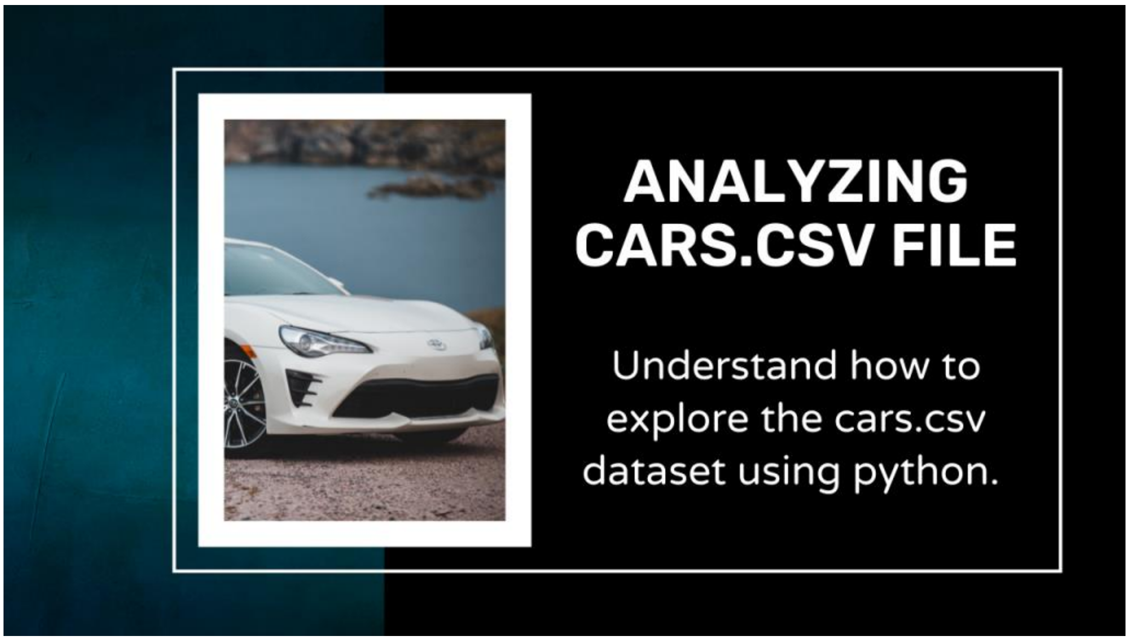
Correlation is simple with Seaborn and Pandas

[illegible]

By Abishek James


ANALYZING
CARS.CSV FILE
Understand how to explore the cars.csv dataset using python.

Datasets can tell many stories. A great place to start, to see these stories unfold, is checking for correlations between the variables. One of the first tasks I perform when exploring a dataset to see which variables have correlations. This gives me a better understanding of the data I am working with. It is also a great way to develop an interest in the data and establish some initial questions to try to answer. Simply put, correlations are awesome.

Luckily Python has some amazing libraries which give us the tools we need to look at correlations quickly and efficiently. Let us take a brief look at what correlation is and how to find strong correlations in a dataset using a heat map.

What is Correlation?

Correlation is a way to determine if two variables in a dataset are related in any way. Correlations have many real-world applications. We can see if using certain search terms are correlated to views on YouTube. Or, we can see if ads are correlated to sales. When building machine learning models correlations are an important factor in determining features. Not only can this help us to see which features are linear related, but if features are strongly correlated, we can remove them to prevent duplicating information.

How Do You Measure Correlation?

In data science we can use the r value, also called Pearson’s correlation coefficient. This measures how closely two sequences of numbers(i.e., columns, lists, series, etc.) are correlated.

Using Python to Find Correlation

Let us look at a larger dataset and see how easy it is to find correlations using Python.

Steps to Analyse Cars.csv Dataset in Python

We will be using **Pandas** and **NumPy** for this analysis. We will also be playing around with visualizations using the Seaborn library. Let us get right into this.

Dataset

Cars.csv is used, we will find out correlation between the features from this dataset. let us understand how to explore the cars.csv dataset using Python.

1. Loading the Cars.csv Dataset

Since the dataset is already in a CSV format, all we need to do is format the data into a Pandas data frame. This was done by using a pandas data frame method called read_csv by importing Pandas library.

