# FEATURE SELECTION ON AN AUTOMOBILE DATASET WITH PYTHON

## WHAT FEATURES OF A CAR ARE IMPORTANT IN DETERMINING ITS PRICE?

DR. ALVIN ANG

# CONTENTS


*I. Describing the Data ..... 5*

**A. df.sample() ..... 5**

**B. df.dtypes ..... 7**

**C. Describing only Numerical Data ..... 8**

**D. Describing only Textual Data ..... 8**

*II. Feature Selection: Selecting Numerical Columns ..... 10*

**A. Display Heatmap of All numerical Columns Correlations ..... 11**

**B. Is Wheel Base Important? ..... 12**

1. KEEP Wheel Base ..... 12

**C. Is Length Important? ..... 12**

1. KEEP Length ..... 12

**D. Is Width Important? ..... 13**

1. KEEP Width ..... 13

**E. Is Height Important? ..... 13**

1. DROP Height ..... 13

**F. Is Curb Weight Important? ..... 14**

1. KEEP Curb Weight ..... 14

**G. Is Engine Size Important? ..... 14**

1. KEEP Engine Size ..... 14

**H. Is Bore Important? ..... 15**

1. KEEP Bore ..... 15

**I. Is Stroke Important? ..... 15**

1. DROP Stroke ..... 15

**J. Is Compression Ratio Important? ..... 16**

1. DROP Compression Ratio ..... 16

**K. Is Horsepower Important? ..... 16**

1. KEEP Horsepower ..... 16

**L. Is Peak RPM Important? ..... 17**

1. DROP Peak RPM ..... 17

**M. Is City MPG Important? ..... 17**

1. KEEP City MPG ..... 17

**N. Is Highway MPG Important? ..... 18**

1. KEEP Highway MPG ..... 18

**O. Is City – L/100km Important? ..... 18**
1. KEEP City-L/100km ..... 18

**P. Is Diesel Important? ..... 19**
1. DROP Diesel ..... 19

**Q. Is GAS Important? ..... 19**
1. DROP Gas ..... 19

***III. Feature Selection: Selecting Textual Columns ..... 20***

**A. Is Make Important? ..... 21**
1. Global Test ..... 22
   a) KEEP Make ..... 23
2. Individual Test ..... 24
   a) Comparing Honda vs Subaru Pricing ..... 25
   b) Comparing Honda vs Peugot vs Porsche Pricing ..... 26

**B. Is Aspiration Important? ..... 27**
1. Global Test ..... 27
   a) DROP Aspiration ..... 28

**C. Is Number of Doors Important? ..... 29**
1. Global Test ..... 29
   a) DROP Num of Doors ..... 30

**D. Is Body Style Important? ..... 31**
1. Global Test ..... 31
   a) KEEP Body Style ..... 32
2. Individual Test ..... 33
   a) Comparing Hatchback vs Convertible Pricing ..... 34

**E. Is Drive Wheels Important? ..... 35**
1. Global Test ..... 35
   a) KEEP Drive Wheels ..... 36
2. Individual Test ..... 37
   a) Comparing FWD vs 4WD ..... 38

**F. Is Engine Location Important? ..... 39**
1. Global Test ..... 39
   a) KEEP Engine Location ..... 40

**G. Is Engine Type Important? ..... 41**
1. Global Test ..... 41
   a) KEEP Engine Type ..... 42
2. Individual Test ..... 43
   a) Comparing Rotor vs L vs OHCV ..... 44

**H. Is Number of Cylinders Important? ..... 45**
1. Global Test ..... 45
   a) KEEP Number of Cylinders ..... 46

2. Individual Test ..... 47
a) Comparing 4 vs 6 vs 8 Cylinders ..... 48

**I. Is Fuel System Important? ..... 49**
1. Global Test ..... 49
a) KEEP Fuel Systems ..... 50
2. Individual Test ..... 51
a) Comparing 2bbl vs mpfi ..... 52

**J. Is Horsepower Important? ..... 53**
1. Global Test ..... 53
a) KEEP Horsepower ..... 54
2. Individual Test ..... 55
a) Comparing Low vs Medium Horsepower ..... 56

***Conclusion ..... 57***

**A. Numerical Columns ..... 57**

**B. Textual Columns ..... 58**

***About Dr. Alvin Ang ..... 59***

## I. DESCRIBING THE DATA

IPYNB: https://www.alvinang.sg/s/Feature_Selection_for_AutomobileEDA_csv.ipynb

Dataset: https://www.alvinang.sg/s/automobileEDA.csv

### A. DF.SAMPLE()

Step 1: Describing the Data

1a) Importing Data

```
[2] import pandas as pd
    import matplotlib.pylab as plt
    import seaborn as sns

    df = pd.read_csv('https://www.alvinang.sg/s/automobileEDA.csv')

    df.sample(5)
```

| | symboling | normalized-losses | make | aspiration | num-of-doors | body-style | drive-wheels | engine-location | wheel-base | length | ... | compression-ratio | horsepower | peak-rpm | city-mpg | highway-mpg | price | city-L/100km | horsepower-binned | diesel | gas |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 122 | alfa-romero | std | two | convertible | rwd | front | 88.6 | 0.811148 | ... | 9.0 | 111.0 | 5000.0 | 21 | 27 | 13495.0 | 11.190476 | Medium | 0 | 1 |
| **1** | 3 | 122 | alfa-romero | std | two | convertible | rwd | front | 88.6 | 0.811148 | ... | 9.0 | 111.0 | 5000.0 | 21 | 27 | 16500.0 | 11.190476 | Medium | 0 | 1 |
| **2** | 1 | 122 | alfa-romero | std | two | hatchback | rwd | front | 94.5 | 0.822681 | ... | 9.0 | 154.0 | 5000.0 | 19 | 26 | 16500.0 | 12.368421 | Medium | 0 | 1 |
| **3** | 2 | 164 | audi | std | four | sedan | fwd | front | 99.8 | 0.848630 | ... | 10.0 | 102.0 | 5500.0 | 24 | 30 | 13950.0 | 9.791667 | Medium | 0 | 1 |
| **4** | 2 | 164 | audi | std | four | sedan | 4wd | front | 99.4 | 0.848630 | ... | 8.0 | 115.0 | 5500.0 | 18 | 22 | 17450.0 | 13.055556 | Medium | 0 | 1 |

### B. DF.DTYPES

```
print(df.dtypes)
```

```
symboling              int64
normalized-losses      int64
make                  object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
```

```
symboling              int64
normalized-losses      int64
make                  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           float64
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                float64
city-L/100km         float64
horsepower-binned     object
diesel                 int64
gas                    int64
dtype: object
```

### C. DESCRIBING ONLY NUMERICAL DATA

```
df.describe()
```

| | symboling | normalized-losses | wheel-base | length | width | height | curb-weight | engine-size | bore | stroke | compression-ratio | horsepower | peak-rpm | city-mpg | highway-mpg | price | city-L/100km | diesel | gas |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 201.000000 | 201.00000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 197.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 |
| **mean** | 0.840796 | 122.00000 | 98.797015 | 0.837102 | 0.915126 | 53.766667 | 2555.666667 | 126.875622 | 3.330692 | 3.256904 | 10.164279 | 103.405534 | 5117.665368 | 25.179104 | 30.686567 | 13207.129353 | 9.944145 | 0.099502 | 0.900498 |
| **std** | 1.254802 | 31.99625 | 6.066366 | 0.059213 | 0.029187 | 2.447822 | 517.296727 | 41.546834 | 0.268072 | 0.319256 | 4.004965 | 37.365700 | 478.113805 | 6.423220 | 6.815150 | 7947.066342 | 2.534599 | 0.300083 | 0.300083 |
| **min** | -2.000000 | 65.00000 | 86.600000 | 0.678039 | 0.837500 | 47.800000 | 1488.000000 | 61.000000 | 2.540000 | 2.070000 | 7.000000 | 48.000000 | 4150.000000 | 13.000000 | 16.000000 | 5118.000000 | 4.795918 | 0.000000 | 0.000000 |
| **25%** | 0.000000 | 101.00000 | 94.500000 | 0.801538 | 0.890278 | 52.000000 | 2169.000000 | 98.000000 | 3.150000 | 3.110000 | 8.600000 | 70.000000 | 4800.000000 | 19.000000 | 25.000000 | 7775.000000 | 7.833333 | 0.000000 | 1.000000 |
| **50%** | 1.000000 | 122.00000 | 97.000000 | 0.832292 | 0.909722 | 54.100000 | 2414.000000 | 120.000000 | 3.310000 | 3.290000 | 9.000000 | 95.000000 | 5125.369458 | 24.000000 | 30.000000 | 10295.000000 | 9.791667 | 0.000000 | 1.000000 |
| **75%** | 2.000000 | 137.00000 | 102.400000 | 0.881788 | 0.925000 | 55.500000 | 2926.000000 | 141.000000 | 3.580000 | 3.410000 | 9.400000 | 116.000000 | 5500.000000 | 30.000000 | 34.000000 | 16500.000000 | 12.368421 | 0.000000 | 1.000000 |
| **max** | 3.000000 | 256.00000 | 120.900000 | 1.000000 | 1.000000 | 59.800000 | 4066.000000 | 326.000000 | 3.940000 | 4.170000 | 23.000000 | 262.000000 | 6600.000000 | 49.000000 | 54.000000 | 45400.000000 | 18.076923 | 1.000000 | 1.000000 |

### D. DESCRIBING ONLY TEXTUAL DATA

```
df.describe(include=['object'])
```

| | make | aspiration | num-of-doors | body-style | drive-wheels | engine-location | engine-type | num-of-cylinders | fuel-system | horsepower-binned |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 201 | 201 | 201 | 201 | 201 | 201 | 201 | 201 | 201 | 200 |
| **unique** | 22 | 2 | 2 | 5 | 3 | 2 | 6 | 7 | 8 | 3 |
| **top** | toyota | std | four | sedan | fwd | front | ohc | four | mpfi | Low |
| **freq** | 32 | 165 | 115 | 94 | 118 | 198 | 145 | 157 | 92 | 115 |

```
symboling              int64
normalized-losses      int64
make                  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           float64
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                float64
city-L/100km         float64
horsepower-binned     object
diesel                 int64
gas                    int64
dtype: object
```

## NUMERICAL DATA

| | symboling | normalized-losses | wheel-base | length | width | height | curb-weight | engine-size | bore | stroke | compression-ratio | horsepower | peak-rpm | city-mpg | highway-mpg | price | city-L/100km | diesel | gas |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 201.000000 | 201.00000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 197.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 | 201.000000 |
| mean | 0.840796 | 122.00000 | 98.797015 | 0.837102 | 0.915126 | 53.766667 | 2555.666667 | 126.875622 | 3.330692 | 3.256904 | 10.164279 | 103.405534 | 5117.665368 | 25.179104 | 30.686567 | 13207.129353 | 9.944145 | 0.099502 | 0.900498 |
| std | 1.254802 | 31.99625 | 6.066366 | 0.059213 | 0.029187 | 2.447822 | 517.296727 | 41.546834 | 0.268072 | 0.319256 | 4.004965 | 37.365700 | 478.113805 | 6.423220 | 6.815150 | 7947.066342 | 2.534599 | 0.300083 | 0.300083 |
| min | -2.000000 | 65.00000 | 86.600000 | 0.678039 | 0.837500 | 47.800000 | 1488.000000 | 61.000000 | 2.540000 | 2.070000 | 7.000000 | 48.000000 | 4150.000000 | 13.000000 | 16.000000 | 5118.000000 | 4.795918 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 101.00000 | 94.500000 | 0.801538 | 0.890278 | 52.000000 | 2169.000000 | 98.000000 | 3.150000 | 3.110000 | 8.600000 | 70.000000 | 4800.000000 | 19.000000 | 25.000000 | 7775.000000 | 7.833333 | 0.000000 | 1.000000 |
| 50% | 1.000000 | 122.00000 | 97.000000 | 0.832292 | 0.909722 | 54.100000 | 2414.000000 | 120.000000 | 3.310000 | 3.290000 | 9.000000 | 95.000000 | 5125.369458 | 24.000000 | 30.000000 | 10295.000000 | 9.791667 | 0.000000 | 1.000000 |
| 75% | 2.000000 | 137.00000 | 102.400000 | 0.881788 | 0.925000 | 55.500000 | 2926.000000 | 141.000000 | 3.580000 | 3.410000 | 9.400000 | 116.000000 | 5500.000000 | 30.000000 | 34.000000 | 16500.000000 | 12.368421 | 0.000000 | 1.000000 |
| max | 3.000000 | 256.00000 | 120.900000 | 1.000000 | 1.000000 | 59.800000 | 4066.000000 | 326.000000 | 3.940000 | 4.170000 | 23.000000 | 262.000000 | 6600.000000 | 49.000000 | 54.000000 | 45400.000000 | 18.076923 | 1.000000 | 1.000000 |

| | make | aspiration | num-of-doors | body-style | drive-wheels | engine-location | engine-type | num-of-cylinders | fuel-system | horsepower-binned |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 201 | 201 | 201 | 201 | 201 | 201 | 201 | 201 | 201 | 200 |
| unique | 22 | 2 | 2 | 5 | 3 | 2 | 6 | 7 | 8 | 3 |
| top | toyota | std | four | sedan | fwd | front | ohc | four | mpfi | Low |
| freq | 32 | 165 | 115 | 94 | 118 | 198 | 145 | 157 | 92 | 115 |

## TEXTUAL DATA

## Step 2: Feature Selection: Selecting Numerical Columns

### Correlation Coefficient

- 1: Strong Positive Linear Correlation.
- 0: No linear correlation, the two variables most likely do not affect each other.
- -1: Strong Negative Linear Correlation

df.corr() only shows correlation for Numerical Data (int and floats).

