Tutorial 5 - Seaborn & Clustering

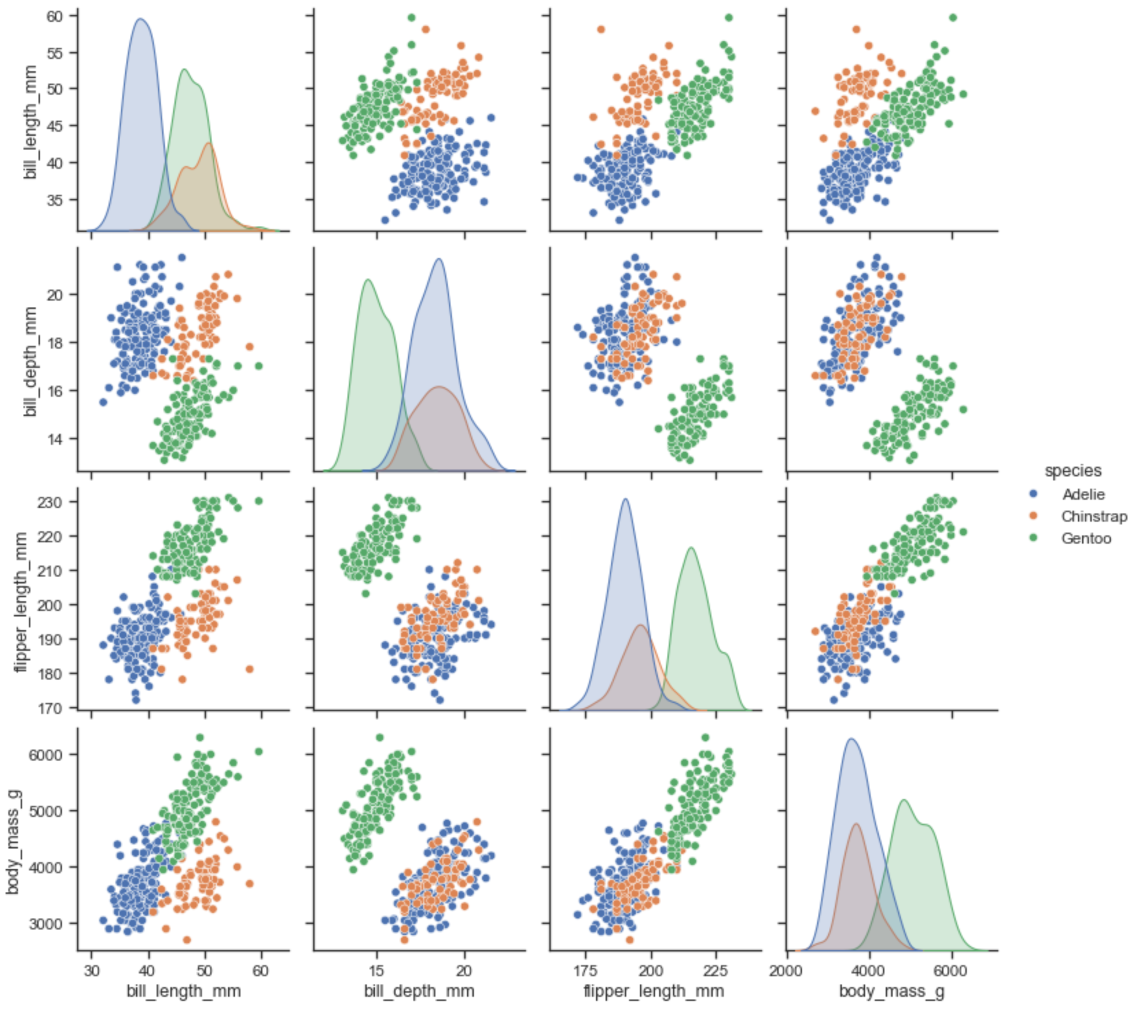
Outline

- Introduction and installation
- Data visualization with `catplot` , `relplot` and `pairplot`
- Correlation matrix
- Clustering
- Quick Markdown & LaTeX syntax

Seaborn

- Documentation
- Installation: `pip install seaborn` or `conda install seaborn -c conda-forge`

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn also provides better integration with Pandas data structures.



