Feature Scaling and Selection

Enhancing our data exploration

Topics to be covered today;

- Scaling of existing metrics.
    - Min-max: “Normalization”. From 0 to 1.
    - Standardization: assuming normal distribution.

      Mean in 0 and standard deviation = 1.
- To be discarded metrics. Features correlated among themselves.
- Bonus track: feature engine.

Scaling of existing metrics

Why is it important to scale features?

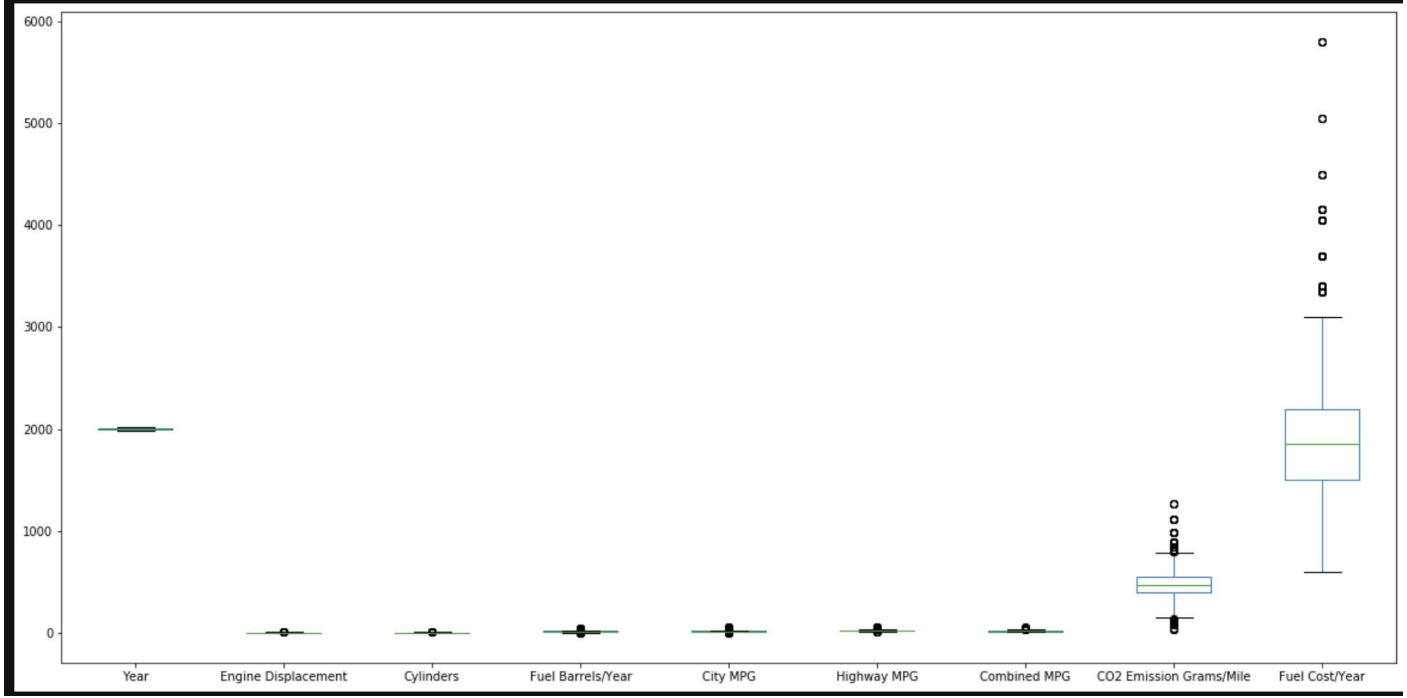
Different reasons behind:

- Ease generic visualizations about dispersions such as histograms.
- Increase accuracy of the models:
    - Some models such as the ones defined by “distance” are very sensitive to those metrics if not scaled: KNN, K-means.
    - Facilitate building of combined features when both of them are in different scales (grams vs kgs etc)

Ease generic visualizations about dispersions such as histograms


```
df.plot.box(figsize = (20,10))
```

