



# Data visualization

## Part II

# In the previous lecture

We learnt how to create basic plots using *matplotlib* library

- Scatter plot
- Histogram
- Bar plot

# In this lecture

We will learn how to create basic plots using *seaborn* library:

- Scatter plot
- Histogram
- Bar plot
- Box and whiskers plot
- Pairwise plots

# Seaborn

- Seaborn is a Python data visualization library based on matplotlib
- It provides a high-level interface for drawing attractive and informative statistical graphics

# Scatter plot

# Importing libraries

- Importing necessary libraries

```
import pandas as pd
```



‘pandas’ library to work with dataframes

```
import numpy as np
```



‘numpy’ library to do numerical operations

```
import matplotlib.pyplot as plt
```



‘matplotlib’ library to do visualization

```
import seaborn as sns
```



‘seaborn’ library to do visualization

# Importing data into Spyder

- Importing data

```
cars_data = pd.read_csv('Toyota.csv', index_col=0,
                        na_values=["??", "????"])
```

Variable explorer

Name	Type	Size
cars_data	DataFrame	(1436, 10)

- Removing missing values from the dataframe

```
cars_data.dropna(axis = 0, inplace=True)
```

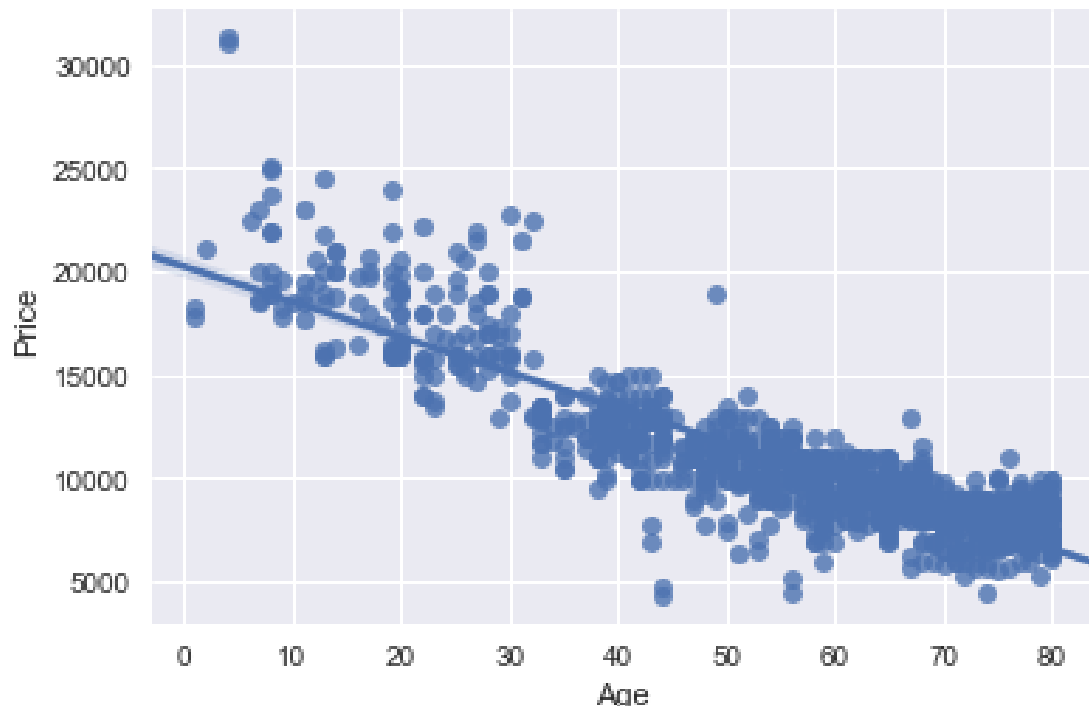
Variable explorer

Name	Type	Size
cars_data	DataFrame	(1096, 10)

# Scatter plot

- Scatter plot of *Price vs Age* with default arguments

```
sns.set(style="darkgrid")
sns.regplot(x=cars_data['Age'], y=cars_data['Price'])
```



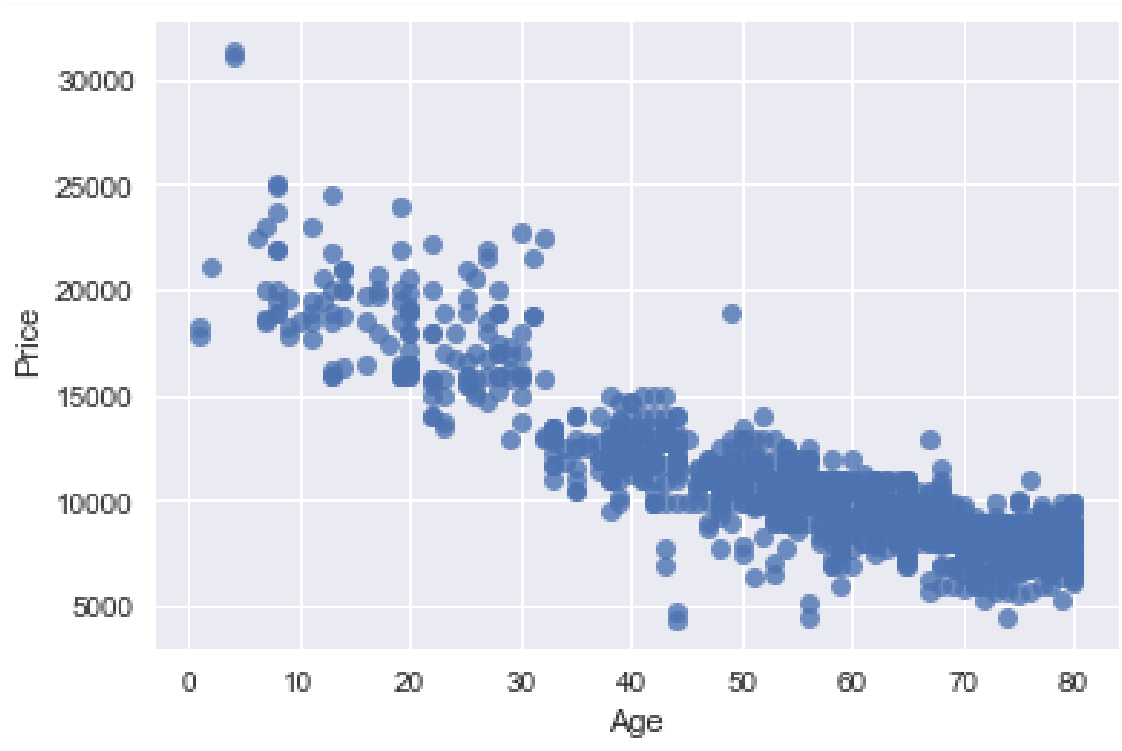
- By default, `fit_reg = True`
- It estimates and plots a regression model relating the x and y variables



