - Wheelbase
Histogram
Skew-Normal Fit
Density
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
90
100
110
120
130
140
Value

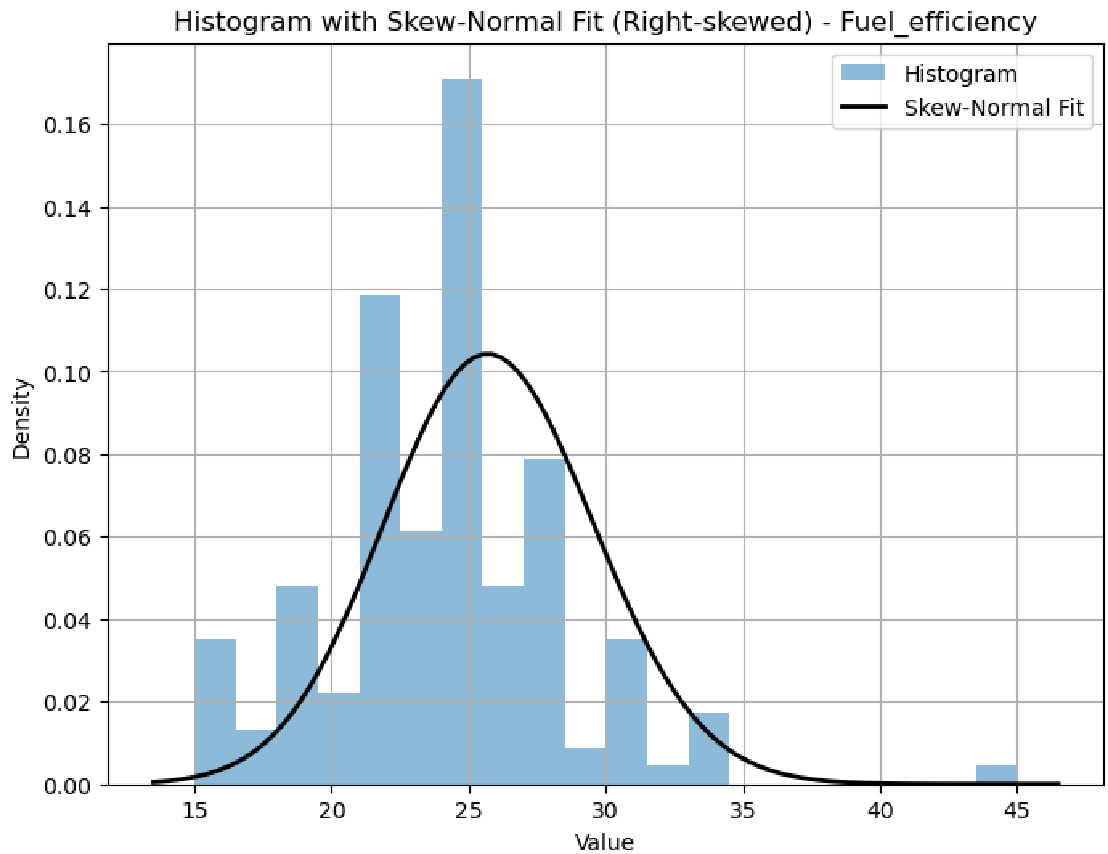


Histogram with Skew-Normal Fit (Right-skewed) - Width
Histogram
Skew-Normal Fit
Density
Value
0.00
0.02
0.04
0.06
0.08
0.10
0.12
62.5
65.0
67.5
70.0
72.5
75.0
77.5
80.0

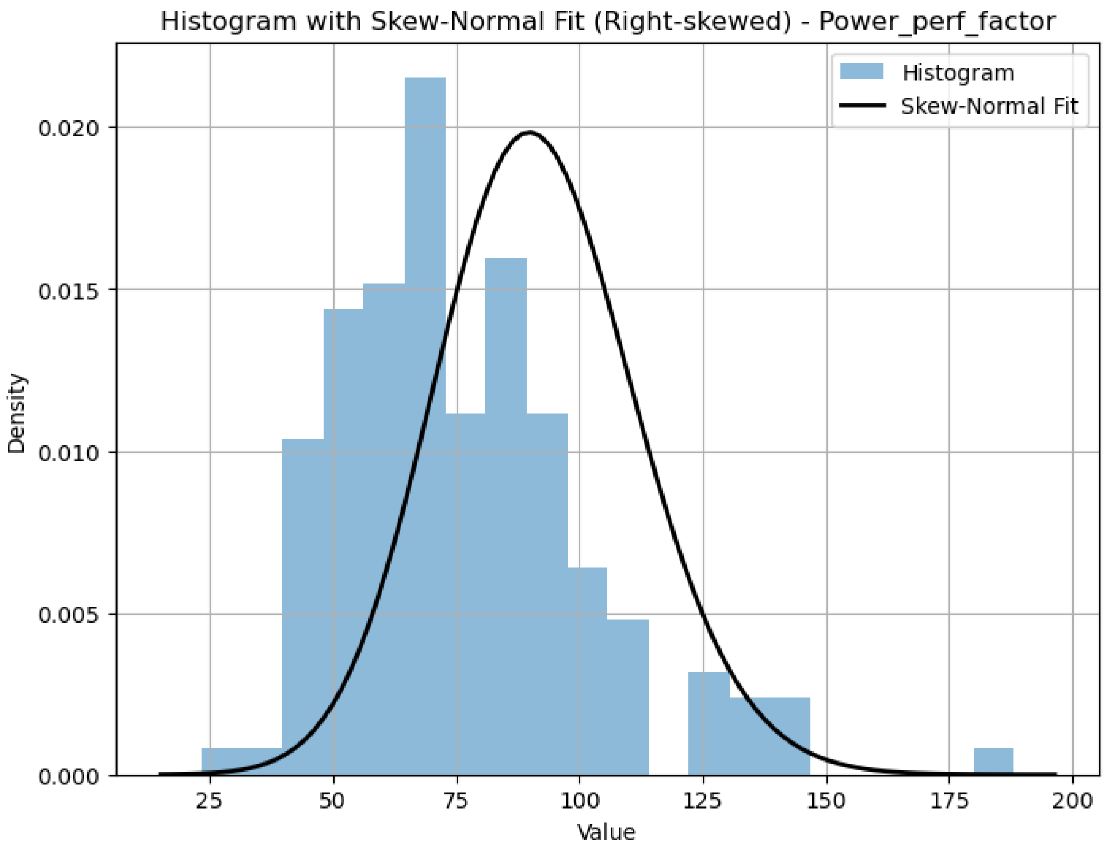

Histogram with Skew-Normal Fit (Left-skewed) - Length
Histogram
Skew-Normal Fit
Density
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
160
180
200
220
Value


Histogram with Skew-Normal Fit (Right-skewed) - Curb_weight
Histogram
Skew-Normal Fit
Density
Value
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5


Histogram with Skew-Normal Fit (Right-skewed) - Fuel_capacity
Histogram
Skew-Normal Fit
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
10
15
20
25
30
Value



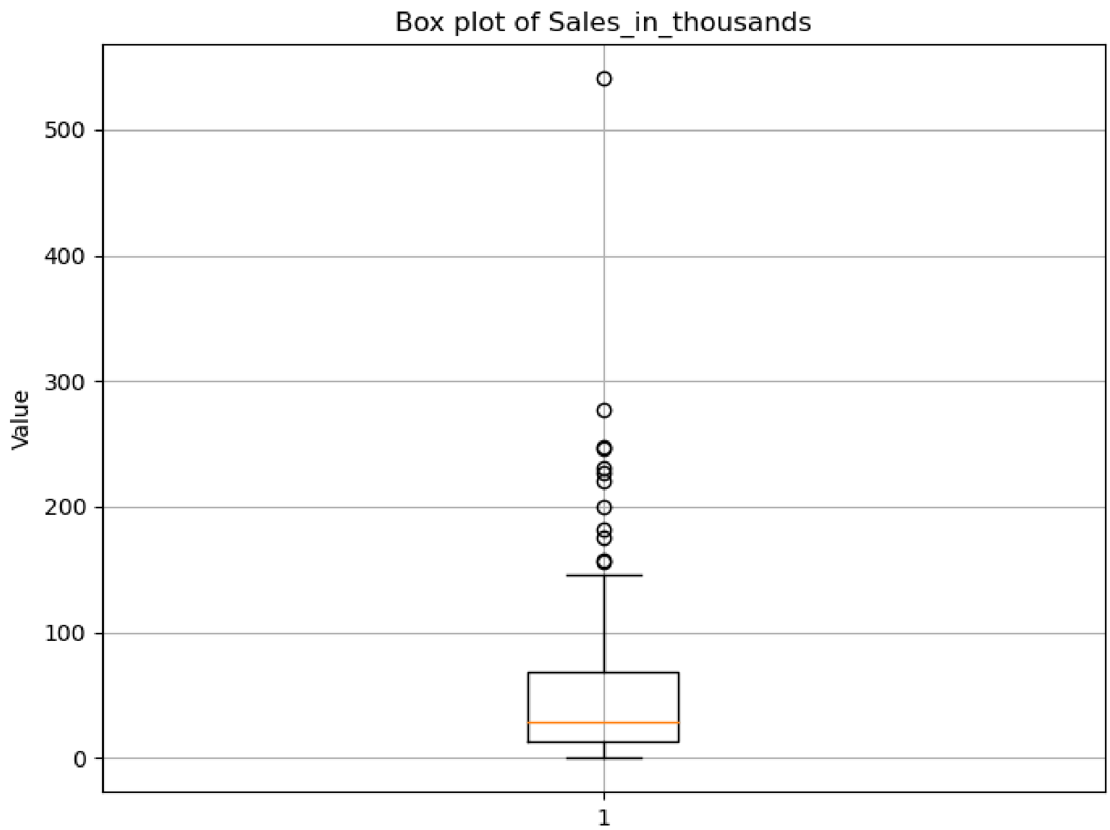
Histogram with Skew-Normal Fit (Right-skewed) - Fuel_efficiency
Histogram
Skew-Normal Fit
Density
Value
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
15
20
25
30
35
40
45