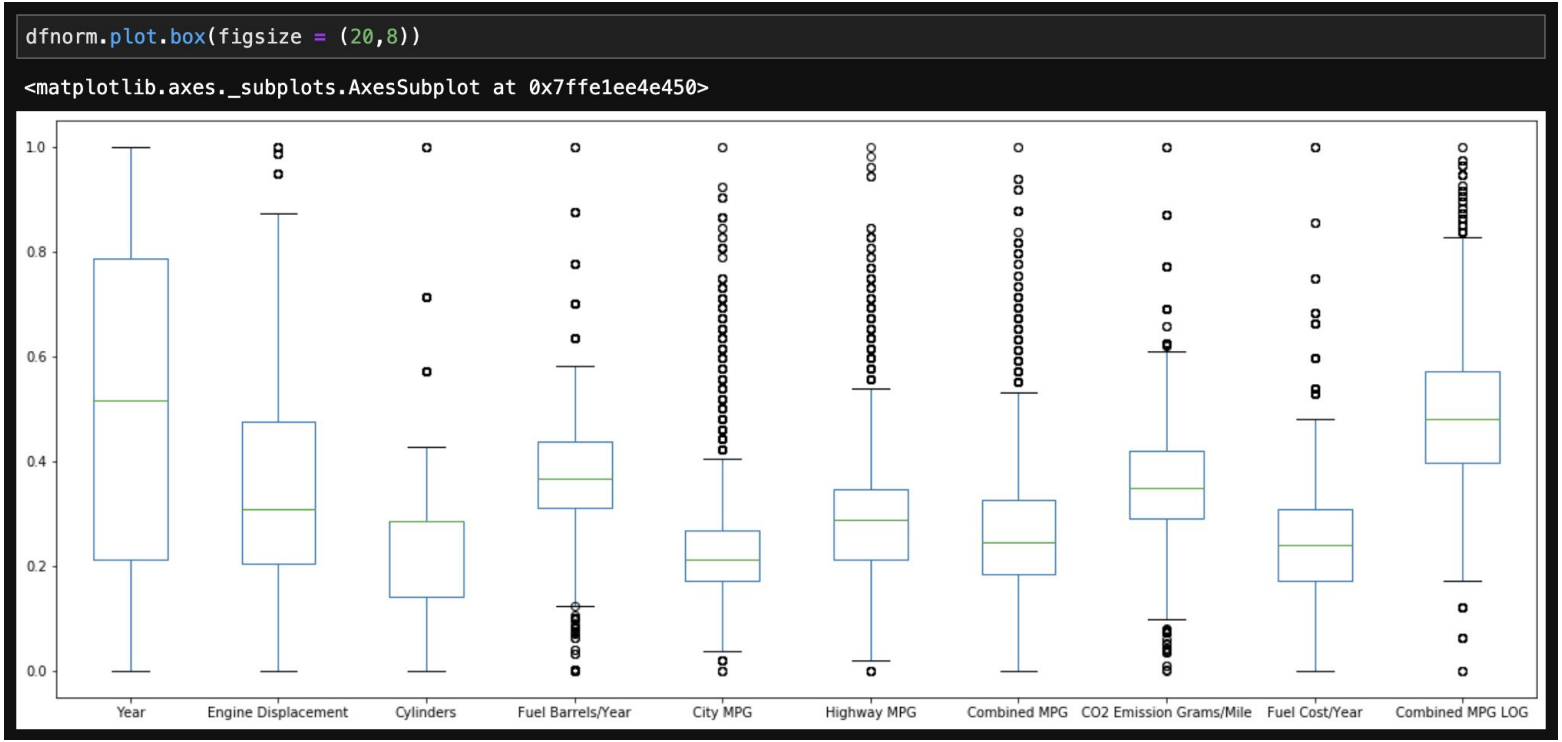
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffe1cc90ed0>
```



Ease generic visualizations about dispersions such as histograms


```
dfnorm.plot.box(figsize = (20,8))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffe1ee4e450>
```



Increases accuracy of all the models?

Note that scaling is not a magic feature. It doesn't lead to increase in accuracy in all models.

Ie; it will have no impact in OLS linear regressions but be very useful in the distance ones.

We will see an example afterwards.

Normalise or standardise? this is the question

- Standardise if you think the scaled column is following a normal distribution pattern.
- Recommendation; test both methods.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Normalise

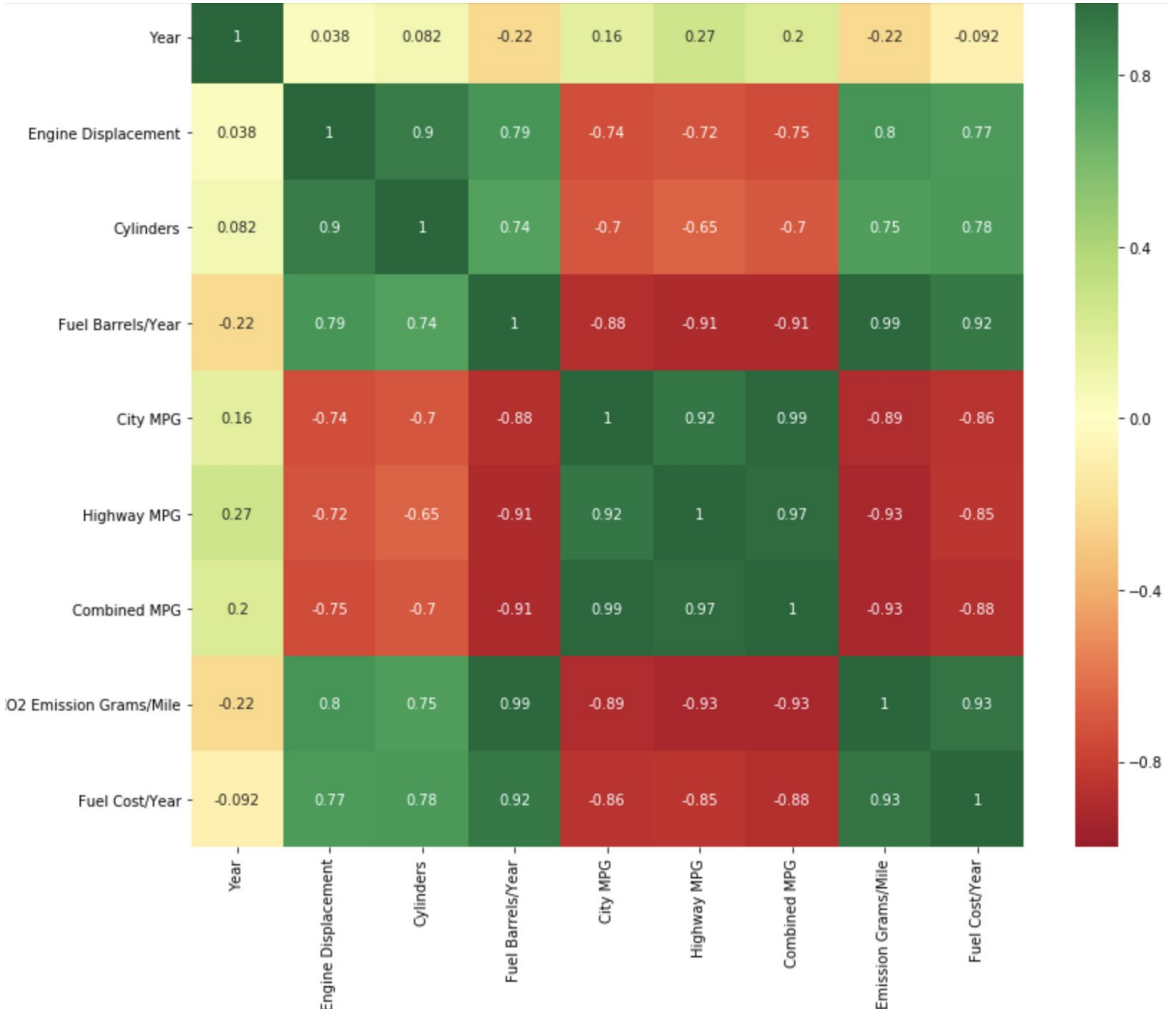
$$x_{new} = \frac{x - \mu}{\sigma}$$

Standardise

To be discarded metrics

What to discard?

- P-valued > 0.05.
- All features predicting a single record have to be different among themselves.
    - One useful way to detect similarity is correlation among features.



Let's see this example adding all the MPG ones

```
]: X=dfnumeric[['City MPG', 'Highway MPG', 'Combined MPG']]
   Y=dfnumeric['Fuel Cost/Year']
   X = sm.add_constant(X)
   results = sm.OLS(Y, X).fit()
   results.summary()
```

]:

OLS Regression Results

Dep. Variable:	Fuel Cost/Year	**R-squared:**	0.767
Model:	OLS	**Adj. R-squared:**	0.766
Method:	Least Squares	**F-statistic:**	3.934e+04
Date:	Thu, 25 Feb 2021	**Prob (F-statistic):**	0.00
Time:	12:44:23	**Log-Likelihood:**	-2.4879e+05
No. Observations:	35952	**AIC:**	4.976e+05
Df Residuals:	35948	**BIC:**	4.976e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3628.9646	5.397	672.364	0.000	3618.386	3639.544
City MPG	19.1152	2.425	7.882	0.000	14.362	23.869
Highway MPG	3.7787	1.359	2.781	0.005	1.115	6.442
Combined MPG	-108.5794	3.527	-30.786	0.000	-115.492	-101.667

Let's use only the combined one

[122]:
```
X=dfnumeric[['Combined MPG']]
Y=dfnumeric['Fuel Cost/Year']
X = sm.add_constant(X)
results = sm.OLS(Y, X).fit()
results.summary()
```

[122]:

OLS Regression Results

Dep. Variable:	Fuel Cost/Year	**R-squared:**	0.766
Model:	OLS	**Adj. R-squared:**	0.766
Method:	Least Squares	**F-statistic:**	1.176e+05
Date:	Thu, 25 Feb 2021	**Prob (F-statistic):**	0.00
Time:	12:42:01	**Log-Likelihood:**	-2.4883e+05
No. Observations:	35952	**AIC:**	4.977e+05
Df Residuals:	35950	**BIC:**	4.977e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3622.1727	5.206	695.800	0.000	3611.969	3632.376
Combined MPG	-86.7854	0.253	-342.999	0.000	-87.281	-86.289

Let’s see some code :D

Thank you for coming =)