# Scatter plot

- Scatter plot of *Price vs Age* without the regression fit line

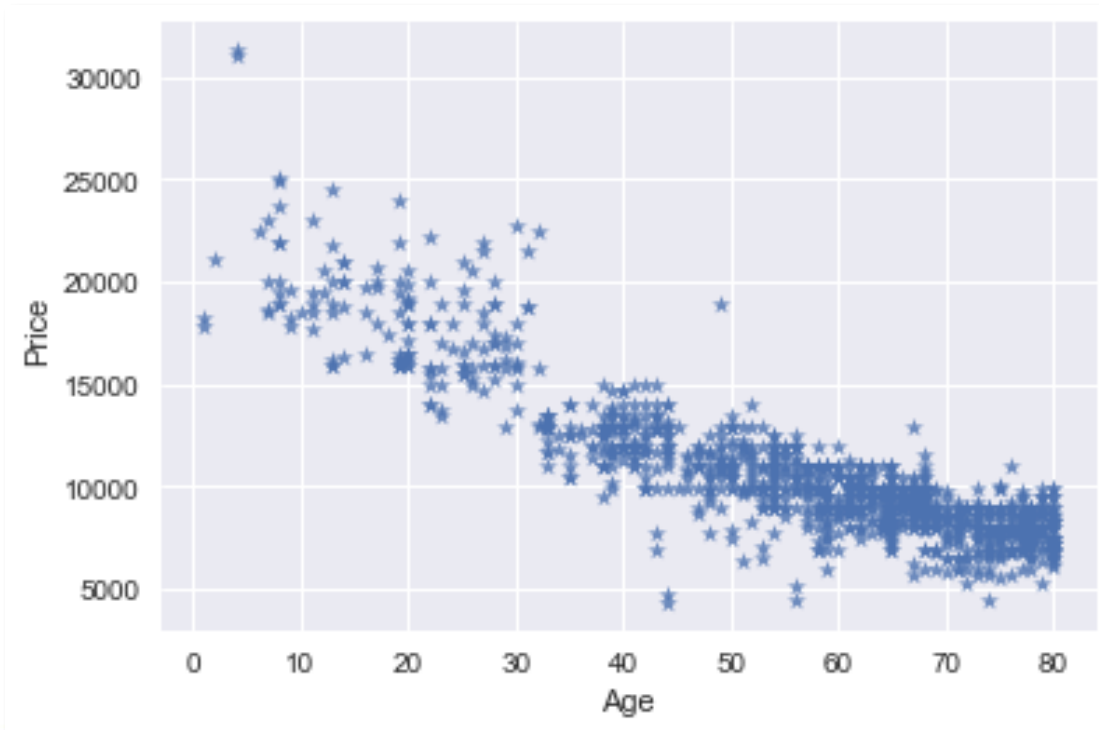
```
sns.regplot(x=cars_data['Age'], y=cars_data['Price'],  
            fit_reg=False)
```



# Scatter plot

- Scatter plot of *Price vs Age* by customizing the appearance of markers

```
sns.regplot(x=cars_data['Age'], y=cars_data['Price'],  
            marker="*", fit_reg=False)
```



# Scatter plot

- Scatter plot of *Price vs Age* by *FuelType*
- Using **hue** parameter, including another variable to show the fuel types categories with different colors

```
sns.lmplot(x='Age', y='Price', data=cars_data,  
           fit_reg=False, hue='FuelType',  
           legend=True, palette="Set1")
```

