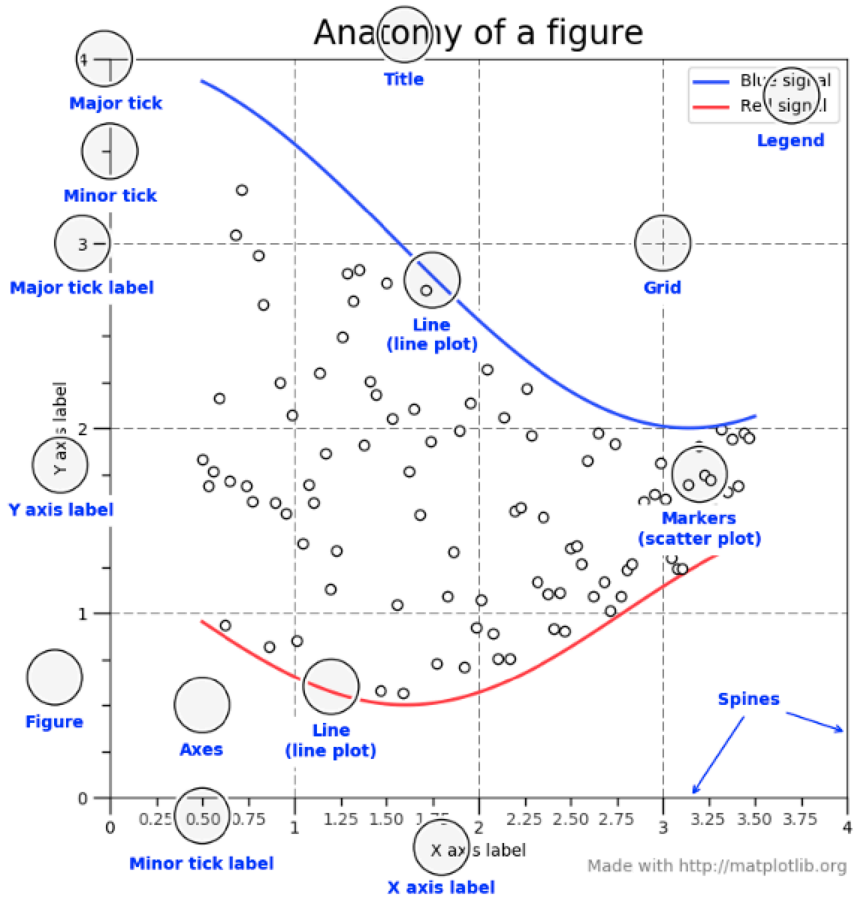
Module 02 Lab 01 - Basic Plots

Created	@July 2, 2025 1:53 PM
Class	IIITH
Event	Bootcamp
Subject	AI



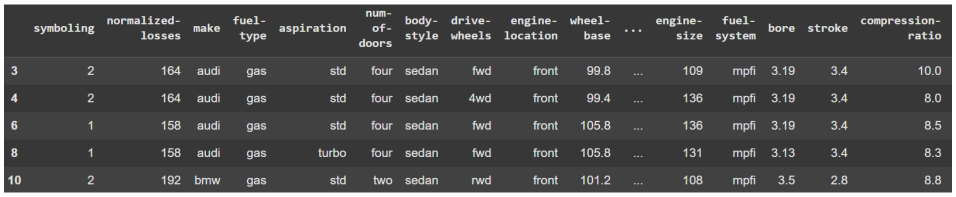
```
# Dividing the dataset into features (X) and the target variable (y)

X = data.drop("price", axis = 1)
y = pd.to_numeric(data["price"])
```