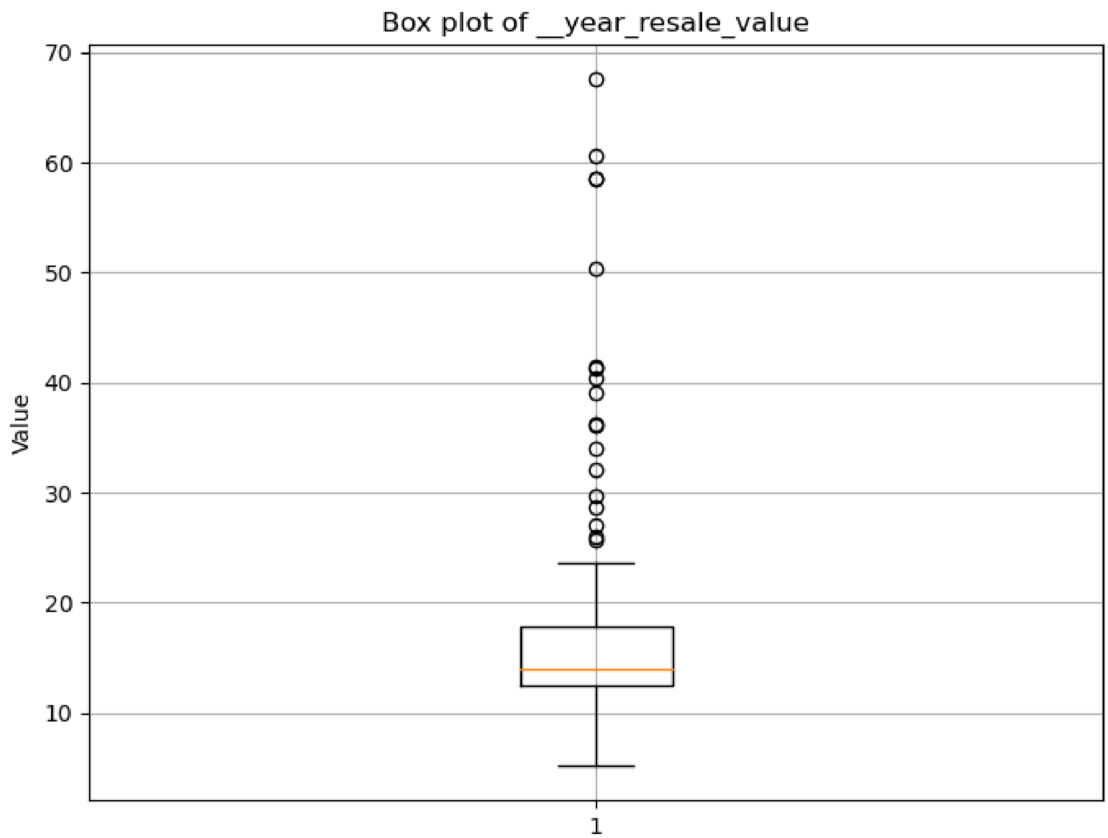


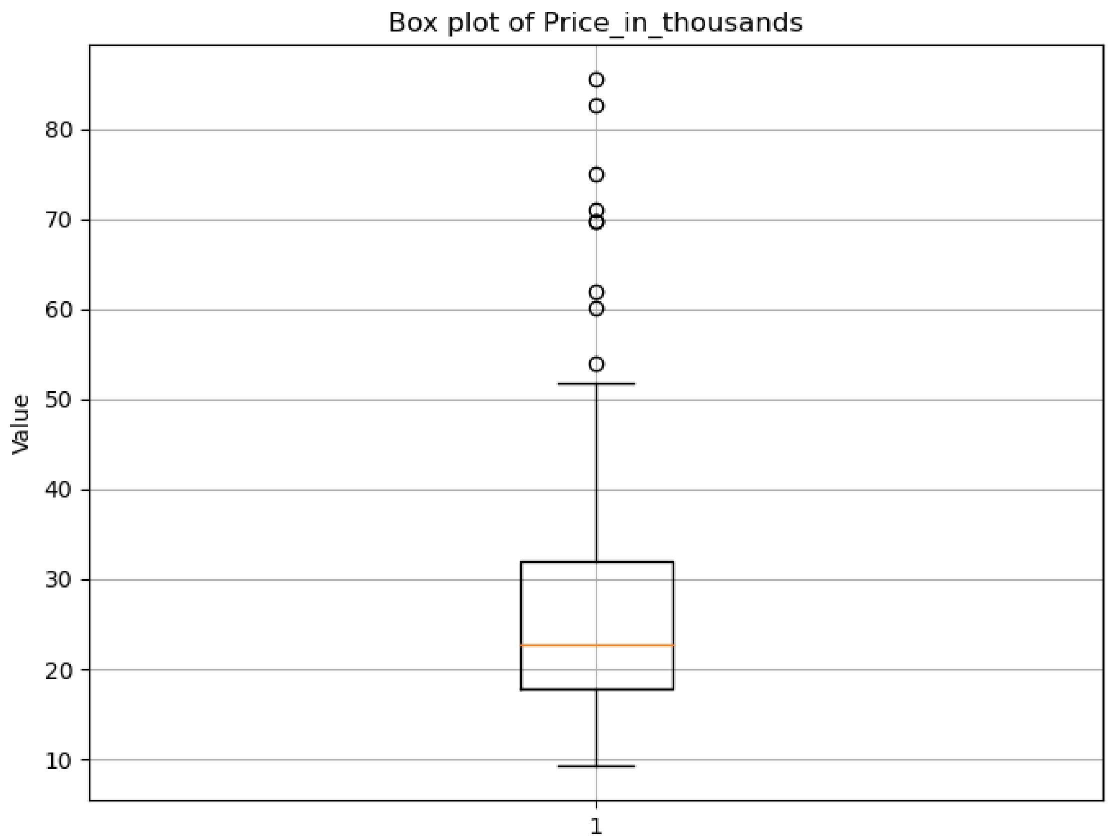
0.8 Outliers

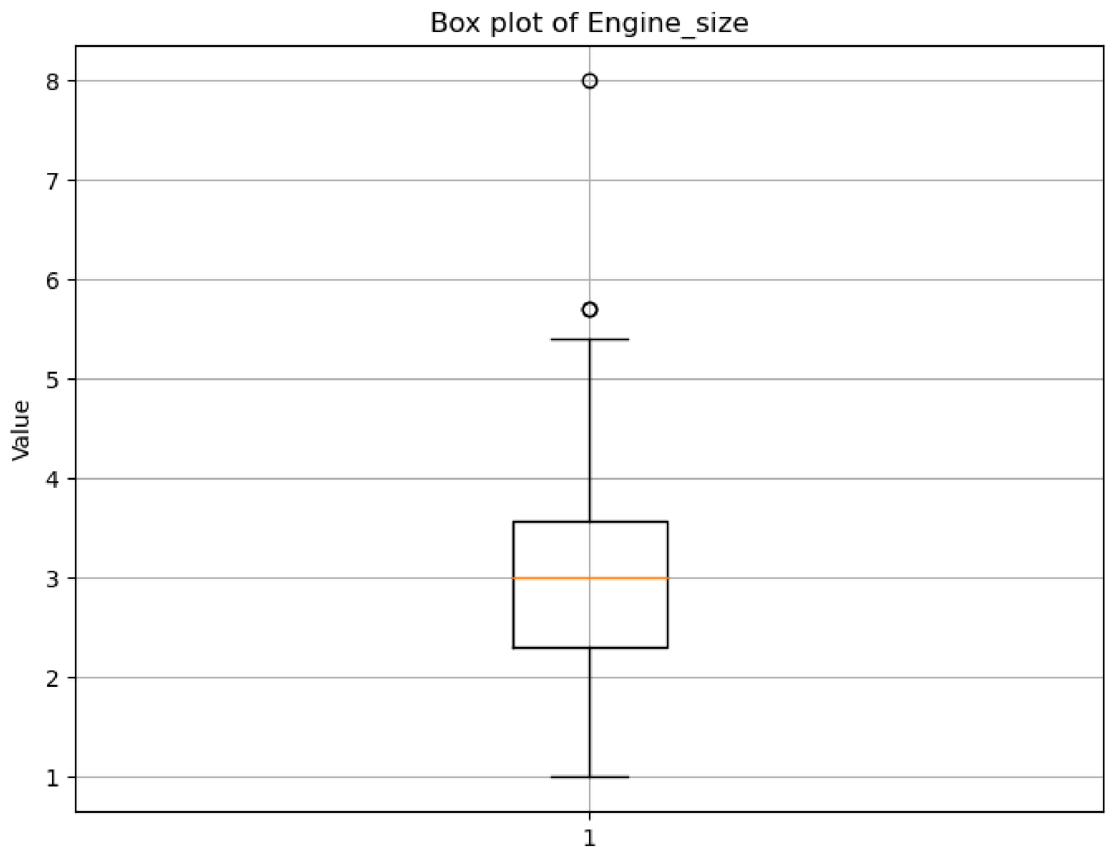
Plot Boxplot

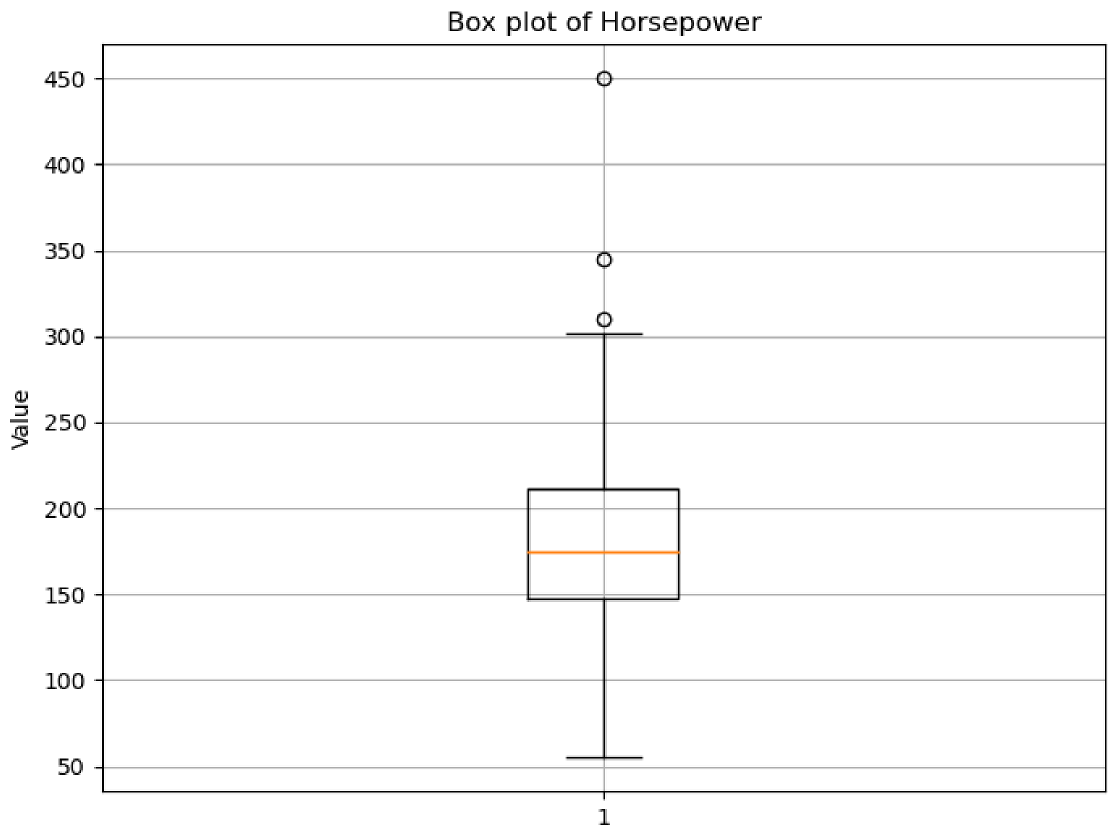
[39]:
```
for col in df2.columns:
    plt.figure(figsize=(8, 6))
    plt.boxplot(df2[col])
    plt.title(f'Box plot of {col}')
    plt.ylabel('Value')
    plt.grid(True)
    plt.show()
```

