Exploratory Data Analysis (EDA) in Python

Dataset Name :-> Cars Dataset

Introduction

What is Exploratory Data Analysis?

    Exploratory Data Analysis (EDA) is an understanding the data sets by summurizing their main characteristics often plotting them visually .This step is very important especially when we arrive at modelling the data in order to apply Machine Learning Algorithms .Plotting in EDA consists of Histograms,Boxplot,ScatterPlot and Many more .It often takes much time to explore the data.Through the process of EDA ,we can ask to define the problem statement or defination on our data set which is very important.

How to perform Exploratory Data Analysis?

    This is one such question that everyone is keep on knowing the answer well,the answer is it depends on the data set that you are working .There is no one method or common methods in order to perform EDA , whereas in this tutorial you can understand some common methods and plots that would be used in the EDA process.

What data are we exploring now?

    Since I am a huge fan of cars ,I got a very beautiful data-set of cars from kaggle .To give a piece of brief information about the data set this data contains more of 10,000 records and more then 10 columns which contains features of the car such as Engine Fuel Type ,Engine HP ,Transmission Type,highway MPG,city MPG and many more .So in this tutorial ,we will explore the data and make it ready for modelling.

We know very well of steps to analyse the data..Data Preprocessing

1.Importing the required libraries for EDA

    Below are the libraries that are used in order to perform EDA(Exploratory data analysis) in this tutorial.

In [15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set(color_codes=True)

import warnings
```

2.Loading the data into the data frame

    Loading the data into the pandas data frames is certainly one of the most important steps in EDA,as we can see that the value from the data set is comma-seperated .So all we have to do is to just read the CSV into a data frame and pandas data frame does the job for us.

In [237…
```
df=pd.read_csv('cars dataset.csv')
df.head()
```

Out[237…

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehic Sty
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Cou
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertit
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Cou
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Cou
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertit

In [239…

```
df.tail()
```

Out[239…

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category
11909	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury
11910	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury
11911	Acura	ZDX	2012	premium unleaded (required)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury
11912	Acura	ZDX	2013	premium unleaded (recommended)	300.0	6.0	AUTOMATIC	all wheel drive	4.0	Crossover,Hatchback,Luxury
11913	Lincoln	Zephyr	2006	regular unleaded	221.0	6.0	AUTOMATIC	front wheel drive	4.0	Luxury

3.Cheking for the data types of data

    Here we check for the datatypes because sometimes the MSRP or the price of the car would be
    stored as a string ,if in that case ,we have to convert that string to the integer data only
    then we can plot the data via a graph .Here ,in this case,the data is already in integer
    format so nothing to worry

In [241…

```
df.dtypes
```

Out[241…

```
Make                  object
Model                 object
Year                   int64
Engine Fuel Type      object
Engine HP            float64
Engine Cylinders     float64
Transmission Type     object
Driven_Wheels         object
Number of Doors      float64
Market Category       object
Vehicle Size          object
Vehicle Style         object
highway MPG            int64
city mpg               int64
Popularity             int64
MSRP                   int64
dtype: object
```

4.Droping irrelevent columns

    This step is certianly needed in every EDA because sometimes there would be many columns that we never use in such cases dropping is the only solution.In this case,the columns such as Engine FUel Type,Market Category,Vehicle Style,Popularity ,Number of doors ,vehicle Size doesn't make any sense to me so I just dropped for this instance

In [243…
```
df=df.drop(['Engine Fuel Type','Market Category','Vehicle Style','Popularity','Number of Doors','Vehicle Size']
df.head()
#df.drop([...], axis=1) removes columns, not rows (since axis=1 specifies column-wise operation).
```

Out[243…

	Make	Model	Year	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	highway MPG	city mpg	MSRP
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

5.Renaming the columns

In this instance ,most of the columns names are very confusing to read,so I just renamed thier column names.This is good approach it improves the readability of the data set .

In [245…
```
df=df.rename(columns={"Engine HP":"HP","Engine Cylinders":"Cylinders","Transmission Type":"Transmission","Drive
```

In [247…
```
df.head()
```

Out[247…

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

6.Dropping the duplicate rows

This is often a handy thing to do because a huge data set as in this case contains more than 10,000 records often have some duplicate data which might be disturbing,so here I remove all the duplicates value from the datset.For example prior to removing I had 11914 rows of data but after removing the duplicates 10925 data meaning that I had 989 of duplicates data.

In [249…
```
df.shape
```

Out[249…
```
(11914, 10)
```

In [251…
```
duplicate_rows_df=df[df.duplicated()]
print("number of duplicate rows:",duplicate_rows_df.shape)
```

```
number of duplicate rows: (989, 10)
```

Now let us remove the duplicate data because it's ok to remove them.

In [253…
```
df.count()
```

Out[253…
```
Make            11914
Model           11914
Year            11914
HP              11845
Cylinders       11884
Transmission    11914
Drive Mode      11914
MPG-H           11914
MPG-C           11914
Price           11914
dtype: int64
```

So seen above there are 11914 rows and we are removing 989 rows of duplicated data.

In [255…
```
df=df.drop_duplicates()
df.head()
```

Out[255…

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650
2	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	20	36350
3	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	29450
4	BMW	1 Series	2011	230.0	6.0	MANUAL	rear wheel drive	28	18	34500

In [257…
```
df.count()
```

Out[257…
```
Make            10925
Model           10925
Year            10925
HP              10856
Cylinders       10895
Transmission    10925
Drive Mode      10925
MPG-H           10925
MPG-C           10925
Price           10925
dtype: int64
```

7.Droppeing or Filling missing or null values

    This is mostly similar to the previous step but in here all the missing values are detected and are dropped later.Now,this is not a good approach to do so,because many people just replace the missing values with the mean or the average of the column,but in this case,I just dropped that missing values.This is because there is nearly 100 missing value compared to 10,000 values this is small number and this is neglisible so I just dropped those values.

In [259…
```
df.isnull().sum()
```

Out[259…
```
Make             0
Model            0
Year             0
HP              69
Cylinders       30
Transmission     0
Drive Mode       0
MPG-H            0
MPG-C            0
Price            0
dtype: int64
```

    This is reason in the above step while counting both Cylinders and Horsepower (HP) had 10856 and 10895 over 10925 rows.

In [261…
```
df=df.dropna()
```

In [263…
```
df.count()
```

Out[263…
```
Make            10827
Model           10827
Year            10827
HP              10827
Cylinders       10827
Transmission    10827
Drive Mode      10827
MPG-H           10827
MPG-C           10827
Price           10827
dtype: int64
```

    Now we have removed all the rows which contain the Null or N/A values (Cylinders and Horsepower (HP))

In [265…
```
df.isnull().sum()
```

