drawing-conclusions-solutions

August 28, 2020

1 Drawing Conclusions for cars models and attributes

using datasets `clean_08.csv` and `clean_18.csv`

[1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
% matplotlib inline
```

[2]:
```python
# load datasets
df_08 = pd.read_csv('clean_08.csv')
df_18 = pd.read_csv('clean_18.csv')
```

[3]:
```python
df_08.head(1)
```

[3]:
```
       model  displ  cyl    trans drive      fuel veh_class  \
0  ACURA MDX    3.7    6  Auto-S5   4WD  Gasoline       SUV

   air_pollution_score  city_mpg  hwy_mpg  cmb_mpg  greenhouse_gas_score  \
0                  7.0      15.0     20.0     17.0                     4

  smartway
0       no
```

1.0.1 Q1: Are more unique models using alternative sources of fuel? By how much?

Let's first look at what the sources of fuel are and which ones are alternative sources.

[4]:
```python
df_08.fuel.value_counts()
```

[4]:
```
Gasoline    984
gas           1
ethanol       1
CNG           1
Name: fuel, dtype: int64
```

[5]:
```python
df_18.fuel.value_counts()
```

[5]:
```
Gasoline       749
Gas             26
Ethanol         26
Diesel          19
Electricity     12
Name: fuel, dtype: int64
```

Looks like the alternative sources of fuel available in 2008 are CNG and ethanol, and those in 2018 ethanol and electricity. (You can use Google if you weren't sure which ones are alternative sources of fuel!)

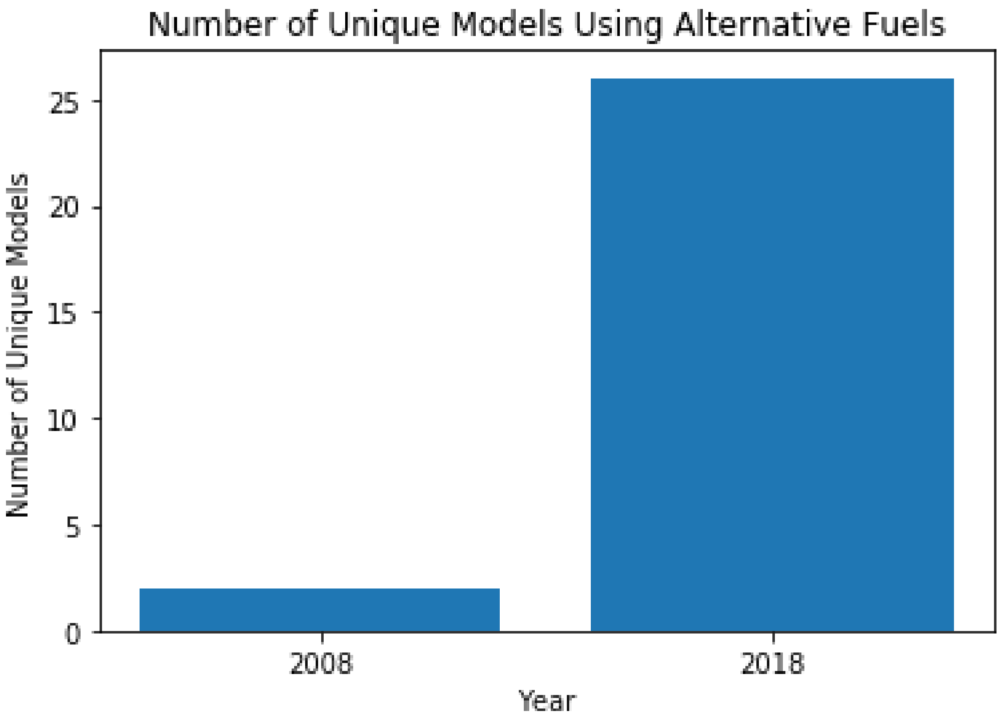
[6]:
```
# how many unique models used alternative sources of fuel in 2008
alt_08 = df_08.query('fuel in ["CNG", "ethanol"]').model.nunique()
alt_08
```

[6]:
```
2
```

[7]:
```