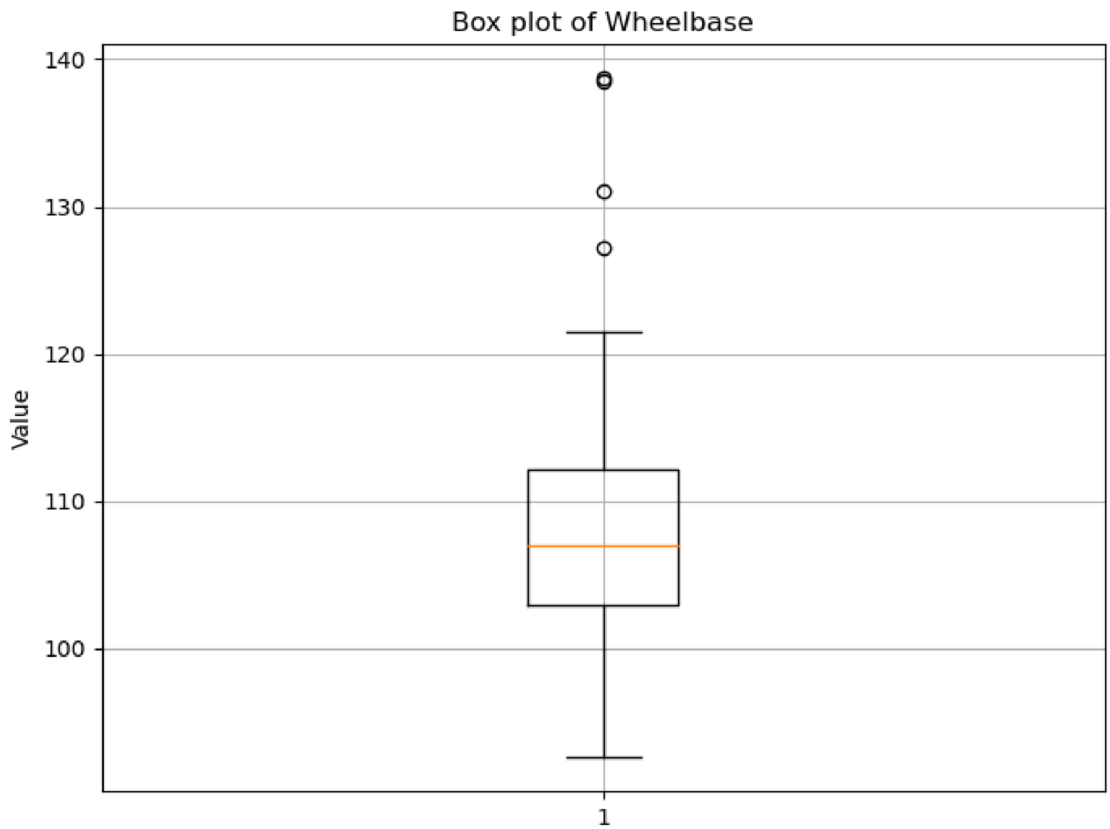

Box plot of Sales_in_thousands
Value
0
100
200
300
400
500
1

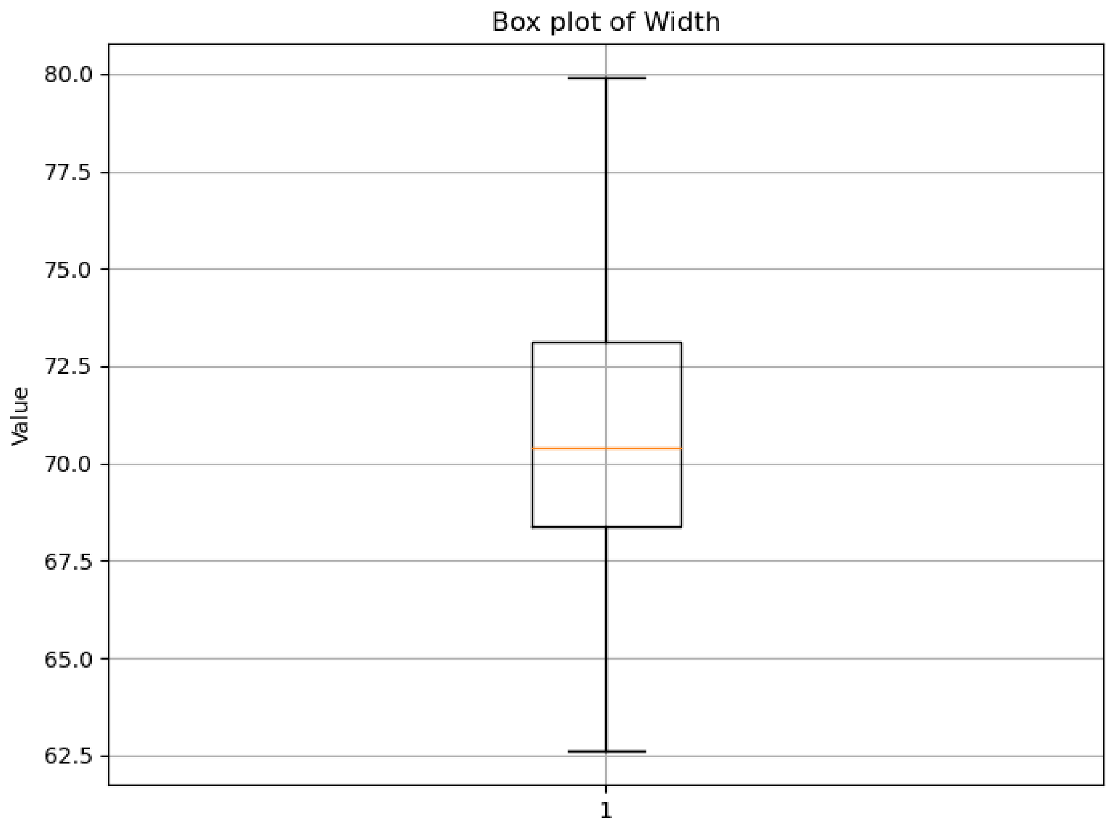

Box plot of __year_resale_value
Value
70
60
50
40
30
20
10
1


Box plot of Price_in_thousands
Value
80
70
60
50
40
30
20
10
1

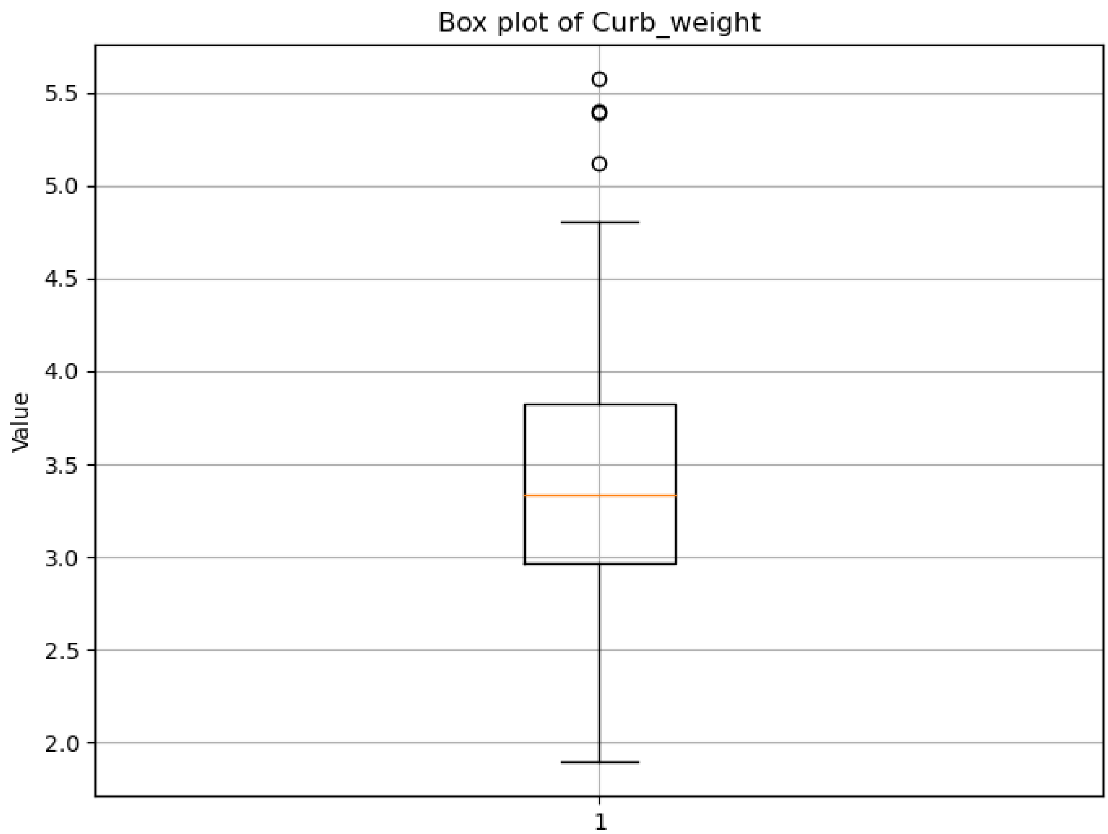

Box plot of Engine_size
Value
8
7
6
5
4
3
2
1
1

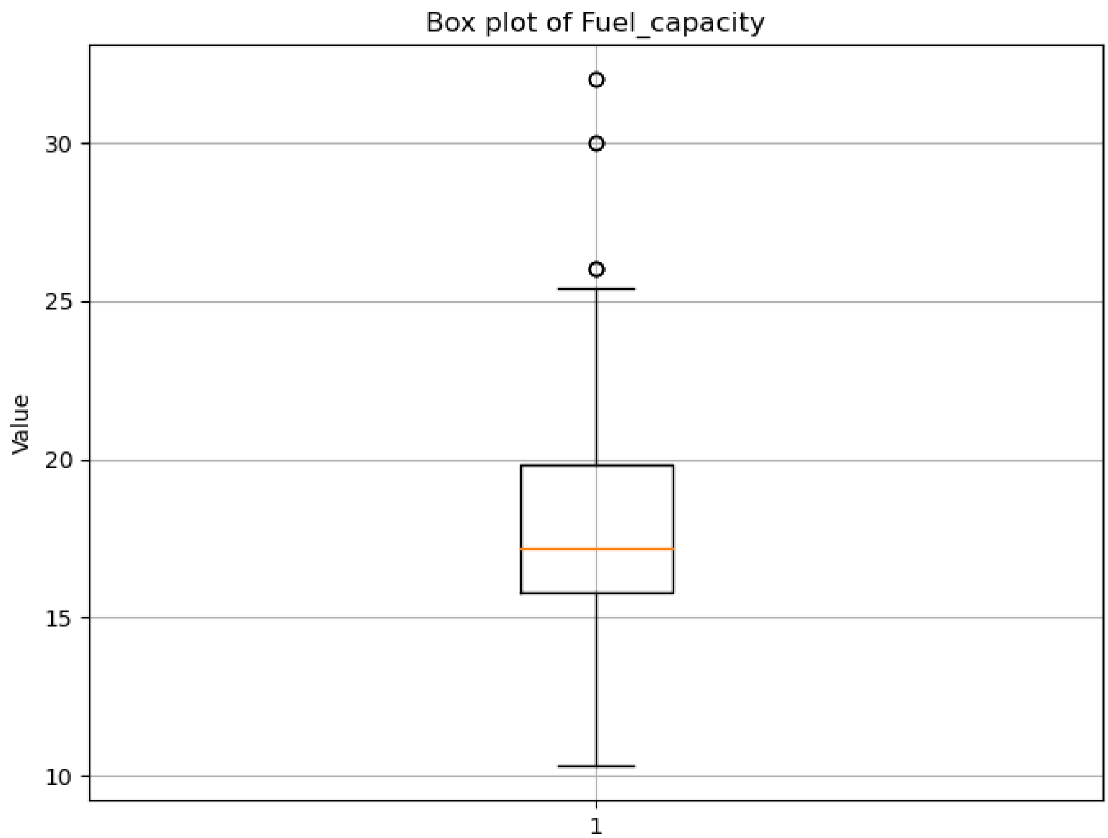

Box plot of Horsepower
Value
450
400
350
300
250
200
150
100
50
1

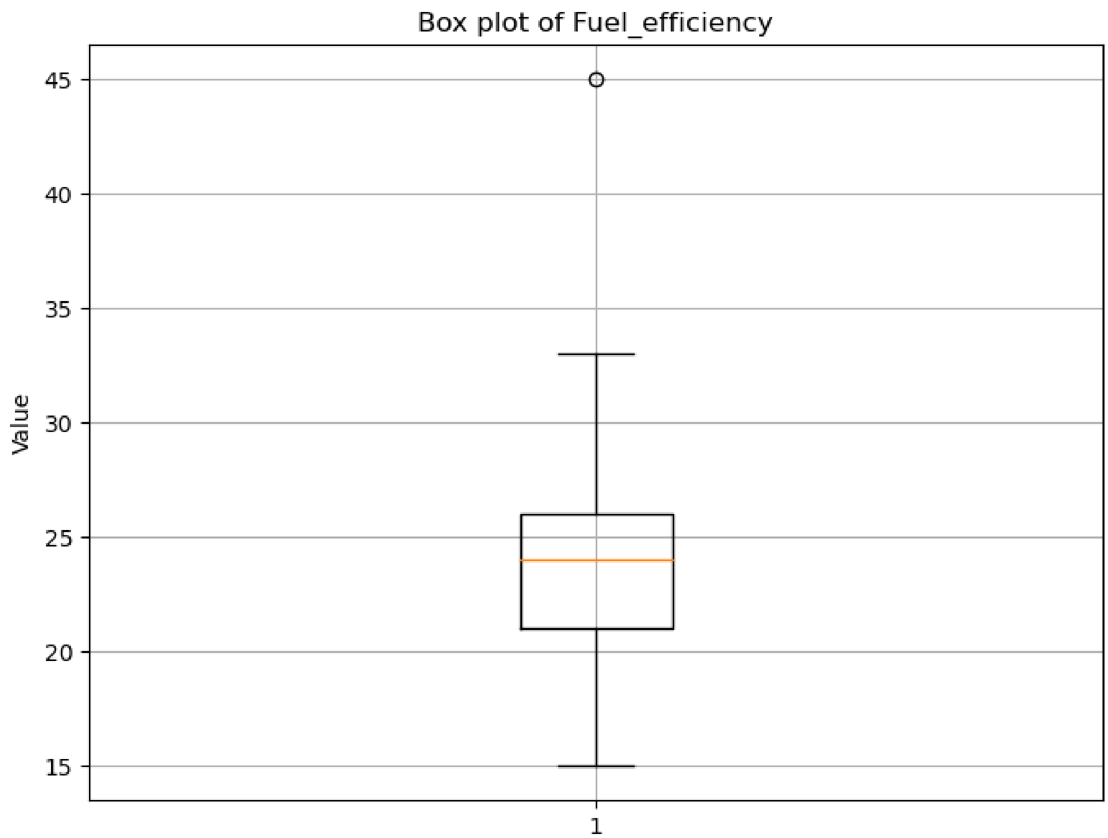

Box plot of Wheelbase
140
130
120
110
100
Value
1

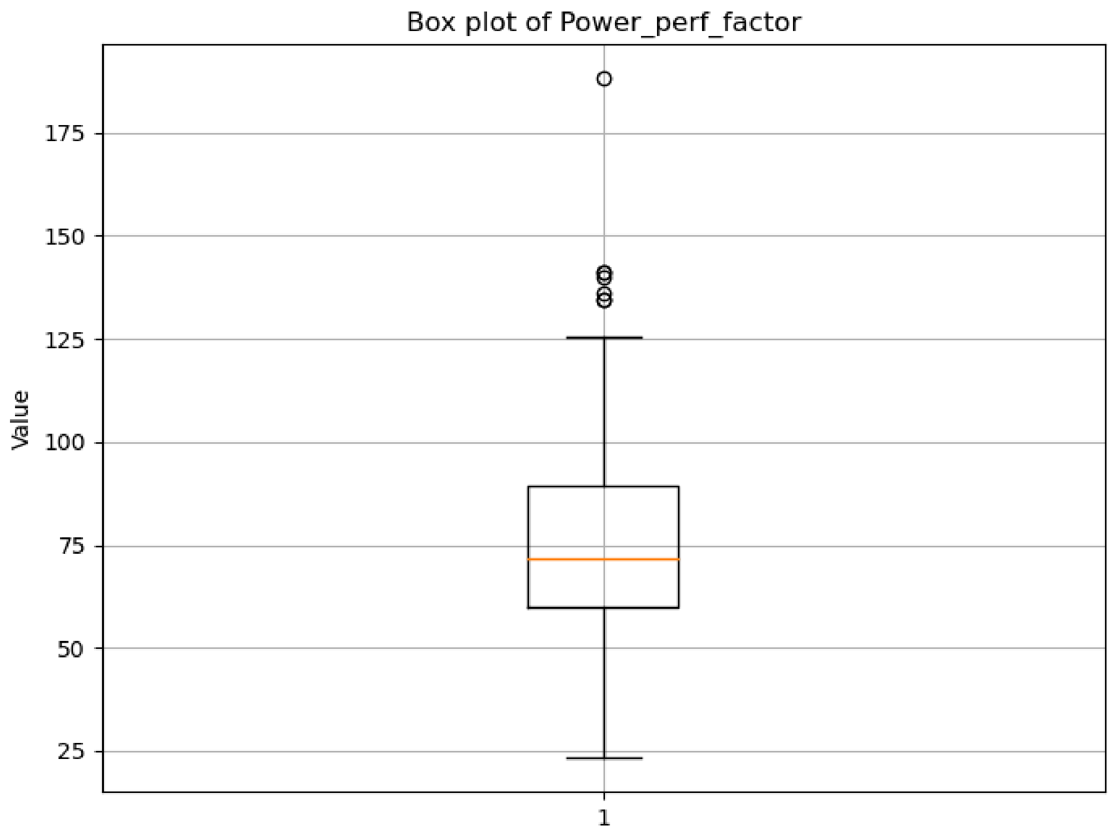

Box plot of Width
80.0
77.5
75.0
72.5
70.0
67.5
65.0
62.5
Value
1


Box plot of Length
Value
220
210
200
190
180
170
160
150
1


Box plot of Curb_weight
Value
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1


Box plot of Fuel_capacity
Value
30
25
20
15
10
1


Box plot of Fuel_efficiency
Value
45
40
35
30
25
20
15
1



0.9 Z-score

Detect Outliers using Z-Score. (Set threshold =3)

[41]:
```
z_scores = (df2 - df2.mean()) / df2.std()
outliers = np.abs(z_scores) > 3

outliers_data = df2[outliers.any(axis=1)]
print("Outliers:")
print(outliers_data)
```

```
Outliers:
    Sales_in_thousands  __year_resale_value  Price_in_thousands  Engine_size  \
18              14.785               14.010              46.225          5.7
26              21.855                5.160               9.235          1.0
39               0.916               58.470              69.725          8.0
40             227.061               15.060              19.460          5.2
41              16.767               15.510              21.315          3.9
42              31.038               13.425              18.575          3.9
43             111.313               11.260              16.980          2.5
52             276.747               16.640              31.930          4.0
```

```
56          540.561           15.075           26.935         4.6
74            9.126           14.010           60.105         4.7
78           22.925           14.010           42.660         5.4
94           16.774           50.375           69.700         4.3
95            3.311           58.600           82.600         5.0
99            0.954           14.010           85.500         5.0
125           1.280           60.625           71.020         3.4
126           1.866           67.550           74.970         3.4

     Horsepower  Wheelbase  Width  Length  Curb_weight  Fuel_capacity  \
18        255.0      117.5   77.0   201.2        5.572           30.0
26         55.0       93.1   62.6   149.4        1.895           10.3
39        450.0       96.2   75.7   176.7        3.375           19.0
40        230.0      138.7   79.3   224.2        4.470           26.0
41        175.0      109.6   78.8   192.6        4.245           32.0
42        175.0      127.2   78.8   208.5        4.298           32.0
43        120.0      131.0   71.5   215.0        3.557           22.0
52        210.0      111.6   70.2   190.7        3.876           21.0
56        220.0      138.5   79.1   224.5        4.241           25.1
74        230.0      112.2   76.4   192.5        5.401           25.4
78        300.0      119.0   79.9   204.8        5.393           30.0
94        275.0      121.5   73.1   203.1        4.133           23.2
95        302.0       99.0   71.3   177.1        4.125           21.1
99        302.0      113.6   73.1   196.6        4.115           23.2
125       300.0       92.6   69.5   174.5        3.032           17.0
126       300.0       92.6   69.5   174.5        3.075           17.0

     Fuel_efficiency  Power_perf_factor
18              15.0         109.509117
26              45.0          23.276272
39              16.0         188.144323
40              17.0          90.211700
41              15.0          71.135292
42              16.0          70.078322
43              19.0          49.645002
52              19.0          87.635496
56              18.0          89.401935
74              15.0         105.760458
78              15.0         123.972047
94              21.0         125.273876
95              20.0         139.982294
99              20.0         141.100985
125             21.0         134.390975
126             23.0         135.914710
```

[65]:
```
outlier_percentages = (outliers.sum() / df2.shape[0]) * 100
```

```
print("Percentage of outliers for each column:")
print(outlier_percentages)
```

```
Percentage of outliers for each column:
Sales_in_thousands      1.315789
__year_resale_value     3.289474
Price_in_thousands      2.631579
Engine_size             0.657895
Horsepower              0.657895
Wheelbase               1.973684
Width                   0.000000
Length                  0.000000
Curb_weight             1.973684
Fuel_capacity           2.631579
Fuel_efficiency         0.657895
Power_perf_factor       0.657895
dtype: float64
```

0.10 Remove outliers

[69]:
```
def remove_outliers_with_iqr(df2):
    for column in df2.select_dtypes(include=['number']).columns:
        Q1 = df2[column].quantile(0.25)  # First quartile (Q1)
        Q3 = df2[column].quantile(0.75)  # Third quartile (Q3)
        IQR = Q3 - Q1  # Interquartile range (IQR)

        lower_bound = Q1 - 1.5 * IQR  # Lower bound
        upper_bound = Q3 + 1.5 * IQR  # Upper bound

        df2 = df2[(df2[column] >= lower_bound) & (df2[column] <= upper_bound)]

    return df2


df3 = remove_outliers_with_iqr(df2)

print("Original shape:", df2.shape)
print("New shape after removing outliers:", df3.shape)
```

```
Original shape: (152, 12)
New shape after removing outliers: (108, 12)
```

[71]:
```
outlier_percentages = {}

# Iterate over each column
for col in df3.columns:
    # Calculate Z-scores for the column
```

```
    z_scores = (df3[col] - df3[col].mean()) / df3[col].std()

    # Identify outliers for the column
    outliers = (np.abs(z_scores) > 3)

    # Calculate percentage of outliers for the column
    percentage_outliers = (outliers.sum() / len(outliers)) * 100

    # Store the percentage of outliers for the column in the dictionary
    outlier_percentages[col] = float(percentage_outliers)  # Convert to Python␣
 ↪float

# Display outlier percentages for each column
print("Percentage of outliers for each column:")
for col, percentage in outlier_percentages.items():
    print(f"{col}: {percentage}%")
```

```
Percentage of outliers for each column:
Sales_in_thousands: 1.8518518518518516%
__year_resale_value: 0.0%
Price_in_thousands: 0.0%
Engine_size: 0.0%
Horsepower: 0.0%
Wheelbase: 0.0%
Width: 0.0%
Length: 0.0%
Curb_weight: 0.0%
Fuel_capacity: 0.0%
Fuel_efficiency: 0.0%
Power_perf_factor: 0.0%
```

Removing the remaining outliers in 'Sales in thousands'

[74]:
```
def remove_outliers_with_tight_iqr(df, column_name, iqr_multiplier=1.25):
    Q1 = df3[column_name].quantile(0.25)  # First quartile (Q1)
    Q3 = df3[column_name].quantile(0.75)  # Third quartile (Q3)
    IQR = Q3 - Q1  # Interquartile range (IQR)

    lower_bound = Q1 - iqr_multiplier * IQR  # Lower bound
    upper_bound = Q3 + iqr_multiplier * IQR  # Upper bound

    filtered_df = df[(df[column_name] >= lower_bound) & (df[column_name] <=␣
 ↪upper_bound)]
    return filtered_df

df4 = remove_outliers_with_tight_iqr(df3, 'Sales_in_thousands',␣
 ↪iqr_multiplier=1.25)
```

```
print("Original shape:", df3.shape)
print("New shape after removing outliers in 'Sales_in_thousands':", df4.shape)
```

```
Original shape: (108, 12)
New shape after removing outliers in 'Sales_in_thousands': (104, 12)
```

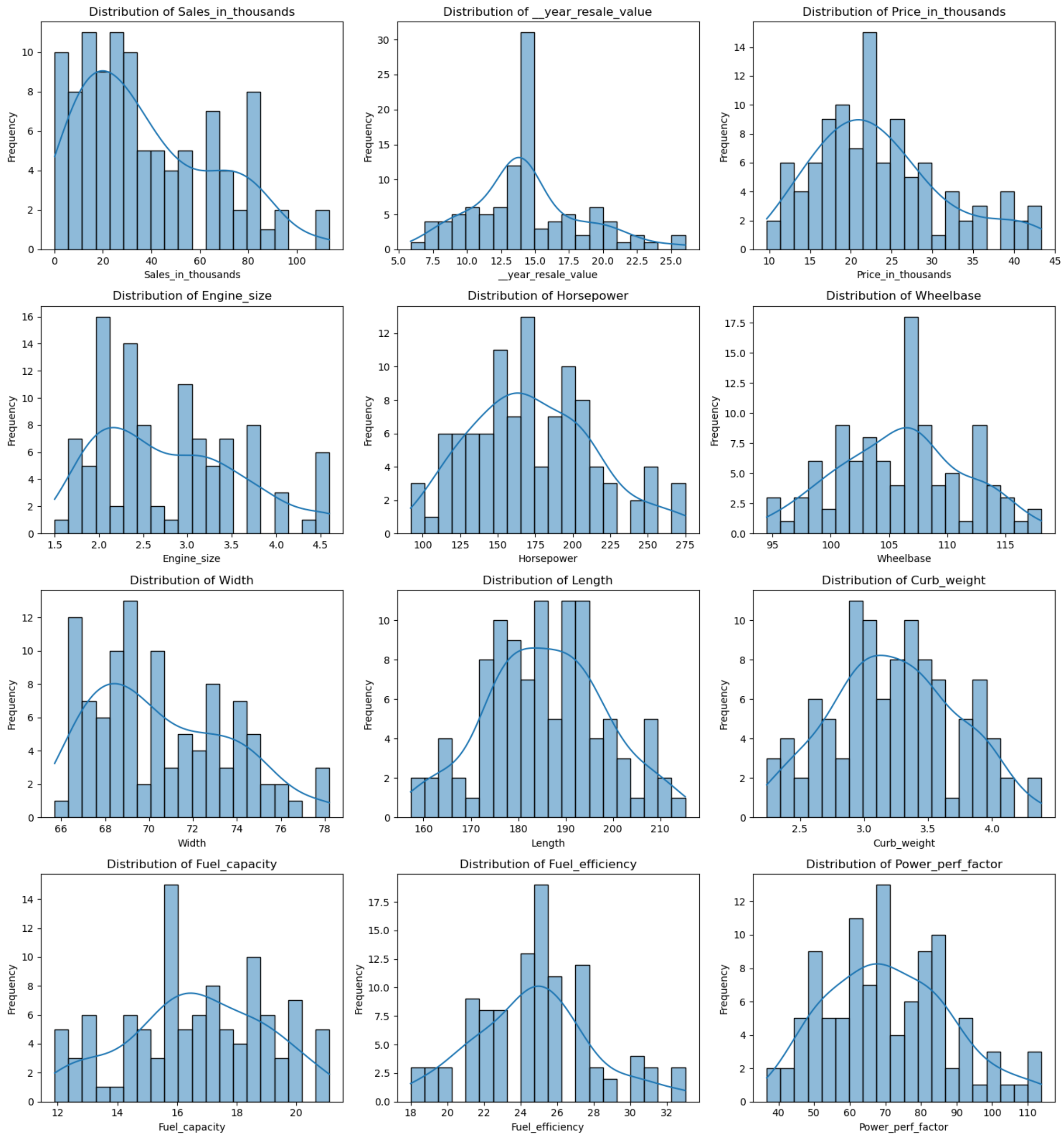
Checking percentages again to confirm

[76]:
```
outlier_percentages = {}


for col in df4.columns:
    # Calculate Z-scores for the column
    z_scores = (df4[col] - df4[col].mean()) / df4[col].std()

    # Identify outliers for the column
    outliers = (np.abs(z_scores) > 3)

    # Calculate percentage of outliers for the column
    percentage_outliers = (outliers.sum() / len(outliers)) * 100

    # Store the percentage of outliers for the column in the dictionary
    outlier_percentages[col] = float(percentage_outliers)  # Convert to Python␣
 ↪float

# Display outlier percentages for each column
print("Percentage of outliers for each column:")
for col, percentage in outlier_percentages.items():
    print(f"{col}: {percentage}%")
```

```
Percentage of outliers for each column:
Sales_in_thousands: 0.0%
__year_resale_value: 0.0%
Price_in_thousands: 0.0%
Engine_size: 0.0%
Horsepower: 0.0%
Wheelbase: 0.0%
Width: 0.0%
Length: 0.0%
Curb_weight: 0.0%
Fuel_capacity: 0.0%
Fuel_efficiency: 0.0%
Power_perf_factor: 0.0%
```

0.11 Distribution Check

[79]:
```python
def plot_distributions(df4):
    num_columns = df4.columns
    num_plots = len(num_columns)
    cols = 3
    rows = num_plots // cols + (num_plots % cols > 0)

    plt.figure(figsize=(15, rows * 4))
    for i, column in enumerate(num_columns):
        plt.subplot(rows, cols, i + 1)
        sns.histplot(df4[column], kde=True, bins=20)
        plt.title(f'Distribution of {column}')
        plt.xlabel(column)  # Label for x-axis
        plt.ylabel('Frequency')  # Label for y-axis
    plt.tight_layout()
    plt.show()


plot_distributions(df4)
```



Use binning technique to remove Skewness

[81]:
```python
import pandas as pd

def apply_binning(df4, num_bins=10, method='width'):
    binned_dfs = {}
    for column in df4.select_dtypes(include=['number']).columns:
        if method == 'width':
            # Equal-width binning
            df4[f"{column}_width_binned"] = pd.cut(df4[column], bins=num_bins, labels=False)
```

```
        elif method == 'frequency':
            # Equal-frequency binning
            df4[f"{column}_freq_binned"] = pd.qcut(df[column], q=num_bins,␣
↪labels=False, duplicates='drop')
    return df4

# Apply equal-width binning
df4_width_binned = apply_binning(df4.copy(), num_bins=10, method='width')

# Apply equal-frequency binning
df4_freq_binned = apply_binning(df4.copy(), num_bins=10, method='frequency')

# Display the head of the dataframes to see the result
print("Equal-width Binned Data:")
print(df4_width_binned.head())
print("\nEqual-frequency Binned Data:")
print(df4_freq_binned.head())
```

```
Equal-width Binned Data:
   Sales_in_thousands  __year_resale_value  Price_in_thousands  Engine_size  \
0              16.919               16.360               21.50          1.8
1              39.384               19.875               28.40          3.2
4              20.397               22.255               23.99          1.8
5              18.780               23.555               33.95          2.8
7              19.747               14.010               26.99          2.5

   Horsepower  Wheelbase  Width  Length  Curb_weight  Fuel_capacity  …  \
0       140.0      101.2   67.3   172.4        2.639           13.2  …
1       225.0      108.1   70.3   192.9        3.517           17.2  …
4       150.0      102.6   68.2   178.0        2.998           16.4  …
5       200.0      108.7   76.1   192.0        3.561           18.5  …
7       170.0      107.3   68.4   176.0        3.179           16.6  …

   Price_in_thousands_width_binned  Engine_size_width_binned  \
0                                3                         0
1                                5                         5
4                                4                         0
5                                7                         4
7                                5                         3

   Horsepower_width_binned  Wheelbase_width_binned  Width_width_binned  \
0                        2                       2                   1
1                        7                       5                   3
4                        3                       3                   1
5                        5                       6                   8
7                        4                       5                   2
```