```
Out[265… Make            0
Model           0
Year            0
HP              0
Cylinders       0
Transmission    0
Drive Mode      0
MPG-H           0
MPG-C           0
Price           0
dtype: int64
```

8. Detecting Outliers

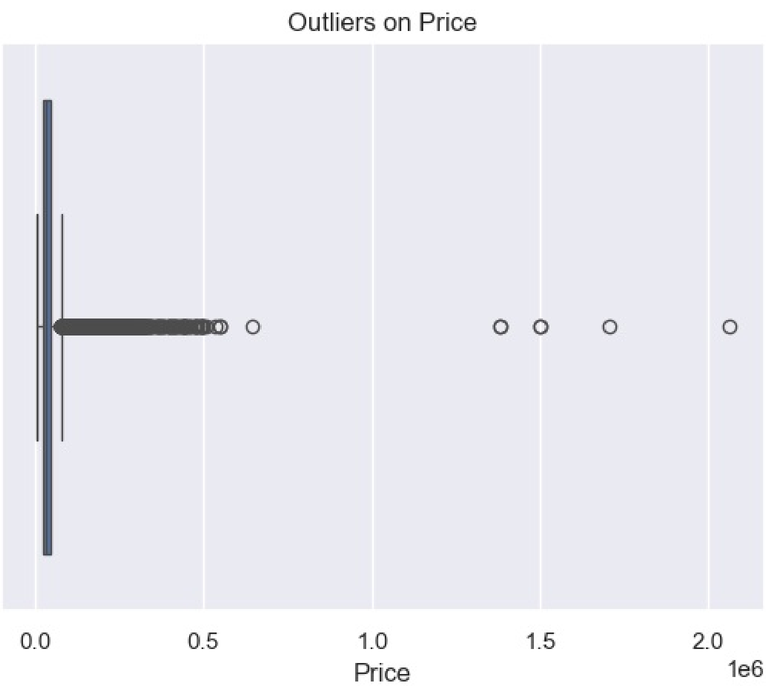
An outlier is a point or set of points that are different from other points .Sometimes they can be very high or very low.It's often a good idea to detect and remove the outliers.Because outliers are one of the primary reasons for resulting in a less accurate model.Hence It's good idea to remove them.The outliers detection and removing that I am going to perform is called IQR score technique.Often outliers can be seen with visualizations using a box plot. Shown below are the box plot of MSRP,Cylinders ,HP and EngineSize.Here in all the plots ,you can find some points are ouside the box they are none other tahn outliers .The technique of finding and removing outliers that I am performing in this assignment is taken hekp of a tutorial from towards data science.

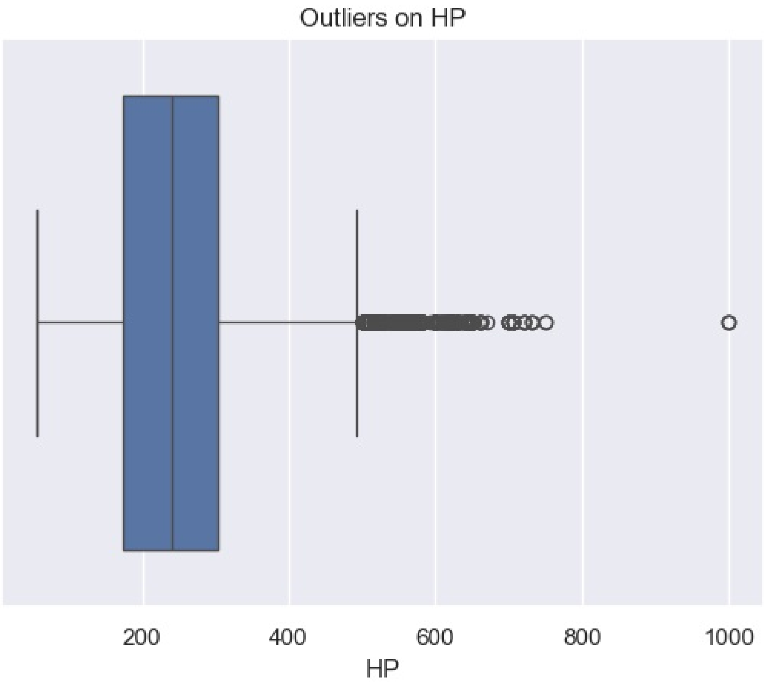
In [267…
```
df.head(2)
```

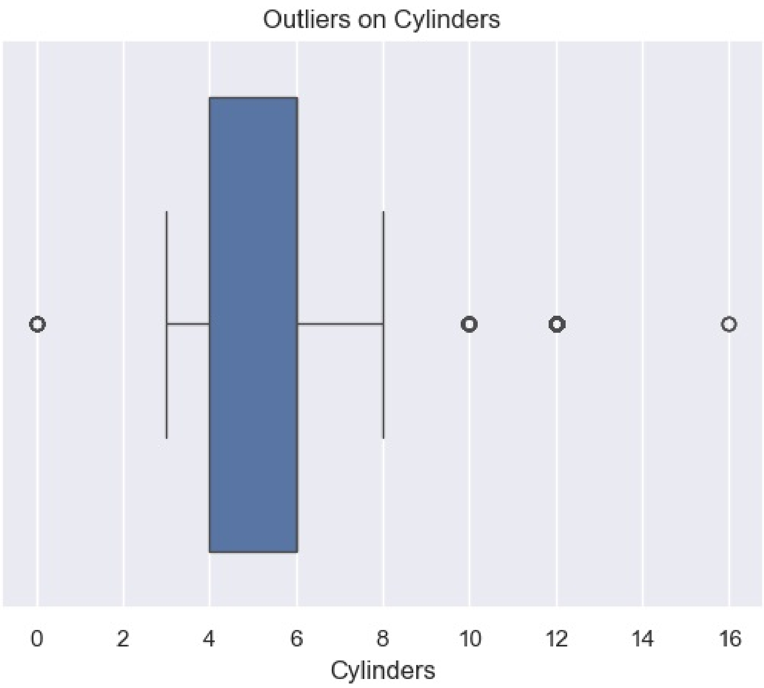
Out[267…

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650

In [269…
```
for i in ['Price','HP','Cylinders']:
    sns.boxplot(data=df,x=i)
    plt.title(f"Outliers on {i}")
    plt.show()
```







To detect outliers using the Interquartile Range (IQR) method, you can follow these steps:

1.Calculate the First (Q1) and Third Quartile (Q3):

Q1 is the 25th percentile.
Q3 is the 75th percentile.

2.Calculate the IQR:

IQR=$Q3-Q1$

3.Define the Outlier Range:

* Any data point below $Q1-1.5\times IQR$ or above $Q3+1.5\times IQR$ is considered an outlier.

4.Identify Outliers:

* Filter the data to find values outside the defined range.

In [271…
```
Q1 = df.select_dtypes(include='number').quantile(0.25)
Q3 = df.select_dtypes(include='number').quantile(0.75)
IQR=Q3-Q1
IQR
```

Out[271…
```
Year            9.0
HP            130.0
Cylinders       2.0
MPG-H           8.0
MPG-C           6.0
Price       21327.5
dtype: float64
```

We don't worry about the above values because it's not important to know each and every one of them because it's just important to know how to use this technique in order to remove the outliers.

In [273…
```
df = df[~((df.select_dtypes(include='number') < (Q1 - 1.5 * IQR)) |(df.select_dtypes(include='number') > (Q3 + 1
df.shape
```

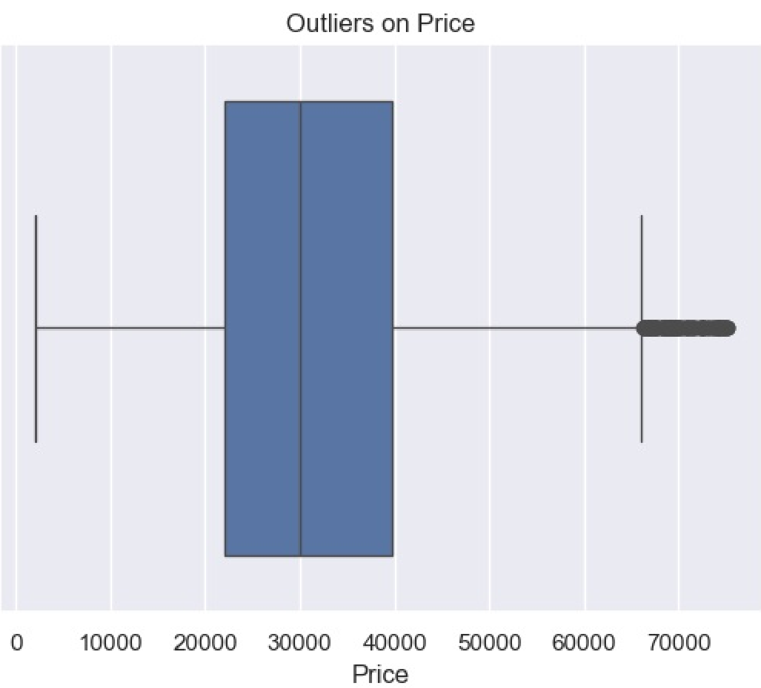
Out[273…
```
(9191, 10)
```

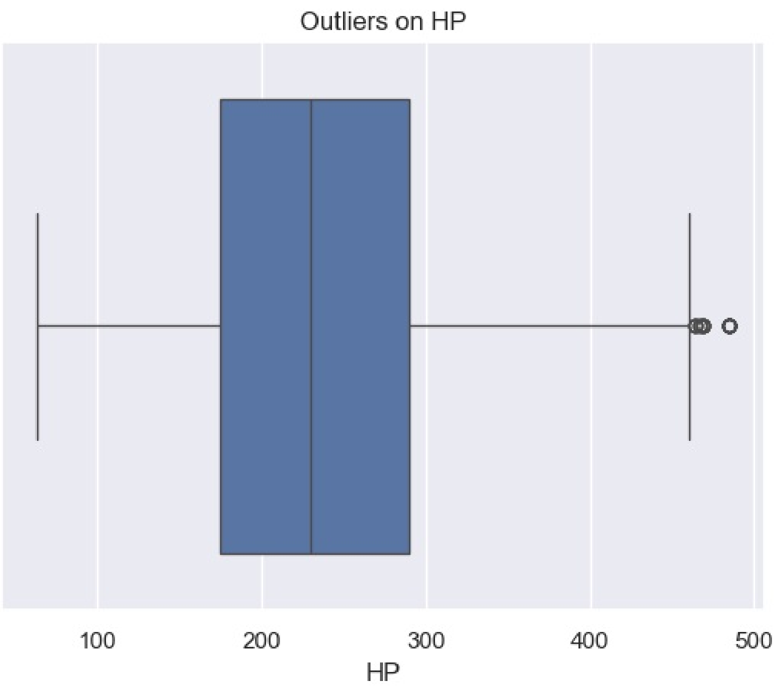
In [275…
```
df.isnull().sum()
```

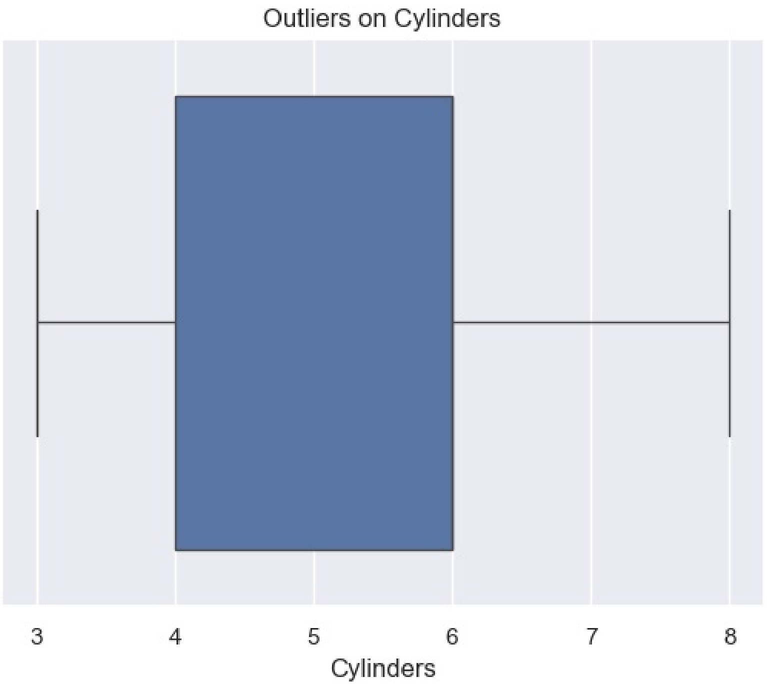
Out[275…
```
Make            0
Model           0
Year            0
HP              0
Cylinders       0
Transmission    0
Drive Mode      0
MPG-H           0
MPG-C           0
Price           0
dtype: int64
```

As seen above there were around 1600 rows were outliers.But you cannot completely remove the outliers because even after you use the above technique there maybe 1-2 outliers unremoved but that ok because there were more than 100 outliers .Something is better than nothing.

In [281…
```
for i in ['Price','HP','Cylinders']:
    sns.boxplot(data=df,x=i)
    plt.title(f"Outliers on {i}")
    plt.show()
```







9.Plot different features against one another (scatter) ,against frequency(hsitogram)

Histogram

