Car Analysis

October 22, 2019

1 Cars features and price dataset

This is an analysis of the Car features and price dataset from Kaggle. My aims for the dataset are the following - Clean the dataset - Conduct Univariate and Bivariate analysis to understand what factors have an effect on pricing and fuel economy - Use Machine Learning algorithms to find - Predictive features for Fuel Economy - Regression model to predict MSRP

1.1 Import the necessary libraries and data file and perform initial data assessment

```
In [1]: # Import libraries necessary for this project
        import numpy as np
        import pandas as pd
        from time import time
        from IPython.display import display # Allows the use of display() for DataFrames
        import matplotlib.pyplot as plt
        import pandas_profiling
        import seaborn as sns
        # Pretty display for notebooks
        %matplotlib inline
        import visuals as vs


        # Load the Census dataset
        data = pd.read_csv("data.csv")
```

```
In [2]: #create a profile report
        data.profile_report(style={'full_width':True})
```

```
<IPython.core.display.HTML object>
```

```
Out[2]:
```

I really like using the profiling function, because it gives a lot of information in one like of code, and can help with the data wrangling and univariate analysis

Based on the report we can see - We have missing values in the dataset - We have duplicated values - Highway MPG and City MPG are highly correlated

Based on this information, we can begin the data wrangling process

1.2 Data Wrangling

In this step I will clean up the data 1. Deal with the null values 2. Deal with the outliers

In [3]:
```
#Check if the dataset has any null values
data.isnull().sum()
```

Out[3]:
```
Make                    0
Model                   0
Year                    0
Engine_Fuel_Type        3
Engine_HP              69
Engine_Cylinders       30
Transmission_Type       0
Driven_Wheels           0
Number_of_Doors         6
Market_Category      3742
Vehicle_Size            0
Vehicle_Style           0
highway_MPG             0
city_mpg                0
Popularity              0
MSRP                    0
dtype: int64
```

In [4]:
```
#Check if there is anything specific that stands out about the null values in the Mark
data[data['Market_Category'].isnull()]
```

Out[4]:
```
           Make   Model  Year            Engine_Fuel_Type  Engine_HP  \
87       Nissan   200SX  1996            regular unleaded      115.0
88       Nissan   200SX  1996            regular unleaded      115.0
91       Nissan   200SX  1997            regular unleaded      115.0
92       Nissan   200SX  1997            regular unleaded      115.0
93       Nissan   200SX  1998            regular unleaded      115.0
94       Nissan   200SX  1998            regular unleaded      115.0
203    Chrysler     300  2015            regular unleaded      300.0
204    Chrysler     300  2015            regular unleaded      292.0
205    Chrysler     300  2015            regular unleaded      292.0
206    Chrysler     300  2015            regular unleaded      292.0
209    Chrysler     300  2015            regular unleaded      292.0
210    Chrysler     300  2015            regular unleaded      292.0
211    Chrysler     300  2016            regular unleaded      300.0
213    Chrysler     300  2016            regular unleaded      292.0
214    Chrysler     300  2016            regular unleaded      292.0
215    Chrysler     300  2016            regular unleaded      292.0
216    Chrysler     300  2016            regular unleaded      292.0
219    Chrysler     300  2016            regular unleaded      292.0
220    Chrysler     300  2016            regular unleaded      292.0
221    Chrysler     300  2016            regular unleaded      300.0
```

```
222    Chrysler       300  2016                  regular unleaded      292.0
223    Chrysler       300  2017                  regular unleaded      292.0
224    Chrysler       300  2017                  regular unleaded      292.0
225    Chrysler       300  2017                  regular unleaded      300.0
228    Chrysler       300  2017                  regular unleaded      300.0
229    Chrysler       300  2017                  regular unleaded      292.0
231    Chrysler       300  2017                  regular unleaded      292.0
360       Mazda         3  2015                  regular unleaded      155.0
361       Mazda         3  2015                  regular unleaded      155.0
362       Mazda         3  2015                  regular unleaded      155.0
...         ...       ...   ...                               ...        ...
11686    Suzuki      XL-7  2006                  regular unleaded      185.0
11687    Suzuki      XL-7  2006                  regular unleaded      185.0
11744    Nissan    Xterra  2013                  regular unleaded      261.0
11745    Nissan    Xterra  2013                  regular unleaded      261.0
11746    Nissan    Xterra  2013                  regular unleaded      261.0
11747    Nissan    Xterra  2013                  regular unleaded      261.0
11748    Nissan    Xterra  2013                  regular unleaded      261.0
11749    Nissan    Xterra  2013                  regular unleaded      261.0
11750    Nissan    Xterra  2013                  regular unleaded      261.0
11751    Nissan    Xterra  2014                  regular unleaded      261.0
11752    Nissan    Xterra  2014                  regular unleaded      261.0
11753    Nissan    Xterra  2014                  regular unleaded      261.0
11754    Nissan    Xterra  2014                  regular unleaded      261.0
11755    Nissan    Xterra  2014                  regular unleaded      261.0
11756    Nissan    Xterra  2014                  regular unleaded      261.0
11757    Nissan    Xterra  2014                  regular unleaded      261.0
11758    Nissan    Xterra  2015                  regular unleaded      261.0
11759    Nissan    Xterra  2015                  regular unleaded      261.0
11760    Nissan    Xterra  2015                  regular unleaded      261.0
11761    Nissan    Xterra  2015                  regular unleaded      261.0
11762    Nissan    Xterra  2015                  regular unleaded      261.0
11763    Nissan    Xterra  2015                  regular unleaded      261.0
11764    Nissan    Xterra  2015                  regular unleaded      261.0
11792    Subaru        XT  1991                  regular unleaded       97.0
11793    Subaru        XT  1991                  regular unleaded      145.0
11794    Subaru        XT  1991                  regular unleaded      145.0
11809    Toyota  Yaris iA  2017                  regular unleaded      106.0
11810    Toyota  Yaris iA  2017                  regular unleaded      106.0
11867       GMC     Yukon  2015  premium unleaded (recommended)      420.0
11868       GMC     Yukon  2015  premium unleaded (recommended)      420.0

       Engine_Cylinders Transmission_Type      Driven_Wheels  Number_of_Doors  \
87                  4.0            MANUAL  front wheel drive              2.0
88                  4.0            MANUAL  front wheel drive              2.0
91                  4.0            MANUAL  front wheel drive              2.0
92                  4.0            MANUAL  front wheel drive              2.0
93                  4.0            MANUAL  front wheel drive              2.0
```

94	4.0	MANUAL	front wheel drive	2.0
203	6.0	AUTOMATIC	all wheel drive	4.0
204	6.0	AUTOMATIC	rear wheel drive	4.0
205	6.0	AUTOMATIC	rear wheel drive	4.0
206	6.0	AUTOMATIC	all wheel drive	4.0
209	6.0	AUTOMATIC	all wheel drive	4.0
210	6.0	AUTOMATIC	all wheel drive	4.0
211	6.0	AUTOMATIC	all wheel drive	4.0
213	6.0	AUTOMATIC	all wheel drive	4.0
214	6.0	AUTOMATIC	rear wheel drive	4.0
215	6.0	AUTOMATIC	all wheel drive	4.0
216	6.0	AUTOMATIC	all wheel drive	4.0
219	6.0	AUTOMATIC	rear wheel drive	4.0
220	6.0	AUTOMATIC	rear wheel drive	4.0
221	6.0	AUTOMATIC	all wheel drive	4.0
222	6.0	AUTOMATIC	all wheel drive	4.0
223	6.0	AUTOMATIC	all wheel drive	4.0
224	6.0	AUTOMATIC	all wheel drive	4.0
225	6.0	AUTOMATIC	all wheel drive	4.0
228	6.0	AUTOMATIC	all wheel drive	4.0
229	6.0	AUTOMATIC	rear wheel drive	4.0
231	6.0	AUTOMATIC	all wheel drive	4.0
360	4.0	AUTOMATIC	front wheel drive	4.0
361	4.0	MANUAL	front wheel drive	4.0
362	4.0	MANUAL	front wheel drive	4.0
...	...	...	...	...
11686	6.0	AUTOMATIC	rear wheel drive	4.0
11687	6.0	AUTOMATIC	rear wheel drive	4.0
11744	6.0	AUTOMATIC	four wheel drive	4.0
11745	6.0	MANUAL	four wheel drive	4.0
11746	6.0	MANUAL	four wheel drive	4.0
11747	6.0	AUTOMATIC	four wheel drive	4.0
11748	6.0	AUTOMATIC	rear wheel drive	4.0
11749	6.0	AUTOMATIC	four wheel drive	4.0
11750	6.0	AUTOMATIC	rear wheel drive	4.0
11751	6.0	AUTOMATIC	four wheel drive	4.0
11752	6.0	AUTOMATIC	rear wheel drive	4.0
11753	6.0	AUTOMATIC	four wheel drive	4.0
11754	6.0	AUTOMATIC	four wheel drive	4.0
11755	6.0	MANUAL	four wheel drive	4.0
11756	6.0	AUTOMATIC	rear wheel drive	4.0
11757	6.0	MANUAL	four wheel drive	4.0
11758	6.0	MANUAL	four wheel drive	4.0
11759	6.0	AUTOMATIC	four wheel drive	4.0
11760	6.0	AUTOMATIC	four wheel drive	4.0
11761	6.0	MANUAL	four wheel drive	4.0
11762	6.0	AUTOMATIC	rear wheel drive	4.0
11763	6.0	AUTOMATIC	four wheel drive	4.0

```
11764              6.0         AUTOMATIC   rear wheel drive              4.0
11792              4.0            MANUAL  front wheel drive              2.0
11793              6.0         AUTOMATIC  front wheel drive              2.0
11794              6.0            MANUAL    all wheel drive              2.0
11809              4.0            MANUAL  front wheel drive              4.0
11810              4.0         AUTOMATIC  front wheel drive              4.0
11867              8.0         AUTOMATIC   rear wheel drive              4.0
11868              8.0         AUTOMATIC   four wheel drive              4.0

      Market_Category Vehicle_Size Vehicle_Style  highway_MPG  city_mpg  \
87                NaN      Compact         Coupe           36        26
88                NaN      Compact         Coupe           36        26
91                NaN      Compact         Coupe           35        25
92                NaN      Compact         Coupe           35        25
93                NaN      Compact         Coupe           35        25
94                NaN      Compact         Coupe           35        25
203               NaN        Large         Sedan           27        18
204               NaN        Large         Sedan           31        19
205               NaN        Large         Sedan           31        19
206               NaN        Large         Sedan           27        18
209               NaN        Large         Sedan           27        18
210               NaN        Large         Sedan           27        18
211               NaN        Large         Sedan           27        18
213               NaN        Large         Sedan           27        18
214               NaN        Large         Sedan           31        19
215               NaN        Large         Sedan           27        18
216               NaN        Large         Sedan           27        18
219               NaN        Large         Sedan           31        19
220               NaN        Large         Sedan           31        19
221               NaN        Large         Sedan           27        18
222               NaN        Large         Sedan           27        18
223               NaN        Large         Sedan           27        18
224               NaN        Large         Sedan           27        18
225               NaN        Large         Sedan           27        18
228               NaN        Large         Sedan           27        18
229               NaN        Large         Sedan           30        19
231               NaN        Large         Sedan           27        18
360               NaN      Compact         Sedan           41        30
361               NaN      Compact         Sedan           41        29
362               NaN      Compact         Sedan           41        29
...               ...          ...           ...          ...       ...
11686             NaN      Midsize       4dr SUV           21        16
11687             NaN      Midsize       4dr SUV           21        16
11744             NaN      Midsize       4dr SUV           20        15
11745             NaN      Midsize       4dr SUV           20        16
11746             NaN      Midsize       4dr SUV           20        16
11747             NaN      Midsize       4dr SUV           20        15
11748             NaN      Midsize       4dr SUV           22        16
```

11749	NaN	Midsize	4dr SUV	20	15
11750	NaN	Midsize	4dr SUV	22	16
11751	NaN	Midsize	4dr SUV	20	15
11752	NaN	Midsize	4dr SUV	22	16
11753	NaN	Midsize	4dr SUV	20	15
11754	NaN	Midsize	4dr SUV	20	15
11755	NaN	Midsize	4dr SUV	20	16
11756	NaN	Midsize	4dr SUV	22	16
11757	NaN	Midsize	4dr SUV	20	16
11758	NaN	Midsize	4dr SUV	20	15
11759	NaN	Midsize	4dr SUV	20	15
11760	NaN	Midsize	4dr SUV	20	15
11761	NaN	Midsize	4dr SUV	20	15
11762	NaN	Midsize	4dr SUV	22	16
11763	NaN	Midsize	4dr SUV	20	15
11764	NaN	Midsize	4dr SUV	22	16
11792	NaN	Compact	Coupe	29	22
11793	NaN	Compact	Coupe	26	18
11794	NaN	Compact	Coupe	23	16
11809	NaN	Compact	Sedan	39	30
11810	NaN	Compact	Sedan	40	32
11867	NaN	Large	4dr SUV	21	15
11868	NaN	Large	4dr SUV	21	14

	Popularity	MSRP
87	2009	2000
88	2009	2000
91	2009	2000
92	2009	2000
93	2009	2000
94	2009	2000
203	1013	37570
204	1013	31695
205	1013	38070
206	1013	44895
209	1013	34195
210	1013	40570
211	1013	38095
213	1013	45190
214	1013	32260
215	1013	37755
216	1013	41055
219	1013	38555
220	1013	35255
221	1013	38590
222	1013	34760
223	1013	41135
224	1013	45270

```
225            1013  38670
228            1013  38175
229            1013  32340
231            1013  34840
360             586  23795
361             586  19595
362             586  18445
...             ...    ...
11686           481  25499
11687           481  21999
11744          2009  26900
11745          2009  29440
11746          2009  25850
11747          2009  30490
11748          2009  24850
11749          2009  24990
11750          2009  22940
11751          2009  31370
11752          2009  25300
11753          2009  27350
11754          2009  25440
11755          2009  26300
11756          2009  23390
11757          2009  30320
11758          2009  26670
11759          2009  27720
11760          2009  25710
11761          2009  30590
11762          2009  23660
11763          2009  31640
11764          2009  25670
11792           640   2000
11793           640   2000
11794           640   2000
11809          2031  15950
11810          2031  17050
11867           549  64520
11868           549  67520

[3742 rows x 16 columns]
```

From the table above it seems that there is noting specific about the Nan's in the Market Category Column. Since, market category is not an independent characteristic (depends on other factors, like the make, model, style etc). Therefore, for downstream analysis, we can drop this column.