### P Value

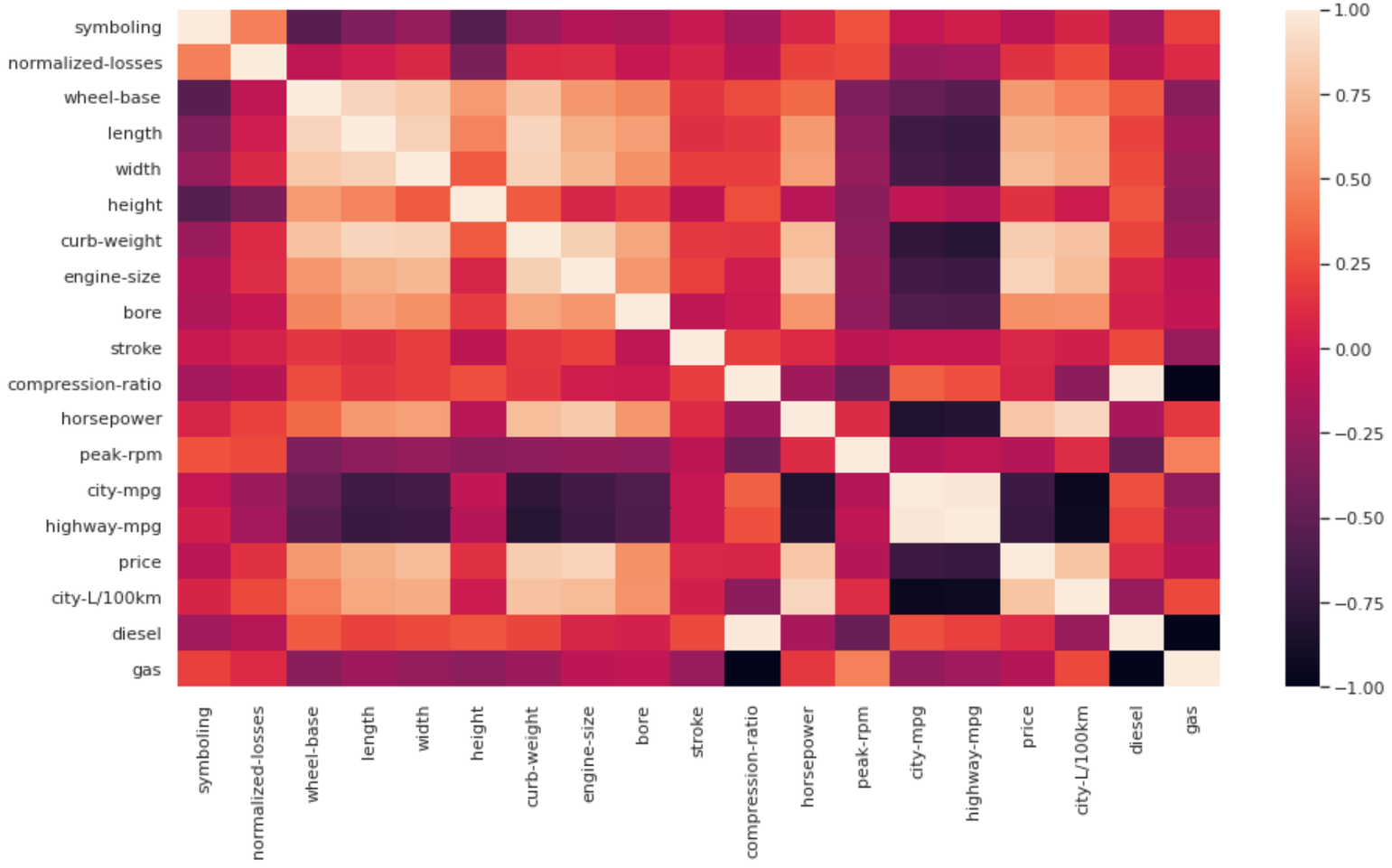
- Why do we still need p-value if we already have the Correlation Coefficient?
- Because p-value double confirms the linear relationship.

If:

- p-value < 0.001
  > strong correlation significance.
- p-value < 0.05
  > moderate correlation significance.
- p-value < 0.1
  > weak correlation significance.
- p-value > 0.1
  > no correlation significance.

### A. DISPLAY HEATMAP OF ALL NUMERICAL COLUMNS CORRELATIONS

```
sns.set(rc = {'figure.figsize':(15,8)})
sns.heatmap(df.corr(), annot = False)

#displaying ALL Numerical Columns Correlations
```

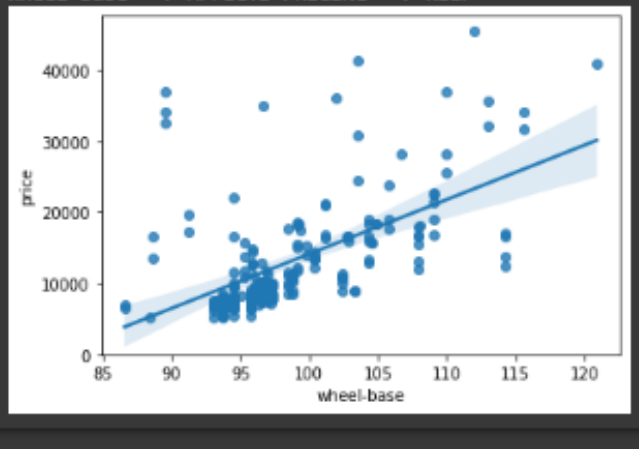


### B. IS WHEEL BASE IMPORTANT?

2a) Is 'Wheel Base' Important?

```
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

sns.regplot(x = "wheel-base", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['wheel-base'], df['price'])

print("Wheel Base --> Correlation Coefficient is {:.2f} --> OK Positive Correlation".format(pearson_coef))
print("Wheel Base --> P-value is {:.2f} --> STRONG Significance".format(p_value))
print("Wheel Base --> AFFECTS PRICING --> KEEP")
```

```
Wheel Base --> Correlation Coefficient is 0.58 --> OK Positive Correlation
Wheel Base --> P-value is 0.00 --> STRONG Significance
Wheel Base --> AFFECTS PRICING --> KEEP
```



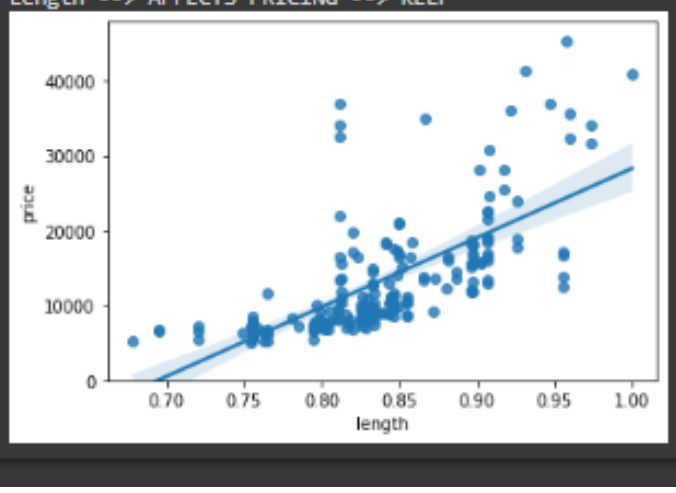

1. KEEP WHEEL BASE

### C. IS LENGTH IMPORTANT?

2b) Is 'Length" Important?

```
sns.regplot(x = "length", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['length'], df['price'])

print("Length --> Correlation Coefficient is {:.2f} --> OK Positive Correlation".format(pearson_coef))
print("Length --> P-value is {:.2f} --> STRONG Significance".format(p_value))
print("Length --> AFFECTS PRICING --> KEEP")
```

```
Length --> Correlation Coefficient is 0.69 --> OK Positive Correlation
Length --> P-value is 0.00 --> STRONG Significance
Length --> AFFECTS PRICING --> KEEP
```



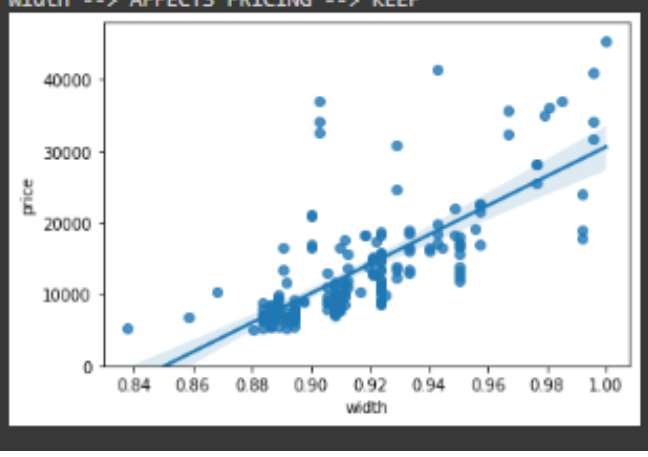

1. KEEP LENGTH

### D. IS WIDTH IMPORTANT?

2c) Is 'Width" Important?

```
[ ] sns.regplot(x = "width", y = "price", data = df)
    plt.ylim(0,)

    pearson_coef, p_value = stats.pearsonr(df['width'], df['price'])

    print("Width --> Correlation Coefficient is {:.2f} --> STRONG Positive Correlation".format(pearson_coef))
    print("Width --> P-value is {:.2f} --> STRONG Significance".format(p_value))
    print("Width --> AFFECTS PRICING --> KEEP")
```

```
Width --> Correlation Coefficient is 0.75 --> STRONG Positive Correlation
Width --> P-value is 0.00 --> STRONG Significance
Width --> AFFECTS PRICING --> KEEP
```



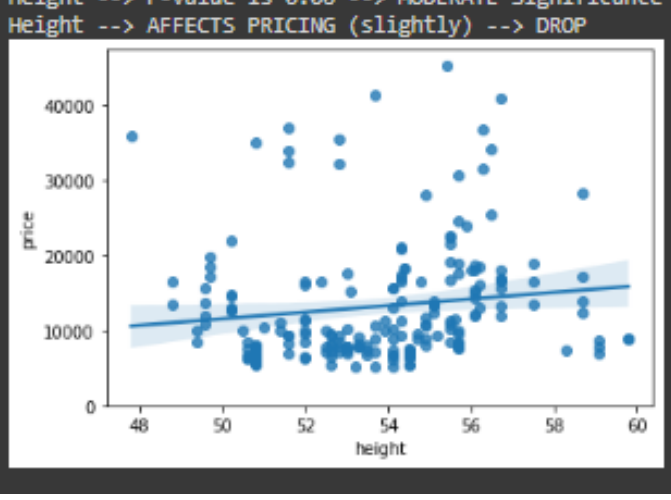

1. KEEP WIDTH

### E. IS HEIGHT IMPORTANT?

2d) Is 'Height' Important?

```
sns.regplot(x = "height", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['height'], df['price'])

print("Height --> Correlation Coefficient is {:.2f} --> Weak Positive Correlation".format(pearson_coef))
print("Height --> P-value is {:.2f} --> MODERATE Significance".format(p_value))
print("Height --> AFFECTS PRICING (slightly) --> DROP")
```

```
Height --> Correlation Coefficient is 0.14 --> Weak Positive Correlation
Height --> P-value is 0.06 --> MODERATE Significance
Height --> AFFECTS PRICING (slightly) --> DROP
```



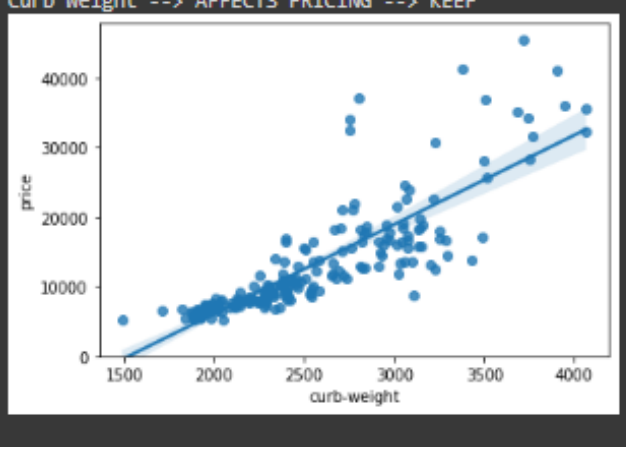

1. DROP HEIGHT

## F. IS CURB WEIGHT IMPORTANT?

2e) Is 'Curb-Weight' Important?

```
sns.regplot(x = "curb-weight", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['curb-weight'], df['price'])

print("Curb Weight --> Correlation Coefficient is {:.2f} --> STRONG Positive Correlation".format(pearson_coef))
print("Curb Weight --> P-value is {:.2f} --> STRONG Significance".format(p_value))
print("Curb Weight --> AFFECTS PRICING --> KEEP")
```

```
Curb Weight --> Correlation Coefficient is 0.83 --> STRONG Positive Correlation
Curb Weight --> P-value is 0.00 --> STRONG Significance
Curb Weight --> AFFECTS PRICING --> KEEP
```



