Data Visulaization

- Data Visualization allows us to quickly interpret the data and adjust different variables to see their effect.
- Interpreting the data through numbers is quiet difficult, but interpreting them through graphs makes easier to understand.
- Several visualization tools are making it possible.

Why Visualize the data?

- Observe the patterns
- Identify extreme values that could be anomalies
- Easy Interpretation

Popular Plotting Libraries of Python

- ***matplotlib:*** To create 2D plots and graphs
- ***pandas visualization:*** Easy to interface, built on Matplotlib
- ***seaborn:*** provides a high-level interface for drawing attractive and informative statistical graphics
- ***ggplot:*** based on R's ggplot2, uses Grammar of Graphics
- ***plotly:*** can create interactive plots

matplotlib Library

matplotlib Library

- Matplotlib is a 2D plotting library which produces good quality figures
- Although it has its origins in emulating the MATLAB graphics commands, it is independent of MATLAB
- It makes heavy use of NumPy and other extension code to provide good performance even for large arrays

seaborn Library

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

Univariate Plots

- Plots using single variable/attribute

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
cars_data = pd.read_csv('UsedCarsPrice.csv',index_col=0,na_values=["??", "????"])
cars_data.head(10)
```

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	13500	23.0	46986.0	Diesel	90.0	1.0	0	2000	three	1165
1	13750	23.0	72937.0	Diesel	90.0	1.0	0	2000	3	1165
2	13950	24.0	41711.0	Diesel	90.0	NaN	0	2000	3	1165
3	14950	26.0	48000.0	Diesel	90.0	0.0	0	2000	3	1165
4	13750	30.0	38500.0	Diesel	90.0	0.0	0	2000	3	1170
5	12950	32.0	61000.0	Diesel	90.0	0.0	0	2000	3	1170
6	16900	27.0	NaN	Diesel	NaN	NaN	0	2000	3	1245
7	18600	30.0	75889.0	NaN	90.0	1.0	0	2000	3	1245
8	21500	27.0	19700.0	Petrol	192.0	0.0	0	1800	3	1185
9	12950	23.0	71138.0	Diesel	NaN	NaN	0	1900	3	1105

```
cars_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436 entries, 0 to 1435
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Price       1436 non-null   int64
```

dtypes: float64(4), int64(4), object(2)
memory usage: 123.4+ KB

```
[4]: cars_data.shape
```

```
[4]: (1436, 10)
```

```
[5]: cars_data.describe()
```

[5]:

	Price	Age	KM	HP	MetColor	Automatic	CC	Weight
count	1436.000000	1336.000000	1421.000000	1430.000000	1286.000000	1436.000000	1436.000000	1436.00000
mean	10730.824513	55.672156	68647.239972	101.478322	0.674961	0.055710	1566.827994	1072.45961
std	3626.964585	18.589804	37333.023589	14.768255	0.468572	0.229441	187.182436	52.64112
min	4350.000000	1.000000	1.000000	69.000000	0.000000	0.000000	1300.000000	1000.00000
25%	8450.000000	43.000000	43210.000000	90.000000	0.000000	0.000000	1400.000000	1040.00000
50%	9900.000000	60.000000	63634.000000	110.000000	1.000000	0.000000	1600.000000	1070.00000
75%	11950.000000	70.000000	87000.000000	110.000000	1.000000	0.000000	1600.000000	1085.00000
max	32500.000000	80.000000	243000.000000	192.000000	1.000000	1.000000	2000.000000	1615.00000

```
[6]: cars_data.describe(include='all')
```

[6]:

	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	
count	1436.000000	1336.000000	1421.000000	1336	1430.000000	1286.000000	1436.000000	1436.000000	1436	14

[7]:
```
cars_data.isna().sum()
```

[7]:
```
Price          0
Age          100
KM            15
FuelType     100
HP             6
MetColor     150
Automatic      0
CC             0
Doors          0
Weight         0
dtype: int64
```

Drop Missing Values

[8]:
```
cars_data.dropna(axis=0, inplace=True)
cars_data.shape
```

[8]:
```
(1096, 10)
```

Univariate Plots

- Considering one attribute for plotting the graph
- Examples: Barplot, Histogram, Piechart, donut chart etc.

Bivariate Plots

- Considering two attributes at a time - one on x-axis and the other on y-axis
- Examples: Scatter plot, pairplot, heatmap etc.

Bar Plot

- A bar plot is a plot that presents categorical data with rectangular bars with lengths proportional to the counts that they represent
- Bar plot is used when we need to represent the frequency distribution of categorical variables
- A bar diagram makes it easy to compare sets of data between different groups

9]:
```
S=cars_data['FuelType'].value_counts()
S
```

9]:
```
Petrol    968
Diesel    116
CNG        12
Name: FuelType, dtype: int64
```

0]:
```
petrol_count= S[0]
diesel_count = S[1]
cng_count = S[2]
```

jupyter Day6_Stats_DV Last Checkpoint: last month

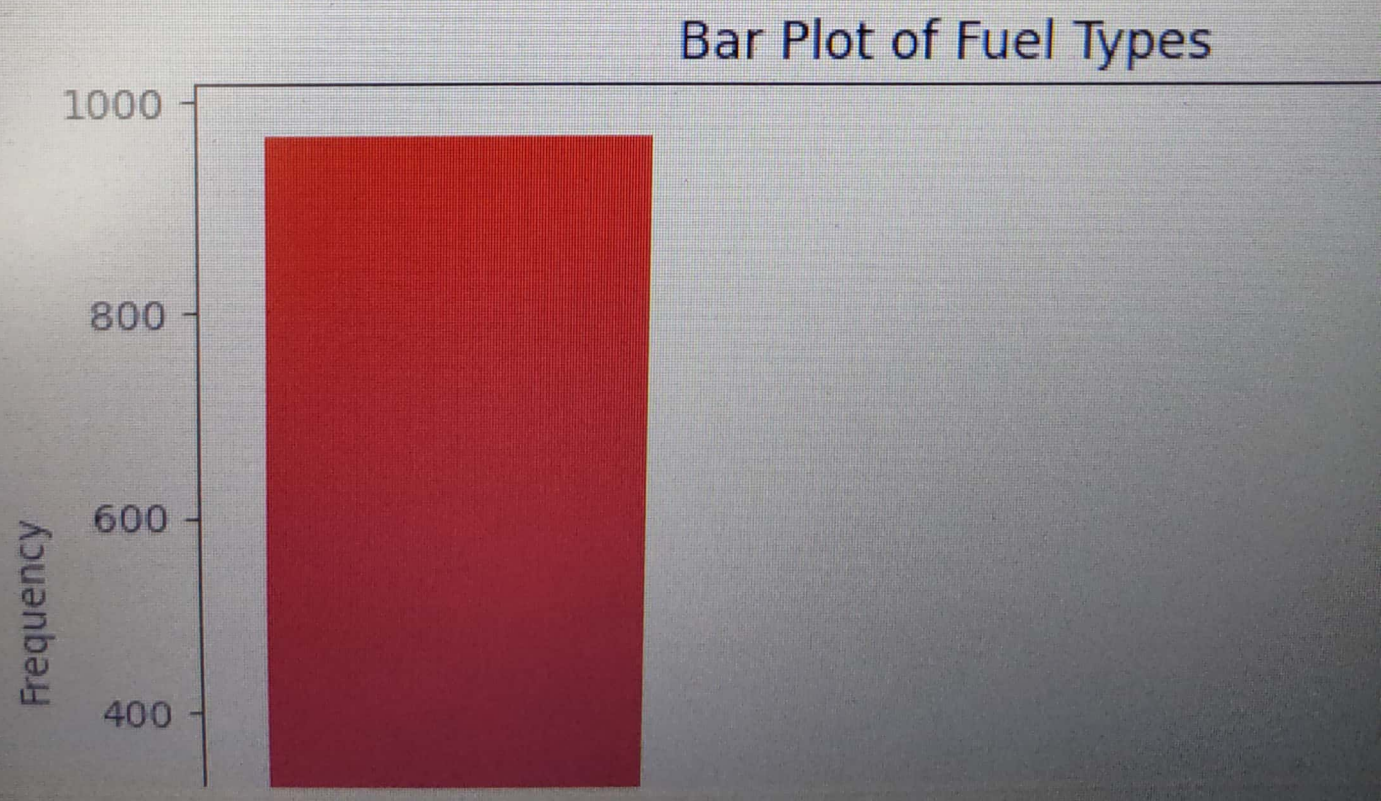
Edit View Run Kernel Settings Help

Markdown

[11]:
```
counts= [petrol_count, diesel_count,cng_count]
fuelType = ('Petrol', 'Diesel', 'CNG')
index=np.arange(len(fuelType))
index
```

[11]:
```
array([0, 1, 2])
```

[12]:
```
plt.bar(index, counts, color=['red', 'blue', 'orange'])
plt.title('Bar Plot of Fuel Types')
plt.xlabel('Fuel Types')
plt.ylabel('Frequency')
plt.xticks(index, fuelType,rotation=45)
plt.figure(figsize=(6,4))
plt.show()
```



Search

localhost:8888/notebooks/Day6_Stats_DV.ipynb

jupyter Day6_Stats_DV Last Checkpoint: last month

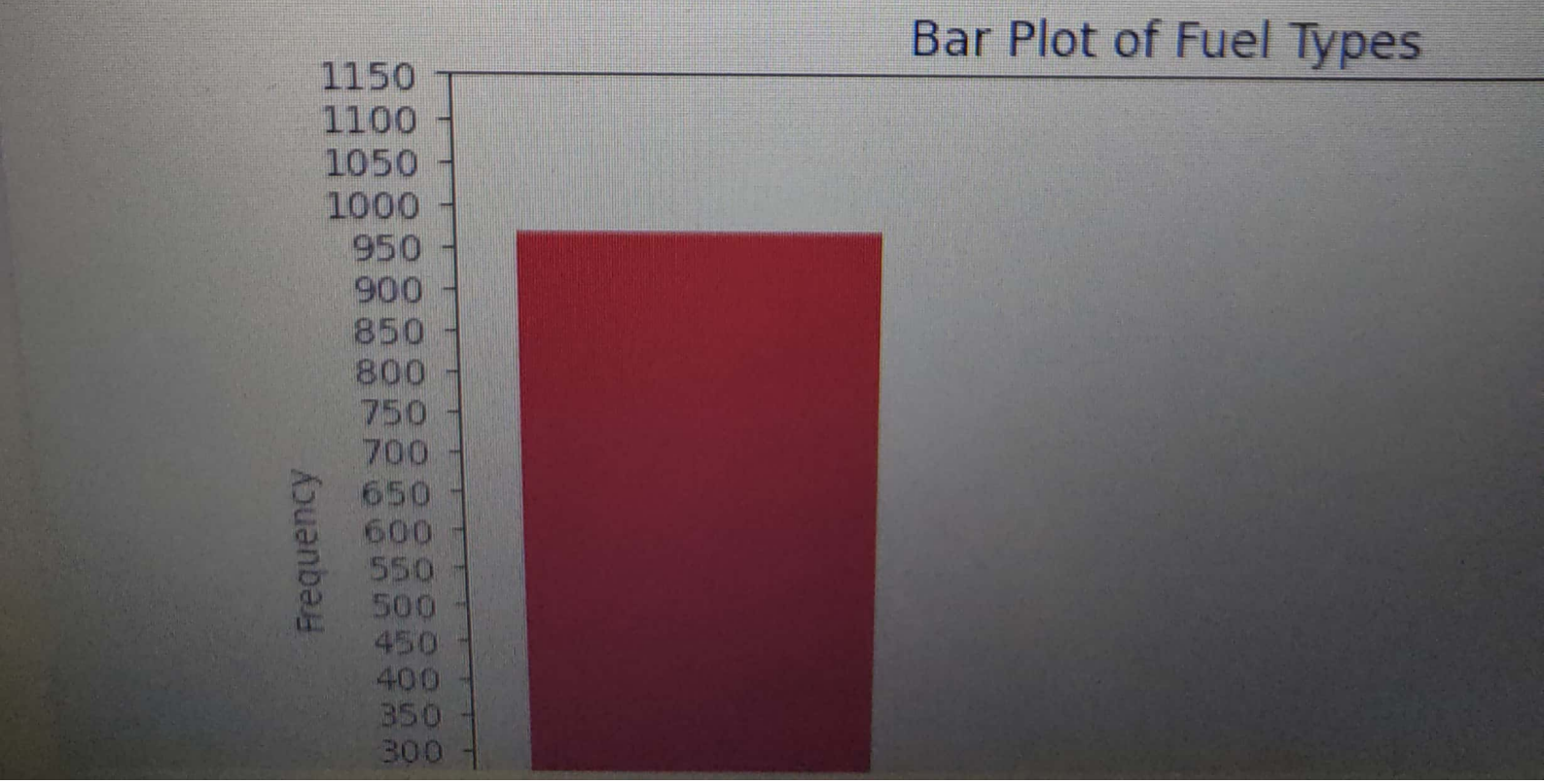
le Edit View Run Kernel Settings Help

Markdown

Fuel Types