In [5]:
```
#drop the market category column
data = data.drop(['Market_Category'], axis=1)
data.isnull().sum()
```

```
Out[5]: Make                  0
        Model                 0
        Year                  0
        Engine_Fuel_Type      3
        Engine_HP            69
        Engine_Cylinders     30
        Transmission_Type     0
        Driven_Wheels         0
        Number_of_Doors       6
        Vehicle_Size          0
        Vehicle_Style         0
        highway_MPG           0
        city_mpg              0
        Popularity            0
        MSRP                  0
        dtype: int64
```

1.2.1 Null Values

The two major sources of null values are horsepower and engine cylinders. We will look at both of them and see if we can add the missing values to these or would we have to drop them

Horsepower

```
In [6]: #Create a dataframe to further investigate the null values of Engine HP
        df_temp = data[data['Engine_HP'].isnull()]
```

```
In [7]: df_temp
```

```
Out[7]:            Make        Model  Year               Engine_Fuel_Type  \
        539        FIAT         500e  2015                       electric
        540        FIAT         500e  2016                       electric
        541        FIAT         500e  2017                       electric
        2905    Lincoln  Continental  2017   premium unleaded (recommended)
        2906    Lincoln  Continental  2017   premium unleaded (recommended)
        2907    Lincoln  Continental  2017   premium unleaded (recommended)
        2908    Lincoln  Continental  2017   premium unleaded (recommended)
        4203       Ford       Escape  2017               regular unleaded
        4204       Ford       Escape  2017               regular unleaded
        4205       Ford       Escape  2017               regular unleaded
        4206       Ford       Escape  2017               regular unleaded
        4705      Honda       Fit EV  2013                       electric
        4706      Honda       Fit EV  2014                       electric
        4785       Ford        Focus  2015                       electric
        4789       Ford        Focus  2016                       electric
        4798       Ford        Focus  2017                       electric
        4914       Ford     Freestar  2005               regular unleaded
        4915       Ford     Freestar  2005               regular unleaded
```

```
4916           Ford     Freestar  2005                   regular unleaded
4917           Ford     Freestar  2005                   regular unleaded
4918           Ford     Freestar  2005                   regular unleaded
4919           Ford     Freestar  2005                   regular unleaded
5778     Mitsubishi       i-MiEV  2014                           electric
5825      Chevrolet       Impala  2015  flex-fuel (unleaded/natural gas)
5830      Chevrolet       Impala  2015  flex-fuel (unleaded/natural gas)
5831      Chevrolet       Impala  2016  flex-fuel (unleaded/natural gas)
5833      Chevrolet       Impala  2016  flex-fuel (unleaded/natural gas)
5839      Chevrolet       Impala  2017  flex-fuel (unleaded/natural gas)
5840      Chevrolet       Impala  2017  flex-fuel (unleaded/natural gas)
6385         Nissan         Leaf  2014                           electric
...             ...          ...   ...                                ...
6578  Mercedes-Benz      M-Class  2015                             diesel
6908        Lincoln          MKZ  2017                   regular unleaded
6910        Lincoln          MKZ  2017                   regular unleaded
6916        Lincoln          MKZ  2017                   regular unleaded
6918        Lincoln          MKZ  2017                   regular unleaded
6921          Tesla      Model S  2014                           electric
6922          Tesla      Model S  2014                           electric
6923          Tesla      Model S  2014                           electric
6924          Tesla      Model S  2014                           electric
6925          Tesla      Model S  2015                           electric
6926          Tesla      Model S  2015                           electric
6927          Tesla      Model S  2015                           electric
6928          Tesla      Model S  2015                           electric
6929          Tesla      Model S  2015                           electric
6930          Tesla      Model S  2016                           electric
6931          Tesla      Model S  2016                           electric
6932          Tesla      Model S  2016                           electric
6933          Tesla      Model S  2016                           electric
6934          Tesla      Model S  2016                           electric
6935          Tesla      Model S  2016                           electric
6936          Tesla      Model S  2016                           electric
6937          Tesla      Model S  2016                           electric
6938          Tesla      Model S  2016                           electric
8374         Toyota      RAV4 EV  2013                           electric
8375         Toyota      RAV4 EV  2014                           electric
9850            Kia      Soul EV  2015                           electric
9851            Kia      Soul EV  2015                           electric
9852            Kia      Soul EV  2016                           electric
9853            Kia      Soul EV  2016                           electric
9854            Kia      Soul EV  2016                           electric

      Engine_HP  Engine_Cylinders Transmission_Type      Driven_Wheels  \
539         NaN               0.0      DIRECT_DRIVE  front wheel drive
540         NaN               0.0      DIRECT_DRIVE  front wheel drive
541         NaN               0.0      DIRECT_DRIVE  front wheel drive
```

2905	NaN	6.0	AUTOMATIC	all wheel drive
2906	NaN	6.0	AUTOMATIC	front wheel drive
2907	NaN	6.0	AUTOMATIC	front wheel drive
2908	NaN	6.0	AUTOMATIC	all wheel drive
4203	NaN	4.0	AUTOMATIC	front wheel drive
4204	NaN	4.0	AUTOMATIC	all wheel drive
4205	NaN	4.0	AUTOMATIC	all wheel drive
4206	NaN	4.0	AUTOMATIC	front wheel drive
4705	NaN	0.0	DIRECT_DRIVE	front wheel drive
4706	NaN	0.0	DIRECT_DRIVE	front wheel drive
4785	NaN	0.0	DIRECT_DRIVE	front wheel drive
4789	NaN	0.0	DIRECT_DRIVE	front wheel drive
4798	NaN	0.0	DIRECT_DRIVE	front wheel drive
4914	NaN	6.0	AUTOMATIC	front wheel drive
4915	NaN	6.0	AUTOMATIC	front wheel drive
4916	NaN	6.0	AUTOMATIC	front wheel drive
4917	NaN	6.0	AUTOMATIC	front wheel drive
4918	NaN	6.0	AUTOMATIC	front wheel drive
4919	NaN	6.0	AUTOMATIC	front wheel drive
5778	NaN	NaN	DIRECT_DRIVE	rear wheel drive
5825	NaN	6.0	AUTOMATIC	front wheel drive
5830	NaN	6.0	AUTOMATIC	front wheel drive
5831	NaN	6.0	AUTOMATIC	front wheel drive
5833	NaN	6.0	AUTOMATIC	front wheel drive
5839	NaN	6.0	AUTOMATIC	front wheel drive
5840	NaN	6.0	AUTOMATIC	front wheel drive
6385	NaN	0.0	DIRECT_DRIVE	front wheel drive
...	...	...	...	...
6578	NaN	4.0	AUTOMATIC	all wheel drive
6908	NaN	4.0	AUTOMATIC	front wheel drive
6910	NaN	4.0	AUTOMATIC	front wheel drive
6916	NaN	4.0	AUTOMATIC	front wheel drive
6918	NaN	4.0	AUTOMATIC	front wheel drive
6921	NaN	0.0	DIRECT_DRIVE	rear wheel drive
6922	NaN	0.0	DIRECT_DRIVE	rear wheel drive
6923	NaN	0.0	DIRECT_DRIVE	all wheel drive
6924	NaN	0.0	DIRECT_DRIVE	rear wheel drive
6925	NaN	0.0	DIRECT_DRIVE	rear wheel drive
6926	NaN	0.0	DIRECT_DRIVE	all wheel drive
6927	NaN	0.0	DIRECT_DRIVE	all wheel drive
6928	NaN	0.0	DIRECT_DRIVE	all wheel drive
6929	NaN	0.0	DIRECT_DRIVE	rear wheel drive
6930	NaN	0.0	DIRECT_DRIVE	all wheel drive
6931	NaN	0.0	DIRECT_DRIVE	all wheel drive
6932	NaN	0.0	DIRECT_DRIVE	all wheel drive
6933	NaN	0.0	DIRECT_DRIVE	rear wheel drive
6934	NaN	0.0	DIRECT_DRIVE	all wheel drive
6935	NaN	0.0	DIRECT_DRIVE	all wheel drive

6936	NaN	0.0	DIRECT_DRIVE	all wheel drive
6937	NaN	0.0	DIRECT_DRIVE	all wheel drive
6938	NaN	0.0	DIRECT_DRIVE	rear wheel drive
8374	NaN	0.0	DIRECT_DRIVE	front wheel drive
8375	NaN	0.0	DIRECT_DRIVE	front wheel drive
9850	NaN	0.0	DIRECT_DRIVE	front wheel drive
9851	NaN	0.0	DIRECT_DRIVE	front wheel drive
9852	NaN	0.0	DIRECT_DRIVE	front wheel drive
9853	NaN	0.0	DIRECT_DRIVE	front wheel drive
9854	NaN	0.0	DIRECT_DRIVE	front wheel drive

	Number_of_Doors	Vehicle_Size	Vehicle_Style	highway_MPG	city_mpg	\
539	2.0	Compact	2dr Hatchback	108	122	
540	2.0	Compact	2dr Hatchback	103	121	
541	2.0	Compact	2dr Hatchback	103	121	
2905	4.0	Large	Sedan	25	17	
2906	4.0	Large	Sedan	27	18	
2907	4.0	Large	Sedan	27	18	
2908	4.0	Large	Sedan	25	17	
4203	4.0	Compact	4dr SUV	30	23	
4204	4.0	Compact	4dr SUV	28	22	
4205	4.0	Compact	4dr SUV	28	22	
4206	4.0	Compact	4dr SUV	30	23	
4705	4.0	Compact	4dr Hatchback	105	132	
4706	4.0	Compact	4dr Hatchback	105	132	
4785	4.0	Compact	4dr Hatchback	99	110	
4789	4.0	Compact	4dr Hatchback	99	110	
4798	4.0	Compact	4dr Hatchback	99	110	
4914	4.0	Midsize	Passenger Minivan	22	16	
4915	4.0	Midsize	Passenger Minivan	22	16	
4916	4.0	Midsize	Cargo Minivan	22	16	
4917	4.0	Midsize	Passenger Minivan	22	16	
4918	4.0	Midsize	Passenger Minivan	21	16	
4919	4.0	Midsize	Passenger Minivan	21	16	
5778	4.0	Compact	4dr Hatchback	99	126	
5825	4.0	Large	Sedan	25	17	
5830	4.0	Large	Sedan	25	17	
5831	4.0	Large	Sedan	25	17	
5833	4.0	Large	Sedan	25	17	
5839	4.0	Large	Sedan	25	17	
5840	4.0	Large	Sedan	25	17	
6385	4.0	Compact	4dr Hatchback	101	126	
...	...	...	...	...	...	
6578	4.0	Midsize	4dr SUV	29	22	
6908	4.0	Midsize	Sedan	38	41	
6910	4.0	Midsize	Sedan	38	41	
6916	4.0	Midsize	Sedan	38	41	
6918	4.0	Midsize	Sedan	38	41	

6921	4.0	Large	Sedan	90	88
6922	4.0	Large	Sedan	97	94
6923	4.0	Large	Sedan	94	86
6924	4.0	Large	Sedan	90	88
6925	4.0	Large	Sedan	97	94
6926	4.0	Large	Sedan	102	101
6927	4.0	Large	Sedan	106	95
6928	4.0	Large	Sedan	98	89
6929	4.0	Large	Sedan	90	88
6930	NaN	Large	Sedan	105	102
6931	NaN	Large	Sedan	101	98
6932	NaN	Large	Sedan	105	92
6933	NaN	Large	Sedan	100	97
6934	NaN	Large	Sedan	107	101
6935	4.0	Large	Sedan	102	101
6936	4.0	Large	Sedan	107	101
6937	4.0	Large	Sedan	100	91
6938	4.0	Large	Sedan	90	88
8374	4.0	Midsize	4dr SUV	74	78
8375	4.0	Midsize	4dr SUV	74	78
9850	4.0	Compact	Wagon	92	120
9851	4.0	Compact	Wagon	92	120
9852	4.0	Compact	Wagon	92	120
9853	4.0	Compact	Wagon	92	120
9854	4.0	Compact	Wagon	92	120

	Popularity	MSRP
539	819	31800
540	819	31800
541	819	31800
2905	61	55915
2906	61	62915
2907	61	53915
2908	61	64915
4203	5657	29100
4204	5657	30850
4205	5657	26850
4206	5657	25100
4705	2202	36625
4706	2202	36625
4785	5657	29170
4789	5657	29170
4798	5657	29120
4914	5657	28030
4915	5657	23930
4916	5657	21630
4917	5657	26530
4918	5657	29030

```
4919         5657   32755
5778          436   22995
5825         1385   40660
5830         1385   37535
5831         1385   40810
5833         1385   37570
5839         1385   37675
5840         1385   40915
6385         2009   35020
...           ...     ...
6578          617   49800
6908           61   35010
6910           61   39510
6916           61   36760
6918           61   47670
6921         1391   79900
6922         1391   69900
6923         1391  104500
6924         1391   93400
6925         1391   69900
6926         1391   75000
6927         1391   85000
6928         1391  105000
6929         1391   80000
6930         1391   79500
6931         1391   66000
6932         1391  134500
6933         1391   74500
6934         1391   71000
6935         1391   75000
6936         1391   89500
6937         1391  112000
6938         1391   70000
8374         2031   49800
8375         2031   49800
9850         1720   35700
9851         1720   33700
9852         1720   33950
9853         1720   31950
9854         1720   35950

[69 rows x 15 columns]
```

There are some specific models of cars that are missing horsepower values. Let's check which models are these

```
In [8]: df_temp['Model'].unique()
```

```
Out[8]: array(['500e', 'Continental', 'Escape', 'Fit EV', 'Focus', 'Freestar',
```

```
'i-MiEV', 'Impala', 'Leaf', 'M-Class', 'MKZ', 'Model S', 'RAV4 EV',
'Soul EV'], dtype=object)
```