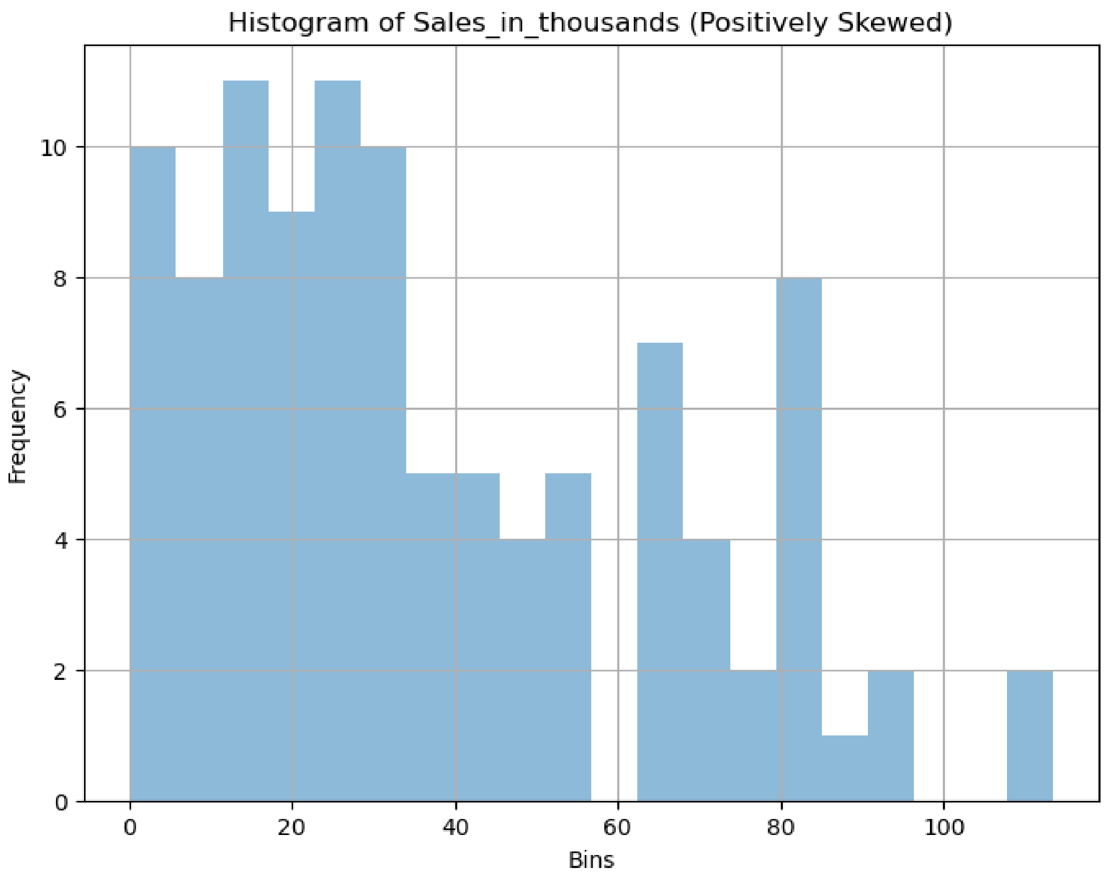
```
   Length_width_binned  Curb_weight_width_binned  Fuel_capacity_width_binned  \
0                    2                         1                           1
1                    6                         5                           5
4                    3                         3                           4
5                    5                         6                           7
7                    3                         4                           5

   Fuel_efficiency_width_binned  Power_perf_factor_width_binned
0                             6                               2
1                             4                               7
4                             5                               3
5                             2                               6
7                             5                               4

[5 rows x 24 columns]

Equal-frequency Binned Data:
   Sales_in_thousands  __year_resale_value  Price_in_thousands  Engine_size  \
0              16.919               16.360               21.50          1.8
1              39.384               19.875               28.40          3.2
4              20.397               22.255               23.99          1.8
5              18.780               23.555               33.95          2.8
7              19.747               14.010               26.99          2.5

   Horsepower  Wheelbase  Width  Length  Curb_weight  Fuel_capacity  …  \
0       140.0      101.2   67.3   172.4        2.639           13.2  …
1       225.0      108.1   70.3   192.9        3.517           17.2  …
4       150.0      102.6   68.2   178.0        2.998           16.4  …
5       200.0      108.7   76.1   192.0        3.561           18.5  …
7       170.0      107.3   68.4   176.0        3.179           16.6  …

   Price_in_thousands_freq_binned  Engine_size_freq_binned  \
0                             4.0                      0.0
1                             6.0                      5.0
4                             5.0                      0.0
5                             7.0                      4.0
7                             6.0                      3.0

   Horsepower_freq_binned  Wheelbase_freq_binned  Width_freq_binned  \
0                     2.0                    1.0                1.0
1                     7.0                    5.0                4.0
4                     2.0                    2.0                2.0
5                     6.0                    6.0                8.0
7                     4.0                    5.0                2.0

   Length_freq_binned  Curb_weight_freq_binned  Fuel_capacity_freq_binned  \
0                 0.0                      1.0                        0.0
1                 6.0                      6.0                        4.0
```

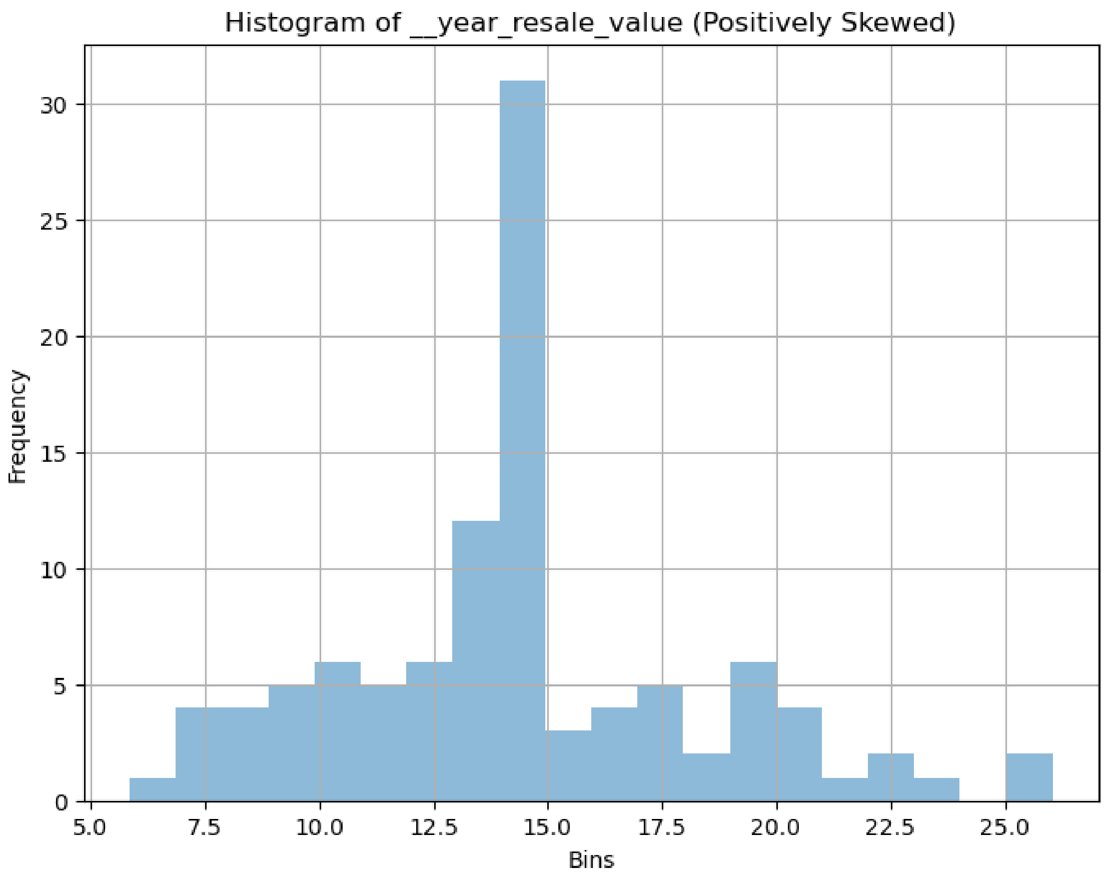
```
4               2.0                       2.0                       3.0
5               6.0                       6.0                       5.0
7               1.0                       3.0                       3.0

   Fuel_efficiency_freq_binned  Power_perf_factor_freq_binned
0                          8.0                            2.0
1                          5.0                            7.0
4                          7.0                            3.0
5                          2.0                            6.0
7                          7.0                            4.0

[5 rows x 24 columns]
```

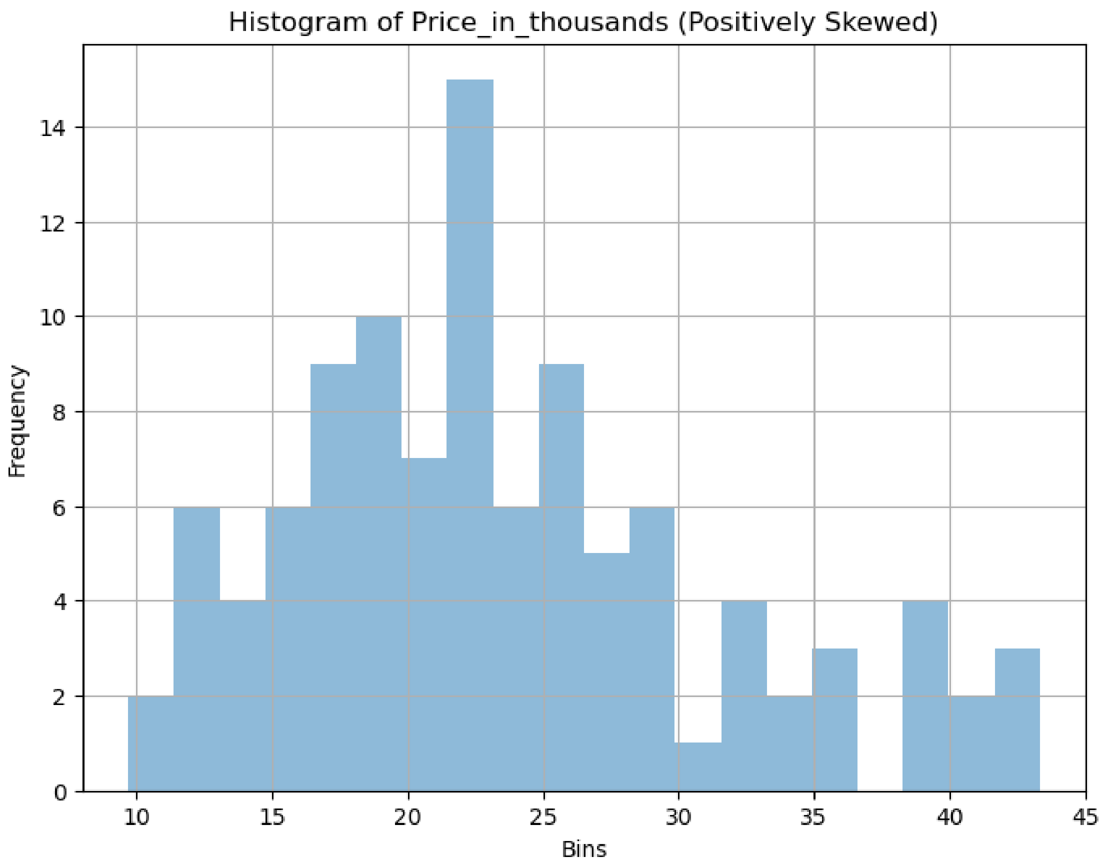
0.12 Distribution check

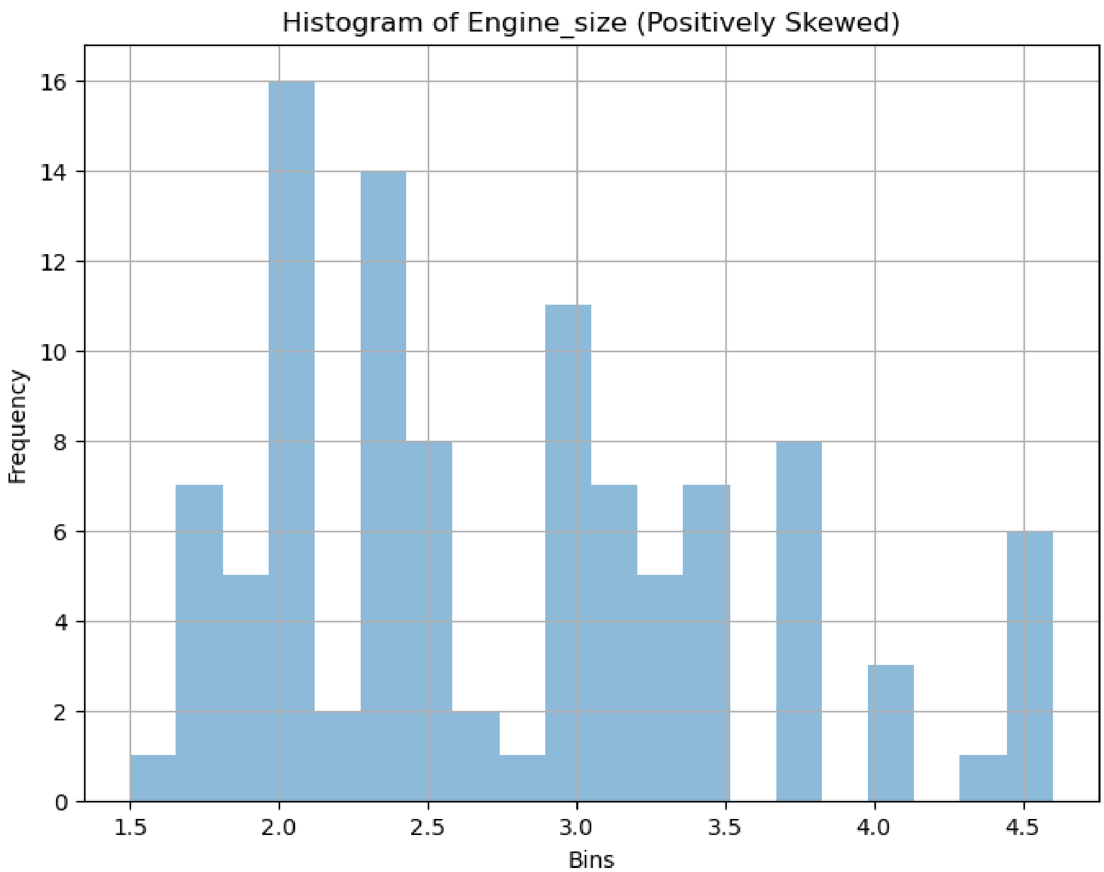
To confirm if skewness is removed or not

[85]:
```python
for column in df4_freq_binned.columns:
    plt.figure(figsize=(8, 6))
    plt.hist(df4_freq_binned[column], bins=20, alpha=0.5)

    # Calculate skewness
    skewness = df4_freq_binned[column].skew()

    # Label the plot title based on skewness
    if skewness > 0:
        skew_label = 'Positively Skewed'
    elif skewness < 0:
        skew_label = 'Negatively Skewed'
    else:
        skew_label = 'Symmetric'

    plt.title(f'Histogram of {column} ({skew_label})')
    plt.xlabel('Bins')
    plt.ylabel('Frequency')
    plt.grid(True)
    plt.show()
```

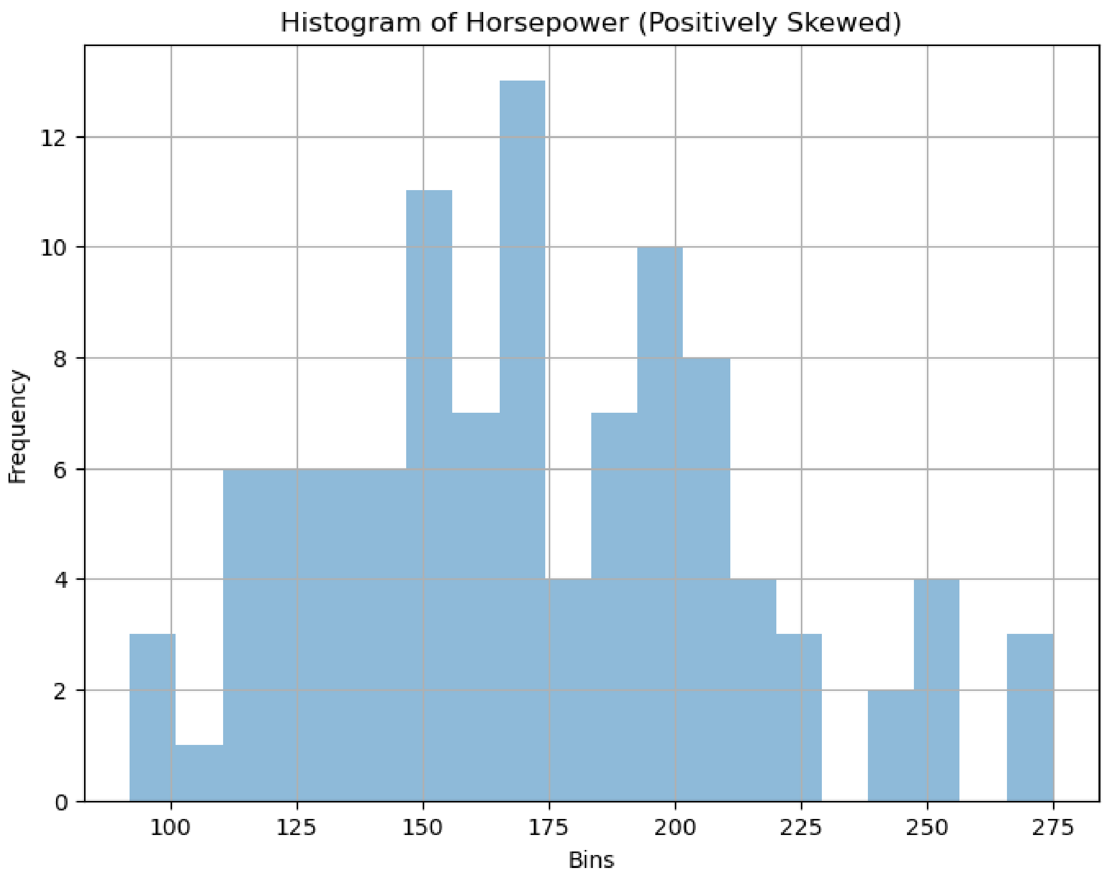

Histogram of Sales_in_thousands (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
0
20
40
60
80
100


Histogram of __year_resale_value (Positively Skewed)
Frequency
Bins
0
5
10
15
20
25
30
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0

