Multicollinearity of Features

Multicollinearity of Features

```
## Introduction
In the previous section you have learned about correlation and covariance. Now that we're
moving towards regression models with multiple predictors, let's explore what it means when
predictors are correlated with each other.
```

Objectives

You will be able to:

- Understand multicollinearity and possible negative impacts on regression outcome
- Use heatmaps to visually inspect multicollinearity

Possible negative impacts of multicollinearity

As we're performing a regression analysis, the main goal is to identify the relationship between each predictor and the outcome variable. The interpretation of a regression coefficient is that it represents the average change in the dependent variable for each 1 unit change in a predictor, assuming that all the other predictor variables are kept constant. And it is exactly because of that reason that multicollinearity can cause problems.

Because the idea behind regression is that you can change one variable and keep the others constant, correlation is a problem, because it indicates that changes in one predictor are associated with changes in another one as well. Because of this, the estimates of the coefficients can have big fluctuations as a result of small changes in the model. As a result, you may not be able to trust the p-values associated with correlated predictors.

In this lecture, we'll learn about methods to identify multicollinearity, and will remove preditors that are highly correlated with others. You'll learn about other (and less ad-hoc) ways to deal with multicollinearity later on.

Identifying multicollinearity

To illustrate ways to identify multicollinearity, let's have a look at the "auto-mpg" data again.

In [1]:
```
## import numpy as np
import pandas as pd
data = pd.read_csv("auto-mpg.csv")
data['horsepower'].astype(str).astype(int) # don't worry about this for now
data.head()
```

Out[1]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

To understand the correlation structure of the predictors, we'll take a copy of the data but this time without the target variable (mpg) in it. Also, we'll remove the "car name" column as keeping those in won't lead to meaningful results.

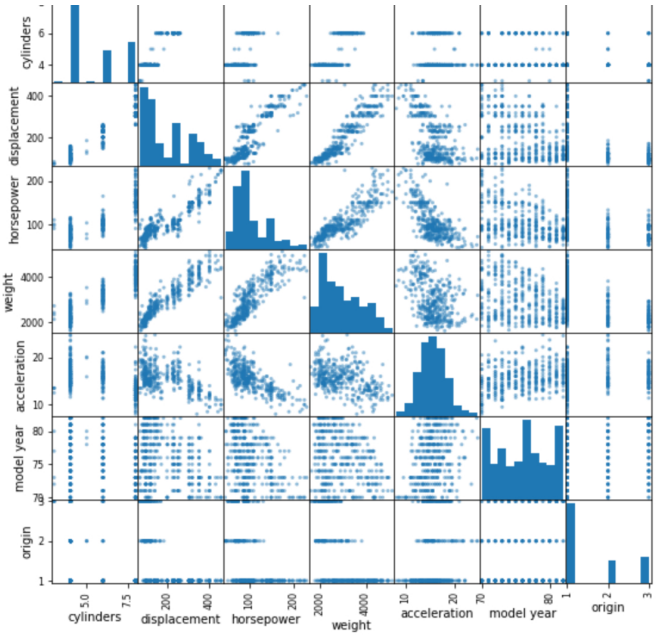
In [2]:
```
data_pred= data.iloc[:,1:8]
data_pred.head()
```

Out[2]:

	cylinders	displacement	horsepower	weight	acceleration	model year	origin
0	8	307.0	130	3504	12.0	70	1
1	8	350.0	165	3693	11.5	70	1
2	8	318.0	150	3436	11.0	70	1
3	8	304.0	150	3433	12.0	70	1
4	8	302.0	140	3449	10.5	70	1

For an initial idea on how the predictors relate, we can take a look at scatterplots between predictors. You can use Pandas to generate a scatter matrix as follows:

In [4]:
```
pd.plotting.scatter_matrix(data_pred,figsize  = [9, 9]);
```



This matrix has the cool feature that it returns scatterplots for relationships between two predictors, and histograms for a single feature on the diagonal. Have a quick look at this. When talking about correlation, what sort of scatter plots will catch our eye? You're right: the ones with scatter plots that reveal some sort of linear relationship. Seems like we can detect a few: weight and displacement seem to be highly correlatyed. Weight and horsepower as well, and (not surprisingly) displacement and horsepower. This is nice, but it would be hard to evaluate to examine each plot in detail when having a ton of features. Let's look at the correlation matrix instead. Instead of returning scatter plots and histograms, a correlation matrix returns pairwise correlations. Recall that correlations take a value between -1 and 1, -1 being a perfectly negative linear relationship, and +1 a perfectly positive linear relationship.

In [4]:
```
data_pred.corr()
```

Out[4]:

	cylinders	displacement	horsepower	weight	acceleration	model year	origin
cylinders	1.000000	0.950823	0.842983	0.897527	-0.504683	-0.345647	-0.568932
displacement	0.950823	1.000000	0.897257	0.932994	-0.543800	-0.369855	-0.614535
horsepower	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361	-0.455171
weight	0.897527	0.932994	0.864538	1.000000	-0.416839	-0.309120	-0.585005
acceleration	-0.504683	-0.543800	-0.689196	-0.416839	1.000000	0.290316	0.212746
model year	-0.345647	-0.369855	-0.416361	-0.309120	0.290316	1.000000	0.181528
origin	-0.568932	-0.614535	-0.455171	-0.585005	0.212746	0.181528	1.000000

Note that correlations on the diagonal are automatically equal to one as they represent correlations between a variable and the variable itself. So when do we consider a correlation "high"? Generally, a correlation with an absolute value around 0.7-0.8 or higher is considered a high correlation. If we take 0.75 as a cut-off, how many high correlations do we have?

In [5]:
```
abs(data_pred.corr()) > 0.75
```

Out[5]:

	cylinders	displacement	horsepower	weight	acceleration	model year	origin
cylinders	True	True	True	True	False	False	False
displacement	True	True	True	True	False	False	False
horsepower	True	True	True	True	False	False	False
weight	True	True	True	True	False	False	False
acceleration	False	False	False	False	True	False	False
model year	False	False	False	False	False	True	False
origin	False	False	False	False	False	False	True

It seems like the variables "cylinder", "displacement", "horsepower" and "weight" are all pretty highly correlated among each other. In our analysis, we would decide to remove three of those. Again here, it would be nice to have easier visuals in case when our predictor base grows (sometimes models have 100s of predictors!). A nice visualization of the correlation matrix is the heatmap.

In [6]:
```
import seaborn as sns
sns.heatmap(data_pred.corr(), center=0);
```