One maybe tempted to drop these values, but with a little help from Google, we can find the missing horsepower values and add them to the dataframe

```
In [9]: #First make a new copy of the dataframe to work with
        df = data.copy()
```

```
In [10]: #Here we add values of missing horsepower
         result = []
         for i in df['Model']:
             if i == '500e':
                 result.append(111)
             elif i == 'Continental':
                 result.append(400)
             elif i == 'Escape':
                 result.append(168)
             elif i == 'Fit EV':
                 result.append(123)
             elif i == 'Focus':
                 result.append(143)
             elif i == 'Freestar':
                 result.append(201)
             elif i == 'i-MiEV':
                 result.append(66)
             elif i == 'Impala':
                 result.append(305)
             elif i == 'Leaf':
                 result.append(107)
             elif i == 'M-Class':
                 result.append(201)
             elif i == 'MKZ':
                 result.append(245)
             elif i == 'Model S':
                 result.append(600)
             elif i == 'RAV4 EV':
                 result.append(154)
             elif i == 'Soul EV':
                 result.append(109)
             else:
                 result.append(" ")
         df["Result"] = result
```

```
In [11]: #Here I will replace the missing values with a blank space, so it will be easy to mel
         df["Engine_HP"] = df["Engine_HP"].fillna('')
```

```
In [12]: df.head()
```

```
Out[12]:   Make       Model  Year             Engine_Fuel_Type Engine_HP  \
         0  BMW  1 Series M  2011  premium unleaded (required)       335
         1  BMW    1 Series  2011  premium unleaded (required)       300
         2  BMW    1 Series  2011  premium unleaded (required)       300
         3  BMW    1 Series  2011  premium unleaded (required)       230
         4  BMW    1 Series  2011  premium unleaded (required)       230

            Engine_Cylinders Transmission_Type     Driven_Wheels  Number_of_Doors  \
         0               6.0            MANUAL  rear wheel drive              2.0
         1               6.0            MANUAL  rear wheel drive              2.0
         2               6.0            MANUAL  rear wheel drive              2.0
         3               6.0            MANUAL  rear wheel drive              2.0
         4               6.0            MANUAL  rear wheel drive              2.0

           Vehicle_Size Vehicle_Style  highway_MPG  city_mpg  Popularity   MSRP Result
         0      Compact         Coupe           26        19        3916  46135
         1      Compact   Convertible           28        19        3916  40650
         2      Compact         Coupe           28        20        3916  36350
         3      Compact         Coupe           28        18        3916  29450
         4      Compact   Convertible           28        18        3916  34500
```

Next, we will merge the new and the old horsepower columns to make a single column

```
In [13]: convert_dict = {'Result': str, 'Engine_HP': str}
         df = df.astype(convert_dict)

In [14]: df['Final_HP'] = df["Engine_HP"] + df["Result"]

In [15]: df['Final_HP'] = df['Final_HP'].astype(float)

In [16]: #Drop the old HP column
         df = df.drop(columns=['Engine_HP', 'Result'])

In [17]: #Recheck null values
         df.isnull().sum()

Out[17]: Make                  0
         Model                 0
         Year                  0
         Engine_Fuel_Type      3
         Engine_Cylinders     30
         Transmission_Type     0
         Driven_Wheels         0
         Number_of_Doors       6
         Vehicle_Size          0
         Vehicle_Style         0
         highway_MPG           0
         city_mpg              0
         Popularity            0
```

```
MSRP                   0
Final_HP               0
dtype: int64
```

Engine Cylinders

In [18]:
```
#Make a temp dataframe to explore the missing values in the engine cylinders
df_temp = df[df['Engine_Cylinders'].isnull()]
```

In [19]:
```
df_temp
```

Out[19]:
```
            Make    Model  Year              Engine_Fuel_Type  \
1983   Chevrolet  Bolt EV  2017                      electric
1984   Chevrolet  Bolt EV  2017                      electric
3716  Volkswagen   e-Golf  2015                      electric
3717  Volkswagen   e-Golf  2015                      electric
3718  Volkswagen   e-Golf  2016                      electric
3719  Volkswagen   e-Golf  2016                      electric
5778  Mitsubishi   i-MiEV  2014                      electric
5779  Mitsubishi   i-MiEV  2016                      electric
5780  Mitsubishi   i-MiEV  2017                      electric
8373      Toyota  RAV4 EV  2012                      electric
8695       Mazda     RX-7  1993              regular unleaded
8696       Mazda     RX-7  1994              regular unleaded
8697       Mazda     RX-7  1995              regular unleaded
8698       Mazda     RX-8  2009  premium unleaded (required)
8699       Mazda     RX-8  2009  premium unleaded (required)
8700       Mazda     RX-8  2009  premium unleaded (required)
8701       Mazda     RX-8  2009  premium unleaded (required)
8702       Mazda     RX-8  2009  premium unleaded (required)
8703       Mazda     RX-8  2009  premium unleaded (required)
8704       Mazda     RX-8  2009  premium unleaded (required)
8705       Mazda     RX-8  2010  premium unleaded (required)
8706       Mazda     RX-8  2010  premium unleaded (required)
8707       Mazda     RX-8  2010  premium unleaded (required)
8708       Mazda     RX-8  2010  premium unleaded (required)
8709       Mazda     RX-8  2010  premium unleaded (required)
8710       Mazda     RX-8  2011  premium unleaded (required)
8711       Mazda     RX-8  2011  premium unleaded (required)
8712       Mazda     RX-8  2011  premium unleaded (required)
8713       Mazda     RX-8  2011  premium unleaded (required)
8714       Mazda     RX-8  2011  premium unleaded (required)

      Engine_Cylinders Transmission_Type      Driven_Wheels  Number_of_Doors  \
1983               NaN      DIRECT_DRIVE  front wheel drive              4.0
1984               NaN      DIRECT_DRIVE  front wheel drive              4.0
3716               NaN      DIRECT_DRIVE  front wheel drive              4.0
3717               NaN      DIRECT_DRIVE  front wheel drive              4.0
```

```
3718              NaN      DIRECT_DRIVE  front wheel drive            4.0
3719              NaN      DIRECT_DRIVE  front wheel drive            4.0
5778              NaN      DIRECT_DRIVE   rear wheel drive            4.0
5779              NaN      DIRECT_DRIVE   rear wheel drive            4.0
5780              NaN      DIRECT_DRIVE   rear wheel drive            4.0
8373              NaN      DIRECT_DRIVE  front wheel drive            4.0
8695              NaN            MANUAL   rear wheel drive            2.0
8696              NaN            MANUAL   rear wheel drive            2.0
8697              NaN            MANUAL   rear wheel drive            2.0
8698              NaN            MANUAL   rear wheel drive            4.0
8699              NaN         AUTOMATIC   rear wheel drive            4.0
8700              NaN            MANUAL   rear wheel drive            4.0
8701              NaN            MANUAL   rear wheel drive            4.0
8702              NaN            MANUAL   rear wheel drive            4.0
8703              NaN         AUTOMATIC   rear wheel drive            4.0
8704              NaN         AUTOMATIC   rear wheel drive            4.0
8705              NaN            MANUAL   rear wheel drive            4.0
8706              NaN         AUTOMATIC   rear wheel drive            4.0
8707              NaN         AUTOMATIC   rear wheel drive            4.0
8708              NaN            MANUAL   rear wheel drive            4.0
8709              NaN            MANUAL   rear wheel drive            4.0
8710              NaN         AUTOMATIC   rear wheel drive            4.0
8711              NaN            MANUAL   rear wheel drive            4.0
8712              NaN            MANUAL   rear wheel drive            4.0
8713              NaN            MANUAL   rear wheel drive            4.0
8714              NaN         AUTOMATIC   rear wheel drive            4.0

     Vehicle_Size  Vehicle_Style  highway_MPG  city_mpg  Popularity   MSRP  \
1983      Compact  4dr Hatchback          110       128        1385  40905
1984      Compact  4dr Hatchback          110       128        1385  36620
3716      Compact  4dr Hatchback          105       126         873  33450
3717      Compact  4dr Hatchback          105       126         873  35445
3718      Compact  4dr Hatchback          105       126         873  28995
3719      Compact  4dr Hatchback          105       126         873  35595
5778      Compact  4dr Hatchback           99       126         436  22995
5779      Compact  4dr Hatchback           99       126         436  22995
5780      Compact  4dr Hatchback          102       121         436  22995
8373      Midsize        4dr SUV           74        78        2031  49800
8695      Compact          Coupe           23        15         586   7523
8696      Compact          Coupe           23        15         586   8147
8697      Compact          Coupe           23        15         586   8839
8698      Compact          Coupe           22        16         586  31930
8699      Compact          Coupe           23        16         586  26435
8700      Compact          Coupe           22        16         586  27860
8701      Compact          Coupe           22        16         586  31000
8702      Compact          Coupe           22        16         586  26435
8703      Compact          Coupe           23        16         586  31700
8704      Compact          Coupe           23        16         586  28560
```

```
8705       Compact          Coupe           22          16           586   32140
8706       Compact          Coupe           23          16           586   26645
8707       Compact          Coupe           23          16           586   32810
8708       Compact          Coupe           22          16           586   26645
8709       Compact          Coupe           22          16           586   32110
8710       Compact          Coupe           23          16           586   32960
8711       Compact          Coupe           22          16           586   32260
8712       Compact          Coupe           22          16           586   32290
8713       Compact          Coupe           22          16           586   26795
8714       Compact          Coupe           23          16           586   26795

      Final_HP
1983  200.0000
1984  200.0000
3716  115.0000
3717  115.0000
3718  115.0000
3719  115.0000
5778   66.0000
5779   66.0660
5780   66.0660
8373  154.0154
8695  255.0000
8696  255.0000
8697  255.0000
8698  232.0000
8699  212.0000
8700  232.0000
8701  232.0000
8702  232.0000
8703  212.0000
8704  212.0000
8705  232.0000
8706  212.0000
8707  212.0000
8708  232.0000
8709  232.0000
8710  212.0000
8711  232.0000
8712  232.0000
8713  232.0000
8714  212.0000
```

These cars are either electric or Mazdas with a rotary engine. In either case, they did not have any cylinders, so we can safely replace the NaN with 0.

```
In [20]: #Replace NaN with 0 in the Engine Cylinders Column
         df['Engine_Cylinders'].fillna(0, inplace = True);
```

In [21]:
```
df.isnull().sum()
```

Out[21]:
```
Make                 0
Model                0
Year                 0
Engine_Fuel_Type     3
Engine_Cylinders     0
Transmission_Type    0
Driven_Wheels        0
Number_of_Doors      6
Vehicle_Size         0
Vehicle_Style        0
highway_MPG          0
city_mpg             0
Popularity           0
MSRP                 0
Final_HP             0
dtype: int64
```

Engine Fuel

In [22]:
```
df_temp = df[df['Engine_Fuel_Type'].isnull()]
```

In [23]:
```
df_temp
```

Out[23]:
```
         Make   Model  Year Engine_Fuel_Type  Engine_Cylinders  \
11321  Suzuki  Verona  2004              NaN               6.0
11322  Suzuki  Verona  2004              NaN               6.0
11323  Suzuki  Verona  2004              NaN               6.0

      Transmission_Type      Driven_Wheels  Number_of_Doors Vehicle_Size  \
11321         AUTOMATIC  front wheel drive              4.0      Midsize
11322         AUTOMATIC  front wheel drive              4.0      Midsize
11323         AUTOMATIC  front wheel drive              4.0      Midsize

      Vehicle_Style  highway_MPG  city_mpg  Popularity   MSRP  Final_HP
11321         Sedan           25        17         481  17199     155.0
11322         Sedan           25        17         481  20199     155.0
11323         Sedan           25        17         481  18499     155.0
```

In [24]:
```
#Replace NaN with regular unleaded in the Engine Fuel Type Column
df['Engine_Fuel_Type'].fillna('regular unleaded', inplace = True);
```

Number of doors

In [25]:
```
df_temp = df[df['Number_of_Doors'].isnull()]
```

In [26]:
```
df_temp
```

```
Out[26]:         Make     Model  Year               Engine_Fuel_Type  Engine_Cylinders  \
         4666  Ferrari       FF  2013  premium unleaded (required)              12.0
         6930    Tesla  Model S  2016                      electric               0.0
         6931    Tesla  Model S  2016                      electric               0.0
         6932    Tesla  Model S  2016                      electric               0.0
         6933    Tesla  Model S  2016                      electric               0.0
         6934    Tesla  Model S  2016                      electric               0.0

              Transmission_Type     Driven_Wheels  Number_of_Doors Vehicle_Size  \
         4666   AUTOMATED_MANUAL   all wheel drive              NaN        Large
         6930       DIRECT_DRIVE   all wheel drive              NaN        Large
         6931       DIRECT_DRIVE   all wheel drive              NaN        Large
         6932       DIRECT_DRIVE   all wheel drive              NaN        Large
         6933       DIRECT_DRIVE  rear wheel drive              NaN        Large
         6934       DIRECT_DRIVE   all wheel drive              NaN        Large

              Vehicle_Style  highway_MPG  city_mpg  Popularity    MSRP  Final_HP
         4666         Coupe           16        11        2774  295000     651.0
         6930         Sedan          105       102        1391   79500     600.0
         6931         Sedan          101        98        1391   66000     600.0
         6932         Sedan          105        92        1391  134500     600.0
         6933         Sedan          100        97        1391   74500     600.0
         6934         Sedan          107       101        1391   71000     600.0
```

In this case we see that there are two car models that don't have the number of doors. For all the Teslas we will have 4 doors, while the ferrari has 2 doors. First I will change the NaN to 4, and then deal with the ferrari later if needed.

```
In [27]: #Replace NaN with 4 in the Number of doors Column
         df['Number_of_Doors'].fillna(4, inplace = True);

In [28]: df.isnull().sum()

Out[28]: Make                 0
         Model                0
         Year                 0
         Engine_Fuel_Type     0
         Engine_Cylinders     0
         Transmission_Type    0
         Driven_Wheels        0
         Number_of_Doors      0
         Vehicle_Size         0
         Vehicle_Style        0
         highway_MPG          0
         city_mpg             0
         Popularity           0
         MSRP                 0
         Final_HP             0
         dtype: int64
```

As we can see above, we have taken care of the null values
Next up, we will look at duplicated values