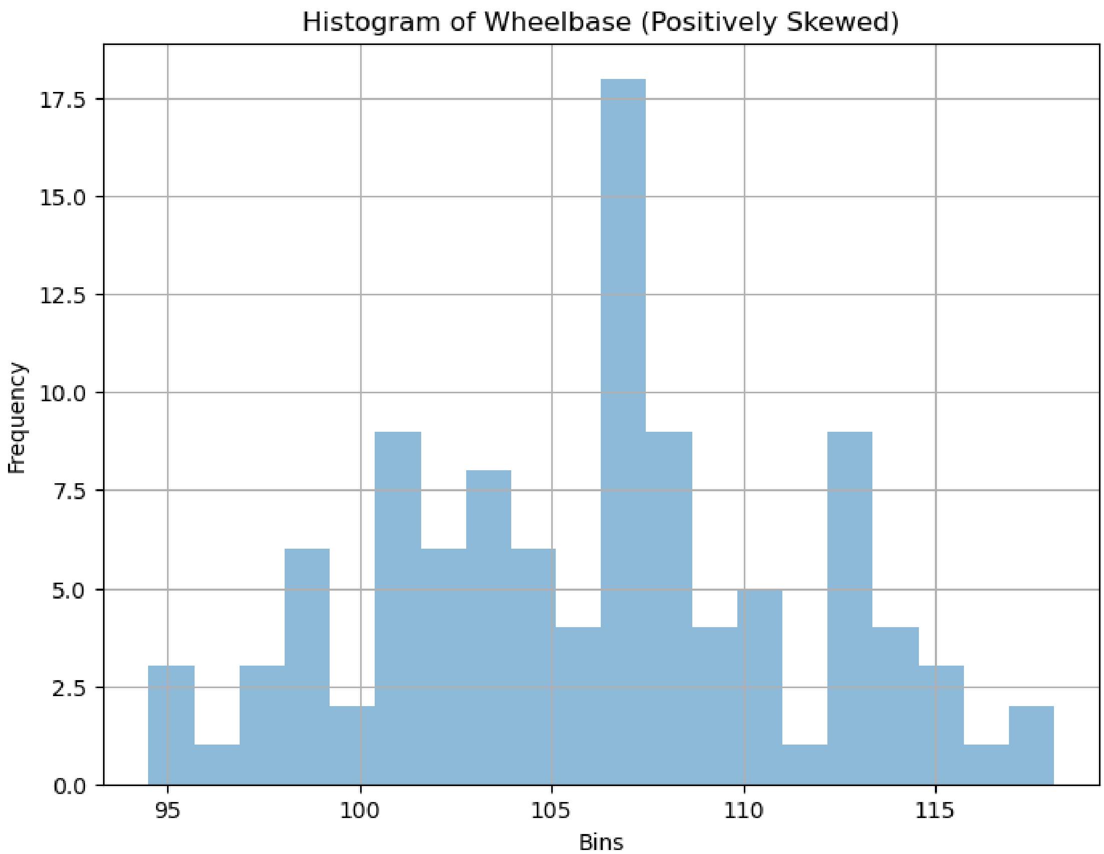

Histogram of Price_in_thousands (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
14
10
15
20
25
30
35
40
45

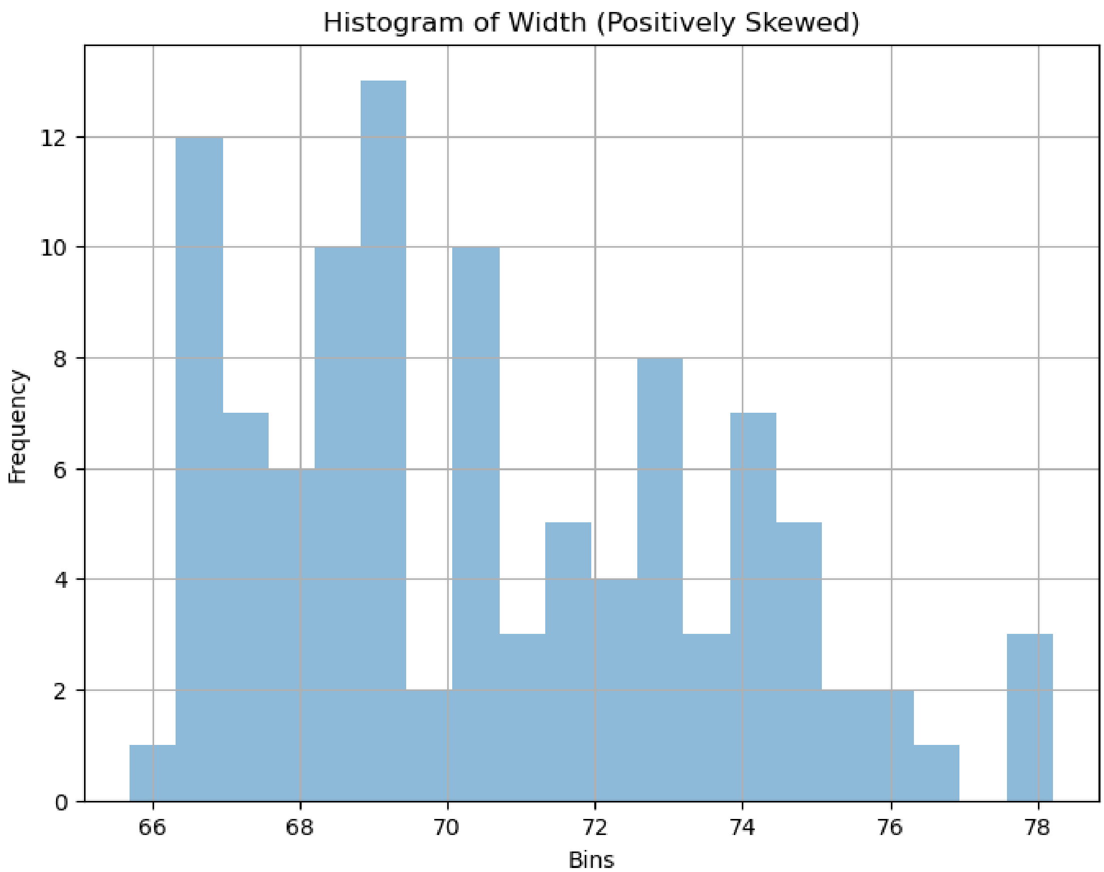


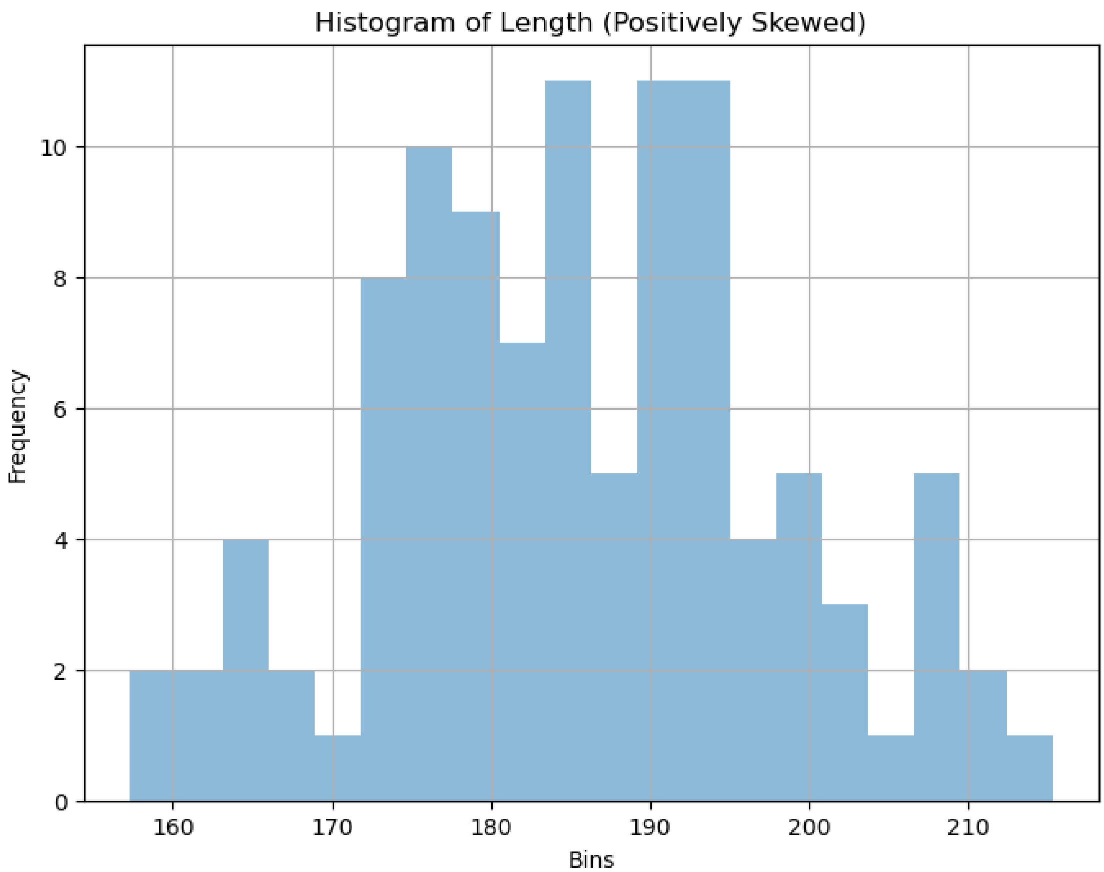
Histogram of Engine_size (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
14
16
1.5
2.0
2.5
3.0
3.5
4.0
4.5

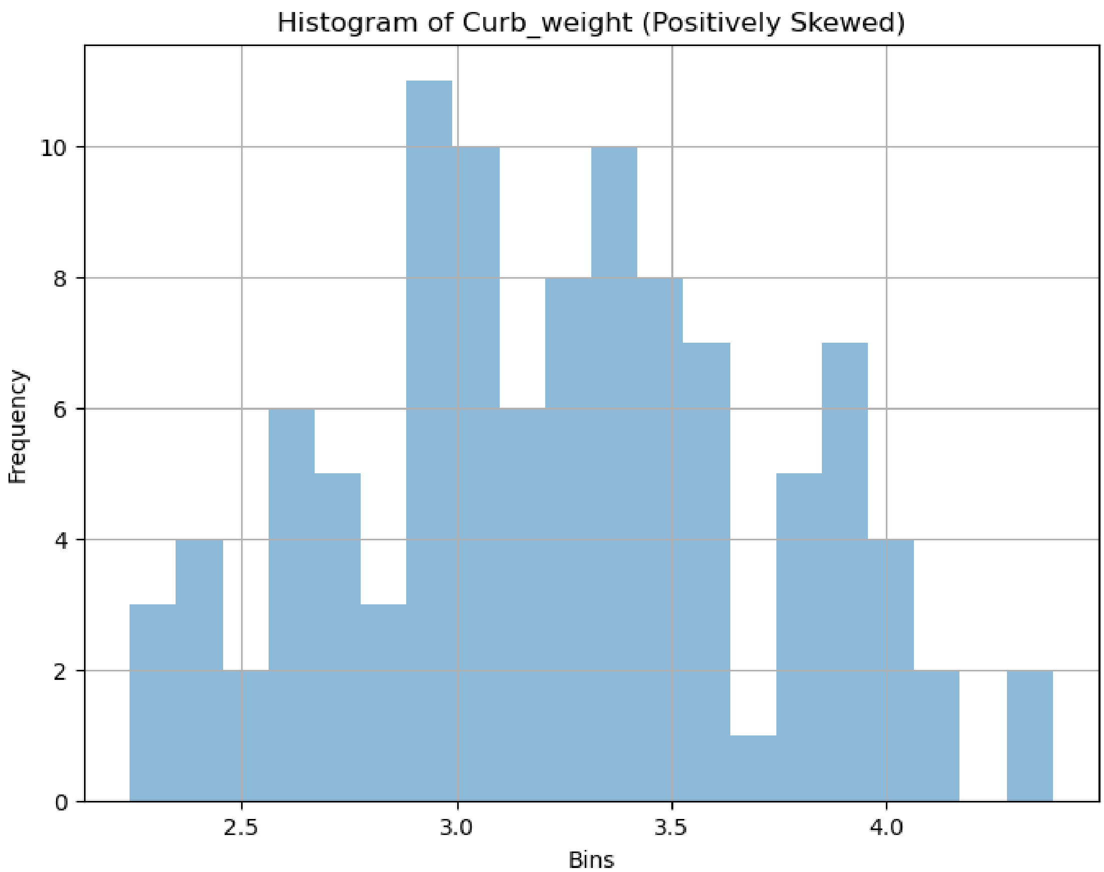


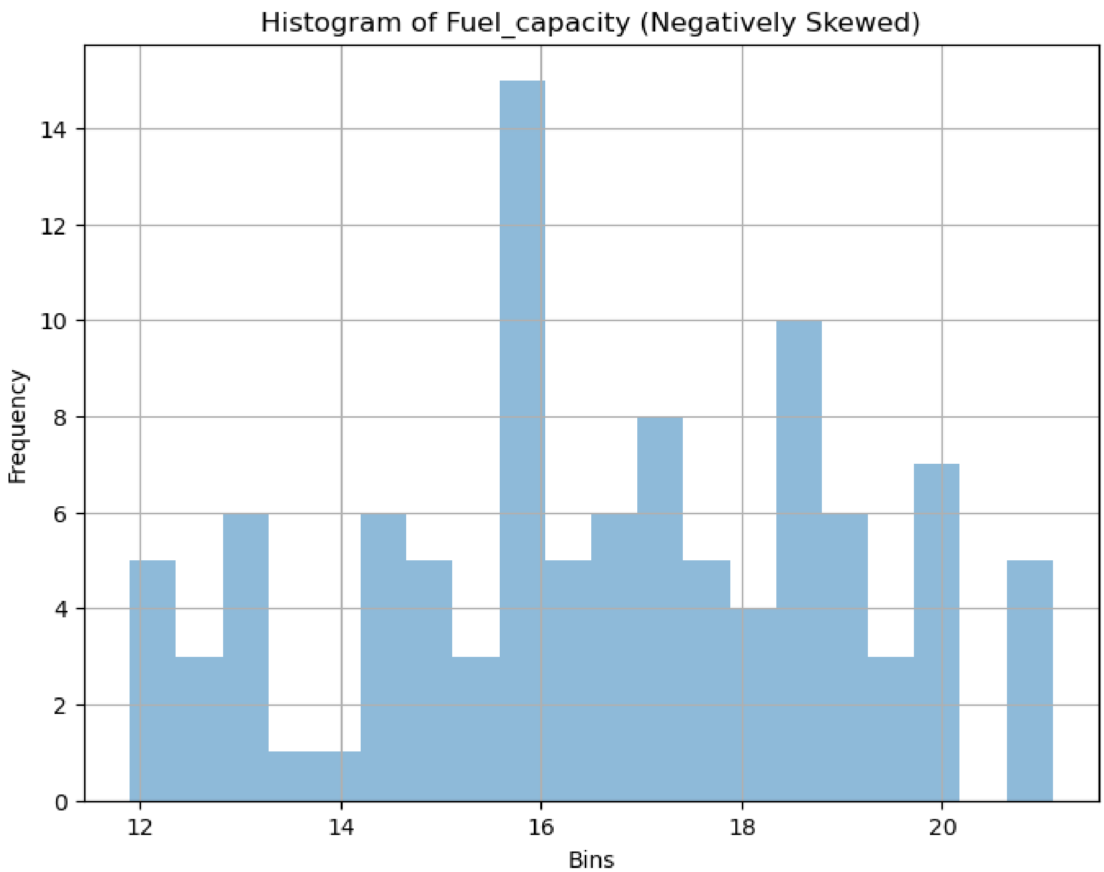
Histogram of Horsepower (Positively Skewed)
Frequency
0
2
4
6
8
10
12
100
125
150
175
200
225
250
275
Bins

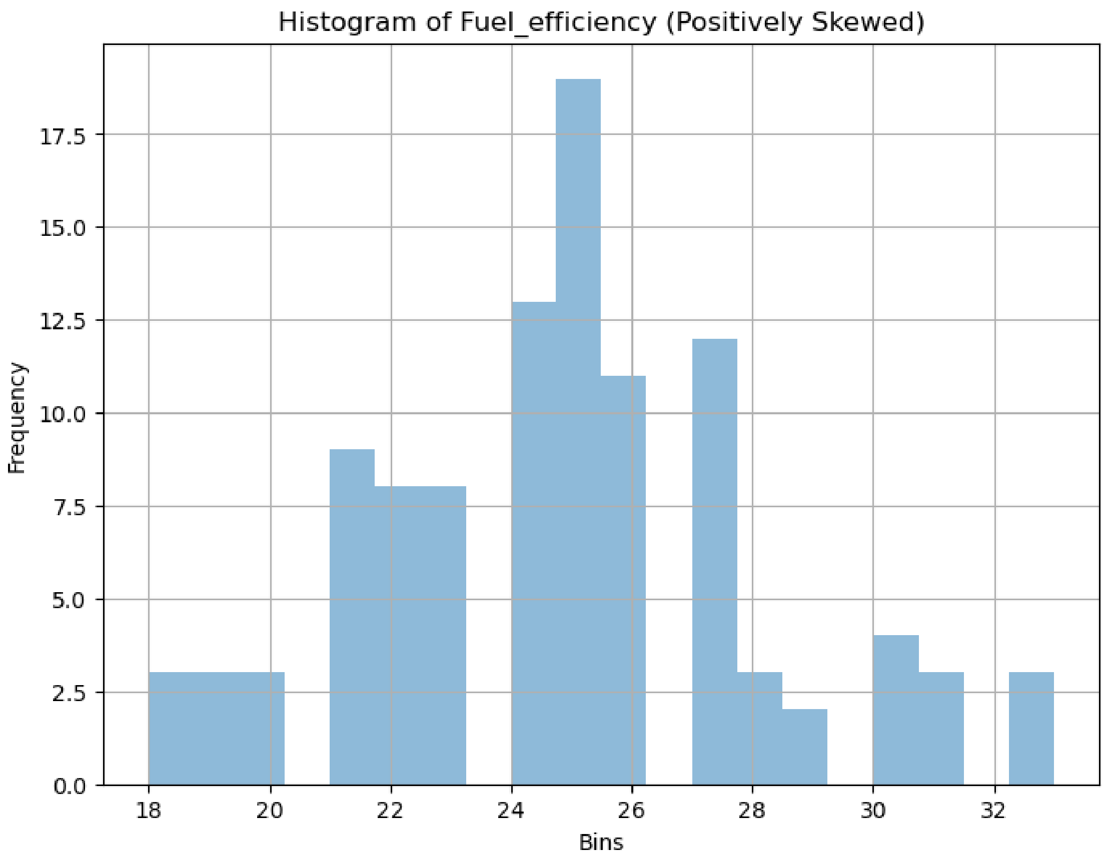

Histogram of Wheelbase (Positively Skewed)
Frequency
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
95
100
105
110
115
Bins