```
Histogram refers to the frequency of occurrence of variables in an interval. In this case,
there are mainly 10 different types of car manufacturing companies, but it is often important
to know who has the most number of cars. To do this histogram is one of the trivial solutions
which lets us know the total number of car manufactured by a different company.
```

In [277…

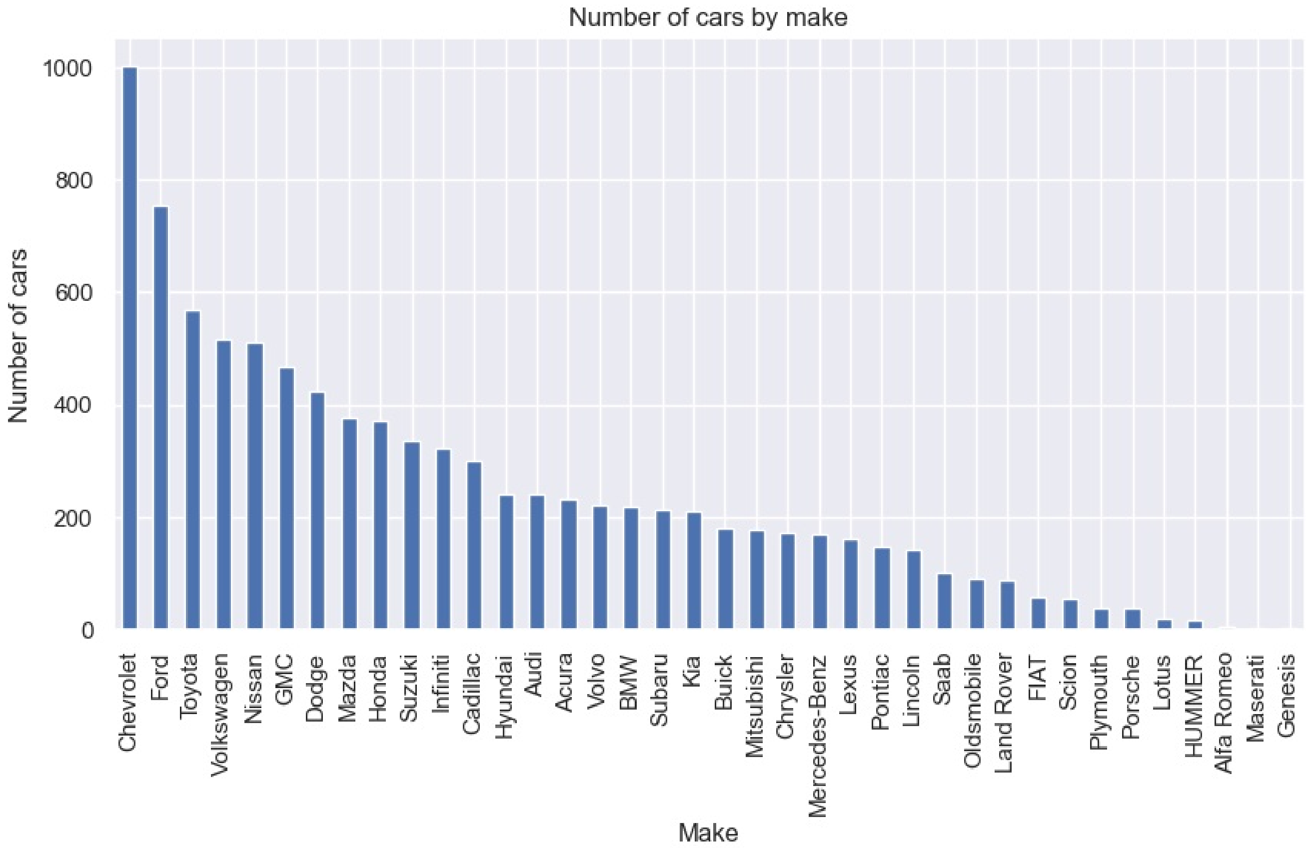
```
df.head(2)
```

Out[277…

	Make	Model	Year	HP	Cylinders	Transmission	Drive Mode	MPG-H	MPG-C	Price
0	BMW	1 Series M	2011	335.0	6.0	MANUAL	rear wheel drive	26	19	46135
1	BMW	1 Series	2011	300.0	6.0	MANUAL	rear wheel drive	28	19	40650

In [279…

```
df.Make.value_counts().nlargest(40).plot(kind='bar', figsize=(10,5))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make');
```



Heat Maps

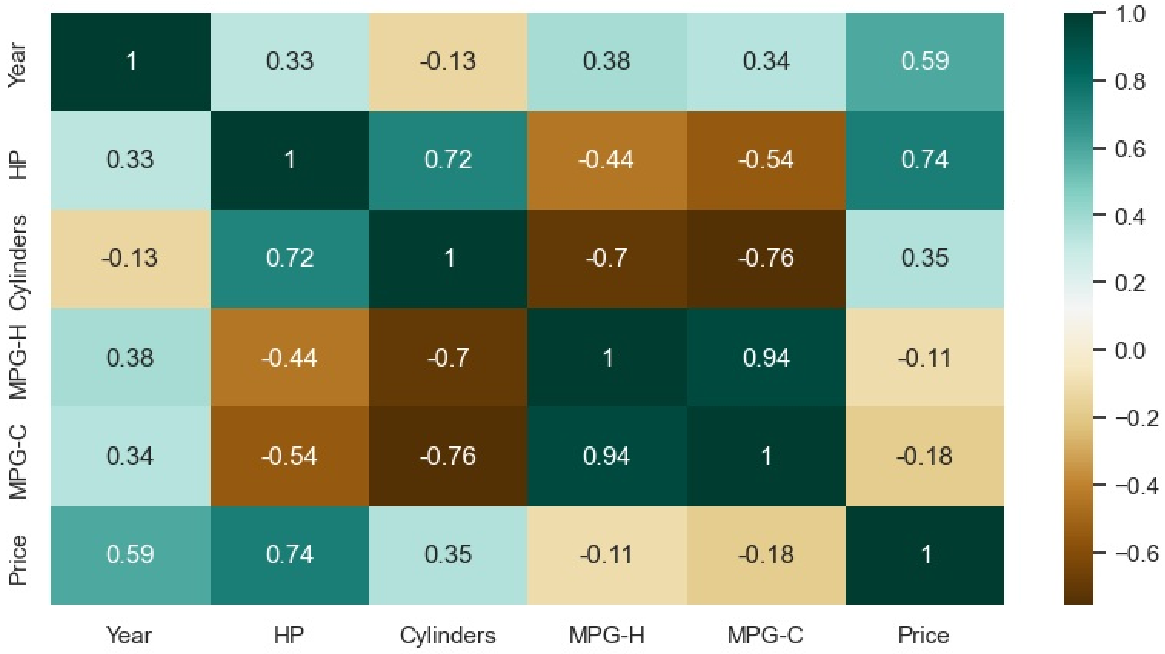
Heat Maps is a type of plot which is necessary when we need to find the dependent variables. One of the best way to find the relationship between the features can be done using heat maps. In the below heat map we know that the price feature depends mainly on the Engine Size, Horsepower, and Cylinders.

In [292…

```
plt.figure(figsize=(10,5))
c= df.select_dtypes(include='number').corr()
sns.heatmap(c,cmap="BrBG",annot=True)
c
```

Out[292…

	Year	HP	Cylinders	MPG-H	MPG-C	Price
Year	1.000000	0.326726	-0.133920	0.378479	0.338145	0.592983
HP	0.326726	1.000000	0.715237	-0.443807	-0.544551	0.739042
Cylinders	-0.133920	0.715237	1.000000	-0.703856	-0.755540	0.354013
MPG-H	0.378479	-0.443807	-0.703856	1.000000	0.939141	-0.106320
MPG-C	0.338145	-0.544551	-0.755540	0.939141	1.000000	-0.180515
Price	0.592983	0.739042	0.354013	-0.106320	-0.180515	1.000000

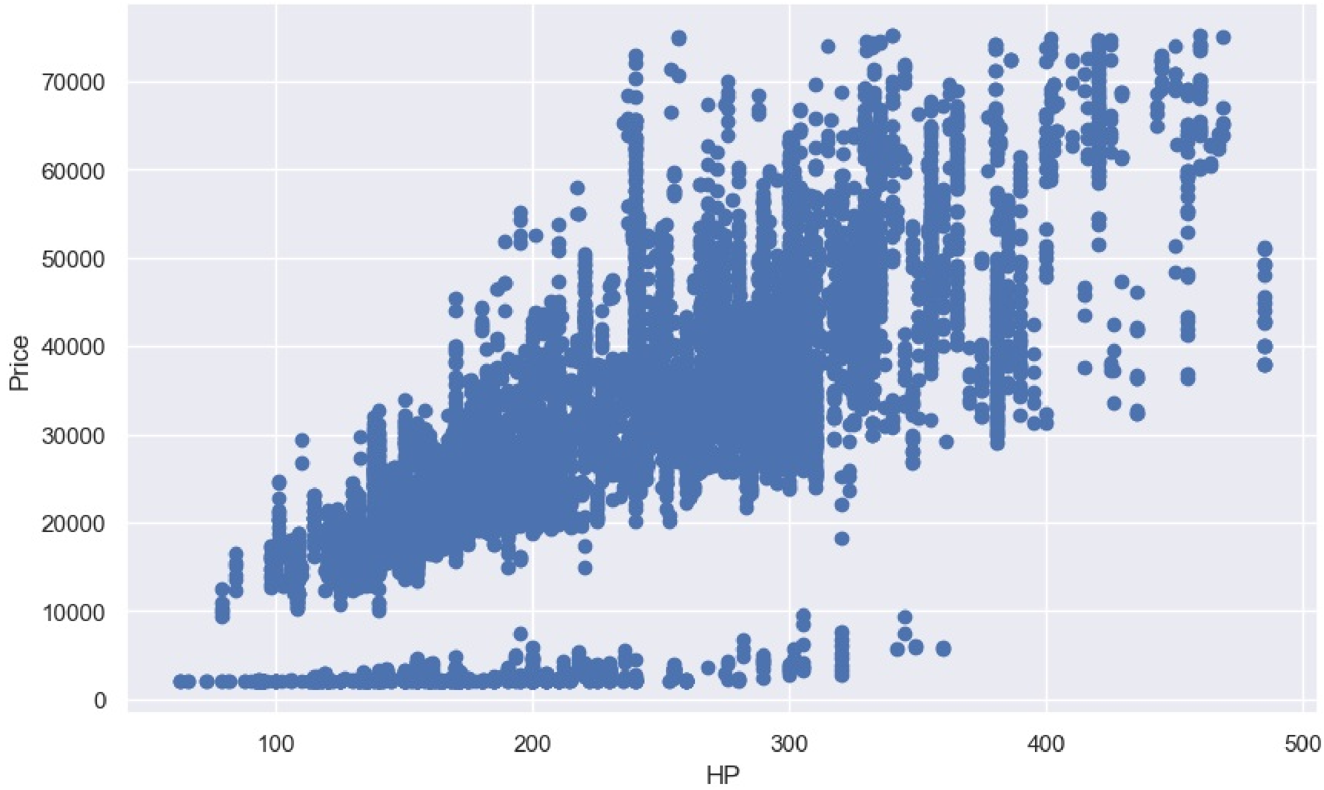


Scatterplot¶

We generally use scatter plots to find the correlation between two variables. Here the scatter plots are plotted between Horsepower and Price and we can see the plot below. With the plot given below, we can easily draw a trend line. These features provide a good scattering of points.

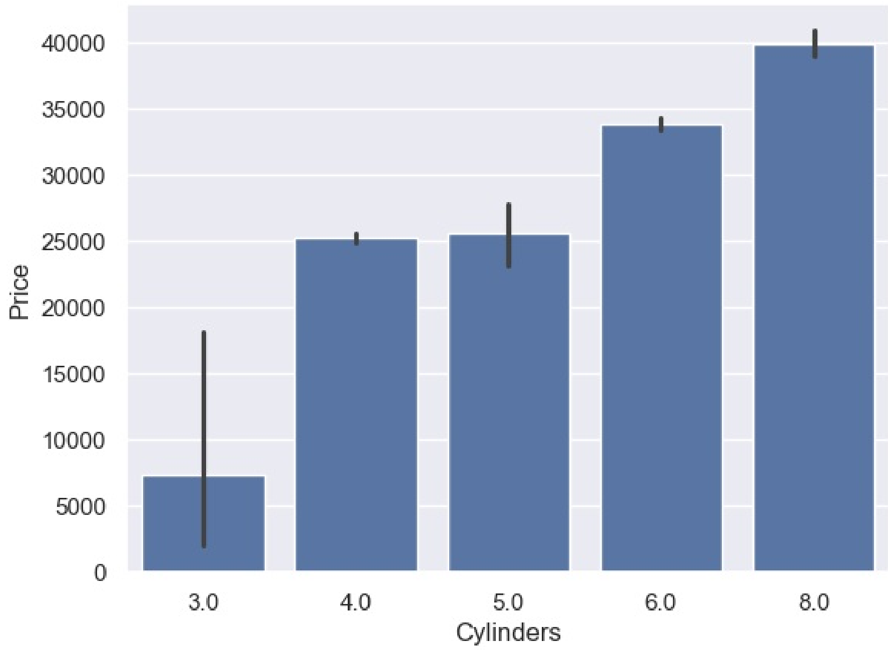
In [295…

