

Lecture 03:

Dimension Reduction and PCA



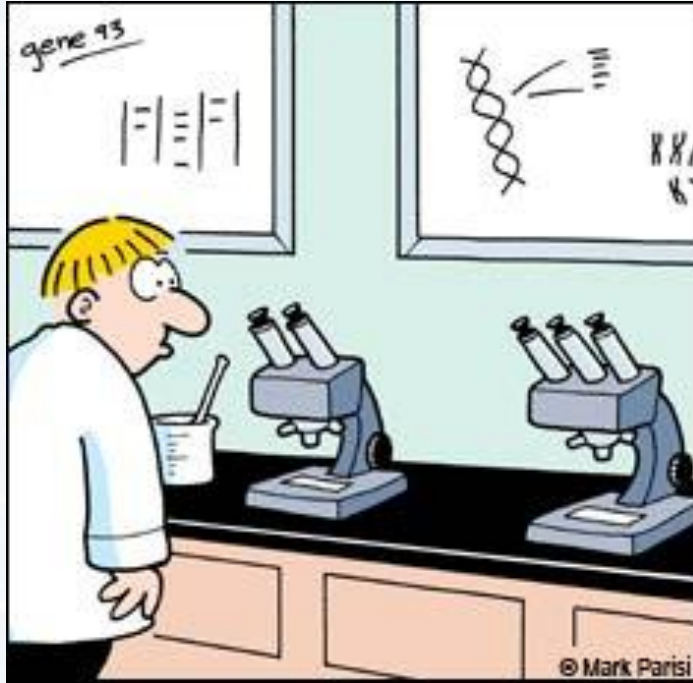
Markus Hohle

University California, Berkeley

Machine Learning Algorithms

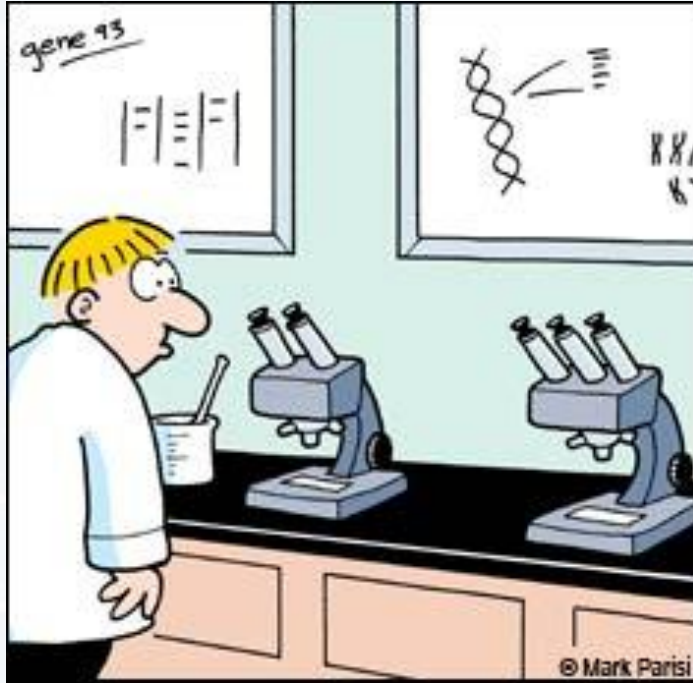
MSSE 277B, 3 Units

Spring 2025



Outline

- The Problem
- Mathematical formulation of the Problem
- Examples

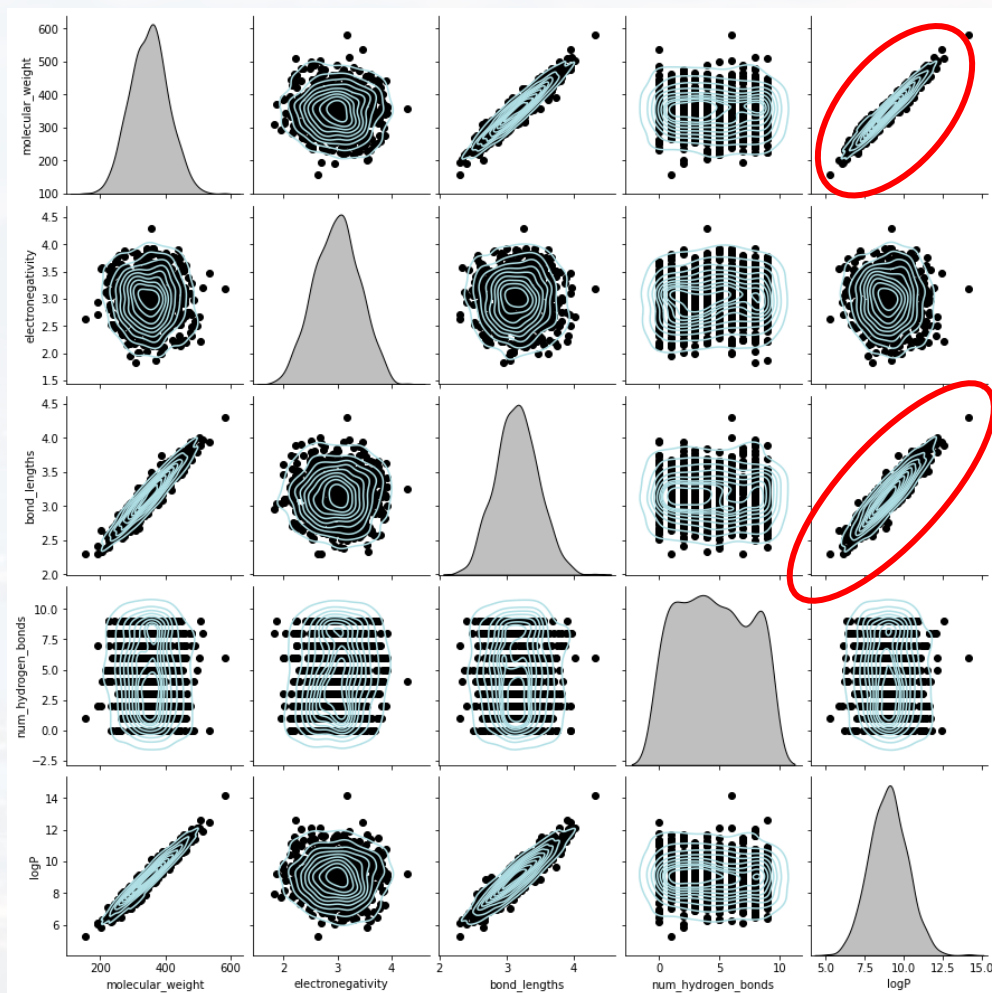


Outline

- The Problem
- Mathematical formulation of the Problem
- Examples



some features correlate!



	label	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP
	Toxic	382.602	2.00269	3.61153	3	9.82666
	Toxic	408.961	2.93626	3.47904	6	9.85889
	Non-Toxic	239.548	2.71413	2.63922	8	6.75962
	Non-Toxic	315.58	2.85598	2.86034	9	8.70674
	Non-Toxic	282.521	2.83877	2.9664	1	7.8173

$$\text{corr}(x, y) = \frac{\text{cov}(x, y)}{\sqrt{\text{var}(x)\text{var}(y)}}$$