Bar Plot

In [ ]:
```
import pandas as pd
import seaborn as sns
```

```
C:\Users\artui\AppData\Local\Temp\ipykernel_30100\432526209.py:1: DeprecationWarnin
g:
Pyarrow will become a required dependency of pandas in the next major release of pan
das (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better inte
roperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

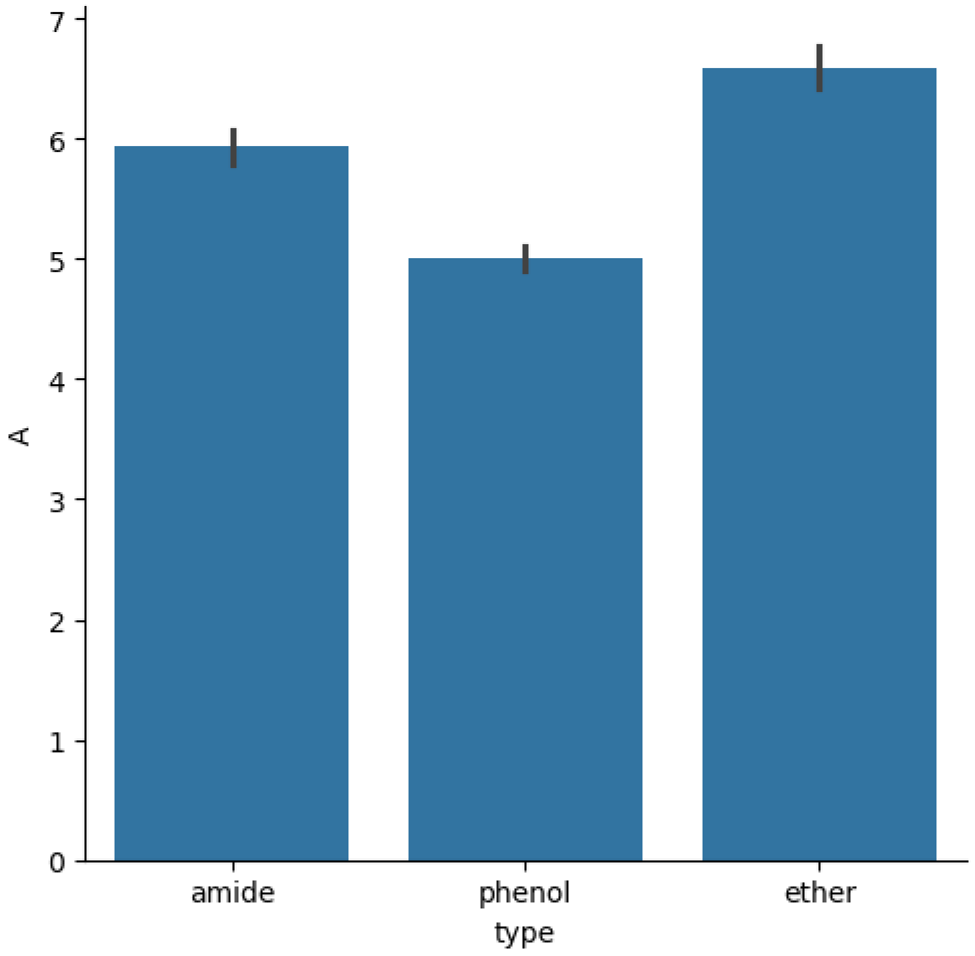
In [ ]:
```
df = pd.read_csv("Datasets/compounds.csv")
df.head()
```

```
Out[ ]:
```

	A	B	C	D	type	Start assignment
0	6.4	2.9	4.3	1.3	amide	1
1	5.7	4.4	1.5	0.4	phenol	2
2	6.7	3.0	5.2	2.3	ether	0
3	5.8	2.8	5.1	2.4	ether	1
4	6.4	3.2	5.3	2.3	ether	0