1.2.2 Duplicated Values

```
In [29]: dup_rows = df[df.duplicated()]
```

```
In [30]: dup_rows.head(20)
```

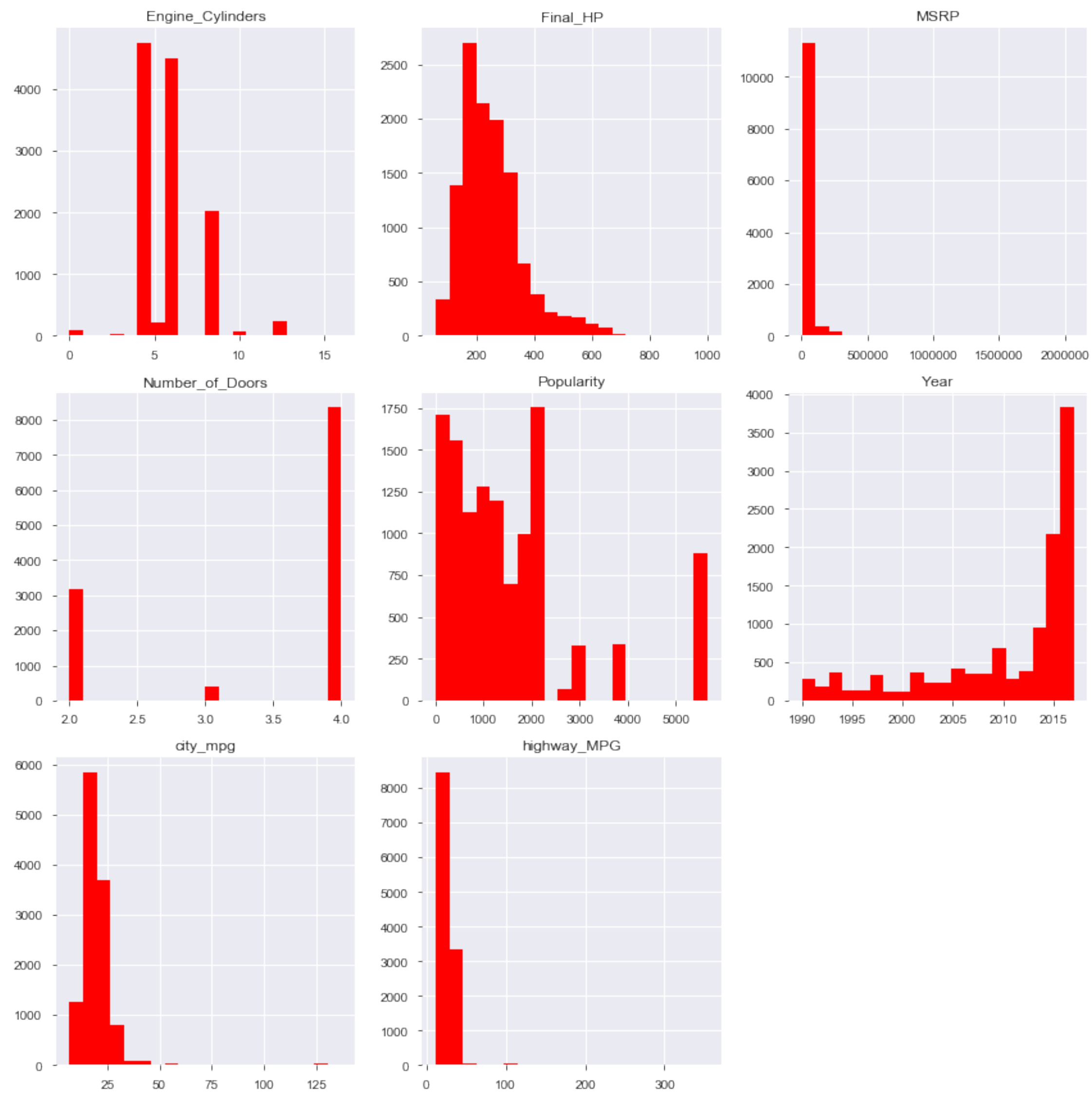
```
Out[30]:        Make                  Model  Year              Engine_Fuel_Type  \
         14      BMW               1 Series  2013  premium unleaded (required)
         18     Audi                    100  1992              regular unleaded
         20     Audi                    100  1992              regular unleaded
         24     Audi                    100  1993              regular unleaded
         25     Audi                    100  1993              regular unleaded
         88   Nissan                  200SX  1996              regular unleaded
         92   Nissan                  200SX  1997              regular unleaded
         94   Nissan                  200SX  1998              regular unleaded
         109   Volvo                    240  1992              regular unleaded
         126     BMW  3 Series Gran Turismo  2015  premium unleaded (required)
         137     BMW               3 Series  2015  premium unleaded (required)
         141     BMW               3 Series  2015  premium unleaded (required)
         252   Mazda                    323  1992              regular unleaded
         413     BMW    4 Series Gran Coupe  2015  premium unleaded (required)
         414     BMW    4 Series Gran Coupe  2015  premium unleaded (required)
         431     BMW               4 Series  2015  premium unleaded (required)
         432     BMW               4 Series  2015  premium unleaded (required)
         435     BMW               4 Series  2015  premium unleaded (required)
         436     BMW               4 Series  2015  premium unleaded (required)
         677 Pontiac                   6000  1990              regular unleaded

              Engine_Cylinders Transmission_Type      Driven_Wheels  Number_of_Doors  \
         14                6.0            MANUAL   rear wheel drive              2.0
         18                6.0            MANUAL  front wheel drive              4.0
         20                6.0            MANUAL  front wheel drive              4.0
         24                6.0            MANUAL  front wheel drive              4.0
         25                6.0            MANUAL  front wheel drive              4.0
         88                4.0            MANUAL  front wheel drive              2.0
         92                4.0            MANUAL  front wheel drive              2.0
         94                4.0            MANUAL  front wheel drive              2.0
         109               4.0            MANUAL   rear wheel drive              4.0
         126               4.0         AUTOMATIC    all wheel drive              4.0
         137               4.0         AUTOMATIC    all wheel drive              4.0
         141               4.0         AUTOMATIC   rear wheel drive              4.0
         252               4.0            MANUAL  front wheel drive              2.0
         413               4.0         AUTOMATIC   rear wheel drive              4.0
         414               4.0         AUTOMATIC    all wheel drive              4.0
         431               4.0         AUTOMATIC    all wheel drive              2.0
```

```
432              4.0         AUTOMATIC    all wheel drive              2.0
435              4.0         AUTOMATIC   rear wheel drive              2.0
436              4.0         AUTOMATIC   rear wheel drive              2.0
677              6.0         AUTOMATIC  front wheel drive              4.0

    Vehicle_Size  Vehicle_Style  highway_MPG  city_mpg  Popularity   MSRP  \
14       Compact          Coupe           28        19        3916  31500
18       Midsize          Sedan           24        17        3105   2000
20       Midsize          Sedan           24        17        3105   2000
24       Midsize          Sedan           24        17        3105   2000
25       Midsize          Sedan           24        17        3105   2000
88       Compact          Coupe           36        26        2009   2000
92       Compact          Coupe           35        25        2009   2000
94       Compact          Coupe           35        25        2009   2000
109      Midsize          Sedan           26        19         870   2000
126      Midsize  4dr Hatchback           33        22        3916  41850
137      Midsize          Sedan           33        22        3916  39500
141      Midsize          Sedan           35        23        3916  37500
252      Compact  2dr Hatchback           33        25         586   2000
413      Midsize          Sedan           34        23        3916  40300
414      Midsize          Sedan           33        22        3916  42300
431      Midsize          Coupe           33        22        3916  42750
432      Midsize    Convertible           33        21        3916  50750
435      Midsize    Convertible           34        23        3916  48750
436      Midsize          Coupe           35        23        3916  40750
677      Midsize          Wagon           27        17         210   2000

     Final_HP
14      230.0
18      172.0
20      172.0
24      172.0
25      172.0
88      115.0
92      115.0
94      115.0
109     114.0
126     240.0
137     240.0
141     240.0
252      82.0
413     240.0
414     240.0
431     240.0
432     240.0
435     240.0
436     240.0
677     140.0
```

We see that there are duplicated values, but it just could be multiple cars with the same attributes, and therefore, I will not drop them

1.2.3 Data exploration for outliers

```
In [31]: df.hist(bins = 20,grid=True, figsize = (12,12), color = 'red');
         plt.tight_layout()
```



Observations

- There are some outliers in the price. Even though it is true that there are cars that are USD 2M, they will completely skew the distribution, so we will drop any car >500K
- We will also limit highway MPG to 130.

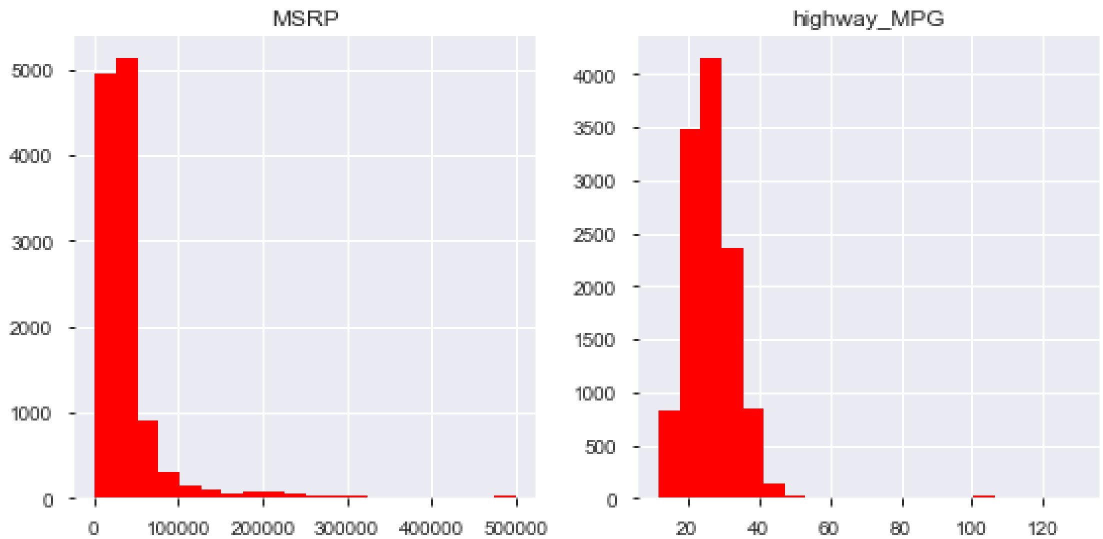
Limit the values of price and highway MPG

```
In [32]: df_clean = df.copy()
```

```
In [33]: df_clean['MSRP'] = df_clean['MSRP'].clip(upper = 500000)
```

```
In [34]: df_clean['highway_MPG'] = df_clean['highway_MPG'].clip(upper = 130)
```

```
In [35]: df_clean.hist(column = ['highway_MPG','MSRP'],bins = 20,grid=True, figsize = (8,4), co
         plt.tight_layout()
```



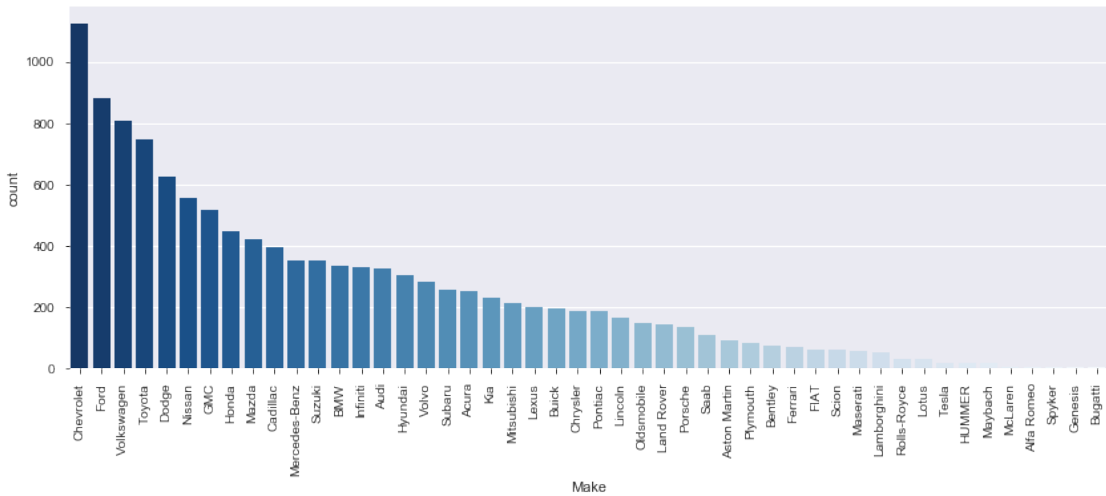
We can see that the data is still right skewed, and we will log transform this before applying ML algorithms

Now that we have cleaned up the data, we will generate a profile report

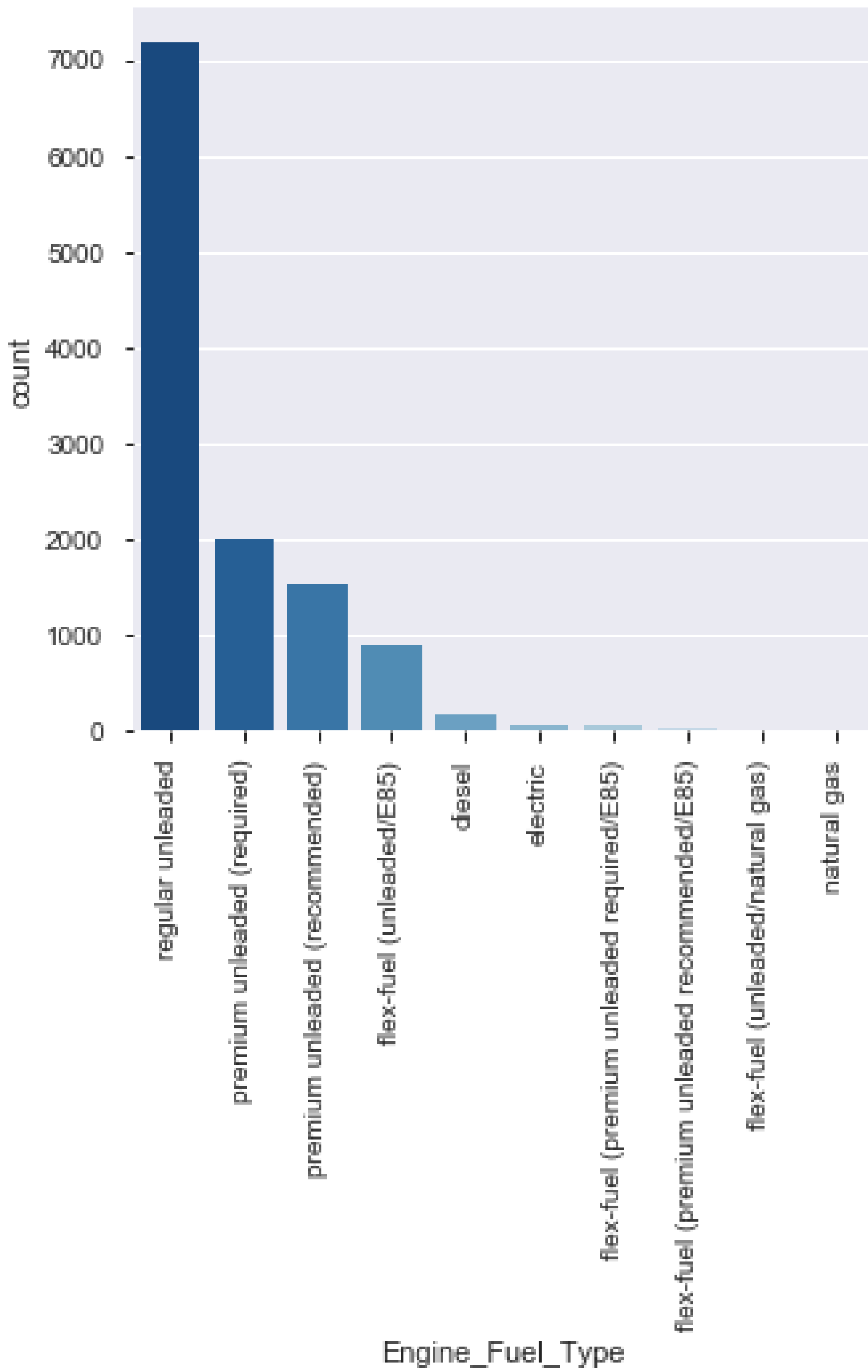
1.3 Univariate Data Exploration

```
In [36]: #We will write a function that we can use for univariate exploration
         #Variable is the column name
         #x and y are length and width of the plot
         def univar(variable, x, y):
             plt.figure(figsize=[x, y])
             sns.countplot(data = df_clean, x= variable,order = df_clean[variable].value_counts
             plt.xticks(rotation=90);
```