# Scatter plot

- Scatter plot of *Price vs Age by FuelType*



Similarly, custom the appearance of the markers using

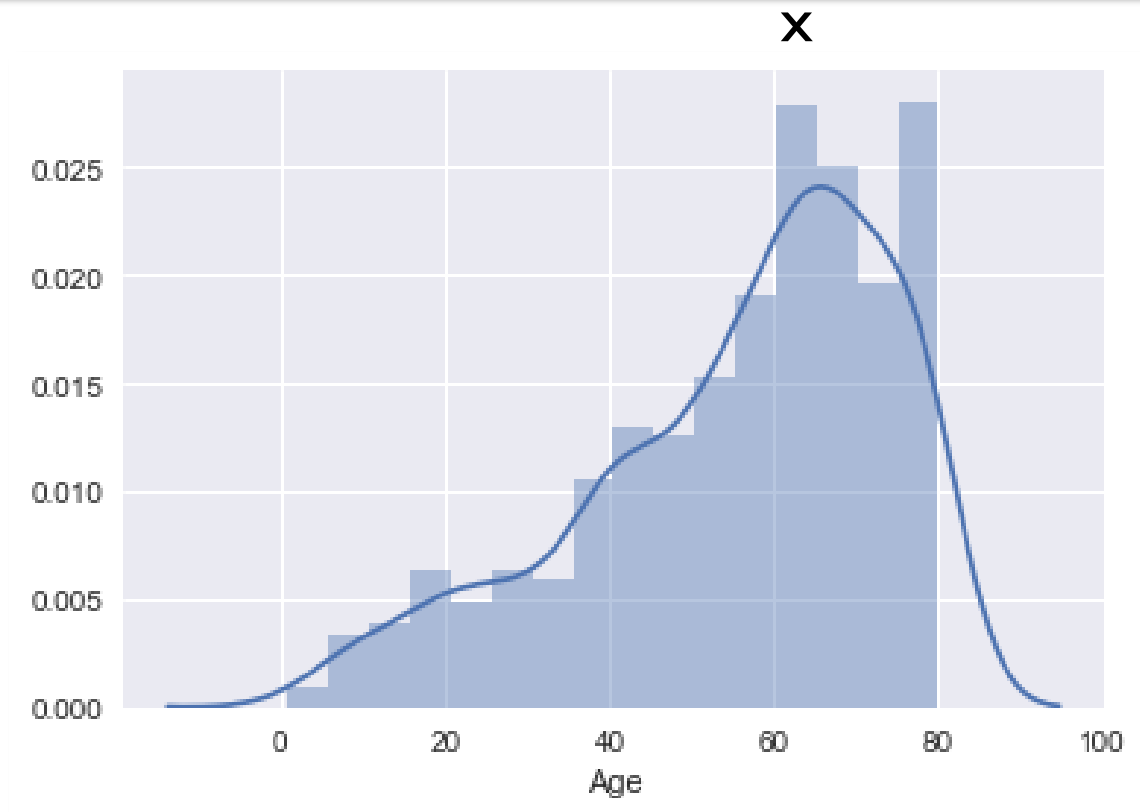
- transparency
- shape
- size

# Histogram

# Histogram

- Histogram with default kernel density estimate

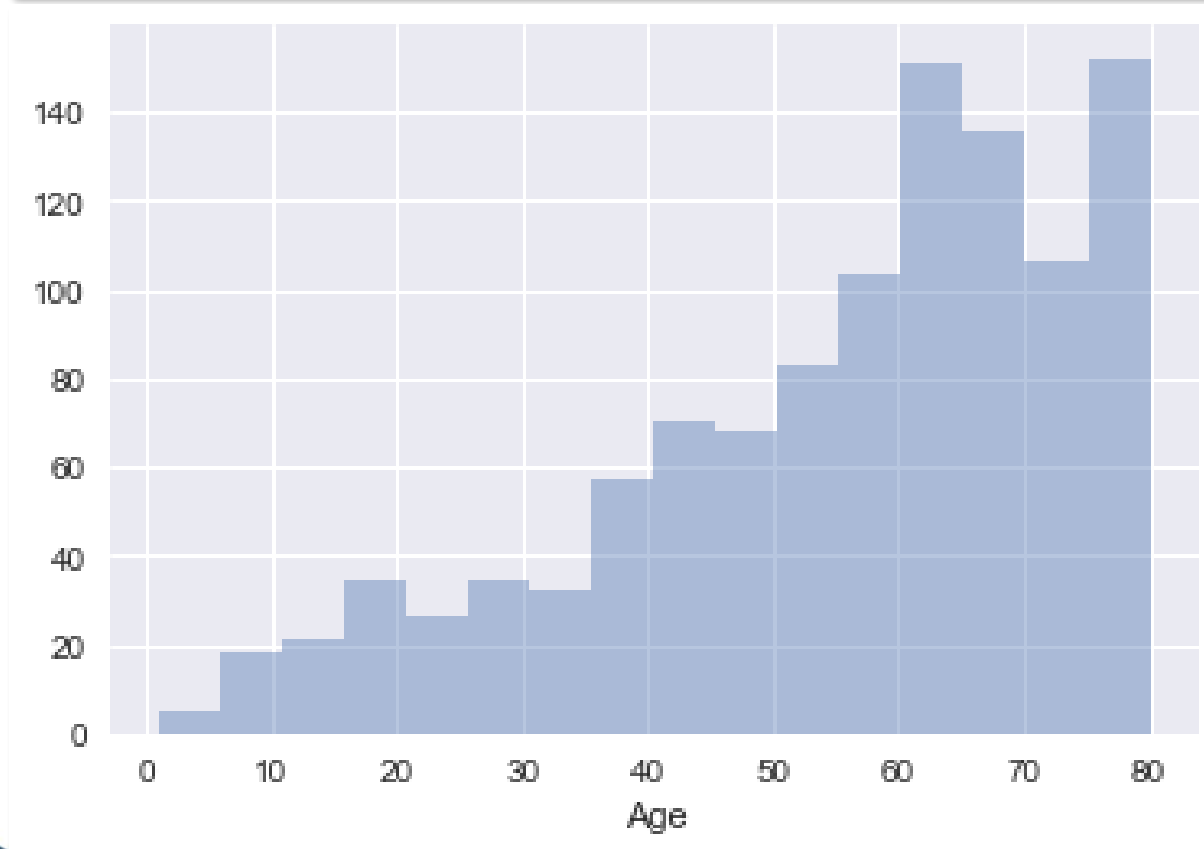