```
In [ ]: sns.catplot(data=df, x='type', y='A', kind='bar')
```

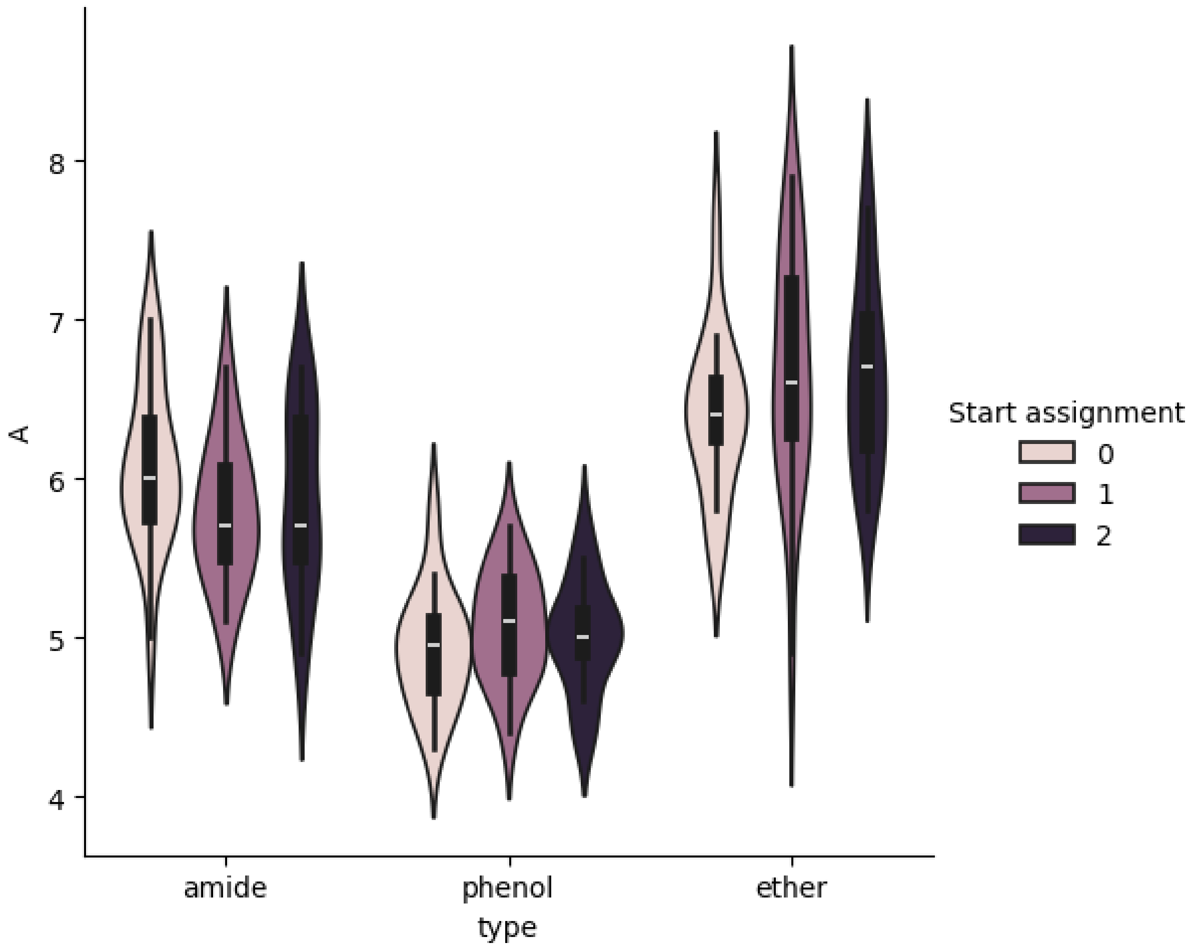
```
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x2ae01b6f4a0>
```



Violin plot

```
In [ ]: sns.catplot(data=df, kind='violin', x='type', y='A', hue='Start assignment')
```

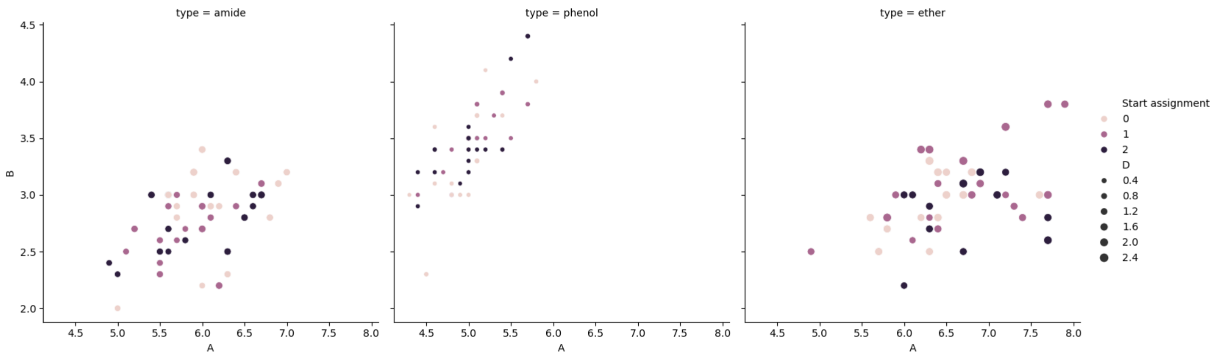
```
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x2ae035febd0>
```



Scatter plot

```
In [ ]: sns.relplot(data=df, x='A', y='B', col='type', hue='Start assignment', size='D')
```

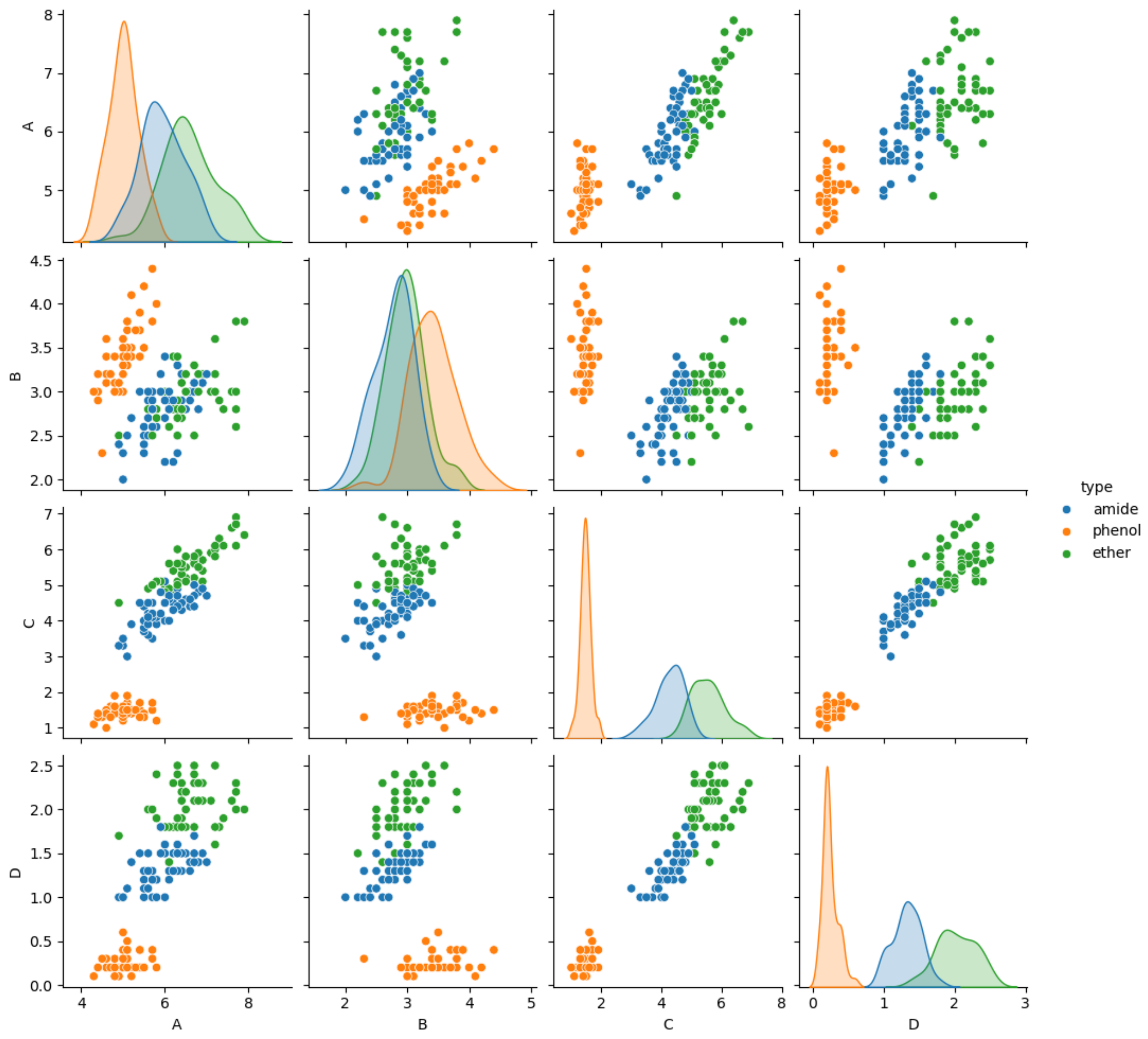
```
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x2ae036b7590>
```



Pair Plot

```
In [ ]: newdf = df.drop(['Start assignment'], axis=1)
        sns.pairplot(data=newdf, hue='type')
```