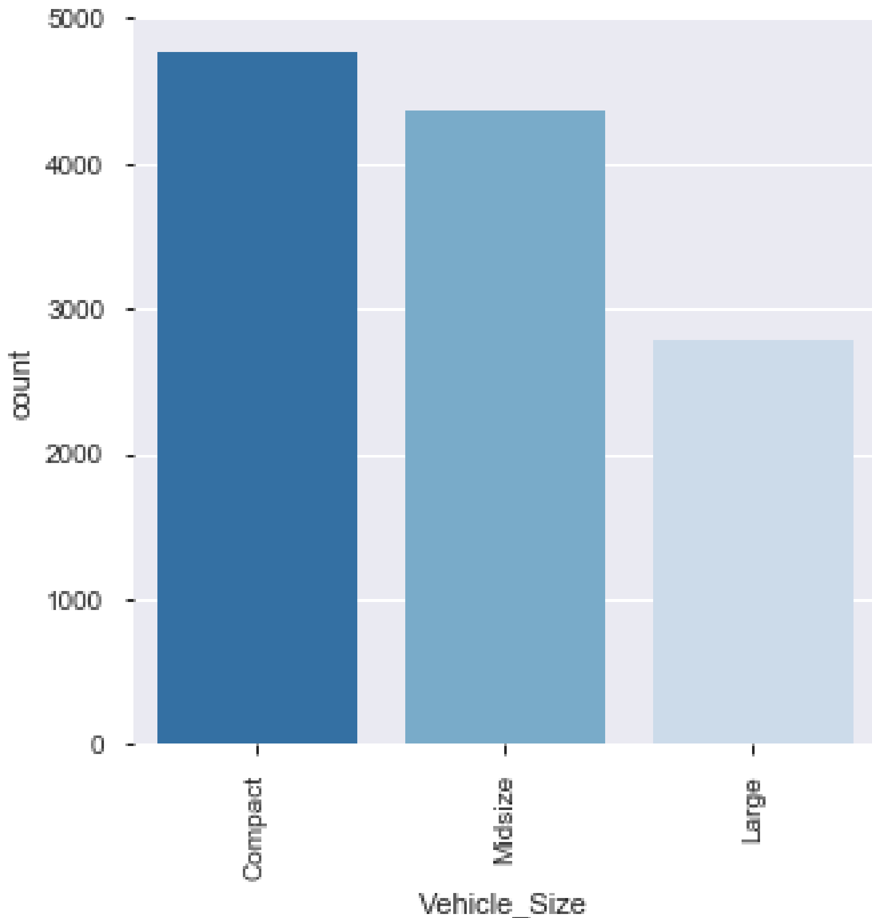
```
In [37]: #Make of the car
         univar('Make', 14, 5)
```



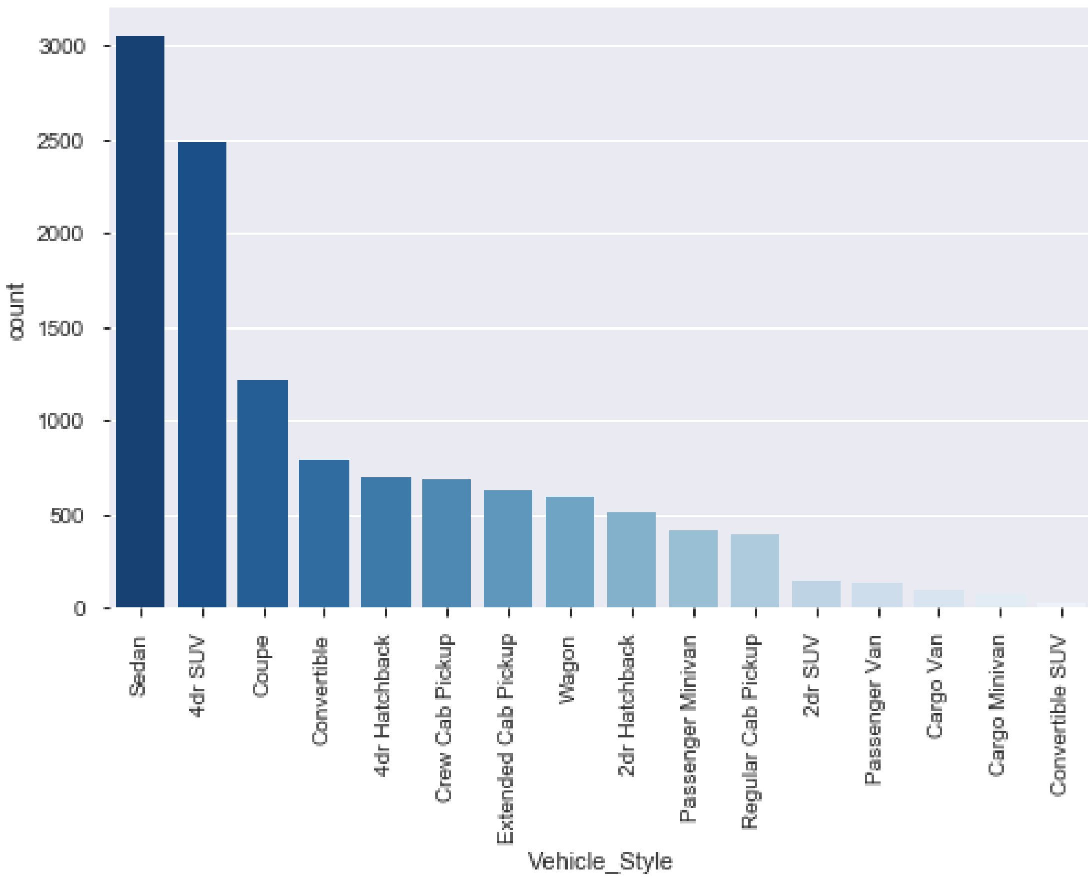
In [38]:
```
#Engine fuel type
univar('Engine_Fuel_Type', 5, 5)
```



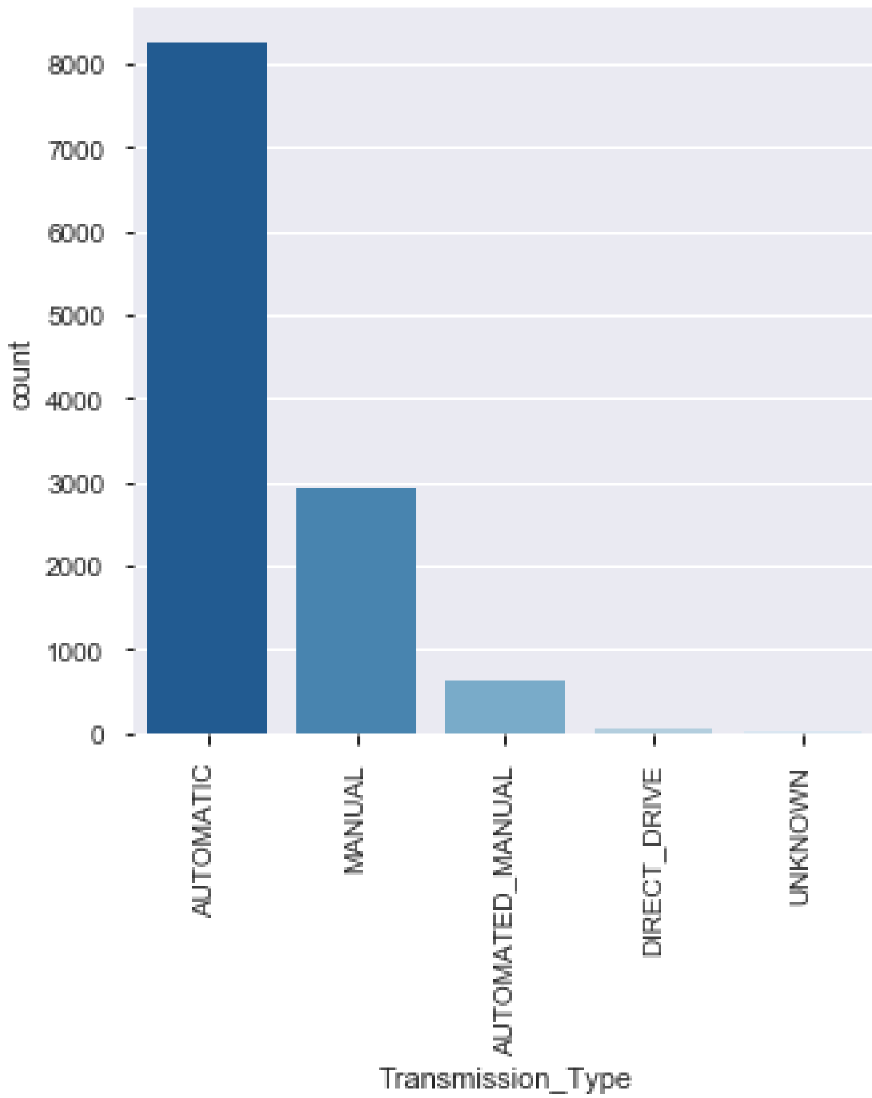

```
In [39]: #Vehicle Size
         univar('Vehicle_Size', 5, 5)
```



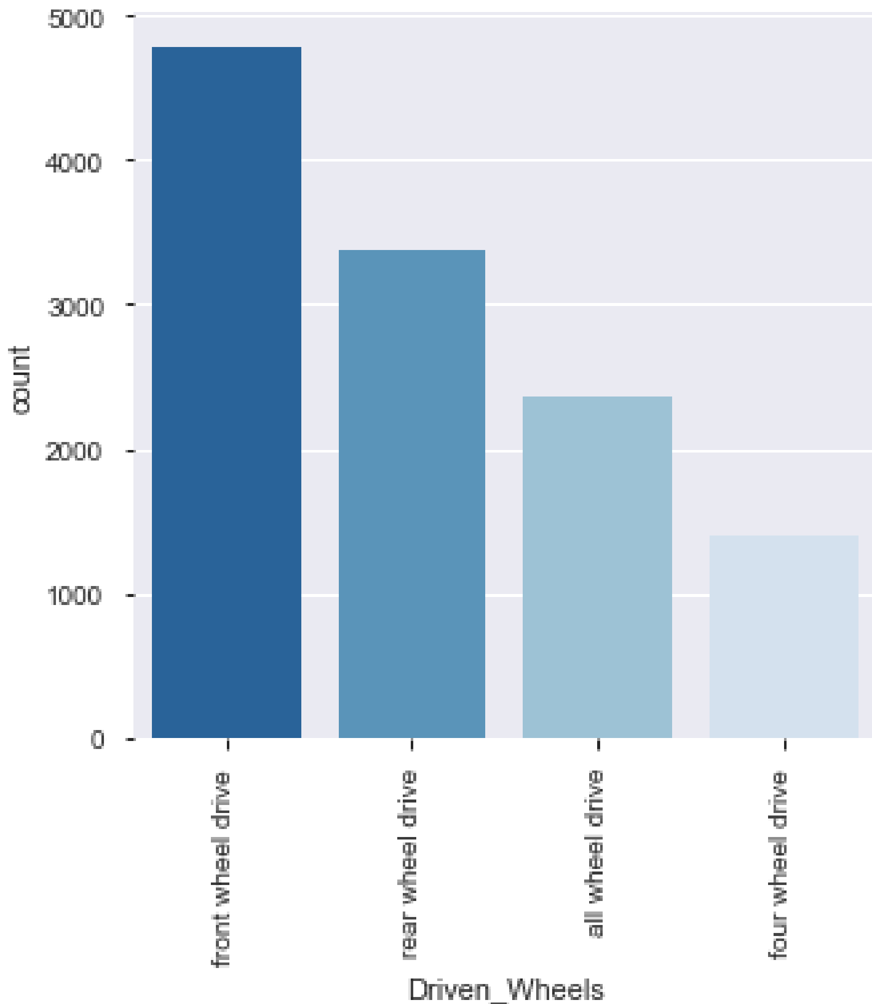

```
In [40]: #Vehicle Style
         univar('Vehicle_Style',8,5)
```




```
In [41]: #Transmission
         univar('Transmission_Type',5,5)
```




```
In [42]: #Drive wheels
         univar('Driven_Wheels',5,5)
```

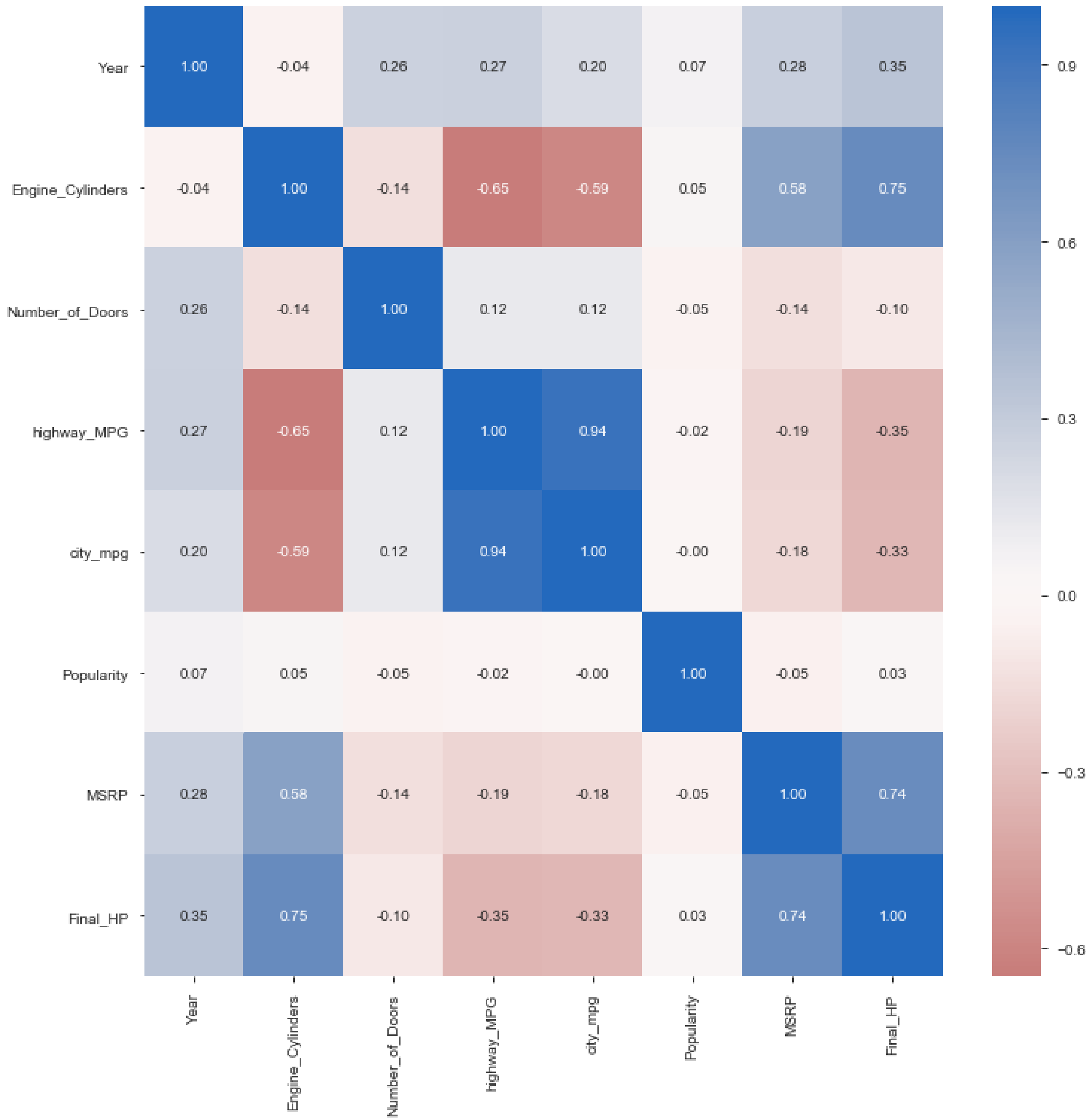


1.3.1 Univariate Data Exploration Summary

Most common categories - Brand: Chevy - Fuel: Regular unleaded - Type: Sedan - Transmission: Automatic - Drive: Front Wheel Drive

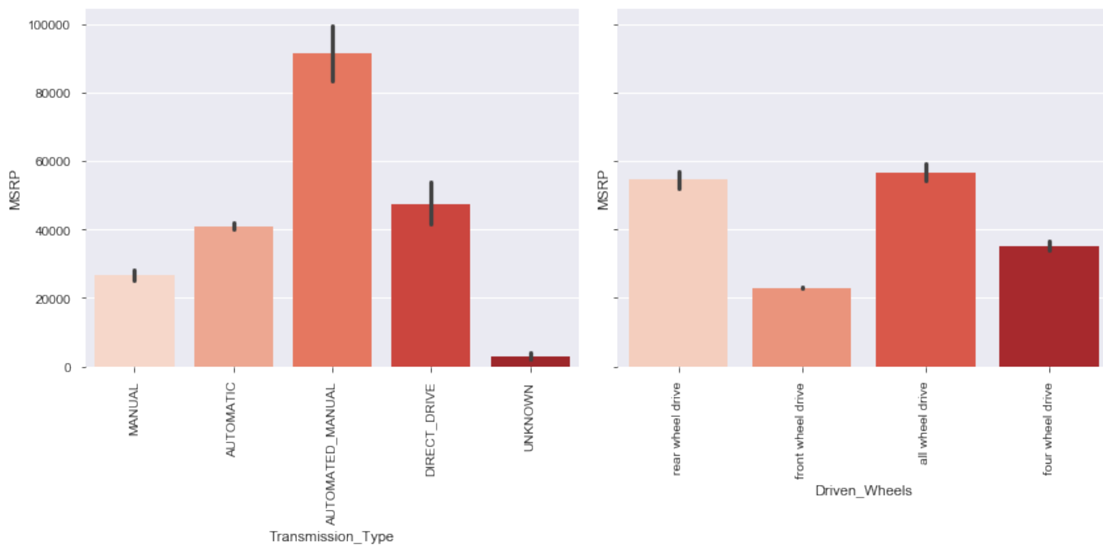
1.4 Bivariate Data Exploration

```
In [43]: # correlation plot
         plt.figure(figsize = [12,12])
         sns.heatmap(df_clean.corr(), annot = True, fmt = '.2f',
                     cmap = 'vlag_r', center = 0)
         plt.show()
```

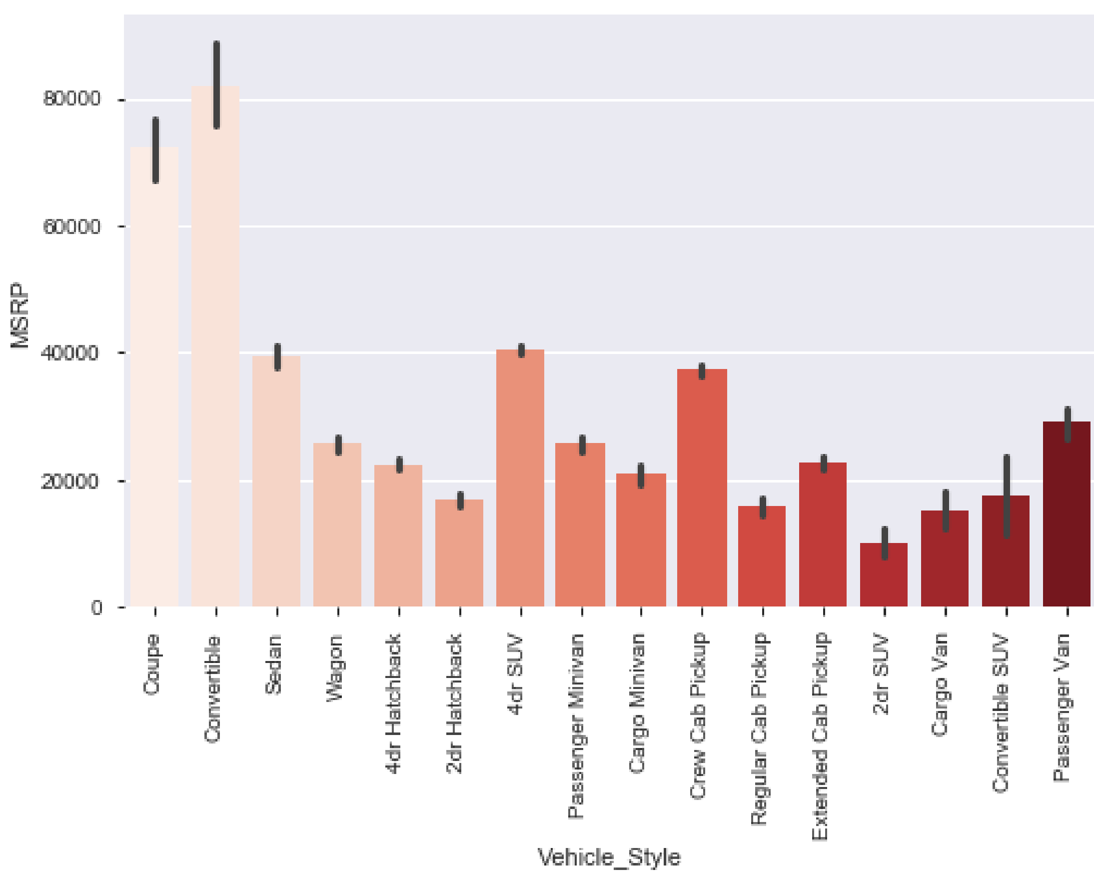


1.4.1 MSRP

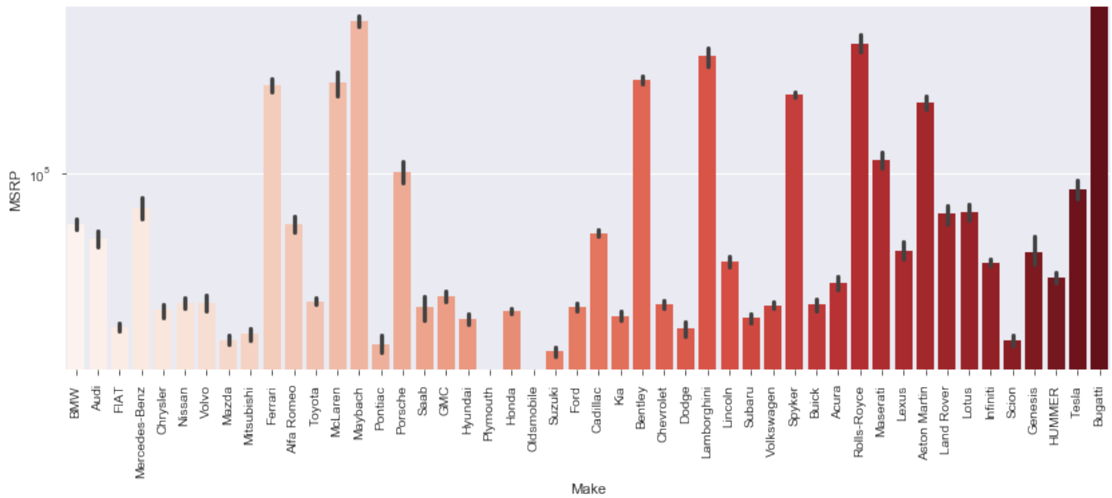
```
In [44]: #MSRP vs Transmission and Wheels driven
         fig, axarr = plt.subplots(1, 2, figsize=(12, 6), sharey = True)

         sns.barplot(x = 'Transmission_Type', y = 'MSRP', data = df_clean,palette="Reds", ax=a
         sns.barplot(x = 'Driven_Wheels', y = 'MSRP', data = df_clean,palette="Reds", ax=axarr
         plt.sca(axarr[0])
         plt.xticks(rotation=90)
         plt.sca(axarr[1])
         plt.xticks(rotation=90)
         plt.tight_layout()
```




```
In [45]: #Let's check the effect of vehicle size on the MSRP
         plt.figure(figsize=[8, 5])
         sns.barplot(x = 'Vehicle_Style', y = 'MSRP', data = df_clean,palette="Reds");
         plt.xticks(rotation=90);
```

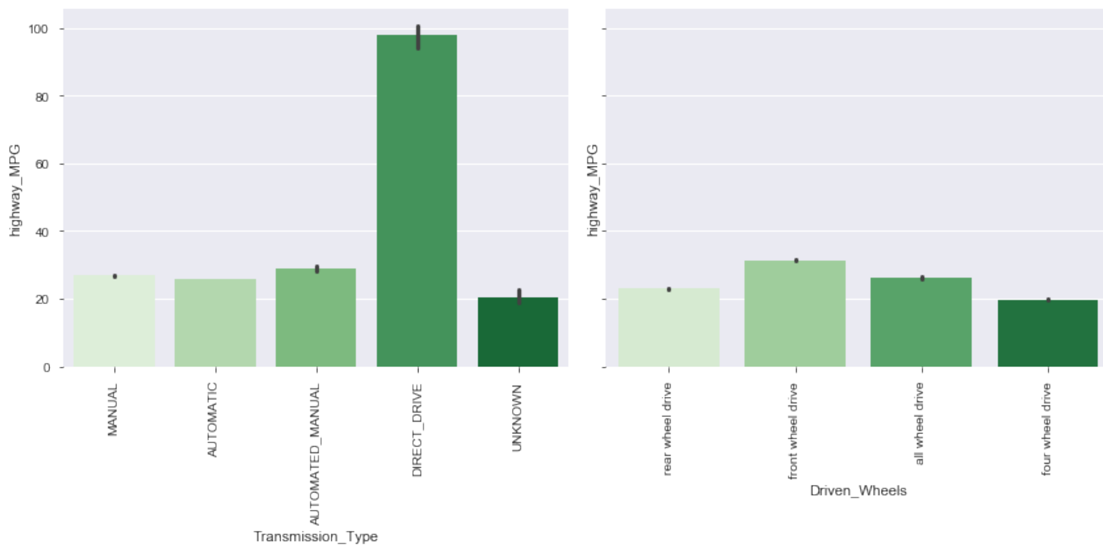


```
In [46]: #Car Make vs MSRP
         plt.figure(figsize=[14, 5])
         sns.barplot(x = 'Make', y = 'MSRP', data = df_clean,palette="Reds");
         plt.yscale('symlog')
         plt.ylim(15000,500000)
         plt.xticks(rotation=90);
```

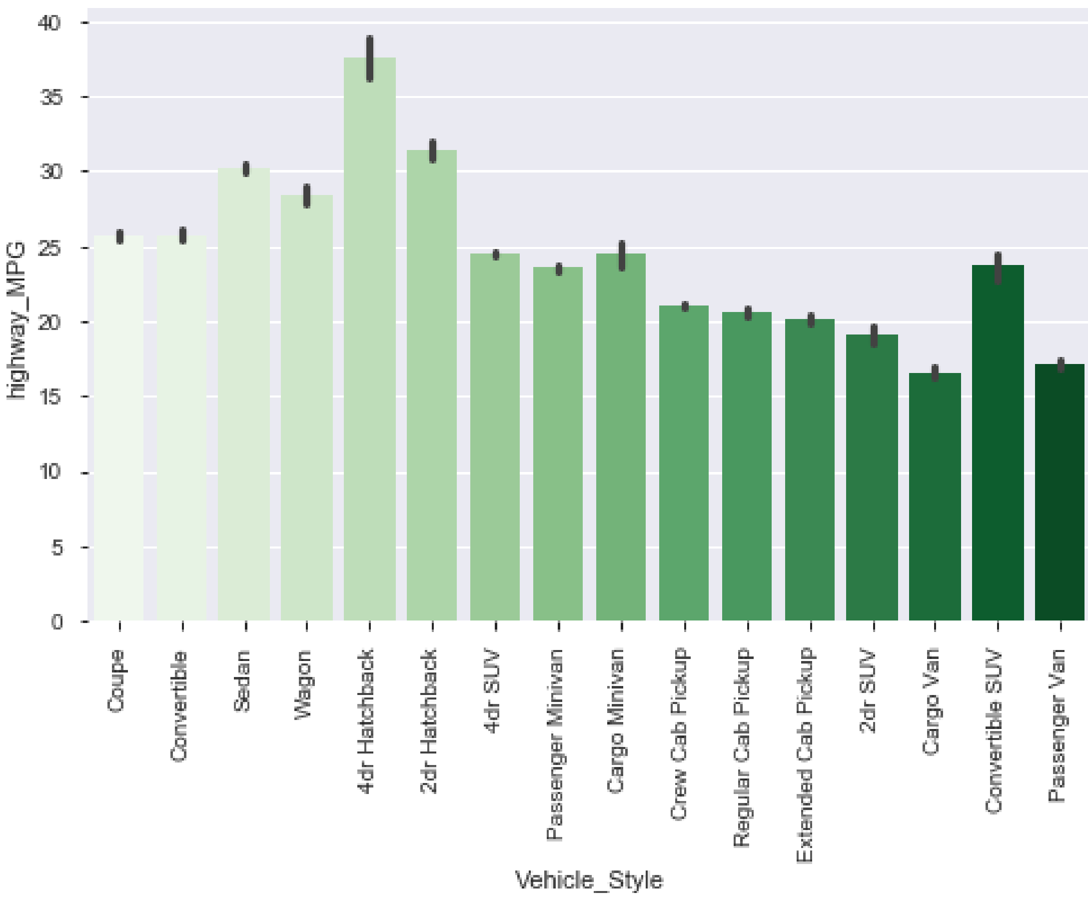


1.4.2 Highway MPG

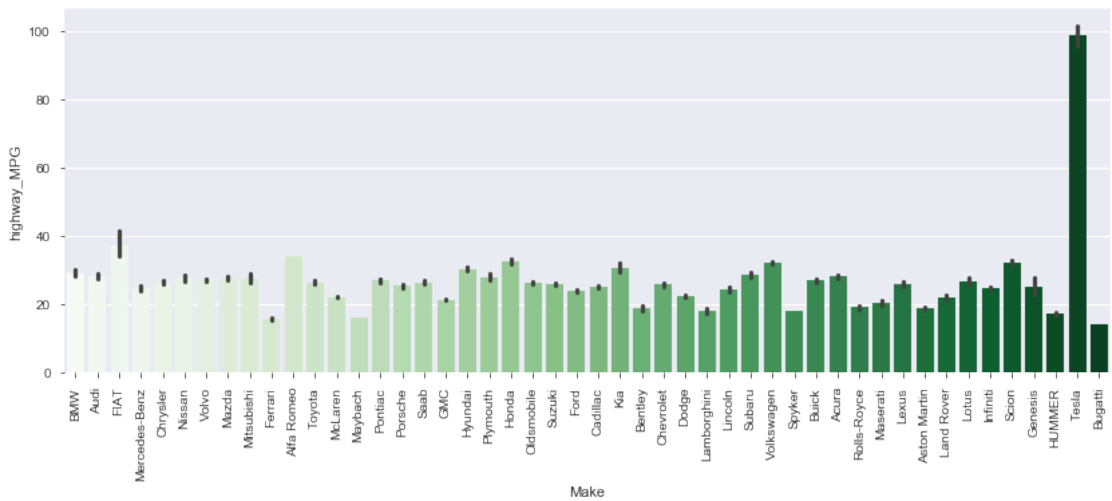
```
In [47]: #MSRP vs Transmission and Wheels driven
         fig, axarr = plt.subplots(1, 2, figsize=(12, 6), sharey = True)

         sns.barplot(x = 'Transmission_Type', y = 'highway_MPG', data = df_clean,palette="Gree
         sns.barplot(x = 'Driven_Wheels', y = 'highway_MPG', data = df_clean,palette="Greens",
         plt.sca(axarr[0])
         plt.xticks(rotation=90)
         plt.sca(axarr[1])
         plt.xticks(rotation=90)
         plt.tight_layout()
```




```
In [48]: #Let's check the effect of vehicle size on the Mileage
         plt.figure(figsize=[8, 5])
         sns.barplot(x = 'Vehicle_Style', y = 'highway_MPG', data = df_clean,palette="Greens")
         plt.xticks(rotation=90);
```



In [49]:
```
#Car Make vs MPG
plt.figure(figsize=[14, 5])
sns.barplot(x = 'Make', y = 'highway_MPG', data = df_clean,palette="Greens");
#plt.ylim(15000,500000)
plt.xticks(rotation=90);
```



1.4.3 Bivariate Data Exploration Summary

MSRP

- Automated Manuals are most expensive (these are mostly exotic supercars)
- FWD are cheapest
- Coupes and Convertibles are most expensive
- There are three tiers based on brands
  - Tier 1 consists of ultra premium cars are Buggati, Maybach, Ferrari, etc
  - The second tier is BMW, Audi, Mercedes, Infiniti etc
  - Most mass market cars like Ford, Chevy form the third tier

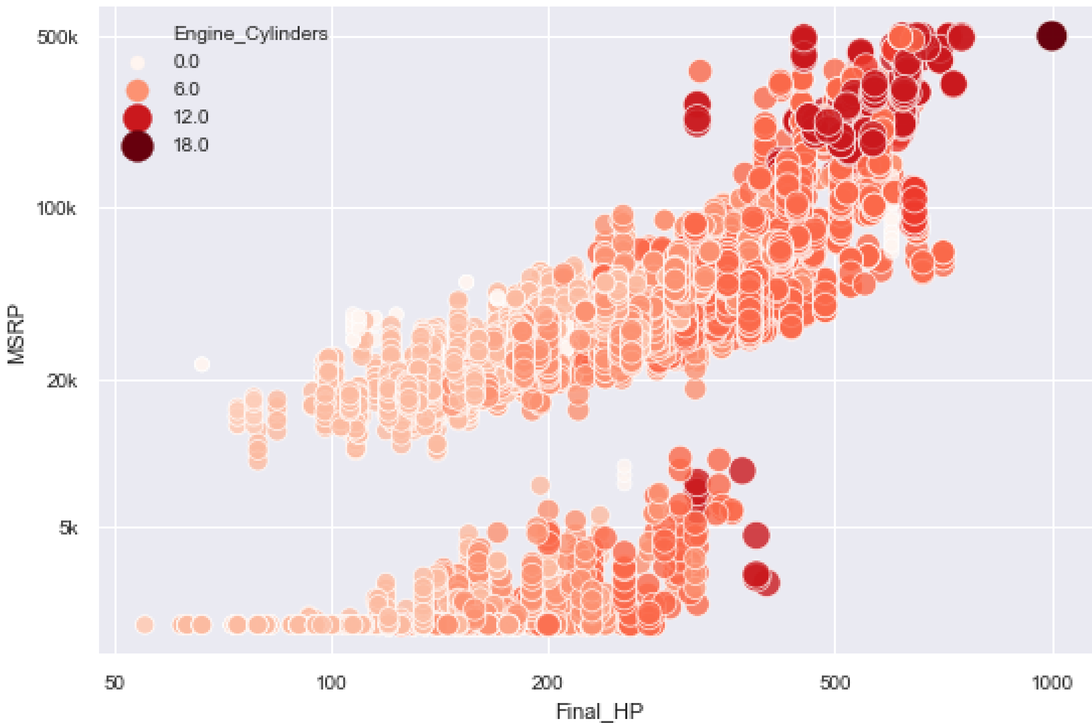
MPG

- Direct drive has highest MPG (these electric cars)
- FWD cars have better MPG than rear wheel or AWD (these are more mass market cars, with less perfomance)
- 4DR Hatchbacks have best MPG (most electric cars fall in this category)
- Tesla as a brand has the best MPG (They only make electric cars)

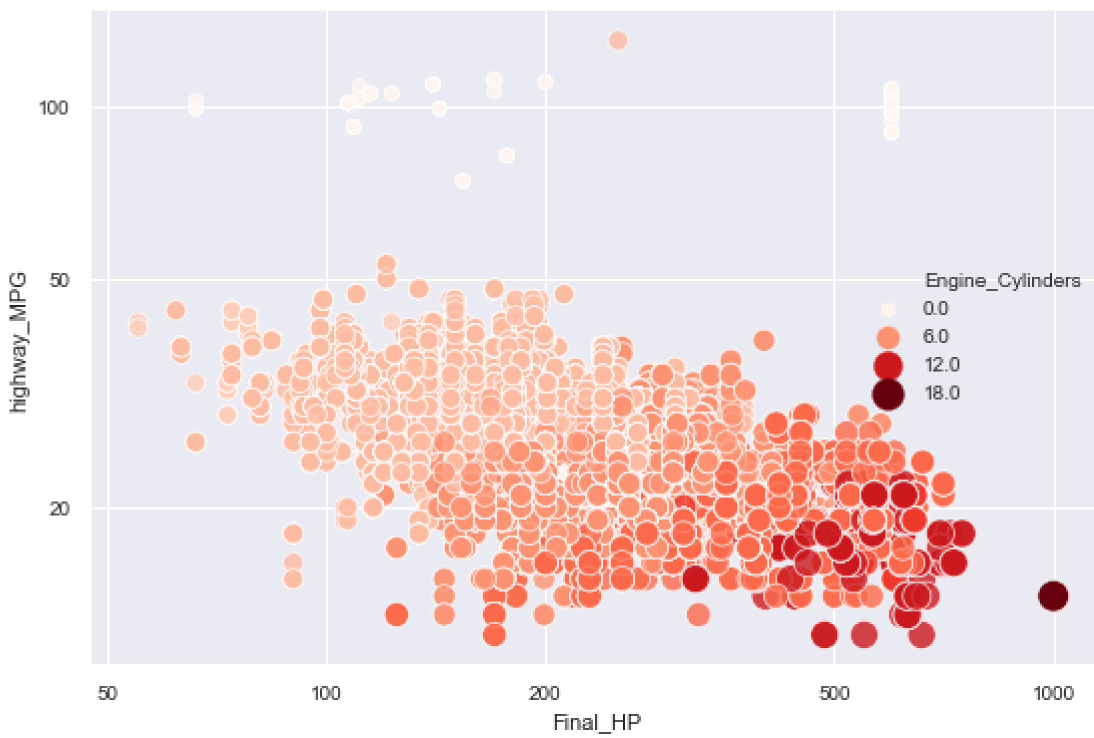
1.5 Multivariate Exploration

In [50]:
```
#Let's first look at the MSRP vs Horsepower
plt.figure(figsize=[9, 6])
sns.set_style("darkgrid")
sns.scatterplot(data = df_clean, x = 'Final_HP', y = 'MSRP', hue = 'Engine_Cylinders'

#Rescale the plot to better visualize the distribution
plt.xscale('log')
plt.xticks([50, 100, 200, 500, 1000], [50, 100, 200, 500, 1000])
```

```
plt.yscale('log')
plt.yticks([ 5000,20000, 100000,  500000], [ '5k' ,'20k',  '100k', '500k']);
```




```
In [51]: plt.figure(figsize=[9, 6])
         sns.set_style("darkgrid")
         sns.scatterplot(data = df_clean, x = 'Final_HP', y = 'highway_MPG', hue = 'Engine_Cyli
         plt.xscale('log')
         plt.xticks([50, 100, 200, 500, 1000], [50, 100, 200, 500, 1000])
         plt.yscale('log')
         plt.yticks([20, 50, 100], [ 20, 50, 100]);
```

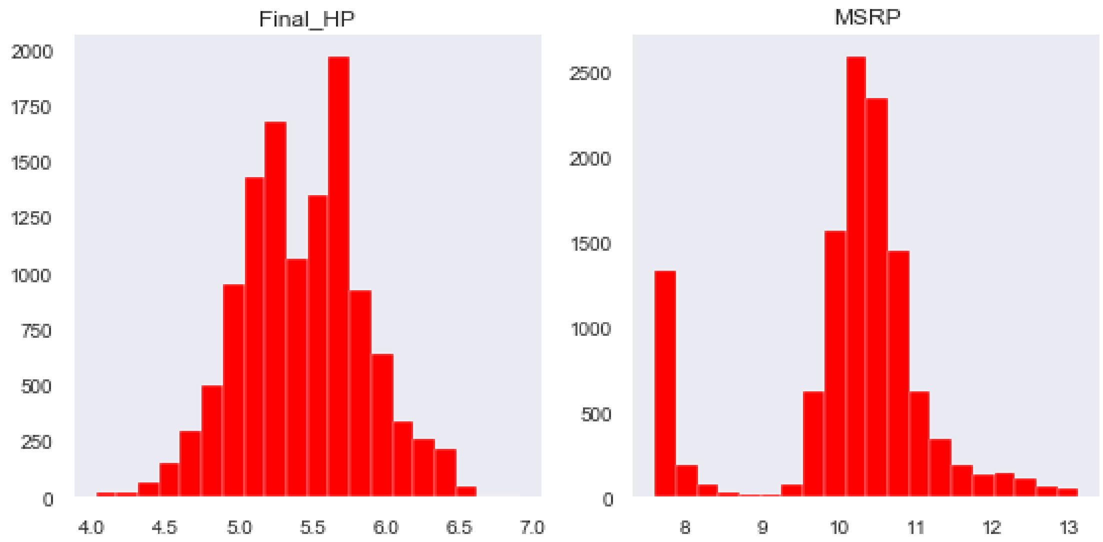


1.6 Building a classifcation model for Fuel Economy

```
In [83]: df_class = df_clean.copy()
```

1.6.1 Transform and scale the dataset

```
In [84]: # Split the data into features and target label
         mileage_raw = df_class[['highway_MPG']]
         # in the features dataset, we will drop the highway_MPG, because that is the target
         # We will also drop city_mpg because that is highly correlated to highway_MPG,
         # and Engine_Cylinders, which are highly correlated to Horsepower
         features_raw = df_class.drop(['highway_MPG','city_mpg','Engine_Cylinders'], axis = 1)

In [85]: # Log-transform the skewed features
         skewed = ['MSRP', 'Final_HP']
         features_log_transformed = pd.DataFrame(data = features_raw)
         features_log_transformed[skewed] = features_raw[skewed].apply(lambda x: np.log(x + 1))

In [86]: features_log_transformed.hist(column = ['Final_HP','MSRP'],bins = 20,grid=False, figs
         plt.tight_layout()
```




```
In [88]: # Import sklearn.preprocessing.StandardScaler
         from sklearn.preprocessing import MinMaxScaler

         # Initialize a scaler, then apply it to the features
         scaler = MinMaxScaler() # default=(0, 1)
         numerical = ['Year', 'Number_of_Doors', 'Final_HP']

         features_log_minmax_transform = pd.DataFrame(data = features_log_transformed)
         features_log_minmax_transform[numerical] = scaler.fit_transform(features_log_transform

         # Show an example of a record with scaling applied
         display(features_log_minmax_transform.head(n = 10))
```