```
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(df['HP'], df['Price'])
ax.set_xlabel('HP')
ax.set_ylabel('Price')
plt.show()
```



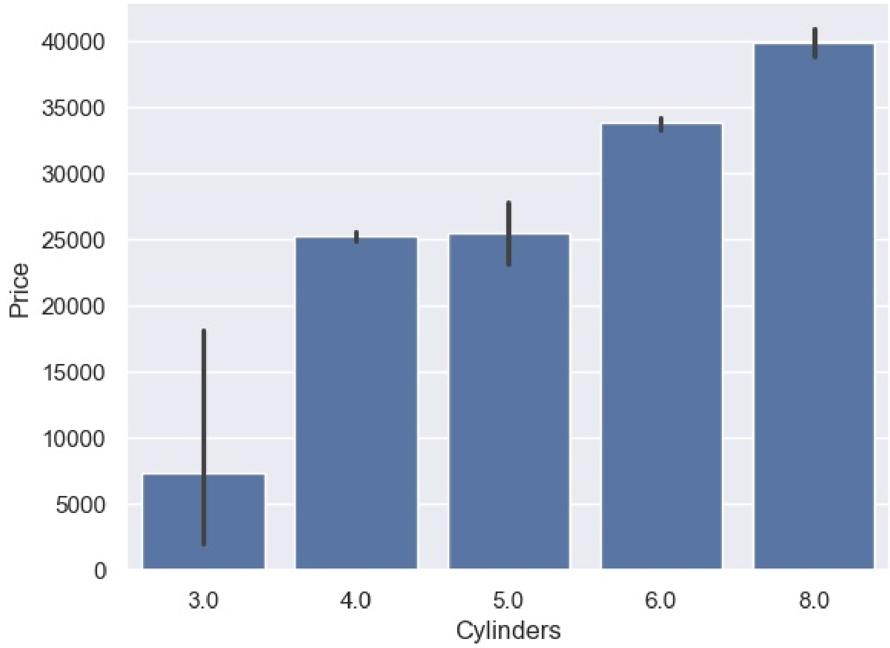
In [314…

```
#fig, ax = plt.subplots(figsize=(10,6))
sns.barplot(data=df, x=df['Cylinders'], y=df['Price'])
plt.xlabel('Cylinders')
plt.ylabel('Price')
plt.show()
```

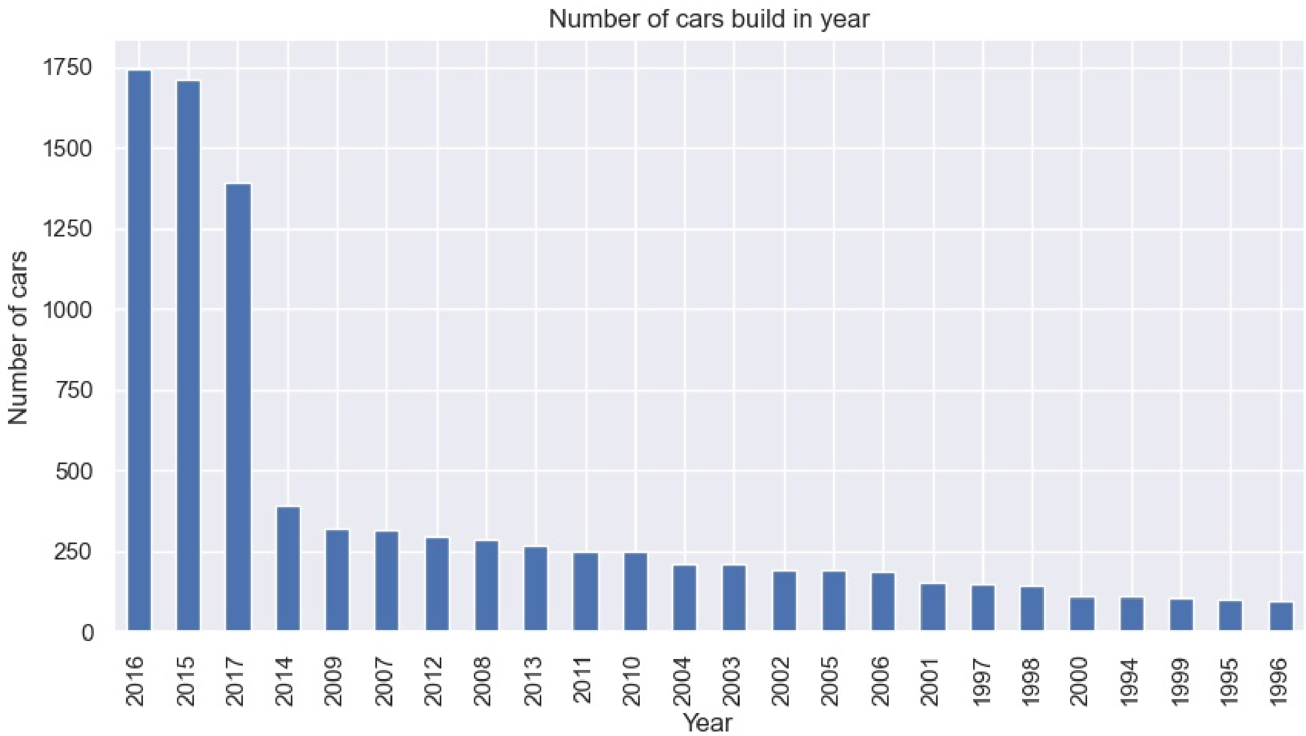


In [417…

```
#fig, ax = plt.subplots(figsize=(10,6))
sns.barplot(data=df, x=df['Cylinders'], y=df['Price'])
plt.xlabel('Cylinders')
plt.ylabel('Price')
plt.show()
```



In [429…
```
df.Year.value_counts().nlargest(40).plot(kind='bar', figsize=(10,5))
plt.title("Number of cars build in year")
plt.ylabel('Number of cars')
plt.xlabel('Year');
```



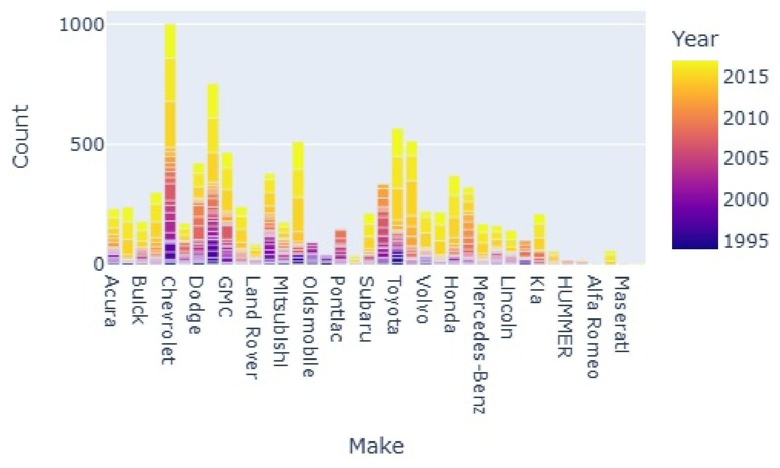
In [5]:
```
import plotly.express as px

# Group by both 'Year' and 'Make' to get counts of each make per year
year_make_counts = df.groupby(['Year', 'Make']).size().reset_index(name='Count')