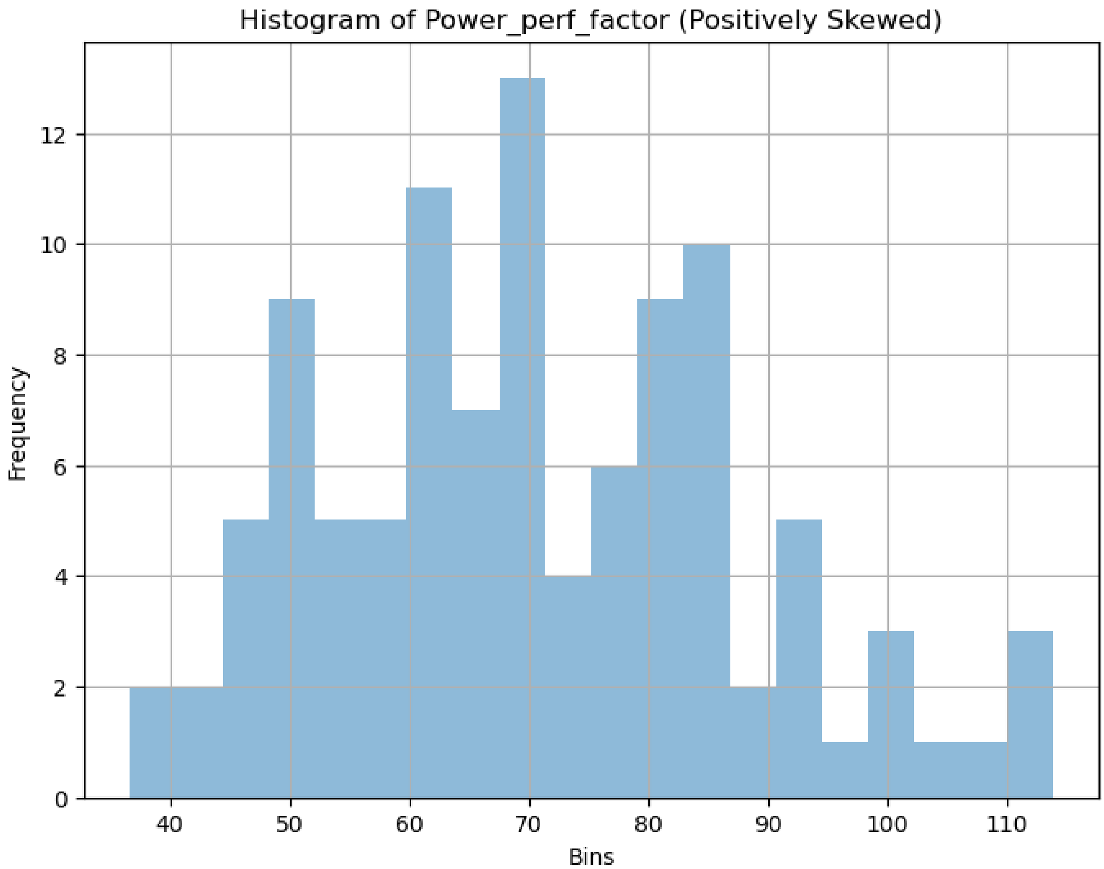

Histogram of Width (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
66
68
70
72
74
76
78

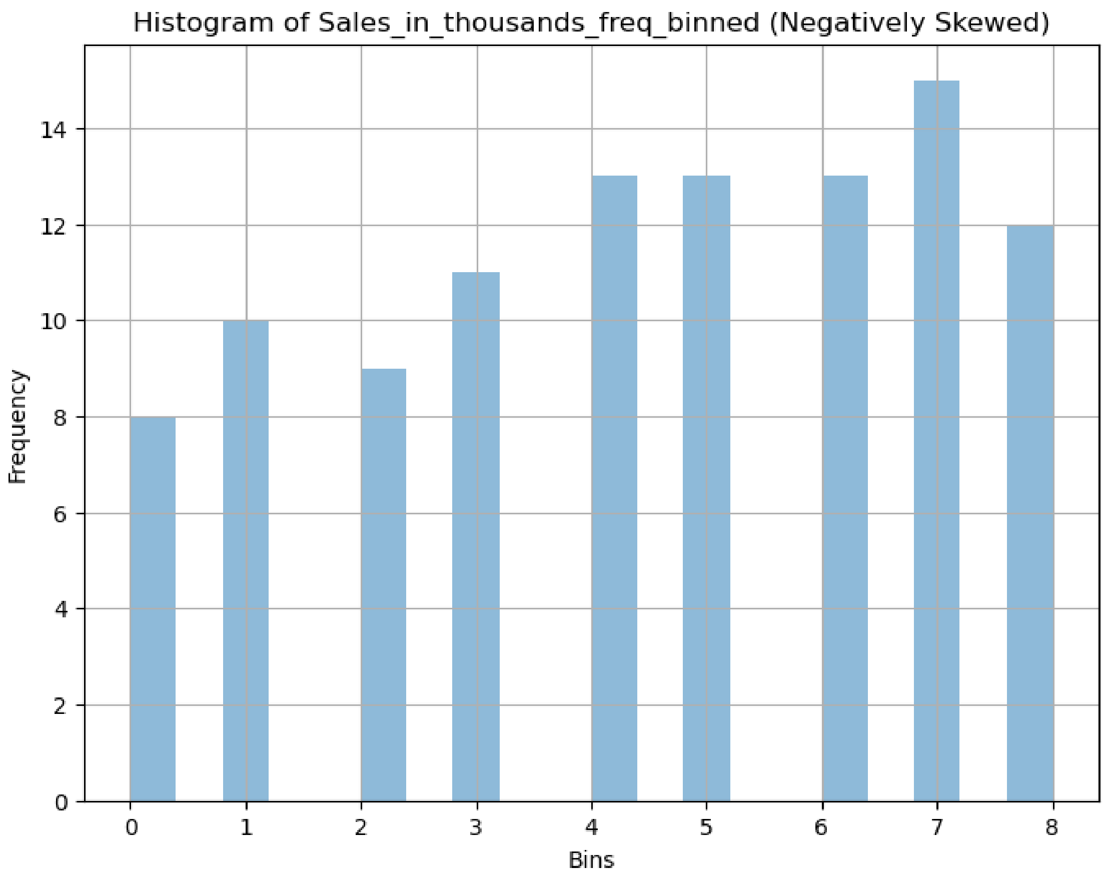

Histogram of Length (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
160
170
180
190
200
210

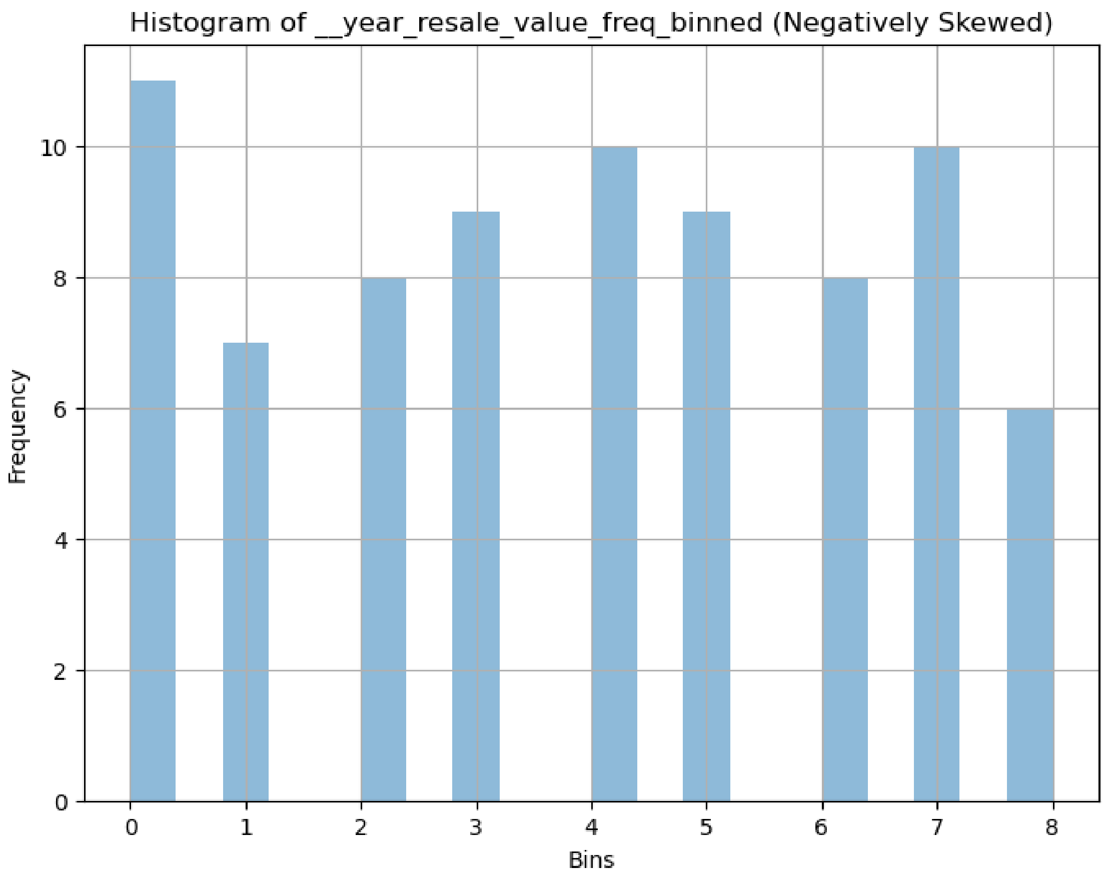

Histogram of Curb_weight (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
2.5
3.0
3.5
4.0

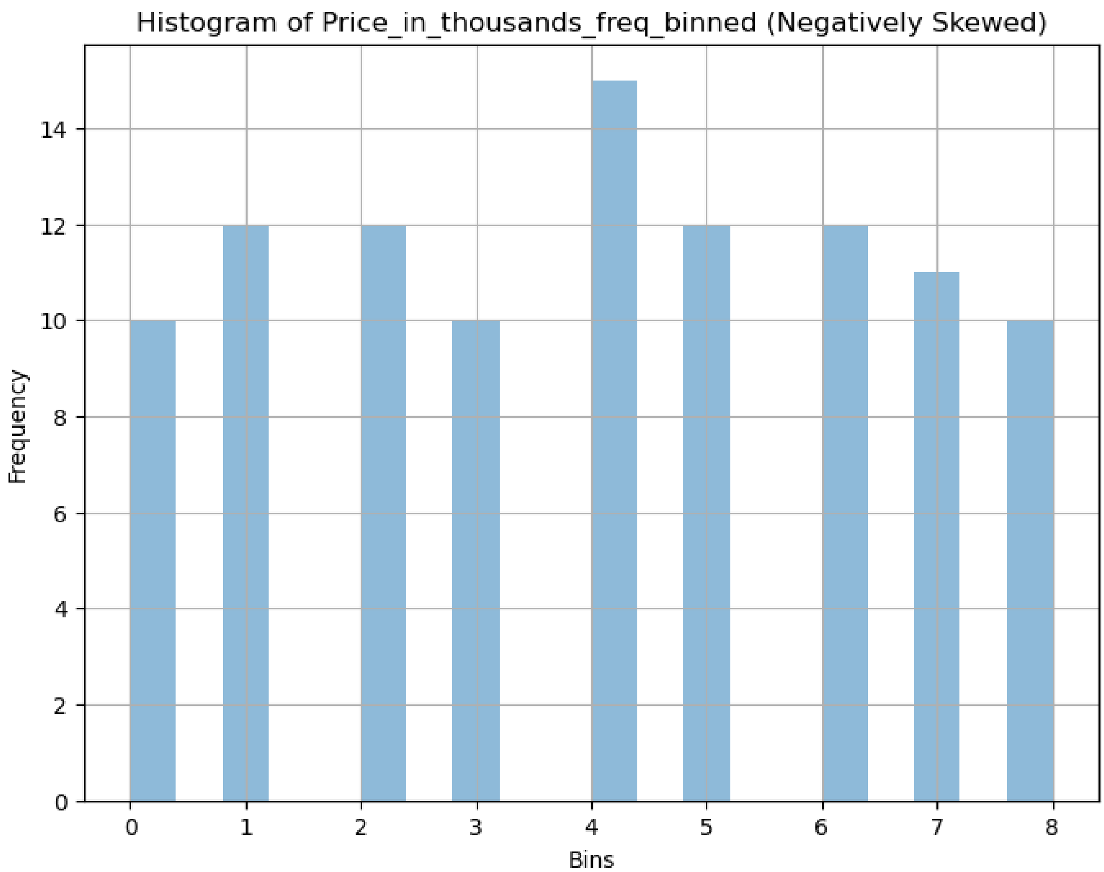

Histogram of Fuel_capacity (Negatively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
14
12
14
16
18
20

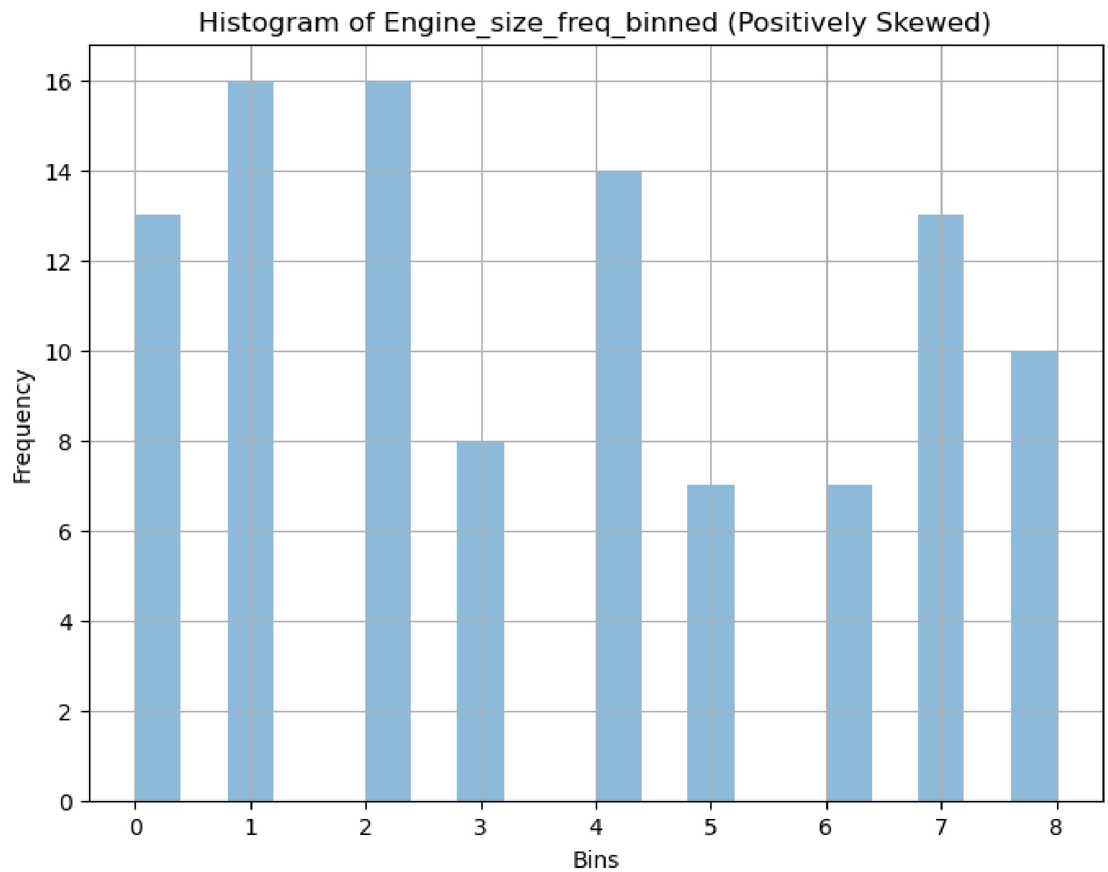

Histogram of Fuel_efficiency (Positively Skewed)
Frequency
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
18
20
22
24
26
28
30
32
Bins

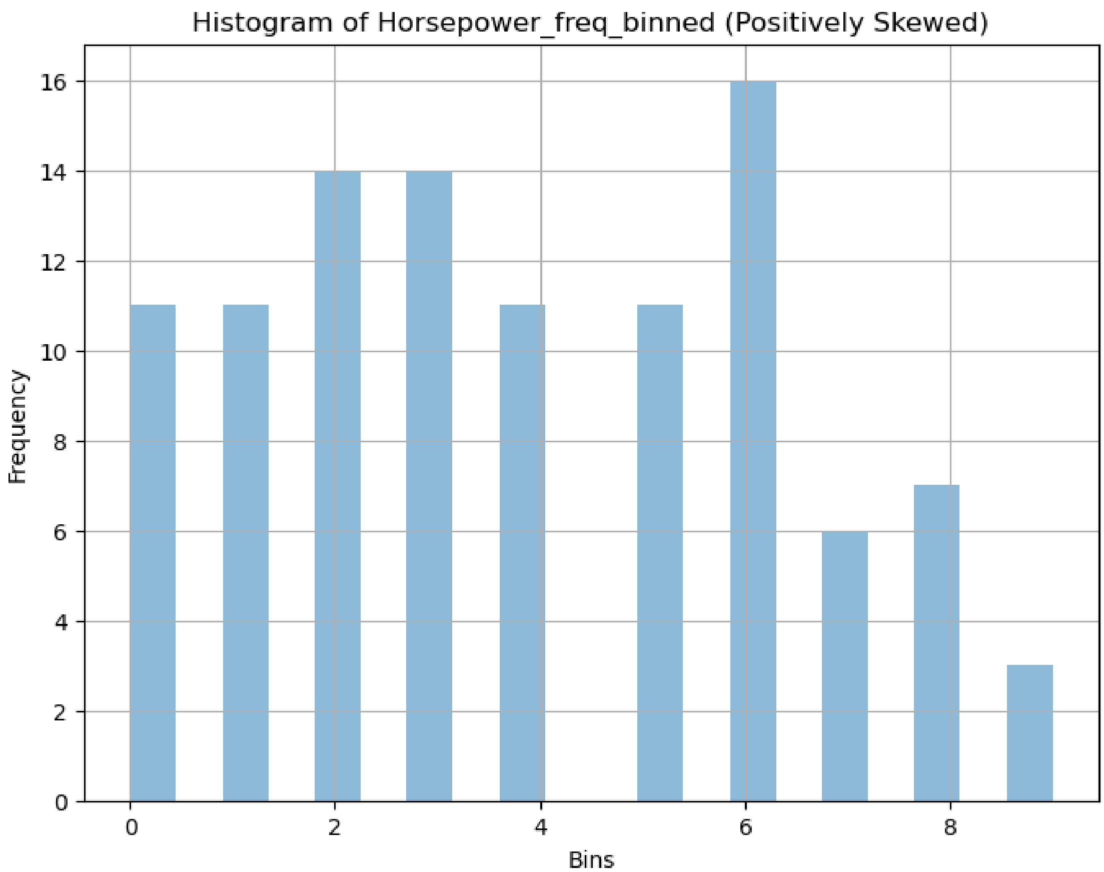

Histogram of Power_perf_factor (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
40
50
60
70
80
90
100
110