# how many unique models used alternative sources of fuel in 2018
alt_18 = df_18.query('fuel in ["Ethanol", "Electricity"]').model.nunique()
alt_18
```

[7]:
```
26
```

[8]:
```
plt.bar(["2008", "2018"], [alt_08, alt_18])
plt.title("Number of Unique Models Using Alternative Fuels")
plt.xlabel("Year")
plt.ylabel("Number of Unique Models");
```



Since 2008, the number of unique models using alternative sources of fuel increased by 24. We can also look at proportions.

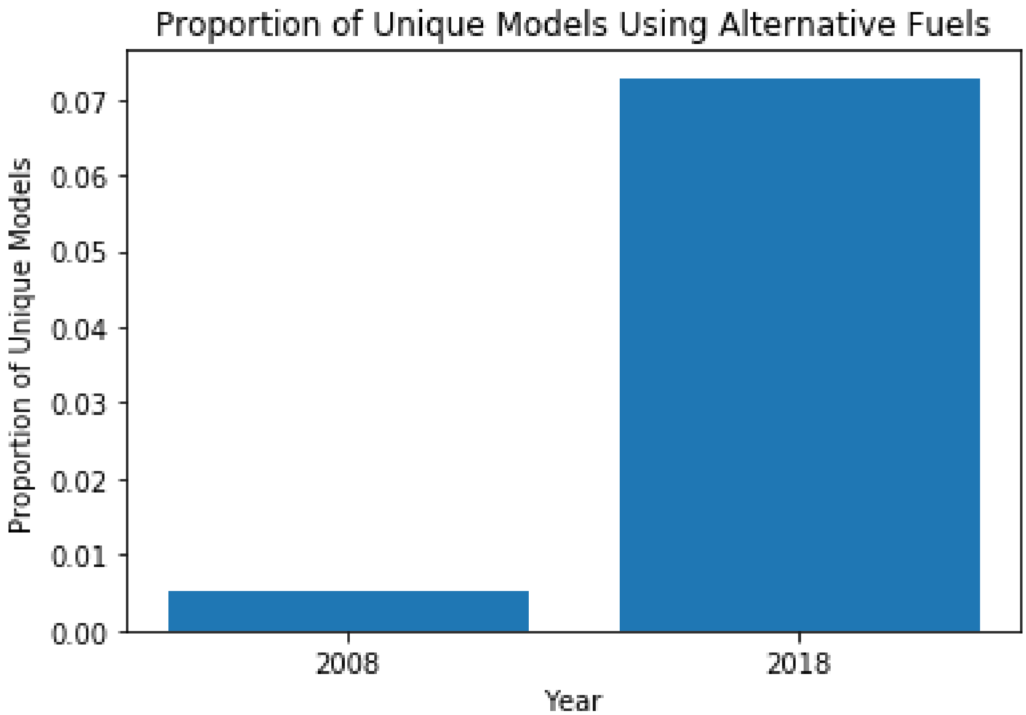
[9]:
```
# total unique models each year
total_08 = df_08.model.nunique()
total_18 = df_18.model.nunique()
total_08, total_18
```

[9]: (377, 357)

[10]:
```
prop_08 = alt_08/total_08
prop_18 = alt_18/total_18
prop_08, prop_18
```

[10]: (0.005305039787798408, 0.07282913165266107)

[11]:
```
plt.bar(["2008", "2018"], [prop_08, prop_18])
plt.title("Proportion of Unique Models Using Alternative Fuels")
plt.xlabel("Year")
plt.ylabel("Proportion of Unique Models");
```



1.0.2 Q2: How much have vehicle classes improved in fuel economy?

Let's look at the average fuel economy for each vehicle class for both years.

[12]:
```
veh_08 = df_08.groupby('veh_class').cmb_mpg.mean()
veh_08
```

[12]:
```
veh_class
SUV              18.471429
large car        18.509091
midsize car      21.601449
minivan          19.117647
pickup           16.277108
small car        21.105105
station wagon    22.366667
van              14.952381
Name: cmb_mpg, dtype: float64
```

[13]:
```
veh_18 = df_18.groupby('veh_class').cmb_mpg.mean()
veh_18
```

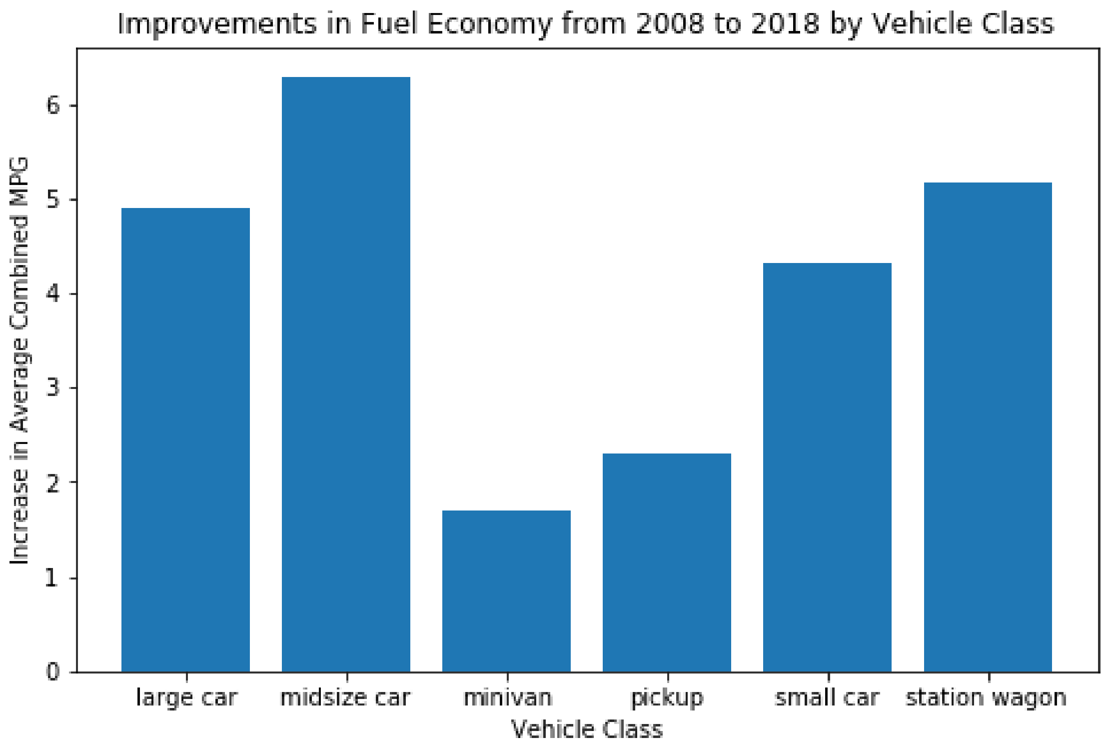
[13]:
```
veh_class
large car          23.409091
midsize car        27.884058
```

```
minivan            20.800000
pickup             18.589744
small SUV          24.074074
small car          25.421053
special purpose    18.500000
standard SUV       18.197674
station wagon      27.529412
Name: cmb_mpg, dtype: float64
```

[14]:
```python
# how much they've increased by for each vehicle class
inc = veh_18 - veh_08
inc
```

[14]:
```
veh_class
SUV                     NaN
large car          4.900000
midsize car        6.282609
minivan            1.682353
pickup             2.312635
small SUV               NaN
small car          4.315948
special purpose         NaN
standard SUV            NaN
station wagon      5.162745
van                     NaN
Name: cmb_mpg, dtype: float64
```

[15]:
```python
# only plot the classes that exist in both years
inc.dropna(inplace=True)
plt.subplots(figsize=(8, 5))
plt.bar(inc.index, inc)
plt.title('Improvements in Fuel Economy from 2008 to 2018 by Vehicle Class')
plt.xlabel('Vehicle Class')
plt.ylabel('Increase in Average Combined MPG');
```



1.0.3 Q3: What are the characteristics of SmartWay vehicles? Have they changed over time?

We can analyze this by filtering each dataframe by SmartWay classification and exploring these datasets.

[16]:
```
# smartway labels for 2008
df_08.smartway.unique()
```

[16]:
```
array(['no', 'yes'], dtype=object)
```

[17]:
```
# get all smartway vehicles in 2008
smart_08 = df_08.query('smartway == "yes"')
```

[18]:
```
# explore smartway vehicles in 2008
smart_08.describe()
```

[18]:
```
            displ         cyl  air_pollution_score    city_mpg     hwy_mpg  \
count  380.000000  380.000000           380.000000  380.000000  380.000000
mean     2.602895    4.826316             7.365789   20.984211   28.413158
std      0.623436    1.002025             1.148195    3.442672    3.075194
min      1.300000    4.000000             6.000000   17.000000   22.000000
25%      2.275000    4.000000             7.000000   19.000000   26.000000
```

```
50%       2.400000    4.000000                7.000000   20.000000   28.000000
75%       3.000000    6.000000                7.000000   22.000000   30.000000
max       5.000000    8.000000                9.500000   48.000000   45.000000

          cmb_mpg  greenhouse_gas_score
count  380.000000            380.000000
mean    23.736842              6.868421
std      3.060379              0.827338
min     20.000000              6.000000
25%     22.000000              6.000000
50%     23.000000              7.000000
75%     25.000000              7.000000
max     46.000000             10.000000
```

Use what you've learned so for to further explore this dataset on 2008 smartway vehicles.

[19]:
```
# smartway labels for 2018
df_18.smartway.unique()
```

[19]:
```
array(['No', 'Yes', 'Elite'], dtype=object)
```

[20]:
```
# get all smartway vehicles in 2018
smart_18 = df_18.query('smartway in ["Yes", "Elite"]')
```

[21]:
```
smart_18.describe()
```

[21]:
```
            displ         cyl  air_pollution_score    city_mpg     hwy_mpg  \
count  108.000000  108.000000           108.000000  108.000000  108.000000
mean     1.787963    3.935185             5.212963   34.907407   41.472222
std      0.408031    0.416329             1.798498   16.431982   13.095236
min      1.200000    3.000000             3.000000   25.000000   27.000000
25%      1.500000    4.000000             3.000000   28.000000   36.000000
50%      1.700000    4.000000             5.500000   28.500000   37.000000
75%      2.000000    4.000000             7.000000   31.250000   40.250000
max      3.500000    6.000000             7.000000  113.000000   99.000000

          cmb_mpg  greenhouse_gas_score
count  108.000000            108.000000
mean    37.361111              7.925926
std     14.848429              1.197378
min     26.000000              7.000000
25%     31.000000              7.000000
50%     32.000000              7.000000
75%     35.000000              9.000000
max    106.000000             10.000000
```

Use what you've learned so for to further explore this dataset on 2018 smartway vehicles.

1.0.4 Q4: What features are associated with better fuel economy?

You can explore trends between cmb_mpg and the other features in this dataset, or filter this dataset like in the previous question and explore the properties of that dataset. For example, you can select all vehicles that have the top 50% fuel economy ratings like this.

[22]:
```
top_08 = df_08.query('cmb_mpg > cmb_mpg.mean()')
top_08.describe()
```

[22]:

	displ	cyl	air_pollution_score	city_mpg	hwy_mpg	\
count	519.000000	519.000000	519.000000	519.000000	519.000000	
mean	2.667823	4.890173	6.998073	20.317919	27.603083	
std	0.665551	1.034856	1.159565	3.198257	3.051120	
min	1.300000	4.000000	4.000000	17.000000	20.000000	
25%	2.300000	4.000000	6.000000	18.000000	25.000000	
50%	2.500000	4.000000	7.000000	20.000000	27.000000	
75%	3.000000	6.000000	7.000000	21.000000	29.000000	
max	6.000000	8.000000	9.500000	48.000000	45.000000	

	cmb_mpg	greenhouse_gas_score
count	519.000000	519.000000
mean	22.992293	6.639692
std	2.926371	0.804935
min	20.000000	6.000000
25%	21.000000	6.000000
50%	22.000000	6.000000
75%	24.000000	7.000000
max	46.000000	10.000000

[23]:
```
top_18 = df_18.query('cmb_mpg > cmb_mpg.mean()')
top_18.describe()
```

[23]:

	displ	cyl	air_pollution_score	city_mpg	hwy_mpg	\
count	328.000000	328.000000	328.000000	328.000000	328.000000	
mean	1.964329	4.021341	4.856707	27.472561	35.304878	
std	0.398593	0.465477	1.860802	11.033692	9.024857	
min	1.200000	3.000000	1.000000	21.000000	27.000000	
25%	1.600000	4.000000	3.000000	23.000000	31.000000	
50%	2.000000	4.000000	5.000000	25.000000	33.000000	
75%	2.000000	4.000000	7.000000	28.000000	36.000000	
max	3.500000	6.000000	7.000000	113.000000	99.000000	

	cmb_mpg	greenhouse_gas_score
count	328.000000	328.000000
mean	30.411585	6.329268
std	10.081539	1.410358
min	25.000000	4.000000
25%	26.000000	5.000000

```
50%     28.000000                 6.000000
75%     31.000000                 7.000000
max    106.000000                10.000000
```

2