The read_csv data frame method is used by passing the path of the CSV file as an argument to the function. The code results in a neatly organized pandas data frame when we make use of the head function.

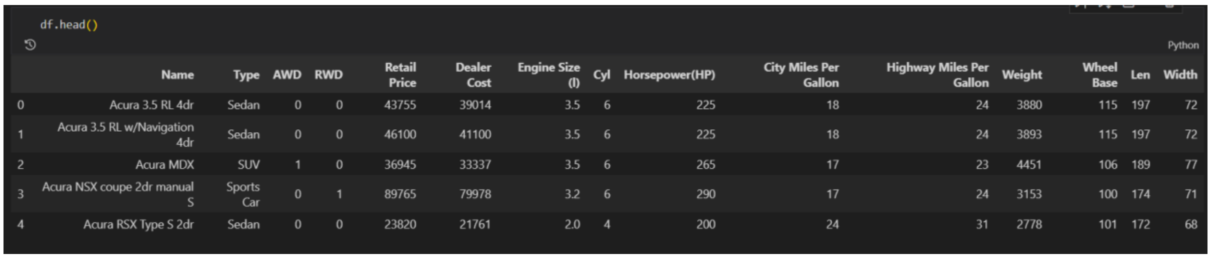
Let’s start by importing all the necessary modules and libraries into our code.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Now the data is loaded with the help of the pandas module.

```
df = pd.read_csv('.\Cars.csv')
```

Now let's use **head()** function that returns the first n rows for the object based on position.

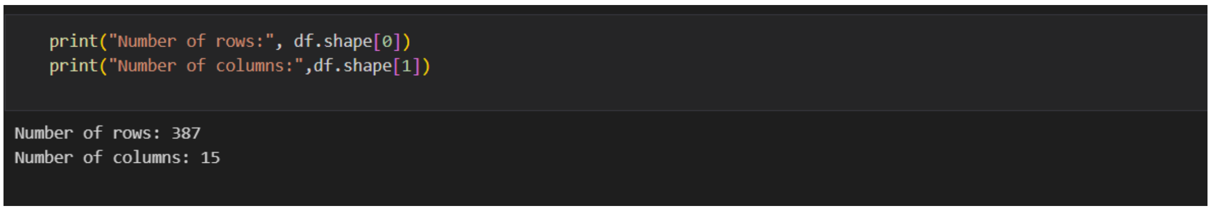


```
df.head()
```

Python

	Name	Type	AWD	RWD	Retail Price	Dealer Cost	Engine Size (l)	Cyl	Horsepower(HP)	City Miles Per Gallon	Highway Miles Per Gallon	Weight	Wheel Base	Len	Width
0	Acura 3.5 RL 4dr	Sedan	0	0	43755	39014	3.5	6	225	18	24	3880	115	197	72
1	Acura 3.5 RL w/Navigation 4dr	Sedan	0	0	46100	41100	3.5	6	225	18	24	3893	115	197	72
2	Acura MDX	SUV	1	0	36945	33337	3.5	6	265	17	23	4451	106	189	77
3	Acura NSX coupe 2dr manual S	Sports Car	0	1	89765	79978	3.2	6	290	17	24	3153	100	174	71
4	Acura RSX Type S 2dr	Sedan	0	0	23820	21761	2.0	4	200	24	31	2778	101	172	68

Use **Shape** attribute in Pandas enables us to obtain the shape of a Dataset.



```
print("Number of rows:", df.shape[0])
print("Number of columns:",df.shape[1])
```

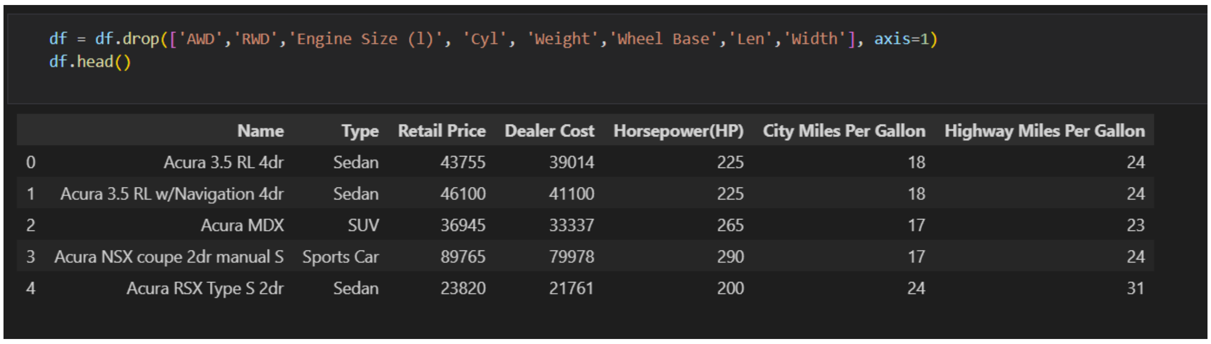
```
Number of rows: 387
Number of columns: 15
```

This data set has 387 rows and 15 features having data about different car brands such as Audi, BMW, Mercedes, Cadillac, Chevrolet, Dodge, Ford, Honda, Hyundai and more and has multiple features about these cars such as Name, Type, AWD, RWD, Retail price, Dealer Cost and more such features.

2. Removing irrelevant features

We aim to remove the same irrelevant features from our dataset.

The features that we are going to remove are AWD, RWD, Engine Size, Cylinder, Weight, Wheel base, Length, Width. All those features are not necessary to determine the costs. You can remove or keep features according to your preferences.

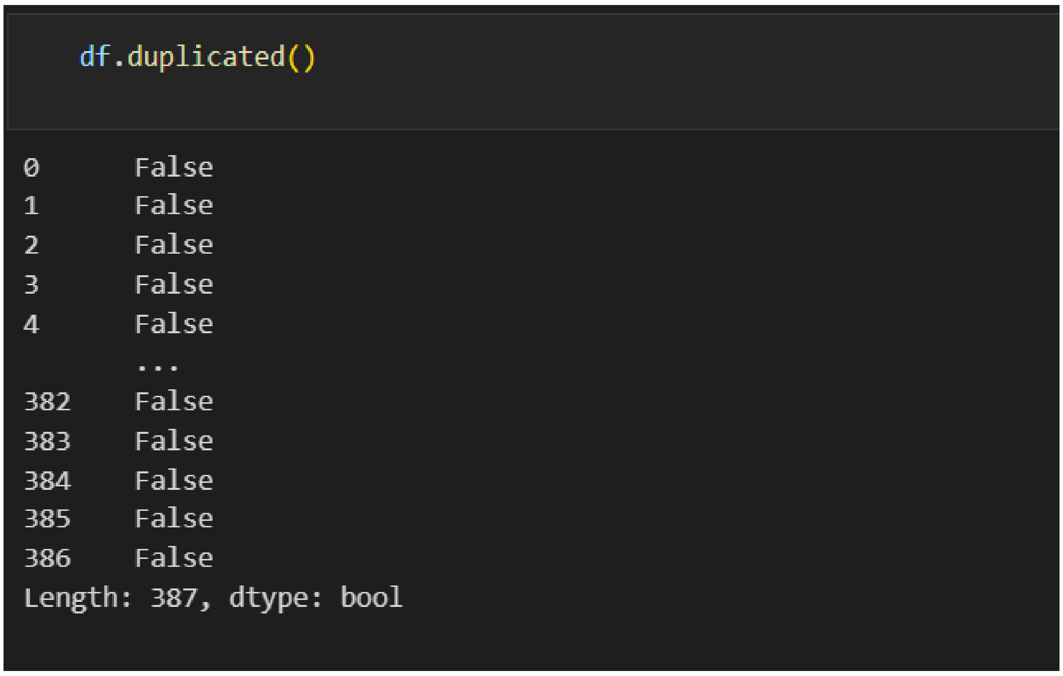


```
df = df.drop(['AWD','RWD','Engine Size (l)', 'Cyl', 'Weight','Wheel Base','Len','Width'], axis=1)
df.head()
```

	Name	Type	Retail Price	Dealer Cost	Horsepower(HP)	City Miles Per Gallon	Highway Miles Per Gallon
0	Acura 3.5 RL 4dr	Sedan	43755	39014	225	18	24
1	Acura 3.5 RL w/Navigation 4dr	Sedan	46100	41100	225	18	24
2	Acura MDX	SUV	36945	33337	265	17	23
3	Acura NSX coupe 2dr manual S	Sports Car	89765	79978	290	17	24
4	Acura RSX Type S 2dr	Sedan	23820	21761	200	24	31

3. Finding duplicate data

In any dataset, there might be duplicate/redundant data and we can remove if it is found. We use **duplicated () method** to find duplicate rows in a Dataset.



```
df.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
382    False
383    False
384    False
385    False
386    False
Length: 387, dtype: bool
```

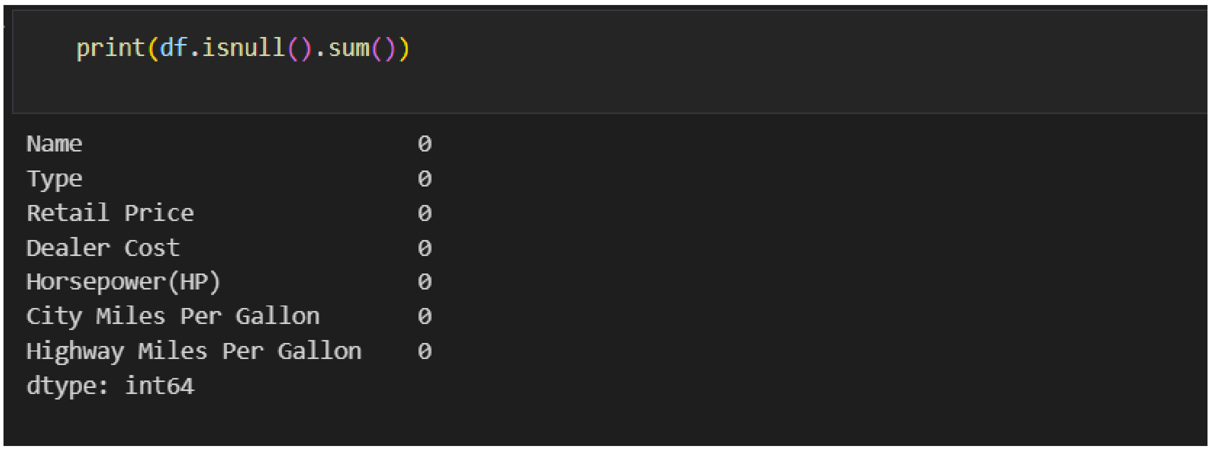
Here we got the value False that means there is no duplicate values. The value True denoting duplicate. In other words, the value True means the entry is identical to a previous one.

4. Finding the missing or null values

No dataset is perfect and having missing values in the dataset is a common thing to happen. Now, there are several approaches to deal with the missing value.

One can either drop either row or fill the empty values with the mean of all values in that column. It is better to take the mean of the column values rather than deleting the entire row as every row is important for a developer.

Let us first look at how many null values we have in our dataset. How many missing values exist in the collection, in this case we can **use.sum()** chained on:



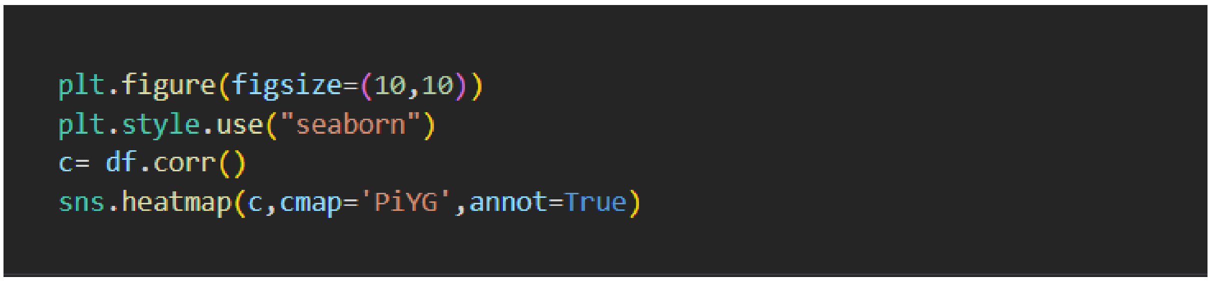
```
print(df.isnull().sum())
```

```
Name                        0
Type                        0
Retail Price                0
Dealer Cost                 0
Horsepower(HP)              0
City Miles Per Gallon       0
Highway Miles Per Gallon    0
dtype: int64
```

There are no missing values.

5. Visualizing Heatmaps

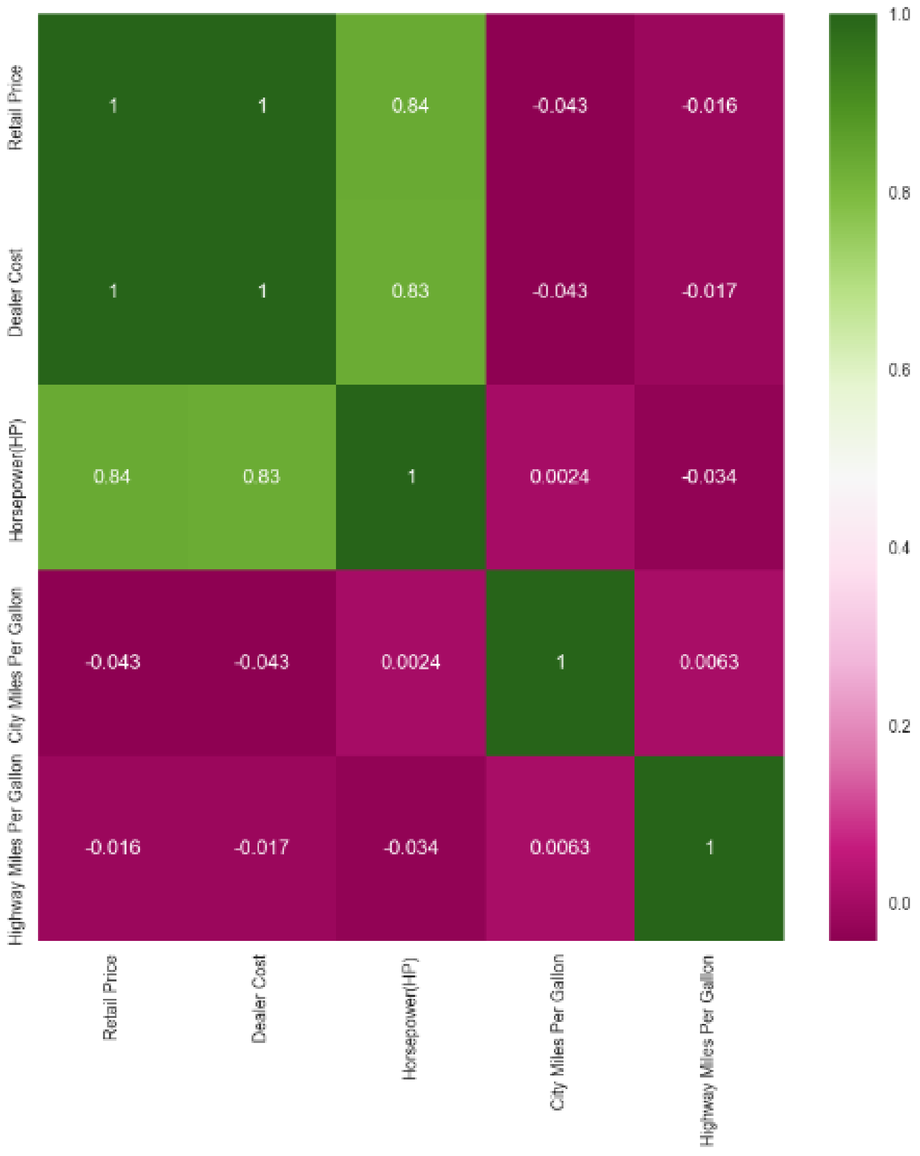
Heatmaps are the maps that are one of the best ways to find the correlation between the features.



```
plt.figure(figsize=(10,10))
plt.style.use("seaborn")
c= df.corr()
sns.heatmap(c,cmap='PiYG',annot=True)
```

The heatmaps is plotted.

```
Out[6]: <Axes: >
```



Correlations we found

- A strong positive correlation between Retail Price and Rotten Dealer Cost.
- No correlation between City Miles and Highway Miles.

Conclusion

Correlations are useful to help explore a new dataset. By using seaborn’s heatmap we easily saw where the strongest correlations are. Now you can go to Kaggle and check out a few more datasets to see what other correlations might spark your interest!

Hope you understood the concept and would apply the same in various other CSV files. Thank you for reading!