```
<Figure size 600x400 with 0 Axes>
```

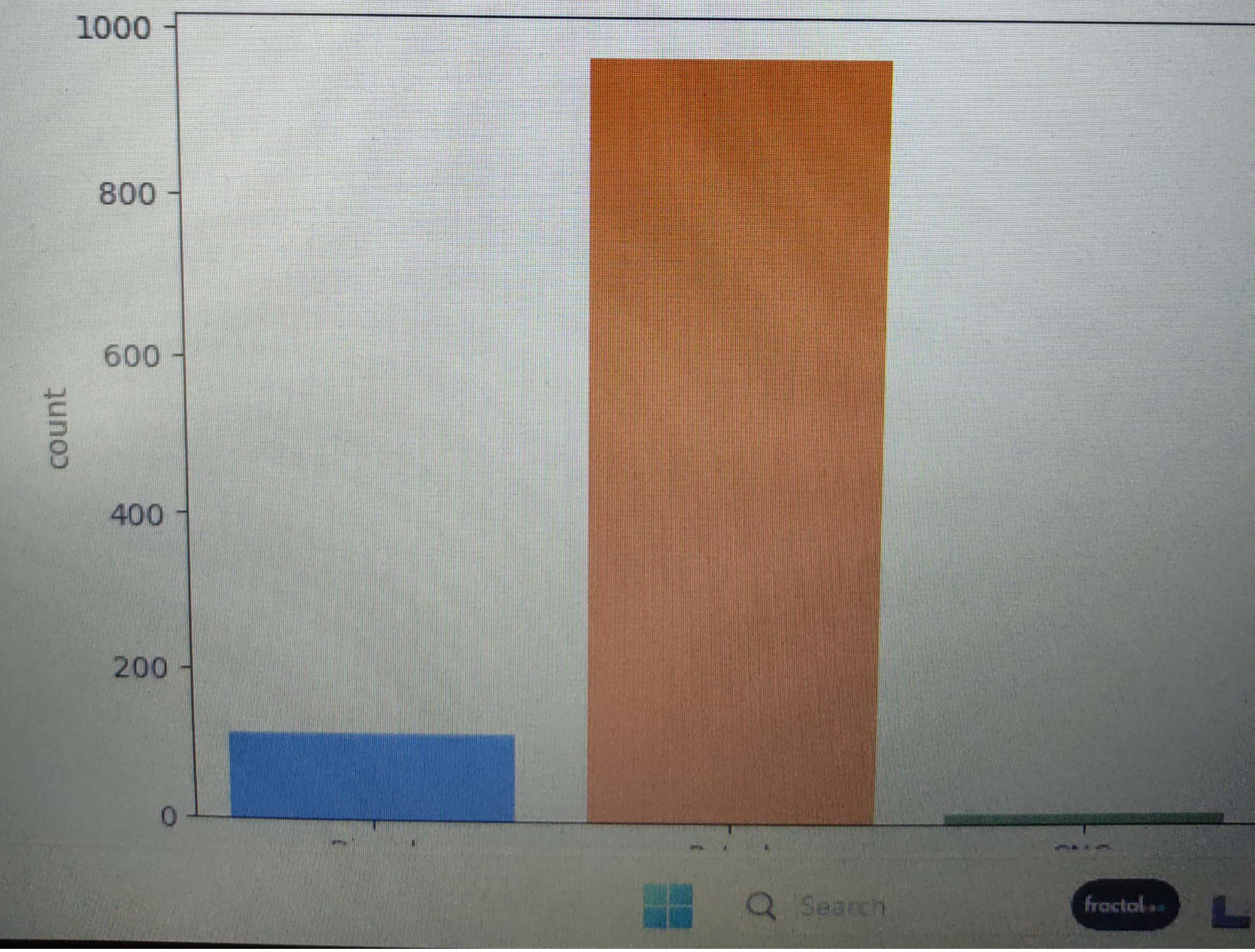
[13]:
```
#Changing range of y-axis
plt.bar(index, counts, color=['red', 'blue', 'orange'])
plt.title('Bar Plot of Fuel Types')
plt.xlabel('Fuel Types')
plt.ylabel('Frequency')
plt.xticks(index, fuelType,rotation=90)
plt.yticks(np.arange(0, 1200,50))
plt.show()
```



Bar Plot using seaborn library

```
# Frequency distribution of fuel type of the cars
sns.countplot(x='FuelType', data=cars_data)
```

```
<Axes: xlabel='FuelType', ylabel='count'>
```

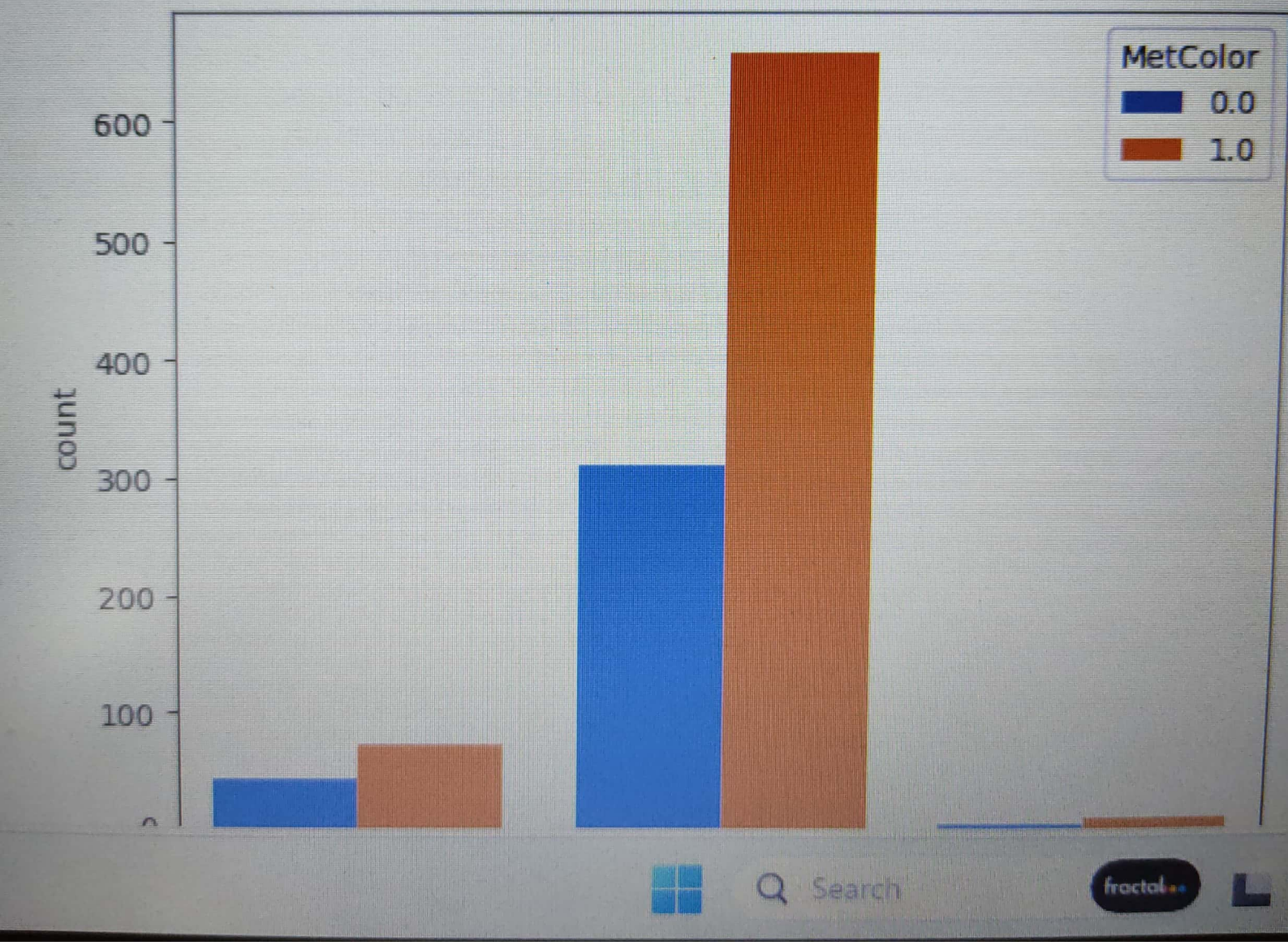


Grouped bar plot

- Grouped bar plot of FuelType and Automatic
- Visualization of crosstab

```
: sns.countplot(x='FuelType', data=cars_data, hue='MetColor')
```

```
: <Axes: xlabel='FuelType', ylabel='count'>
```

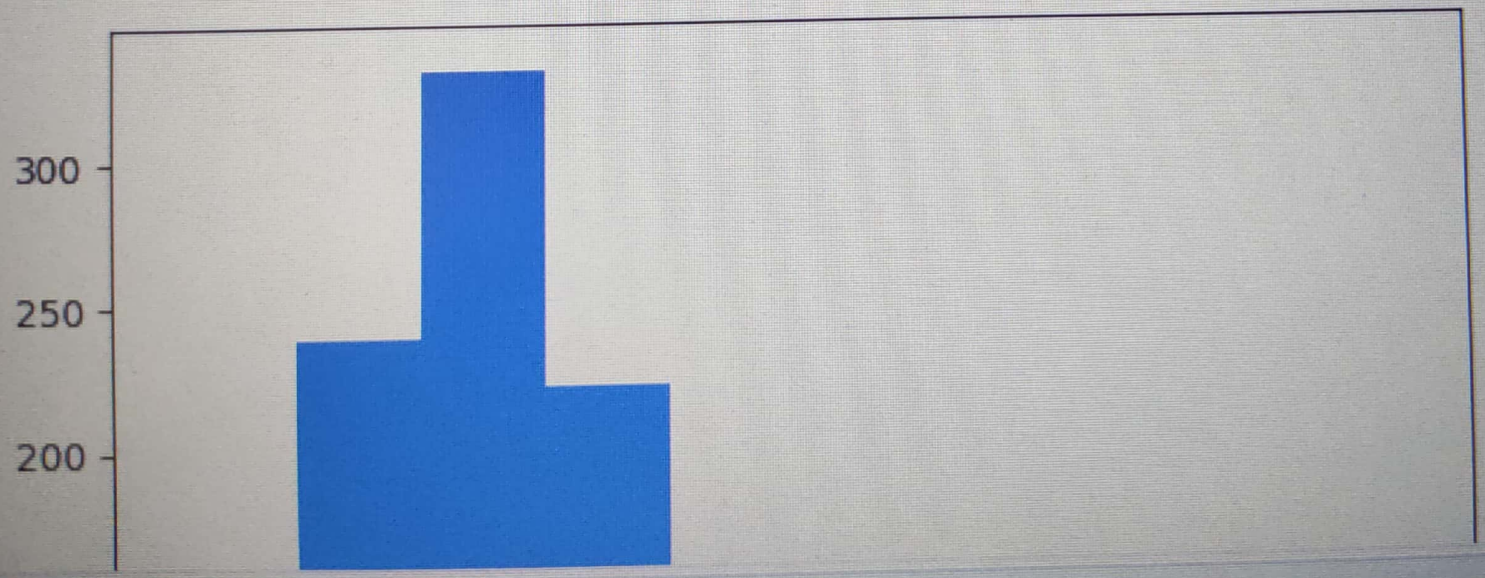


Histogram

- It is a graphical representation of data using bars of different heights
- Histogram groups numbers into ranges and the height of each bar depicts the frequency of each range or bin
- Histograms are used when we need to represent the frequency distribution of numerical variables

[16]:
```
#using matplotlib
plt.hist(cars_data['KM'])
```

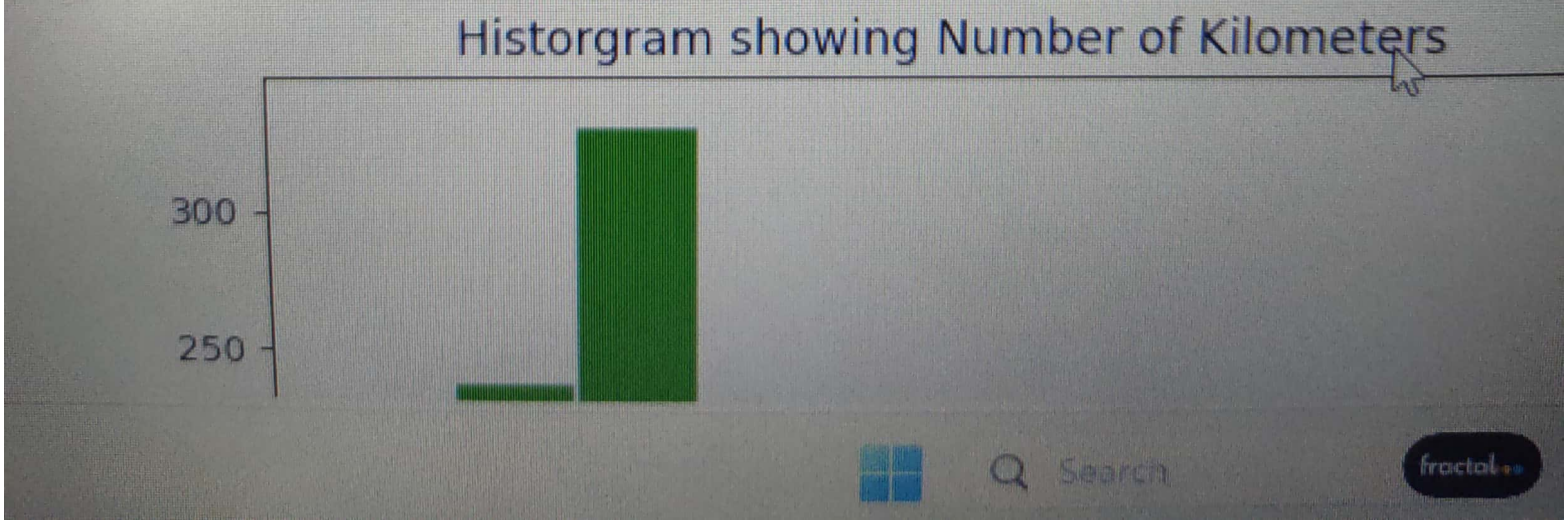
[16]:
```
(array([ 92., 239., 331., 222., 111.,  51.,  25.,  13.,  10.,   2.]),
 array([1.000000e+00, 2.430090e+04, 4.860080e+04, 7.290070e+04,
        9.720060e+04, 1.215005e+05, 1.458004e+05, 1.701003e+05,
        1.944002e+05, 2.187001e+05, 2.430000e+05]),
 <BarContainer object of 10 artists>)
```



```
cars_data['KM'].describe()
```

```
count      1096.000000
mean      69268.826642
std       38070.667467
min           1.000000
25%       43590.500000
50%       63393.500000
75%       88031.750000
max      243000.000000
Name: KM, dtype: float64
```

```
# Additional effects
plt.hist(cars_data['KM'], color = 'green', edgecolor='white', bins=10)
plt.title('Historgram showing Number of Kilometers')
plt.xlabel('Kilometer')
plt.ylabel('Frequency')
plt.show()
```



0 50000 100000 150000 200000 250000

Kilometer

Inference:

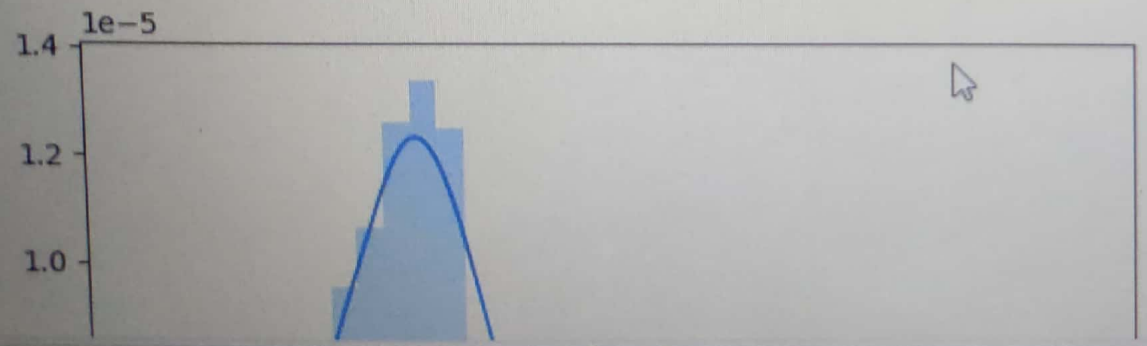
- Frequency distribution of kilometre of the cars shows that most of the cars have travelled between 25000 – 100000 km and there are only few cars with more distance travelled

Kernel Density Estimate

- Provides the curve showing distribution of the data
- Plot is based on probability ditribution
- Observe y-axis

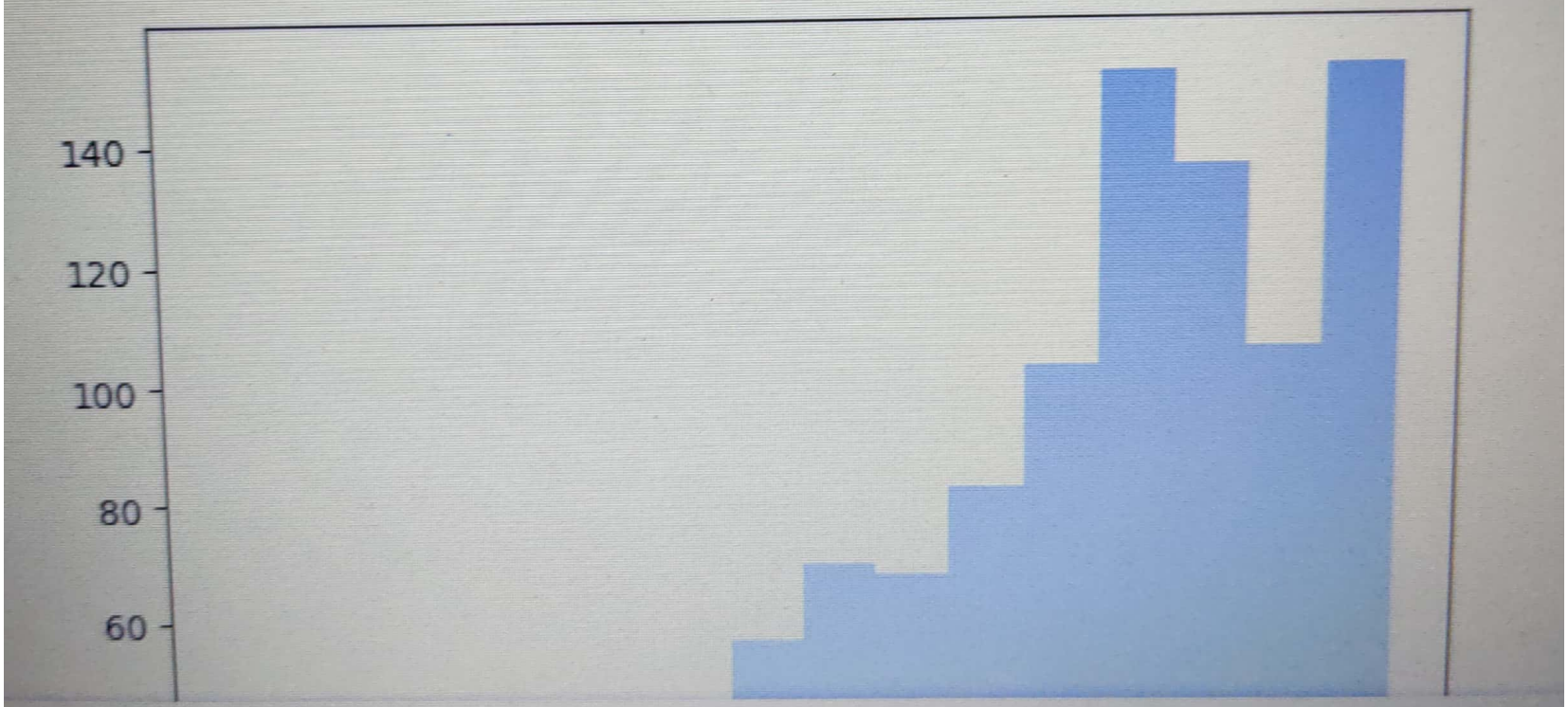
```
# Histogram with default kernel density estimate
sns.distplot(cars_data['KM'])
```

```
<Axes: xlabel='KM', ylabel='Density'>
```



```
# Histogram without  kernel density estimate
# Based on count or frequency
# Observe y-axis range
sns.distplot(cars_data['Age'], kde=False)
```

```
<Axes: xlabel='Age'>
```




```
#Histogram with fixed no. of bins
sns.distplot(cars_data['KM'], kde=True, bins=20)
```

```
<Axes: xlabel='KM', ylabel='Density'>
```

1e−5

localhost:8888/notebooks/Day6_Stats_DV.ipynb

upyter Day6_Stats_DV Last Checkpoint: last month

Edit View Run Kernel Settings Help

Markdown

Pie-Chart

```
[2]: df_FuelType = cars_data.groupby(['FuelType']).size().reset_index()
     # renaming the column as total cars
     df_FuelType = df_FuelType.rename(columns= {0 : "Total Cars"})
     df_FuelType.head()
```

[2]:

	FuelType	Total Cars
0	CNG	12
1	Diesel	116
2	Petrol	968

```
[3]: plt.figure(figsize=(8,6))
     plt.pie(df_FuelType['Total Cars'],labels=df_FuelType['FuelType'],  autopct='%.2f%%')
     plt.show()   # '%  %'
```

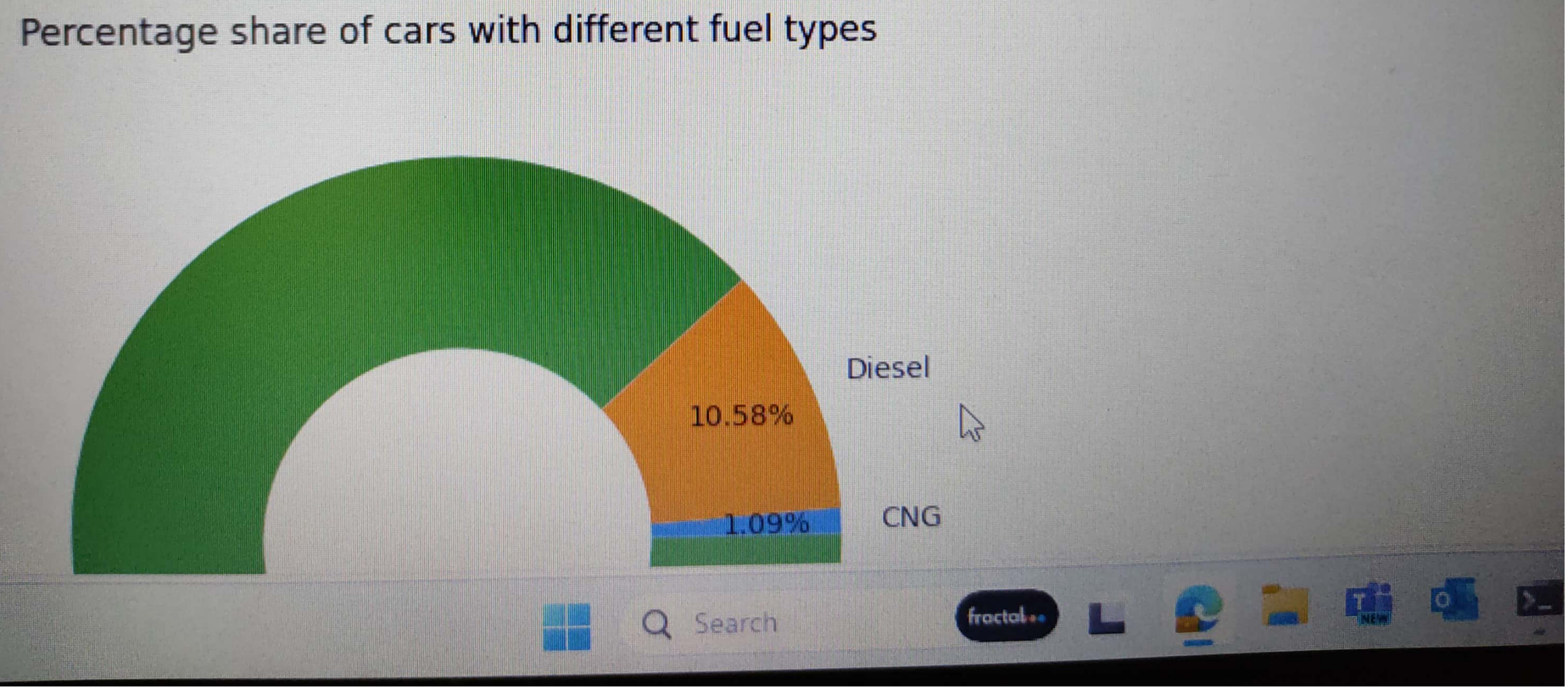
Search

DELL

Markdown

Donut Chart

[24]:
```
plt.figure(figsize=(6,8))
plt.pie(df_FuelType['Total Cars'], labels=df_FuelType['FuelType'], autopct='%.2f%%', pctdistance=.8)
# draw a circle centered at (0,0)
centre_circle = plt.Circle((0,0),0.50,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('Percentage share of cars with different fuel types')
plt.show()
```



Search

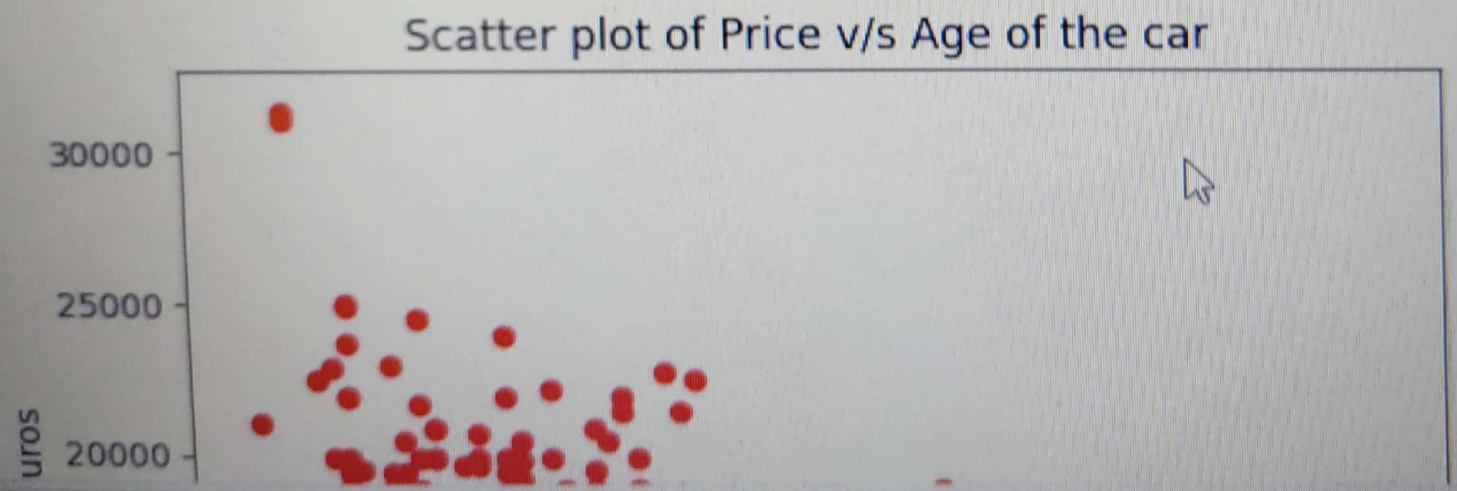
Bivariate Plots

- Plots which are used to understand the relationship between two variables

Scatter Plot

- A scatter plot is a set of points that represents the values obtained for two different variables plotted on a horizontal and vertical ax
- Scatter plots are used to convey the relationship between two numerical variables
- Scatter plots are sometimes called correlation plots because they show how two variables are correlated

[25]:
```
#plt.figure(figsize=(10,8))
plt.scatter(cars_data['Age'], cars_data['Price'], c='red')
plt.title("Scatter plot of Price v/s Age of the car")
plt.xlabel('Age in Months')
plt.ylabel('Price in Euros')
plt.show()
```



Markdown

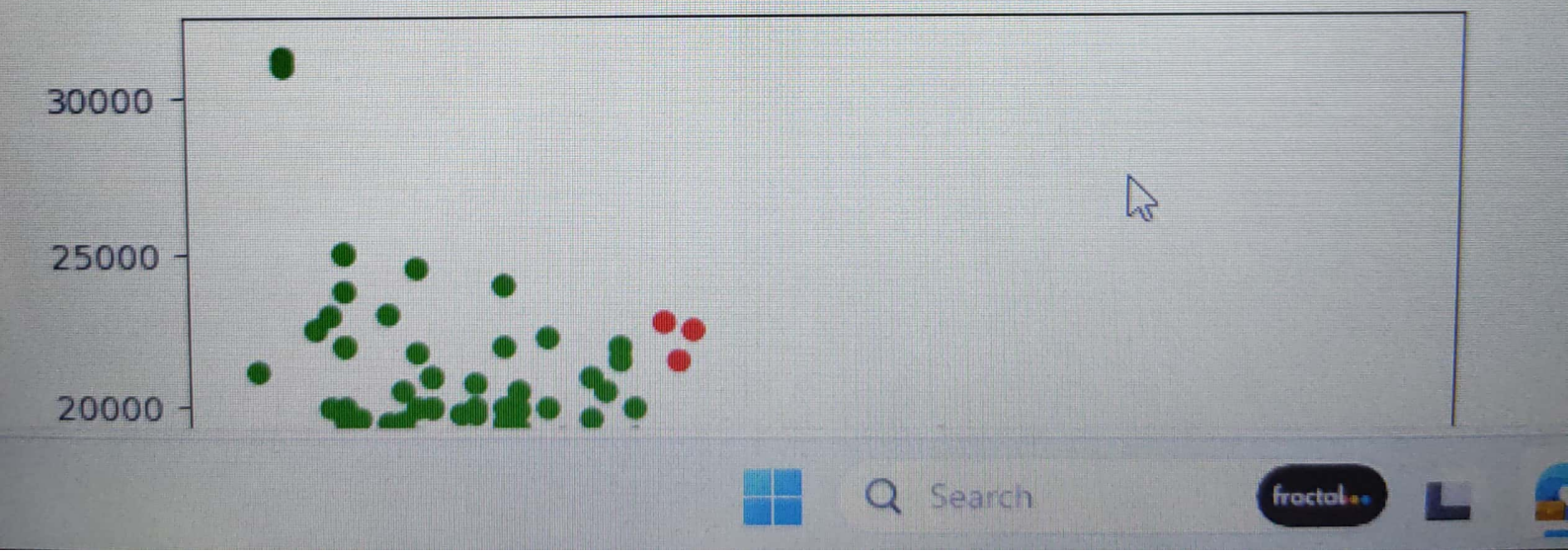
Inferences:

- Price of car and Age of car are linearly related.
- There is a high-degree of negative correlation
  - As the Age of car increases, the Price decreases.
- There are more old cars than the newer cars
  - Above 30 months, the graph is more dense
- There are few outliers
- Stronger negative correlation below the age 30

Conditional Scatter Plot

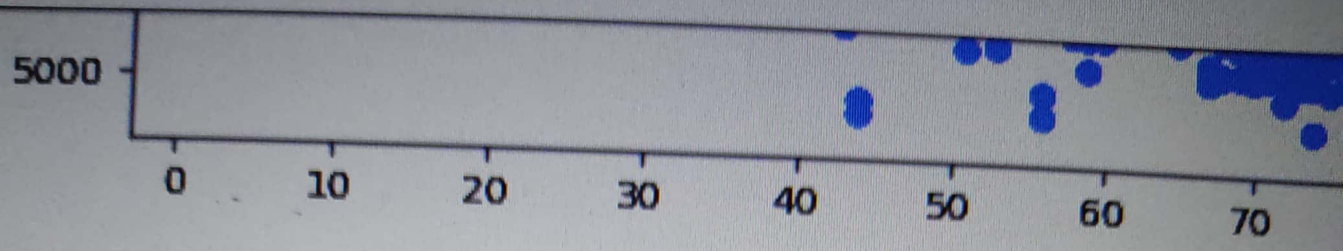
```
col = np.where(cars_data['Age']<30,'g',np.where(cars_data['Price']<15000,'b','r'))
plt.scatter(x= cars_data['Age'], y=cars_data['Price'], c=col)
```