```
sns.distplot(cars_data['Age'] )
```



# Histogram

- Histogram without kernel density estimate

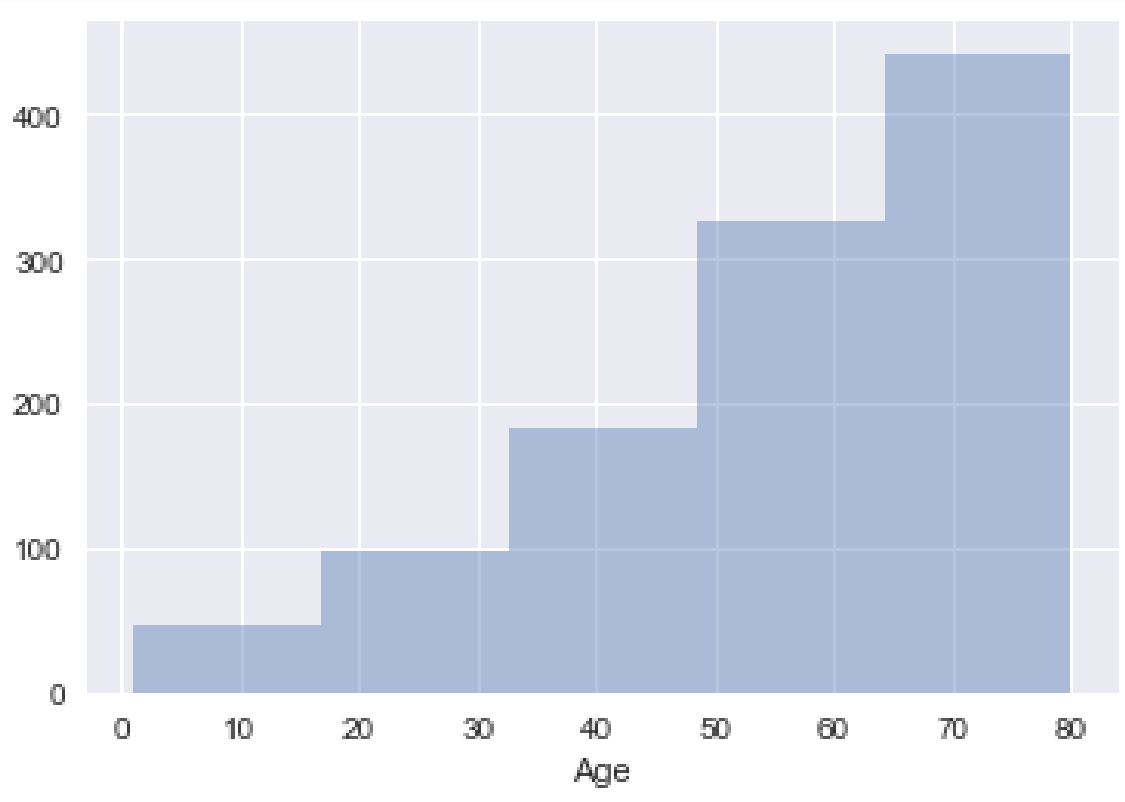
```
sns.distplot(cars_data[ 'Age' ], kde=False)
```



# Histogram

- Histogram with fixed no. of bins

```
sns.distplot(cars_data['Age'], kde = False, bins=5 )
```



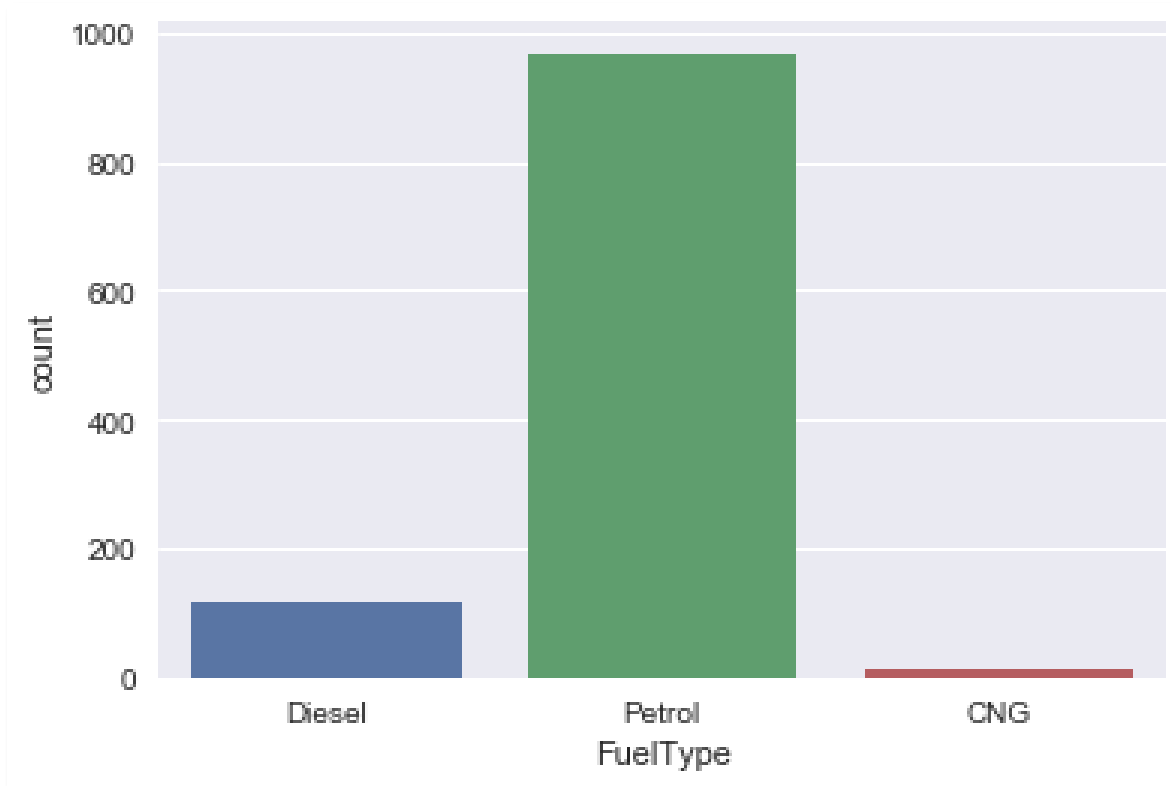


# Bar plot

# Bar plot

- Frequency distribution of fuel type of the cars

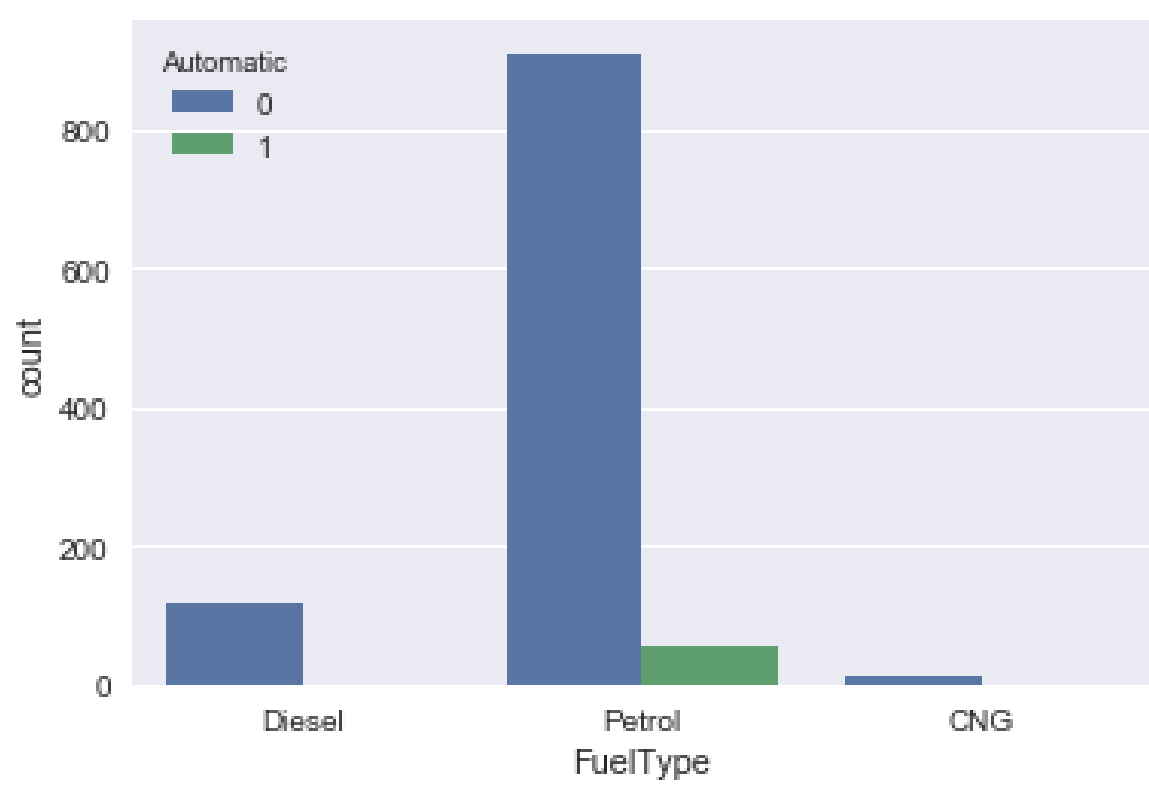
```
sns.countplot(x="FuelType", data=cars_data)
```



# Grouped bar plot

- Grouped bar plot of *FuelType* and *Automatic*

```
sns.countplot(x="FuelType", data=cars_data, hue = "Automatic")
```



```
pd.crosstab(index = cars_data['Automatic'],
             columns = cars_data2['FuelType'],
             dropna = True)
```

Out[5]:

FuelType	CNG	Diesel	Petrol
Automatic			
0	15	144	1104
1	0	0	73

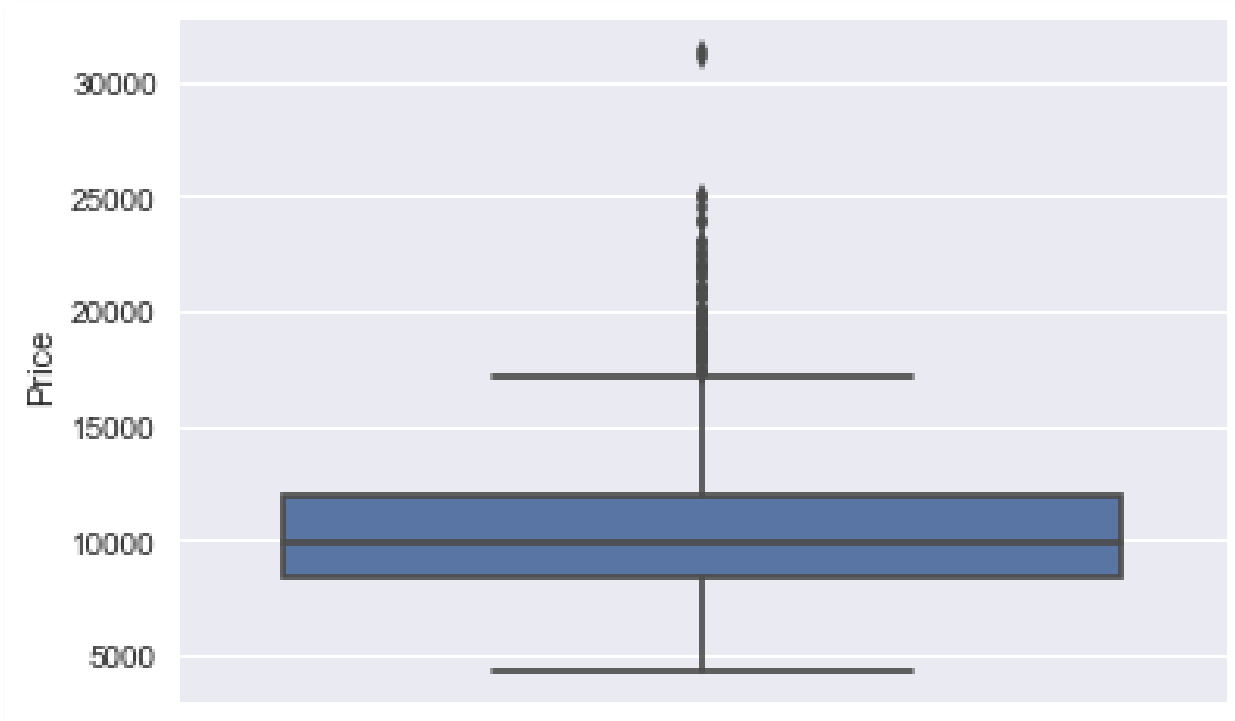
# Box and whiskers plot

---

# Box and whiskers plot – numerical variable

- Box and whiskers plot of *Price* to visually interpret the five-number summary

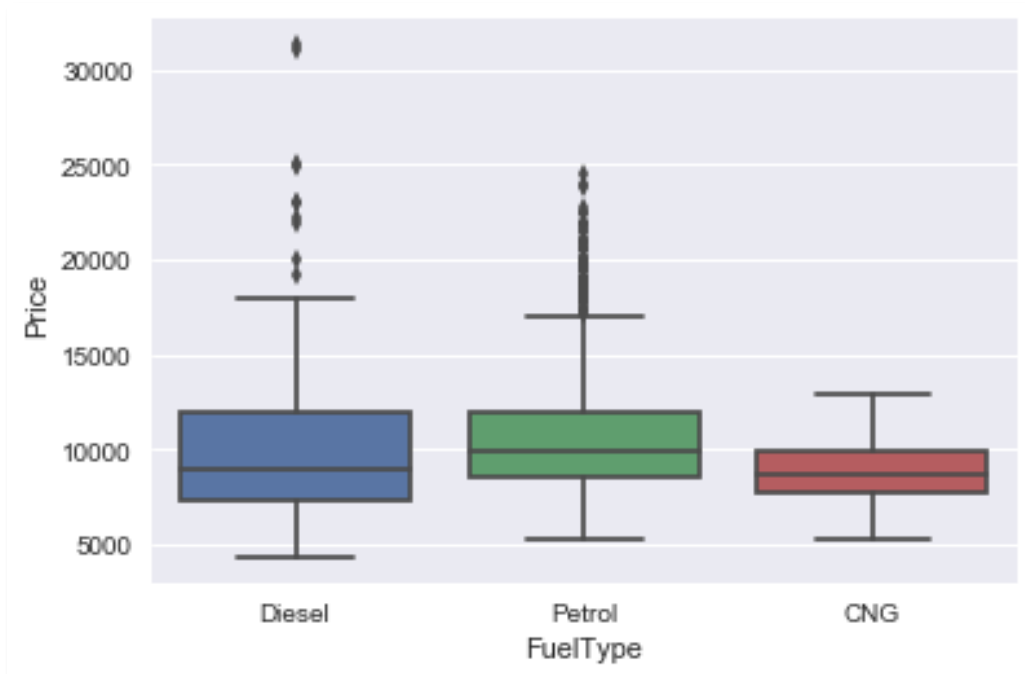
```
sns.boxplot(y=cars_data["Price"] )
```



# Box and whiskers plot

- Box and whiskers plot for numerical vs categorical variable
- Price of the cars for various fuel types

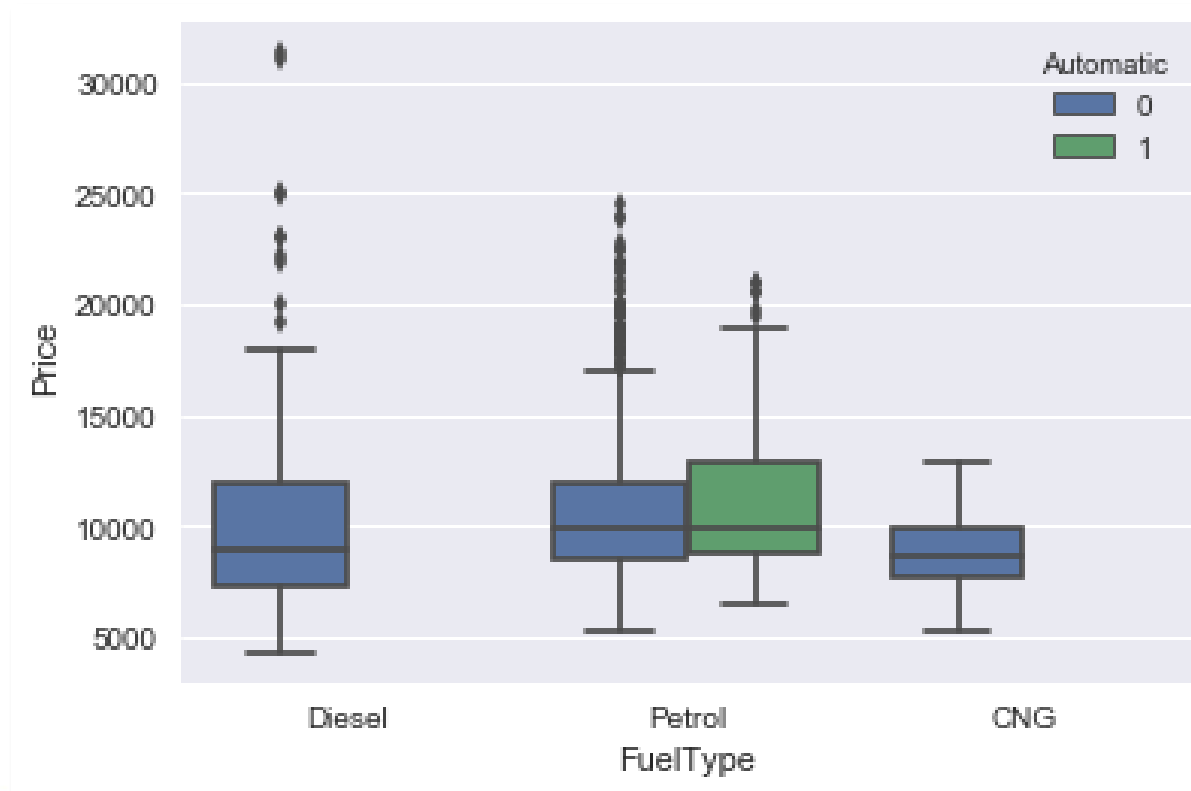
```
sns.boxplot(x = cars_data['FuelType'], y = cars_data["Price"])
```