```
Out[ ]: <seaborn.axisgrid.PairGrid at 0x2ae04388830>
```



Correlation matrix

The **correlation coefficient** between two random variables X and Y are defined as:

$$\frac{\mathrm{Cov(X, Y)}}{\sqrt{\mathrm{Var(X)Var(Y)}}} = \frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{[\sum_i (X_i - \bar{X})^2][\sum_i (Y_i - \bar{Y})^2]}}$$

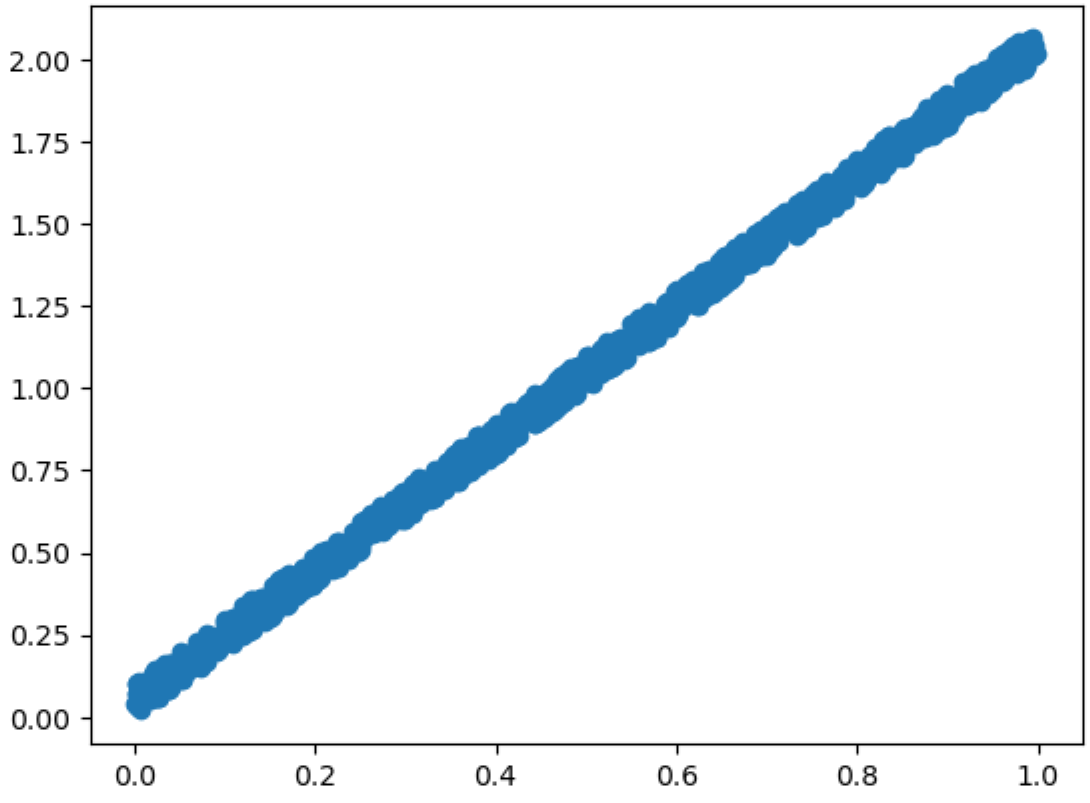
The correlation coefficient should be in the range of $[-1, 1]$. When $X = Y$, the correlation coefficient will be 1, and when $X = -Y$ the correlation coefficient will be -1.

In [ ]:
```
import numpy as np


X = np.random.random(1000)
Y = np.random.random(1000)
Y = 2 * X + np.random.random(1000) * 0.1

def corrcoef(X, Y):
    X_shift = X - np.mean(X)
    Y_shift = Y - np.mean(Y)
    return np.sum(X_shift * Y_shift) / np.sqrt(np.sum(X_shift ** 2) * np.sum(Y_shif

corrcoef(X, Y)
```

Out[ ]:
```
0.9986565748511288
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.scatter(X, Y)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2ae065cd130>
```



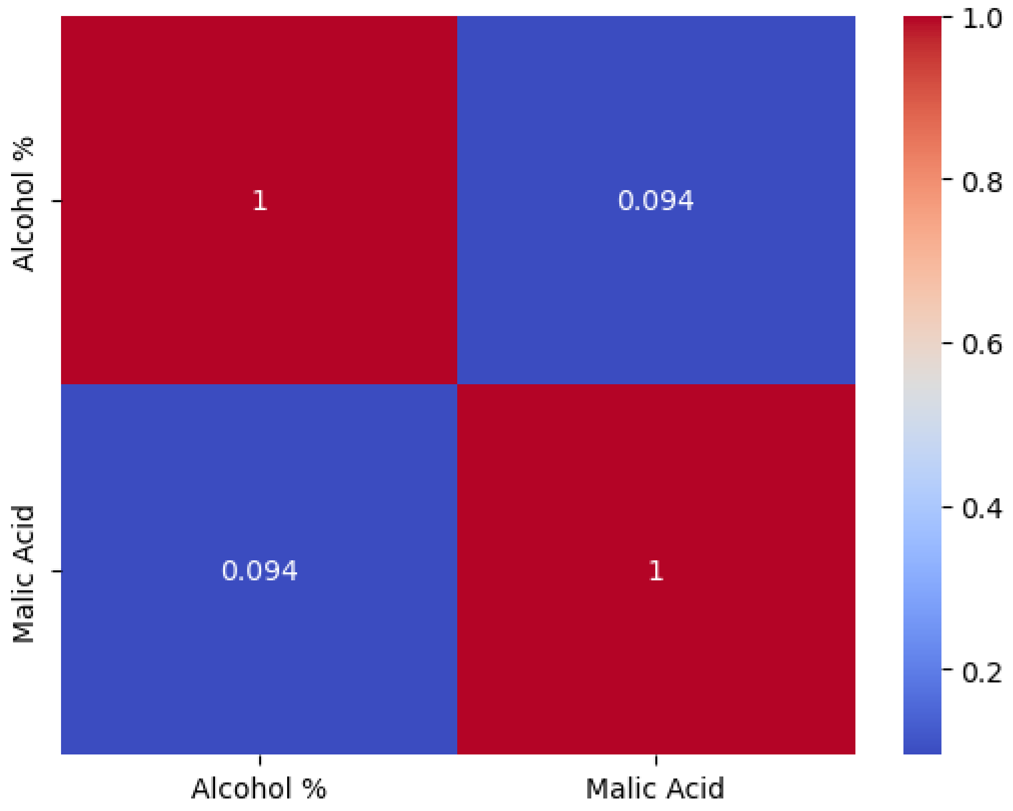
In [ ]:
```
wines = pd.read_csv("Datasets/wines.csv").iloc[:, :-2]
features = wines.values
features.shape
```

In [ ]:
```
# use numpy to calculate corr coef
corrmat = np.corrcoef(features.T)
corrmat
```