# Create a bar plot, coloring by 'Make'
fig = px.bar(year_make_counts, x='Make', y='Count', color='Year')
fig.show()

from PIL import Image
Image.open('newplot (4).png')
```

Out[5]:



1. Price Distribution by Model

In [9]:

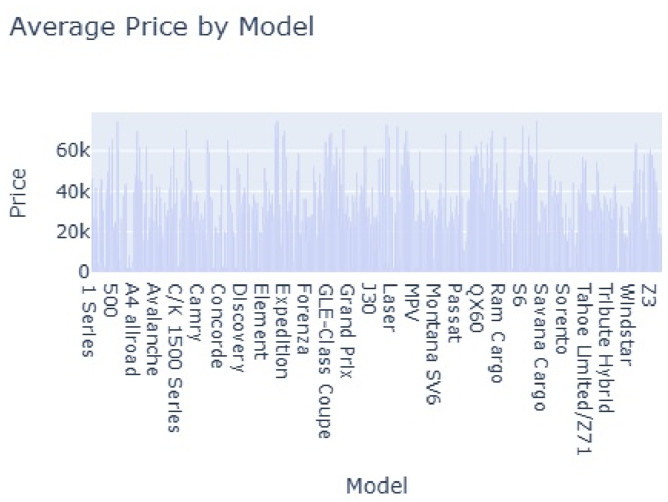
```
import plotly.express as px

# Group by Model and calculate average price
price_by_model = df.groupby('Model')['Price'].mean().reset_index()

# Create bar chart
fig = px.bar(price_by_model, x='Model', y='Price', title='Average Price by Model')
fig.show()

Image.open('newplot (3).png')
```

Out[9]:

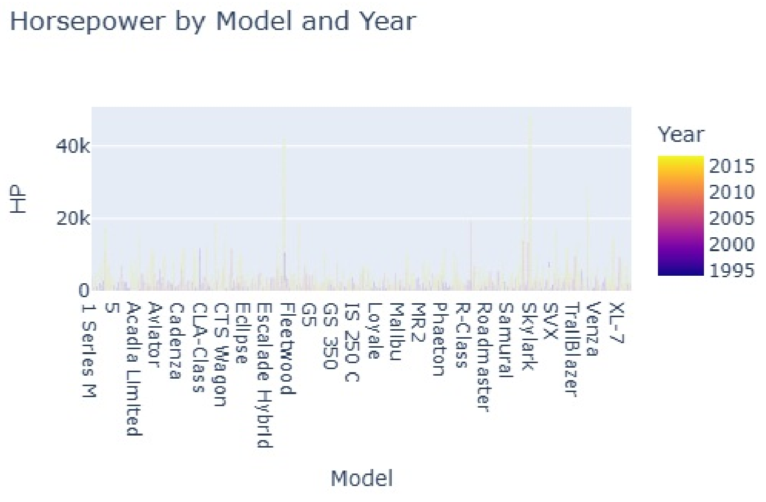


Horsepower (HP) by Model and Year

In [13]:

```
# Bar chart to show HP by Model and Year
fig = px.bar(df, x='Model', y='HP', color='Year', title='Horsepower by Model and Year')
fig.show()
Image.open('newplot (2).png')
```

Out[13]:

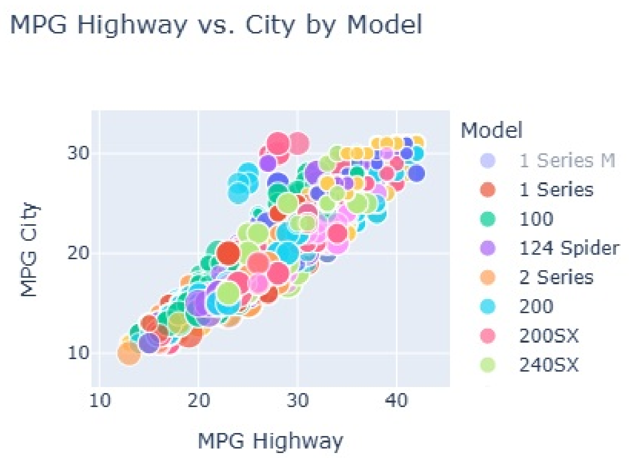


MPG Highway vs. City by Model

In [15]:

```
# Scatter plot for MPG highway vs. city
fig = px.scatter(df, x='MPG-H', y='MPG-C', color='Model', size='HP',
                 title='MPG Highway vs. City by Model', labels={'MPG-H': 'MPG Highway', 'MPG-C': 'MPG City'})
fig.show()
Image.open('newplot (1).png')
```

Out[15]:



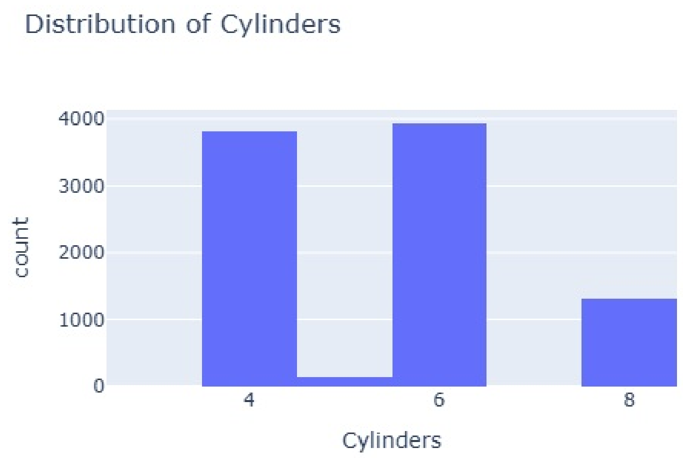
Cylinder Count Distribution

In [19]:

```
# Histogram of cylinder counts
fig = px.histogram(df, x='Cylinders', title='Distribution of Cylinders')
fig.show()

Image.open('newplot(5).png')
```

Out[19]:



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js