```
<matplotlib.collections.PathCollection at 0x1fa01846310>
```



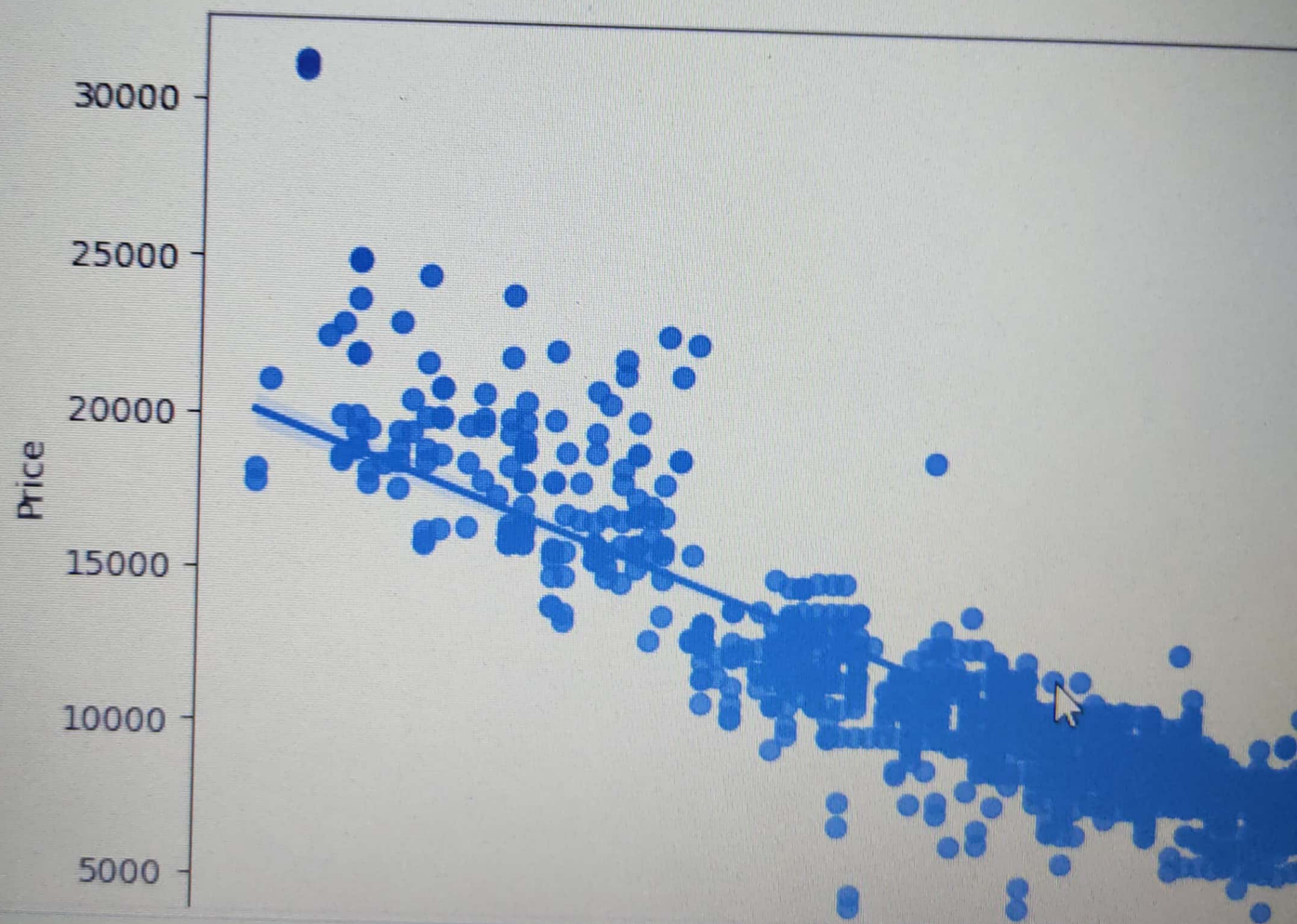
File Edit View Run Kernel Settings Help

Markdown



[27]:
```
#Using Seaborn Library
sns.regplot(x= cars_data['Age'], y=cars_data['Price'])
```

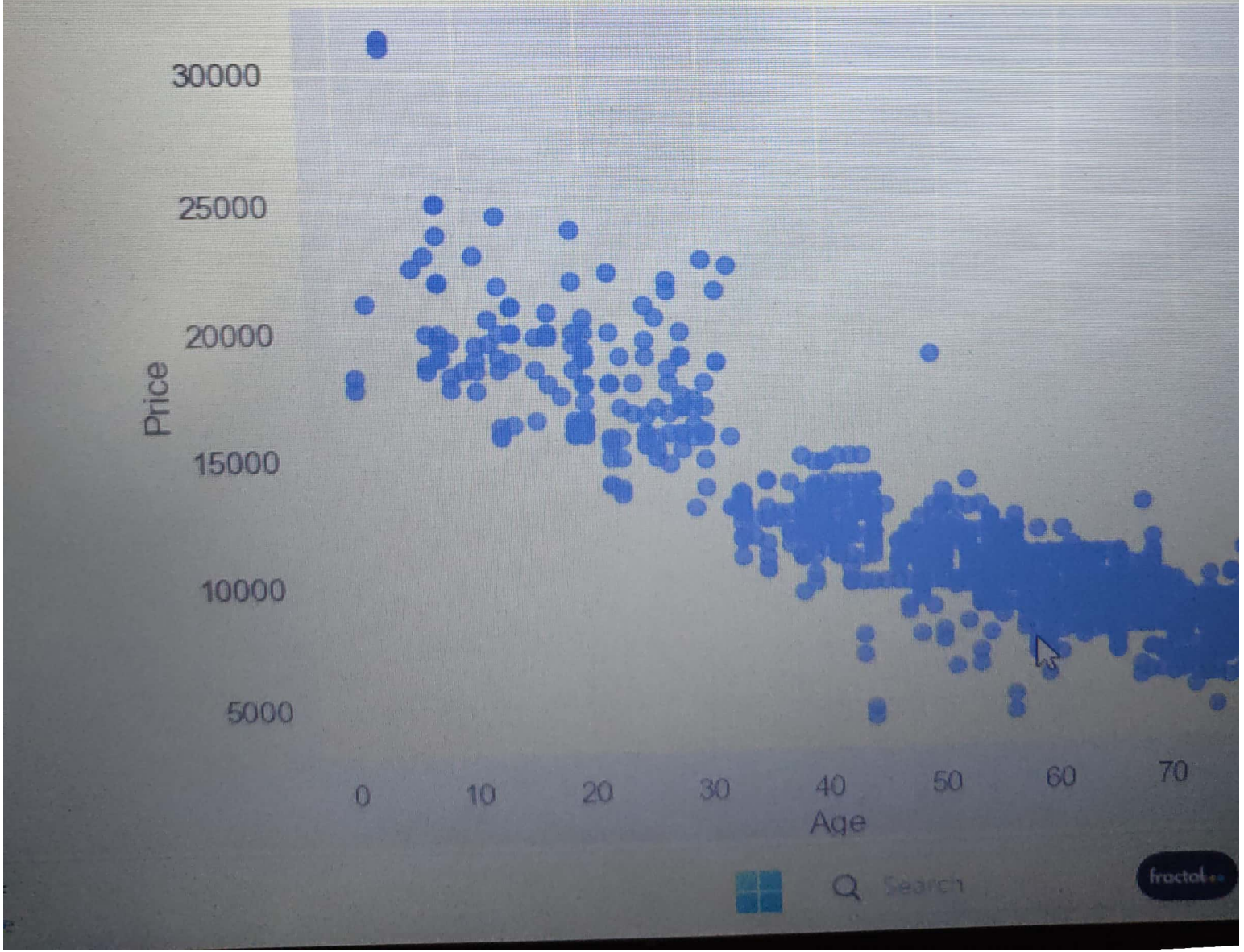
[27]: <Axes: xlabel='Age', ylabel='Price'>



Search

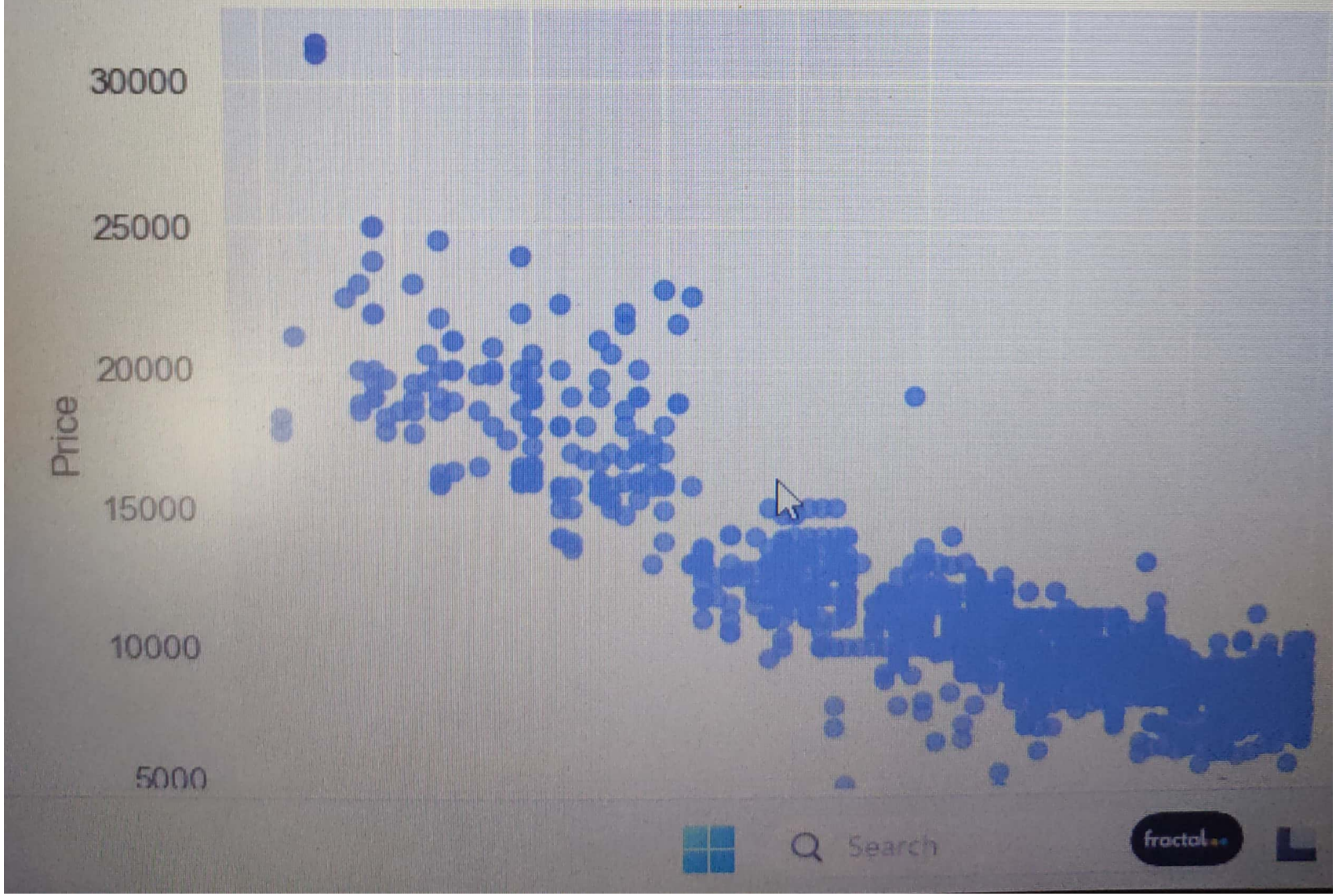
[28]:
```
# Without Regression Fit Line
sns.set(style='darkgrid')
sns.regplot(x= cars_data['Age'], y=cars_data['Price'], fit_reg=False)
```

[28]: <Axes: xlabel='Age', ylabel='Price'>



```
# Using markers
#sns.set(style='darkgrid')
sns.regplot(x= cars_data['Age'], y=cars_data['Price'],
            marker='o', fit_reg=False)
```

```
<Axes: xlabel='Age', ylabel='Price'>
```

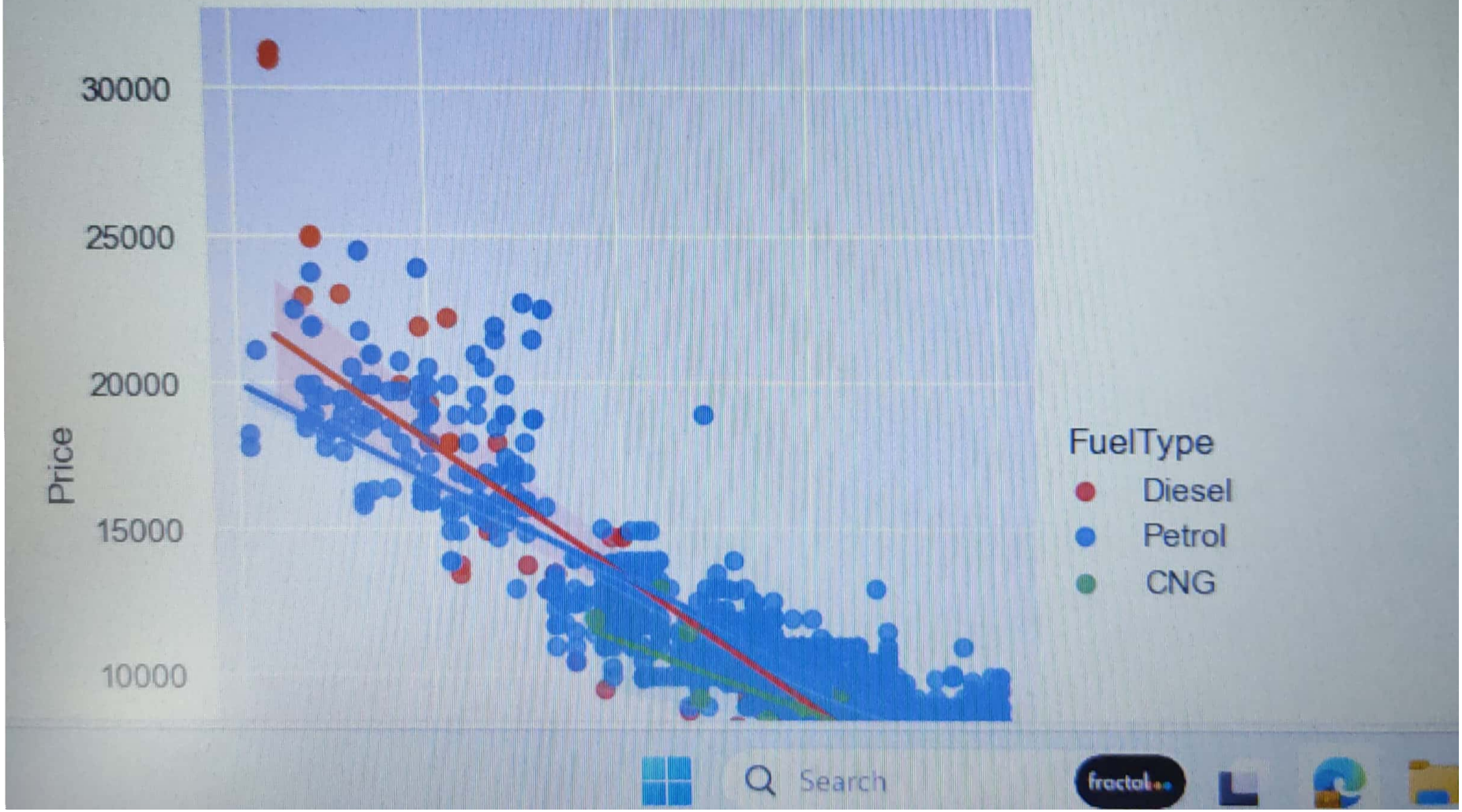


Scatter plot of Price v/s Age by FuelType

- Using hue parameter, including another variable to show the fuel types categories with different colors

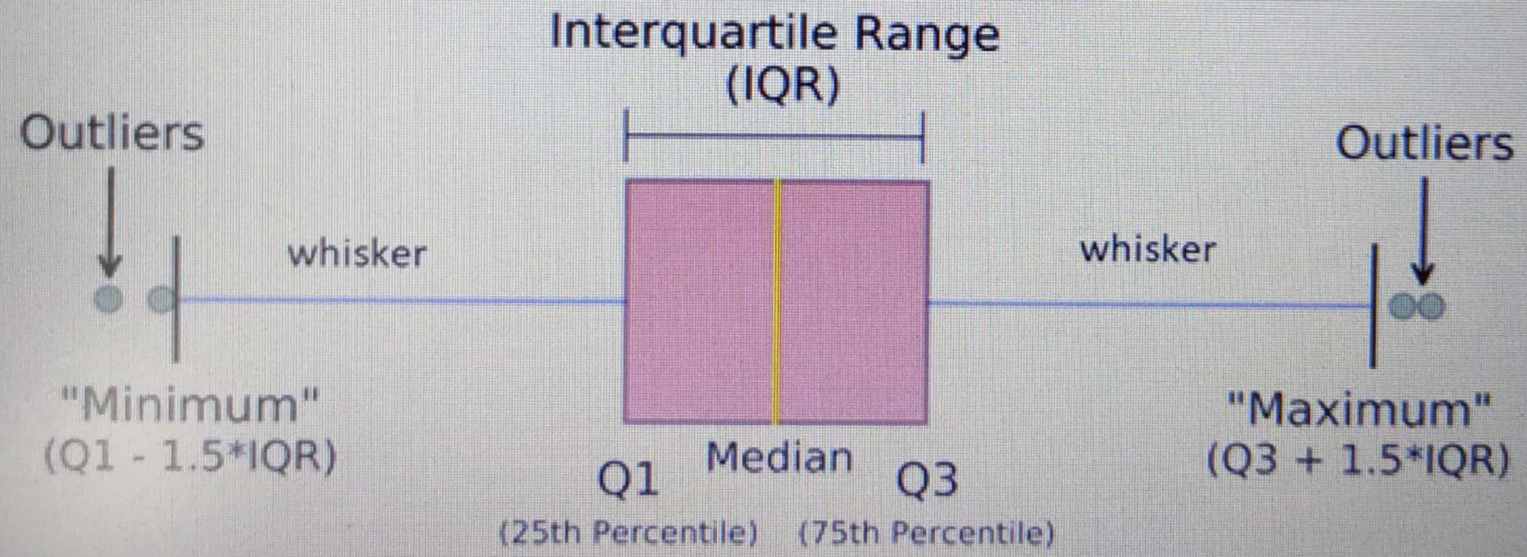
```
sns.lmplot(x='Age', y='Price', data=cars_data,
           fit_reg=True, hue='FuelType',legend=True, palette ="Set1")
```

```
<seaborn.axisgrid.FacetGrid at 0x1fa01b064f0>
```



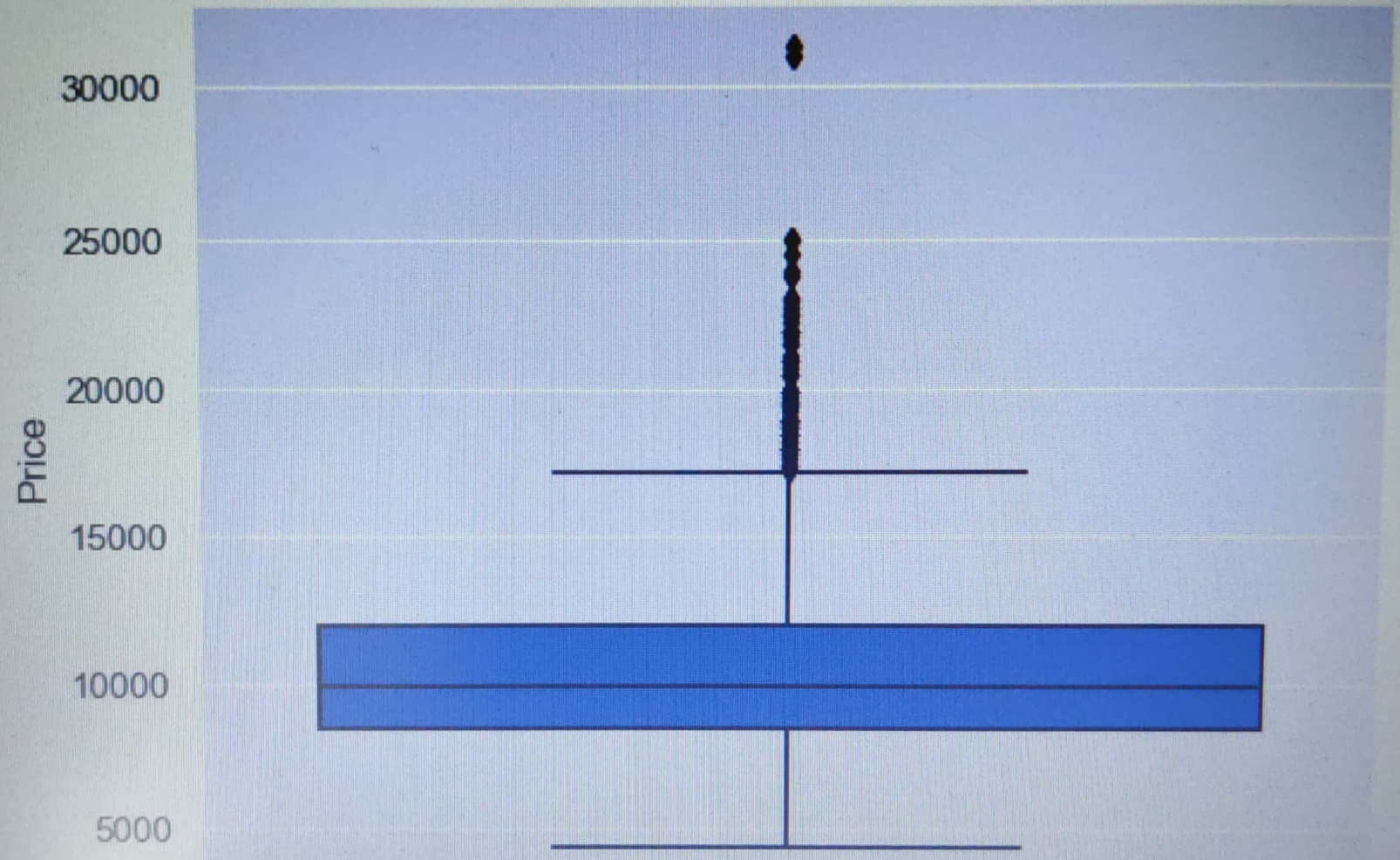
Box and Whisker Plot

- Gives visual interpretation of 5-number summary
- Five number summary includes:
  - Minimum: shows minimum value as a horizontal line (lower whisker), excluding outliers
  - Maximum: shows maximum value as a horizontal line (upper whisker), excluding outliers
  - three quartiles:
    - lower line of the box: 1st Quartile
    - Middle line: 2nd Quartile or Median
    - Uppler line: 3rd Quartile
- Outliers:
  - Greater than 1.5 * IQR and Lesser than 1.5*IQR



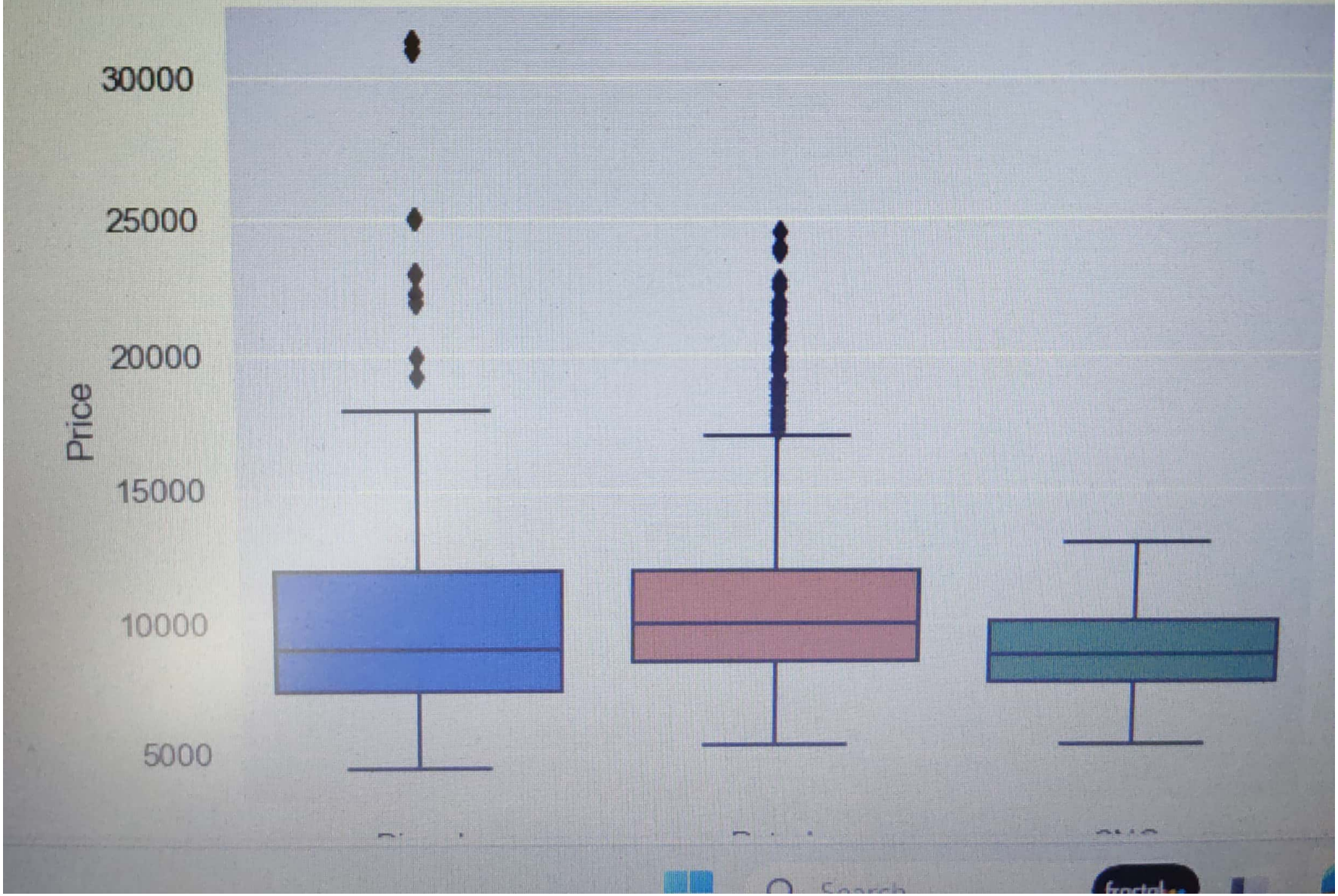
1]:
```
#Box and whiskers plot of Price to visually interpret the five-number summary
sns.boxplot(y= cars_data['Price'])
```

1]: <Axes: ylabel='Price'>



```
#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'])
```

```
<Axes: xlabel='FuelType', ylabel='Price'>
```



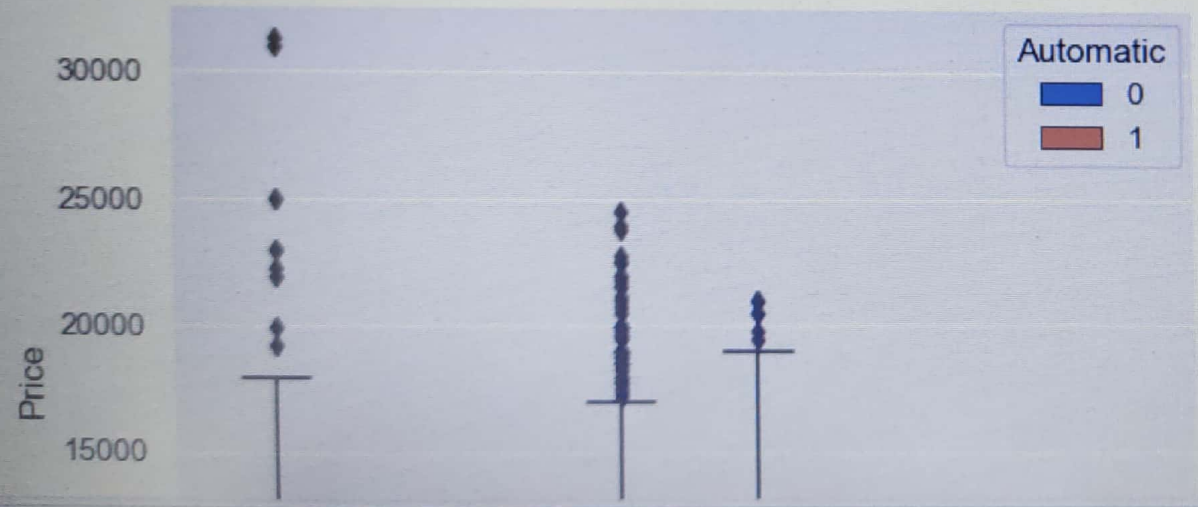
Inteferences:

- There are outliers in Diesel and Petrol cars, but CNG cars doesn't.
- Diesel cars have extremely high outliers.
- Petrol cars have more number of outliers
- More number of outliers in Petrol cars are causing the median value of Petrol cars being higher than that of Diesel cars. But, in reality, in the market, diesel cars tend to be costlier.
- Price range for CNG is narrower compared to other two types

33]:
```
#Grouped box and whiskers plot of Price vs FuelType and Automatic

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

33]:
```
<Axes: xlabel='FuelType', ylabel='Price'>
```



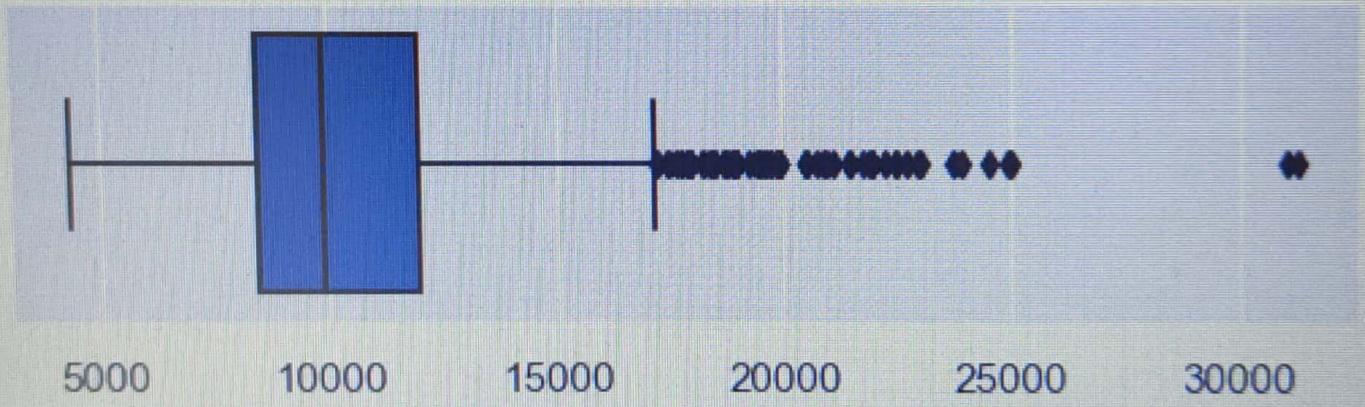
Box-whiskers plot and Histogram

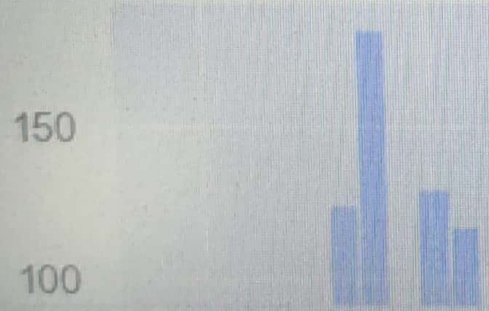
- Let's plot box-whiskers plot and histogram on the same window
- Split the plotting window into 2 parts

34]:

```
f, (ax_box,ax_hist) = plt.subplots(2, gridspec_kw={'height_ratios': (0.35,0.65)})
sns.boxplot(cars_data['Price'], ax=ax_box)
sns.distplot(cars_data['Price'], ax=ax_hist, kde=False)
```

34]: <Axes: xlabel='Price'>





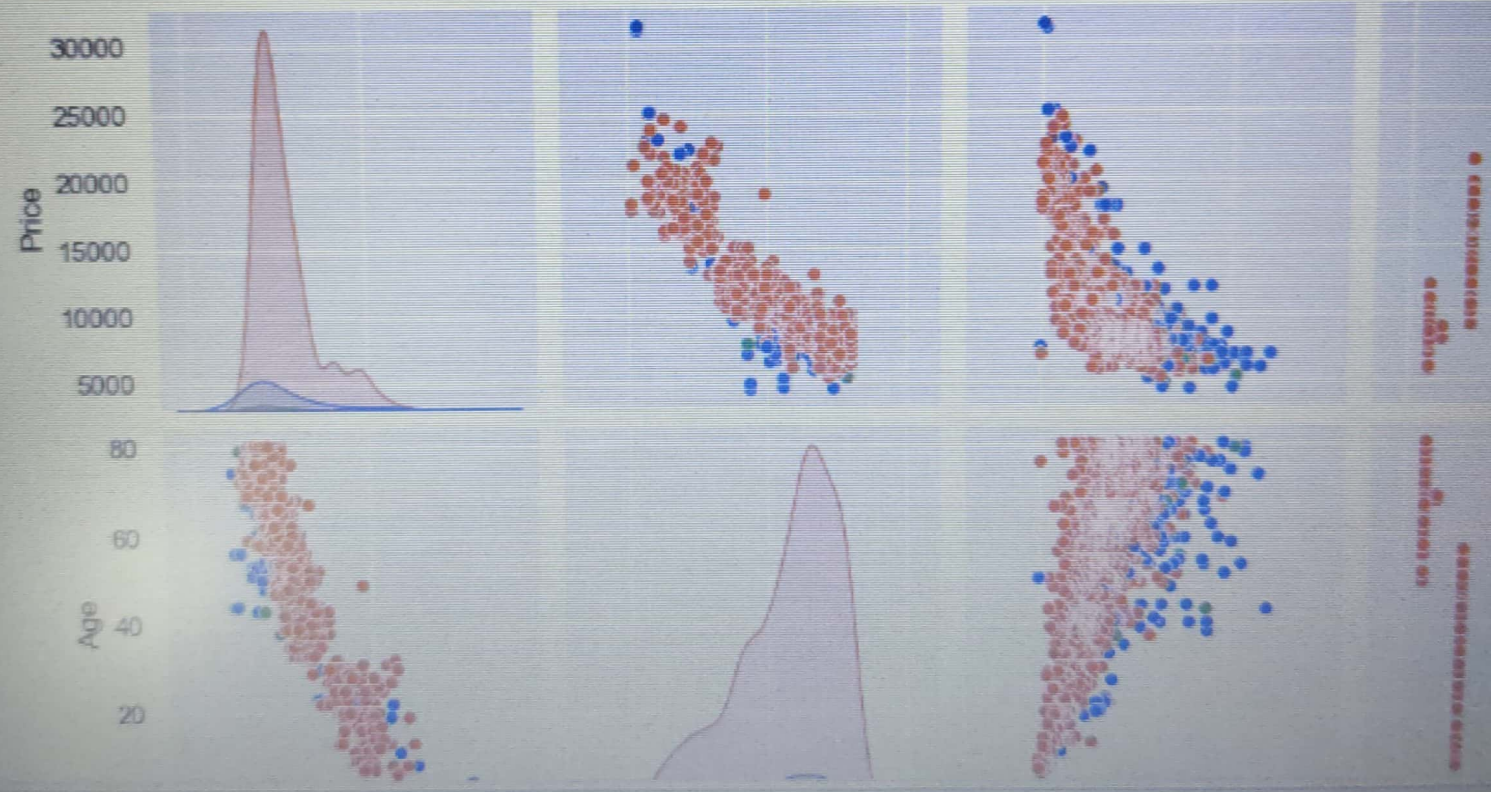
dit View Run Kernel Settings Help

Markdown

Pairwise Plots

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

5]:
```
sns.pairplot(cars_data, vars=["Price", "Age", "KM", "CC", "HP", "Weight"],
             kind="scatter", hue="FuelType")
plt.show()
```



Search

Heatmap (Correlation Plot)

[36]:
```
#numerical_data = cars_data.select_dtypes(exclude=[object])
numerical_data = cars_data[['Price', 'Age','KM', 'HP', 'CC','Weight']]
numerical_data.shape
```

[36]: (1096, 6)

[37]:
```
numerical_data.head()
```

[37]:

	Price	Age	KM	HP	CC	Weight
0	13500	23.0	46986.0	90.0	2000	1165
1	13750	23.0	72937.0	90.0	2000	1165
3	14950	26.0	48000.0	90.0	2000	1165
4	13750	30.0	38500.0	90.0	2000	1170
5	12950	32.0	61000.0	90.0	2000	1170

5	12950	32.0	61000.0	90.0	2000	1170

]:
```
corr_matrix= numerical_data.corr()
corr_matrix
```

]:

	Price	Age	KM	HP	CC	Weight
Price	1.000000	-0.877706	-0.601944	0.334261	0.099880	0.532614
Age	-0.877706	1.000000	0.525695	-0.162063	-0.084851	-0.442295
KM	-0.601944	0.525695	1.000000	-0.368629	0.319733	-0.029703
HP	0.334261	-0.162063	-0.368629	1.000000	0.037291	0.084527
CC	0.099880	-0.084851	0.319733	0.037291	1.000000	0.623643
Weight	0.532614	-0.442295	-0.029703	0.084527	0.623643	1.000000

99]:
```
ax=sns.heatmap(corr_matrix, annot=True)
plt.show()
```