```
Out[ ]: array([[ 1.        ,  0.09439694,  0.2115446 , -0.31023514,  0.27079823,
                 0.28910112,  0.23681493, -0.15592947,  0.13669791,  0.5463642 ,
                -0.0717472 ,  0.07234319,  0.64372004],
               [ 0.09439694,  1.        ,  0.16404547,  0.2885004 , -0.0545751 ,
                -0.335167  , -0.41100659,  0.29297713, -0.22074619,  0.24898534,
                -0.56129569, -0.36871043, -0.19201056],
               [ 0.2115446 ,  0.16404547,  1.        ,  0.44336719,  0.28658669,
                 0.12897954,  0.11507728,  0.18623045,  0.00965194,  0.25888726,
                -0.07466689,  0.00391123,  0.22362626],
               [-0.31023514,  0.2885004 ,  0.44336719,  1.        , -0.08333309,
                -0.32111332, -0.35136986,  0.36192172, -0.19732684,  0.01873198,
                -0.27395522, -0.27676855, -0.44059693],
               [ 0.27079823, -0.0545751 ,  0.28658669, -0.08333309,  1.        ,
                 0.21440123,  0.19578377, -0.25629405,  0.23644061,  0.19995001,
                 0.0553982 ,  0.06600394,  0.39335085],
               [ 0.28910112, -0.335167  ,  0.12897954, -0.32111332,  0.21440123,
                 1.        ,  0.8645635 , -0.4499353 ,  0.61241308, -0.05513642,
                 0.43368134,  0.69994936,  0.49811488],
               [ 0.23681493, -0.41100659,  0.11507728, -0.35136986,  0.19578377,
                 0.8645635 ,  1.        , -0.53789961,  0.65269177, -0.1723794 ,
                 0.54347857,  0.7871939 ,  0.49419313],
               [-0.15592947,  0.29297713,  0.18623045,  0.36192172, -0.25629405,
                -0.4499353 , -0.53789961,  1.        , -0.3658451 ,  0.13905701,