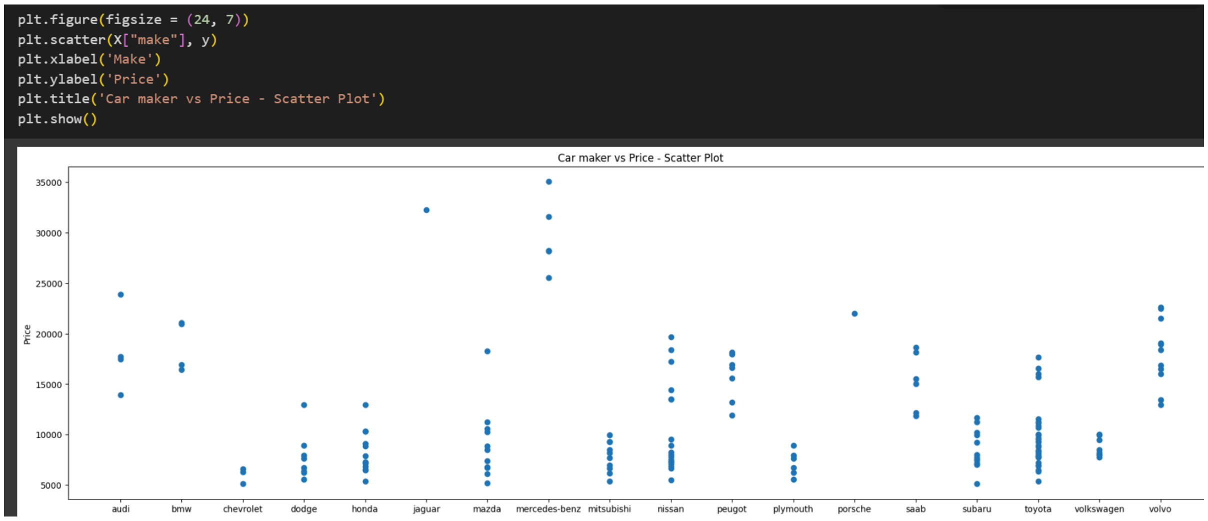
1. **Replaces missing values:** `data.replace("?", np.nan, inplace = True)` replaces all occurrences of the string "?" with `np.nan` (Not a Number), which is the standard way to represent missing values in NumPy and pandas. The `inplace=True` argument modifies the DataFrame directly.
2. **Removes rows with missing values:** `data = data.dropna()` removes all rows that contain at least one `np.nan` value. The resulting DataFrame, with missing values removed, is then assigned back to the `data` variable.

When you use `X = data.drop("price", axis = 1)`, you are **not** dropping the "price" column from the original `data` DataFrame. Instead, the `drop()` method, by default, returns a **new** DataFrame that is a copy of the original `data` but without the "price" column. This new DataFrame is then assigned to the variable `X`.

SCATTER PLOT



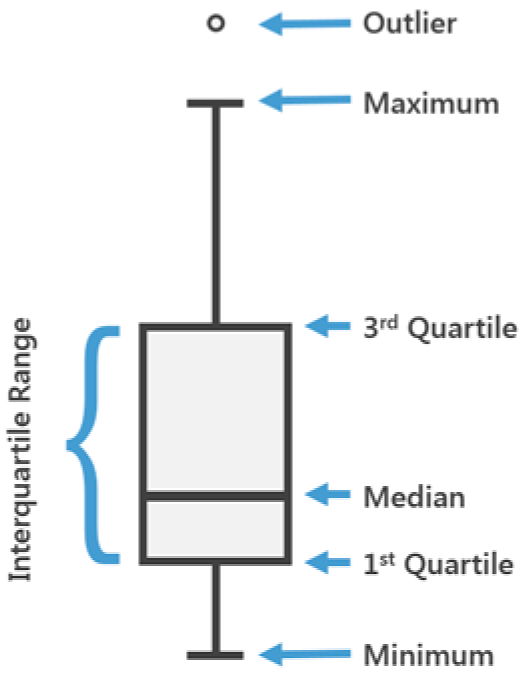
	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.4	10.0
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.4	8.0
6	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.4	8.5
8	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.4	8.3
10	2	192	bmw	gas	std	two	sedan	rwd	front	101.2	...	108	mpfi	3.5	2.8	8.8



```
plt.figure(figsize = (24, 7))
plt.scatter(X["make"], y)
plt.xlabel('Make')
plt.ylabel('Price')
plt.title('Car maker vs Price - Scatter Plot')
plt.show()
```

Scatter plot

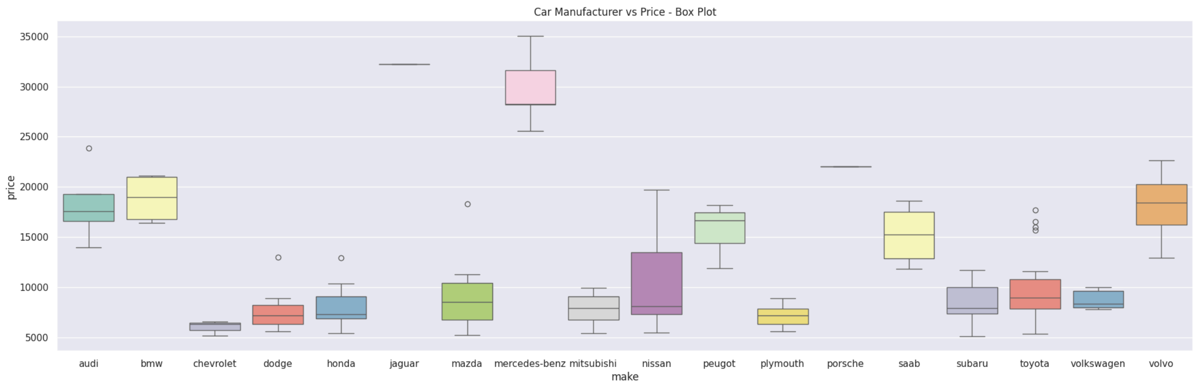
BOX PLOT



We will be creating a Box Plot. This is the type of plot that can be used to obtain more of the statistical details about the data. The straight lines at the maximum and minimum are also called as whiskers. Points outside of whiskers will be inferred as an outliers. The box plot gives us a representation of 25th, 50th ,75th quartiles. From a box plot we can also see the Interquartile range(IQR) where maximum details of the data will be present. It also gives us a clear overview of outlier points in the data.

```
sns.set(rc={'figure.figsize':(24,7)})
sns.boxplot(x=X["make"],y=y, palette="Set3").set_title('Car Manufacturer vs P
rice - Box Plot')
```

IQR



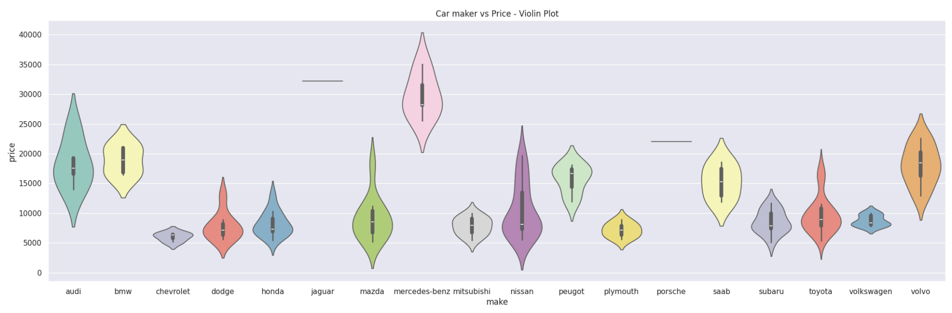
Box plot

It becomes evidently clear that Mercedes-Benz, Jaguar and Porsche have the costliest cars. BMW too has costly cars but the large interquartile range suggests that the company has a broader range of cars in the market. The outliers are also important as for example, Mazda has a car listed much higher than its normal price range.

All of these conclusions could be very hard to get to using the data in a tabular format, whereas it became evidently clear straight away once we used Visualization techniques to understand the dataset.

Violin plot

```
sns.violinplot(x=X["make"],y=y, palette="Set3").set_title('Car maker vs Price -
Violin Plot')
```



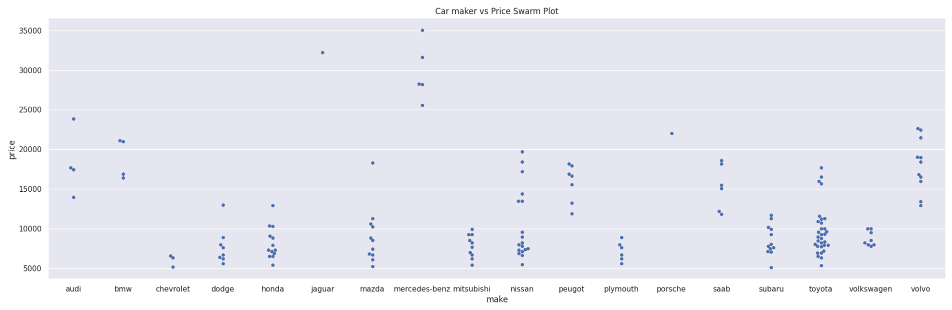
1. **Reveals the full distribution shape**: Unlike box plots, which only show summary statistics (quartiles, median, outliers), violin plots show the entire distribution of the data. This allows you to see if the distribution is unimodal (one peak), multimodal (multiple peaks), skewed, or symmetrical.

2. **Highlights data density**: The width of the violin at any given value indicates the density of the data at that point. This helps you understand where the data is most concentrated.

Swarm Plots.

Another plot type are the Swarm Plots. They offer an alternate way of plotting the distribution of an attribute or the joint distribution of a couple of attributes. Unlike strip plots, swarm plots attempt to avoid obscuring points by calculating non-overlapping positions instead of adding random jitter. This sort of gives them appearance of a swarm of bees, or perhaps a honeycomb.

```
sns.swarmplot(x=X["make"],y=y).set_title('Car maker vs Price Swarm Plot')
```

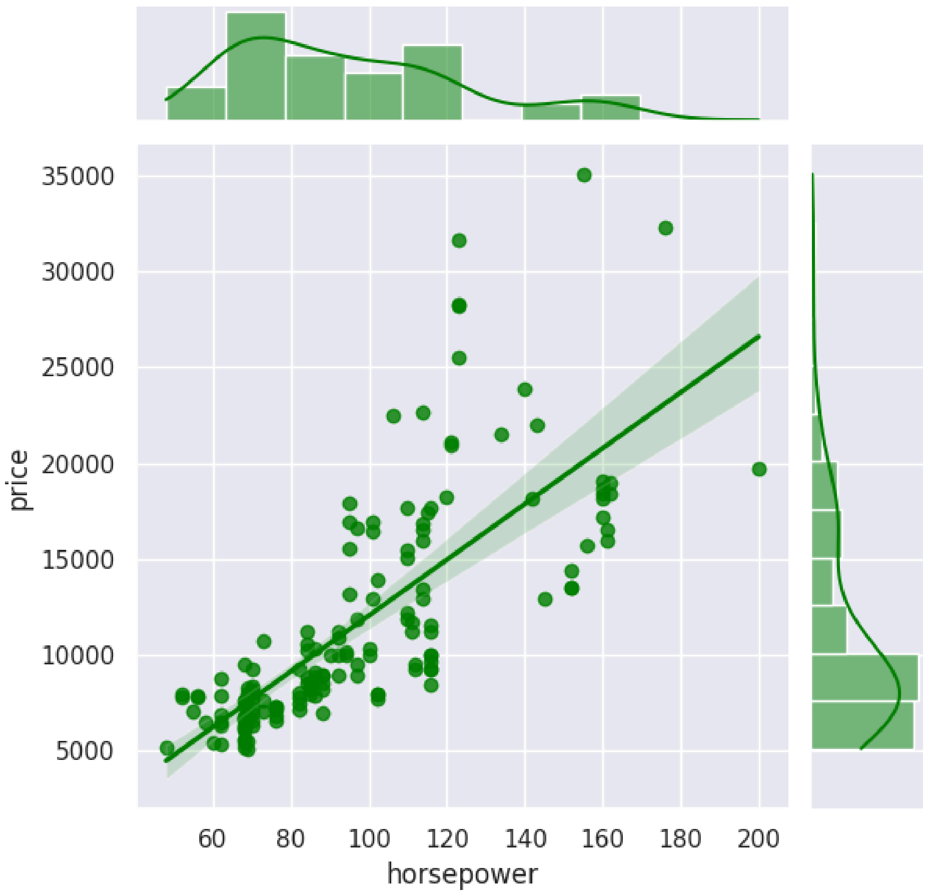


JOINT PLOT

The jointplot displays a relationship between 2 variables (bivariate) as well as 1D profiles (univariate) in the margins.

I have also used the parameter "reg" which fits a regression line through the points making the correlation sign apparent. A line with positive slope would indicate a postiive correlation and vice versa. A line parallel to x-axis would indicate no correlation between the 2 variables (implying independence).

```
sns.jointplot(x=pd.to_numeric(X["horsepower"]), y = y, kind="reg", color = 'gr
een')
```


35000
30000
25000
20000
15000
10000
5000
price
60
80
100
120
140
160
180
200
horsepower