# Grouped box and whiskers plot

- Grouped box and whiskers plot of *Price* vs *FuelType* and *Automatic*

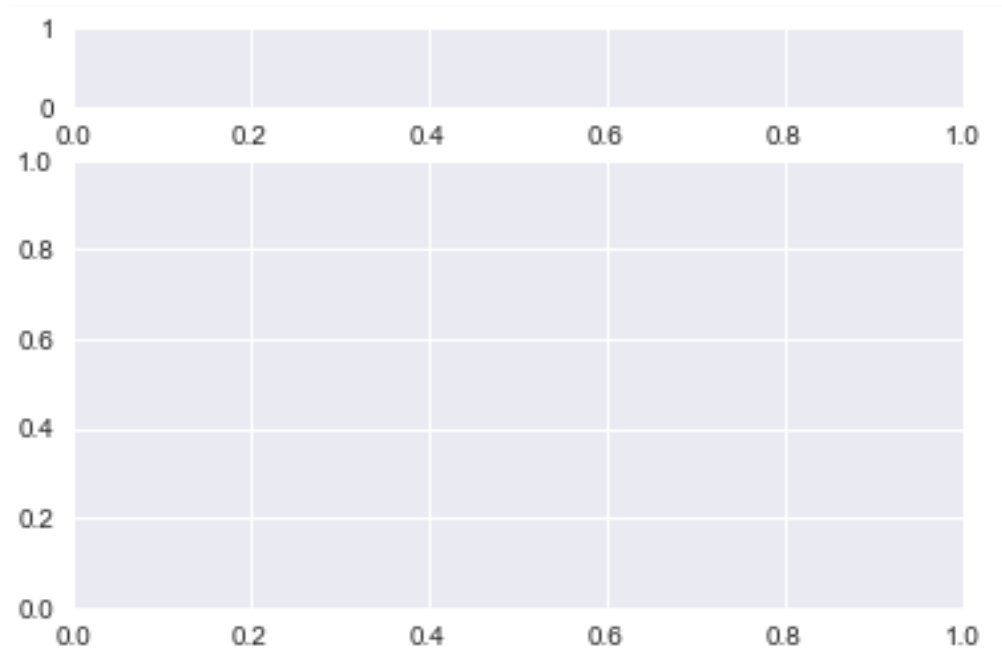
```
sns.boxplot(x = "FuelType", y = cars_data["Price"],  
            hue = "Automatic", data = cars_data)
```



# Box-whiskers plot and Histogram

- Let's plot box-whiskers plot and histogram on the same window
- Split the plotting window into 2 parts

```
f,(ax_box, ax_hist)=plt.subplots(2, gridspec_kw={"height_ratios": (.15, .85)})
```



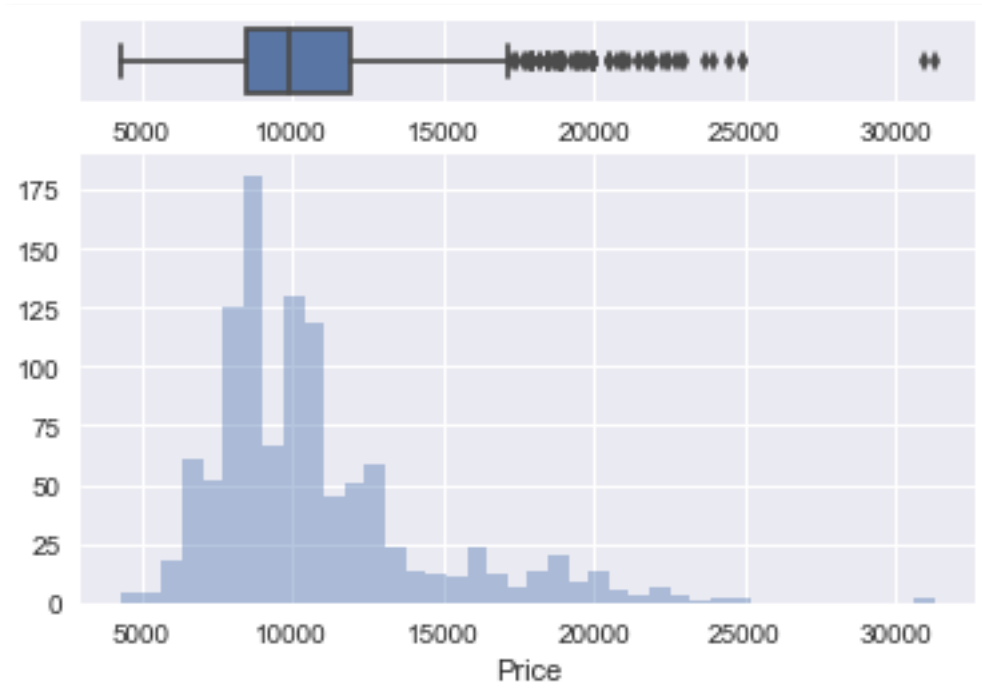


# Box-whiskers plot and Histogram

- Now, add create two plots

```
sns.boxplot(cars_data["Price"] , ax=ax_box)
```

```
sns.distplot(cars_data["Price"], ax=ax_hist, kde = False)
```