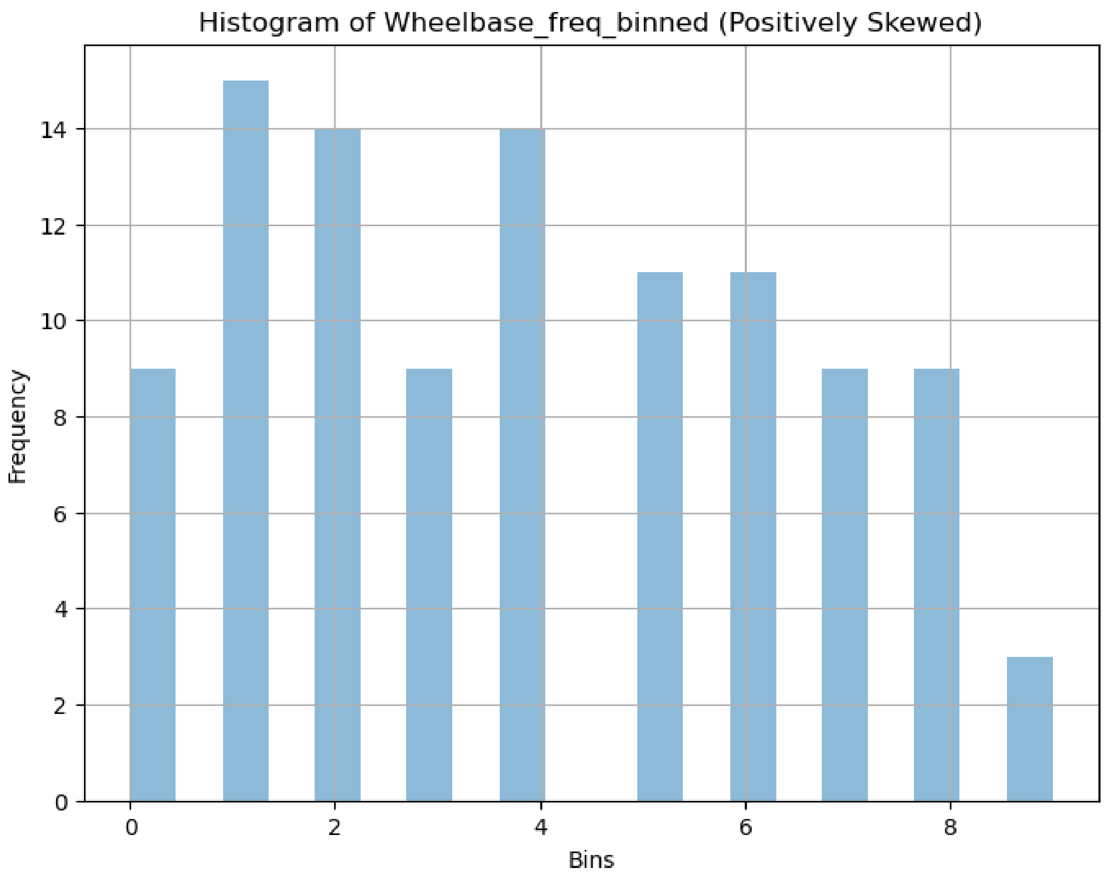

Histogram of Sales_in_thousands_freq_binned (Negatively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
14
0
1
2
3
4
5
6
7
8

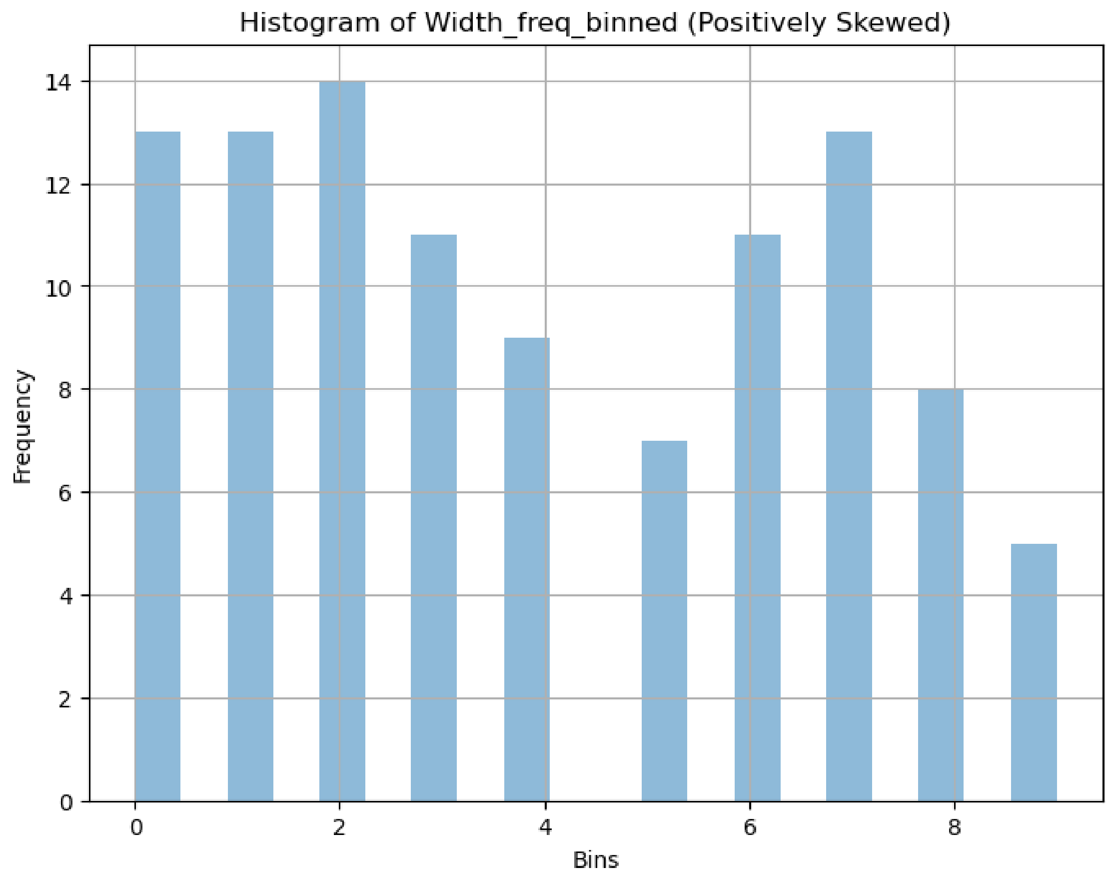

Histogram of __year_resale_value_freq_binned (Negatively Skewed)
Frequency
Bins
0
2
4
6
8
10
0
1
2
3
4
5
6
7
8

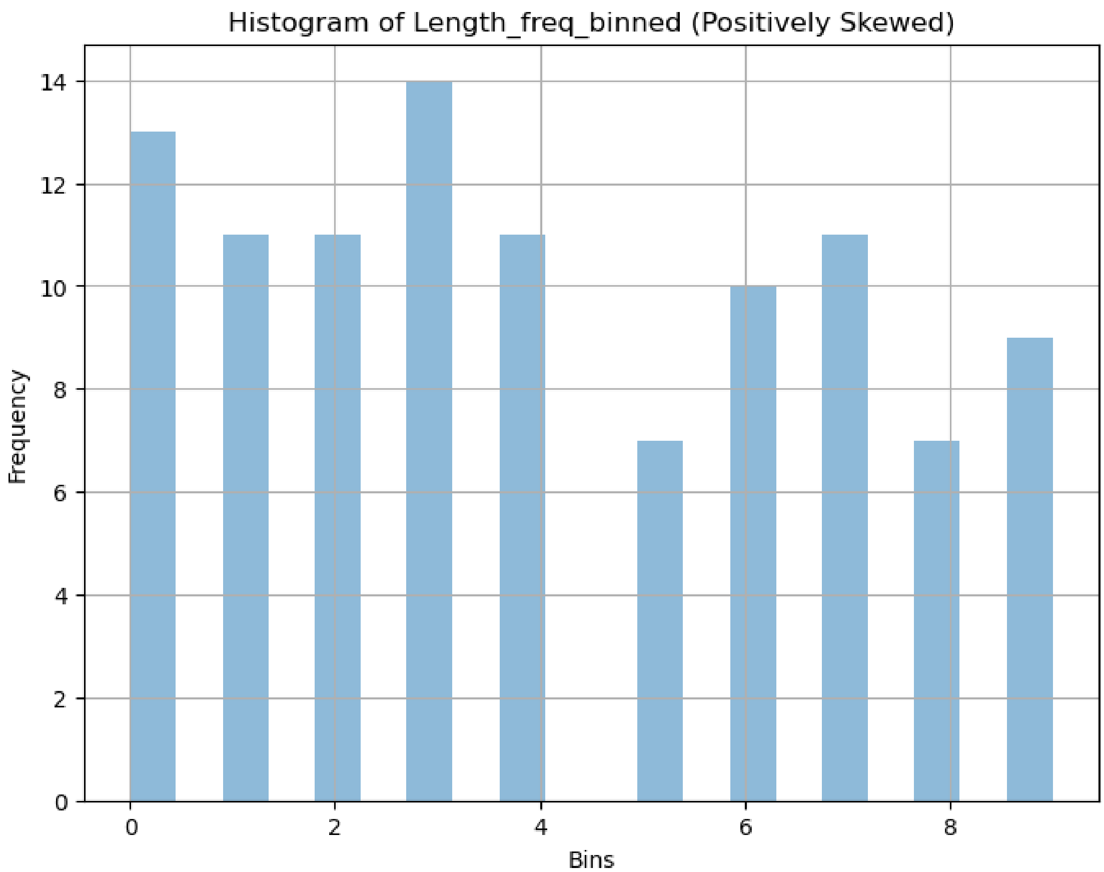

Histogram of Price_in_thousands_freq_binned (Negatively Skewed)
Frequency
0
2
4
6
8
10
12
14
0
1
2
3
4
5
6
7
8
Bins


Histogram of Engine_size_freq_binned (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
14
16
0
1
2
3
4
5
6
7
8

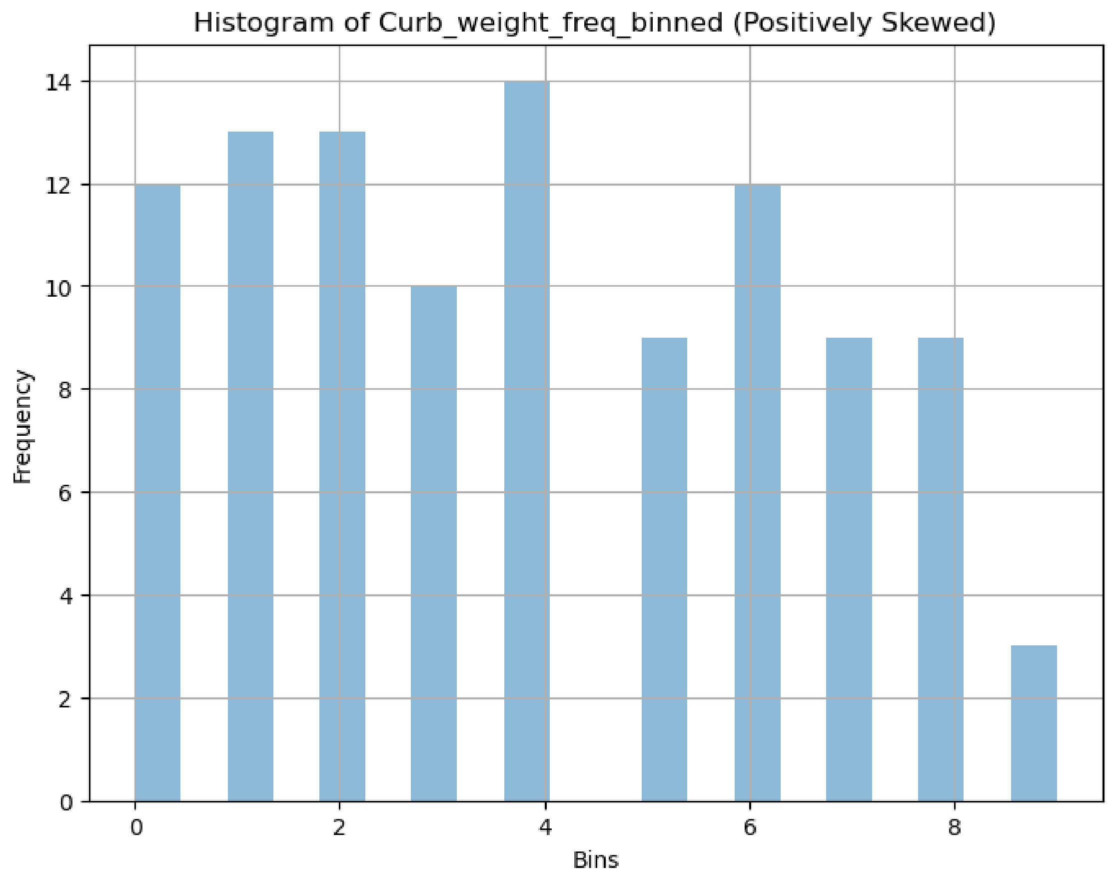

Histogram of Horsepower_freq_binned (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
14
16

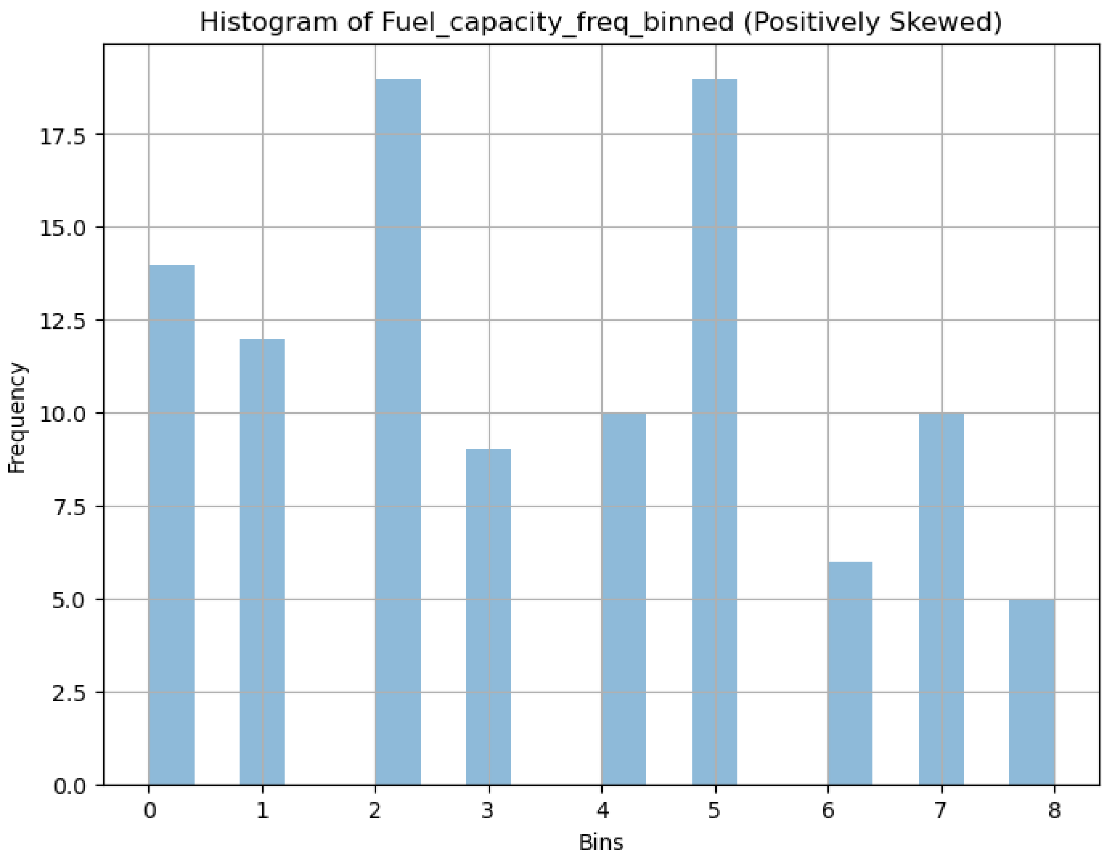

Histogram of Wheelbase_freq_binned (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
14


Histogram of Width_freq_binned (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
14