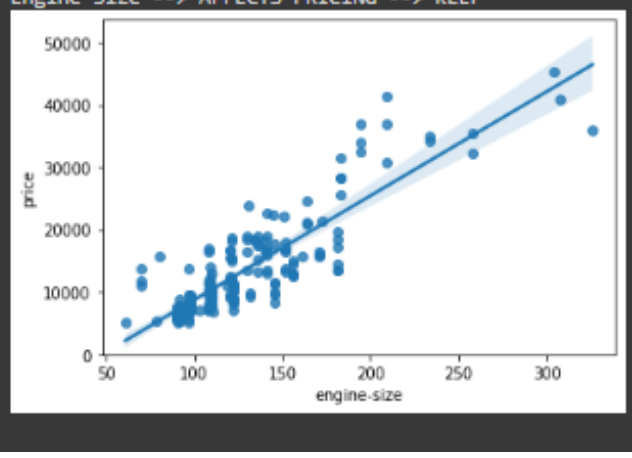

1. KEEP CURB WEIGHT

## G. IS ENGINE SIZE IMPORTANT?

2f) Is 'Engine-Size' Important?

```
sns.regplot(x = "engine-size", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['engine-size'], df['price'])

print("Engine Size --> Correlation Coefficient is {:.2f} --> STRONG Positive Correlation".format(pearson_coef))
print("Engine Size --> P-value is {:.2f} --> STRONG Significance".format(p_value))
print("Engine Size --> AFFECTS PRICING --> KEEP")
```

```
Engine Size --> Correlation Coefficient is 0.87 --> STRONG Positive Correlation
Engine Size --> P-value is 0.00 --> STRONG Significance
Engine Size --> AFFECTS PRICING --> KEEP
```



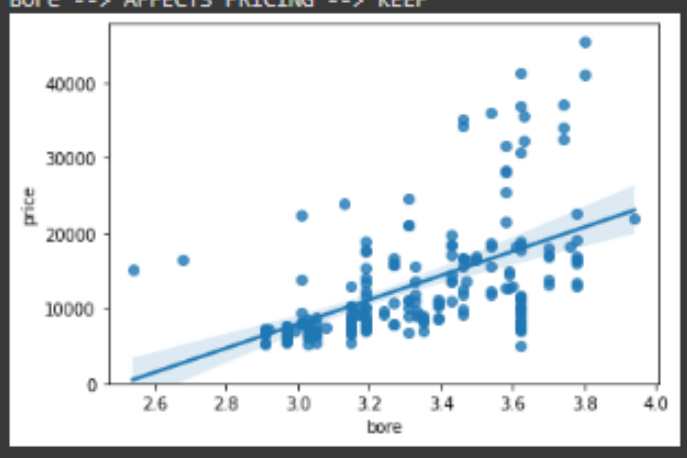

1. KEEP ENGINE SIZE

## H. IS BORE IMPORTANT?

2g) Is 'Bore' Important?

```
sns.regplot(x = "bore", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['bore'], df['price'])

print("Bore --> Correlation Coefficient is {:.2f} --> OK Positive Correlation".format(pearson_coef))
print("Bore --> P-value is {:.2f} --> STRONG Significance".format(p_value))
print("Bore --> AFFECTS PRICING --> KEEP")
```

```
Bore --> Correlation Coefficient is 0.54 --> OK Positive Correlation
Bore --> P-value is 0.00 --> STRONG Significance
Bore --> AFFECTS PRICING --> KEEP
```




1. KEEP BORE

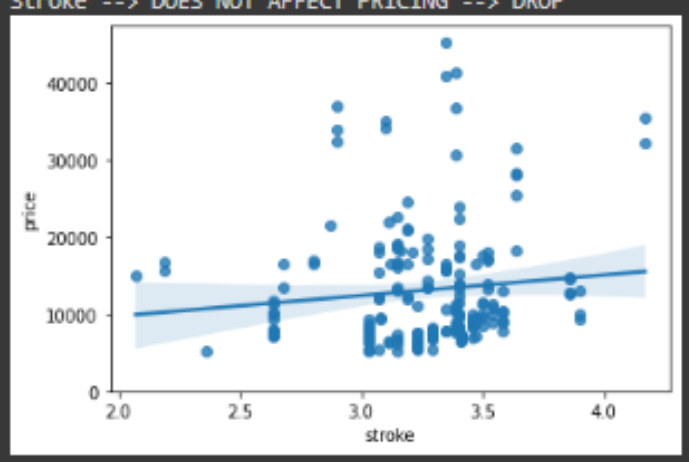
## I. IS STROKE IMPORTANT?

2h) Is "Stroke" Important?

```
df2 = df.dropna()
```

```
sns.regplot(x = "stroke", y = "price", data = df2)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df2['stroke'], df2['price'])

print("Stroke --> Correlation Coefficient is {:.2f} --> VERY WEAK Correlation".format(pearson_coef))
print("Stroke --> P-value is {:.2f} --> NO Significance at all".format(p_value))
print("Stroke --> DOES NOT AFFECT PRICING --> DROP")
```

```
Stroke --> Correlation Coefficient is 0.11 --> VERY WEAK Correlation
Stroke --> P-value is 0.13 --> NO Significance at all
Stroke --> DOES NOT AFFECT PRICING --> DROP
```



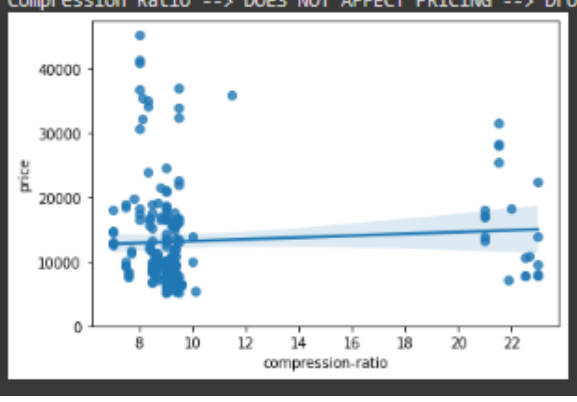

1. DROP STROKE

## J. IS COMPRESSION RATIO IMPORTANT?

2i) Is "Compression-Ratio" Important?

```
sns.regplot(x = "compression-ratio", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['compression-ratio'], df['price'])

print("Compression Ratio --> Correlation Coefficient is {:.2f} --> NO Positive Correlation AT ALL".format(pearson_coef))
print("Compression Ratio --> P-value is {:.2f} --> NO Significance".format(p_value))
print("Compression Ratio --> DOES NOT AFFECT PRICING --> Drop")

#the plot below seems to show a misfit for linear regression
#perhaps should use clustering instead
```

```
Compression Ratio --> Correlation Coefficient is 0.07 --> NO Positive Correlation AT ALL
Compression Ratio --> P-value is 0.32 --> NO Significance
Compression Ratio --> DOES NOT AFFECT PRICING --> Drop
```



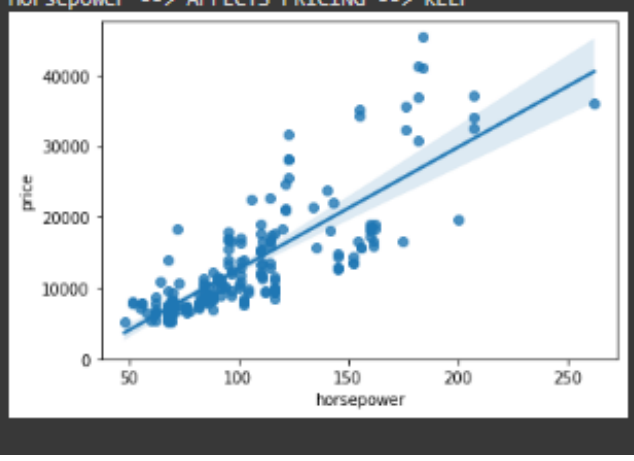

1. DROP COMPRESSION RATIO

## K. IS HORSEPOWER IMPORTANT?

2j) Is "Horspower" Important?

```
sns.regplot(x = "horsepower", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['horsepower'], df['price'])

print("Horsepower --> Correlation Coefficient is {:.2f} --> STRONG Positive Correlation".format(pearson_coef))
print("Horsepower --> P-value is {:.2f} --> STRONG Significance".format(p_value))
print("Horsepower --> AFFECTS PRICING --> KEEP")
```

```
Horsepower --> Correlation Coefficient is 0.81 --> STRONG Positive Correlation
Horsepower --> P-value is 0.00 --> STRONG Significance
Horsepower --> AFFECTS PRICING --> KEEP
```



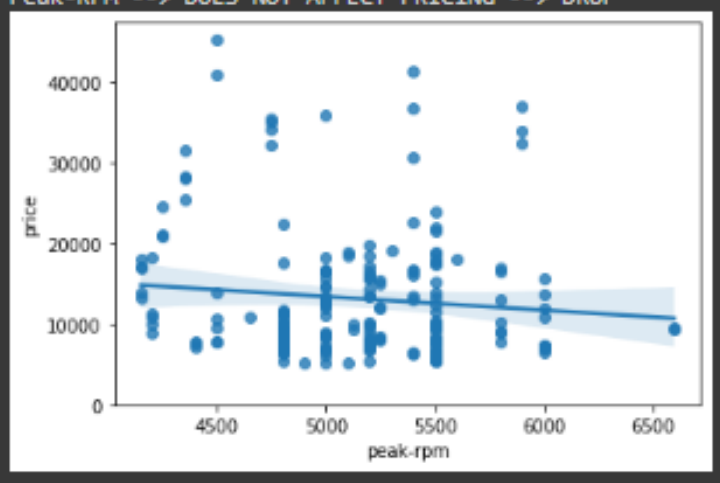

1. KEEP HORSEPOWER

## L. IS PEAK RPM IMPORTANT?

2k) Is "Peak-RPM" Important?

```
sns.regplot(x = "peak-rpm", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['peak-rpm'], df['price'])

print("Peak-RPM --> Correlation Coefficient is {:.2f} --> NO Correlation".format(pearson_coef))
print("Peak-RPM --> P-value is {:.2f} --> NO Significance".format(p_value))
print("Peak-RPM --> DOES NOT AFFECT PRICING --> DROP")
```

```
Peak-RPM --> Correlation Coefficient is -0.10 --> NO Correlation
Peak-RPM --> P-value is 0.15 --> NO Significance
Peak-RPM --> DOES NOT AFFECT PRICING --> DROP
```



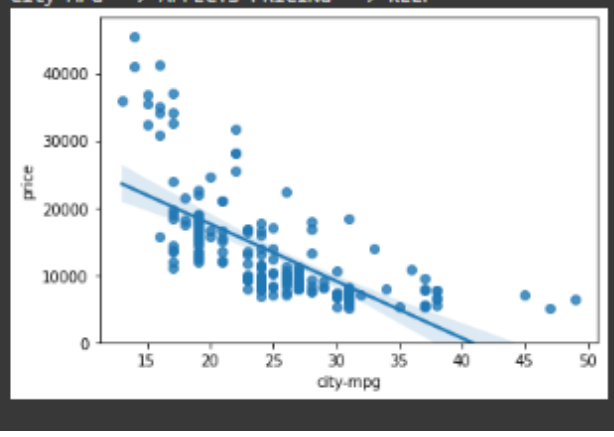

1. DROP PEAK RPM

## M. IS CITY MPG IMPORTANT?

2l) Is "City-mpg" Important?

```
sns.regplot(x = "city-mpg", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['city-mpg'], df['price'])

print("City-MPG --> Correlation Coefficient is {:.2f} --> Quite Strong Negative Correlation".format(pearson_coef))
print("City-MPG --> P-value is {:.2f} --> STRONG Significance".format(p_value))
print("City-MPG --> AFFECTS PRICING --> KEEP")
```

```
City-MPG --> Correlation Coefficient is -0.69 --> Quite Strong Negative Correlation
City-MPG --> P-value is 0.00 --> STRONG Significance
City-MPG --> AFFECTS PRICING --> KEEP
```



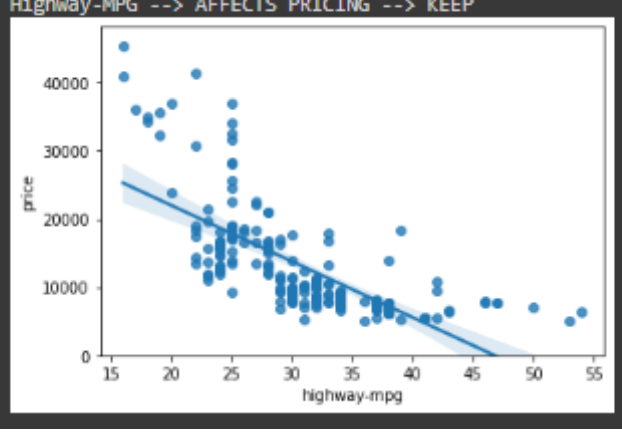

1. KEEP CITY MPG

## N. IS HIGHWAY MPG IMPORTANT?

2m) Is "Highway - mpg" Important?

```
sns.regplot(x = "highway-mpg", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['highway-mpg'], df['price'])

print("Highway-MPG --> Correlation Coefficient is {:.2f} --> Strong Negative Correlation".format(pearson_coef))
print("Highway-MPG --> P-value is {:.2f} --> STRONG Significance".format(p_value))
print("Highway-MPG --> AFFECTS PRICING --> KEEP")
```

```
Highway-MPG --> Correlation Coefficient is -0.70 --> Strong Negative Correlation
Highway-MPG --> P-value is 0.00 --> STRONG Significance
Highway-MPG --> AFFECTS PRICING --> KEEP
```