                -0.26263963, -0.5032696 , -0.31138519],
               [ 0.13669791, -0.22074619,  0.00965194, -0.19732684,  0.23644061,
                 0.61241308,  0.65269177, -0.3658451 ,  1.        , -0.02524993,
                 0.29554425,  0.5190671 ,  0.3304167 ],
               [ 0.5463642 ,  0.24898534,  0.25888726,  0.01873198,  0.19995001,
                -0.05513642, -0.1723794 ,  0.13905701, -0.02524993,  1.        ,
                -0.52181319, -0.42881494,  0.31610011],
               [-0.0717472 , -0.56129569, -0.07466689, -0.27395522,  0.0553982 ,
                 0.43368134,  0.54347857, -0.26263963,  0.29554425, -0.52181319,
                 1.        ,  0.56546829,  0.23618345],
               [ 0.07234319, -0.36871043,  0.00391123, -0.27676855,  0.06600394,
                 0.69994936,  0.7871939 , -0.5032696 ,  0.5190671 , -0.42881494,
                 0.56546829,  1.        ,  0.31276108],
               [ 0.64372004, -0.19201056,  0.22362626, -0.44059693,  0.39335085,
                 0.49811488,  0.49419313, -0.31138519,  0.3304167 ,  0.31610011,
                 0.23618345,  0.31276108,  1.        ]])
```

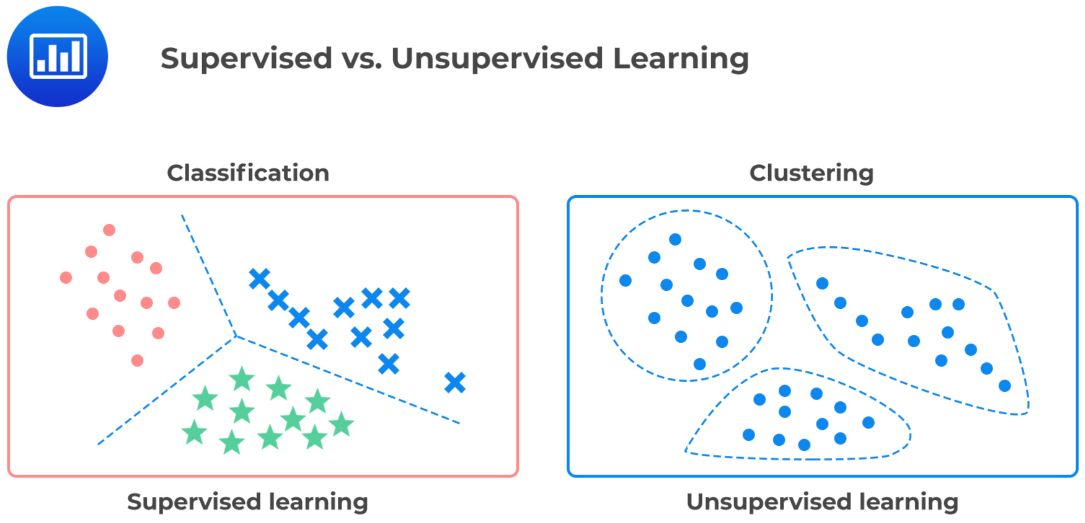
```
In [ ]: # seaborn vis
        sns.heatmap(corrmat[:2, :2], cmap='coolwarm', xticklabels=wines.columns[:2], ytickl
```

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



Clustering

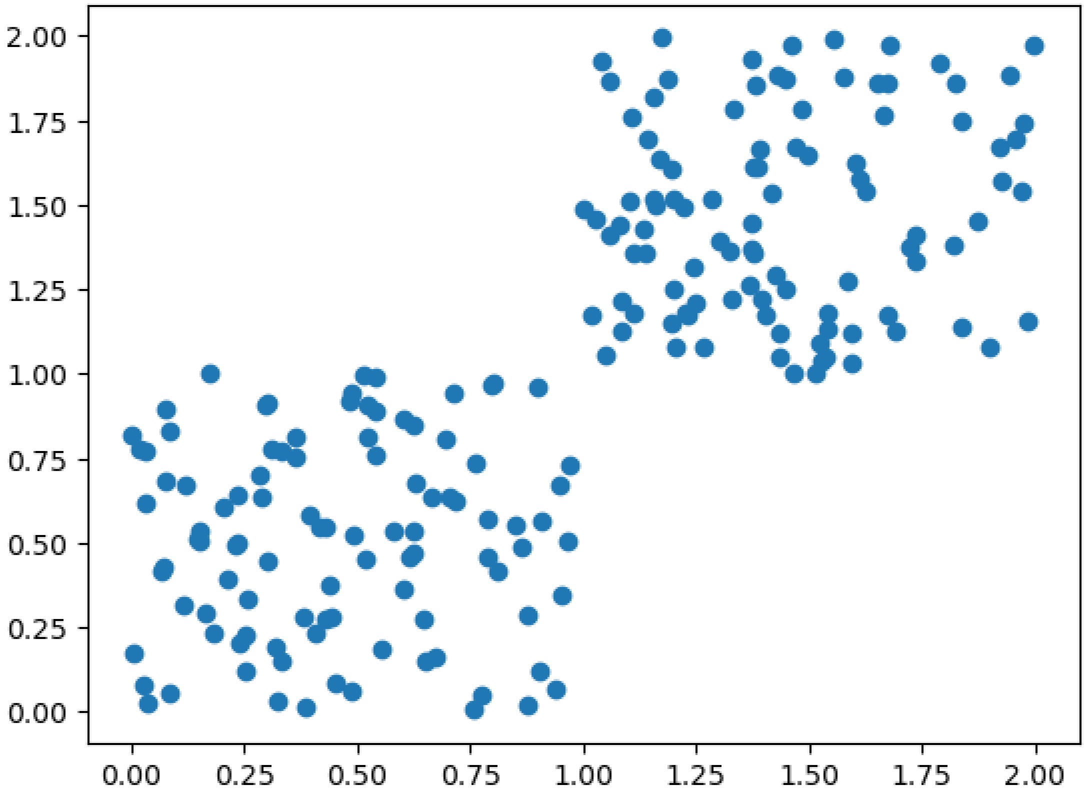
Clustering is a machine learning technique that involves grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups. It's widely used for exploratory data analysis to find natural groupings, patterns, or structures within data without prior knowledge of group definitions.