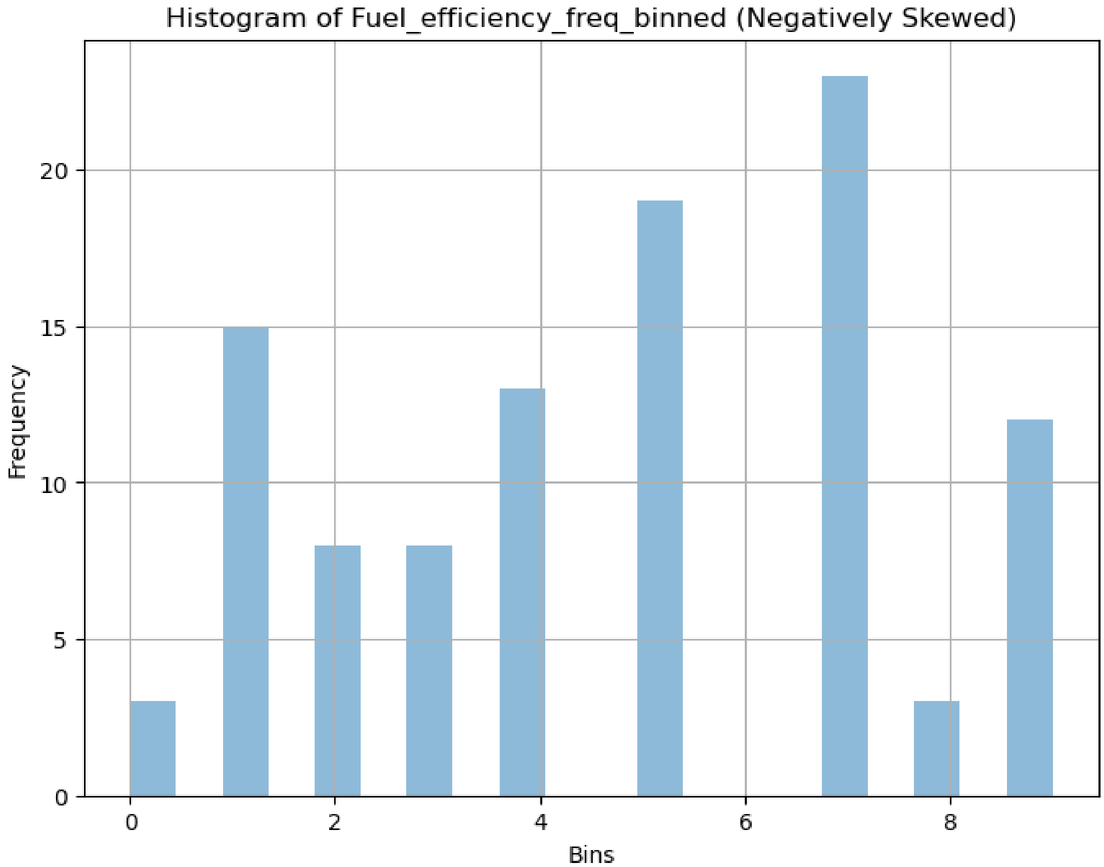


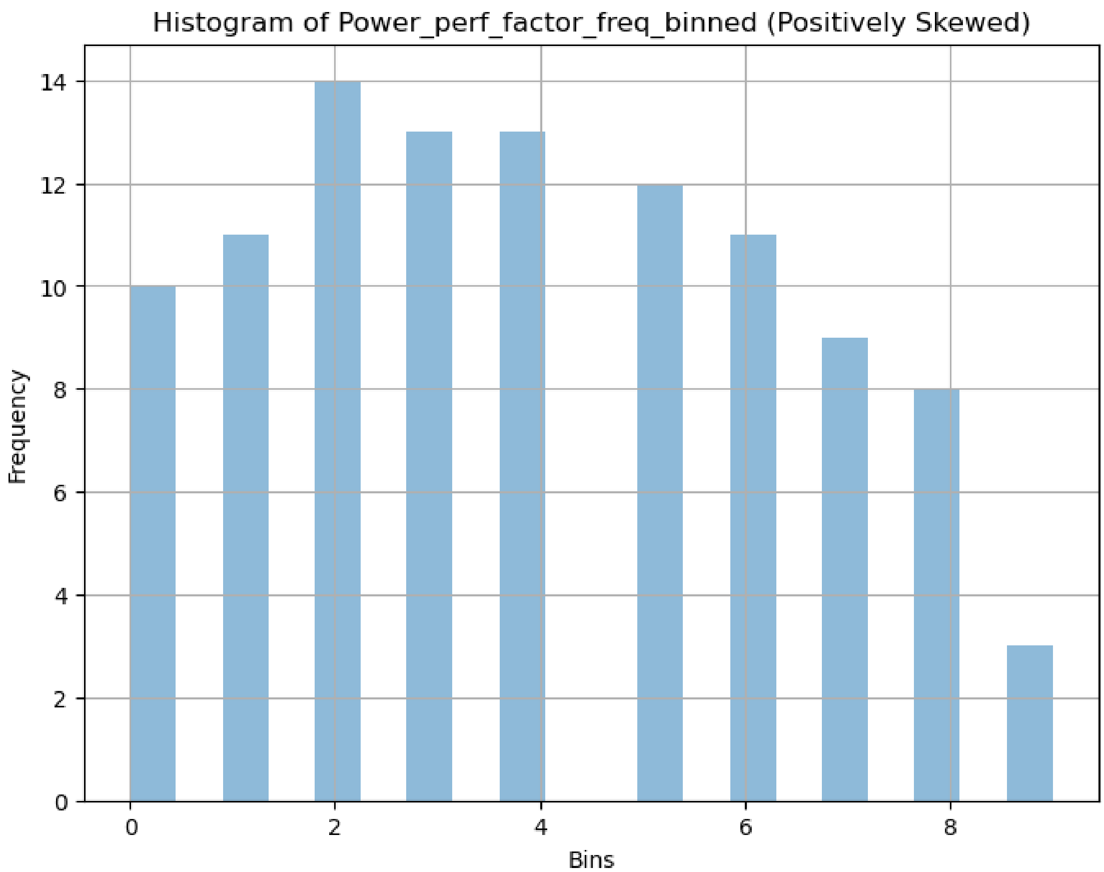
Histogram of Length_freq_binned (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
14


Histogram of Curb_weight_freq_binned (Positively Skewed)
Frequency
Bins
0
2
4
6
8
10
12
14



Histogram of Fuel_capacity_freq_binned (Positively Skewed)
Frequency
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1
2
3
4
5
6
7
8
Bins


Histogram of Fuel_efficiency_freq_binned (Negatively Skewed)
Frequency
Bins
0
5
10
15
20
0
2
4
6
8



0.12.1 Standardization and Normalization

Apply standardization and normalization on the columns

0.13 Standardization

[95]:
```
def standardize_data(df4_freq_binned):
    scaler = StandardScaler()
    # Standardize each column
    df_standardized = pd.DataFrame(scaler.fit_transform(df4_freq_binned),␣
 ↪columns=df4_freq_binned.columns)
    df_standardized.columns = [str(col) + '_standardized' for col in␣
 ↪df4_freq_binned.columns]
    return df_standardized

# Apply the standardization
df4_standardized = standardize_data(df4_freq_binned)

print("Standardized Data:")
```

```python
print(df4_standardized.head())
```

```
Standardized Data:
   Sales_in_thousands_standardized  __year_resale_value_standardized  \
0                        -0.761616                          0.511522
1                         0.056276                          1.395144
2                        -0.634991                          1.993443
3                        -0.693862                          2.320244
4                        -0.658656                         -0.079235

   Price_in_thousands_standardized  Engine_size_standardized  \
0                        -0.255249                 -1.251454
1                         0.604161                  0.495542
2                         0.054886                 -1.251454
3                         1.295426                 -0.003600
4                         0.428542                 -0.377956

   Horsepower_standardized  Wheelbase_standardized  Width_standardized  \
0                -0.808936               -0.897280           -1.048615
1                 1.256632                0.387413           -0.086922
2                -0.565928               -0.636617           -0.760107
3                 0.649112                0.499125            1.772350
4                -0.079912                0.238463           -0.695994

   Length_standardized  Curb_weight_standardized  Fuel_capacity_standardized  \
0            -1.060871                 -1.234675                   -1.468112
1             0.566424                  0.583048                    0.233099
2            -0.616342                 -0.491438                   -0.107144
3             0.494982                  0.674142                    0.785992
4            -0.775102                 -0.116713                   -0.022083

   …  Price_in_thousands_freq_binned_standardized  \
0  …                                     0.003861
1  …                                     0.806962
2  …                                     0.405411
3  …                                     1.208512
4  …                                     0.806962

   Engine_size_freq_binned_standardized  Horsepower_freq_binned_standardized  \
0                             -1.373471                            -0.737313
1                              0.526010                             1.249230
2                             -1.373471                            -0.737313
3                              0.146114                             0.851921
4                             -0.233782                             0.057304

   Wheelbase_freq_binned_standardized  Width_freq_binned_standardized  \
0                           -1.131595                       -1.055943
```

```
1                           0.411826                           0.017198
2                          -0.745740                          -0.698230
3                           0.797682                           1.448053
4                           0.411826                          -0.698230

   Length_freq_binned_standardized  Curb_weight_freq_binned_standardized  \
0                        -1.442394                             -1.091320
1                         0.660536                              0.800302
2                        -0.741418                             -0.712996
3                         0.660536                              0.800302
4                        -1.091906                             -0.334672

   Fuel_capacity_freq_binned_standardized  \
0                               -1.435957
1                                0.232641
2                               -0.184508
3                                0.649791
4                               -0.184508

   Fuel_efficiency_freq_binned_standardized  \
0                                  1.217257
1                                  0.090301
2                                  0.841605
3                                 -1.036655
4                                  0.841605

   Power_perf_factor_freq_binned_standardized
0                                   -0.766583
1                                    1.206799
2                                   -0.371906
3                                    0.812122
4                                    0.022770

[5 rows x 24 columns]
```

0.14 Normalization

[101]:
```
def normalize_data(df4_standardized):
    scaler = MinMaxScaler()
    # Normalize each column
    df_normalized = pd.DataFrame(scaler.fit_transform(df4_standardized),␣
 ↪columns=df4_standardized.columns)
    df_normalized.columns = [str(col) + '_normalized' for col in␣
 ↪df4_standardized.columns]
    return df_normalized
```

```python
df4_normalized = normalize_data(df4_standardized)


print("Normalized Data:")
print(df4_normalized.head())
```

```
Normalized Data:
   Sales_in_thousands_standardized_normalized  \
0                                    0.148412
1                                    0.346763
2                                    0.179120
3                                    0.164843
4                                    0.173381

   __year_resale_value_standardized_normalized  \
0                                     0.520059
1                                     0.694156
2                                     0.812036
3                                     0.876424
4                                     0.403665

   Price_in_thousands_standardized_normalized  \
0                                    0.350896
1                                    0.556064
2                                    0.424935
3                                    0.721091
4                                    0.514139

   Engine_size_standardized_normalized  Horsepower_standardized_normalized  \
0                             0.096774                            0.262295
1                             0.548387                            0.726776
2                             0.096774                            0.316940
3                             0.419355                            0.590164
4                             0.322581                            0.426230

   Wheelbase_standardized_normalized  Width_standardized_normalized  \
0                           0.283898                          0.128
1                           0.576271                          0.368
2                           0.343220                          0.200
3                           0.601695                          0.832
4                           0.542373                          0.216

   Length_standardized_normalized  Curb_weight_standardized_normalized  \
0                        0.260345                             0.185841
1                        0.613793                             0.594783
2                        0.356897                             0.353051
3                        0.598276                             0.615277
```

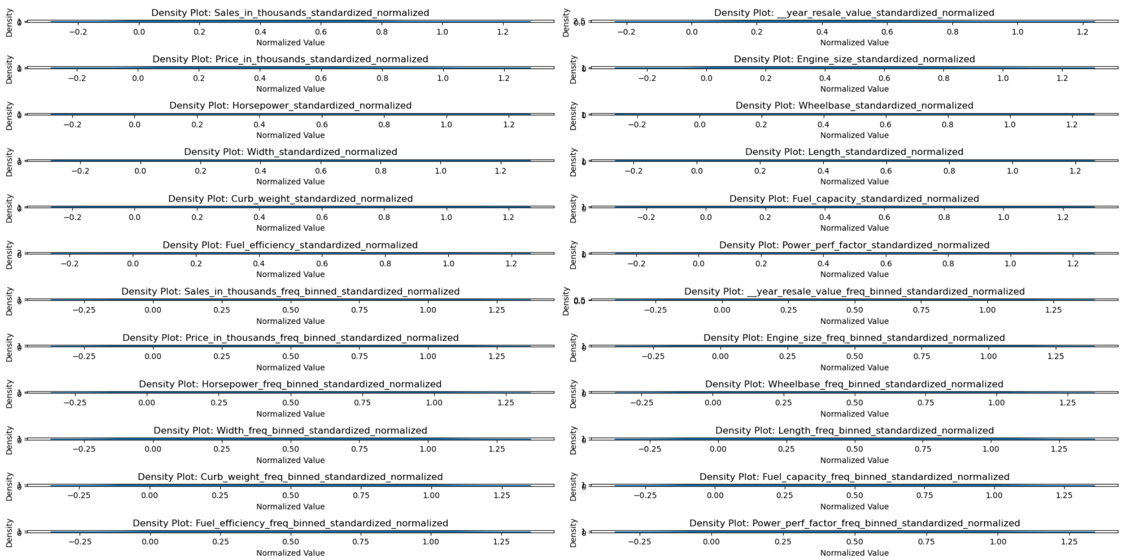
```
4                        0.322414                                  0.437354

   Fuel_capacity_standardized_normalized  …  \
0                               0.141304  …
1                               0.576087  …
2                               0.489130  …
3                               0.717391  …
4                               0.510870  …

   Price_in_thousands_freq_binned_standardized_normalized  \
0                                              0.500
1                                              0.750
2                                              0.625
3                                              0.875
4                                              0.750

   Engine_size_freq_binned_standardized_normalized  \
0                                            0.000
1                                            0.625
2                                            0.000
3                                            0.500
4                                            0.375

   Horsepower_freq_binned_standardized_normalized  \
0                                       0.222222
1                                       0.777778
2                                       0.222222
3                                       0.666667
4                                       0.444444

   Wheelbase_freq_binned_standardized_normalized  \
0                                      0.111111
1                                      0.555556
2                                      0.222222
3                                      0.666667
4                                      0.555556

   Width_freq_binned_standardized_normalized  \
0                                  0.111111
1                                  0.444444
2                                  0.222222
3                                  0.888889
4                                  0.222222

   Length_freq_binned_standardized_normalized  \
0                                   0.000000
1                                   0.666667
2                                   0.222222
```

```
3                                     0.666667
4                                     0.111111

   Curb_weight_freq_binned_standardized_normalized  \
0                                          0.111111
1                                          0.666667
2                                          0.222222
3                                          0.666667
4                                          0.333333

   Fuel_capacity_freq_binned_standardized_normalized  \
0                                               0.000
1                                               0.500
2                                               0.375
3                                               0.625
4                                               0.375

   Fuel_efficiency_freq_binned_standardized_normalized  \
0                                             0.888889
1                                             0.555556
2                                             0.777778
3                                             0.222222
4                                             0.777778

   Power_perf_factor_freq_binned_standardized_normalized
0                                             0.222222
1                                             0.777778
2                                             0.333333
3                                             0.666667
4                                             0.444444

[5 rows x 24 columns]
```

0.15 Distribution check

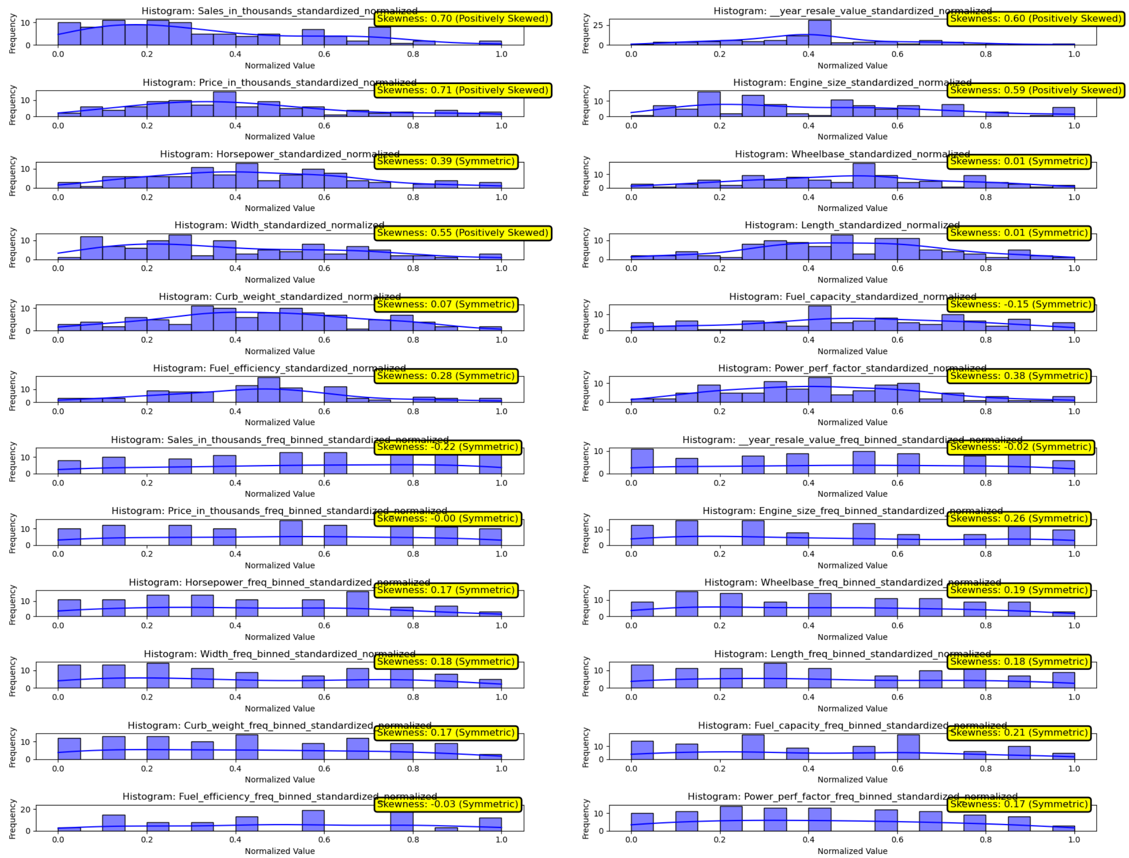
[109]:
```
def plot_density_normalized(df):
    num_columns = df.columns
    plt.figure(figsize=(20, 10))  # Increased width for better visibility
    for i, column in enumerate(num_columns):
        plt.subplot((len(num_columns) + 1) // 2, 2, i + 1)  # Arrange plots in␣
 ↪2 columns
        sns.kdeplot(df[column], fill=True, lw=3)  # Increased line width (lw)␣
 ↪to 3
        plt.title(f'Density Plot: {column}')
        plt.xlabel('Normalized Value')
        plt.ylabel('Density')
    plt.tight_layout()
```

```
    plt.show()


plot_density_normalized(df4_normalized)
```



[113]:
```
def plot_histograms_with_skewness(df):
    num_columns = df.columns
    plt.figure(figsize=(20, 15))  # Increased figure size for clarity
    for i, column in enumerate(num_columns):
        plt.subplot((len(num_columns) + 1) // 2, 2, i + 1)  # Arrange plots in␣
↪2 columns
        sns.histplot(df[column], kde=True, bins=20, color='blue')
        plt.title(f'Histogram: {column}')
        plt.xlabel('Normalized Value')
        plt.ylabel('Frequency')

        # Calculate and annotate skewness
        skew_value = df[column].skew()
        skew_type = "Symmetric"
        if skew_value > 0.5:
            skew_type = "Positively Skewed"
        elif skew_value < -0.5:
            skew_type = "Negatively Skewed"
        plt.annotate(f'Skewness: {skew_value:.2f} ({skew_type})', xy=(0.7, 0.
↪9), xycoords='axes fraction', fontsize=12,
                     bbox=dict(boxstyle="round,pad=0.3", fc="yellow",␣
↪ec="black", lw=2))
```

```
    plt.tight_layout()
    plt.show()

# Call the function to plot histograms with skewness indicators
plot_histograms_with_skewness(df4_normalized)
```



0.16 The END!