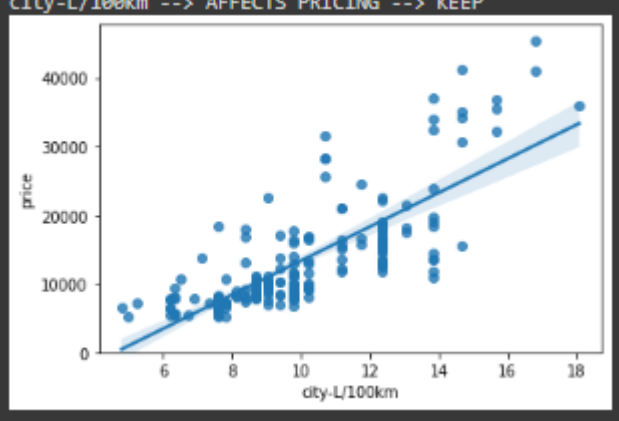

1. KEEP HIGHWAY MPG

## O. IS CITY – L/100KM IMPORTANT?

2n) Is "City-L/100km" Important?

```
sns.regplot(x = "city-L/100km", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['city-L/100km'], df['price'])

print("city-L/100km --> Correlation Coefficient is {:.2f} --> Strong Positive Correlation".format(pearson_coef))
print("city-L/100km --> P-value is {:.2f} --> STRONG Significance".format(p_value))
print("city-L/100km --> AFFECTS PRICING --> KEEP")
```

```
city-L/100km --> Correlation Coefficient is 0.79 --> Strong Negative Correlation
city-L/100km --> P-value is 0.00 --> STRONG Significance
city-L/100km --> AFFECTS PRICING --> KEEP
```



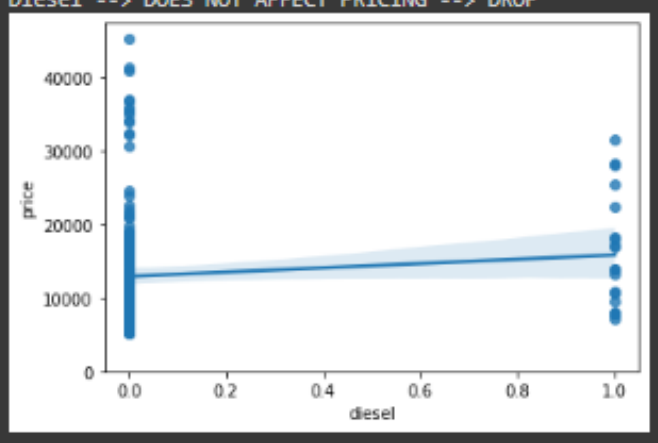

1. KEEP CITY-L/100KM

## P. IS DIESEL IMPORTANT?

2o) Is "Diesel" Important?

```
sns.regplot(x = "diesel", y = "price", data = df)
plt.ylim(0,)

pearson_coef, p_value = stats.pearsonr(df['diesel'], df['price'])

print("Diesel --> Correlation Coefficient is {:.2f} --> Weak Positive Correlation".format(pearson_coef))
print("Diesel --> P-value is {:.2f} --> NO Significance".format(p_value))
print("Diesel --> DOES NOT AFFECT PRICING --> DROP")
```

```
Diesel --> Correlation Coefficient is 0.11 --> Weak Positive Correlation
Diesel --> P-value is 0.12 --> NO Significance
Diesel --> DOES NOT AFFECT PRICING --> DROP
```



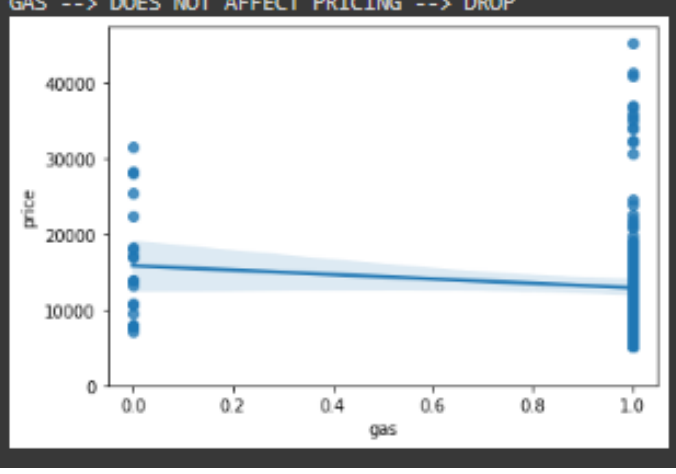

1. DROP DIESEL

## Q. IS GAS IMPORTANT?

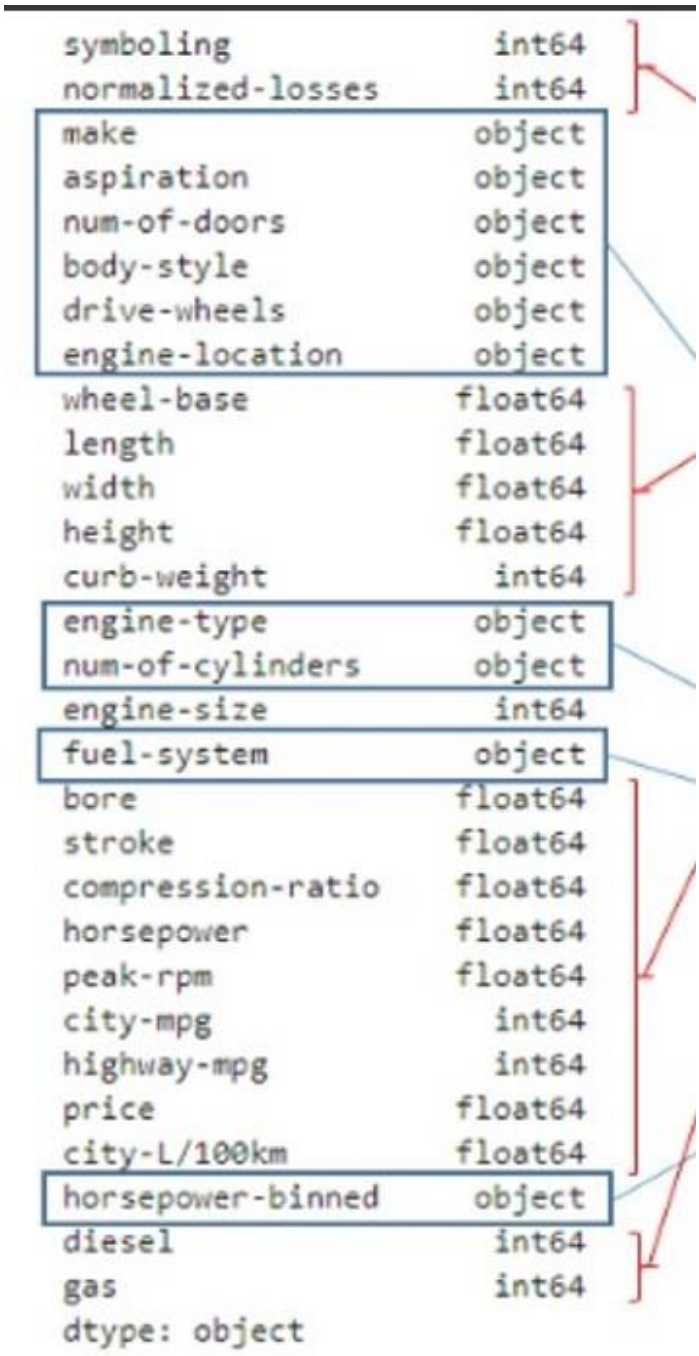
2p) Is "Gas" Important?

```
[14] sns.regplot(x = "gas", y = "price", data = df)
     plt.ylim(0,)

     pearson_coef, p_value = stats.pearsonr(df['gas'], df['price'])

     print("GAS --> Correlation Coefficient is {:.2f} --> Weak Negative Correlation".format(pearson_coef))
     print("GAS --> P-value is {:.2f} --> NO Significance".format(p_value))
     print("GAS --> DOES NOT AFFECT PRICING --> DROP")
```

```
GAS --> Correlation Coefficient is -0.11 --> Weak Negative Correlation
GAS --> P-value is 0.12 --> NO Significance
GAS --> DOES NOT AFFECT PRICING --> DROP
```




1. DROP GAS



## A. IS MAKE IMPORTANT?

3a) Is "Make" Important?

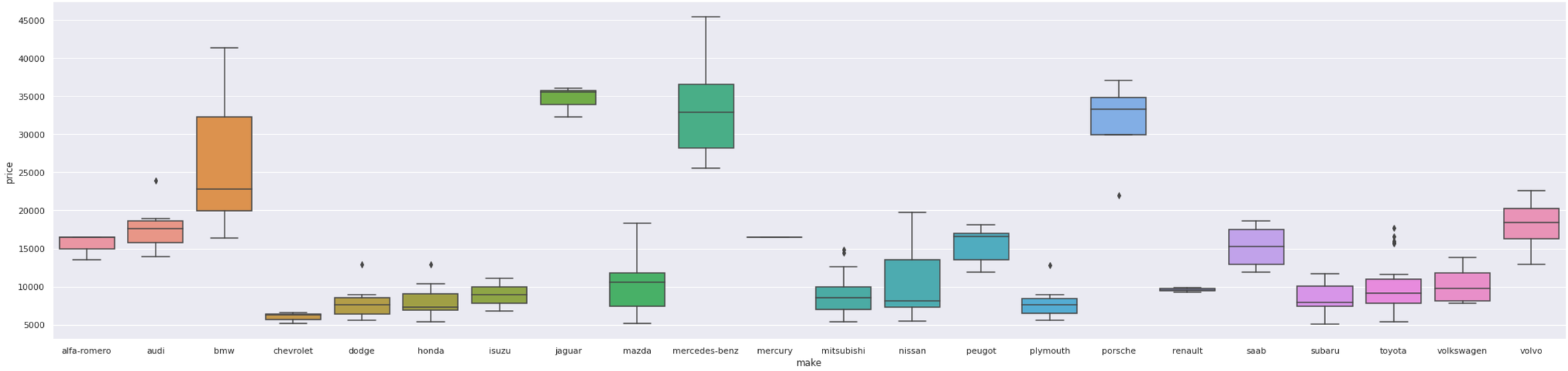
```
make_counts = df['make'].value_counts().to_frame()
make_counts.rename(columns={'make':'value_counts'},inplace=True)
make_counts.index.name = 'make'
make_counts
```

| make | value_counts |
|---|---|
| toyota | 32 |
| nissan | 18 |
| mazda | 17 |
| mitsubishi | 13 |
| honda | 13 |
| volkswagen | 12 |
| subaru | 12 |
| peugot | 11 |
| volvo | 11 |
| dodge | 9 |
| mercedes-benz | 8 |
| bmw | 8 |
| plymouth | 7 |
| audi | 6 |
| saab | 6 |
| porsche | 4 |
| jaguar | 3 |
| chevrolet | 3 |
| alfa-romero | 3 |
| isuzu | 2 |
| renault | 2 |
| mercury | 1 |

1. GLOBAL TEST

3a)(i) Global Test

```
sns.set(rc = {'figure.figsize':(35,8)})
sns.boxplot(x = "make", y = "price", data = df)
```



```
from statsmodels.stats.api import anova_lm

model_make = ols('price ~ make', df).fit()
anova_lm(model_make)

#H0: There is no difference between Makes
#H1: There is a difference between Makes
#P value << alpha (0.05)
#Accept H1: there IS a difference.
```

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| **make** | 21.0 | 1.005272e+10 | 4.787009e+08 | 33.232103 | 1.068343e-50 |
| **Residual** | 179.0 | 2.578454e+09 | 1.440477e+07 | NaN | NaN |

```
print("Make affects Price --> KEEP")
```

```
Make affects Price --> KEEP
```

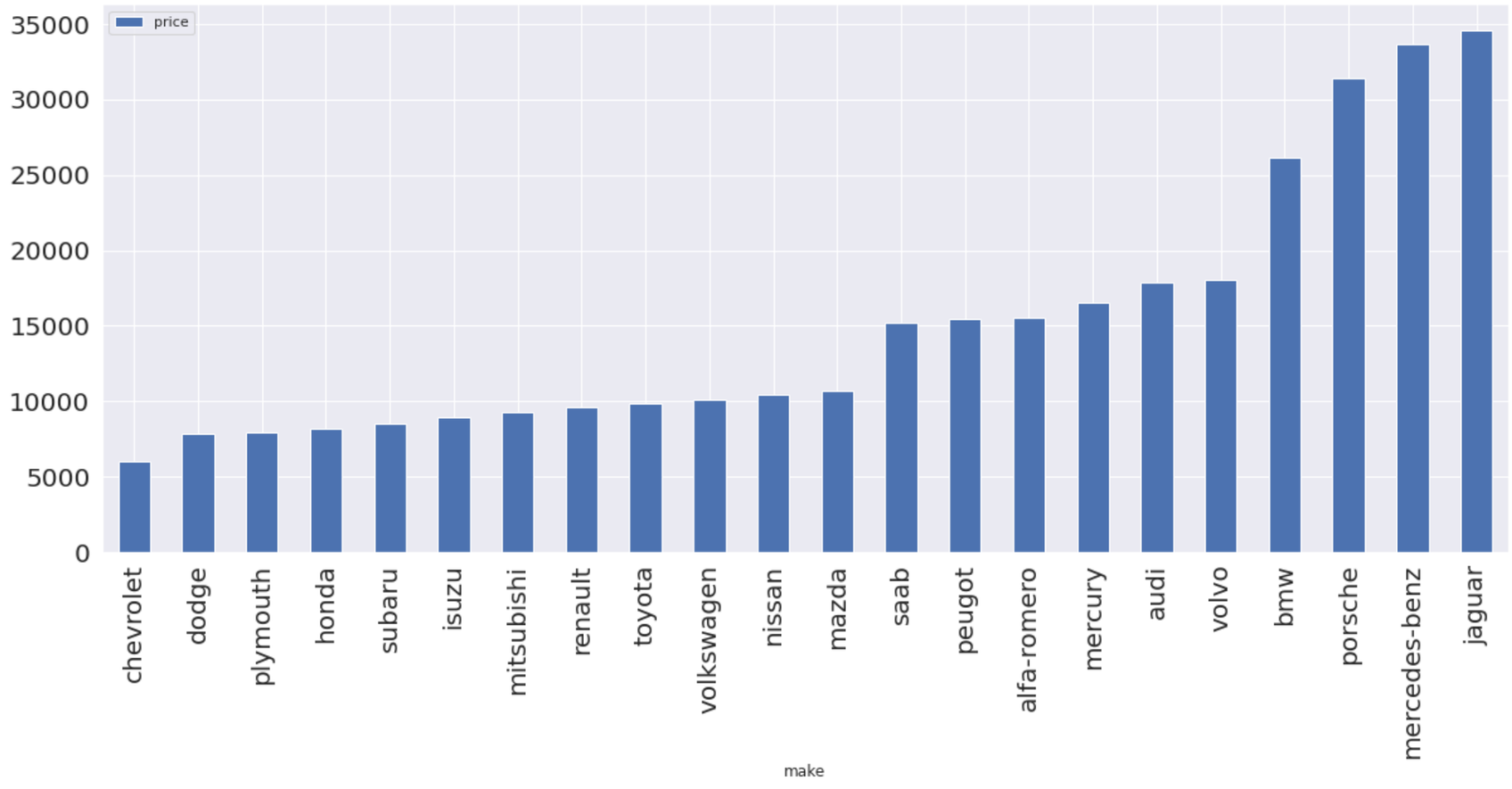
*a) KEEP Make*

2. INDIVIDUAL TEST

3a)(ii) Individual Test

```
[38] df_make_price = df[["make","price"]]\
                        .groupby("make")\
                        .mean()\
                        .sort_values(by="price", ascending = True)
```

```
df_make_price.plot(kind = 'bar', figsize=  (20,8), fontsize=20, rot = 90)
```



*a) Comparing Honda vs Subaru Pricing*

Comparing Honda vs Subaru Pricing

```
[103] from scipy import stats

    df_anova_make = df[["make", "price"]]
    grouped_anova_make = df_anova_make.groupby(["make"])

    anova_results_l = stats.f_oneway(
        grouped_anova_make.get_group("honda")["price"],
        grouped_anova_make.get_group("subaru")["price"])

    anova_results_l
```

```
F_onewayResult(statistic=0.19744030127462606, pvalue=0.6609478240622193)
```

```
print("P value = 0.66 --> NO DIFFERENCE between Honda vs Subaru Pricing")
```

```
P value = 0.66 --> NO DIFFERENCE between Honda vs Subaru Pricing
```

+ Code + Text

*b) Comparing Honda vs Peugot vs Porsche Pricing*

Comparing Honda vs Peugot vs Porsche Pricing

```
[104] anova_results_2 = stats.f_oneway(
          grouped_anova_make.get_group("honda")["price"],
          grouped_anova_make.get_group("peugot")["price"],
          grouped_anova_make.get_group("porsche")["price"])

      anova_results_2
```

```
F_onewayResult(statistic=91.08407300042406, pvalue=3.3129346197245934e-12)
```

```
[67] print("P value = 0 --> GOT DIFFERENCE between Honda vs Peugot vs Porsche Pricing")
```

```
P value = 0 --> GOT DIFFERENCE between Honda vs Peugot vs Porsche Pricing
```

## B. IS ASPIRATION IMPORTANT?

3b) Is "Aspiration" Important?

```
[73] aspiration_counts = df['aspiration'].value_counts().to_frame()
     aspiration_counts.rename(columns={'aspiration':'value_counts'},inplace=True)
     aspiration_counts.index.name = 'aspiration'
     aspiration_counts
```

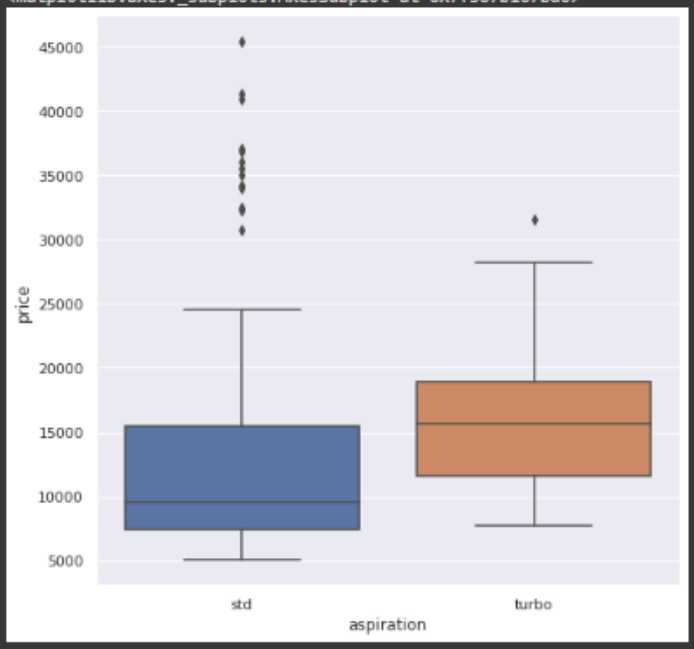
| aspiration | value_counts |
|---|---|
| std | 165 |
| turbo | 36 |

### 1. GLOBAL TEST

3b)(i) Global Test

```
sns.set(rc = {'figure.figsize':(8,8)})
sns.boxplot(x = "aspiration", y = "price", data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307b107bd0>
```



```
from statsmodels.formula.api import ols
from statsmodels.stats.api import anova_lm

model_aspiration = ols('price ~ aspiration', df).fit()
anova_lm(model_aspiration)

#H0: There is no difference between Aspiration
#H1: There is a difference between Aspiration
#P value > alpha (0.01)
#Accept H0: there IS NO difference.
```

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| **aspiration** | 1.0 | 4.073355e+08 | 4.073355e+08 | 6.631286 | 0.010746 |
| **Residual** | 199.0 | 1.222384e+10 | 6.142632e+07 | NaN | NaN |

```
[78] print("Aspiration DOES NOT affect Price --> DROP")
```

```
Aspiration DOES NOT affect Price --> DROP
```

*a) DROP Aspiration*

## C. IS NUMBER OF DOORS IMPORTANT?

3c) Is "Num-Of-Doors" Important?

```
[80] num_of_doors_counts = df['num-of-doors'].value_counts().to_frame()
     num_of_doors_counts.rename(columns={'num-of-doors':'value_counts'},inplace=True)
     num_of_doors_counts.index.name = 'num_of_doors'
     num_of_doors_counts
```

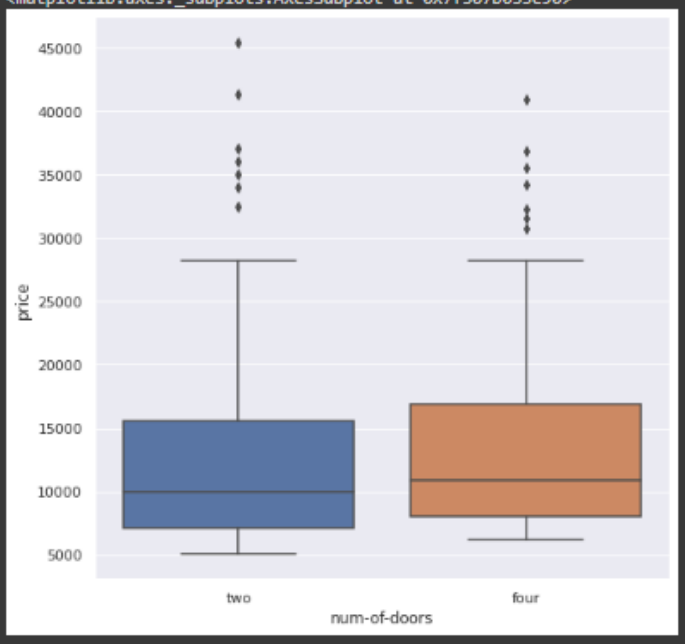
| num_of_doors | value_counts |
|---|---|
| four | 115 |
| two | 86 |

1. GLOBAL TEST

3c)(i) Global Test

```
sns.set(rc = {'figure.figsize':(8,8)})
sns.boxplot(x = "num-of-doors", y = "price", data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307b033e90>
```



```
[96] from statsmodels.formula.api import ols
     from statsmodels.stats.api import anova_lm

     numofdoors = df["num-of-doors"]

     model_numofdoors = ols('price ~ numofdoors', df).fit()
     anova_lm(model_numofdoors)

     #H0: There is no difference between Num-Of-Doors
     #H1: There is a difference between Num-Of-Doors
     #P value > alpha (0.05)
     #Accept H0: there IS NO difference.
```

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| numofdoors | 1.0 | 2.274569e+07 | 2.274569e+07 | 0.358997 | 0.549745 |
| Residual | 199.0 | 1.260843e+10 | 6.335893e+07 | NaN | NaN |

```
[91] print("Num Of Doors DOES NOT affect Price --> DROP")
```

```
Num Of Doors DOES NOT affect Price --> DROP
```

*a) DROP Num of Doors*

## D. IS BODY STYLE IMPORTANT?

3d) Is "Body-Style" Important?

```
body_style_counts = df['body-style'].value_counts().to_frame()
body_style_counts.rename(columns={'body-style':'value_counts'},inplace=True)
body_style_counts.index.name = 'body-style'
body_style_counts
```

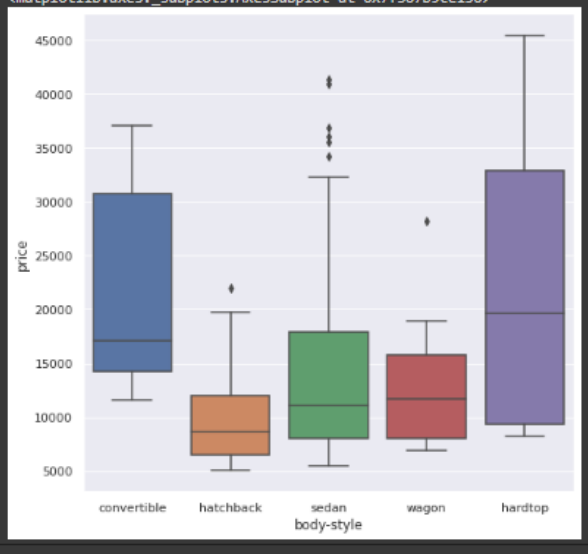
| body-style | value_counts |
|---|---|
| sedan | 94 |
| hatchback | 68 |
| wagon | 25 |
| hardtop | 8 |
| convertible | 6 |

1. GLOBAL TEST

3d)(i) Global Test

```
sns.set(rc = {'figure.figsize':(8,8)})
sns.boxplot(x = "body-style", y = "price", data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307b9ce150>
```



```
[97] from statsmodels.formula.api import ols
     from statsmodels.stats.api import anova_lm

     bodystyle = df["body-style"]

     model_bs = ols('price ~ bodystyle', df).fit()
     anova_lm(model_bs)

     #H0: There is no difference between Body Style
     #H1: There is a difference between Body Style
     #P value < alpha (0.05)
     #Accept H1: there IS difference.
```

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| bodystyle | 4.0 | 1.983646e+09 | 4.959114e+08 | 9.128752 | 8.779795e-07 |
| Residual | 196.0 | 1.064753e+10 | 5.432412e+07 | NaN | NaN |

```
[98] print("Body Style affects Price --> KEEP")

     Body Style affects Price --> KEEP
```

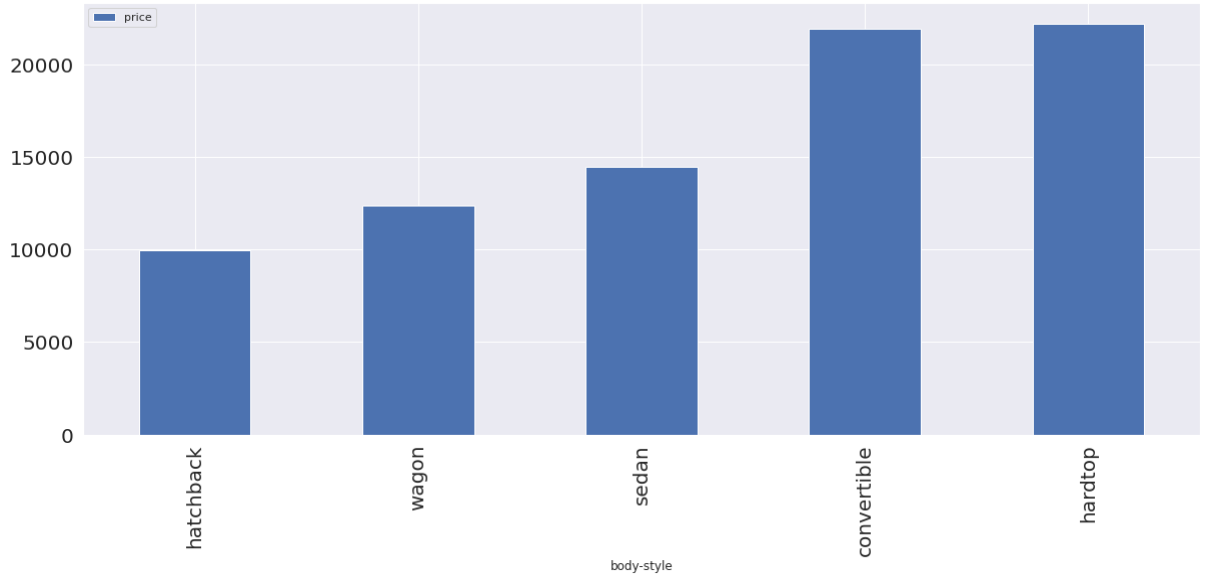
*a) KEEP Body Style*

2. INDIVIDUAL TEST

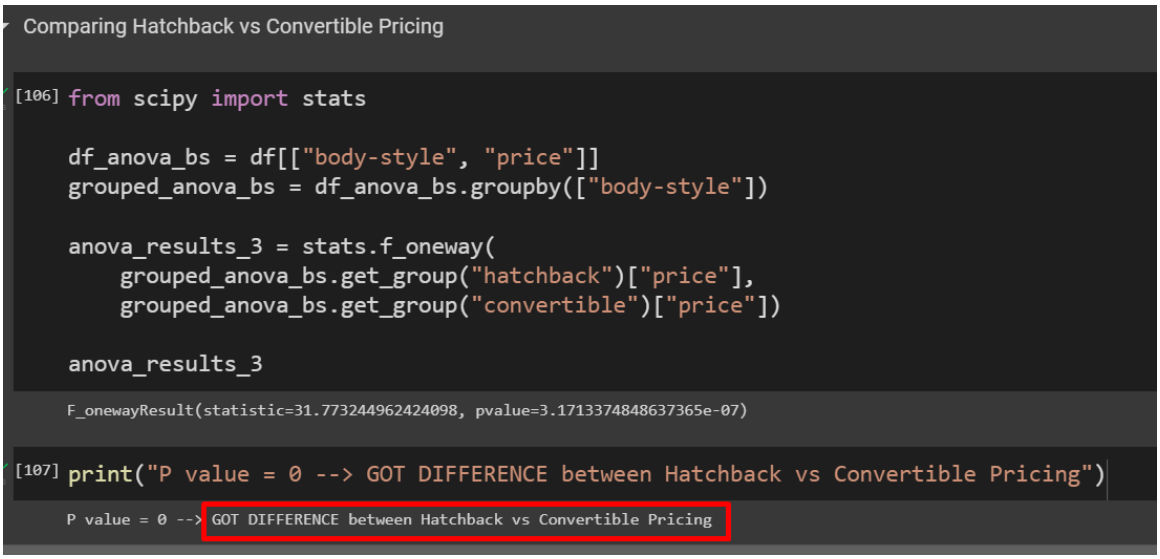
3d)(ii) Individual Test

```
[99] df_bodystyle_price = df[["body-style","price"]]\
                          .groupby("body-style")\
                          .mean()\
                          .sort_values(by="price", ascending = True)
```

```
df_bodystyle_price.plot(kind = 'bar', figsize=  (20,8), fontsize=20, rot = 90)
```



*a) Comparing Hatchback vs Convertible Pricing*



## E. IS DRIVE WHEELS IMPORTANT?

3e) Is "Drive Wheels" Important?

```
[108] drive_wheels_counts = df['drive-wheels'].value_counts().to_frame()
      drive_wheels_counts.rename(columns={'drive-wheels':'value_counts'},inplace=True)
      drive_wheels_counts.index.name = 'drive-wheels'
      drive_wheels_counts
```

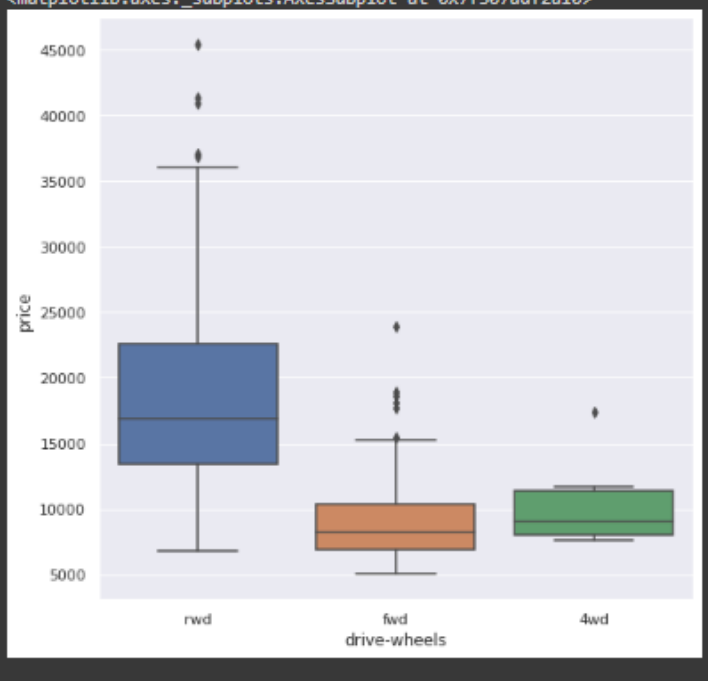
| drive-wheels | value_counts |
|---|---|
| fwd | 118 |
| rwd | 75 |
| 4wd | 8 |

### 1. GLOBAL TEST

3e)(i) Global Test

```
sns.set(rc = {'figure.figsize':(8,8)})
sns.boxplot(x = "drive-wheels", y = "price", data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307adf2a10>
```



```
[110] from statsmodels.formula.api import ols
      from statsmodels.stats.api import anova_lm

      drivewheels = df["drive-wheels"]

      model_dw = ols('price ~ drivewheels', df).fit()
      anova_lm(model_dw)

      #H0: There is no difference between Drive Wheels
      #H1: There is a difference between Drive Wheels
      #P value < alpha (0.05)
      #Accept H1: there IS difference.
```

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| drivewheels | 2.0 | 5.141172e+09 | 2.570586e+09 | 67.954065 | 3.394544e-23 |
| Residual | 198.0 | 7.490001e+09 | 3.782829e+07 | NaN | NaN |

```
[111] print("Drive Wheels affects Price --> KEEP")

      Drive Wheels affects Price --> KEEP
```

*a) KEEP Drive Wheels*

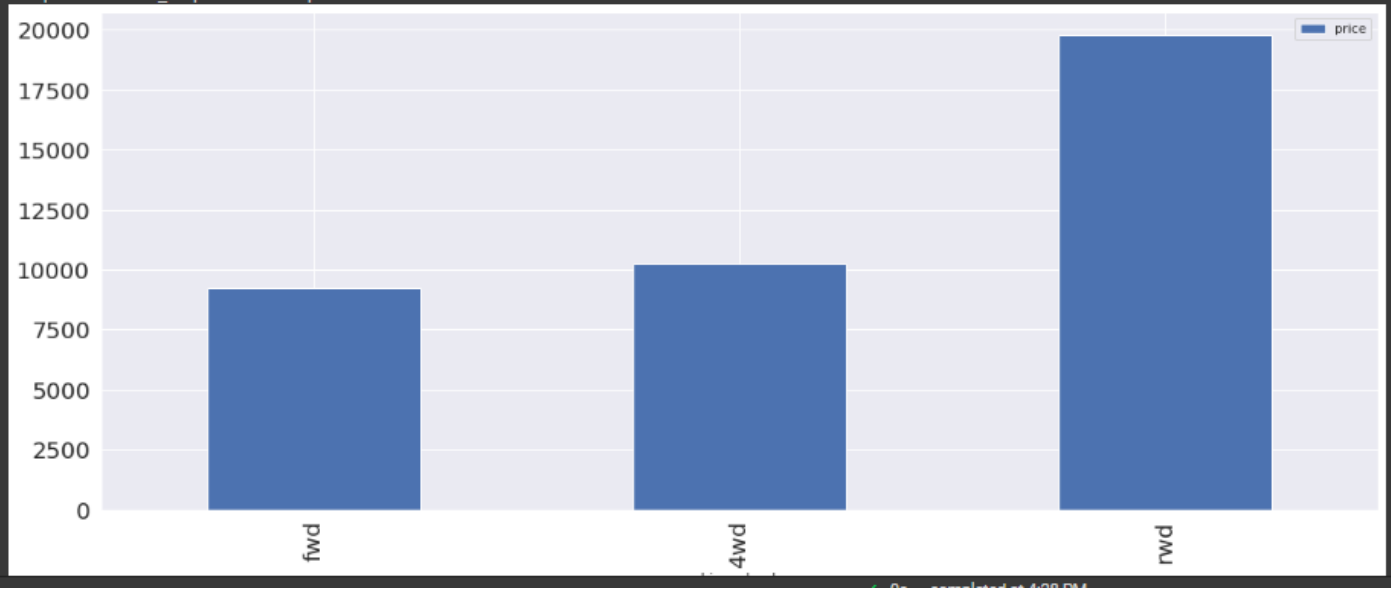
2. INDIVIDUAL TEST

3e)(ii) Individual Test

```
[112] df_drivewheels_price = df[["drive-wheels","price"]]\
                        .groupby("drive-wheels")\
                        .mean()\
                        .sort_values(by="price", ascending = True)
```

```
df_drivewheels_price.plot(kind = 'bar', figsize=  (20,8), fontsize=20, rot = 90)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307acdb9d0>
```



*a) Comparing FWD vs 4WD*

Comparing FWD vs 4WD

```
[115] from scipy import stats

      df_anova_dw = df[["drive-wheels", "price"]]
      grouped_anova_dw = df_anova_dw.groupby(["drive-wheels"])

      anova_results_4 = stats.f_oneway(
          grouped_anova_dw.get_group("fwd")["price"],
          grouped_anova_dw.get_group("4wd")["price"])

      anova_results_4
```

```
F_onewayResult(statistic=0.6654657502523033, pvalue=0.41620116697845666)
```

```
[116] print("P value = 0.4 --> NO DIFFERENCE between FWD vs 4WD")
```

```
P value = 0.4 --> NO DIFFERENCE between FWD vs 4WD
```

## F. IS ENGINE LOCATION IMPORTANT?

### 3f) Is "Engine-Location" Important?

```
[117] engine_location_counts = df['engine-location'].value_counts().to_frame()
      engine_location_counts.rename(columns={'engine-location':'value_counts'},inplace=True)
      engine_location_counts.index.name = 'engine-location'
      engine_location_counts
```

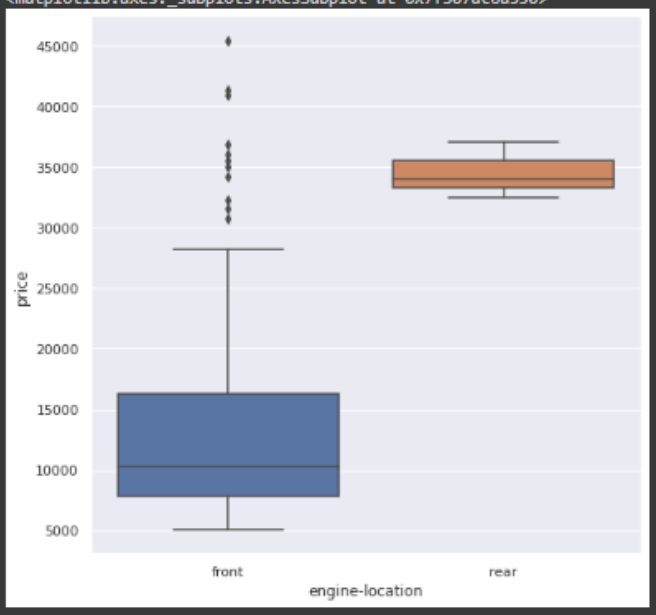
| engine-location | value_counts |
|---|---|
| front | 198 |
| rear | 3 |

1. GLOBAL TEST

### 3f)(i) Global Test

```
[118] sns.set(rc = {'figure.figsize':(8,8)})
      sns.boxplot(x = "engine-location", y = "price", data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307ac6a550>
```



```
[119] from statsmodels.formula.api import ols
      from statsmodels.stats.api import anova_lm

      engineloc = df["engine-location"]

      model_el = ols('price ~ engineloc', df).fit()
      anova_lm(model_el)

      #H0: There is no difference between Engine Location
      #H1: There is a difference between Engine Location
      #P value < alpha (0.05)
      #Accept H1: there IS difference.
```

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| engineloc | 1.0 | 1.384401e+09 | 1.384401e+09 | 24.49555 | 0.000002 |
| Residual | 199.0 | 1.124677e+10 | 5.651644e+07 | NaN | NaN |

```
[120] print("Engine Location affects Price --> KEEP")

      Engine Location affects Price --> KEEP
```

*a) KEEP Engine Location*

## G. IS ENGINE TYPE IMPORTANT?

3g) Is "Engine Type" Important?

```
engine_type_counts = df['engine-type'].value_counts().to_frame()
engine_type_counts.rename(columns={'engine-type':'value_counts'},inplace=True)
engine_type_counts.index.name = 'engine-type'
engine_type_counts
```

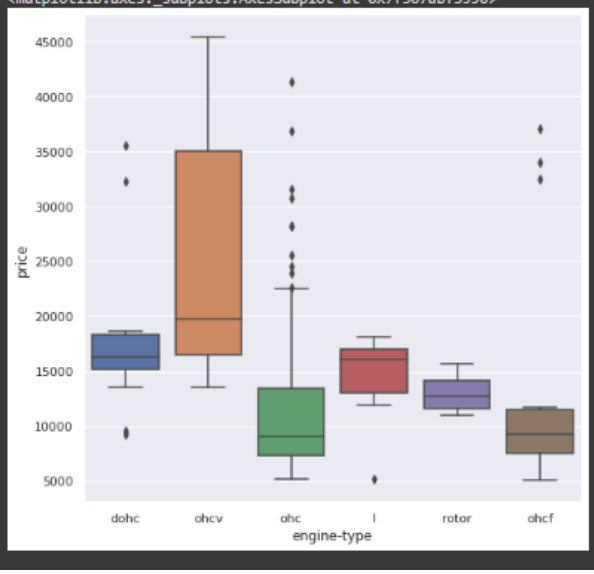
| engine-type | value_counts |
|---|---|
| ohc | 145 |
| ohcf | 15 |
| ohcv | 13 |
| dohc | 12 |
| [illegible] | 12 |
| rotor | 4 |

1. GLOBAL TEST

3g)(i) Global Test

```
sns.set(rc = {'figure.figsize':(8,8)})
sns.boxplot(x = "engine-type", y = "price", data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307abf3990>
```



```
[123] from statsmodels.formula.api import ols
      from statsmodels.stats.api import anova_lm

      enginetype = df["engine-type"]

      model_et = ols('price ~ enginetype', df).fit()
      anova_lm(model_et)

      #H0: There is no difference between Engine Type
      #H1: There is a difference between Engine Type
      #P value < alpha (0.05)
      #Accept H1: there IS difference.
```

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| enginetype | 5.0 | 2.545911e+09 | 5.091822e+08 | 9.845111 | 2.086549e-08 |
| Residual | 195.0 | 1.008526e+10 | 5.171929e+07 | NaN | NaN |

```
[124] print("Engine Type affects Price --> KEEP")
```

```
Engine Type affects Price --> KEEP
```

*a) KEEP Engine Type*

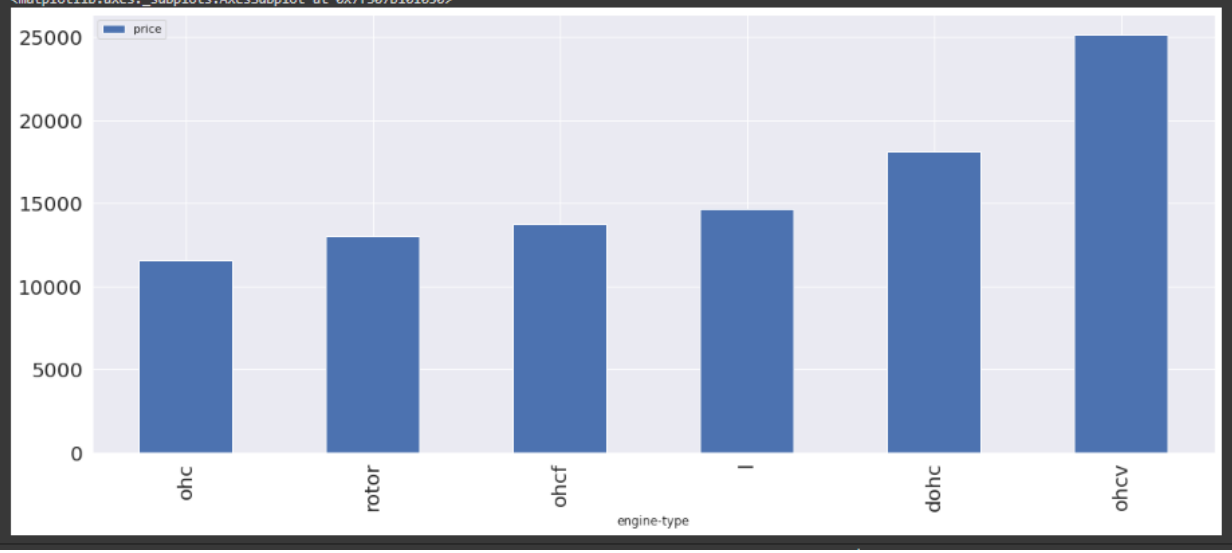
2. INDIVIDUAL TEST

3g)(ii) Individual Test

```
[125] df_enginetype_price = df[["engine-type","price"]]\
                          .groupby("engine-type")\
                          .mean()\
                          .sort_values(by="price", ascending = True)
```

```
df_enginetype_price.plot(kind = 'bar', figsize=  (20,8), fontsize=20, rot = 90)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307b101050>
```



*a) Comparing Rotor vs L vs OHCV*

Comparing Rotor vs L vs OHCV

```
[135] from scipy import stats

      df_anova_et = df[["engine-type", "price"]]
      grouped_anova_et = df_anova_et.groupby(["engine-type"])

      anova_results_5 = stats.f_oneway(
          grouped_anova_et.get_group("rotor")["price"],
          grouped_anova_et.get_group("l")["price"],
          grouped_anova_et.get_group("dohc")["price"])

      anova_results_5
```

```
F_onewayResult(statistic=1.623160633354665, pvalue=0.217383425909766)
```

```
[134] print("P value = 0.22 --> NO DIFFERENCE between ROTOR vs L vs DOHC")
```

```
P value = 0.22 --> NO DIFFERENCE between ROTOR vs L vs DOHC
```

## H. IS NUMBER OF CYLINDERS IMPORTANT?

3h) Is "Num-Of-Cylinders" Important?

```
no_of_cyl_counts = df['num-of-cylinders'].value_counts().to_frame()
no_of_cyl_counts.rename(columns={'num-of-cylinders':'value_counts'},inplace=True)
no_of_cyl_counts.index.name = 'num-of-cylinders'
no_of_cyl_counts
```

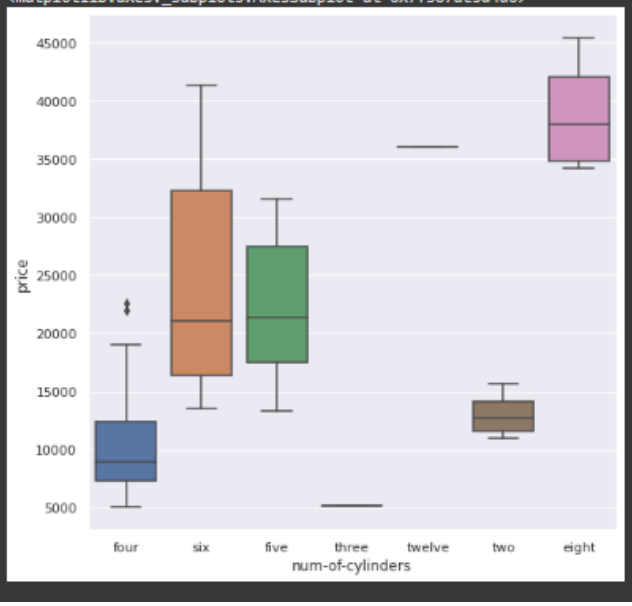
| num-of-cylinders | value_counts |
|---|---|
| four | 157 |
| six | 24 |
| five | 10 |
| two | 4 |
| eight | 4 |
| three | 1 |
| twelve | 1 |

1. GLOBAL TEST

3h)(i) Global Test

```
sns.set(rc = {'figure.figsize':(8,8)})
sns.boxplot(x = "num-of-cylinders", y = "price", data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307ac5d4d0>
```



```
[ ] from statsmodels.formula.api import ols
    from statsmodels.stats.api import anova_lm

    noc = df["num-of-cylinders"]

    model_noc = ols('price ~ noc', df).fit()
    anova_lm(model_noc)

    #H0: There is no difference between No. Of Cylinders
    #H1: There is a difference between No. Of Cylinders
    #P value < alpha (0.05)
    #Accept H1: there IS difference.
```

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| noc | 6.0 | 7.951727e+09 | 1.325288e+09 | 54.943653 | 2.870145e-39 |
| Residual | 194.0 | 4.679446e+09 | 2.412085e+07 | NaN | NaN |

```
[ ] print("Number of Cylinders affects Price --> KEEP")
```

```
Number of Cylinders affects Price --> KEEP
```

*a) KEEP Number of Cylinders*

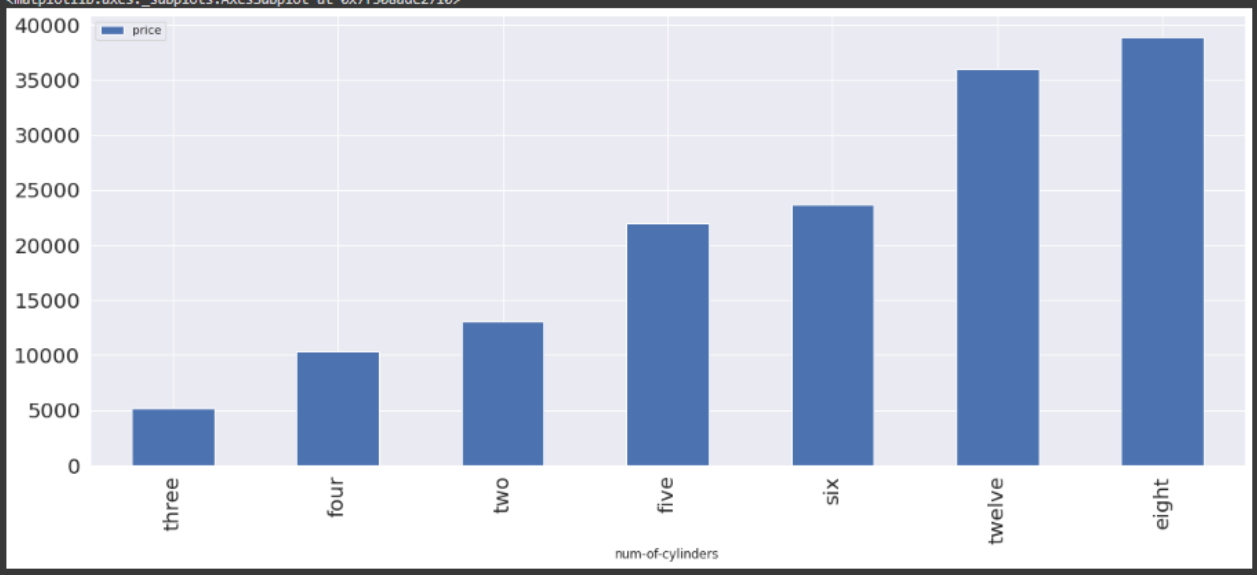
2. INDIVIDUAL TEST

3h)(ii) Individual Test

```
[ ] df_noc_price = df[["num-of-cylinders","price"]]\
                      .groupby("num-of-cylinders")\
                      .mean()\
                      .sort_values(by="price", ascending = True)
```

```
df_noc_price.plot(kind = 'bar', figsize=  (20,8), fontsize=20, rot = 90)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f308ade2710>
```



*a) Comparing 4 vs 6 vs 8 Cylinders*

## I. IS FUEL SYSTEM IMPORTANT?

3i) Is "Fuel-System" Important?

```
fs_counts = df['fuel-system'].value_counts().to_frame()
fs_counts.rename(columns={'fuel-system':'value_counts'},inplace=True)
fs_counts.index.name = 'fuel-system'
fs_counts
```

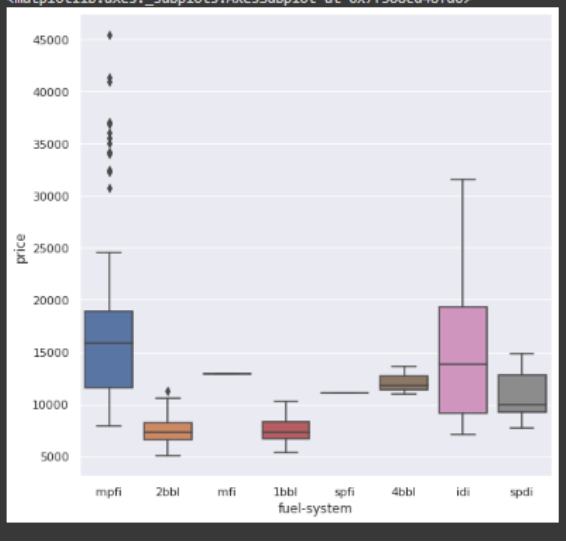
| fuel-system | value_counts |
|---|---|
| mpfi | 92 |
| 2bbl | 64 |
| idi | 20 |
| 1bbl | 11 |
| spdi | 9 |
| 4bbl | 3 |
| mfi | 1 |
| spfi | 1 |

1. GLOBAL TEST

3i)(i) Global Test

```
sns.set(rc = {'figure.figsize':(8,8)})
sns.boxplot(x = "fuel-system", y = "price", data = df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f308cd46fd0>



```
[ ] from statsmodels.formula.api import ols
    from statsmodels.stats.api import anova_lm

    fs = df["fuel-system"]

    model_fs = ols('price ~ fs', df).fit()
    anova_lm(model_fs)

    #H0: There is no difference between Fuel Systems
    #H1: There is a difference between Fuel Systems
    #P value < alpha (0.05)
    #Accept H1: there IS difference.
```

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| fs | 7.0 | 4.455276e+09 | 6.364679e+08 | 15.024444 | 1.314258e-15 |
| Residual | 193.0 | 8.175897e+09 | 4.236216e+07 | NaN | NaN |

```
print("Fuel Systems affect Price --> KEEP")
```

```
Fuel Systems affect Price --> KEEP
```

*a) KEEP Fuel Systems*

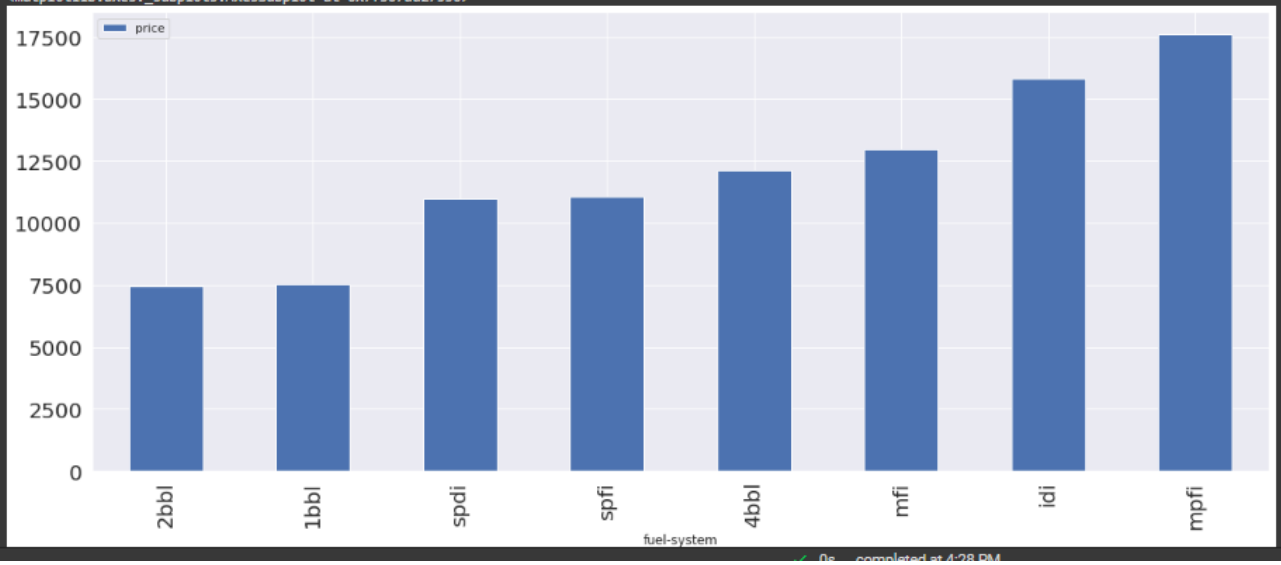
2. INDIVIDUAL TEST

3i)(ii) Individual Test

```
[ ] df_fs_price = df[["fuel-system","price"]]\
                    .groupby("fuel-system")\
                    .mean()\
                    .sort_values(by="price", ascending = True)
```

```
df_fs_price.plot(kind = 'bar', figsize=  (20,8), fontsize=20, rot = 90)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307aa27550>
```



*a) Comparing 2bbl vs mpfi*

Comparing 2bbl vs mpfi

```
[ ] from scipy import stats

    df_anova_fs = df[["fuel-system", "price"]]
    grouped_anova_fs = df_anova_fs.groupby(["fuel-system"])

    anova_results_7 = stats.f_oneway(
        grouped_anova_fs.get_group("2bbl")["price"],
        grouped_anova_fs.get_group("mpfi")["price"])

    anova_results_7
```

```
F_onewayResult(statistic=86.55178928944206, pvalue=1.290189545502851e-16)
```

```
[ ] print("P value = 0 --> GOT DIFFERENCE between 2bbl vs mpfi Fuel System")
```

```
P value = 0 --> GOT DIFFERENCE between 2bbl vs mpfi Fuel System
```

## J. IS HORSEPOWER IMPORTANT?

### 3j) Is "Horsepower" Important?

```
[ ] hp_counts = df['horsepower-binned'].value_counts().to_frame()
    hp_counts.rename(columns={'horsepower-binned':'value_counts'},inplace=True)
    hp_counts.index.name = 'horsepower-binned'
    hp_counts
```

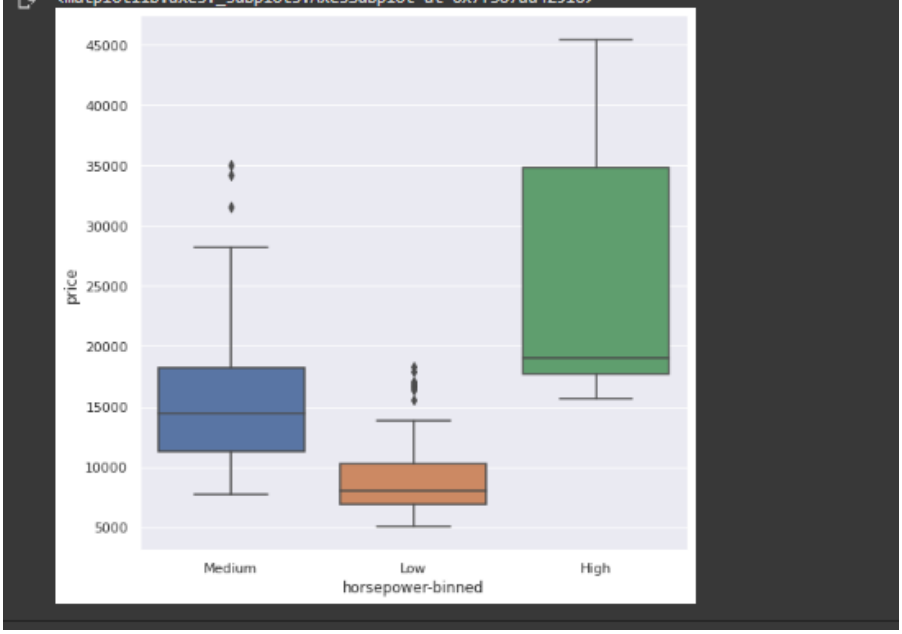
| horsepower-binned | value_counts |
|---|---|
| Low | 115 |
| Medium | 62 |
| High | 23 |

1. GLOBAL TEST

### 3j)(i) Global Test

```
sns.set(rc = {'figure.figsize':(8,8)})
sns.boxplot(x = "horsepower-binned", y = "price", data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307aa42910>
```



```
[ ] from statsmodels.formula.api import ols
    from statsmodels.stats.api import anova_lm

    hp = df["horsepower-binned"]

    model_hp = ols('price ~ hp', df).fit()
    anova_lm(model_hp)

    #H0: There is no difference between Horsepower
    #H1: There is a difference between Horsepower
    #P value < alpha (0.05)
    #Accept H1: there IS difference.
```

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| hp | 2.0 | 6.180537e+09 | 3.090268e+09 | 102.687105 | 2.810961e-31 |
| Residual | 197.0 | 5.928523e+09 | 3.009403e+07 | NaN | NaN |

```
[ ] print("Horsepower affect Price --> KEEP")

    Horsepower affect Price --> KEEP
```

*a) KEEP Horsepower*

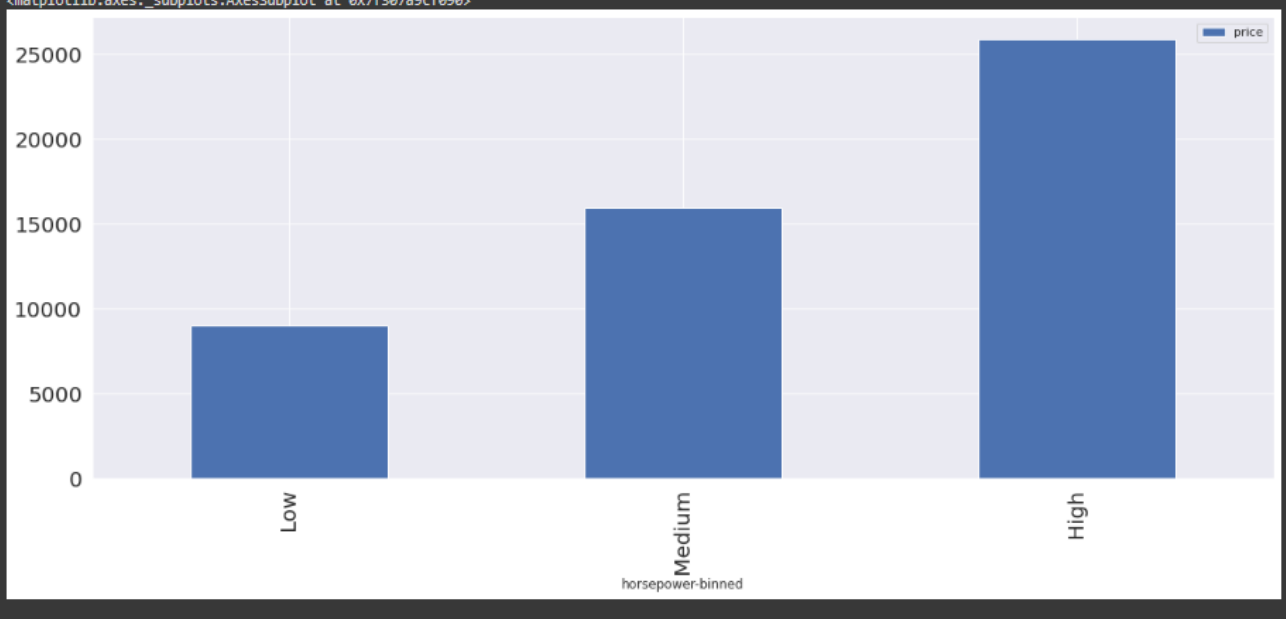
2. INDIVIDUAL TEST

3j)(ii) Individual Test

```
[ ] df_hp_price = df[["horsepower-binned","price"]]\
                    .groupby("horsepower-binned")\
                    .mean()\
                    .sort_values(by="price", ascending = True)
```

```
df_hp_price.plot(kind = 'bar', figsize=  (20,8), fontsize=20, rot = 90)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f307a9cf090>
```



*a) Comparing Low vs Medium Horsepower*

Comparing Low vs Medium Horsepower

```
[ ] from scipy import stats

    df_anova_hp = df[["horsepower-binned", "price"]]
    grouped_anova_hp = df_anova_hp.groupby(["horsepower-binned"])

    anova_results_8 = stats.f_oneway(
        grouped_anova_hp.get_group("Low")["price"],
        grouped_anova_hp.get_group("Medium")["price"])

    anova_results_8
```

```
F_onewayResult(statistic=91.8162651039427, pvalue=9.531067210943379e-18)
```

```
[ ] print("P value = 0 --> GOT DIFFERENCE between Low vs Medium Horsepower")
```

```
P value = 0 --> GOT DIFFERENCE between Low vs Medium Horsepower
```

## CONCLUSION

### A. NUMERICAL COLUMNS

| Columns | KEEP / DROP |
|---|---|
| Wheel Base | KEEP |
| Length | KEEP |
| Width | KEEP |
| Height | Drop |
| Curb Weight | KEEP |
| Engine Size | KEEP |
| Bore | KEEP |
| Stroke | Drop |
| Compression Ratio | Drop |
| Horsepower | KEEP |
| Peak RPM | Drop |
| City MPG | KEEP |
| Highway MPG | KEEP |
| City-L/100km | KEEP |
| Diesel | Drop |
| Gas | Drop |

### B. TEXTUAL COLUMNS

| Column | Keep / Drop |
|---|---|
| Make | KEEP |
| Aspiration | Drop |
| Number of Doors | Drop |
| Body Style | KEEP |
| Drive Wheels | KEEP |
| Engine Location | KEEP |
| Engine Type | KEEP |
| Number of Cylinders | KEEP |
| Fuel System | KEEP |
| Horsepower | KEEP |

## ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.