```
In [ ]: from sklearn.preprocessing import StandardScaler
        import matplotlib.pyplot as plt
```

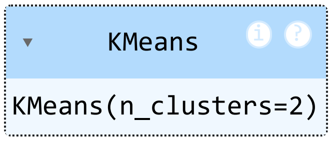
```

def generate_data():
    return np.vstack([
        np.random.random((100, 2)),
        np.random.random((100, 2)) + 1
    ])

data = generate_data()
plt.scatter(data[:, 0], data[:, 1])
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2ae07ff4a40>
```



In [ ]:
```
# cluster with K-Means
from sklearn.cluster import KMeans

model = KMeans(2)
model.fit(data)
```

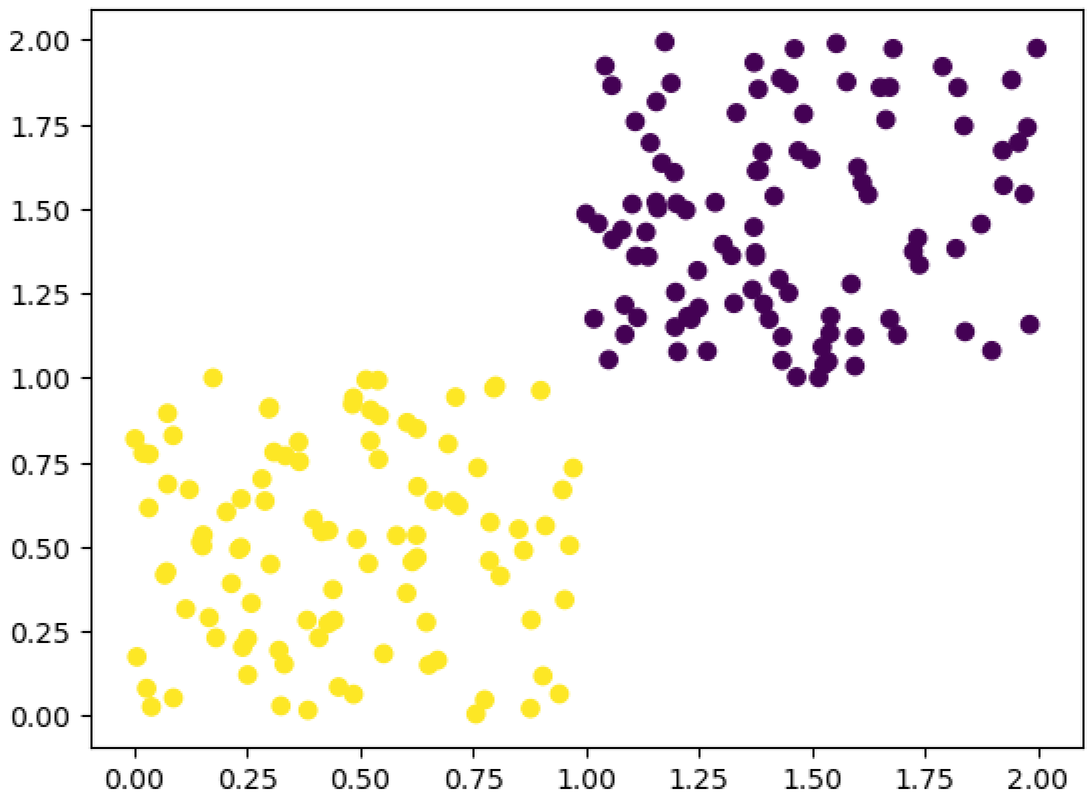
Out[ ]:



```
KMeans(n_clusters=2)
```

In [ ]:
```
plt.scatter(data[:, 0], data[:, 1], c=model.labels_)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2ae087e9130>
```



Quick Markdown & LaTeX Syntax

Header 1

Header 2

Header 3

List:

- Foo
- Bar

Bold

Italic

Inline Math: $A, B, C, D, \alpha, \beta, \gamma, \lambda, \delta$

Displaymode Math:

$$\frac{\partial f}{\partial X}$$

$$\mathrm{A}, \mathbf{X}$$

Hyperlink

```
In [ ]:
```

List:

- Foo
- Bar

bold *italic*

$$ \mathrm{A}, \mathbf{X}, \boldsymbol{\alpha}

Hyperlink