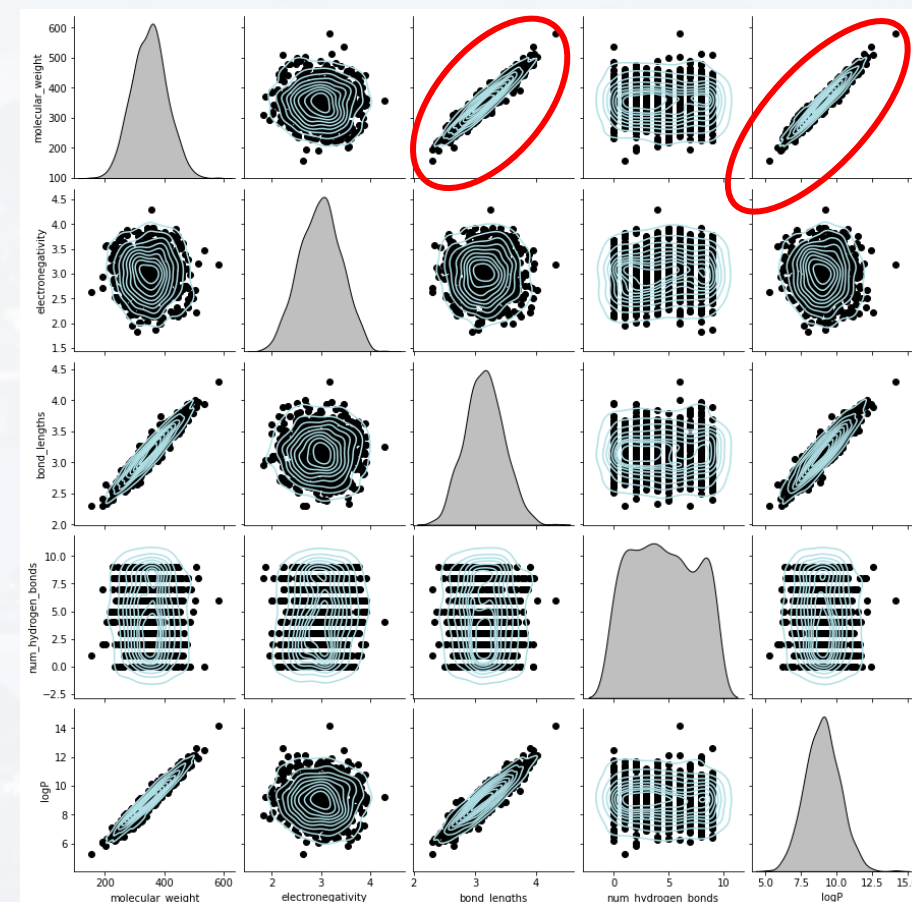


some features correlate!

correlation means:

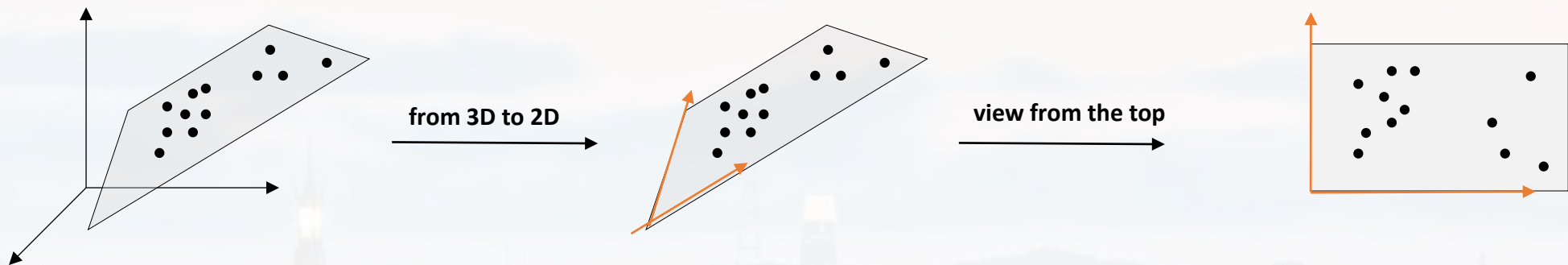
- features are **not mutually independent**
 - we can predict feature ***a*** from feature ***b*** to some extent
 - we don't need all features
- **reducing number of features** (dimensions) without losing information

label	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP
Toxic	382.602	2.00269	3.61153	3	9.82666
Toxic	408.961	2.93626	3.47904	6	9.85889
Non-Toxic	239.548	2.71413	2.63922	8	6.75962
Non-Toxic	315.58	2.85598	2.86034	9	8.70674
Non-Toxic	282.521	2.83877	2.9664	1	7.8173





some features correlate!



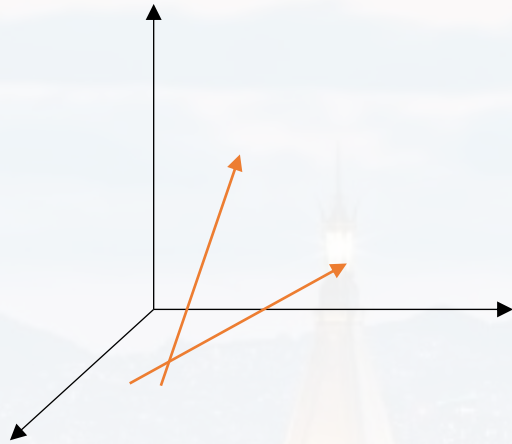
each data point is
represented by
three features...

... but those features correlate
 $(x, y) \rightarrow z$

new coordinate system



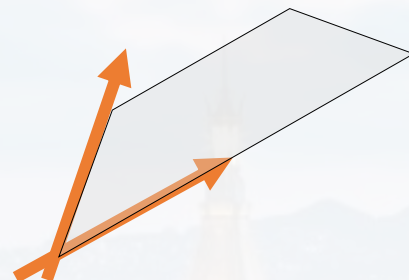
some features correlate!



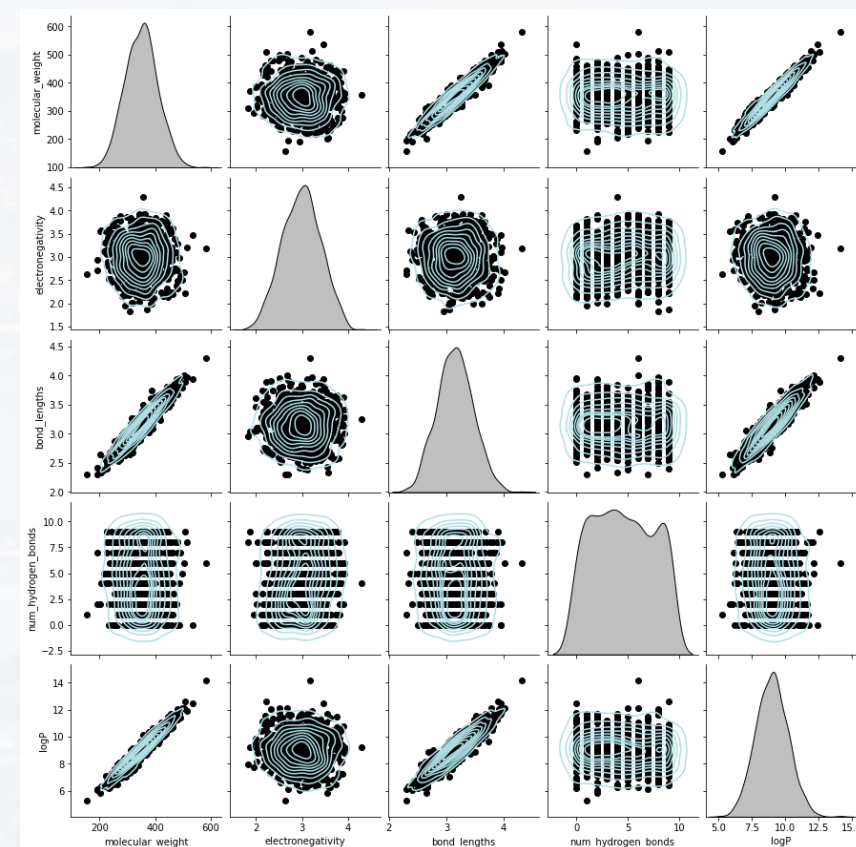
some features correlate!

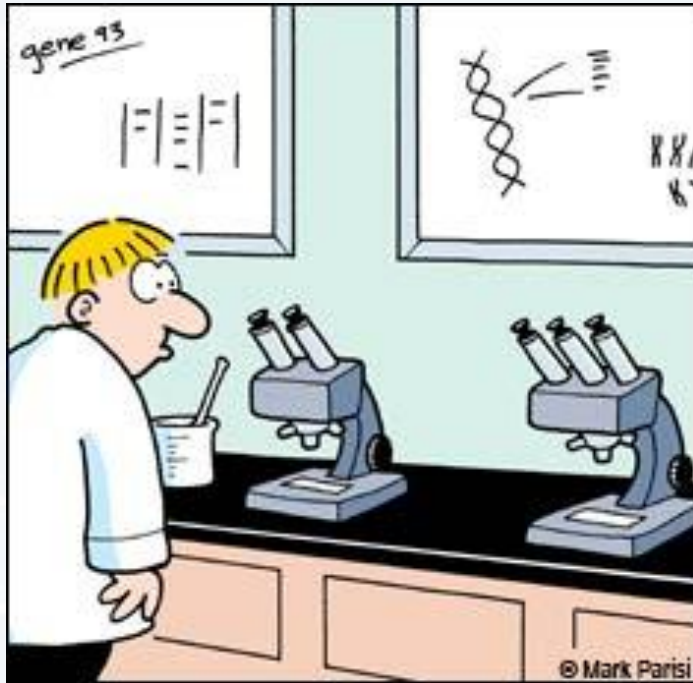
The **axis** of the **new coordinate** system are called **eigenvectors**

eigen: loosely translated from German “proper”



How do we find the eigenvectors based on correlation?





Outline

- The Problem
- Mathematical formulation of the Problem
- Examples



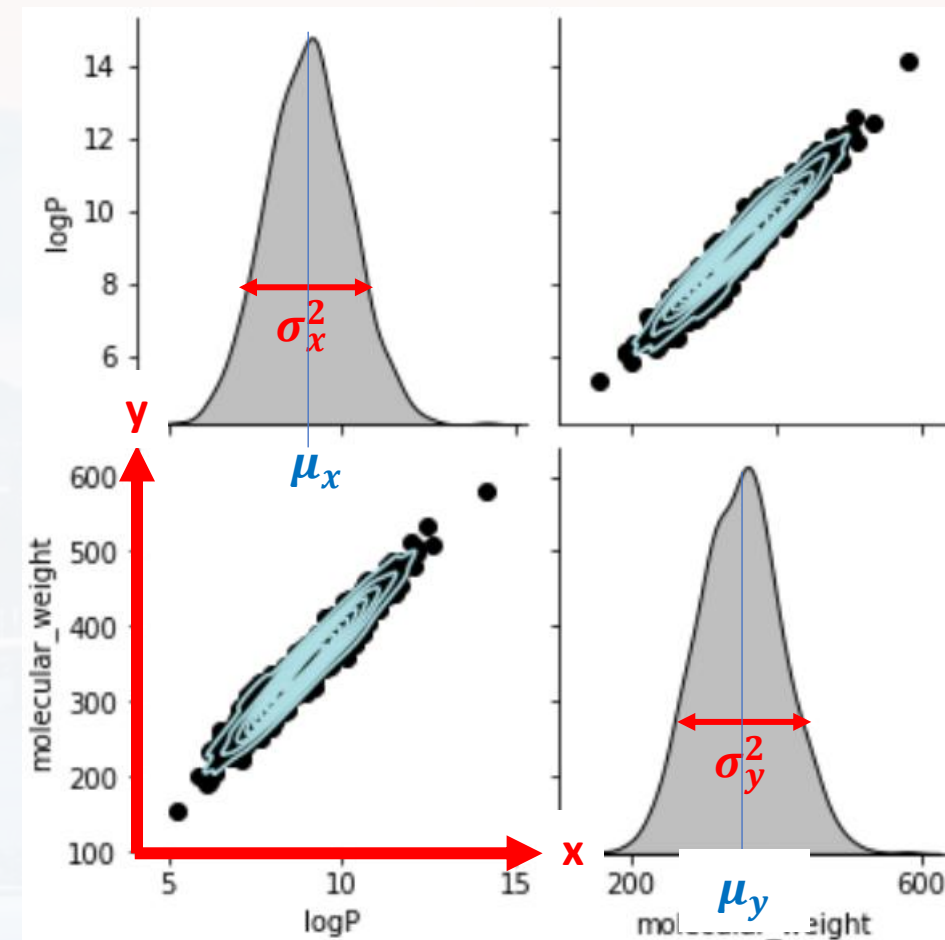
$$\text{corr}(x, y) := \frac{\text{cov}(x, y)}{\sqrt{\text{var}(x)\text{var}(y)}}$$

$$\text{var}(x) \equiv \sigma_x^2 := \sum_i^N (x_i - \mu_x)^2$$

$$\text{cov}(x, y) := \sum_j^M \sum_i^N (x_i - \mu_x)(y_j - \mu_y)$$

$$\sigma_{tot}^2 = \boxed{\sigma_x^2} + \boxed{\sigma_y^2} + \boxed{2 \text{ cov}(x, y)}$$

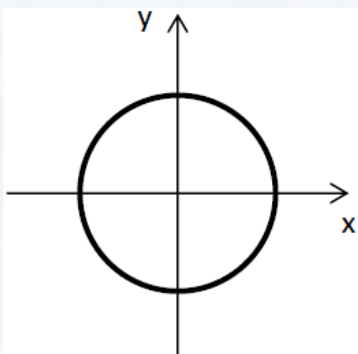
Let's try to remember this structure!





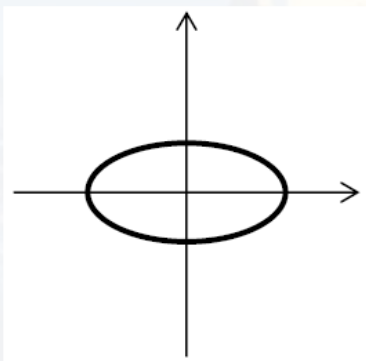
$$\sigma_{tot}^2 = \sigma_x^2 + \sigma_y^2 + 2 \operatorname{cov}(x, y)$$

about cone sections:



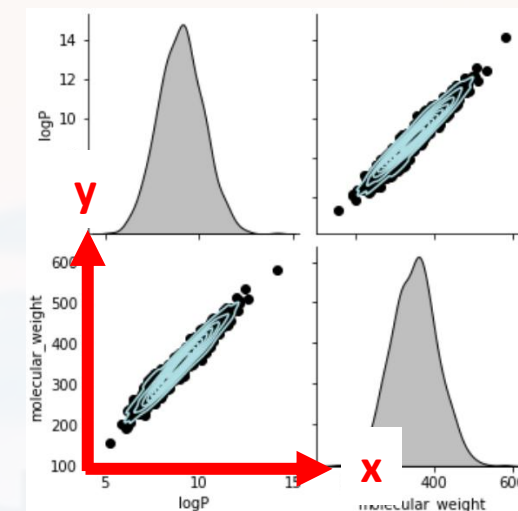
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = \text{const}$$

$$a = b \rightarrow x^2 + y^2 = r^2$$



$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = \text{const}$$

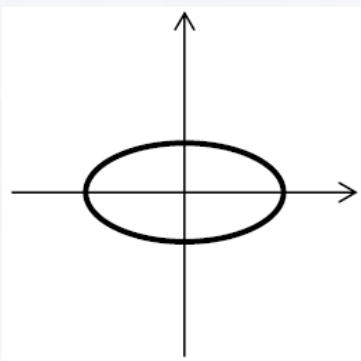
$$a \neq b$$





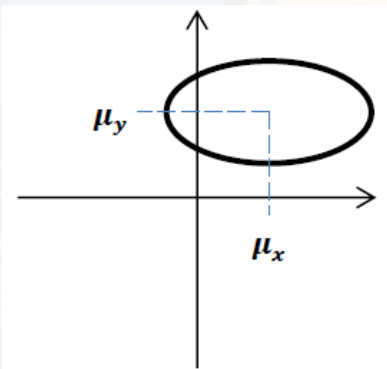
$$\sigma_{tot}^2 = \sigma_x^2 + \sigma_y^2 + 2 \operatorname{cov}(x, y)$$

about cone sections:



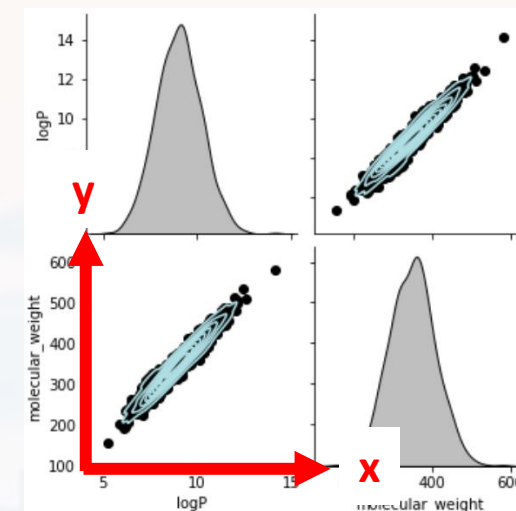
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = \text{const}$$

$$a \neq b$$



$$\frac{(x - \mu_x)^2}{a^2} + \frac{(y - \mu_y)^2}{b^2} = \text{const}$$

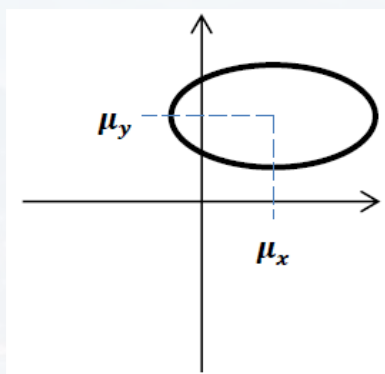
$$a \neq b$$





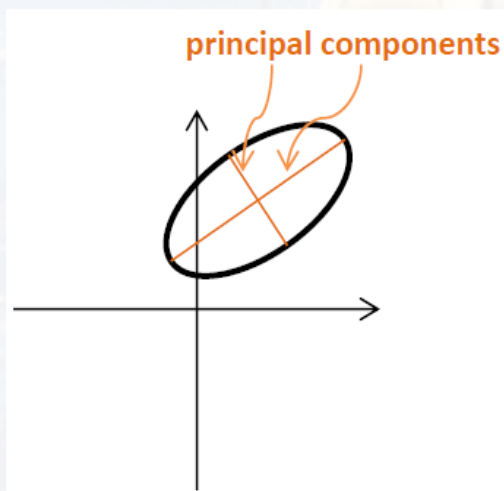
$$\sigma_{tot}^2 = \sigma_x^2 + \sigma_y^2 + 2 \operatorname{cov}(x, y)$$

about cone sections:



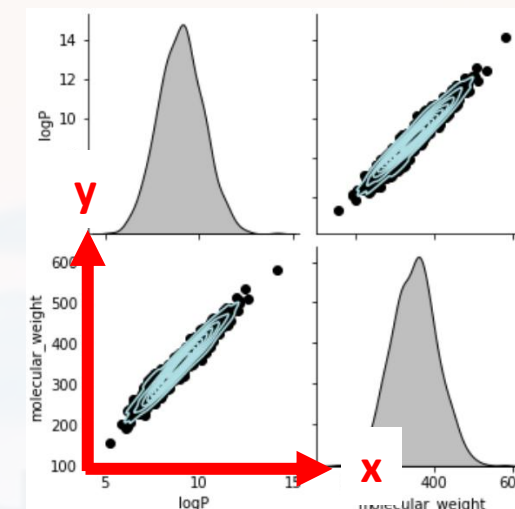
$$\frac{(x - \mu_x)^2}{a^2} + \frac{(y - \mu_y)^2}{b^2} = \text{const}$$

$$a \neq b$$



$$\frac{(x - \mu_x)^2}{a^2} + \frac{(y - \mu_y)^2}{b^2} + 2c(x - \mu_x)(y - \mu_y) = \text{const}$$

$$a \neq b$$





$$\sigma_{tot}^2 = \sigma_x^2 + \sigma_y^2 + 2 \operatorname{cov}(x, y)$$

$$= \sum_i^N (x_i - \mu_x)^2 + \sum_j^M (y_j - \mu_y)^2 + 2 \sum_j^M \sum_i^N (x_i - \mu_x)(y_j - \mu_y)$$

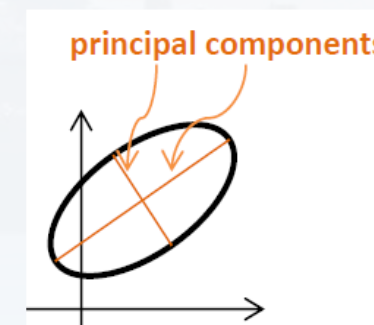
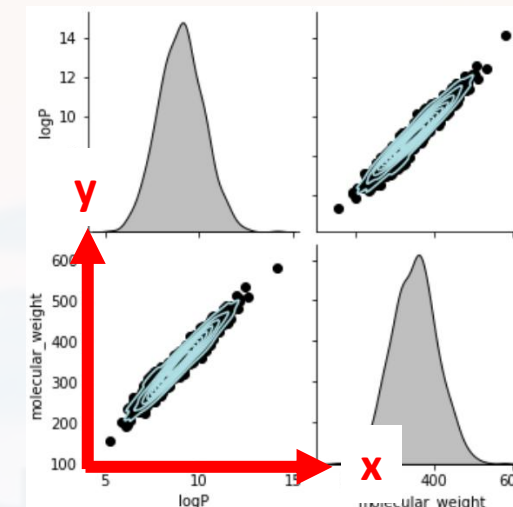
$$\operatorname{const} = \frac{(x - \mu_x)^2}{a^2} + \frac{(y - \mu_y)^2}{b^2} + 2c(x - \mu_x)(y - \mu_y)$$

$$\operatorname{const} = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} 1/a^2 & c \\ c & 1/b^2 \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}$$

more general:

$$\operatorname{const} = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} \alpha & \gamma_{12} \\ \gamma_{21} & \beta \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix} \quad \text{covariance matrix}$$

$$= v^T S v \quad \dots \text{called } \mathbf{quadric} \text{ (also in N-D)}$$





$$\sigma_{tot}^2 = \sigma_x^2 + \sigma_y^2 + 2 \operatorname{cov}(x, y)$$

$$= \sum_i^N (x_i - \mu_x)^2 + \sum_j^M (y_j - \mu_y)^2 + 2 \sum_j^M \sum_i^N (x_i - \mu_x)(y_j - \mu_y)$$

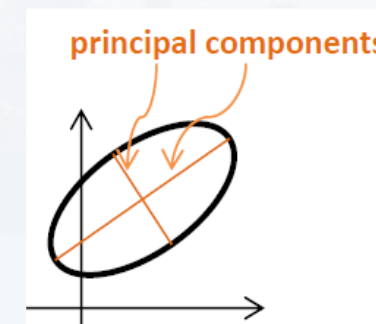
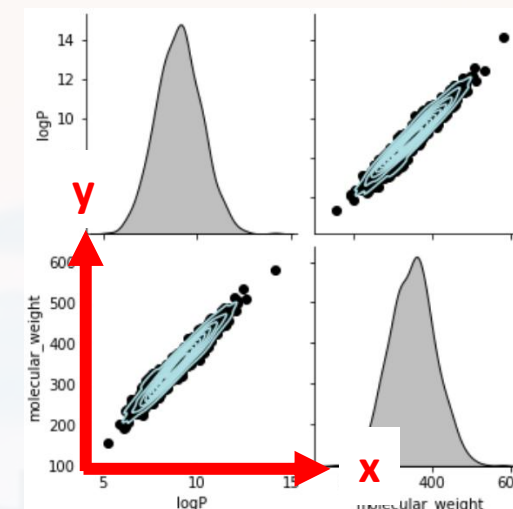
$$\text{const} = \frac{(x - \mu_x)^2}{a^2} + \frac{(y - \mu_y)^2}{b^2} + 2c(x - \mu_x)(y - \mu_y)$$

$$\text{const} = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} 1/a^2 & c \\ c & 1/b^2 \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}$$

more general:

$$\text{const} = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} \alpha & \gamma_{12} \\ \gamma_{21} & \beta \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}$$

$$= v^T S v \quad \dots \text{called } \mathbf{quadratic} \text{ (also in N-D)}$$

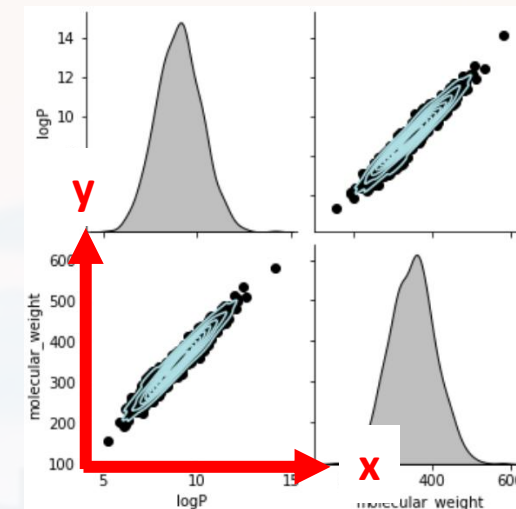




$$\sigma_{tot}^2 = \sigma_x^2 + \sigma_y^2 + 2 \text{cov}(x, y)$$

$$\text{const} = \frac{(x - \mu_x)^2}{a^2} + \frac{(y - \mu_y)^2}{b^2} + 2 c(x - \mu_x)(y - \mu_y)$$

$$\text{const} = \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}^T \begin{pmatrix} 1/a^2 & c \\ c & 1/b^2 \end{pmatrix} \begin{pmatrix} x - \mu_x \\ y - \mu_y \end{pmatrix}$$

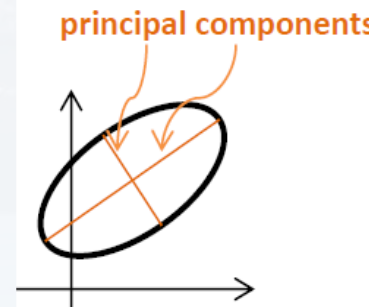


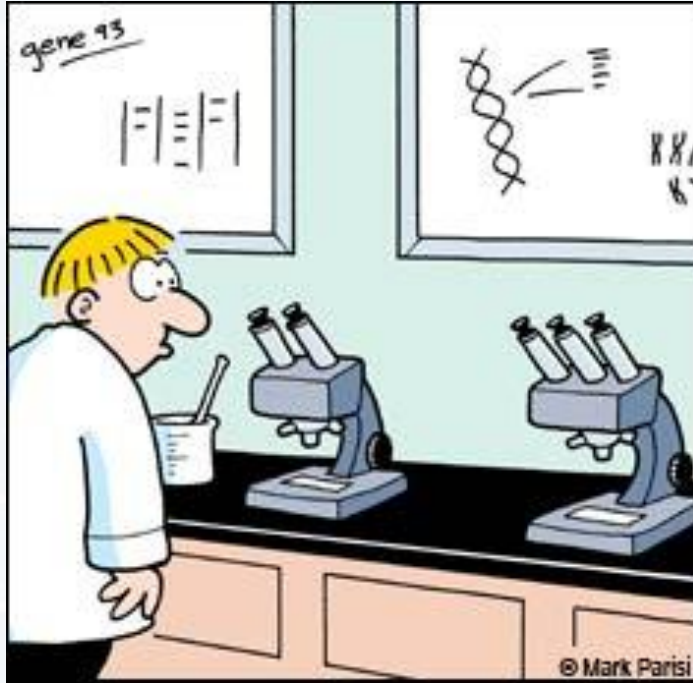
- geometrically, the **covariance matrix** can be interpreted as quadric
- the covariances are the **non-diagonal** elements of the **covariance matrix**
- aim: finding a coordinate transformation, where the **covariance matrix** is diagonal

$$\begin{pmatrix} \lambda_1 & \dots & 0 & \dots & 0 \\ 0 & & \lambda_i & \dots & 0 \\ 0 & & 0 & & \lambda_N \end{pmatrix}$$

the diagonal entries are called **eigenvalues** (= variances in new coordinate system)

- all variables are independent
- principal components of the **covariance matrix** are **parallel** to the **new coordinate axes** (= eigenvectors)





Outline

- The Problem
- Mathematical formulation of the Problem
- Examples

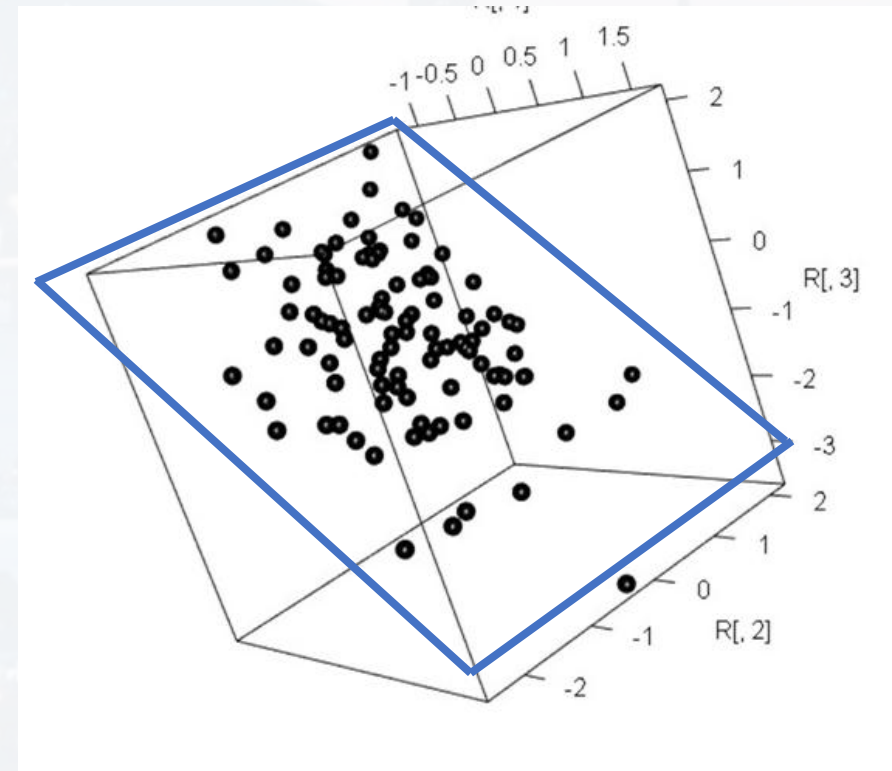


```
from sklearn.decomposition import PCA
```

Let us take a look at some artificial data first:

see **PCA_simple.ipynb**

- 3D data cloud
- however, all data points seem to be located on **one plane**
- PCA should be able to **reduce dimensions**





```
from sklearn.decomposition import PCA
```

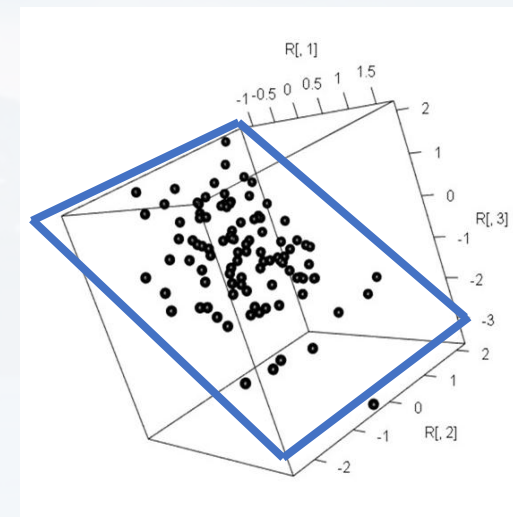
Let us take a look at some artificial data first:

```
XYZ = pd.read_csv('Rot.txt', delim_whitespace = True,\n                  header = None)
```

```
XYZ = np.array(XYZ)
```

```
fig = plt.figure(figsize = (12, 12))  
ax = fig.add_subplot(projection = '3d')  
ax.scatter(XYZ[:,0], XYZ[:,1], XYZ[:,2], c = 'black',\n           marker = 'o', s = 40)  
ax.set_xlabel('X')  
ax.set_ylabel('Y')  
ax.set_zlabel('Z')  
ax.tick_params(axis = 'both', which = 'major', labels = 30)  
plt.show()
```

- 3D data cloud
- however, all data points seem to be located on **one plane**
- PCA should be able to **reduce dimensions**





```
from sklearn.decomposition import PCA
```

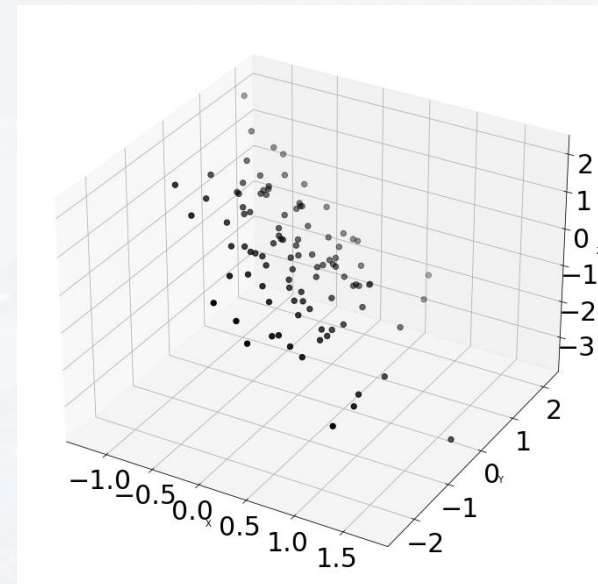
Let us take a look at some artificial data first:

```
XYZ = pd.read_csv('Rot.txt', delim_whitespace = True,\n                  header = None)
```

```
XYZ = np.array(XYZ)
```

```
fig = plt.figure(figsize = (12, 12))  
ax = fig.add_subplot(projection = '3d')  
ax.scatter(XYZ[:,0], XYZ[:,1], XYZ[:,2], c = 'black',\  
           marker = 'o', s = 40)  
ax.set_xlabel('X')  
ax.set_ylabel('Y')  
ax.set_zlabel('Z')  
ax.tick_params(axis = 'both', which = 'major', labelsize = 30)  
plt.show()
```

- 3D data cloud
- however, all data points seem to be located on **one plane**
- PCA should be able to **reduce dimensions**





performing the actual PCA:

```
out = PCA(n_components = 3).fit(XYZ)
```

```
eigenVec = out.components_  
eigenVal = out.explained_variance_  
eigenXYZ = out.transform(XYZ)
```

plotting the eigenvalue spectrum:

```
xplot = np.arange(1,4)
```

```
plt.bar(xplot, eigenVal, color = (0.8, 0.8, 0.8), edgecolor = 'black')  
plt.xlabel('dimension')  
plt.ylabel('eigenvalue')  
plt.yscale('log')  
plt.xticks(xplot)  
plt.show()
```

- 3D data cloud
- however, all data points seem to be located on **one plane**
- PCA should be able to **reduce dimensions**



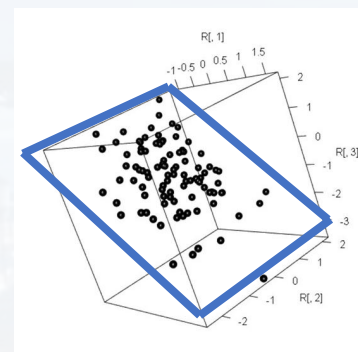
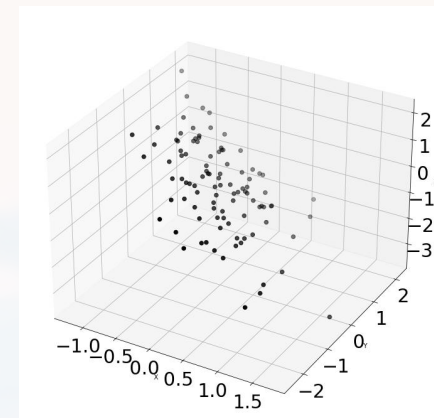
```
out = PCA(n_components = 3).fit(XYZ)
```

```
eigenVec = out.components_  
eigenVal = out.explained_variance_  
eigenXYZ = out.transform(XYZ)
```

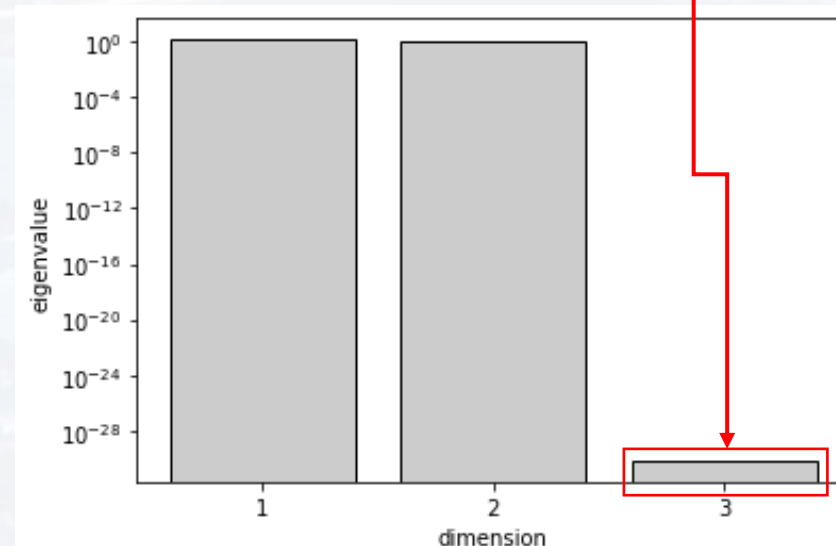
plotting the eigenvalue spectrum:

```
xplot = np.arange(1,4)
```

```
plt.bar(xplot, eigenVal, color = (0.8, 0.8, 0.8), edgecolor = 'black')  
plt.xlabel('dimension')  
plt.ylabel('eigenvalue')  
plt.yscale('log')  
plt.xticks(xplot)  
plt.show()
```



one eigenvalue
is zero

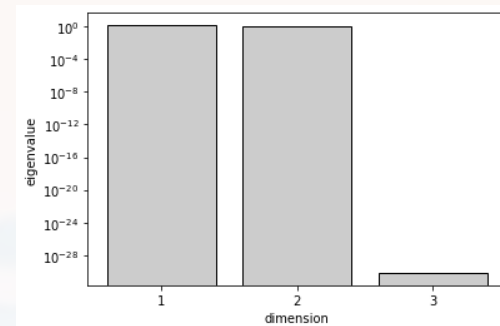




plotting the eigenvalue spectrum:

```
xplot = np.arange(1,4)
```

```
plt.bar(xplot, eigenVal, color = (0.8, 0.8, 0.8), edgecolor = 'black')  
plt.xlabel('dimension')  
plt.ylabel('eigenvalue')  
plt.yscale('log')  
plt.xticks(xplot)  
plt.show()
```



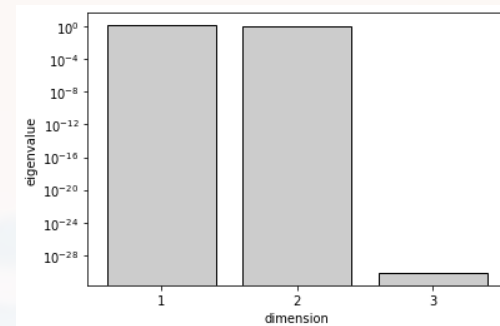
```
fig = plt.figure(figsize = (12, 12))  
ax = fig.add_subplot(projection = '3d')  
ax.scatter(eigenXYZ[:,0], eigenXYZ[:,1], eigenXYZ[:,2], c = 'black', \  
           marker = 'o', s = 40)  
ax.set_xlabel('X')  
ax.set_ylabel('Y')  
ax.set_zlabel('Z')  
ax.tick_params(axis = 'both', which = 'major', labels = 30)  
plt.show()
```




plotting the eigenvalue spectrum:

```
xplot = np.arange(1,4)
```

```
plt.bar(xplot, eigenVal, color = (0.8, 0.8, 0.8), edgecolor = 'black')  
plt.xlabel('dimension')  
plt.ylabel('eigenvalue')  
plt.yscale('log')  
plt.xticks(xplot)  
plt.show()
```

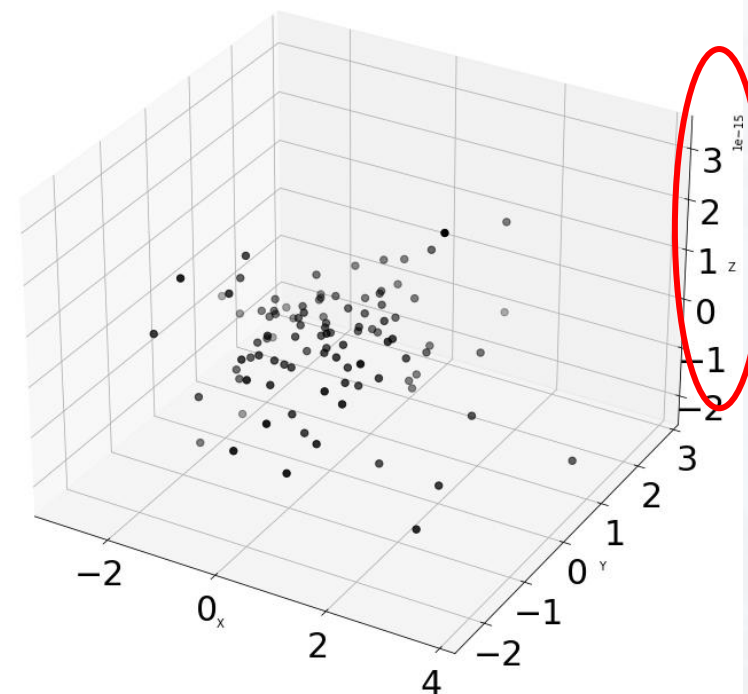


```
fig = plt.figure(figsize = (12, 12))  
ax = fig.add_subplot(projection = '3d')  
ax.scatter(eigenXYZ[:,0], eigenXYZ[:,1], eigenXYZ[:,2], c = 'bl')  
ax.set_xlabel('X')  
ax.set_ylabel('Y')  
ax.set_zlabel('Z')  
ax.tick_params(axis = 'both', which = 'major', labelsize = 30)  
plt.show()
```

check also eg:

```
np.dot(eigenVec[:,0], eigenVec[:,1])
```

almost no variance along new z-coord





let us return to the molecule data set now:

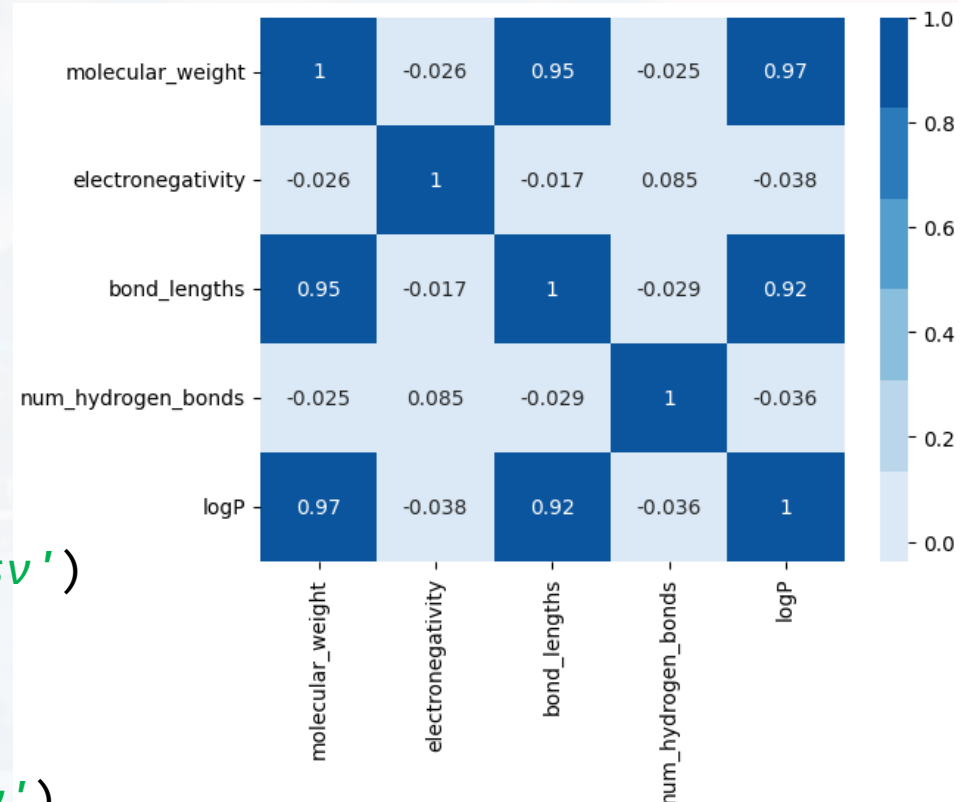
see **NB.ipynb**

	label	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP
	Toxic	382.602	2.00269	3.61153	3	9.82666
	Toxic	408.961	2.93626	3.47904	6	9.85889
	Non-Toxic	239.548	2.71413	2.63922	8	6.75962
	Non-Toxic	315.58	2.85598	2.86034	9	8.70674
	Non-Toxic	282.521	2.83877	2.9664	1	7.8173

```
Train = pd.read_csv('molecular_train_gbc.csv')
TrainY = Train['label']
TrainX = Train.drop('label', axis = 1)

Test = pd.read_csv('molecular_test_gbc.csv')
TestY = Test['label']
TestX = Test.drop('label', axis = 1)
```

$$\text{corr}(x, y) = \frac{\text{cov}(x, y)}{\sqrt{\text{var}(x)\text{var}(y)}}$$





let us return to the molecule data set now:

```
from sklearn.preprocessing import MinMaxScaler
```

```
All      = pd.concat((TrainX, TestX), axis = 0)
scaler    = MinMaxScaler(feature_range = (0, 1)) #mean = 0, std = 1
AllN      = scaler.fit_transform(All)
```

important:
scaling and
normalization

```
(rows, _) = TrainX.shape
TrainXN    = pd.DataFrame(AllN[:rows,:])
```

```
out = PCA(n_components = 5).fit(AllN)
```

```
eigenVec = out.components_  
eigenVal = out.explained_variance_  
eigenXYZ = out.transform(TrainXN)
```



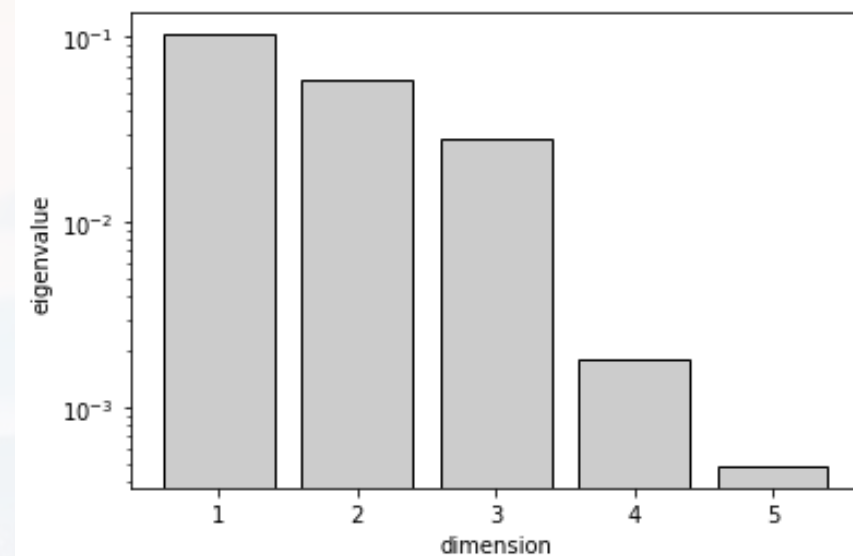

let us return to the molecule data set now:

```
out = PCA(n_components = 5).fit(AllN)
```

```
eigenVec = out.components_  
eigenVal = out.explained_variance_  
eigenXYZ = out.transform(TrainXN)
```

```
xplot = np.arange(1,6)
```

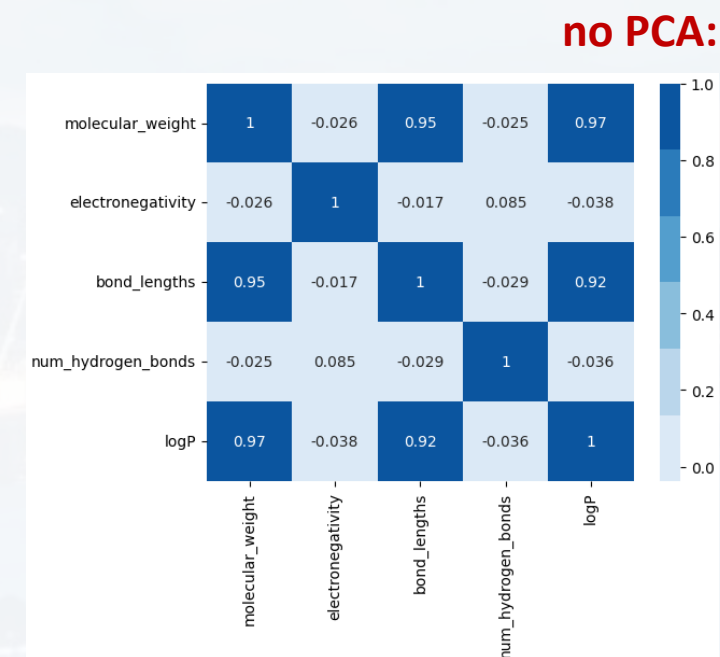
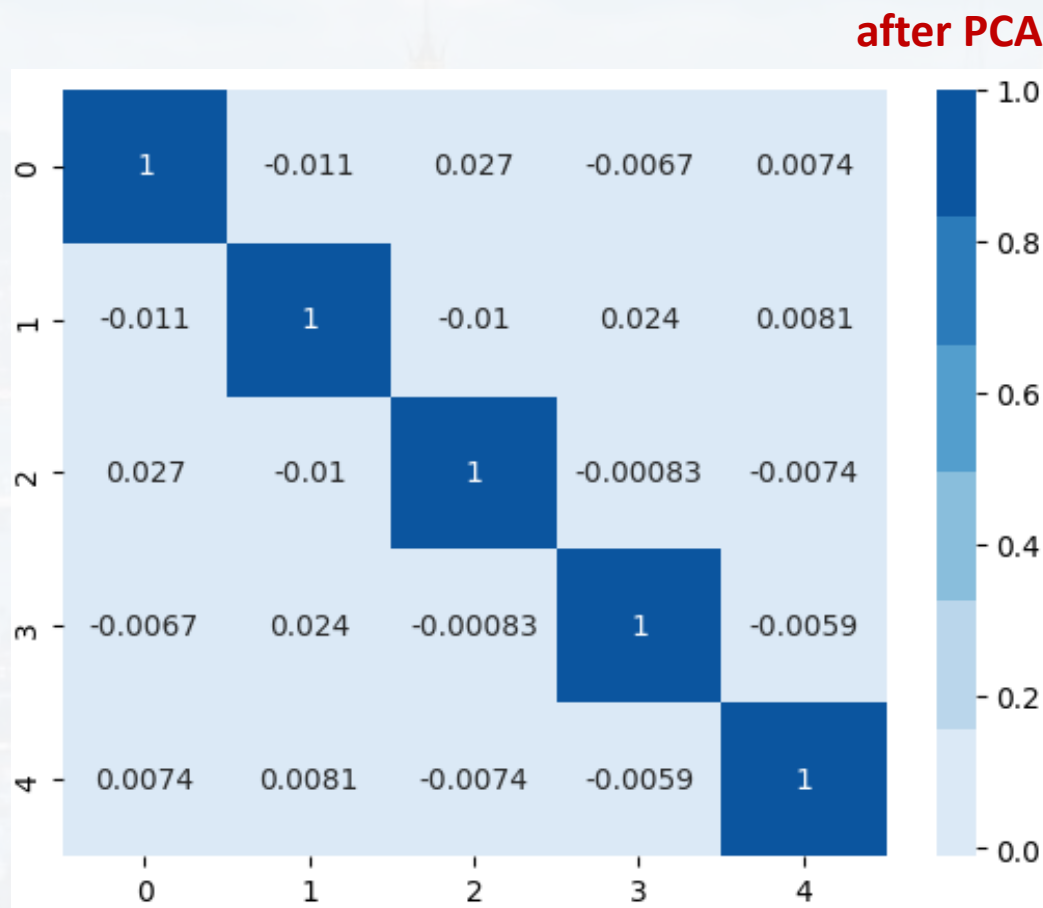
```
plt.bar(xplot, eigenVal, color = (0.8, 0.8, 0.8), edgecolor = 'black')  
plt.xlabel('dimension')  
plt.ylabel('eigenvalue')  
plt.yscale('log')  
plt.xticks(xplot)  
plt.show()
```





let us return to the molecule data set now:

```
sns.heatmap(pd.DataFrame(eigenTrainX).corr(), annot = True,  
             cmap = color_palette("Blues"))  
plt.show()
```





let us return to the molecule data set now:

```
eigenXYZTrainX_UMAP = umap.UMAP(n_components = 3).fit_transform(eigenTrainX)

scatter = go.Scatter3d(x = eigenXYZTrainX_UMAP[:,0],
                      y = eigenXYZTrainX_UMAP[:,1],
                      z = eigenXYZTrainX_UMAP[:,2],
                      mode = 'markers', marker = dict(size = 3, color = cols))

fig = go.Figure(data = [scatter])
fig.update_layout(width = 800, height = 800,
                  margin = dict(r = 10, b = 10, l = 100, t = 10))
fig.show()
```




let us return

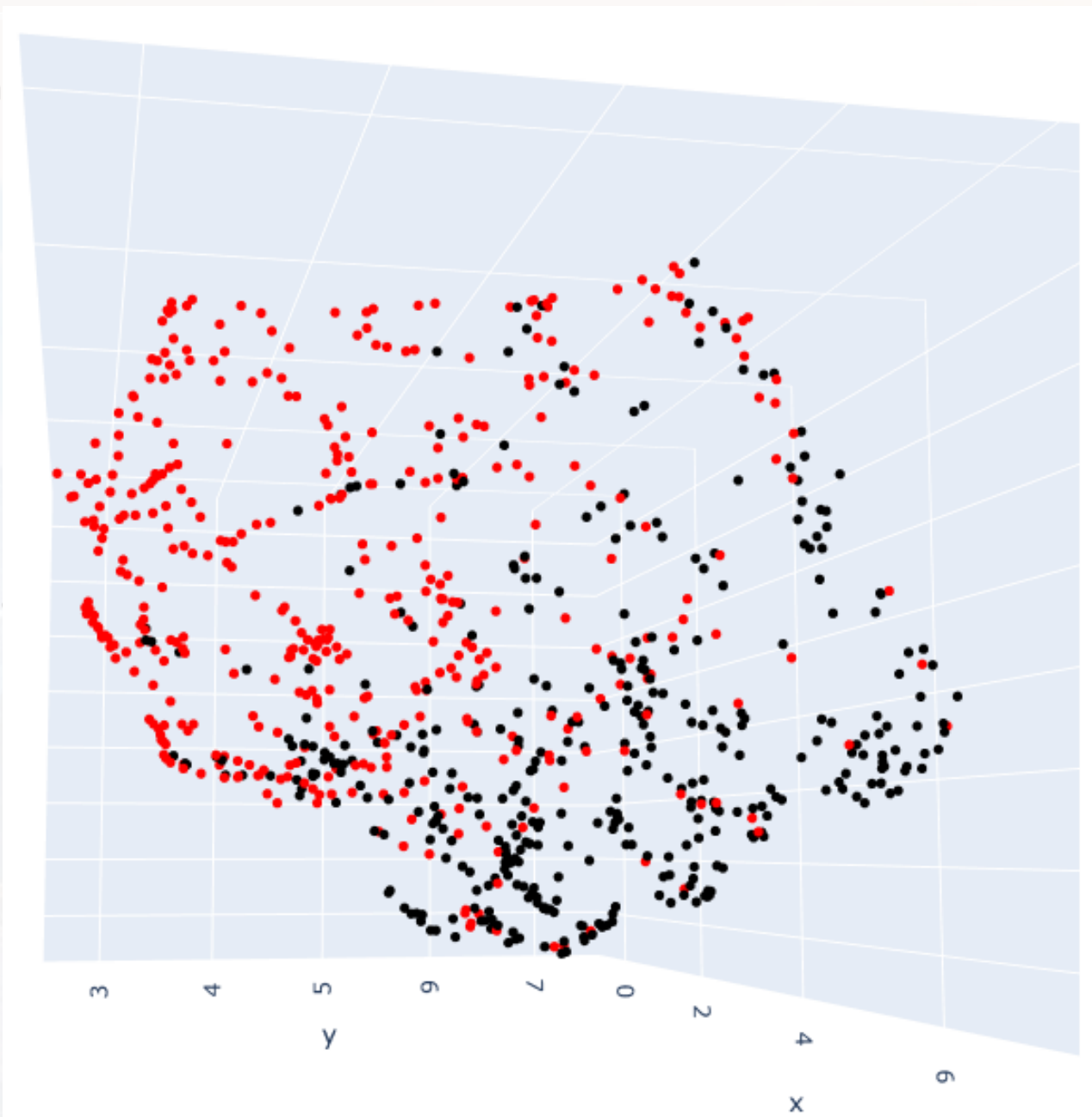
eigenXYZT

scatter =

fig =

fig.update

fig.show()



transform(eigenTrainX)

size = 3, color = cols))

= 10))



Classification Naïve Bayes

$$k_{new} = \underset{k}{\operatorname{argmax}} \left\{ P(C_k) \prod_{i=1}^I P(x_i | C_k) \right\}$$

no PCA:

GaussianNB: Number of mislabelled points out of a total 200 points: 38

after PCA:

GaussianNB: Number of mislabelled points out of a total 200 points: 33

Thank you very much for your attention!