# Pairwise plots

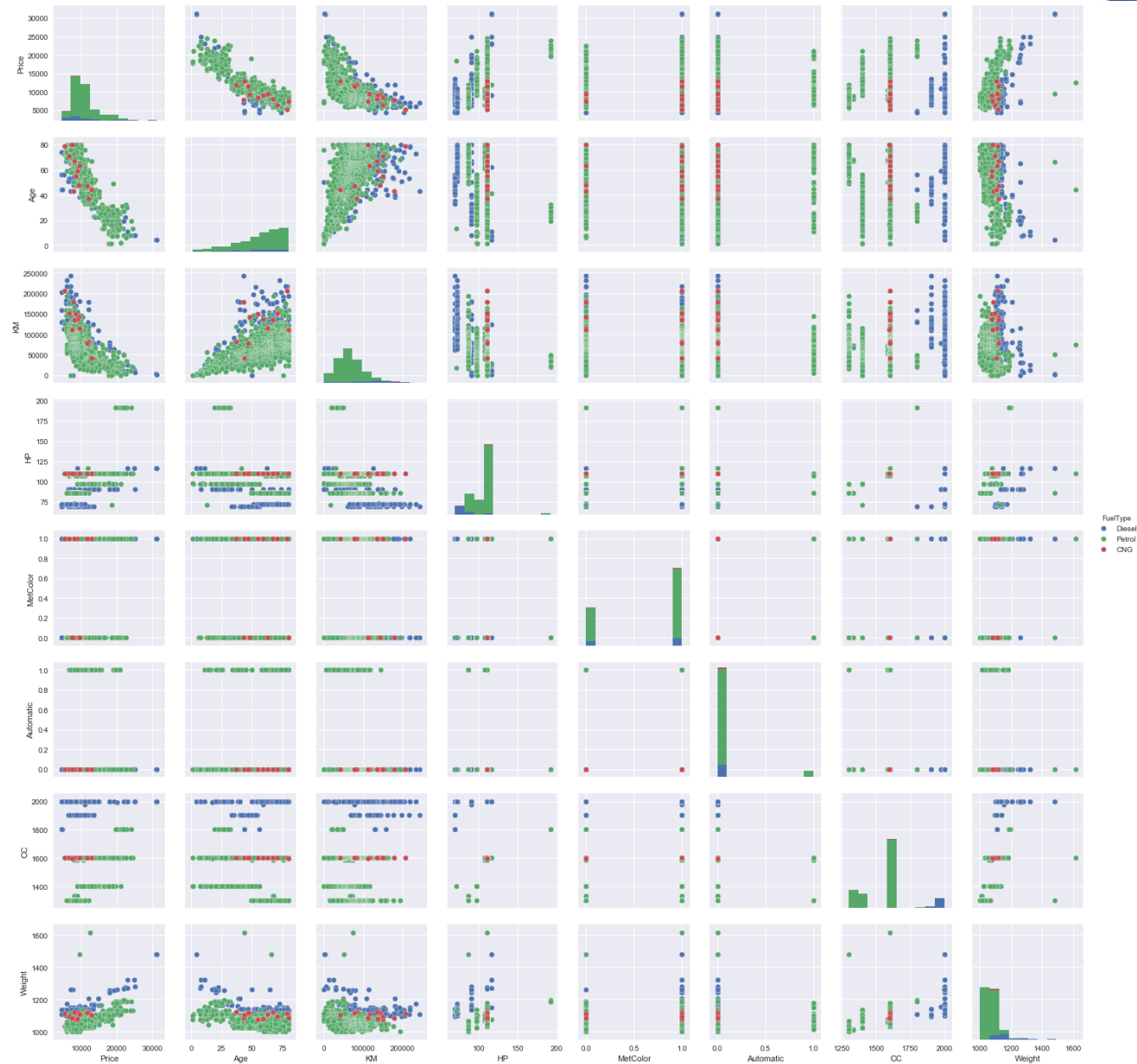
- It is used to plot pairwise relationships in a dataset
- Creates scatterplots for joint relationships and histograms for univariate distributions

## Code:

```
sns.pairplot(cars_data, kind="scatter", hue="FuelType")  
plt.show()
```

# Pairwise plots

Output:



# Summary

We have learnt how to create basic plots using *seaborn* library:

- Scatter plot
- Histogram
- Bar plot
  - Grouped bar plot
- Box and whiskers plot
  - Grouped box and whiskers plot
- Pairwise plots

```
operation == "MIRROR_X":  
    mirror_mod.use_x = True  
    mirror_mod.use_y = False  
    mirror_mod.use_z = False  
operation == "MIRROR_Y":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = True  
    mirror_mod.use_z = False  
operation == "MIRROR_Z":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = False  
    mirror_mod.use_z = True
```

```
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
= ("Selected" + str(modifier_ob.name))  
mirror_ob.select = 0  
= bpy.context.selected_objects  
data.objects[one.name].select  
= 1  
print("please select exactly one mirror")
```

WILLIAM C. LEE

```
def mirror(modifier):  
    #add mirror to the selected  
    #object -mirror_x, mirror_y,  
    #mirror_z  
    mirror_ob = bpy.context.selected_objects[0]  
    mirror_mod = modifier
```

THANK YOU