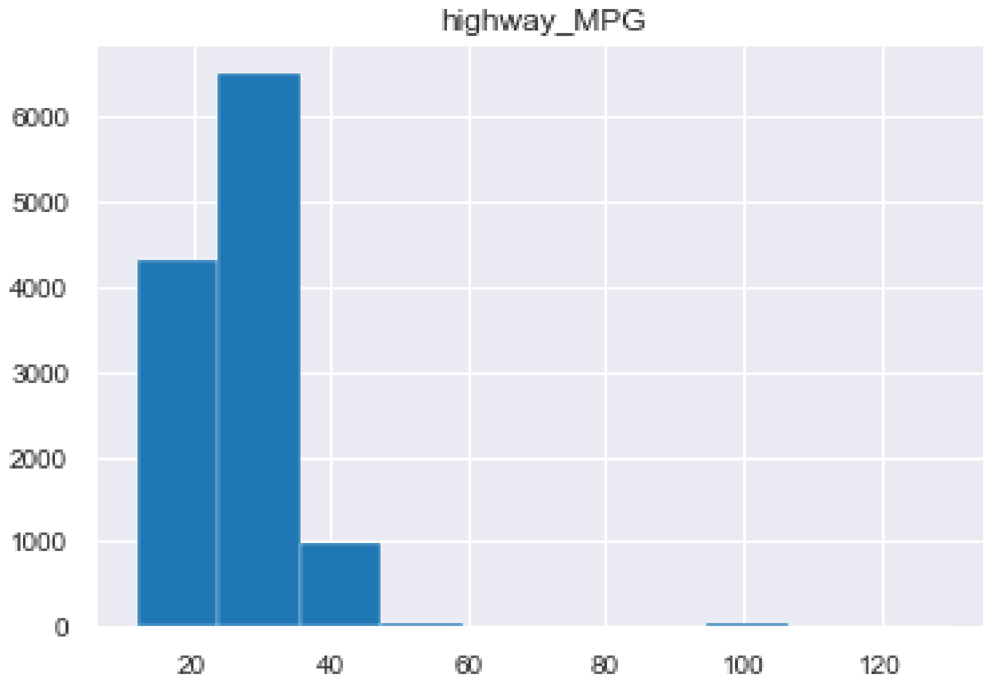
```
  Make        Model      Year             Engine_Fuel_Type Transmission_Type  \
0  BMW  1 Series M  0.777778  premium unleaded (required)            MANUAL
1  BMW    1 Series  0.777778  premium unleaded (required)            MANUAL
2  BMW    1 Series  0.777778  premium unleaded (required)            MANUAL
3  BMW    1 Series  0.777778  premium unleaded (required)            MANUAL
4  BMW    1 Series  0.777778  premium unleaded (required)            MANUAL
5  BMW    1 Series  0.814815  premium unleaded (required)            MANUAL
6  BMW    1 Series  0.814815  premium unleaded (required)            MANUAL
7  BMW    1 Series  0.814815  premium unleaded (required)            MANUAL
8  BMW    1 Series  0.814815  premium unleaded (required)            MANUAL
9  BMW    1 Series  0.851852  premium unleaded (required)            MANUAL

      Driven_Wheels  Number_of_Doors Vehicle_Size Vehicle_Style  Popularity  \
0  rear wheel drive              0.0      Compact         Coupe        3916
1  rear wheel drive              0.0      Compact   Convertible        3916
2  rear wheel drive              0.0      Compact         Coupe        3916
```

```
3  rear wheel drive             0.0      Compact         Coupe         3916
4  rear wheel drive             0.0      Compact   Convertible         3916
5  rear wheel drive             0.0      Compact         Coupe         3916
6  rear wheel drive             0.0      Compact   Convertible         3916
7  rear wheel drive             0.0      Compact         Coupe         3916
8  rear wheel drive             0.0      Compact   Convertible         3916
9  rear wheel drive             0.0      Compact   Convertible         3916

        MSRP  Final_HP
0  10.739349  0.621189
1  10.612779  0.583053
2  10.500977  0.583053
3  10.290483  0.491286
4  10.448744  0.491286
5  10.348205  0.491286
6  10.694238  0.583053
7  10.579005  0.583053
8  10.515994  0.491286
9  10.524091  0.491286
```

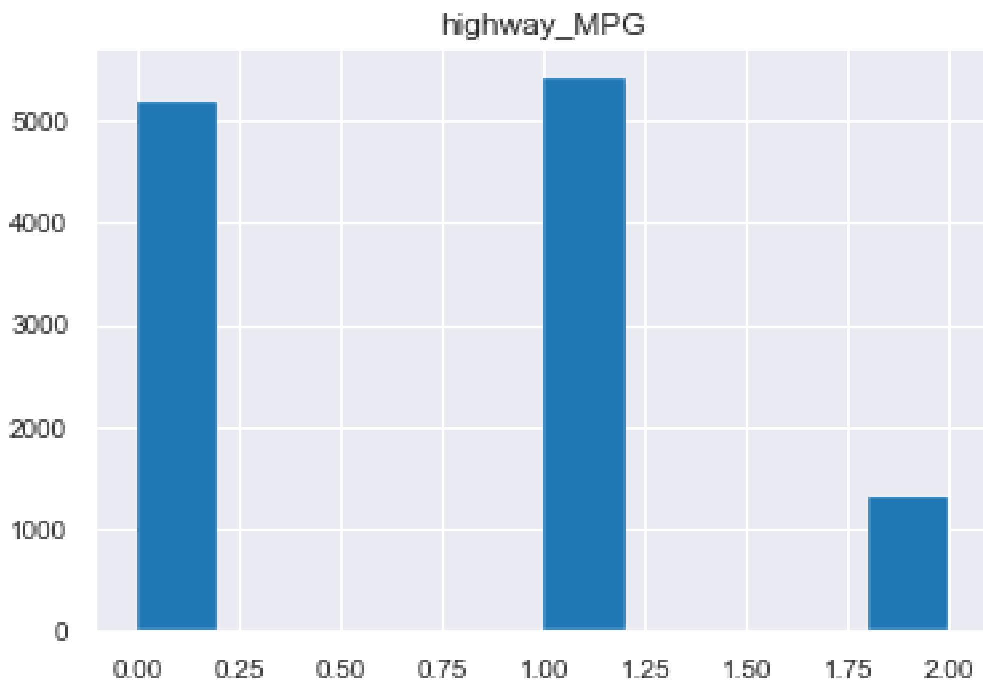
```
In [89]: mileage_raw.head()
```

```
Out[89]:    highway_MPG
         0           26
         1           28
         2           28
         3           28
         4           28
```

```
In [90]: mileage_raw.hist();
```




```
In [91]: #We will split the mileage (MPG) into three categories
         mileage_raw.loc[(mileage_raw['highway_MPG'] < 25), 'highway_MPG'] = 0
         mileage_raw.loc[(mileage_raw['highway_MPG'] >= 25)&(mileage_raw['highway_MPG'] < 35),
         mileage_raw.loc[(mileage_raw['highway_MPG'] >= 35), 'highway_MPG'] = 2

In [92]: mileage_raw.hist()

Out[92]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1a1e1d2e10>]],
               dtype=object)
```



In [93]:
```
# One-hot encode the 'features_log_minmax_transform' data using pandas.get_dummies()
features_final = pd.get_dummies(features_log_minmax_transform)


# Print the number of features after one-hot encoding
encoded = list(features_final.columns)
print("{} total features after one-hot encoding.".format(len(encoded)))

print (encoded)
```

```
1006 total features after one-hot encoding.
['Year', 'Number_of_Doors', 'Popularity', 'MSRP', 'Final_HP', 'Make_Acura', 'Make_Alfa Romeo',
```

In [94]:
```
# Import train_test_split
from sklearn.model_selection import train_test_split

# Split the 'features' and 'income' data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features_final,
                                                    mileage_raw,
                                                    test_size = 0.2,
                                                    random_state = 0)

# Show the results of the split
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))
```

```
Training set has 9531 samples.
Testing set has 2383 samples.
```

```
In [95]: #import necessary ML libraries
         from sklearn.model_selection import train_test_split
         from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoos
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
         from sklearn.metrics import  make_scorer, confusion_matrix, fbeta_score
```

```
In [96]: #Pick three classifiers
         clf_A = LogisticRegression(random_state = 0)
         clf_B = GradientBoostingClassifier(random_state = 0)
         clf_C = RandomForestClassifier(random_state = 0)

         #Fit the data to the three classifiers and print results from them
         for clf in [clf_A, clf_B, clf_C]:
             learner = clf
             start = time() # Get start time
             learner = learner.fit(X_train, y_train)
             end = time() # Get start time
             train_time = end-start


             predictions_test = learner.predict(X_test)
             predictions_train = learner.predict(X_train)

             # Score our model
             print(clf.__class__.__name__)
             print('Accuracy score: ', format(accuracy_score(y_test, predictions_test)))
             print('Precision score: ', format(precision_score(y_test, predictions_test,average
             print('Recall score: ', format(recall_score(y_test, predictions_test,average='wei
             print('F1 score: ', format(f1_score(y_test, predictions_test,average='weighted'))
             print('Time: ', format(train_time))

             print('Confusion Matrix')
             print(confusion_matrix(y_test,predictions_test))
             print( )
```

```
LogisticRegression
Accuracy score:  0.9148132605958875
Precision score:  0.9145730678269199
Recall score:  0.9148132605958875
F1 score:  0.9144162757920861
Time:  0.36118006706237793
Confusion Matrix
[[ 969   60    0]
```

```
 [  64 1007   27]
 [   0   52  204]]

GradientBoostingClassifier
Accuracy score:  0.9093579521611415
Precision score:  0.9103244617115024
Recall score:  0.9093579521611415
F1 score:  0.9092466847423099
Time:  41.488425970077515
Confusion Matrix
[[ 941   88    0]
 [  54 1019   25]
 [   1   48  207]]

RandomForestClassifier
Accuracy score:  0.9618128409567772
Precision score:  0.9617751299666105
Recall score:  0.9618128409567772
F1 score:  0.9617689108879187
Time:  0.35206103324890137
Confusion Matrix
[[1005   24    0]
 [  36 1048   14]
 [   0   17  239]]
```

In [127]:
```python
from sklearn.model_selection import GridSearchCV

#Do a GridSearch to optimize the best model from the previous step

# build a classifier
clf_rf = RandomForestClassifier(random_state = 0)

# Set up the hyperparameter search
parameters = {"n_estimators": [10,50,100] ,"max_depth": [5,50,250], "min_samples_spli

# Run a randomized search over the hyperparameters
random_search = GridSearchCV(clf_rf, parameters)

# Fit the model on the training data
grid_fit = random_search.fit(X_train, y_train)

#Get the estimator
best_clf = grid_fit.best_estimator_

# Make predictions on the test data
#rf_preds = random_search.best_estimator_.predict(X_test)
```

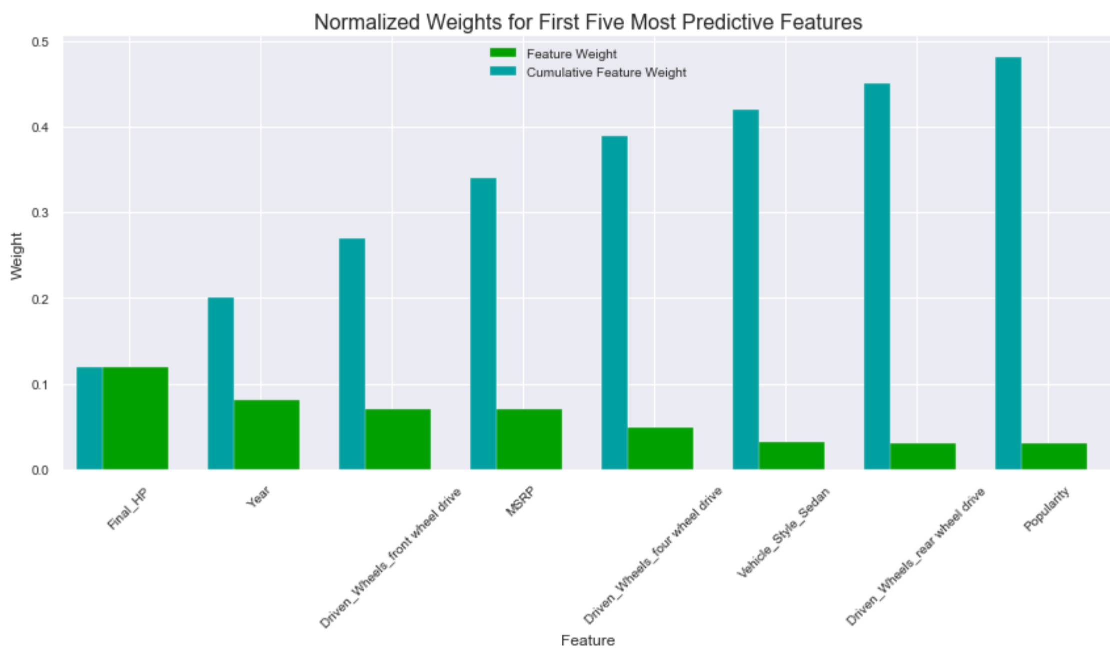
```
    predictions = (clf.fit(X_train, y_train)).predict(X_test)
    best_predictions = best_clf.predict(X_test)

    print('Accuracy score: ', format(accuracy_score(y_test, predictions)))
    print('Precision score: ', format(precision_score(y_test, predictions, average='weig
    print('Recall score: ', format(recall_score(y_test, predictions, average='weighted')
    print('F1 score: ', format(f1_score(y_test, predictions,average='weighted')))
    print('\n\n')
    print('Confusion Matrix')
    print(confusion_matrix(y_test,predictions))
```

```
Accuracy score:  0.9618128409567772
Precision score:  0.9617751299666105
Recall score:  0.9618128409567772
F1 score:  0.9617689108879187



Confusion Matrix
[[1005   24    0]
 [  36 1048   14]
 [   0   17  239]]
```

In [128]:
```
#Make a function to show the top features
def feature_plot(importances, X_train, y_train):

    # Display the five most important features
    indices = np.argsort(importances)[::-1]
    columns = X_train.columns.values[indices[:8]]
    values = importances[indices][:8]

    # Creat the plot
    fig = plt.figure(figsize = (12,7))
    plt.title("Normalized Weights for First Five Most Predictive Features", fontsize
    plt.bar(np.arange(8), values, width = 0.6, align="center", color = '#00A000', \
          label = "Feature Weight")
    plt.bar(np.arange(8) - 0.3, np.cumsum(values), width = 0.2, align = "center", co
          label = "Cumulative Feature Weight")
    plt.xticks(np.arange(8), columns)
    plt.xlim((-0.5, 7.5))
    plt.ylabel("Weight", fontsize = 12)
    plt.xlabel("Feature", fontsize = 12)
    plt.xticks(rotation=45);
    plt.legend(loc = 'upper center')
    plt.tight_layout()
    plt.show()
```

In [129]:
```
# Import a supervised learning model that has 'feature_importances_'
```

```
###Importing a new model is not necessary as the GradientBoostingClassifier already h

# Train the supervised model on the training set using .fit(X_train, y_train)
model = best_clf

# Extract the feature importances using .feature_importances_
importances = model.feature_importances_

# Plot
feature_plot(importances, X_train, y_train)
```



We see that the Final_HP is the top predictive feature. This is similar to what we found in our previous bivariate analysis, where we saw that HP and MPG were inversely related

In [ ]: