Correlation Coefficient

A measure of the linear correlation between two variables.
A coefficient has a value between +1 and −1, where 1 is total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation.

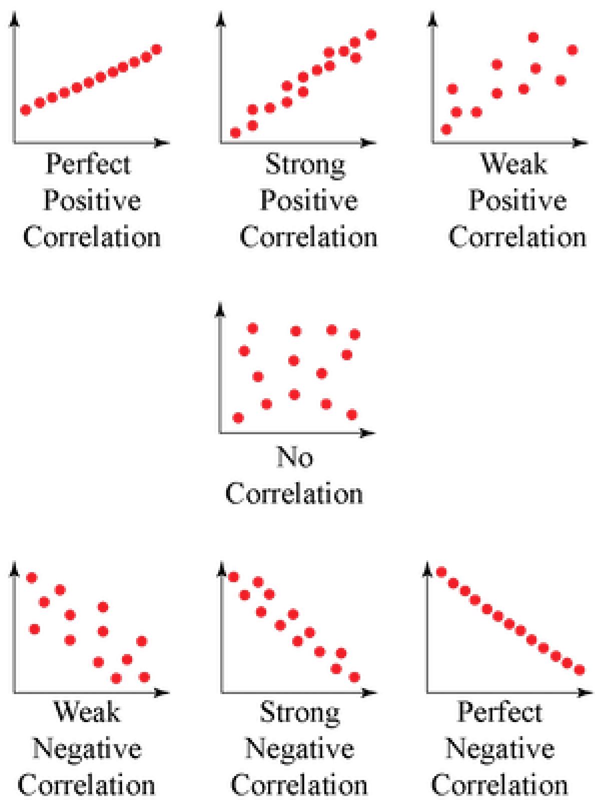


image source: https://d138zd1ktt9iqe.cloudfront.net/media/seo_landing_files/diksha-q-how-to-calculate-correlation-coefficient-01-1609233340.png

Pearson Correlation Coefficient

The Pearson correlation coefficient measures the linear relationship between two datasets. Strictly speaking, Pearson's correlation requires that each dataset be normally distributed.

Formula:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient

x_i = values of the x-variable in a sample

$\bar{x}$ = mean of the values of the x-variable

y_i = values of the y-variable in a sample

$\bar{y}$ = mean of the values of the y-variable

The 'auto-mpg' Dataset

In this lesson, we will use the 'auto-mpg' dataset, which was taken from the StatLib library maintained at Carnegie Mellon University. This dataset was used in the 1983 American Statistical Association Exposition and contains **398** automobile records from 1970 to 1982.

__Attribute Information:__

1. mpg*: miles per gallon
2. cylinders: Number of cylinders between 4 and 8
3. displacement: Engine displacement (cu. inches)
4. horsepower: Engine horsepower
5. weight: Vehicle weight (lbs.)
6. acceleration: Time to accelerate from 0 to 60 mph (sec.)
7. model year: Model year (modulo 100)
8. origin: Origin of car (1. American, 2. European, 3. Japanese)
9. model: car model name

*Target / outcome variable

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

Read Dataset

In [ ]:

```
df = pd.read_csv("https://raw.githubusercontent.com/ThammakornS/ProgStat/main/datas
df.head()
```

Out[ ]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	mod
0	9.0	8	304.0	193.0	4732	18.5	70	1	hi 120(
1	10.0	8	307.0	200.0	4376	15.0	70	1	chevy c2
2	10.0	8	360.0	215.0	4615	14.0	70	1	ford f2!
3	11.0	8	400.0	150.0	4997	14.0	73	1	chevrol impa
4	11.0	8	350.0	180.0	3664	11.0	73	1	oldsmobi omeg

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   year          398 non-null    int64  
 7   origin        398 non-null    int64  
 8   model         398 non-null    object 
dtypes: float64(4), int64(4), object(1)
memory usage: 28.1+ KB
```

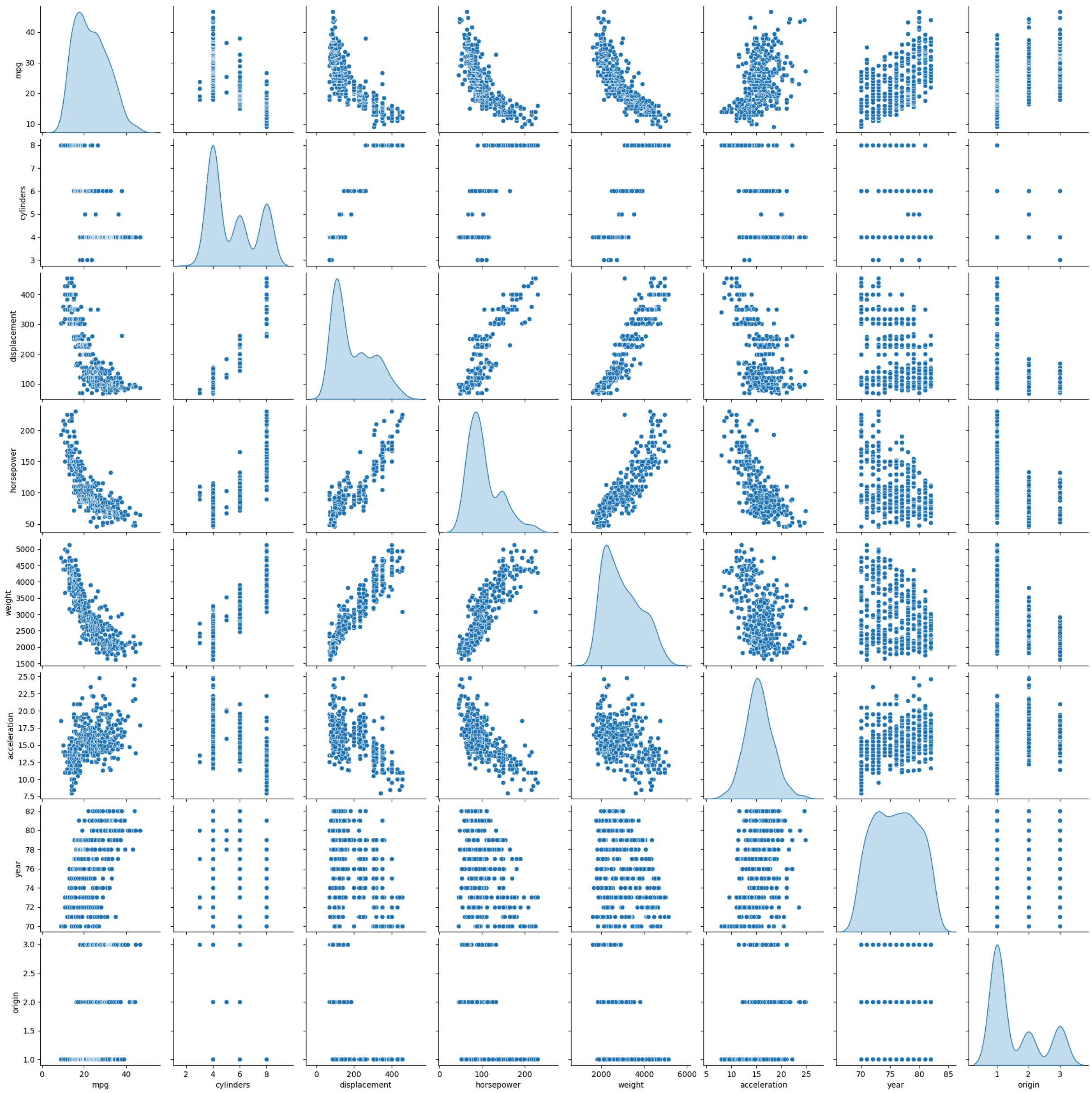
Data Overview

In [ ]:

```
sns.pairplot(data=df.dropna(),
             diag_kind="kde")
```

Out[ ]:

```
<seaborn.axisgrid.PairGrid at 0x7a3cebeefaa0>
```



Calculates a Pearson Correlation Coefficient and the P-Value for Testing Non-Correlation

If we *__assume__* that all data are normally distributed, we can calculate the Pearson correlation coefficient and the p-value for testing non-correlation using pearsonr().

The p-value roughly represents the probability of an uncorrelated system producing datasets with a Pearson correlation at least as extreme as the one computed from these datasets. While p-values are not entirely reliable, they are generally reasonable for datasets larger than approximately 500.

For displacement:

In [ ]:
```
stats.pearsonr(x=df.mpg,
               y=df.displacement)
```

Out[ ]:
```
PearsonRResult(statistic=np.float64(-0.804202824805898), pvalue=np.float64(1.65588
8910192966e-91))
```

For all columns except 'model':

In [ ]:
```
coeffs = {}
coeffs['var'] = df.columns[:-1] # exclude column 'model'
coeffs['coeff'] = []
coeffs['p-value'] = []
for c in df.columns[:-1]:
    if(df[c].isnull().any()):
        coeff, p = stats.pearsonr(x=df.dropna().mpg, y=df.dropna()[c])
    else:
        coeff, p = stats.pearsonr(x=df.mpg, y=df[c])
    coeffs['coeff'].append(coeff)
    coeffs['p-value'].append(p)

coeffs = pd.DataFrame(coeffs)
coeffs.sort_values(by='coeff', ascending=False)
```

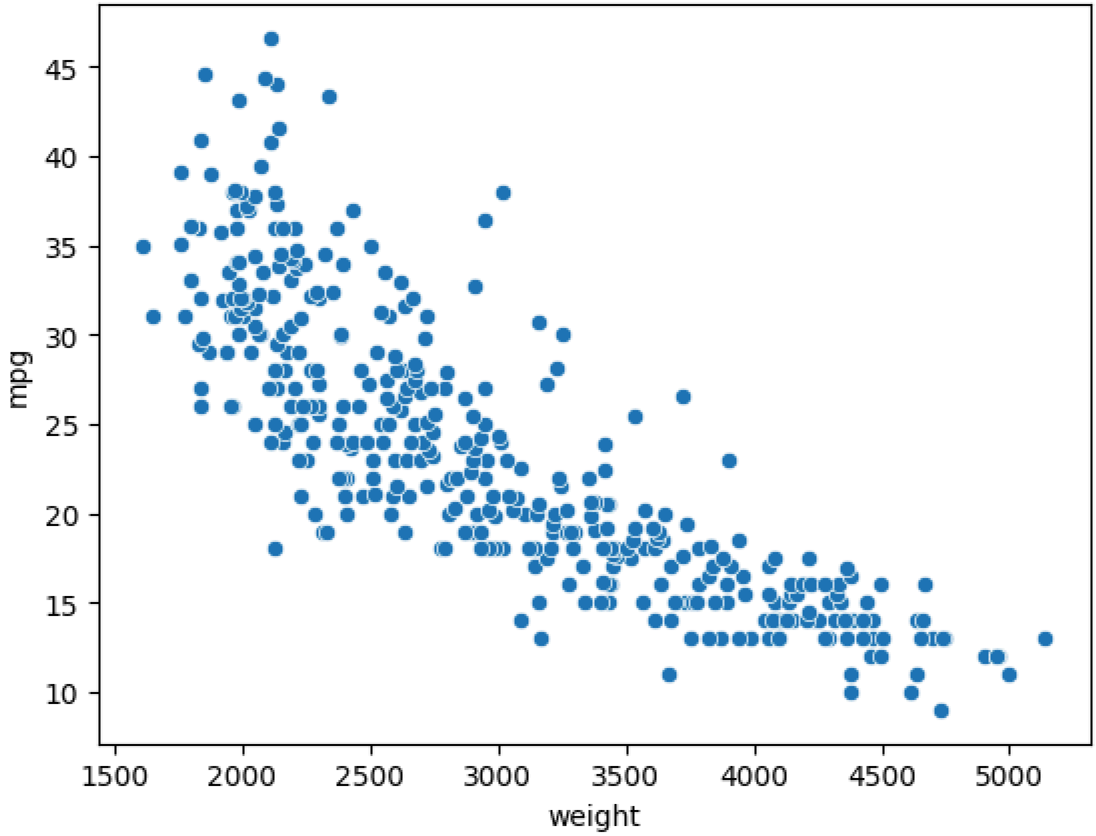
Out[ ]:

	var	coeff	p-value
0	mpg	1.000000	0.000000e+00
6	year	0.579267	4.844936e-37
7	origin	0.563450	1.011482e-34
5	acceleration	0.420289	1.823092e-18
1	cylinders	-0.775396	4.503992e-81
3	horsepower	-0.778427	7.031989e-81
2	displacement	-0.804203	1.655889e-91
4	weight	-0.831741	2.972800e-103

We can see that 'weight' has highest absolute correlation coefficient with 'mpg'. So, let's see the scatter plot between these two variables.

In [ ]:
```
sns.scatterplot(data=df,
                x='weight',
                y='mpg')
```

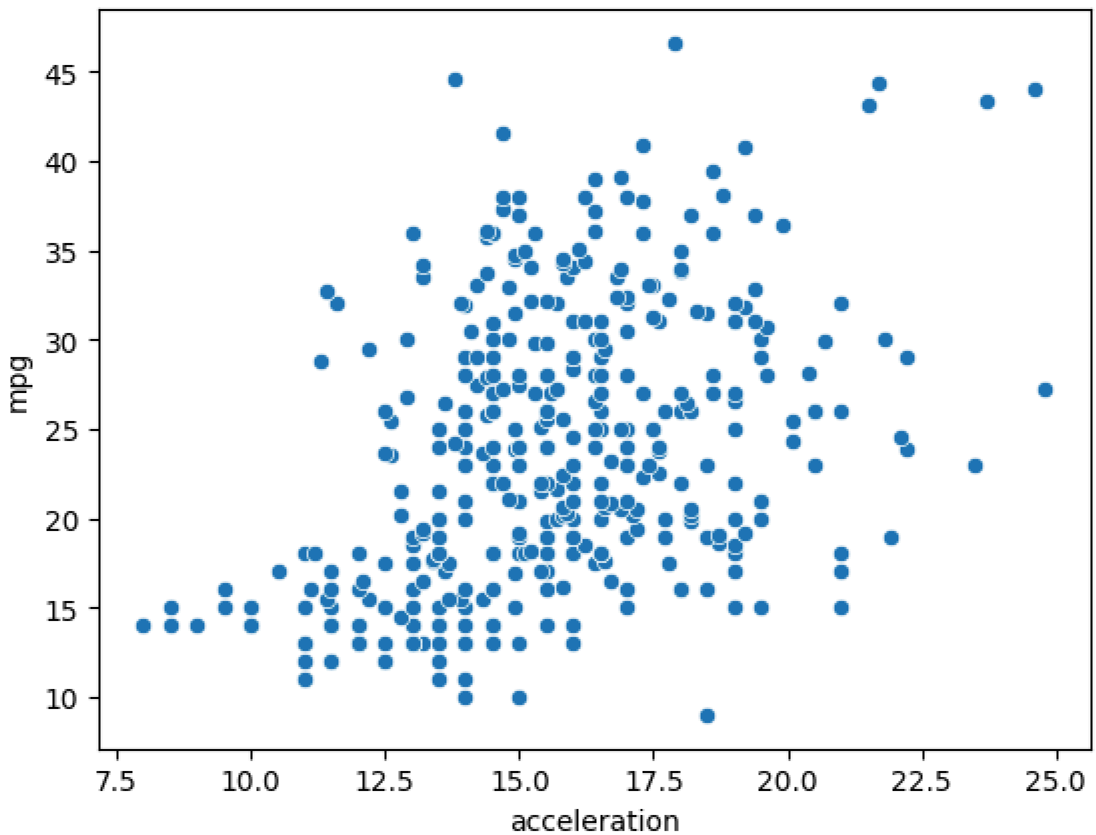
Out[ ]:
```
<Axes: xlabel='weight', ylabel='mpg'>
```



In case of 'acceleration':

In [ ]:
```
sns.scatterplot(data=df,
                x='acceleration',
                y='mpg')
```

Out[ ]: <Axes: xlabel='acceleration', ylabel='mpg'>



Alternatively, we can use method corr() provided by Pandas:

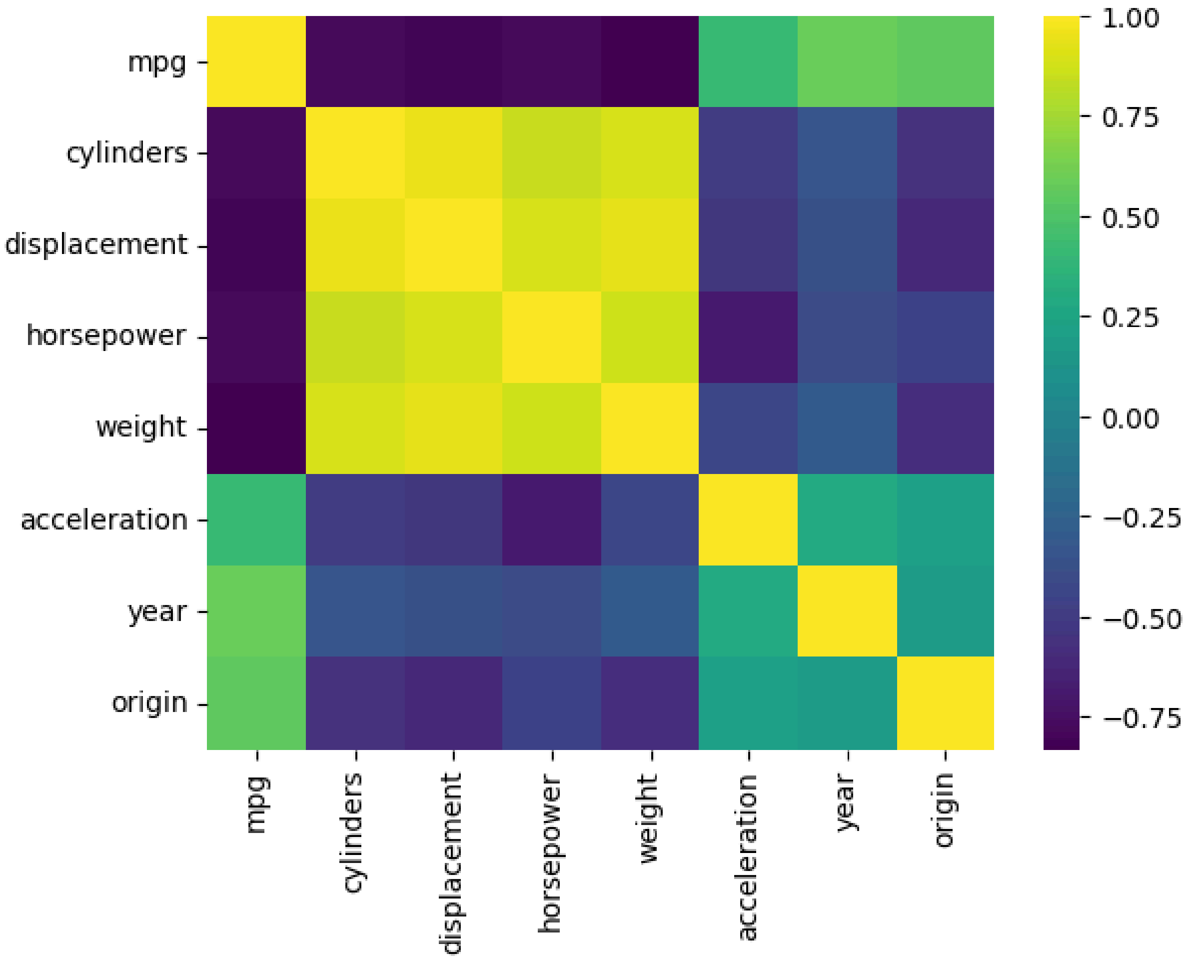
In [ ]:
```
corr = df.corr(method='pearson', numeric_only=True)
corr
```

Out[ ]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	
mpg	1.000000	-0.775396	-0.804203	-0.778427	-0.831741	0.420289	0.5
cylinders	-0.775396	1.000000	0.950721	0.842983	0.896017	-0.505419	-0.3
displacement	-0.804203	0.950721	1.000000	0.897257	0.932824	-0.543684	-0.3
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.4
weight	-0.831741	0.896017	0.932824	0.864538	1.000000	-0.417457	-0.3
acceleration	0.420289	-0.505419	-0.543684	-0.689196	-0.417457	1.000000	0.2
year	0.579267	-0.348746	-0.370164	-0.416361	-0.306564	0.288137	1.0
origin	0.563450	-0.562543	-0.609409	-0.455171	-0.581024	0.205873	0.1

In [ ]:
```
sns.heatmap(corr, cmap='viridis')
```

Out[ ]:
```
<Axes: >
```



However, p-values are not generated by corr().

Check Normality

Since Pearson’s correlation requires that each dataset be normally distributed, we first need to check normality for each feature.

In [ ]:
```
for c in df.columns[:-1]: # exclude column 'model'
    print(c+':')
    print(stats.shapiro(df.dropna()[c]))
    print('')
```

```
mpg:
ShapiroResult(statistic=np.float64(0.9671696219783011), pvalue=np.float64(1.04944070
6338603e-07))

cylinders:
ShapiroResult(statistic=np.float64(0.7506596822226748), pvalue=np.float64(6.88024174
4029972e-24))

displacement:
ShapiroResult(statistic=np.float64(0.8818359417766877), pvalue=np.float64(8.98363711
4582913e-17))

horsepower:
ShapiroResult(statistic=np.float64(0.9040974881446456), pvalue=np.float64(5.02206929
07909105e-15))

weight:
ShapiroResult(statistic=np.float64(0.9414660744821142), pvalue=np.float64(2.60168580
76512468e-11))

acceleration:
ShapiroResult(statistic=np.float64(0.9918671364554664), pvalue=np.float64(0.03052886
200020249))

year:
ShapiroResult(statistic=np.float64(0.9469665948027586), pvalue=np.float64(1.22261700
42045836e-10))

origin:
ShapiroResult(statistic=np.float64(0.6737641638584284), pvalue=np.float64(8.80204422
9203731e-27))
```

All p-values are <= 0.05.
This means all variables are not normally distributed.

Spearman Correlation Coefficient

The Spearman rank-order correlation coefficient is a nonparametric measure of the monotonic relationship between two datasets. Unlike the Pearson correlation, the Spearman correlation ***does not assume that the datasets are normally distributed***.

Like other correlation coefficients, it ranges from -1 to +1, with 0 indicating no correlation. A correlation of -1 or +1 signifies a perfect monotonic relationship. A positive correlation means that as x increases, y also increases, while a negative correlation means that as x increases, y decreases.

Formula:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

ρ = Spearman's rank correlation coefficient
d_i = difference between the two ranks of each observation
n = number of observations

Calculate a Spearman Correlation Coefficient with Associated P-Value

Similar to Pearson's correlation, the p-value roughly represents the probability of an uncorrelated system producing datasets with a Spearman correlation at least as extreme as the one computed from these datasets. While p-values are not entirely reliable, they are generally reasonable for datasets larger than approximately 500.

In [ ]:
```
coeffs, pvals = stats.spearmanr(df.dropna().iloc[:,:-1])
```

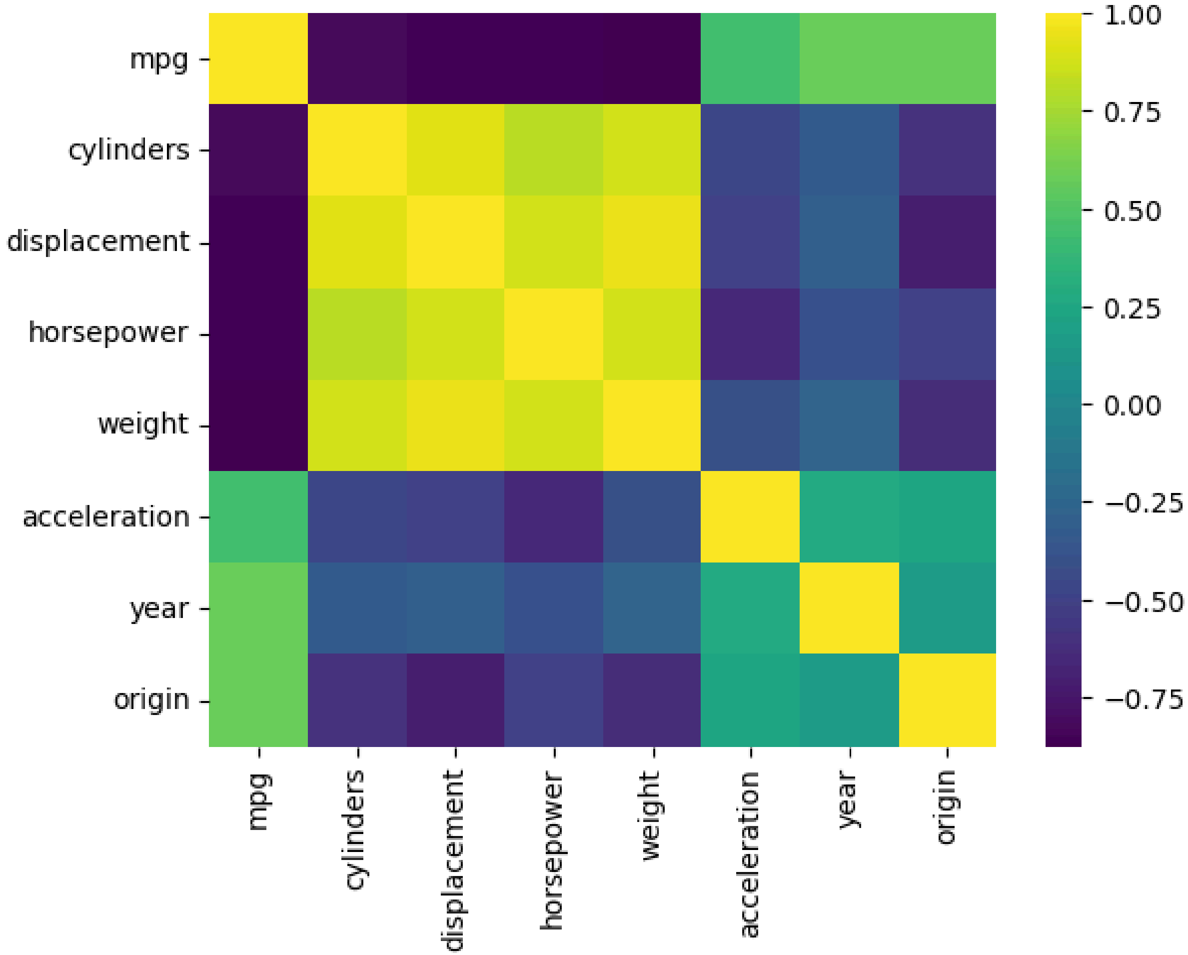
In [ ]:
```
coeffs = pd.DataFrame(coeffs,
                      columns=df.columns[:-1],
                      index=df.columns[:-1])
coeffs
```

Out[ ]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	
mpg	1.000000	-0.823175	-0.855234	-0.853616	-0.875585	0.441539	0.5
cylinders	-0.823175	1.000000	0.913566	0.816188	0.875972	-0.476266	-0.3
displacement	-0.855234	0.913566	1.000000	0.876171	0.945630	-0.499403	-0.3
horsepower	-0.853616	0.816188	0.876171	1.000000	0.878819	-0.658142	-0.3
weight	-0.875585	0.875972	0.945630	0.878819	1.000000	-0.405109	-0.2
acceleration	0.441539	-0.476266	-0.499403	-0.658142	-0.405109	1.000000	0.2
year	0.574841	-0.331087	-0.306582	-0.389498	-0.280981	0.278306	1.0
origin	0.580482	-0.610468	-0.709573	-0.508989	-0.631371	0.227406	0.1

In [ ]:
```
sns.heatmap(coeffs, cmap='viridis')
```

Out[ ]: <Axes: >



In [ ]:
```
pd.DataFrame(pvals,
             columns=df.columns[:-1],
             index=df.columns[:-1])
```

Out[ ]:

	mpg	cylinders	displacement	horsepower	weight	acc
mpg	0.000000e+00	6.649861e-98	2.195778e-113	1.619383e-112	2.662378e-125	3.90
cylinders	6.649861e-98	0.000000e+00	1.810859e-154	6.065373e-95	1.509812e-125	1.37
displacement	2.195778e-113	1.810859e-154	0.000000e+00	1.126737e-125	2.463170e-192	4.06
horsepower	1.619383e-112	6.065373e-95	1.126737e-125	0.000000e+00	2.182674e-127	5.15
weight	2.662378e-125	1.509812e-125	2.463170e-192	2.182674e-127	0.000000e+00	6.48
acceleration	3.903604e-20	1.374921e-23	4.061210e-26	5.157840e-50	6.484246e-17	0.00
year	7.465532e-36	1.757611e-11	5.625748e-10	1.190939e-15	1.515585e-08	2.09
origin	1.097626e-36	2.139199e-41	2.879842e-61	3.183740e-27	5.457059e-45	5.42

Alternatively, we can use method corr() provided by Pandas:

In [ ]:

```
corr = df.corr(method='spearman', numeric_only=True)
corr
```

Out[ ]:

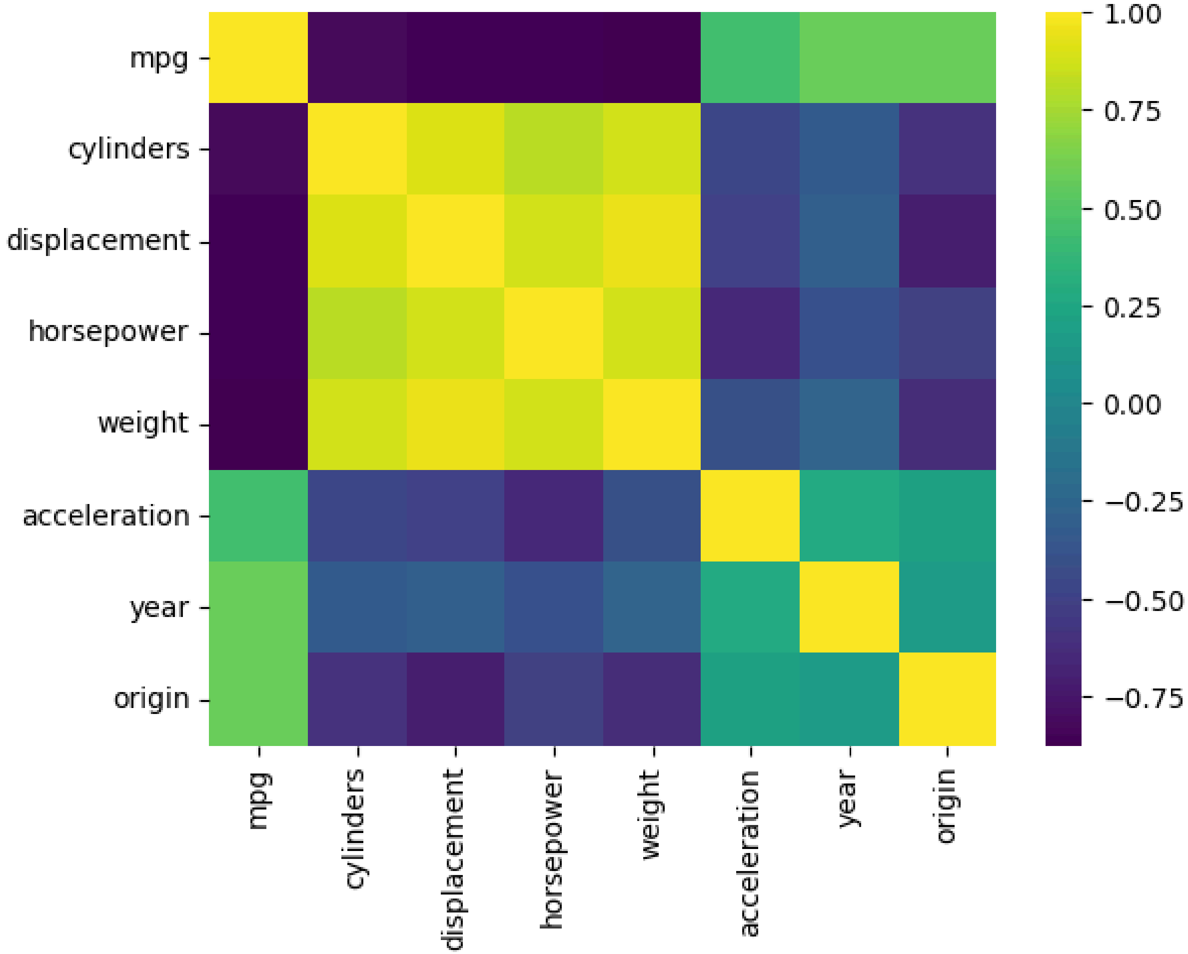
	mpg	cylinders	displacement	horsepower	weight	acceleration	
mpg	1.000000	-0.821864	-0.855692	-0.853616	-0.874947	0.438677	0.5
cylinders	-0.821864	1.000000	0.911876	0.816188	0.873314	-0.474189	-0.3
displacement	-0.855692	0.911876	1.000000	0.876171	0.945986	-0.496512	-0.3
horsepower	-0.853616	0.816188	0.876171	1.000000	0.878819	-0.658142	-0.3
weight	-0.874947	0.873314	0.945986	0.878819	1.000000	-0.404550	-0.2
acceleration	0.438677	-0.474189	-0.496512	-0.658142	-0.404550	1.000000	0.2
year	0.573469	-0.335012	-0.305257	-0.389498	-0.277015	0.274632	1.0
origin	0.580694	-0.604550	-0.707197	-0.508989	-0.628434	0.220574	0.1

In [ ]:

```
sns.heatmap(corr, cmap='viridis')
```

Out[ ]:

```
<Axes: >
```



Key Takeaways

		Outcome Variable	
		Numerical	Categorical
Feature Variable	Numerical	Correlation	Test of means
	Categorical	Test of means	Chi-squared test

In [ ]: