

K-means
optimization
↳ Lloyd's

- Low-voice
- breaks as
needed

Topics:

- code for DBSCAN → outliers
↳ sklearn
scipy
- case-study for Hierarchical clustering
↳ financial
- Gaussian - mixtures
↳ probabilistic
- Gaussian - mixture - model
- Expectation - Maximization (EM)
↳ alt to Gradient Descent

- GMMs
- Misc Topics - - -
- Anomaly detection - - -
- t-SNE & UMAP ←
- Rec Sys
- Time Series

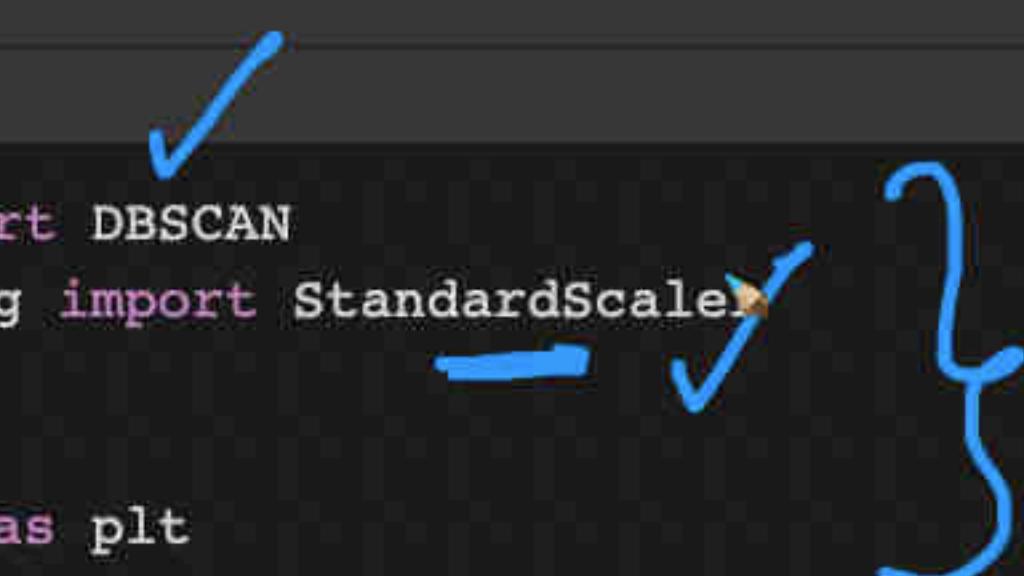
- 12 ↗ { - SQL, ✓ ↓
- ==== { - Ingest to Tableau
===== { - DL - MLP ... (nloc)
- { - DL - CV (1m)
- { - DL - NLP (1m)
- :

DBSCAN_Hierarchical Clustering x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x +

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=CDPgNHG9XB9d

+ Code + Text Reconnect

```
[ ] from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```



▼ Clustering with outliers

▶ Dataset

```
[ ] ↳ 5 cells hidden
```

▶ Simple Visualization

```
[ ] ↳ 2 cells hidden
```

↔ ▶ DBSCAN

```
[ ] ↳ 6 cells hidden
```

DBSCAN_Hierarchical Clustering x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x + colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=EnTOPragUXM8 Reconnect

+ Code + Text

```
[ ] from sklearn.cluster import DBSCAN
[ ] from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Anomaly detection (density)

DBSCAN → eliminates noise pts + (border pt core pt)

Clustering with outliers

Dataset

```
[ ] id = "1dr93lHQUCHiiillwsGoS40VcUj-rW4H1"
print("https://drive.google.com/uc?export=download&id=" + id)
```

<https://drive.google.com/uc?export=download&id=1dr93lHQUCHiiillwsGoS40VcUj-rW4H1>

```
[ ] !wget "https://drive.google.com/uc?export=download&id=1dr93lHQUCHiiillwsGoS40VcUj-rW4H1" -O wholesaledata.csv
```

--2022-06-08 12:37:53-- <https://drive.google.com/uc?export=download&id=1dr93lHQUCHiiillwsGoS40VcUj-rW4H1>

Resolving drive.google.com (drive.google.com)... 172.217.204.101, 172.217.204.138, 172.217.204.102, ...

Connecting to drive.google.com (drive.google.com)|172.217.204.101|:443... connected.

HTTP request sent, awaiting response... 303 See Other

Location: <https://doc-04-ag-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1/kcah3v1ce9nue0jluupdahodsrlquj77/165>

Warning: wildcards not supported in HTTP.

--2022-06-08 12:37:54-- <https://doc-04-ag-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1/kcah3v1ce9nue0jluupdahodsrlquj77/165>

Resolving doc-04-ag-docs.googleusercontent.com (doc-04-ag-docs.googleusercontent.com)... 142.250.97.132, 2607:f8b0:400c:c18::84

Connecting to doc-04-ag-docs.googleusercontent.com|142.250.97.132|:443... connected

DBSCAN_Hierarchical Clusterin Multivariate normal distribution MultivariateNormal.png (842x842) lecture21.pptx

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=EnTOPragUXM8

Reconnect Update

+ Code + Text

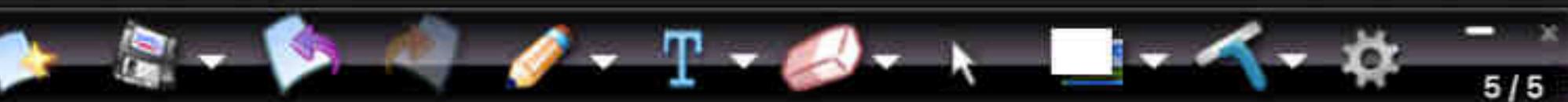
```
[ ] Data columns (total 8 columns):  
 # Column Non-Null Count Dtype  
 --- ---  
 0 Channel 440 non-null int64  
 1 Region 440 non-null int64  
 2 Fresh 440 non-null int64  
 3 Milk 440 non-null int64  
 4 Grocery 440 non-null int64  
 5 Frozen 440 non-null int64  
 6 Detergents_Paper 440 non-null int64  
 7 Delicassen 440 non-null int64  
 dtypes: int64(8)  
 memory usage: 27.6 KB
```

```
[ ] df.head()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

```
[ ] # Dropping categorical variables for simplicity
```

```
df.drop(["Channel", "Region"], axis = 1, inplace = True)
```



DBSCAN_Hierarchical Clusterin Multivariate normal distribution MultivariateNormal.png (842x640) lecture21.pptx

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzIBR6#scrollTo=yjgnlhgMUIRE

+ Code + Text Reconnect

[] 2022-06-08 12:37:54 (96.1 MB/s) - 'wholesale.csv' saved [15021/15021]

{x}

df = pd.read_csv('./wholesale.csv')
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 440 entries, 0 to 439
Data columns (total 8 columns):
 # Column Non-Null Count Dtype
--- ---
 0 Channel 440 non-null int64
 1 Region 440 non-null int64
 2 Fresh 440 non-null int64
 3 Milk 440 non-null int64
 4 Grocery 440 non-null int64
 5 Frozen 440 non-null int64
 6 Detergents_Paper 440 non-null int64
 7 Delicassen 440 non-null int64
 dtypes: int64(8)
 memory usage: 27.6 KB

df.head()

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776

Visualize
2-Variables

6/6

DBSCAN_Hierarchical Clusterin Multivariate normal distribution MultivariateNormal.png (842x640) lecture21.pptx

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=JsF3APAVH7-

+ Code + Text Reconnect

[] df.drop(["Channel", "Region"], axis = 1, inplace = True)

Simple Visualization

{x}

Let's plot two features data now
x = df['Grocery']
y = df['Milk']

plt.scatter(x,y)
plt.xlabel("Groceries")
plt.ylabel("Milk")
plt.show()

Milk

70000
60000
50000
40000
30000
20000
10000
0

70000
60000
50000
40000
30000
20000
10000
0

0 20000 40000 60000 80000

Groceries

7/7

DBSCAN_Hierarchical Clusterin Multivariate normal distribution MultivariateNormal.png (842x842) lecture21.pptx

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=JsF3APAVH7-

+ Code + Text

[] df.drop(["Channel", "Region"], axis = 1, inplace = True)

Reconnect

Simple Visualization

{x}

Let's plot two features data now
x = df['Grocery']
y = df['Milk']

plt.scatter(x,y)
plt.xlabel("Groceries")
plt.ylabel("Milk")
plt.show()

↑ ↓ ⌂ ⚙ 🗑️ :

Milk

70000
60000
50000
40000
30000
20000
10000
0

0 20000 40000 60000 80000

Groceries

The scatter plot displays the relationship between 'Groceries' (X-axis) and 'Milk' (Y-axis). The X-axis ranges from 0 to 80,000 with major ticks every 20,000. The Y-axis ranges from 0 to 70,000 with major ticks every 10,000. A large, dense cluster of blue dots is centered around (10,000, 10,000). Several individual data points are circled in red, including one at approximately (30,000, 70,000), one at (55,000, 55,000), one at (65,000, 5,000), and one at (75,000, 45,000).

DBSCAN_Hierarchical Clusterin Multivariate normal distribution MultivariateNormal.png (842x640) lecture21.pptx

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=JsF3APAVH7-

+ Code + Text

[] df.drop(["Channel", "Region"], axis = 1, inplace = True)

Reconnect

Simple Visualization

{x}

Let's plot two features data now
x = df['Grocery']
y = df['Milk']

plt.scatter(x,y)
plt.xlabel("Groceries")
plt.ylabel("Milk")
plt.show()

↑ ↓ ⌂ ⚙️ 🗑️ :

Milk

70000
60000
50000
40000
30000
20000
10000
0

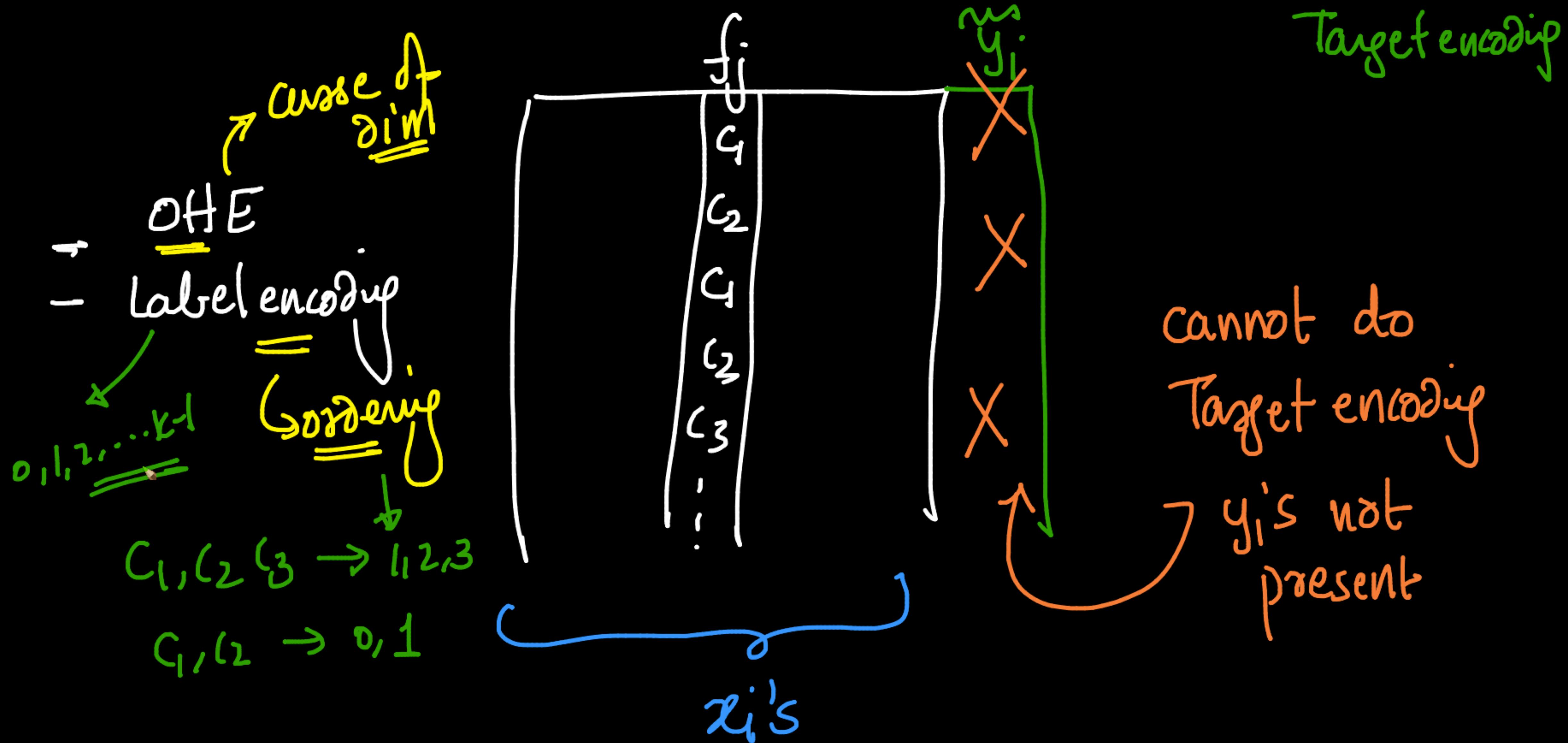
70000
60000
50000
40000
30000
20000
10000
0

0 20000 40000 60000 80000

0 20000 40000 60000 80000

Groceries

9/9



+ Code + Text

[] ↗ 5 cells hidden

Simple Visualization

[] ↗ 2 cells hidden

DBSCAN

Range Query

εps



No ...

120,000

f₂: salary

```
[ ] df = df[['Grocery', 'Milk']]  
std_scaler = StandardScaler().fit(df)  
std_df = std_scaler.transform(df)
```

(?) Yes (why?)

f₁: age
29 ...

```
[ ] # https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html  
dbsc = DBSCAN(eps = .5, min_samples = 15).fit(std_df)
```

```
[ ] # "Noisy samples are given the label -1." --> Reference  
labels = dbsc.labels_  
labels
```

```
<>      array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  
        0,  0,  0,  0,  0, -1,  0,  0,  0, -1,  0,  0,  0,  0,  0,  
        0,  0,  0,  0,  0,  0,  0,  0, -1,  0, -1,  0, -1,  0,  0,  
        0,  0,  0,  0, -1,  0,  0,  0, -1,  0,  0,  0, -1,  0,  0,  
        0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  
        -1, -1,  0,  0,  0,  0,
```

+ Code + Text

Reconnect ▾

1

► Simple Visualization

[] ↳ 2 cells hidden



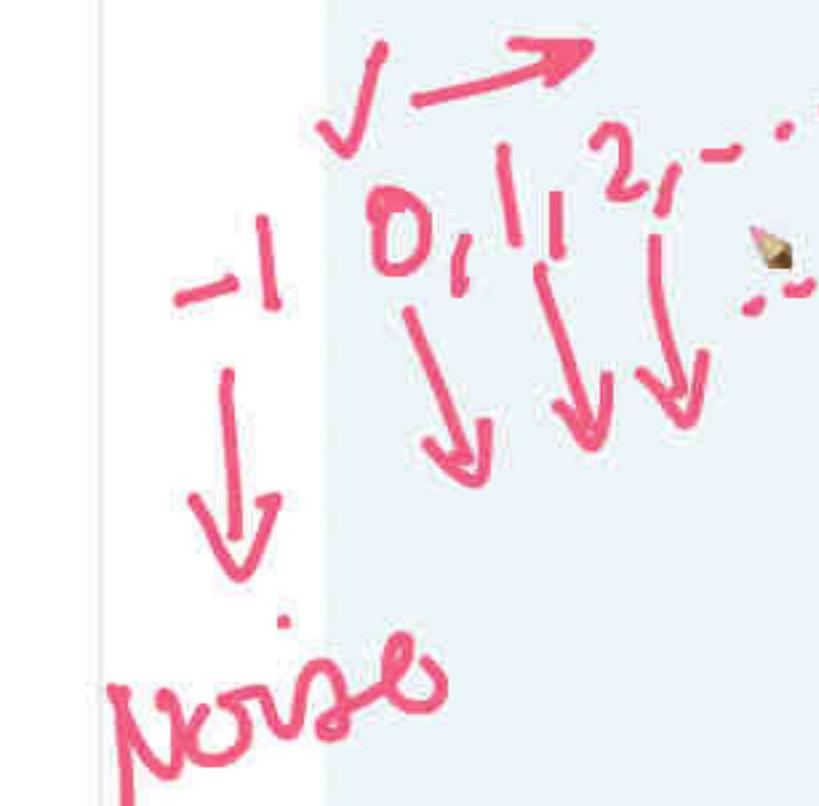
DBSCAN

```
[ ] df = df[['Grocery', 'Milk']]  
std_scaler = StandardScaler().fit(df)  
std_at = std_scaler.transform(df)
```

```
[ ] # https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html  
dbsc = DBSCAN(eps = .5, min_samples = 15).fit(std_df)
```

```
[ ] #'Noisy samples are given the label -1.' --> Reference  
labels = dbsc.labels_  
labels
```

```
array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1,  0, -1,  0, -1,  0,
       0,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  0,  0, -1,  0,
       0,  0,  0, -1,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,
      -1, -1,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0])
```

[Prev](#) [Up](#) [Next](#)**scikit-learn 1.1.1**[Other versions](#)Please [cite us](#) if you use the software.[sklearn.cluster.DBSCAN](#)Examples using [sklearn.cluster.DBSCAN](#)

The number of parallel jobs to run. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See [Glossary](#) for more details.

Attributes:**core_sample_indices_ : ndarray of shape (n_core_samples,)**

Indices of core samples.

components_ : ndarray of shape (n_core_samples, n_features)

Copy of each core sample found by training.

labels_ : ndarray of shape (n_samples)Cluster labels for each point in the dataset given to `fit()`. Noisy samples are given the label `-1`.**n_features_in_ : int**Number of features seen during `fit`.*New in version 0.24.***feature_names_in_ : ndarray of shape (n_features_in_,)**Names of features seen during `fit`. Defined only when `X` has feature names that are all strings.*New in version 1.0.*

DBSCAN_Hierarchical Clusterin Multivariate normal distribution MultivariateNormal.png (842x640) lecture21.pptx sklearn.cluster.DBSCAN — sc

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzIBR6#scrollTo=jRmeqQ3IWNVo

Reconnect

+ Code + Text

```
[ ] X = np.array([
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0,
    0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0,
    0, 0, -1, 0, 0, 0, 0, -1, 0, -1, 0, 0, 0, 0, 0, 0,
    0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0])
```

```
[ ] out_df = df.copy()
out_df['label'] = dbsc.labels_
```

```
[ ] out_df['label'].value_counts()
```

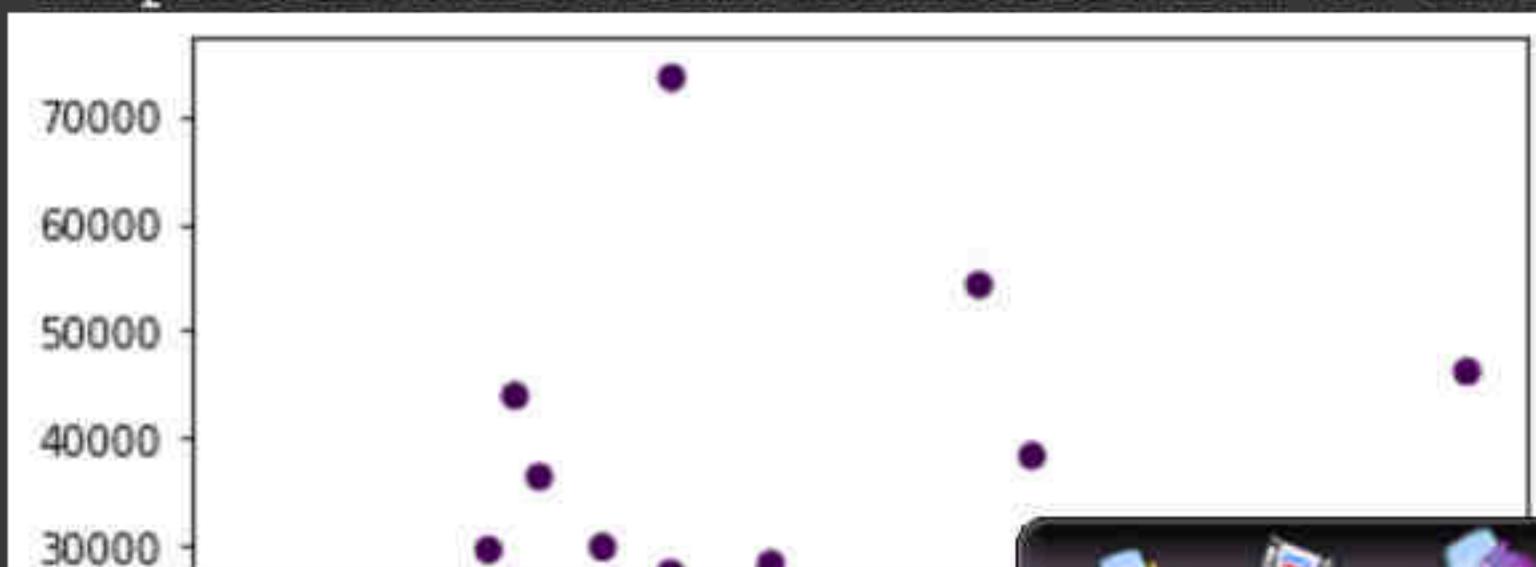
0 404
-1 36

Name: label, dtype: int64

05 15
Eps, Minpts

```
[ ] plt.scatter(out_df['Grocery'], out_df['Milk'], c=out_df['label'])
```

```
<matplotlib.collections.PathCollection at 0x7f36beee4f50>
```



```
+ Code + Text Reconnect ▾
```

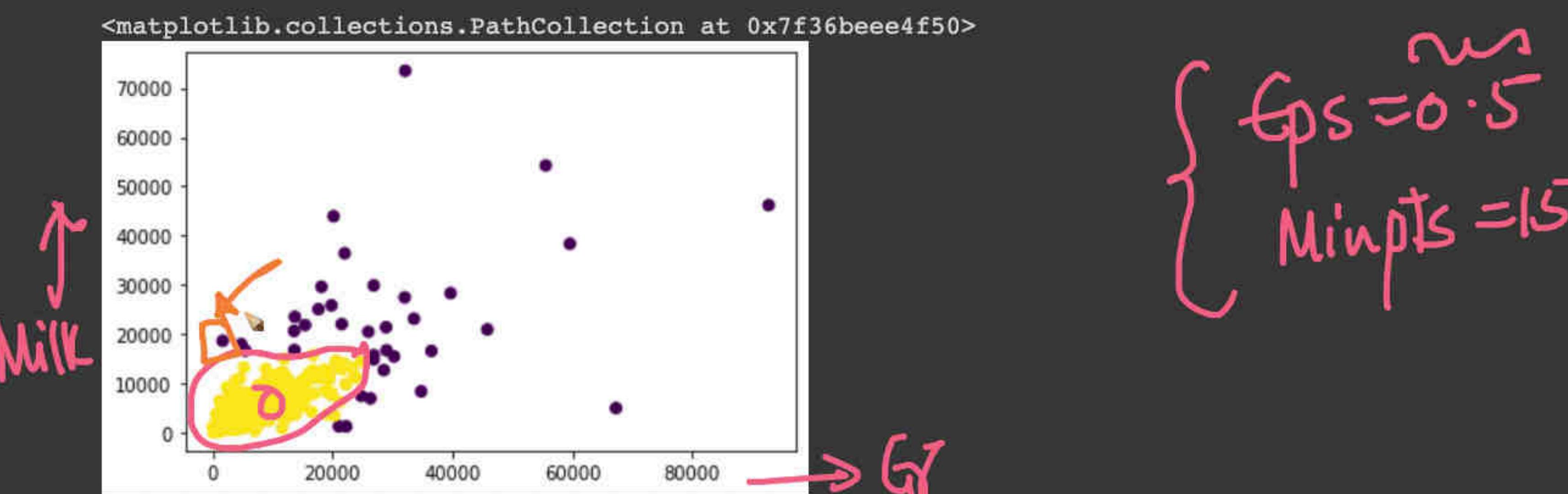
```
[ ] out_df = df.copy()
out_df['label'] = dbsc.labels_
```

```
{x} [ ] out_df['label'].value_counts()
```

0	404
-1	36
Name: label, dtype: int64	

```
[ ] plt.scatter(out_df['Grocery'], out_df['Milk'], c=out_df['label'])
```

• -0.5



DBSCAN_Hierarchical Clustering x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x sklearn.cluster.DBSCAN — scikit-learn 0.24.2 x colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=jRmeqQ3IWNVo

+ Code + Text Reconnect

```
[ ] out_df = df.copy()
out_df['label'] = dbsc.labels_
[ ] out_df['label'].value_counts()
0    404
-1    36
Name: label, dtype: int64
[ ] plt.scatter(out_df['Grocery'], out_df['Milk'], c=out_df['label'])
```

<matplotlib.collections.PathCollection at 0x7f36beee4f50>

Outliers → manually

Suggestions

{ (1) Eps, Minpts are absent

(2) all-dimensions
categorical → other/LE

KMeans

16 / 19

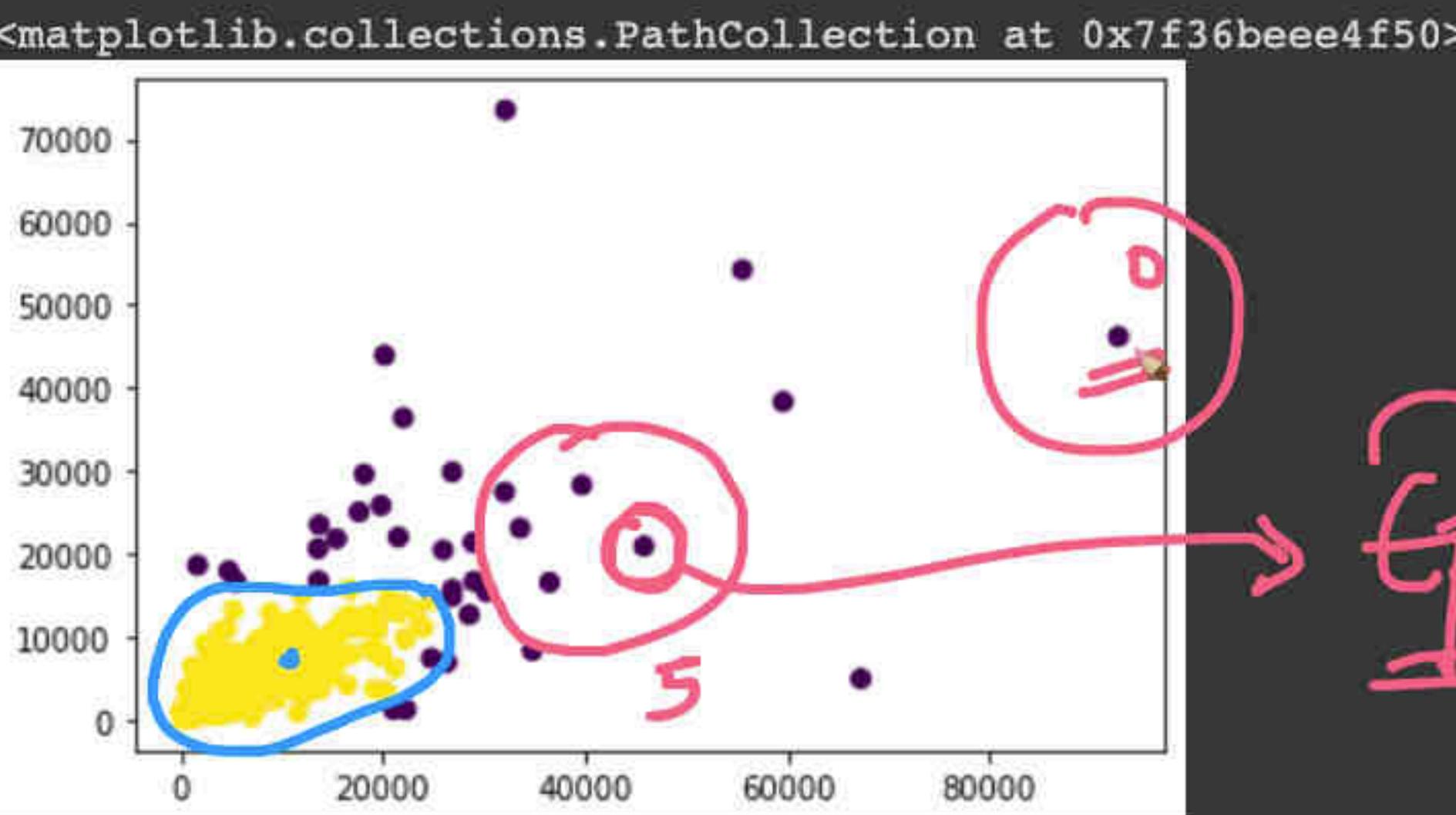
```
+ Code + Text Reconnect ▾
```

[] out_df = df.copy()
out_df['label'] = dbsc.labels_

{x} [] out_df['label'].value_counts()

0	404
-1	36
Name: label, dtype: int64	

[] plt.scatter(out_df['Grocery'], out_df['Milk'], c=out_df['label'])



✓
Eps ; pls

Polar plots ✓

t-SNE / UMAP

DBSCAN_Hierarchical Clusterin Multivariate normal distribution MultivariateNormal.png (842x640) lecture21.pptx sklearn.cluster.DBSCAN — sc

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=jRmeqQ3IWNVo

+ Code + Text Reconnect

[] out_df = df.copy()
out_df['label'] = dbsc.labels_

{x}
[] out_df['label'].value_counts()

0 404
-1 36
Name: label, dtype: int64

[] plt.scatter(out_df['Grocery'], out_df['Milk'], c=out_df['label'])

<matplotlib.collections.PathCollection at 0x7f36beee4f50>

The scatter plot displays two main clusters of data points. One cluster, located in the lower-left quadrant, is highlighted with a red oval and contains numerous yellow points. The other cluster, located in the upper-right quadrant, consists of dark purple points. The x-axis is labeled 'Grocery' and ranges from 0 to 80,000. The y-axis is labeled 'Milk' and ranges from 0 to 70,000.

```
+ Code + Text Reconnect ▾
```

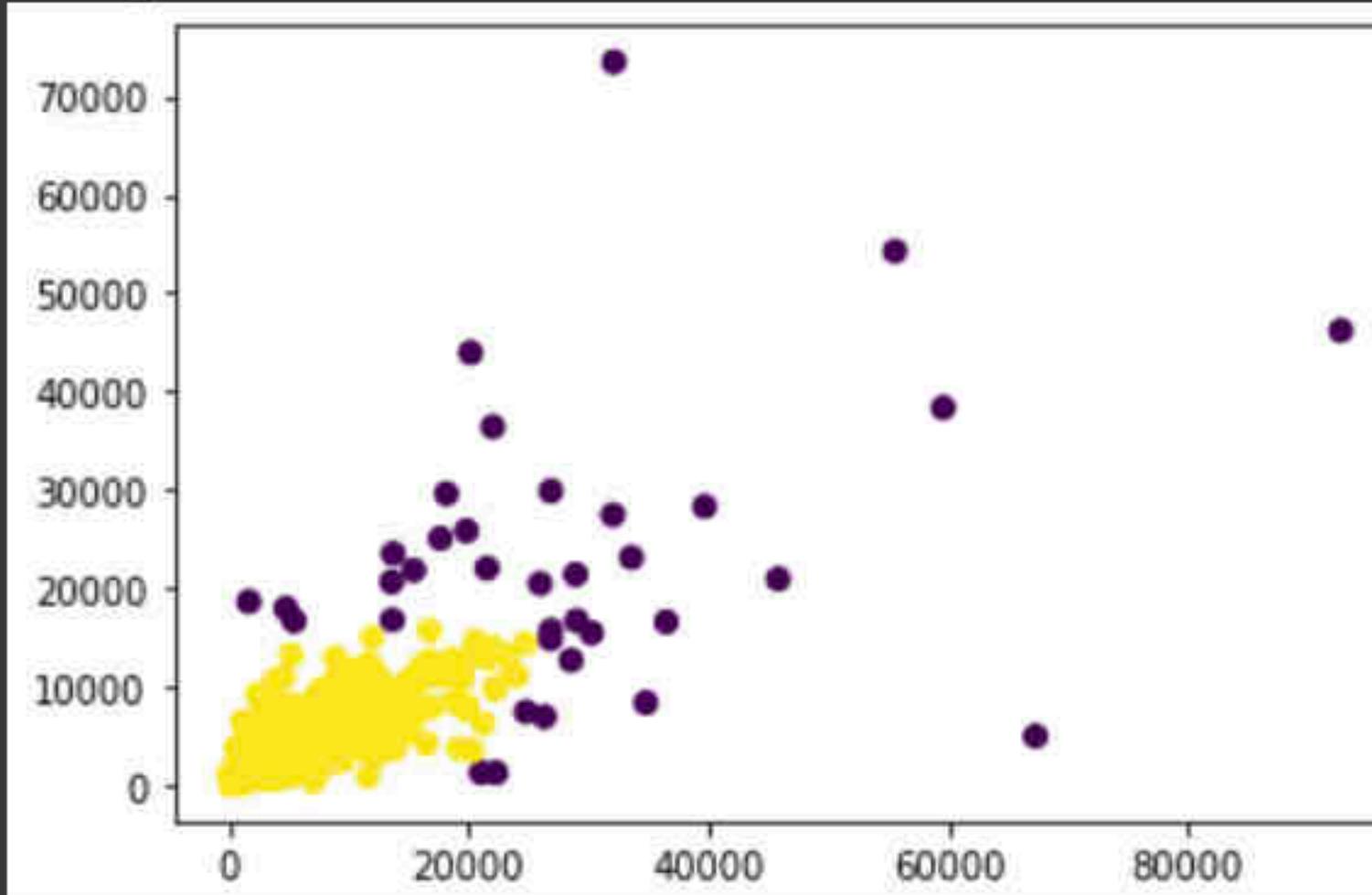
[] out_df = df.copy()
out_df['label'] = dbsc.labels_

{x} [] out_df['label'].value_counts()

0 404
-1 36
Name: label, dtype: int64

[] plt.scatter(out_df['Grocery'], out_df['Milk'], c=out_df['label'])

<matplotlib.collections.PathCollection at 0x7f36beee4f50>



{
 $\text{Eps} = 0.5$
 $\text{Minpts} = 15$

DBSCAN_Hierarchical Clusterin Multivariate normal distribution MultivariateNormal.png (842x640) lecture21.pptx sklearn.cluster.DBSCAN — sc

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=jRmeqQ3IWNVo

+ Code + Text Reconnect

[] out_df = df.copy()
out_df['label'] = dbsc.labels_

{x}
[] out_df['label'].value_counts()

0 404
-1 36
Name: label, dtype: int64

[] plt.scatter(out_df['Grocery'], out_df['Milk'], c=out_df['label'])

<matplotlib.collections.PathCollection at 0x7f36beee4f50>

The scatter plot displays two distinct clusters of data points. One cluster, colored yellow, is located in the lower-left region, primarily between 0 and 40,000 on the x-axis and 0 and 20,000 on the y-axis. The other cluster, colored dark purple, is more scattered, with points ranging from approximately 10,000 to 70,000 on both axes. There are also a few isolated dark purple points located in the upper-right quadrant.

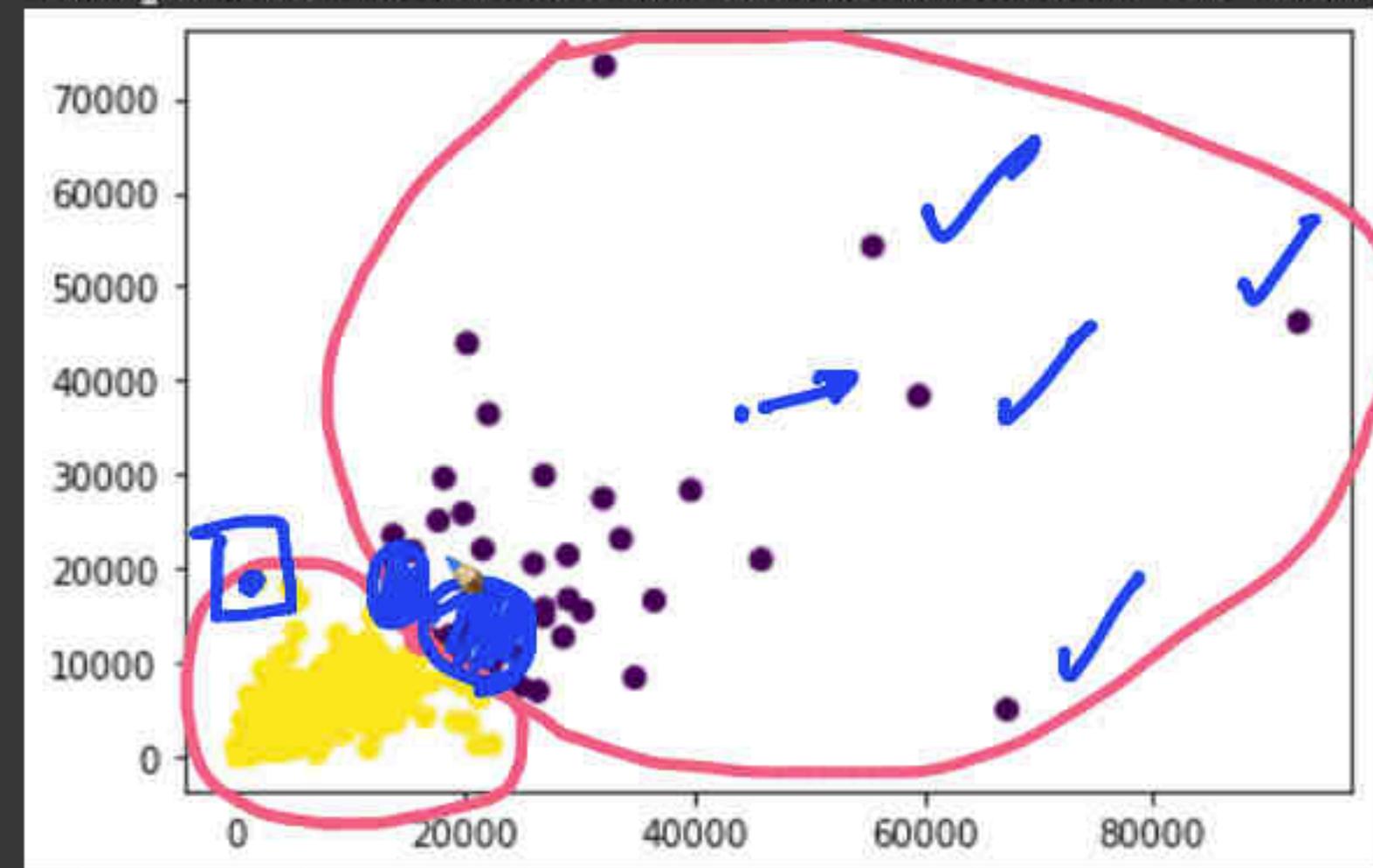
+ Code + Text

437	30243	15488	0
438	2232	1981	1
439	2510	1698	1

{x} 440 rows × 3 columns

```
[ ] plt.scatter(clusters['Grocery'], clusters['Milk'], c=clusters['label'])
```

<matplotlib.collections.PathCollection at 0x7f36be948fd0>



$k=2$

Double-click (or enter) to edit



► Finance Data clustering

+ Code + Text

✓ RAM Disk

[8] !pip install fix-yahoo-finance

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: fix-yahoo-finance in /usr/local/lib/python3.7/dist-packages (0.0.22)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from fix-yahoo-finance) (1.21.6)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from fix-yahoo-finance) (2.27.1)
Requirement already satisfied: multitasking in /usr/local/lib/python3.7/dist-packages (from fix-yahoo-finance) (0.0.10)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from fix-yahoo-finance) (1.3.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->fix-yahoo-finance) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->fix-yahoo-finance) (2022.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas->fix-yahoo-finance)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.7/dist-packages (from requests->fix-yahoo-finance) (2.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->fix-yahoo-finance) (2022.5.18)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->fix-yahoo-finance) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->fix-yahoo-finance) (1.24.3)
```



```
yf_symbols = list(map(lambda x: x + '.NS', list_of_symbols))
```

list_of_symbols

```
['ADANIPORTS.NS',
 'ASIANPAINT.NS',
 'AXISBANK.NS',
 'BAJAJ-AUTO.NS',
 'BAJFINANCE.NS',
 'BAJAJFINSV.NS',
 'BPCL.NS',
 'BHARTIARTL.NS',
 'BRITANNIA.NS',
 'CIPLA.NS',
 'COALINDIA.NS',
 'DIVISLAB.NS',
 'DRREDDY.NS',
```

ASIANP

DBSCAN_Hierarchical Clusterin x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x sklearn.cluster.DBSCAN — sc x + colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGhjYIzjBR6#scrollTo=ggSsNbSeYDPR

+ Code + Text ✓ RAM Disk Update

[8] !pip install fix-yahoo-finance

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: fix-yahoo-finance in /usr/local/lib/python3.7/dist-packages (0.0.22)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from fix-yahoo-finance) (1.21.6)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from fix-yahoo-finance) (2.27.1)
Requirement already satisfied: multitasking in /usr/local/lib/python3.7/dist-packages (from fix-yahoo-finance) (0.0.10)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from fix-yahoo-finance) (1.3.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->fix-yahoo-finance) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->fix-yahoo-finance) (2022.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas->fix-yahoo-finance)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.7/dist-packages (from requests->fix-yahoo-finance) (2.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->fix-yahoo-finance) (2022.5.18)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->fix-yahoo-finance) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->fix-yahoo-finance) (1.24.3)
```

yf_symbols = list(map(lambda x: x + '.NS', list_of_symbols))

ADANIPORTS.NS
ASIANPAINT.NS
AXISBANK.NS
BAJAJ-AUTO.NS
BAJFINANCE.NS
BAJAJFINSV.NS
BPCL.NS
BHARTIARTL.NS
BRITANNIA.NS
CIPLA.NS
COALINDIA.NS
DIVISLAB.NS
DRREDDY.NS

DBSCAN_Hierarchical Clusterin Multivariate normal distribution MultivariateNormal.png (842x640) lecture21.pptx sklearn.cluster.DBSCAN — sc

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=ggSsNbSeYDPR

+ Code + Text RAM Disk Update

[] import yfinance as yf

{x}

stock_financials = {
 ✓ 'marketCap': [],
 ✓ 'regularMarketVolume': [],
 ✓ 'earningsQuarterlyGrowth': [],
 ✓ 'bookValue': [],
 ✓ 'totalRevenue': [],
 ✓ 'returnOnAssets': [],
 ✓ 'profitMargins': [],
 ✓ 'earningsGrowth': []
}

for ticker in yf_symbols:
 stock_info = yf.Ticker(ticker).info
 stock_financials['marketCap'].append(stock_info['marketCap'])
 stock_financials['regularMarketVolume'].append(stock_info['regularMarketVolume'])
 stock_financials['earningsQuarterlyGrowth'].append(stock_info['earningsQuarterlyGrowth'])
 stock_financials['bookValue'].append(stock_info['bookValue'])
 stock_financials['totalRevenue'].append(stock_info['totalRevenue'])
 stock_financials['returnOnAssets'].append(stock_info['returnOnAssets'])
 stock_financials['profitMargins'].append(stock_info['profitMargins'])
 stock_financials['earningsGrowth'].append(stock_info['earningsGrowth'])

df = pd.DataFrame(stock_financials)
df.head

ROCE, RoI --

RAM Disk

24 / 24

DBSCAN_Hierarchical Clusterin Multivariate normal distribution MultivariateNormal.png (842x640) lecture21.pptx sklearn.cluster.DBSCAN — sc

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=ggSsNbSeYDPR

+ Code + Text RAM Disk Update

[] import yfinance as yf

{x}

stock_financials = {
 'marketCap': [],
 'regularMarketVolume': [],
 'earningsQuarterlyGrowth': [],
 'bookValue': [],
 'totalRevenue': [],
 'returnOnAssets': [],
 'profitMargins': [],
 'earningsGrowth': []
}

for ticker in yf_symbols:
 stock_info = yf.Ticker(ticker).info
 stock_financials['marketCap'].append(stock_info['marketCap'])
 stock_financials['regularMarketVolume'].append(stock_info['regularMarketVolume'])
 stock_financials['earningsQuarterlyGrowth'].append(stock_info['earningsQuarterlyGrowth'])
 stock_financials['bookValue'].append(stock_info['bookValue'])
 stock_financials['totalRevenue'].append(stock_info['totalRevenue'])
 stock_financials['returnOnAssets'].append(stock_info['returnOnAssets'])
 stock_financials['profitMargins'].append(stock_info['profitMargins'])
 stock_financials['earningsGrowth'].append(stock_info['earningsGrowth'])

df = pd.DataFrame(stock_financials)
df.head

+ Code + Text

✓ RAM Disk

```
stock_financials = [
    {'marketCap': [],
     'regularMarketVolume': [],
     'earningsQuarterlyGrowth': [],
     'bookValue': [],
     'totalRevenue': [],
     'returnOnAssets': [],
     'profitMargins': [],
     'earningsGrowth': []
}

for ticker in yf_symbols:
    stock_info = yf.Ticker(ticker).info
    stock_financials['marketCap'].append(stock_info['marketCap'])
    stock_financials['regularMarketVolume'].append(stock_info['regularMarketVolume'])
    stock_financials['earningsQuarterlyGrowth'].append(stock_info['earningsQuarterlyGrowth'])
    stock_financials['bookValue'].append(stock_info['bookValue'])
    stock_financials['totalRevenue'].append(stock_info['totalRevenue'])
    stock_financials['returnOnAssets'].append(stock_info['returnOnAssets'])
    stock_financials['profitMargins'].append(stock_info['profitMargins'])
    stock_financials['earningsGrowth'].append(stock_info['earningsGrowth'])
```

▶ df = pd.DataFrame(stock_financials)
df.head

	marketCap	regularMarketVolume	earningsQuarterlyGrowth	bookValue	totalRevenue	returnOnAssets	profitMargins	earningsGrowth
0	1539072720896	2839851		-0.205	181.165	1.593403e+11	0.05149	0.29673
1	2596018061312	1742929		-0.002	143.991	2.898187e+11	0.11496	0.10457
2	2041133006949						0.01399	0.32270

+ Code + Text

```
[ ] stock_financials['marketCap'].append(stock_info['marketCap'])
stock_financials['regularMarketVolume'].append(stock_info['regularMarketVolume'])
stock_financials['earningsQuarterlyGrowth'].append(stock_info['earningsQuarterlyGrowth'])
stock_financials['bookValue'].append(stock_info['bookValue'])
stock_financials['totalRevenue'].append(stock_info['totalRevenue'])
stock_financials['returnOnAssets'].append(stock_info['returnOnAssets'])
stock_financials['profitMargins'].append(stock_info['profitMargins'])
stock_financials['earningsGrowth'].append(stock_info['earningsGrowth'])
```

RAM Disk

df = pd.DataFrame(stock_financials)

df.head

	marketCap	regularMarketVolume	earningsQuarterlyGrowth	bookValue	totalRevenue	returnOnAssets	profitMargins	earningsGrowth
0	1539072720896	2839851	-0.205	181.165	1.593403e+11	0.05149	0.29673	-0.235
1	2596018061312	1742929	-0.002	143.991	2.898187e+11	0.11496	0.10457	-0.002
2	2041133006848	9280420	0.502	385.482	4.375369e+11	0.01288	0.32270	0.498
3	1097901801472	346513	-0.016	1033.043	3.435395e+11	0.11259	0.17948	-0.015
4	3702931324928	1334050	0.797	724.657	1.708877e+11	0.03660	0.41128	0.793

{}

```
[ ] df.shape
```

(50, 8)

```
[ ] df.info()
```

RAM Disk

27 / 27

+ Code + Text

	Column	Non-NaN Count	Dtype
0	marketCap	50	non-null int64
1	regularMarketVolume	50	non-null int64
2	earningsQuarterlyGrowth	48	non-null float64
3	bookValue	49	non-null float64
4	totalRevenue	49	non-null float64
5	returnOnAssets	22	non-null float64
6	profitMargins	50	non-null float64
7	earningsGrowth	48	non-null float64

dtypes: float64(6), int64(2)
memory usage: 3.2 KB

[] import yfinance as yf ✓
 [] stock_prices = yf.download(yf_symbols, start='2020-01-01')['Adj Close'] ✓
 stock_prices.columns = list_of_symbols
 [*****100%*****] 50 of 50 completed

[] stock_prices.shape
 (606, 50)

[] stock_prices.head()
 ADANIPORTS ASIANPAINT AXISBANK BAJAJ-AUTO BAJFINANCE BAJAJFINSV BPCL BHARTIARTL BRITANNIA CIPLA ... SUNPHARMA

Date

RAM Disk

lsb 2 3 4 : 606 → everyday

SI S2 ... Sn

<img alt="A hand-drawn diagram on the right side of the screen. It shows a vertical stack of rectangles of varying heights, representing data points. Above the stack, there are labels 'lsb' and '2 3 4'. To the right of the stack, there are labels '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '30', '31', '32', '33', '34', '35', '36', '37', '38', '39', '40', '41', '42', '43', '44', '45', '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56', '57', '58', '59', '60', '61', '62', '63', '64', '65', '66', '67', '68', '69', '70', '71', '72', '73', '74', '75', '76', '77', '78', '79', '80', '81', '82', '83', '84', '85', '86', '87', '88', '89', '90', '91', '92', '93', '94', '95', '96', '97', '98', '99', '100', '101', '102', '103', '104', '105', '106', '107', '108', '109', '110', '111', '112', '113', '114', '115', '116', '117', '118', '119', '120', '121', '122', '123', '124', '125', '126', '127', '128', '129', '130', '131', '132', '133', '134', '135', '136', '137', '138', '139', '140', '141', '142', '143', '144', '145', '146', '147', '148', '149', '150', '151', '152', '153', '154', '155', '156', '157', '158', '159', '160', '161', '162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '173', '174', '175', '176', '177', '178', '179', '180', '181', '182', '183', '184', '185', '186', '187', '188', '189', '190', '191', '192', '193', '194', '195', '196', '197', '198', '199', '200', '201', '202', '203', '204', '205', '206', '207', '208', '209', '210', '211', '212', '213', '214', '215', '216', '217', '218', '219', '220', '221', '222', '223', '224', '225', '226', '227', '228', '229', '230', '231', '232', '233', '234', '235', '236', '237', '238', '239', '240', '241', '242', '243', '244', '245', '246', '247', '248', '249', '250', '251', '252', '253', '254', '255', '256', '257', '258', '259', '260', '261', '262', '263', '264', '265', '266', '267', '268', '269', '270', '271', '272', '273', '274', '275', '276', '277', '278', '279', '280', '281', '282', '283', '284', '285', '286', '287', '288', '289', '290', '291', '292', '293', '294', '295', '296', '297', '298', '299', '300', '301', '302', '303', '304', '305', '306', '307', '308', '309', '310', '311', '312', '313', '314', '315', '316', '317', '318', '319', '320', '321', '322', '323', '324', '325', '326', '327', '328', '329', '330', '331', '332', '333', '334', '335', '336', '337', '338', '339', '340', '341', '342', '343', '344', '345', '346', '347', '348', '349', '350', '351', '352', '353', '354', '355', '356', '357', '358', '359', '360', '361', '362', '363', '364', '365', '366', '367', '368', '369', '370', '371', '372', '373', '374', '375', '376', '377', '378', '379', '380', '381', '382', '383', '384', '385', '386', '387', '388', '389', '390', '391', '392', '393', '394', '395', '396', '397', '398', '399', '400', '401', '402', '403', '404', '405', '406', '407', '408', '409', '410', '411', '412', '413', '414', '415', '416', '417', '418', '419', '420', '421', '422', '423', '424', '425', '426', '427', '428', '429', '430', '431', '432', '433', '434', '435', '436', '437', '438', '439', '440', '441', '442', '443', '444', '445', '446', '447', '448', '449', '450', '451', '452', '453', '454', '455', '456', '457', '458', '459', '460', '461', '462', '463', '464', '465', '466', '467', '468', '469', '470', '471', '472', '473', '474', '475', '476', '477', '478', '479', '480', '481', '482', '483', '484', '485', '486', '487', '488', '489', '490', '491', '492', '493', '494', '495', '496', '497', '498', '499', '500', '501', '502', '503', '504', '505', '506', '507', '508', '509', '510', '511', '512', '513', '514', '515', '516', '517', '518', '519', '520', '521', '522', '523', '524', '525', '526', '527', '528', '529', '530', '531', '532', '533', '534', '535', '536', '537', '538', '539', '540', '541', '542', '543', '544', '545', '546', '547', '548', '549', '540', '541', '542', '543', '544', '545', '546', '547', '548', '549', '550', '551', '552', '553', '554', '555', '556', '557', '558', '559', '550', '551', '552', '553', '554', '555', '556', '557', '558', '559', '560', '561', '562', '563', '564', '565', '566', '567', '568', '569', '560', '561', '562', '563', '564', '565', '566', '567', '568', '569', '570', '571', '572', '573', '574', '575', '576', '577', '578', '579', '580', '581', '582', '583', '584', '585', '586', '587', '588', '589', '580', '581', '582', '583', '584', '585', '586', '587', '588', '589', '590', '591', '592', '593', '594', '595', '596', '597', '598', '590', '591', '592', '593', '594', '595', '596', '597', '598', '599', '600', '601', '602', '603', '604', '605', '606']</p>

DBSCAN_Hierarchical Clusterin x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x sklearn.cluster.DBSCAN — sc x + colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGhjYIzjBR6#scrollTo=-HTSDIrYdBWt

+ Code + Text

[] stock_prices.columns = list_of_symbols

[*****100%*****] 50 of 50 completed

{x} [] stock_prices.shape

(606, 50)

[] stock_prices.head()

	ADANIPORTS	ASIANPAINT	AXISBANK	BAJAJ-AUTO	BAJFINANCE	BAJAJFINSV	BPCL	BHARTIARTL	BRITANNIA	CIPLA	...	SUNPHARMA
Date												
2020-01-01	370.923370	1770.856567	748.700012	2913.946289	9370.917969	4214.786133	451.707916	401.235718	2883.703125	472.062744	...	422.185211 319.
2020-01-02	376.325409	1768.338379	756.950012	2887.027588	9497.965820	4229.478516	453.601257	397.889709	2896.747559	469.682129	...	422.817139 319.
2020-01-03	375.687012	1729.577393	742.950012	2841.747314	9338.342773	4177.084473	453.501617	394.584534	2882.896729	466.160767	...	432.197937 314.
2020-01-06	373.427948	1685.878662	723.250000	2809.926270	9035.437500	3981.102051	448.070740	382.873474	2867.005859	462.986542	...	427.677643 308.
2020-01-07	377.946014	1702.913940	725.750000	2810.203613	9088.344727	3992.009277	443.536713	376.222260	2880.809570	464.821655	...	433.947693 310.

5 rows × 50 columns

RAM Disk

Update

File Edit View History Bookmarks Profiles Tab Window Help

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGhjYIzjBR6#scrollTo=-HTSDIrYdBWt

29 / 29

DBSCAN_Hierarchical Clusterin x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x sklearn.cluster.DBSCAN — sc x + colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGhjYIzjBR6#scrollTo=-HTSDIrYdBWt

+ Code + Text

```
[ ] stock_prices.columns = list_of_symbols
```

[*****100%*****] 50 of 50 completed

{x} [] stock_prices.shape

(606, 50)

[] stock_prices.head()

2020

	ADANIPORTS	ASIANPAINT	AXISBANK	BAJAJ-AUTO	BAJFINANCE	BAJAJFINSV	BPCL	BHARTIARTL	BRITANNIA	CIPLA	...	SUNPHARMA
Date												
2020-01-01	370.923370	1770.856567	748.700012	2913.946289	9370.917969	4214.786133	451.707916	401.235718	2883.703125	472.062744	...	422.185211 319.
2020-01-02	376.325409	1768.338379	756.950012	2887.027588	9497.965820	4229.478516	453.601257	397.889709	2896.747559	469.682129	...	422.817139 319.
2020-01-03	375.687012	1729.577393	742.950012	2841.747314	9338.342773	4177.084473	453.501617	394.584534	2882.896729	466.160767	...	432.197937 314.
2020-01-06	373.427948	1685.878662	723.250000	2809.926270	9035.437500	3981.102051	448.070740	382.873474	2867.005859	462.986542	...	427.677643 308.
2020-01-07	377.946014	1702.913940	725.750000	2810.203613	9088.344727	3992.009277	443.536713	376.222260	2880.809570	464.821655	...	433.947693 310.

5 rows × 50 columns

RAM Disk

Update

30 / 30

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGhjYIzjBR6#scrollTo=KQ4aycjjdhw4

+ Code + Text

RAM Disk

	2020-01-02	376.325409	1768.338379	756.950012	2887.027588	9497.965820	4229.478516	453.601257	397.889709	2896.747559	469.682129	...	422.817139	319.
{x}	2020-01-03	375.687012	1729.577393	742.950012	2841.747314	9338.342773	4177.084473	453.501617	394.584534	2882.896729	466.160767	...	432.197937	314.
	2020-01-06	373.427948	1685.878662	723.250000	2809.926270	9035.437500	3981.102051	448.070740	382.873474	2867.005859	462.986542	...	427.677643	308.
	2020-01-07	377.946014	1702.913940	725.750000	2810.203613	9088.344727	3992.009277	443.536713	376.222260	2880.809570	464.821655	...	433.947693	310.
		5 rows x 50 columns												



```
##2020 returns
price_2020 = stock_prices.loc["2020-01-02 00:00:00":"2020-12-31 00:00:00"]
stock_prices.loc['returns_2020'] = (price_2020.loc['2020-12-31 00:00:00'] / price_2020.loc['2020-01-02 00:00:00'] - 1)*100
stock_prices
```

Date	ADANIPORTS	ASIANPAINT	AXISBANK	BAJAJ-AUTO	BAJFINANCE	BAJAJFINSV	BPCL	BHARTIARTL	BRITANNIA	Cipla	...	SUNPHAR
2020-01-01 00:00:00	370.923370	1770.856567	748.700012	2913.946289	9370.917969	4214.786133	451.707916	401.235718	2883.703125	472.062744	...	422.1852
2020-01-02 00:00:00	376.325409	1768.338379	756.950012	2887.027588	9497.965820	4229.478516	453.601257	397.889709	2896.747559	469.682129	...	422.8171



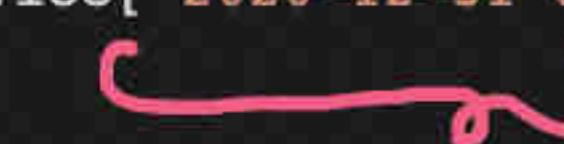
DBSCAN_Hierarchical Clusterin x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x sklearn.cluster.DBSCAN — sc x + colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGhjYIzjBR6#scrollTo=KQ4aycjjdhw4

+ Code + Text RAM Disk

2020-01-02	376.325409	1768.338379	756.950012	2887.027588	9497.965820	4229.478516	453.601257	397.889709	2896.747559	469.682129	...	422.817139	319.
2020-01-03	375.687012	1729.577393	742.950012	2841.747314	9338.342773	4177.084473	453.501617	394.584534	2882.896729	466.160767	...	432.197937	314.
2020-01-06	373.427948	1685.878662	723.250000	2809.926270	9035.437500	3981.102051	448.070740	382.873474	2867.005859	462.986542	...	427.677643	308.
2020-01-07	377.946014	1702.913940	725.750000	2810.203613	9088.344727	3992.009277	443.536713	376.222260	2880.809570	464.821655	...	433.947693	310.
5 rows x 50 columns													



```
##2020 returns
price_2020 = stock_prices.loc["2020-01-02 00:00:00":"2020-12-31 00:00:00"]
stock_prices.loc['returns_2020'] = (price_2020.loc['2020-12-31 00:00:00'] / price_2020.loc['2020-01-02 00:00:00'] - 1)*100
stock_prices
```




	ADANIPORTS	ASIANPAINT	AXISBANK	BAJAJ-AUTO	BAJFINANCE	BAJAJFINSV	BPCL	BHARTIARTL	BRITANNIA	CIPLA	...	SUNPHAR
--	------------	------------	----------	------------	------------	------------	------	------------	-----------	-------	-----	---------

Date

2020-01-01 00:00:00	370.923370	1770.856567	748.700012	2913.946289	9370.917969	4214.786133	451.707916	401.235718	2883.703125	472.062744	...	422.1852
2020-01-02 00:00:00	376.325409	1768.338379	756.950012	2887.027588	9497.965820	4229.478516	453.601257	397.889709	2896.747559	469.682129	...	422.8171



colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGhjYIzjBR6#scrollTo=KQ4aycjjdhw4

+ Code + Text

RAM Disk

	2020-01-02	376.325409	1768.338379	756.950012	2887.027588	9497.965820	4229.478516	453.601257	397.889709	2896.747559	469.682129	...	422.817139	319.
{x}	2020-01-03	375.687012	1729.577393	742.950012	2841.747314	9338.342773	4177.084473	453.501617	394.584534	2882.896729	466.160767	...	432.197937	314.
	2020-01-06	373.427948	1685.878662	723.250000	2809.926270	9035.437500	3981.102051	448.070740	382.873474	2867.005859	462.986542	...	427.677643	308.
	2020-01-07	377.946014	1702.913940	725.750000	2810.203613	9088.344727	3992.009277	443.536713	376.222260	2880.809570	464.821655	...	433.947693	310.
		5 rows x 50 columns												

(end
start - 1) * 100 = %age

```
#2020 returns
price_2020 = stock_prices.loc["2020-01-02 00:00:00":"2020-12-31 00:00:00"]
stock_prices.loc['returns_2020'] = (price_2020.loc['2020-12-31 00:00:00'] / price_2020.loc['2020-01-02 00:00:00'] - 1)*100
stock_prices
```

Date	ADANIPORTS	ASIANPAINT	AXISBANK	BAJAJ-AUTO	BAJFINANCE	BAJAJFINSV	BPCL	BHARTIARTL	BRITANNIA	Cipla	...	SUNPHAR
2020-01-01 00:00:00	370.923370	1770.856567	748.700012	2913.946289	9370.917969	4214.786133	451.707916	401.235718	2883.703125	472.062744	...	422.1852
2020-01-02 00:00:00	376.325409	1768.338379	756.950012	2887.027588	9497.965820	4229.478516	453.601257	397.889709	2896.747559	469.682129	...	422.8171

DBSCAN_Hierarchical Clusterin x Multivariate normal distribution x MultivariateNormal.png (842x640) x lecture21.pptx x sklearn.cluster.DBSCAN — sc x + colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGhjYizlBR6#scrollTo=UUgytmz7d7cz

+ Code + Text RAM Disk

[]	ASIANPAINT	1770.856567	1768.338379	1729.577393	1685.878662	1702.913940	1707.259033	1750.463989	1770.214844	1782.410889	1796.483276	...	2834.8
[]	AXISBANK	748.700012	756.950012	742.950012	723.250000	725.750000	724.500000	742.849976	740.049988	737.400024	747.900024	...	688.2
{x}	BAJAJ-AUTO	2913.946289	2887.027588	2841.747314	2809.926270	2810.203613	2829.906738	2854.373779	2868.711914	2862.097900	2869.082031	...	3833.0
[]	BAJFINANCE	9370.917969	9497.965820	9338.342773	9035.437500	9088.344727	9138.154297	9387.704102	9364.722656	9447.655273	9547.175781	...	12758.4
	5 rows x 607 columns												

prices = stock_prices.iloc[:, -1]
df.index = stock_prices.index
df['return_2020'] = prices
df.head()

	marketCap	regularMarketVolume	earningsQuarterlyGrowth	bookValue	totalRevenue	returnOnAssets	profitMargins	earningsGrc
ADANIPORTS	1539072720896	2839851	-0.205	181.165	1.593403e+11	0.05149	0.29673	-0
ASIANPAINT	2596018061312	1742929	-0.002	143.991	2.898187e+11	0.11496	0.10457	-0
AXISBANK	2041133006848	9280420	0.502	385.482	4.375369e+11	0.01288	0.32270	0
BAJAJ-AUTO	1097901801472	346513	-0.016	1033.043	3.435395e+11	0.11259	0.17948	-0
BAJFINANCE	3702931324928	1334050	0.797	724.657	1.708877e+11	0.03660	0.41128	0

DBSCAN_Hierarchical Clusterin x Multivariate normal distribution x MultivariateNormal.png (842x640) x lecture21.pptx x sklearn.cluster.DBSCAN — sc x + colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=DmFBTokieJxR

+ Code + Text

RAM Disk

[] []

{x}

[]

drop null values
df.isna().sum()

marketCap 0
regularMarketVolume 0
earningsQuarterlyGrowth 2
bookValue 1
totalRevenue 1
returnOnAssets 28
profitMargins 0
earningsGrowth 2
return_2020 0
dtype: int64

50 rows

[] df['returnOnAssets'] = df['returnOnAssets'].replace(np.nan, 0)

[] df.isna().sum()

marketCap 0
regularMarketVolume 0
earningsQuarterlyGrowth 2
bookValue 1
totalRevenue 1
returnOnAssets 0
profitMargins 0
earningsGrowth 2

50 rows

File Edit View History Bookmarks Profiles Tab Window Help

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=DmFBTokieJxR

RAM Disk

Code Text

[] []

{x}

[]

drop null values
df.isna().sum()

marketCap 0
regularMarketVolume 0
earningsQuarterlyGrowth 2
bookValue 1
totalRevenue 1
returnOnAssets 28
profitMargins 0
earningsGrowth 2
return_2020 0
dtype: int64

50 rows

[] df['returnOnAssets'] = df['returnOnAssets'].replace(np.nan, 0)

[] df.isna().sum()

marketCap 0
regularMarketVolume 0
earningsQuarterlyGrowth 2
bookValue 1
totalRevenue 1
returnOnAssets 0
profitMargins 0
earningsGrowth 2

50 rows

File Edit View History Bookmarks Profiles Tab Window Help

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=DmFBTokieJxR

RAM Disk

Code Text

[] []

{x}

[]

drop null values
df.isna().sum()

marketCap 0
regularMarketVolume 0
earningsQuarterlyGrowth 2
bookValue 1
totalRevenue 1
returnOnAssets 28
profitMargins 0
earningsGrowth 2
return_2020 0
dtype: int64

50 rows

[] df['returnOnAssets'] = df['returnOnAssets'].replace(np.nan, 0)

[] df.isna().sum()

marketCap 0
regularMarketVolume 0
earningsQuarterlyGrowth 2
bookValue 1
totalRevenue 1
returnOnAssets 0
profitMargins 0
earningsGrowth 2

50 rows

DBSCAN_Hierarchical Clustering x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x sklearn.cluster.DBSCAN — scikit-learn 0.24.2 documentation x colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=ePGmlW_eevXV

+ Code + Text RAM ✓ Disk ✓ Update

RAM ✓ Disk ✓

Hierarchical Clustering

{x}

Wards

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df)
X = scaler.transform(df)

[] scaled_df = pd.DataFrame(X, columns=df.columns, index=df.index)

[] # import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

Refer <https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.linkage>
Z = sch.linkage(scaled_df, method='ward') #linkage = ward

[] Z.shape
(46, 4)

[] fig, ax = plt.subplots(figsize=(14, 8))
sch.dendrogram(Z, labels=scaled_df.index, ax=ax, color_threshold=2)
plt.xticks(rotation=90)

36 / 36

DBSCAN_Hierarchical Clusterin x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x sklearn.cluster.DBSCAN — sc x +



+ Code + Text

```
[ ] earningsQuarterlyGrowth    2
     bookValue                   1
     totalRevenue                1
     returnOnAssets              28
{x} profitMargins               0
     earningsGrowth              2
     return_2020                 0
     dtype: int64
```

```
[ ] df['returnOnAssets'] = df['returnOnAssets'].replace(np.nan, 0)
```

```
[ ] df.isna().sum()
```

```
✓ marketCap                  0
✓ regularMarketVolume         0
✓ earningsQuarterlyGrowth    2
✓ bookValue                   1
✓ totalRevenue                1
✓ returnOnAssets              0
✓ profitMargins               0
✓ earningsGrowth              2
✓ return_2020                 0
     dtype: int64
```

```
[ ] df.dropna(axis=0, inplace=True)
     df.shape
```

```
(47, 9)
```

```
+ Code + Text
[ ] scaler = StandardScaler()
scaler.fit(df)
X = scaler.transform(df)

{x}
[ ] scaled_df = pd.DataFrame(X, columns=df.columns, index=df.index)

# import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

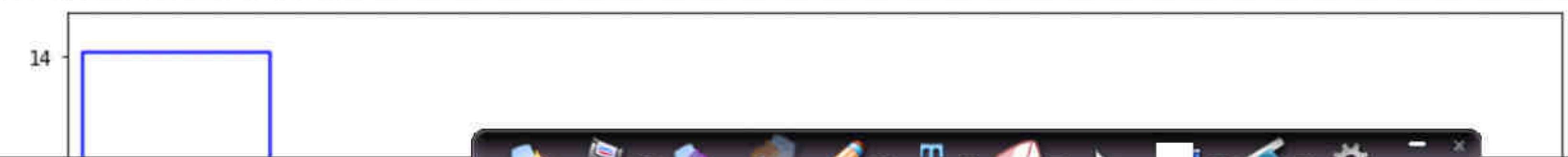
# Refer https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.linkage
Z = sch.linkage(scaled_df, method='ward') #linkage = ward

[ ] Z.shape
(46, 4)
```

→ Scipy
→ sklearn

```
▶ fig, ax = plt.subplots(figsize=(14, 8))
sch.dendrogram(Z, labels=scaled_df.index, ax=ax, color_threshold=2)
plt.xticks(rotation=90)
ax.set_ylabel('distance')
```

Text(0, 0.5, 'distance')



colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=ON7loHfxfQIZ

+ Code + Text

RAM Disk

Update

[] scaler.fit(df)
X = scaler.transform(df)

{x} [] scaled_df = pd.DataFrame(X, columns=df.columns, index=df.index)

[] # import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

Refer <https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.linkage>
Z = sch.linkage(scaled_df, method='ward') #linkage = ward

[] Z.shape

(46, 4)

[] fig, ax = plt.subplots(figsize=(14, 8))
sch.dendrogram(Z, labels=scaled_df.index, ax=ax, color_threshold=2)
plt.xticks(rotation=90)
ax.set_ylabel('distance')

[] Text(0, 0.5, 'distance')

39 / 39

DBSCAN_Hierarchical Clustering x Multivariate normal distribution x MultivariateNormal.png (842x640) x lecture21.pptx x sklearn.cluster.DBSCAN — scikit-learn 0.24.2 documentation x colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=ON7loHfxfQIZ

+ Code + Text

RAM Disk

[] scaler.fit(df)
X = scaler.transform(df)

{x} [] scaled_df = pd.DataFrame(X, columns=df.columns, index=df.index)

[] # import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

Refer <https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.linkage>
Z = sch.linkage(scaled_df, method='ward') #linkage = ward

[] Z.shape

(46, 4)

[] fig, ax = plt.subplots(figsize=(14, 8))
sch.dendrogram(Z, labels=scaled_df.index, ax=ax, color_threshold=2)
plt.xticks(rotation=90)
ax.set_ylabel('distance')

[] Text(0, 0.5, 'distance')

40 / 40

DBSCAN_Hierarchical Clustering x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x sklearn.cluster.DBSCAN — scikit-learn 0.23.2 documentation x +

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGhjYIzjBR6#scrollTo=ON7loHfxfQIZ

+ Code + Text

RAM Disk

import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

Refer <https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.linkage>
Z = sch.linkage(scaled_df, method='ward') #linkage = ward

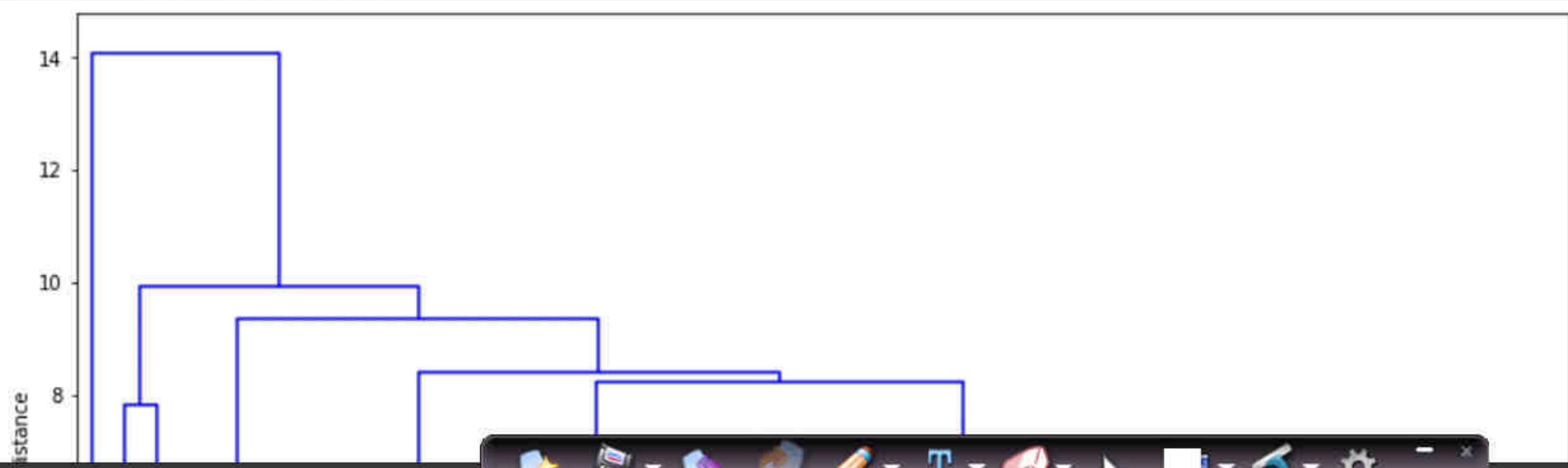
[] Z.shape

(46, 4)

fig, ax = plt.subplots(figsize=(14, 8))
sch.dendrogram(Z, labels=scaled_df.index, ax=ax, color_threshold=2)
plt.xticks(rotation=90)
ax.set_ylabel('distance')



Text(0, 0.5, 'distance')



DBSCAN_Hierarchical Clustering x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x sklearn.cluster.DBSCAN — scikit-learn 0.23.2 documentation x + colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzjBR6#scrollTo=ON7loHfxfQIZ

+ Code + Text

RAM Disk

Update

import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

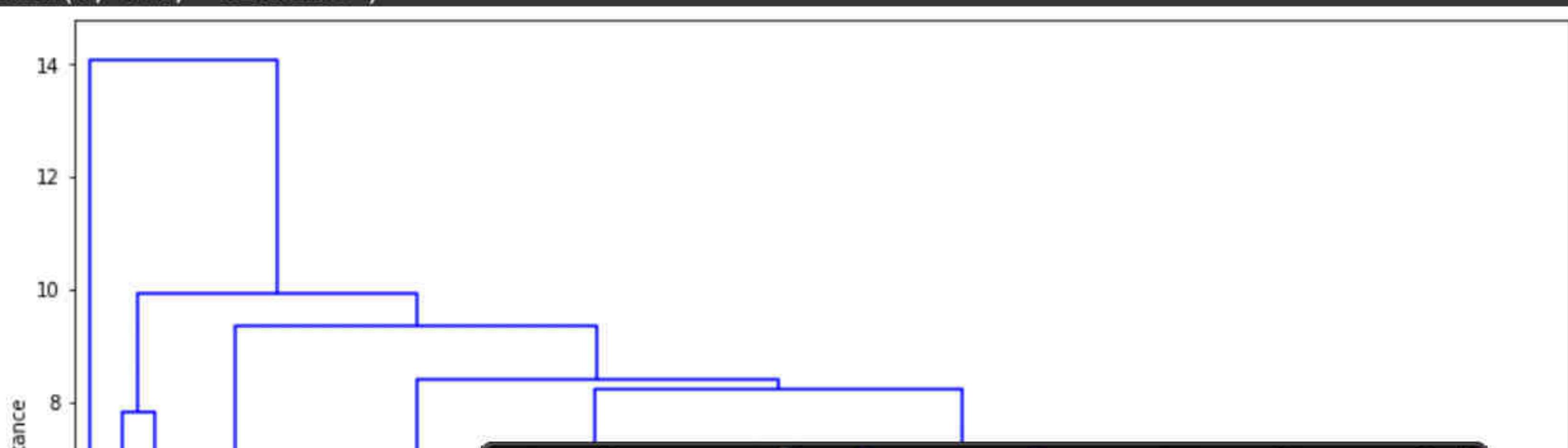
Refer <https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.linkage>
Z = sch.linkage(scaled_df, method='ward') #linkage = ward

[] Z.shape

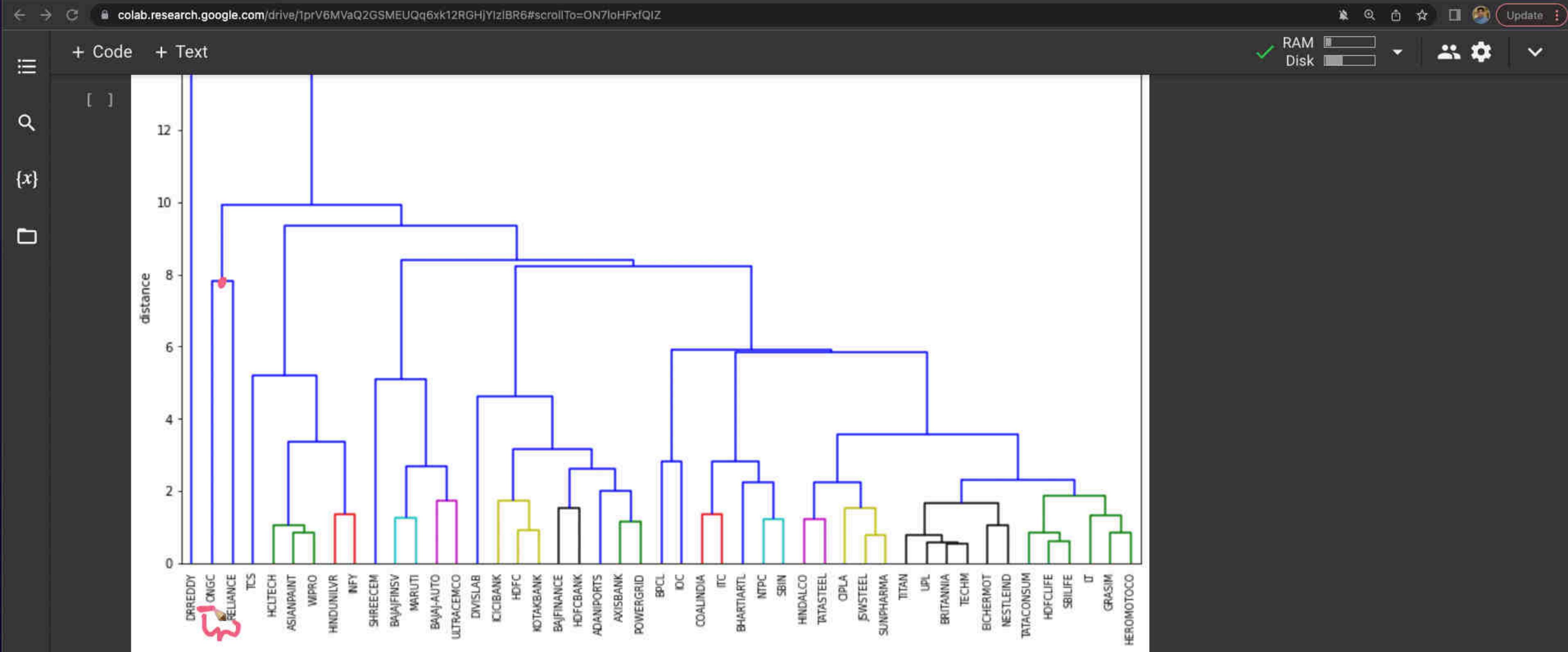
(46, 4)

fig, ax = plt.subplots(figsize=(14, 8))
sch.dendrogram(Z, labels=scaled_df.index, ax=ax, color_threshold=2)
plt.xticks(rotation=90)
ax.set_ylabel('distance')

Text(0, 0.5, 'distance')



42 / 42

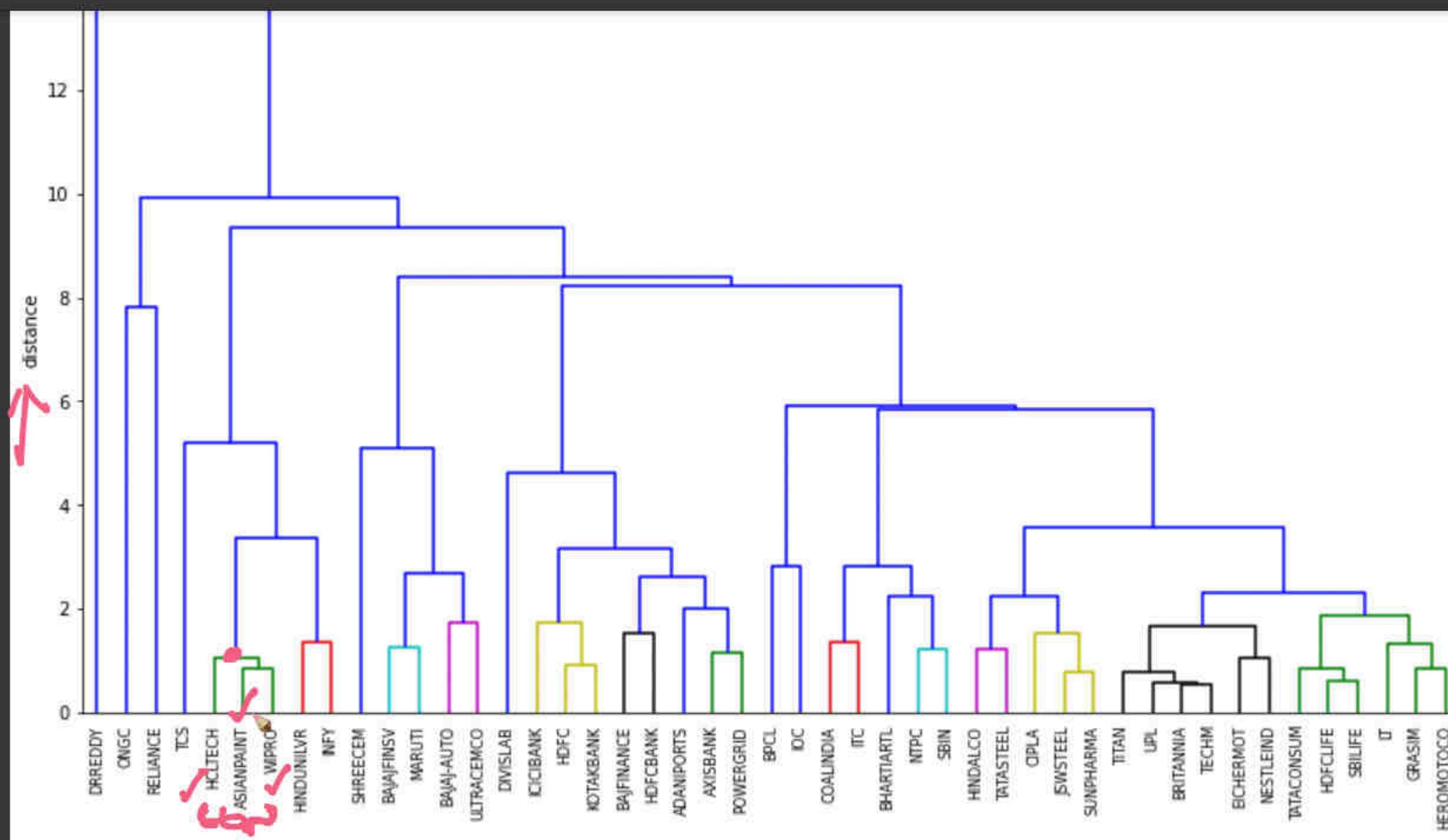


```
[ ] # import hierarchical clustering libraries
from sklearn.cluster import AgglomerativeClustering
```

create clusters

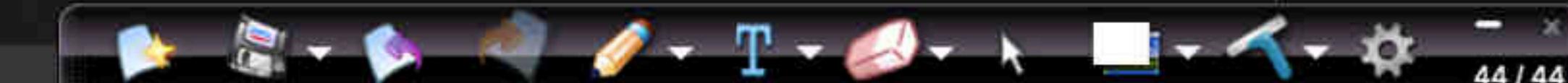
+ Code + Text

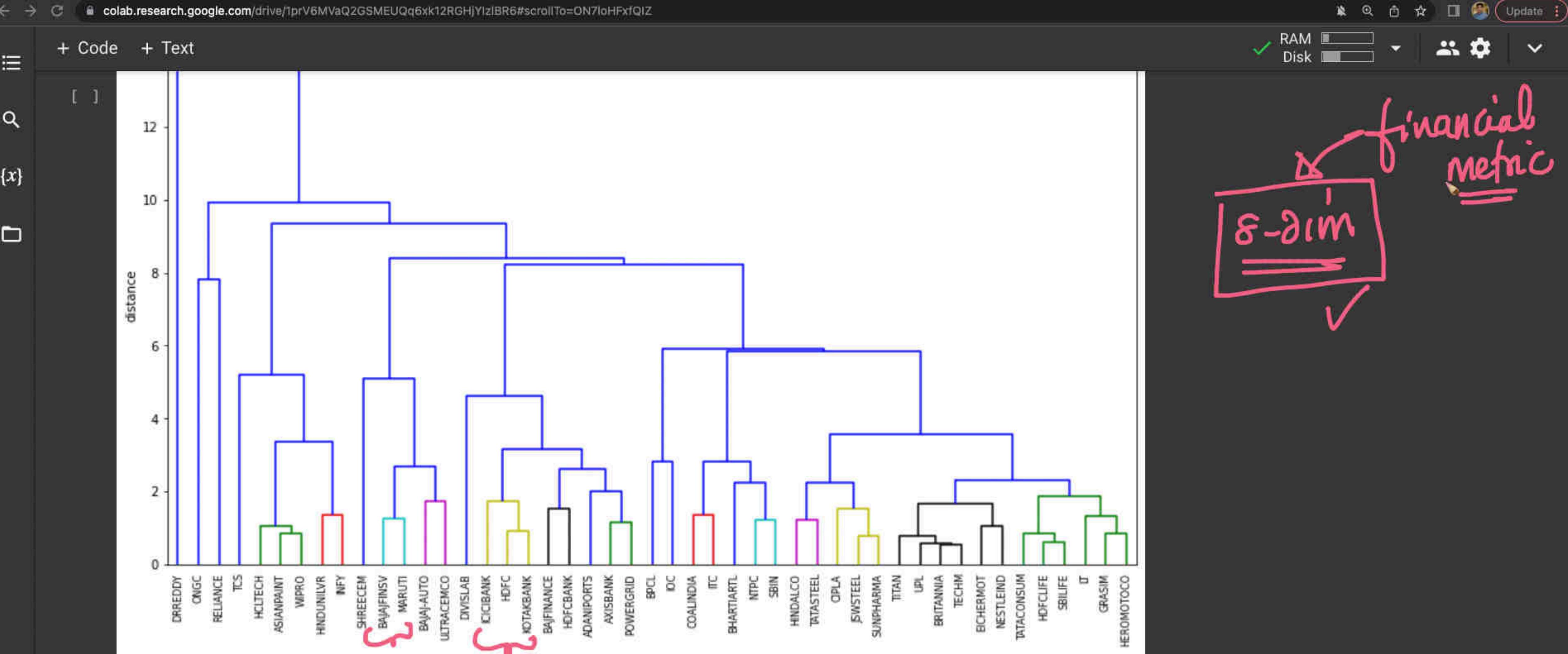
RAM Disk



```
[ ] # import hierarchical clustering libraries  
from sklearn.cluster import AgglomerativeClustering
```

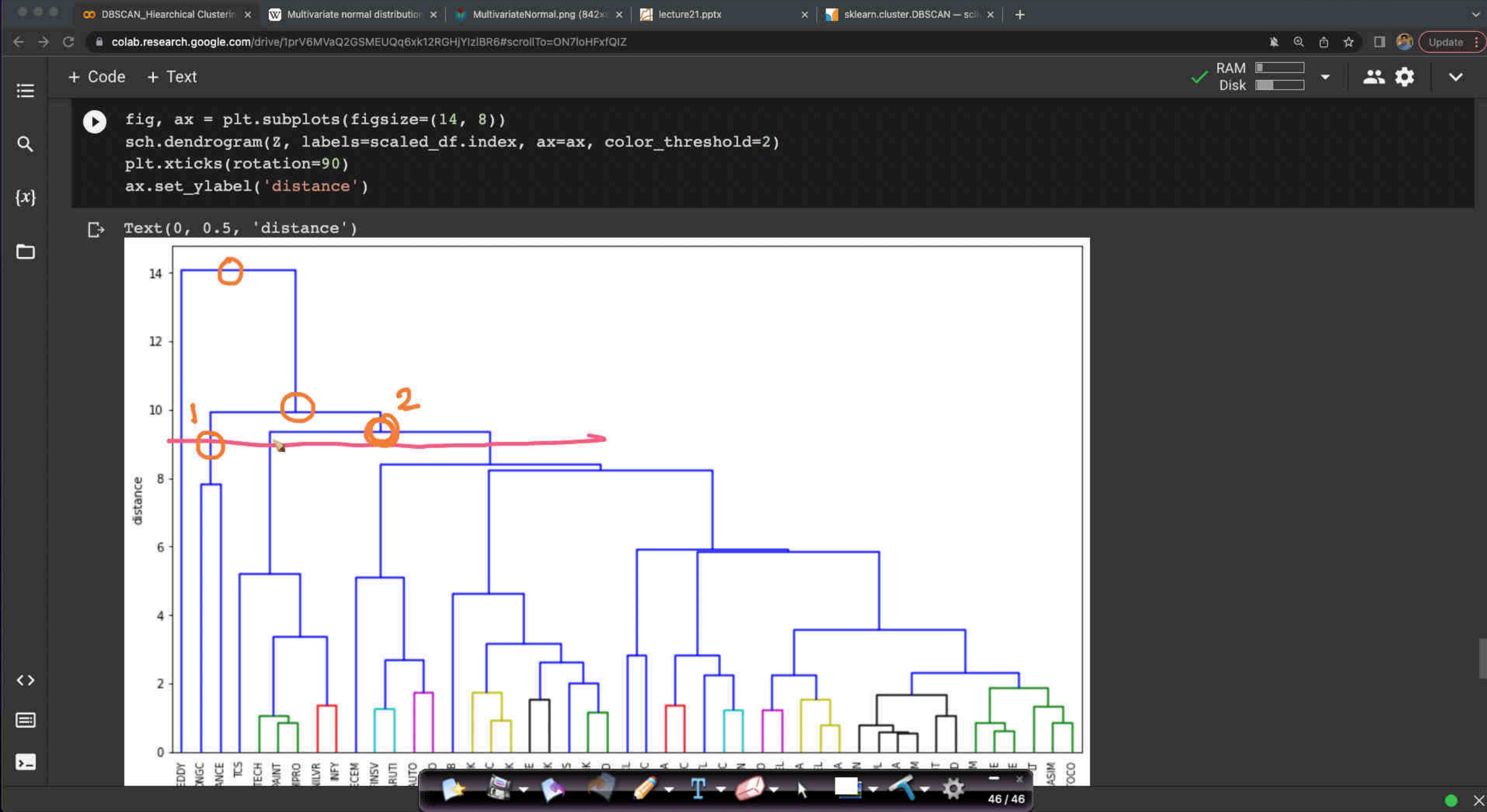
create clusters

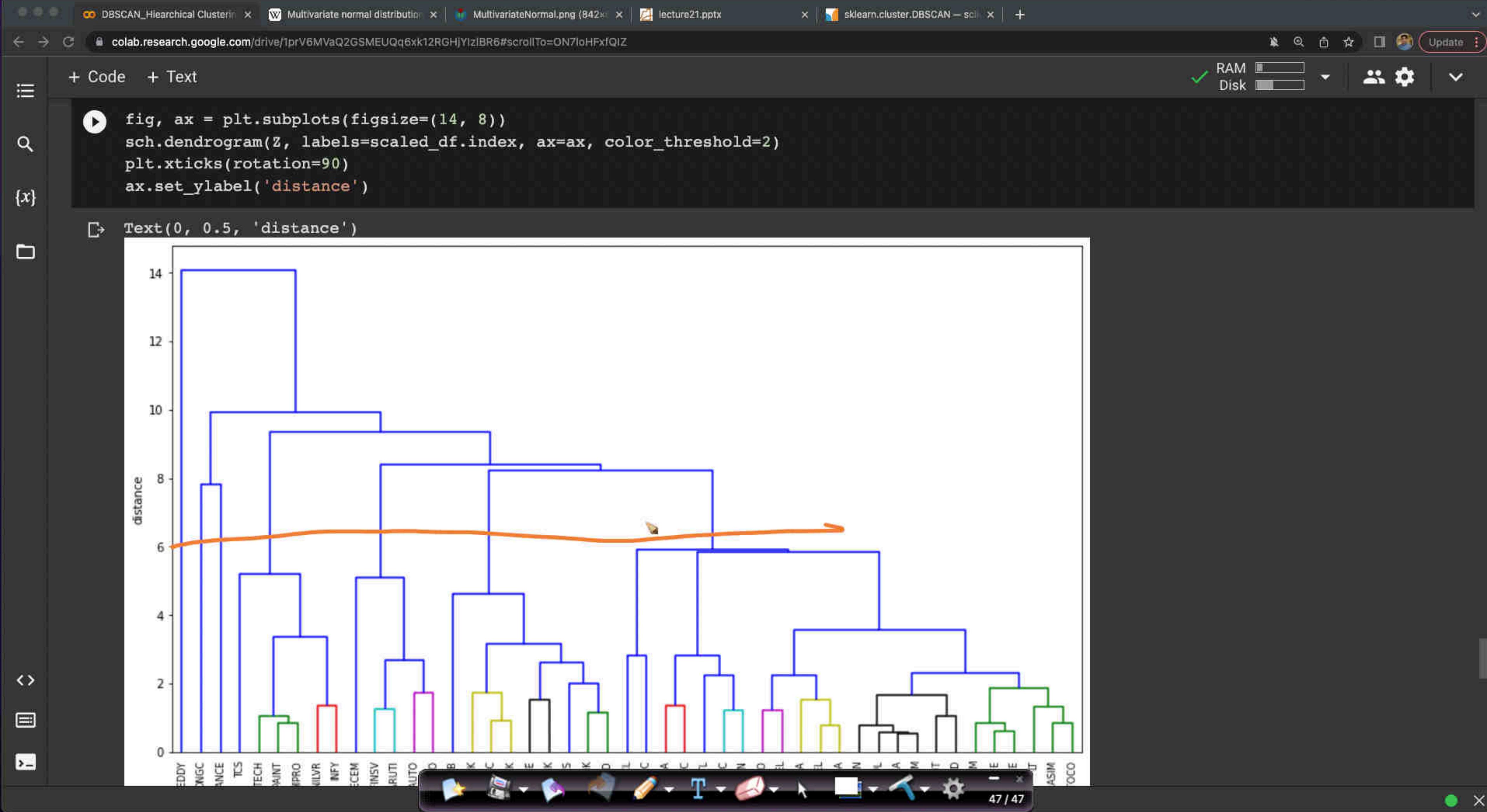


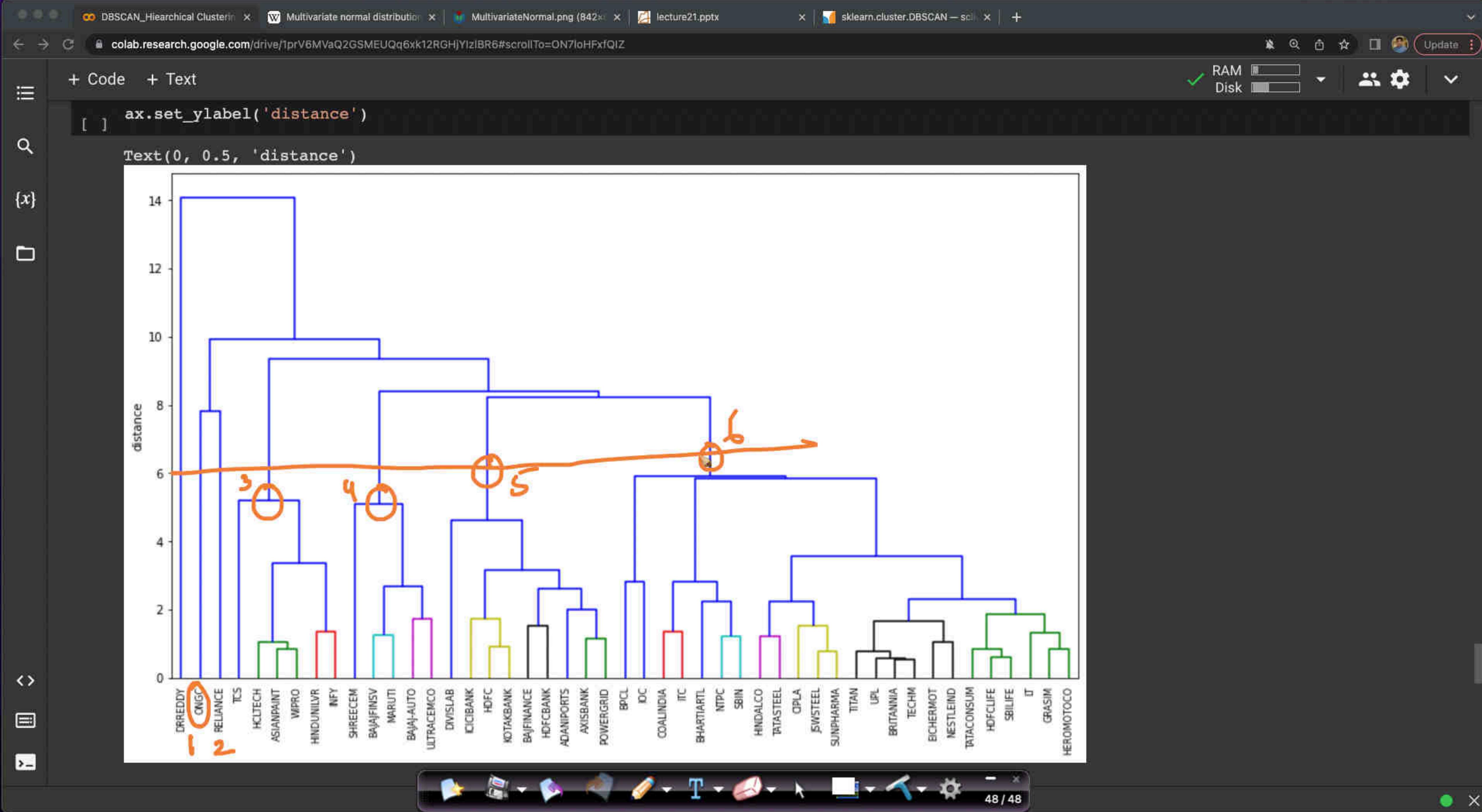


```
[ ] # import hierarchical clustering libraries  
from sklearn.cluster import AgglomerativeClustering
```

```
# create clusters
```

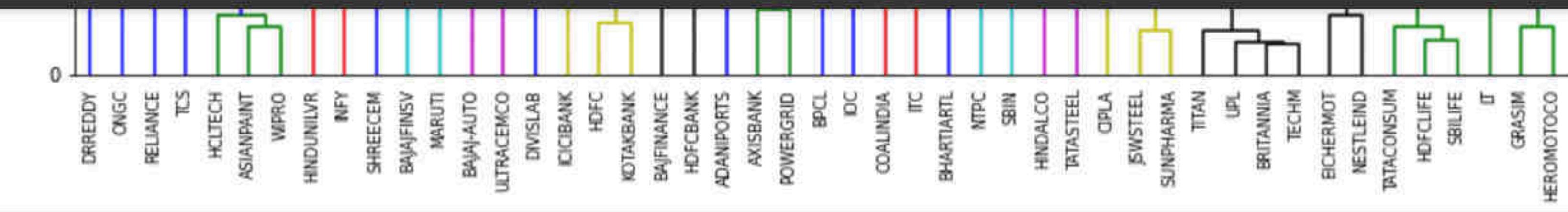






+ Code + Text

✓ RAM Disk



```
# import hierarchical clustering libraries
from sklearn.cluster import AgglomerativeClustering
# create clusters
hc_2020 = AgglomerativeClustering(n_clusters=5, affinity = 'euclidean', linkage = 'ward')
y_pred = hc_2020.fit_predict(scaled_df)
```

```
#Plot a line graph to see the characteristics of the clusters
scaled_df['label'] = pd.Series(y_pred, index=scaled_df.index)

clustered_df = scaled_df.groupby('label').mean()

labels = ['Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5']

plt.figure(figsize=(14,8))
plt.plot(clustered_df.T, label=labels)
plt.xticks(rotation=90)
plt.legend(labels)
```

<matplotlib.legend.Legend at 0x7f36d94d4490>



DBSCAN_Hierarchical Clustering Multivariate normal distribution MultivariateNormal.png (842x624) lecture21.pptx sklearn.cluster.DBSCAN — scikit-learn

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGhjYIzjBR6#scrollTo=O2pEoLE8gIGi

RAM Disk

+ Code + Text

DRIRED CN PELIAN HCLTE ASIANPA WP HNDUNIL SHREEC BAJAFIN MARI BAJAAU ULTRACEM DIVISL CICIBA HD KOTAKBA BAJIFINAN HDFCBA ADANIPOOR AXISBAA POWERGR BF COALING BHARTIA NT SE HINDAL TATASTE GP SWSTI SUNPHAR TITI BRITANN TEC EICHERM NESTLE TATACONS HDFCL SBIL GRAS HEROMOTO

```
# import hierarchical clustering libraries
from sklearn.cluster import AgglomerativeClustering

# create clusters
hc_2020 = AgglomerativeClustering(n_clusters=5, affinity = 'euclidean', linkage = 'ward')

y_pred = hc_2020.fit_predict(scaled_df)

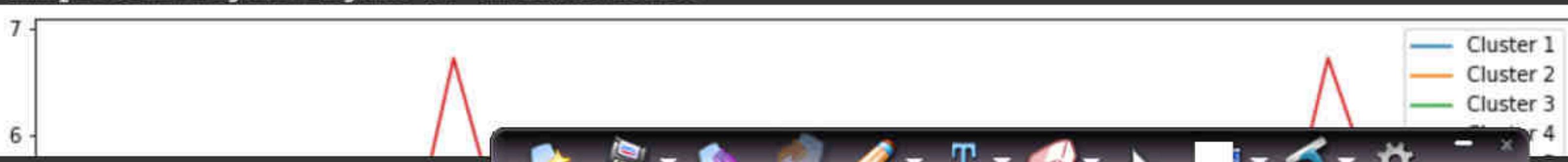
#Plot a line graph to see the characteristics of the clusters
scaled_df['label'] = pd.Series(y_pred, index=scaled_df.index)

clustered_df = scaled_df.groupby('label').mean()

labels = ['Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4', 'Cluster 5']

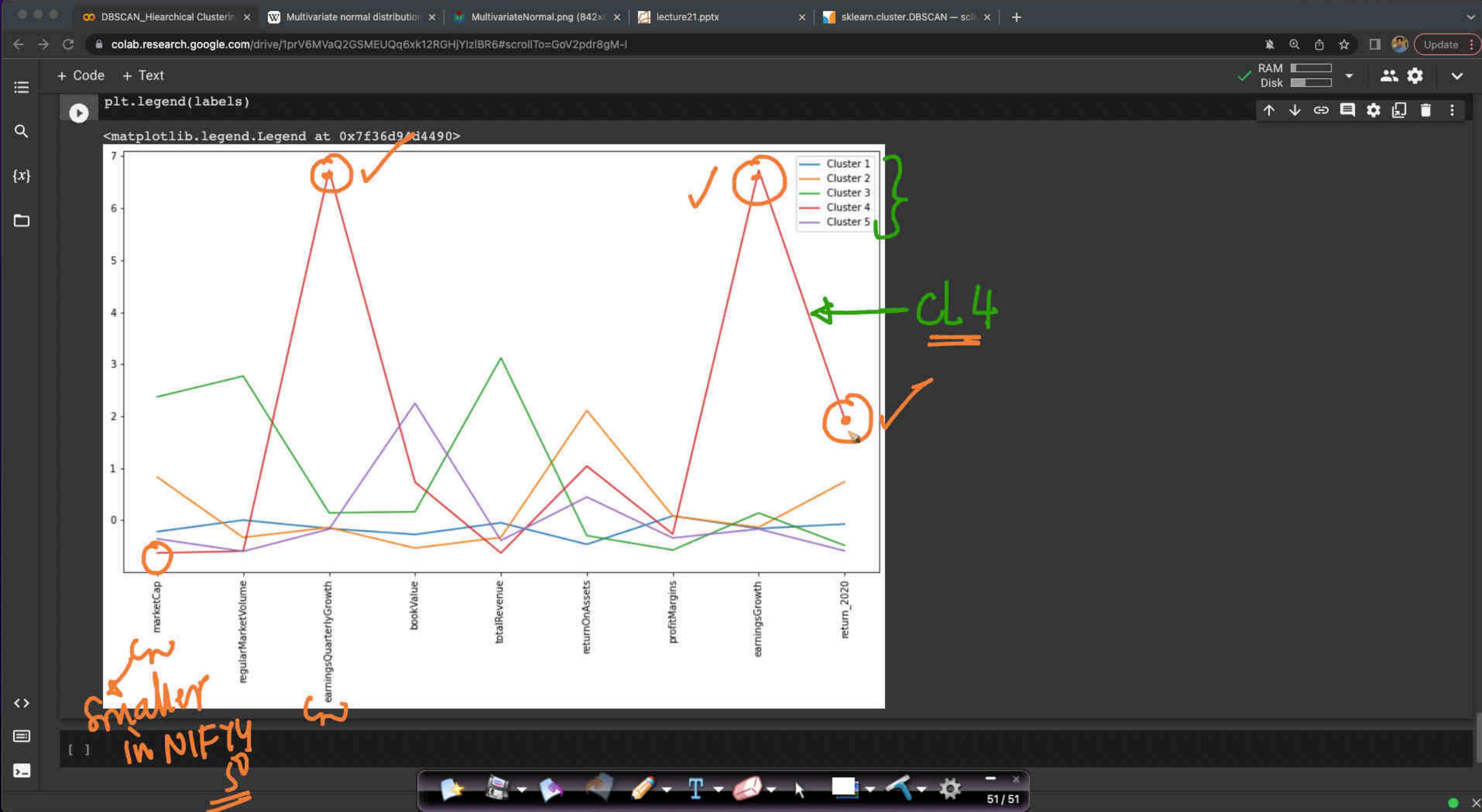
plt.figure(figsize=(14,8))
plt.plot(clustered_df.T, label=labels)
plt.xticks(rotation=90)
plt.legend(labels)
```

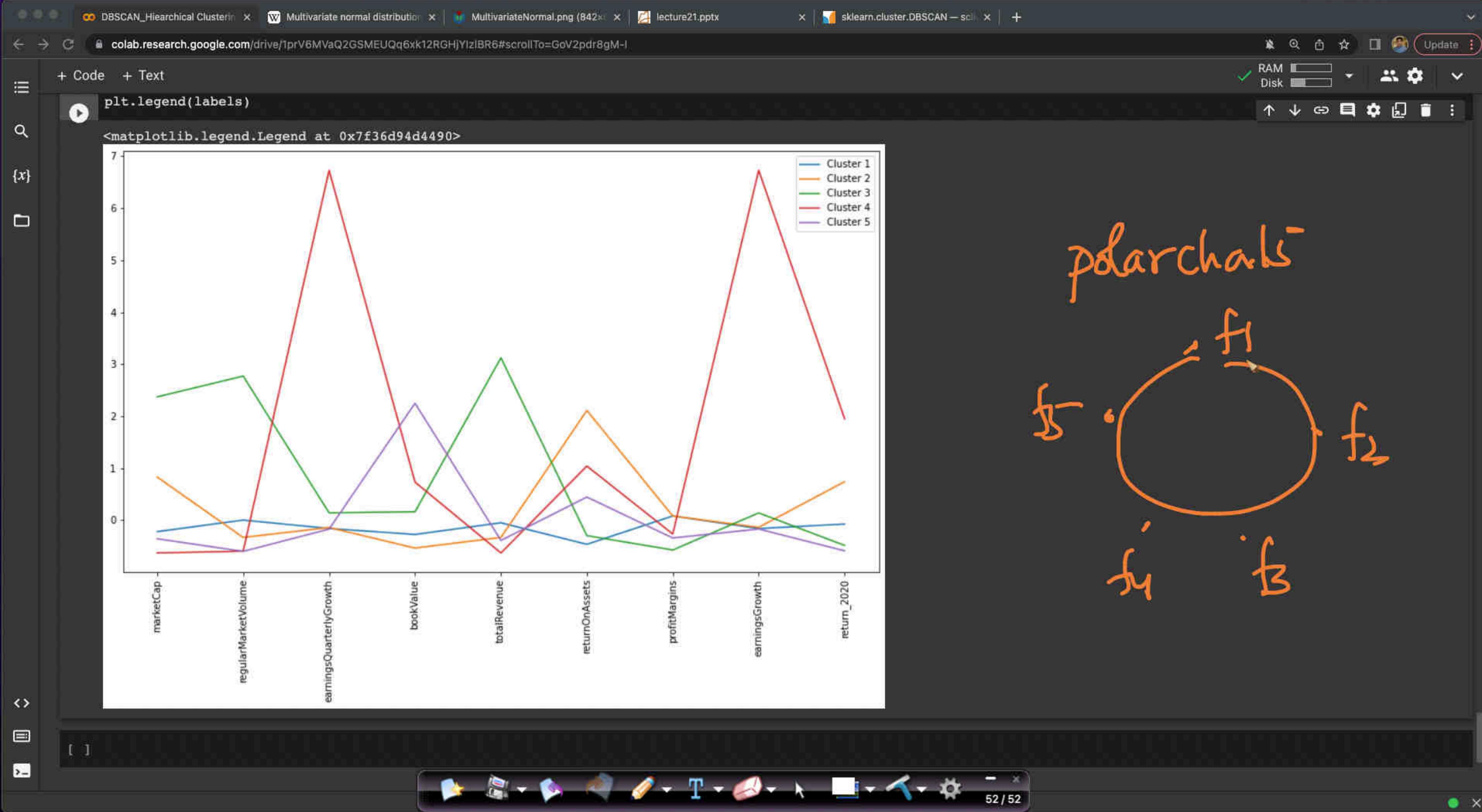
<matplotlib.legend.Legend at 0x7f36d94d4490>

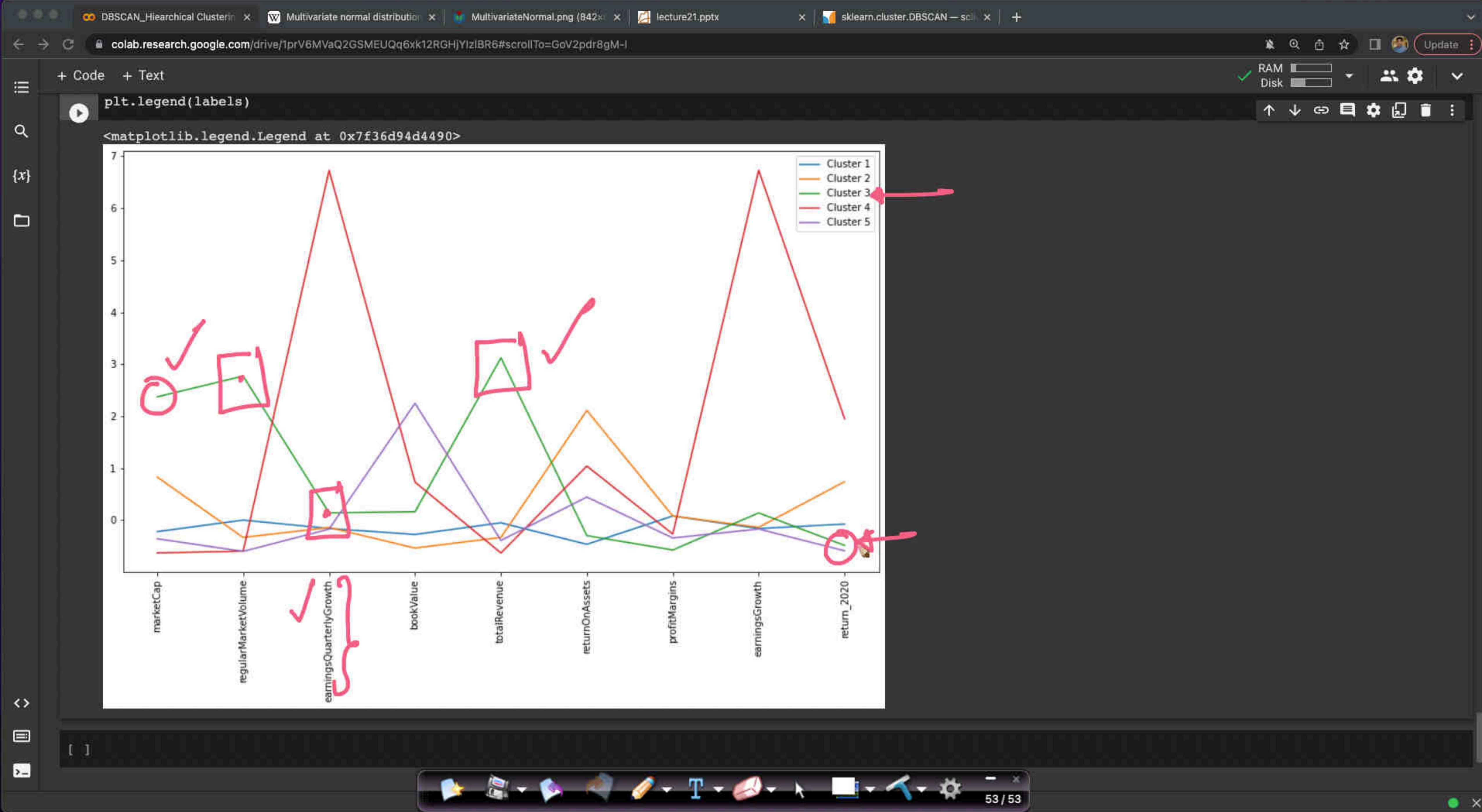


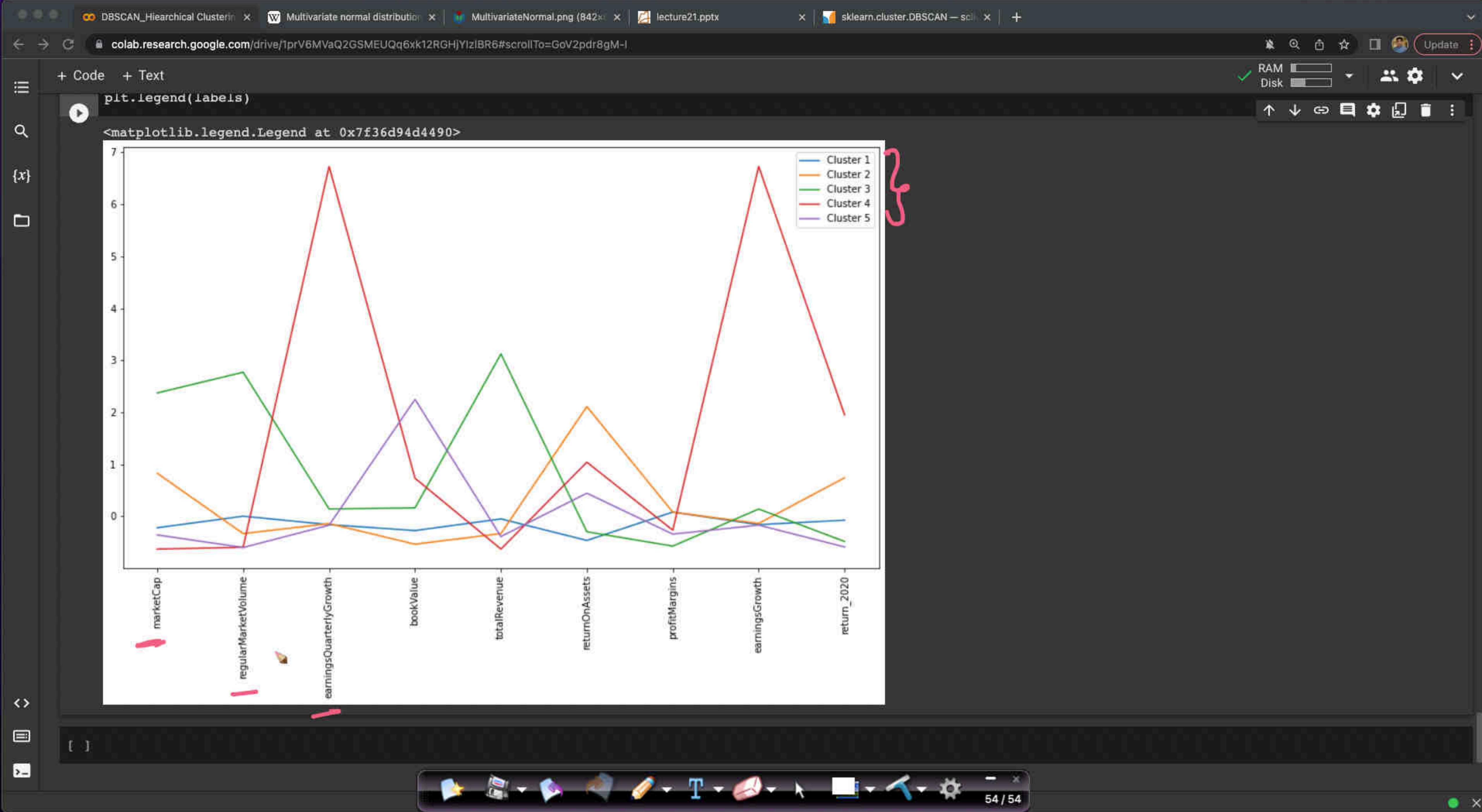
each cluster

mean of each feature



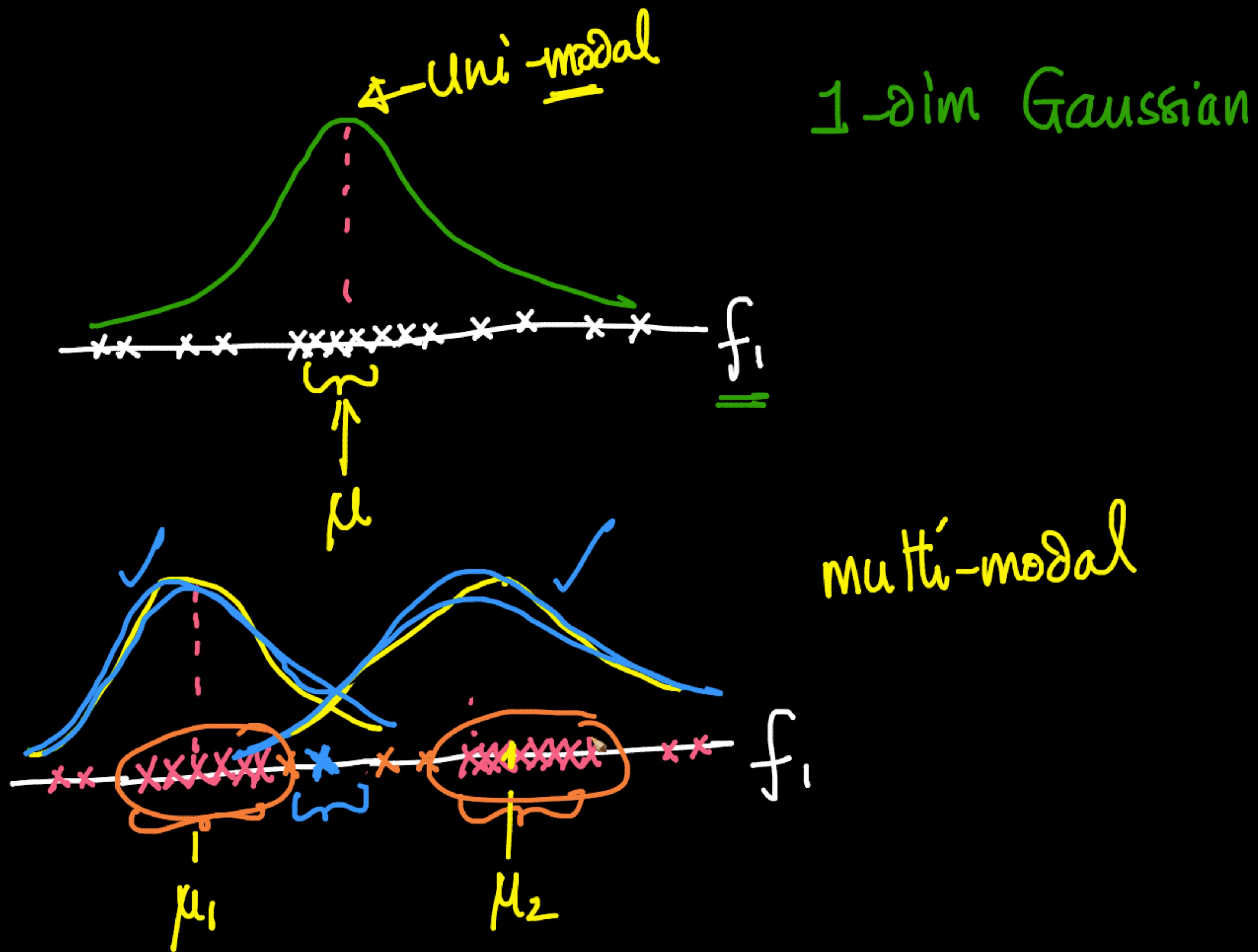


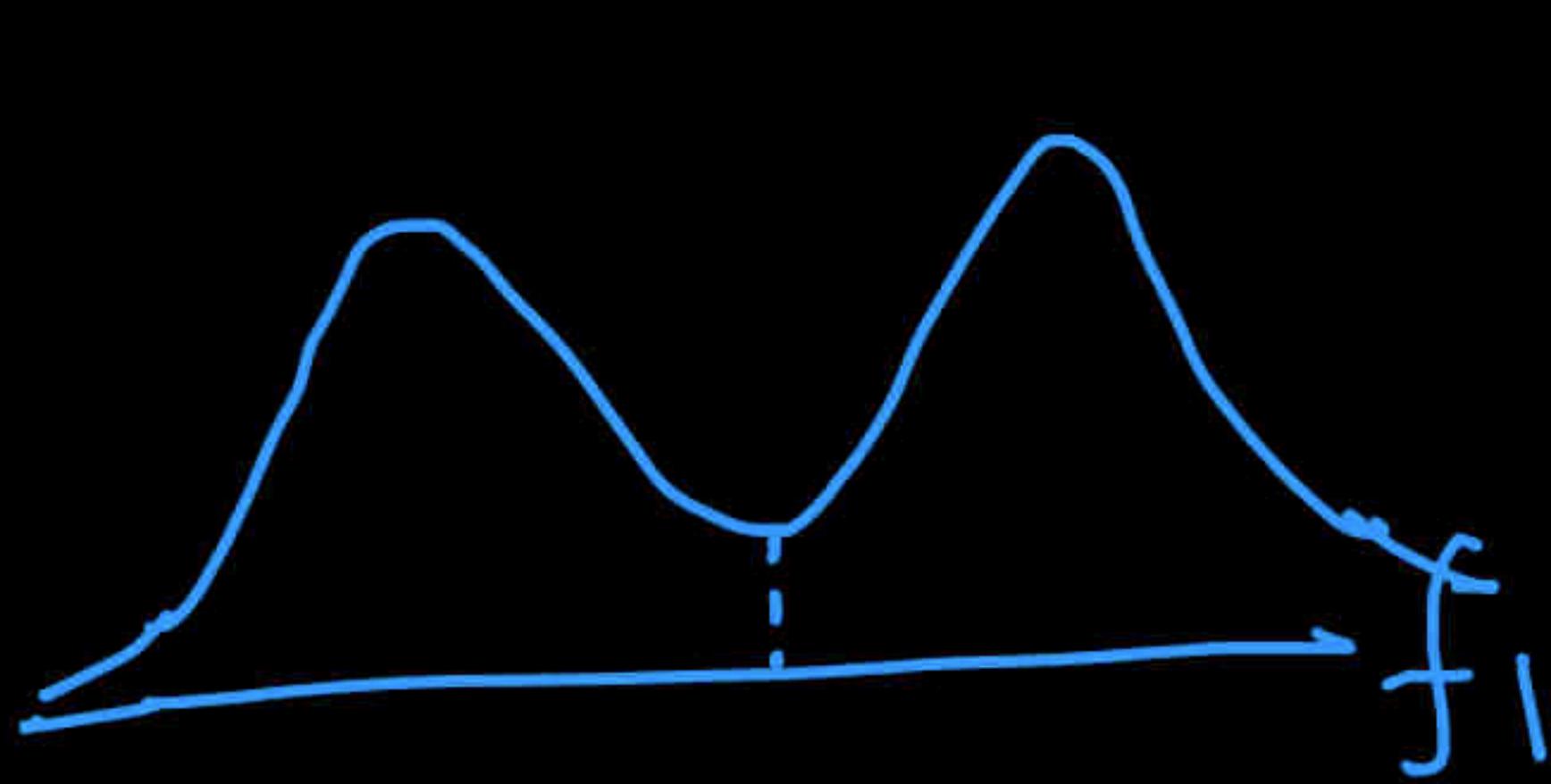




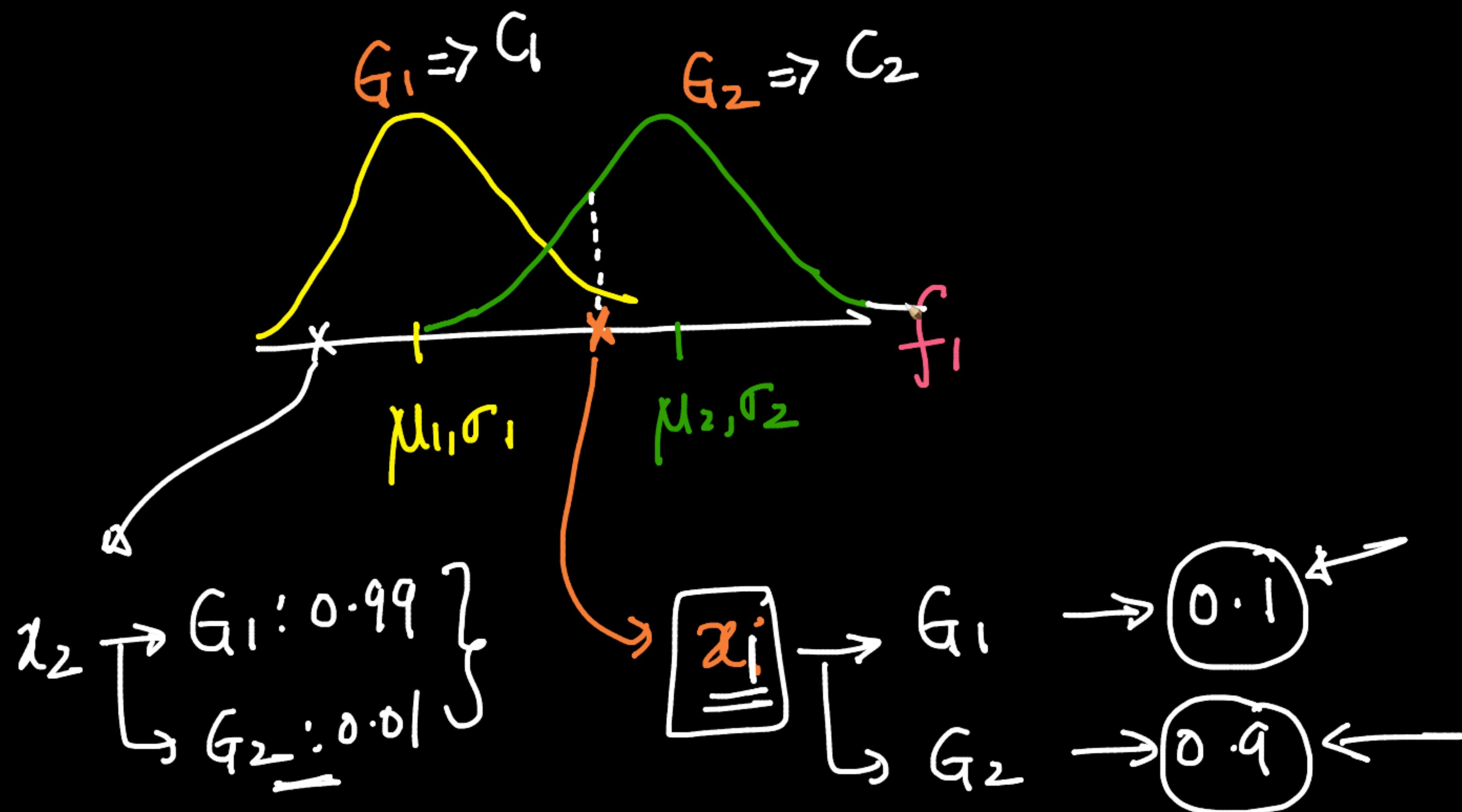


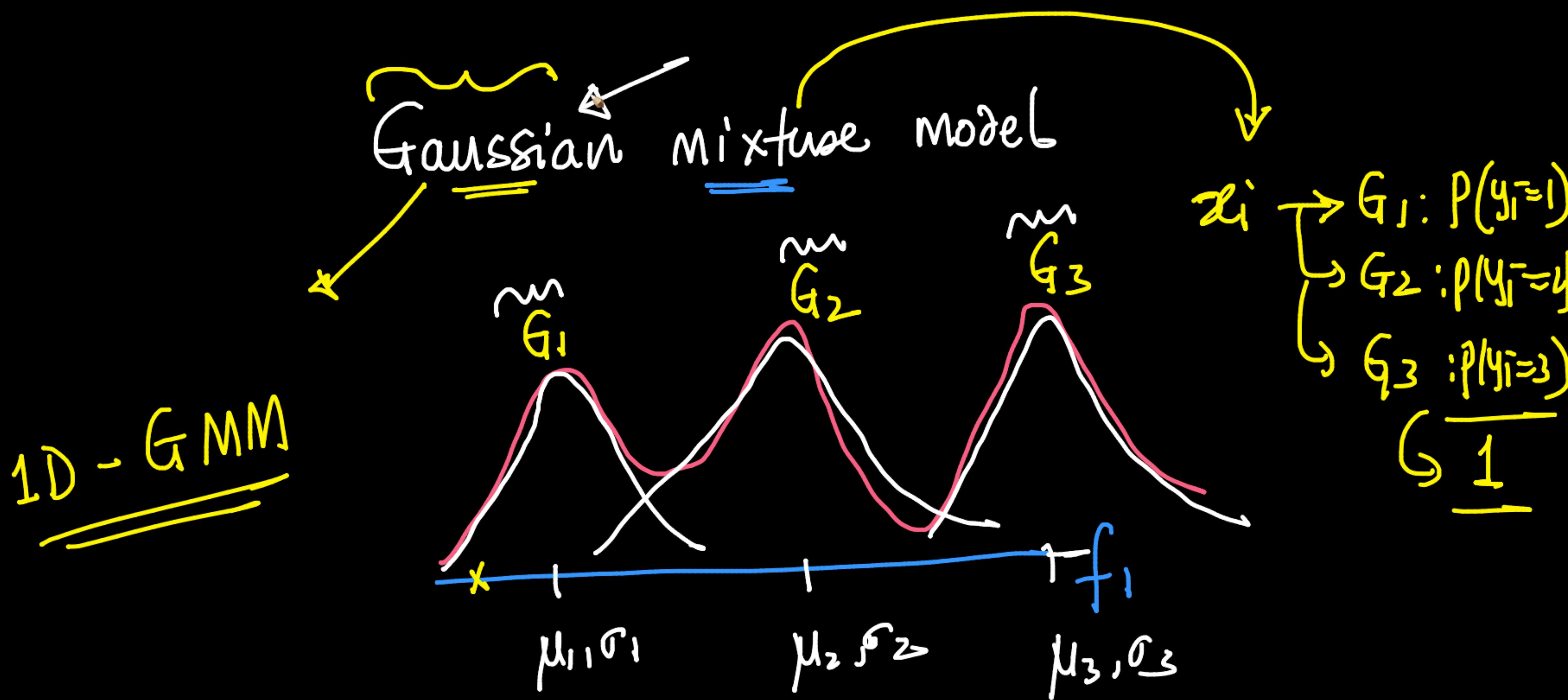






Mixture of Gaussians





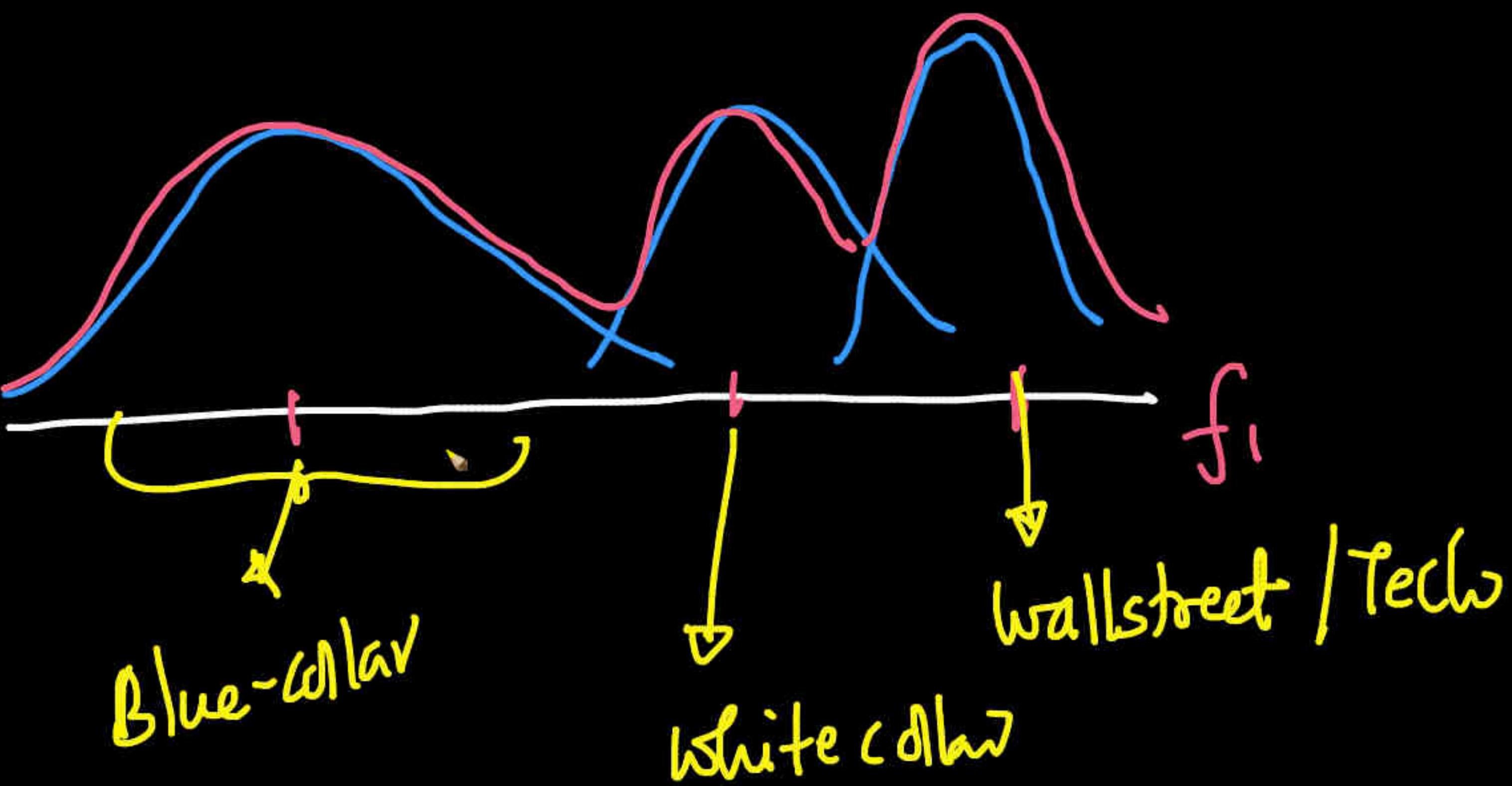


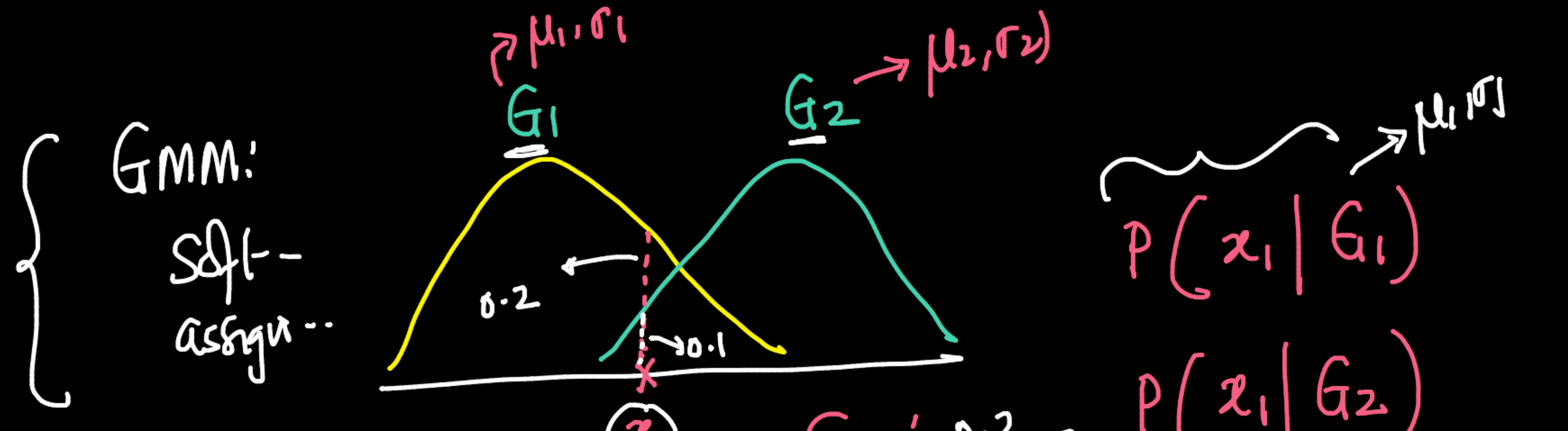
log-normal Mixture-model → Combination of
various dist

Theoretically

e-commerce



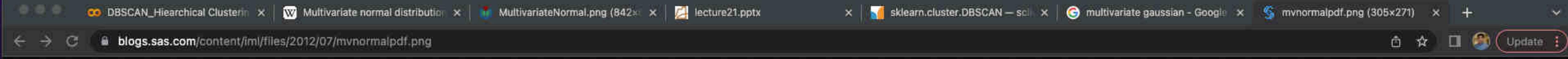


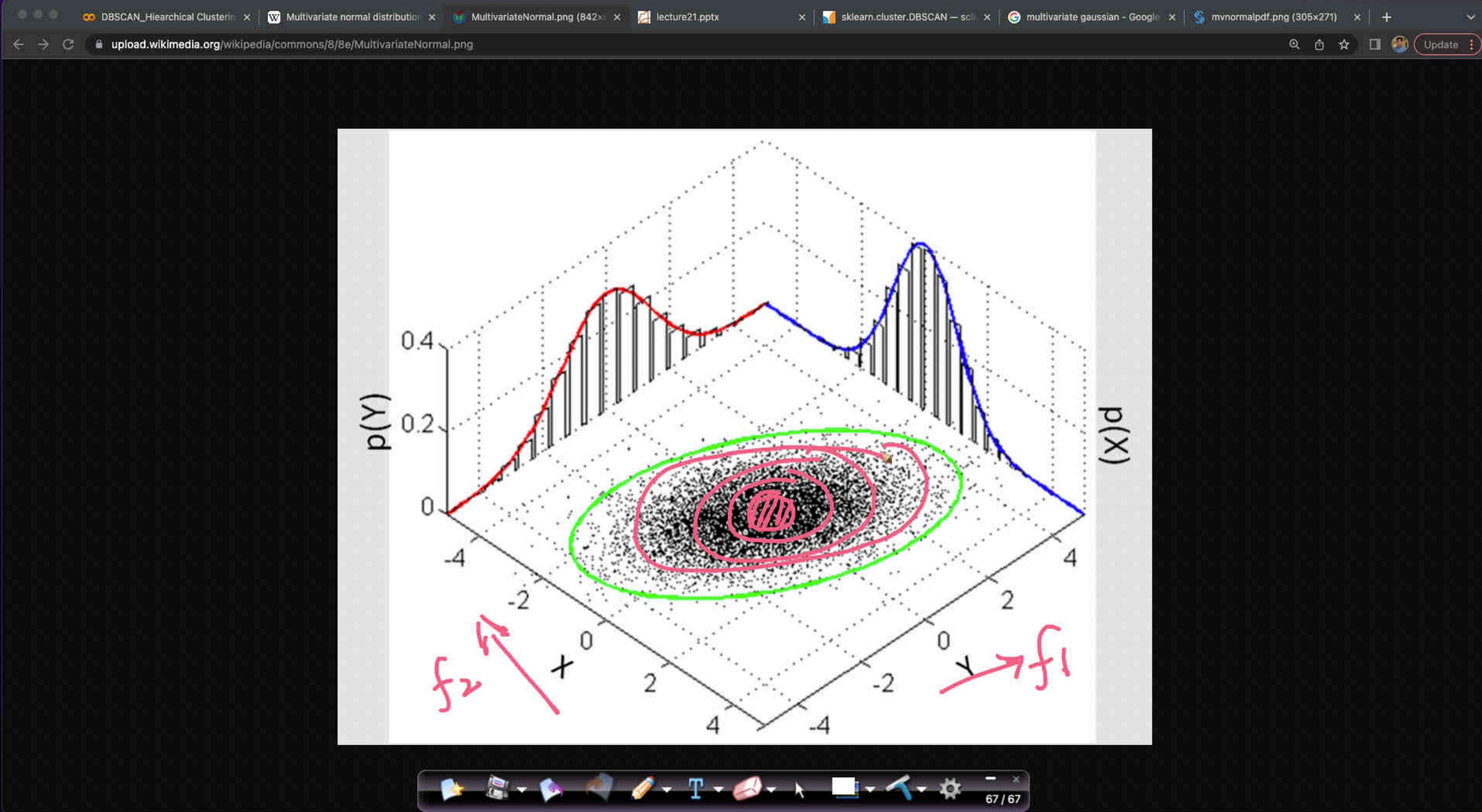


K-means

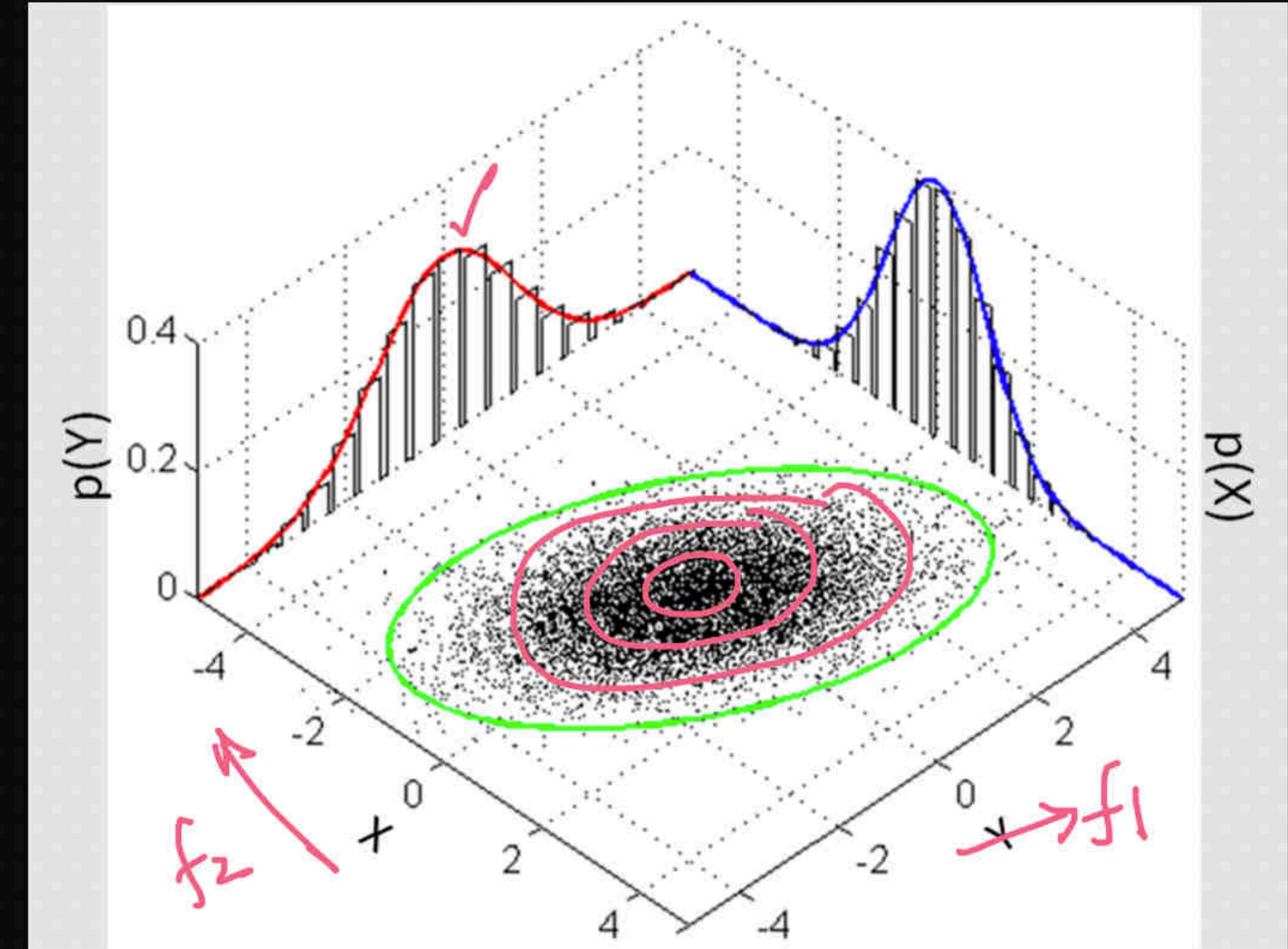
$x_i \rightarrow \begin{cases} G_1 \\ G_2 \end{cases}$

hard assignment

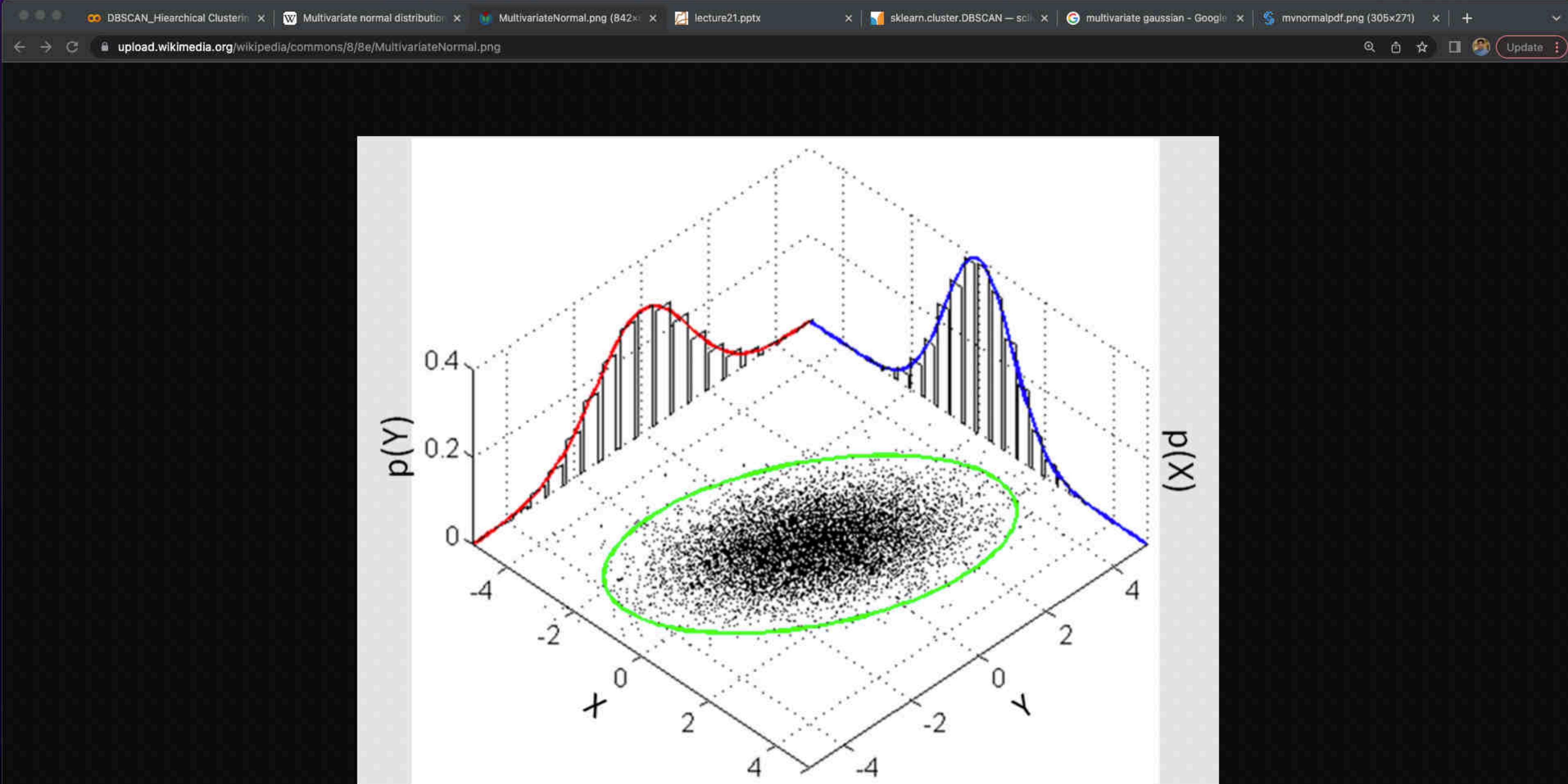




d -dim Gaussian
↓
hill in
 $d+1$ -dim space



2D ~ Gaussian
↓
2D-hill



1D:

$$N(\underline{\mu}, \underline{\Sigma})$$

d-dim:

$$N_d (\underline{\mu}, \underline{\Sigma})$$

d-dim

2D:

$$f_1, f_2$$

D:

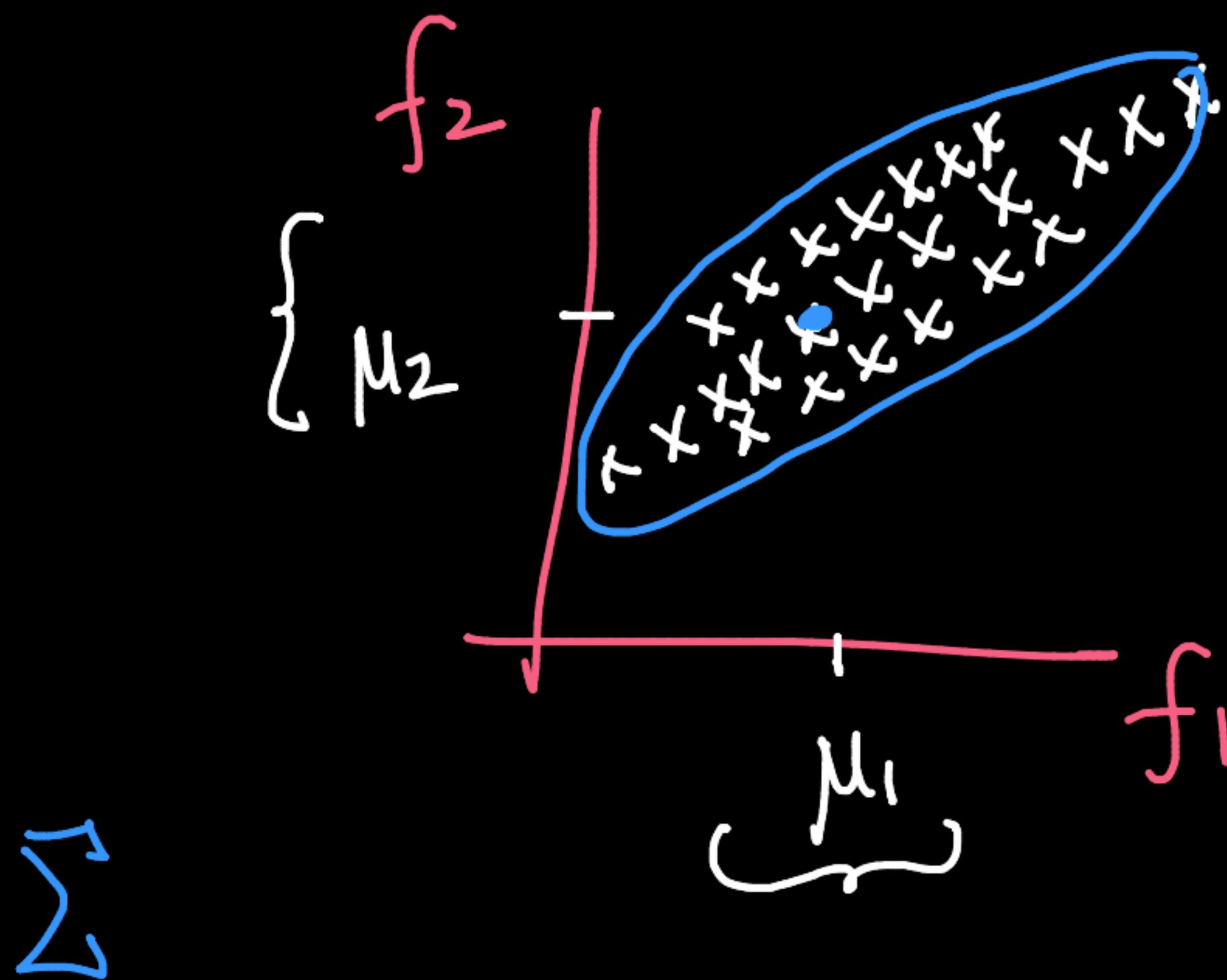
$$\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$$

$$N_2 (\underline{\mu}, \underline{\Sigma}_{2 \times 2})$$

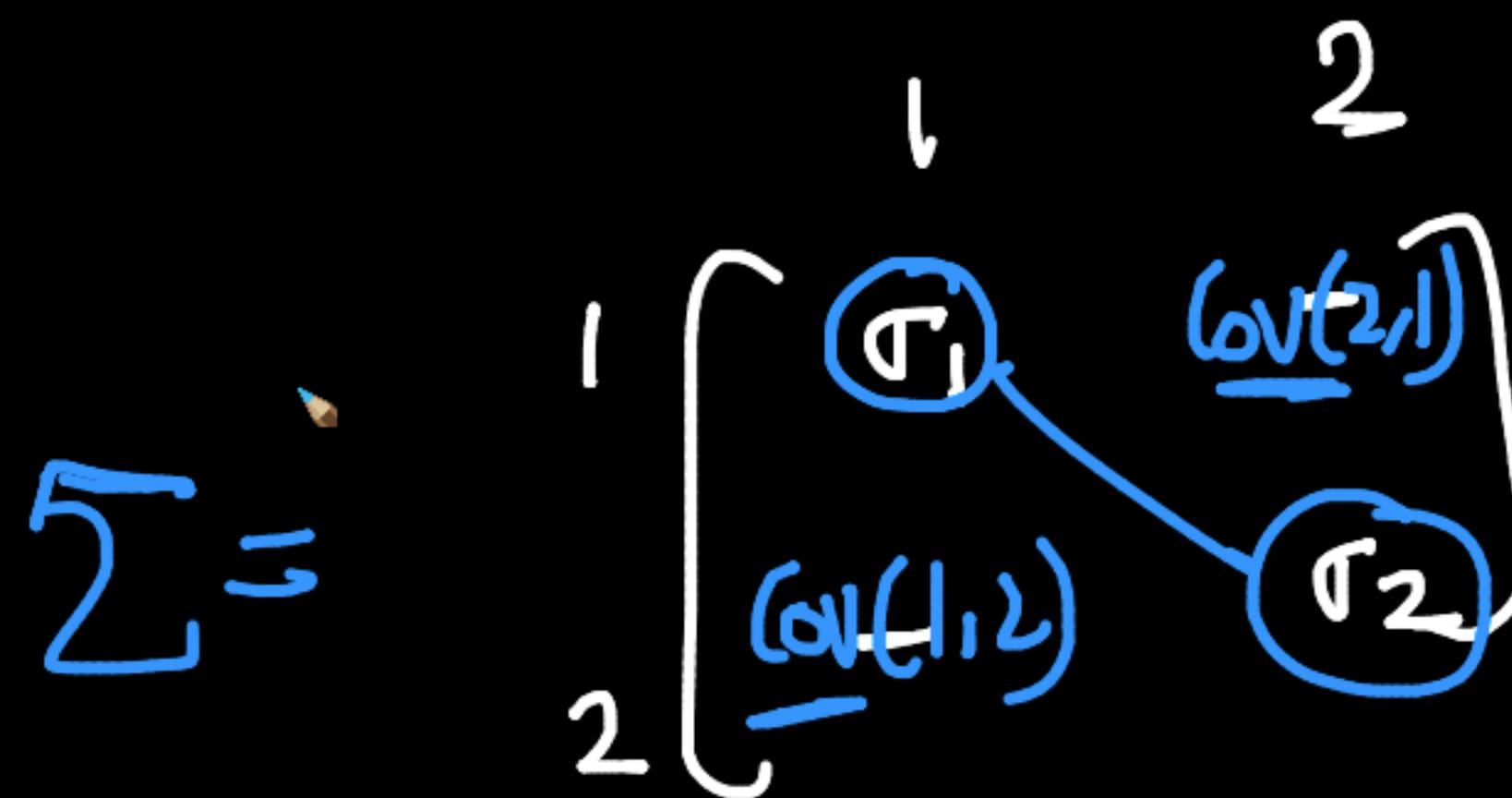
$$[\underline{\mu_1}, \underline{\mu_2}]$$

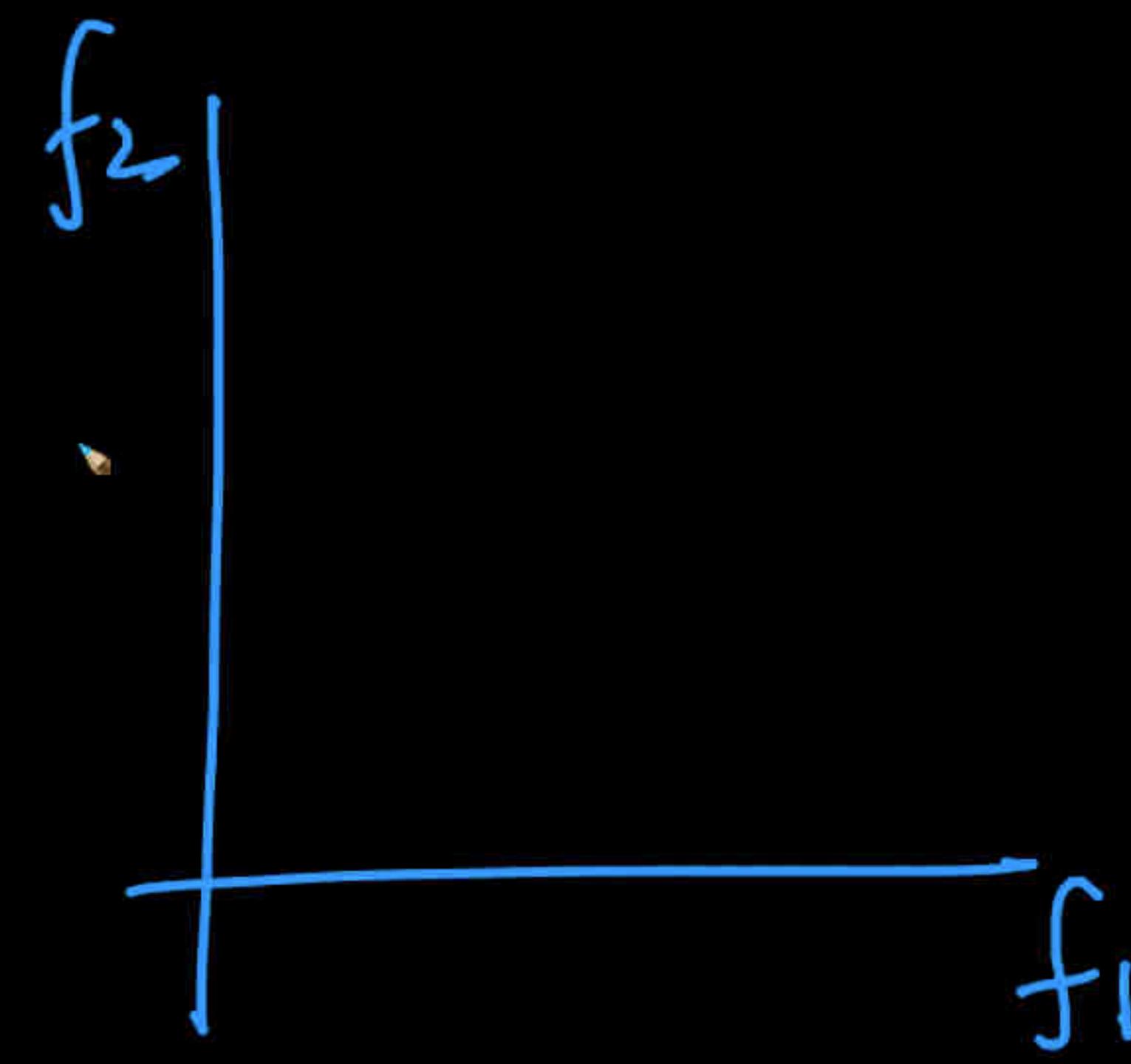
COV-Matrix of \mathcal{D}

$$\text{if } \underline{j} < \underline{i}$$



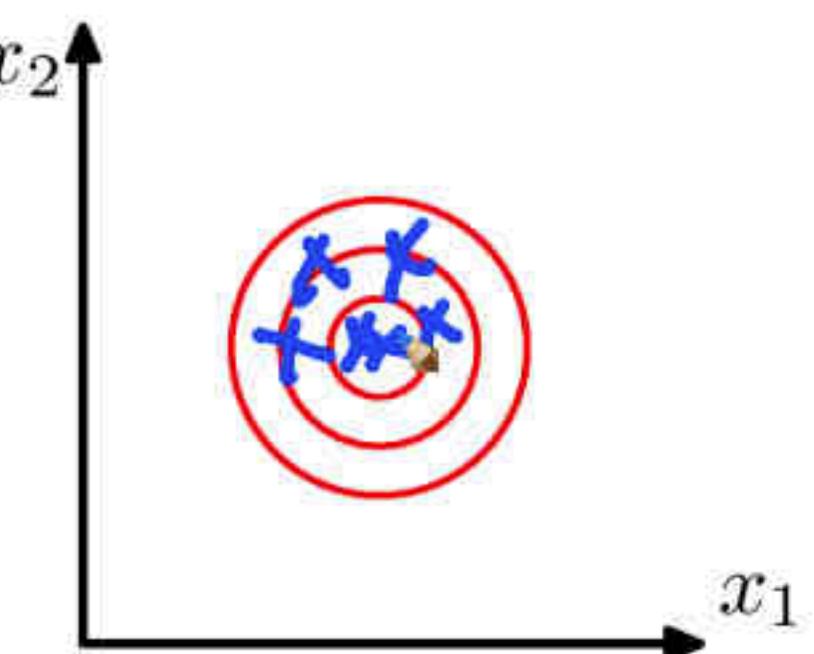
$$\text{if } N_2 \left(\begin{matrix} \tilde{\mu} \\ \Gamma \end{matrix} \right) \downarrow \rightarrow [\sigma_1, \sigma_2] \\ (\mu_1, \mu_2)$$





Multivariate Gaussians

$$P(X=\mathbf{x}_j) = \frac{1}{(2\pi)^{m/2} \|\Sigma\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_j - \boldsymbol{\mu})\right]$$



$\Sigma \propto$ identity matrix

DBSCAN_Hierarchical Clusterin x Multivariate normal distribution x MultivariateNormal.png (842x630) x lecture21.pptx x sklearn.cluster.DBSCAN — sc x multivariate gaussian - Google x mynormalpdf.png (305x271) x + Not Secure | people.csail.mit.edu/dsontag/courses/ml12/slides/lecture21.pdf Update

lecture21.pptx 7 / 38 - 100% + ☰

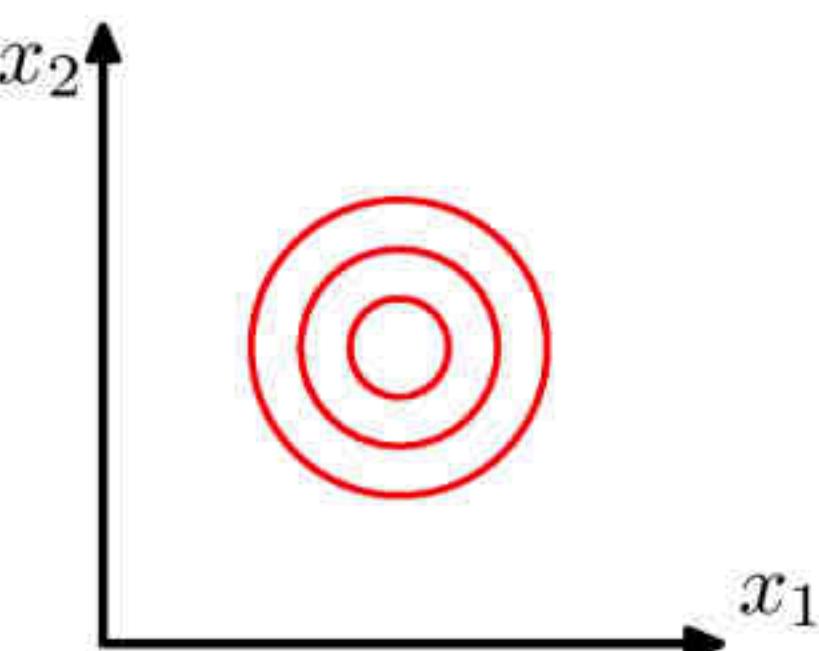
Multivariate Gaussians

$$P(X=\mathbf{x}_j) = \frac{1}{(2\pi)^{m/2} \|\Sigma\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}_j - \boldsymbol{\mu})\right]$$

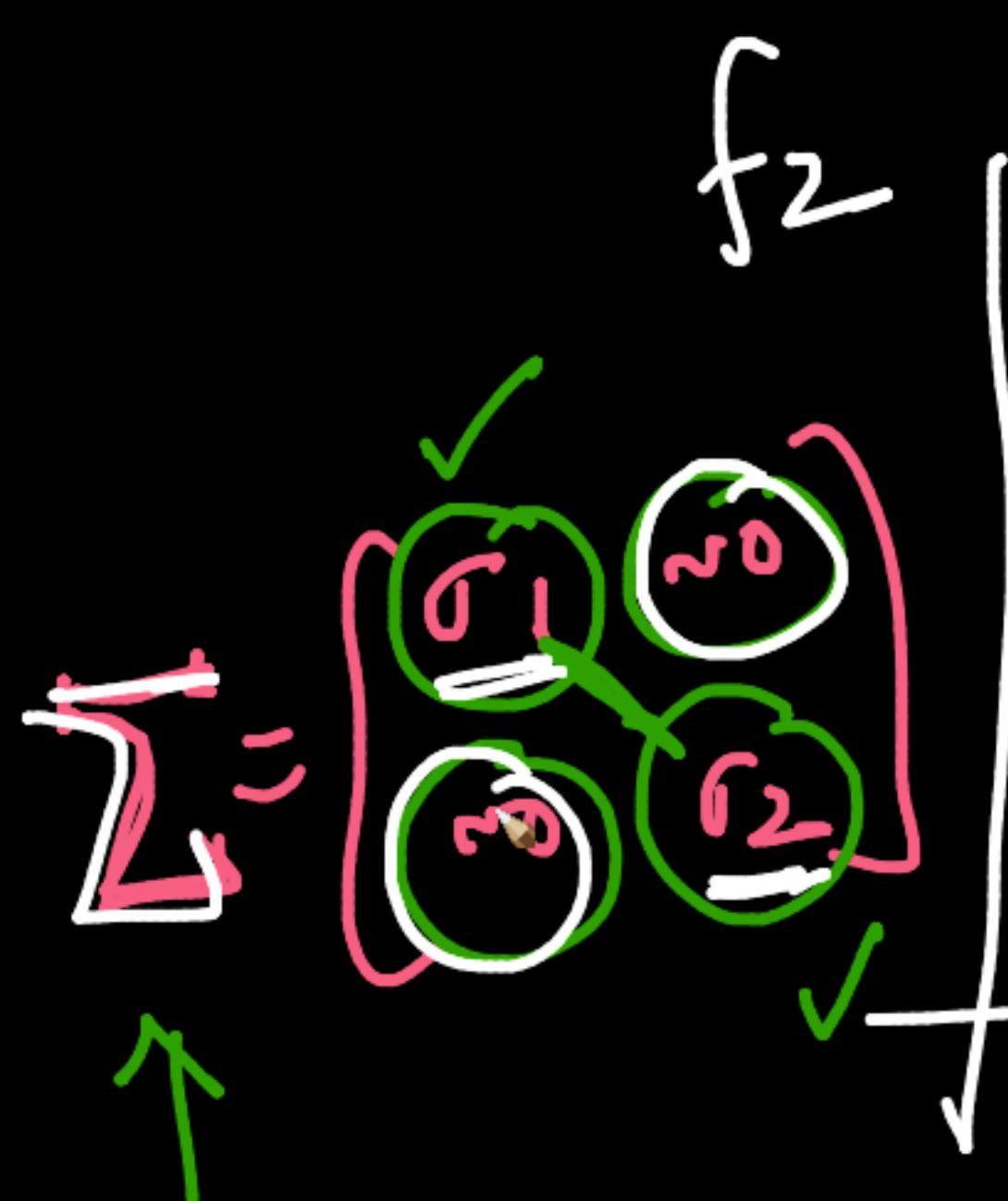
$\Sigma \propto$ identity matrix

Multivariate Gaussians

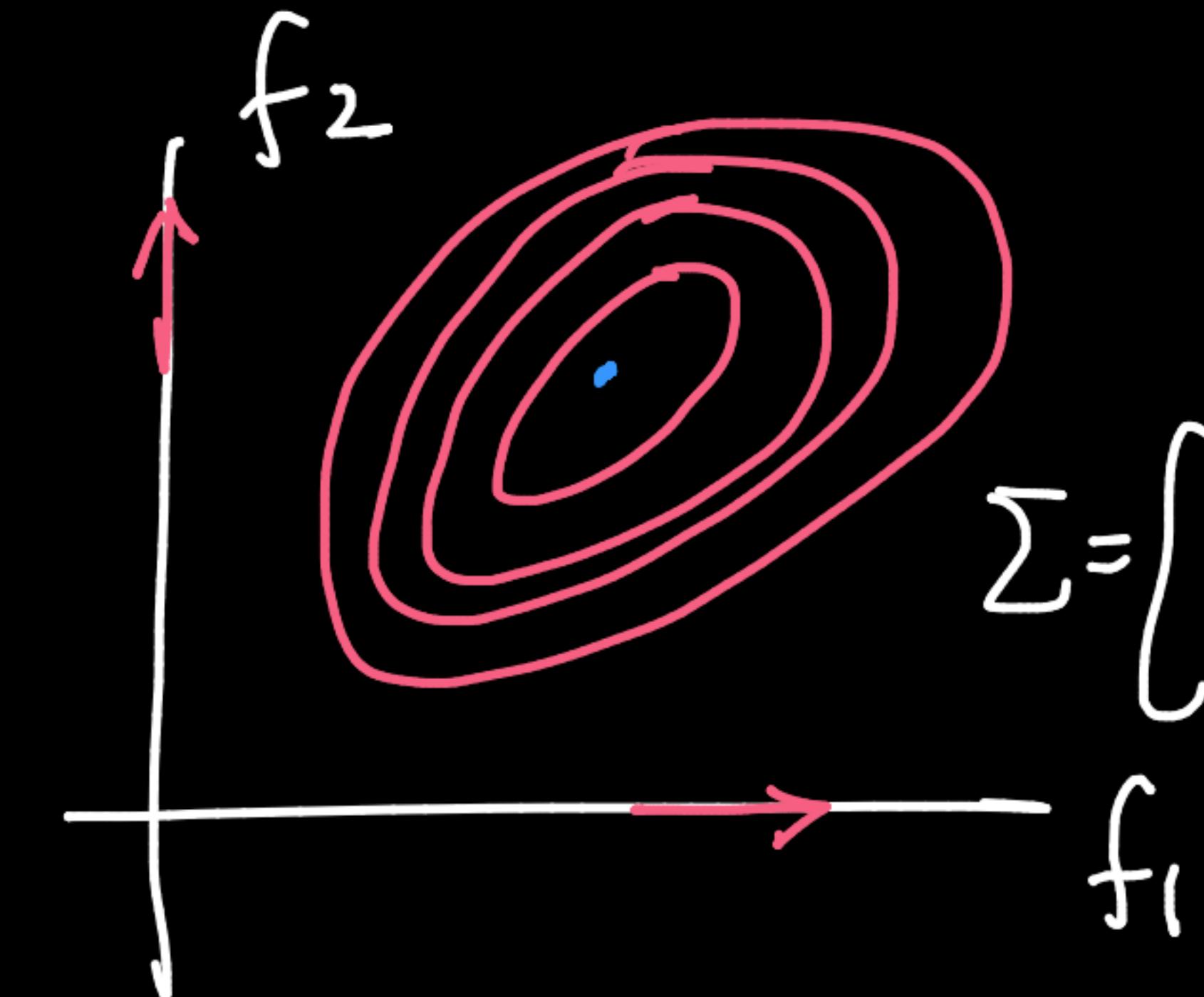
$$P(X=\mathbf{x}_j) = \frac{1}{(2\pi)^{m/2} \|\Sigma\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}_j - \boldsymbol{\mu})\right]$$



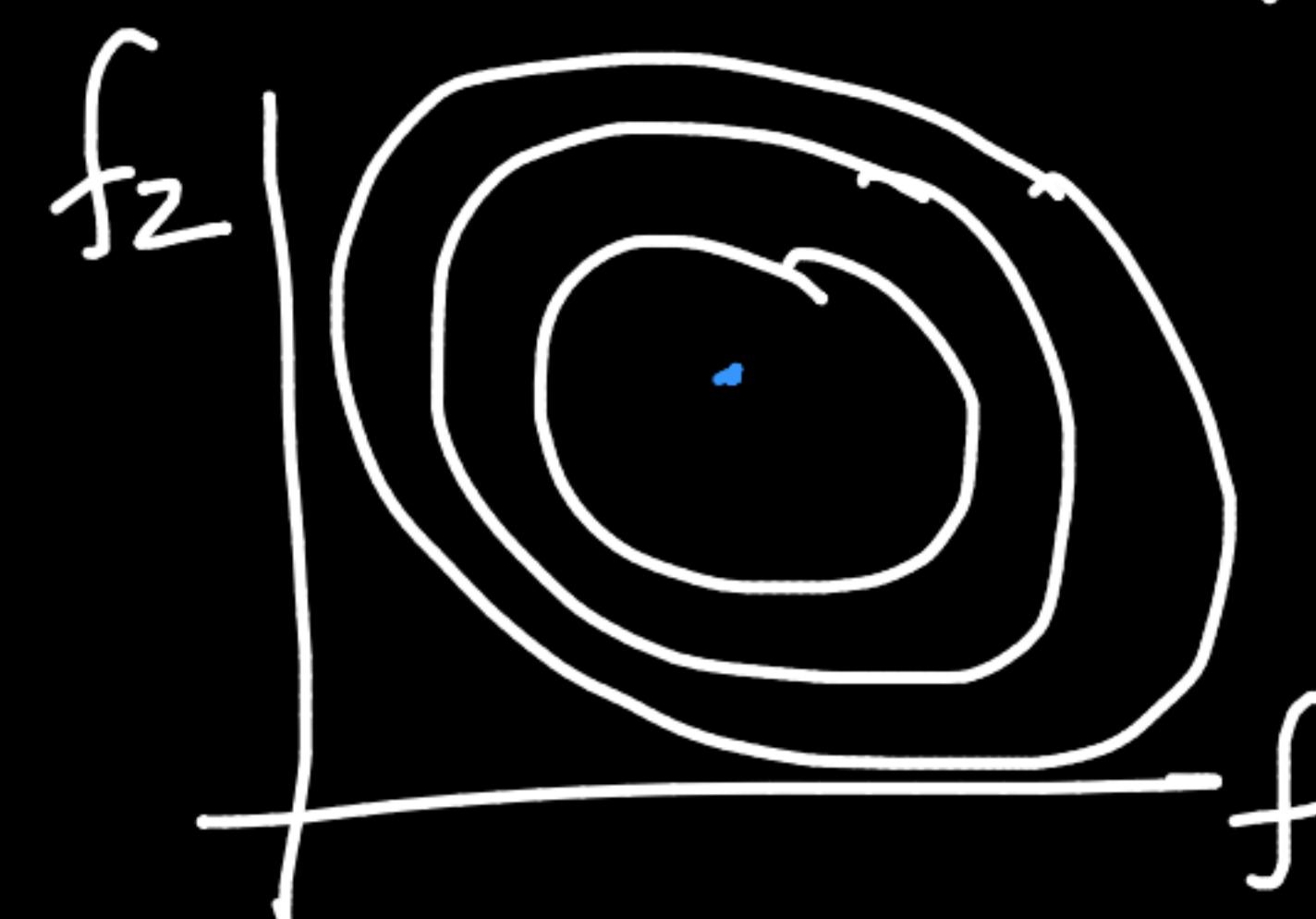
$\Sigma \propto$ identity matrix



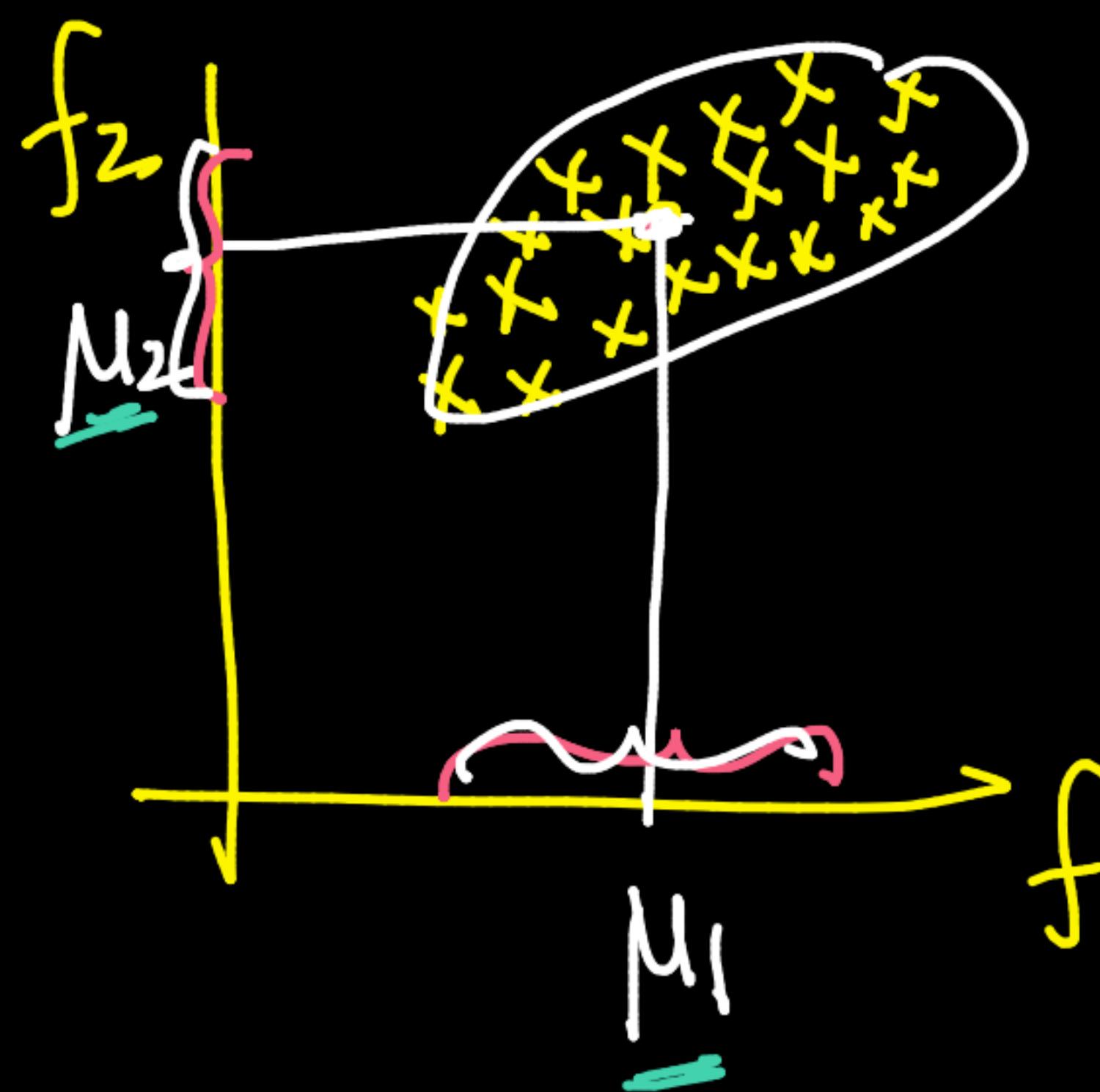
$$\Sigma = \begin{bmatrix} \sigma_1 & \text{tve} \\ \text{tve} & \sigma_2 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} \sigma_1 & \text{tve} \\ \text{tve} & \sigma_2 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} \sigma_1 & \text{tve} \\ \text{tve} & \sigma_2 \end{bmatrix}$$

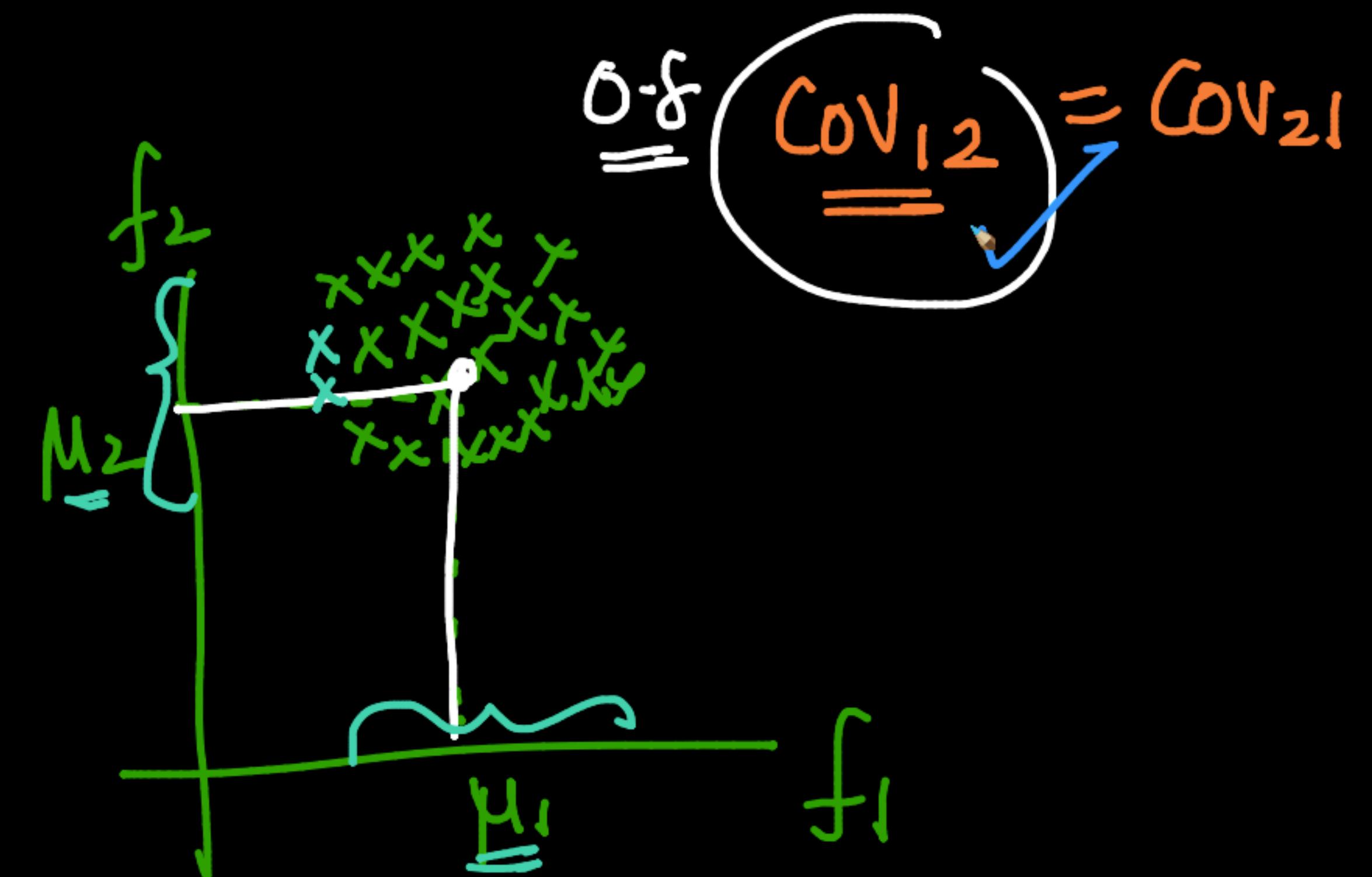


$$\checkmark \mu = [\underline{\mu}_1, \underline{\mu}_2]$$

$$\checkmark \Sigma = \begin{bmatrix} \sigma_1 & \text{COV}_{12} \\ \text{COV}_{21} & \sigma_2 \end{bmatrix}: \text{Symm-Matrix}$$

$\checkmark [\underline{\mu}_1, \underline{\mu}_2] = \mu_{\text{2dim}}$

$\checkmark \underline{\sigma}_1, \underline{\sigma}_2$



$$\checkmark \underline{\sigma}_1, \underline{\sigma}_2 \quad \text{0-8} \quad \text{COV}_{12} = \text{COV}_{21}$$

$$\Sigma = \begin{bmatrix} \sigma_1 & \text{COV}_{12} \\ \text{COV}_{21} & \sigma_2 \end{bmatrix}$$

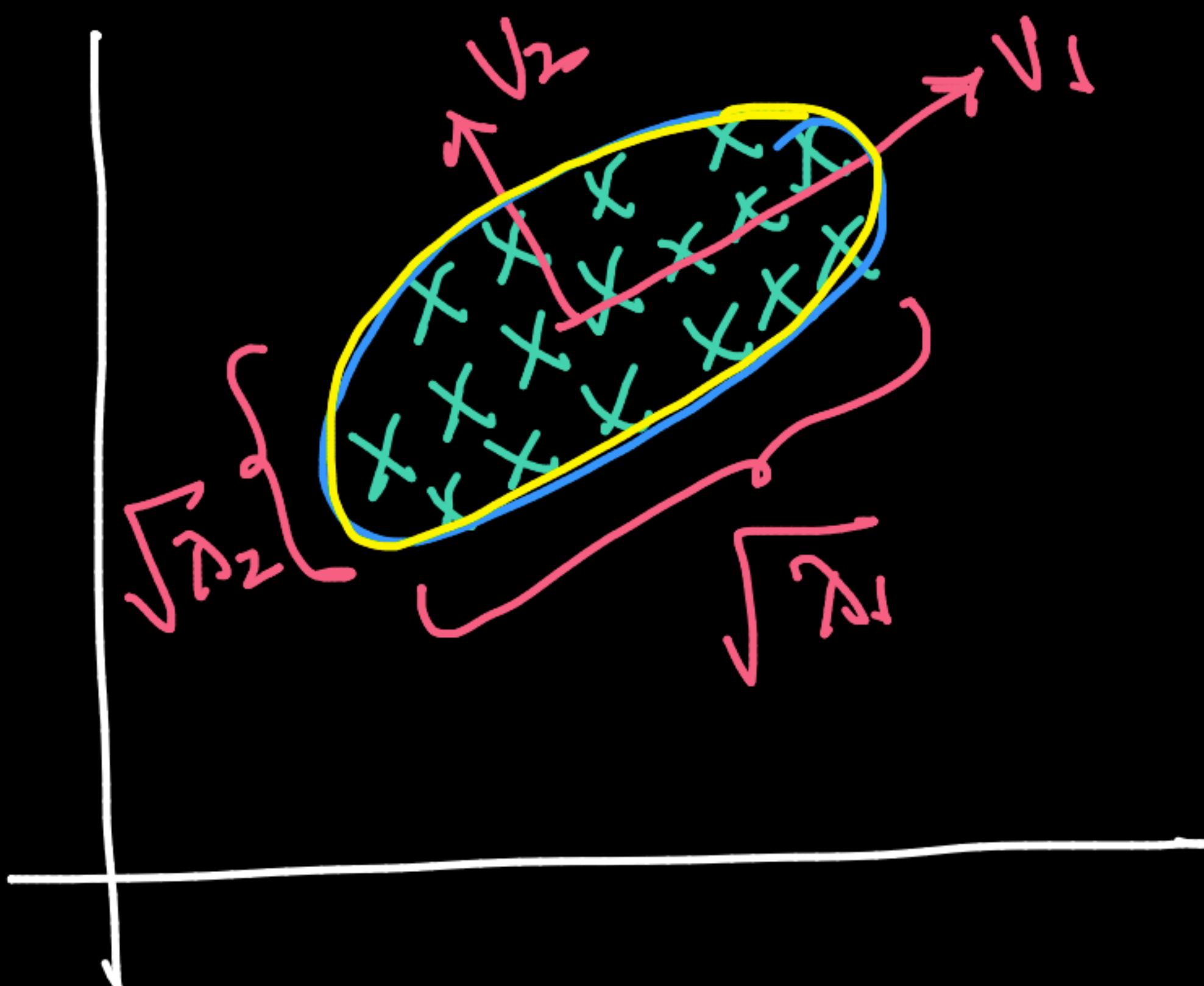
Why Symmetric

$$A_{ij} = A_{ji} \quad i \neq j$$

$$\begin{aligned} \text{COV}(X, Y) &= \text{COV}(Y, X) \\ \text{COV}(X, X) &= \text{Var}(X) \end{aligned} \quad \left. \begin{array}{l} \text{PCA} \\ \hline \end{array} \right\} \equiv$$

PCA

1 & 2 dim
Gaussian



eig-vec $\wedge \Sigma$

$$\tilde{\Sigma} = \text{cov}(\mathcal{P}) \\ \text{with } \mathcal{P} \in \mathbb{R}^2$$

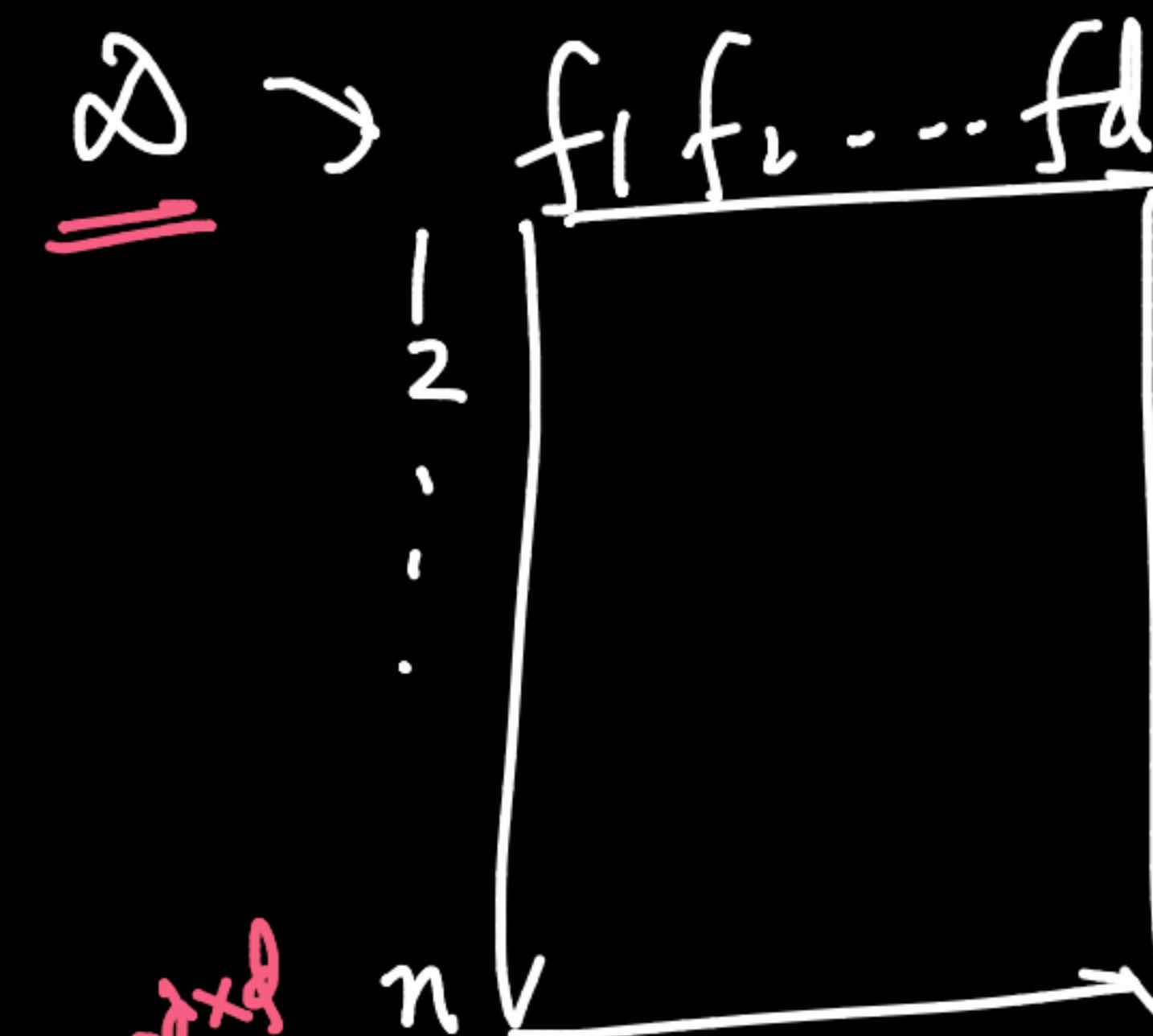
PCA :-

\downarrow

eig-vec of $S\sqrt{\text{cov}(\mathcal{P})}$

$\text{cov}(\mathcal{P})$

d-dim Space



3-Gaussians

$$\text{find } \begin{cases} G_1 : \underline{\mu_1, \Sigma_1} \\ G_2 : \underline{\mu_2, \Sigma_2} \\ G_3 : \underline{\mu_3, \Sigma_3} \end{cases} \quad \text{params}$$

$$x_i \rightarrow \begin{cases} G_1 : p_{i1} \\ G_2 : p_{i2} \\ G_3 : p_{i3} \end{cases}$$

1D:

3-Gaussian

G₁: μ_1, σ_1

G₂: μ_2, σ_2

G₃: μ_3, σ_3

PDF of 1D-Gaussian

$$\underline{x_i} = P(x_i | G_j)$$

$$P(x_i | G_1) = 0.2 / 0.8$$

$$P(x_i | G_2) = 0.1 / 0.8$$

$$P(x_i | G_3) = 0.5 / 0.8$$

1

Learn to edit
Community portal
Recent changes
Upload file
Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item
Print/export
Download as PDF
Printable version
In other projects
Wikimedia Commons
Languages
العربية
Español
Français
हिन्दी
মারাঠী
Português
தமிழ்
اردو
中文
文 60 more

Gaussian distribution is said to be **normally distributed**, and is called a **normal deviate**. Normal distributions are important in **statistics** and are often used in the **natural and social sciences** to represent real-valued **random variables** whose distributions are not known.^{[2][3]} Their importance is partly due to the **central limit theorem**. It states that, under some conditions, the average of many samples (observations) of a random variable with finite mean and variance is itself a random variable—whose distribution **converges** to a normal distribution as the number of samples increases. Therefore, physical quantities that are expected to be the sum of many independent processes, such as **measurement errors**, often have distributions that are nearly normal.^[4]

Moreover, Gaussian distributions have some unique properties that are valuable in analytic studies. For instance, any linear combination of a fixed collection of normal deviates is a normal deviate. Many results and methods, such as **propagation of uncertainty** and **least squares** parameter fitting, can be derived analytically in explicit form when the relevant variables are normally distributed.

A normal distribution is sometimes informally called a **bell curve**.^[5] However, many other distributions are bell-shaped (such as the **Cauchy**, **Student's t**, and **logistic** distributions).

The univariate probability distribution is generalized for vectors in the **multivariate normal distribution** and for matrices in the **matrix normal distribution**.

The red curve is the **standard normal distribution**

Cumulative distribution function	
$\mu = 0, \sigma^2 = 0.2$	Blue
$\mu = 0, \sigma^2 = 1.0$	Red
$\mu = 0, \sigma^2 = 5.0$	Yellow
$\mu = -2, \sigma^2 = 0.5$	Green

Notation $\mathcal{N}(\mu, \sigma^2)$

Parameters $\mu \in \mathbb{R}$ = mean (location)
 $\sigma^2 \in \mathbb{R}_{>0}$ = variance (squared scale)

Support $x \in \mathbb{R}$

PDF
$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

CDF
$$\frac{1}{2} \left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right) \right]$$

Quantile $\mu + \sigma\sqrt{2} \operatorname{erf}^{-1}(2p - 1)$

Mean μ

Median μ

Mode μ

Variance σ^2

MAD $\sigma\sqrt{2/\pi}$

Skewness 0

Ex. 3

kurtosis

Entropy
$$\frac{1}{2} \log(2\pi\sigma^2) + \frac{1}{2}$$

MGF
$$\exp(\mu t + \sigma^2 t^2/2)$$

Learn to edit
Community portal
Recent changes
Upload file
Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item
Print/export
Download as PDF
Printable version
In other projects
Wikimedia Commons
Languages
العربية
Español
Français

(and also its median and mode), while the parameter σ is its standard deviation. The variance of the distribution is σ^2 .^[1] A random variable with a Gaussian distribution is said to be **normally distributed**, and is called a **normal deviate**.

Normal distributions are important in **statistics** and are often used in the **natural and social sciences** to represent real-valued **random variables** whose distributions are not known.^{[2][3]} Their importance is partly due to the **central limit theorem**. It states that, under some conditions, the average of many samples (observations) of a random variable with finite mean and variance is itself a random variable—whose distribution **converges** to a normal distribution as the number of samples increases. Therefore, physical quantities that are expected to be the sum of many independent processes, such as **measurement errors**, often have distributions that are nearly normal.^[4]

Moreover, Gaussian distributions have some unique properties that are valuable in analytic studies. For instance, any linear combination of a fixed collection of normal deviates is a normal deviate. Many results and methods, such as **propagation of uncertainty** and **least squares** parameter fitting, can be derived analytically in e

The red curve is the *standard normal distribution*

Cumulative distribution function

Notation	$\mathcal{N}(\mu, \sigma^2)$
Parameters	$\mu \in \mathbb{R}$ = mean (location) $\sigma^2 \in \mathbb{R}_{>0}$ = variance (squared scale)
Support	$x \in \mathbb{R}$
PDF	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x-\mu}{\sigma}\right)^2}$ (Mu, Sigma circled)
CDF	$\frac{1}{2} \left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right) \right]$
Quantile	$\mu + \sigma\sqrt{2} \operatorname{erf}^{-1}(2p - 1)$
Mean	μ
Median	μ
Mode	μ

d-dim Gaussian: $\mu_d, \Sigma_{d \times d}$

$x_i \in \mathbb{R}^d$

$P(x_i \mid \text{Normal}_d) = \dots$

Non-degenerate case [edit]

The multivariate normal distribution is said to be "non-degenerate" when the symmetric covariance matrix Σ is positive definite. In this case the distribution has density^[5]

μ_d, Σ_{dxd}

$$f_{\mathbf{X}}(x_1, \dots, x_d) = \frac{\exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}))}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}}$$

determinant

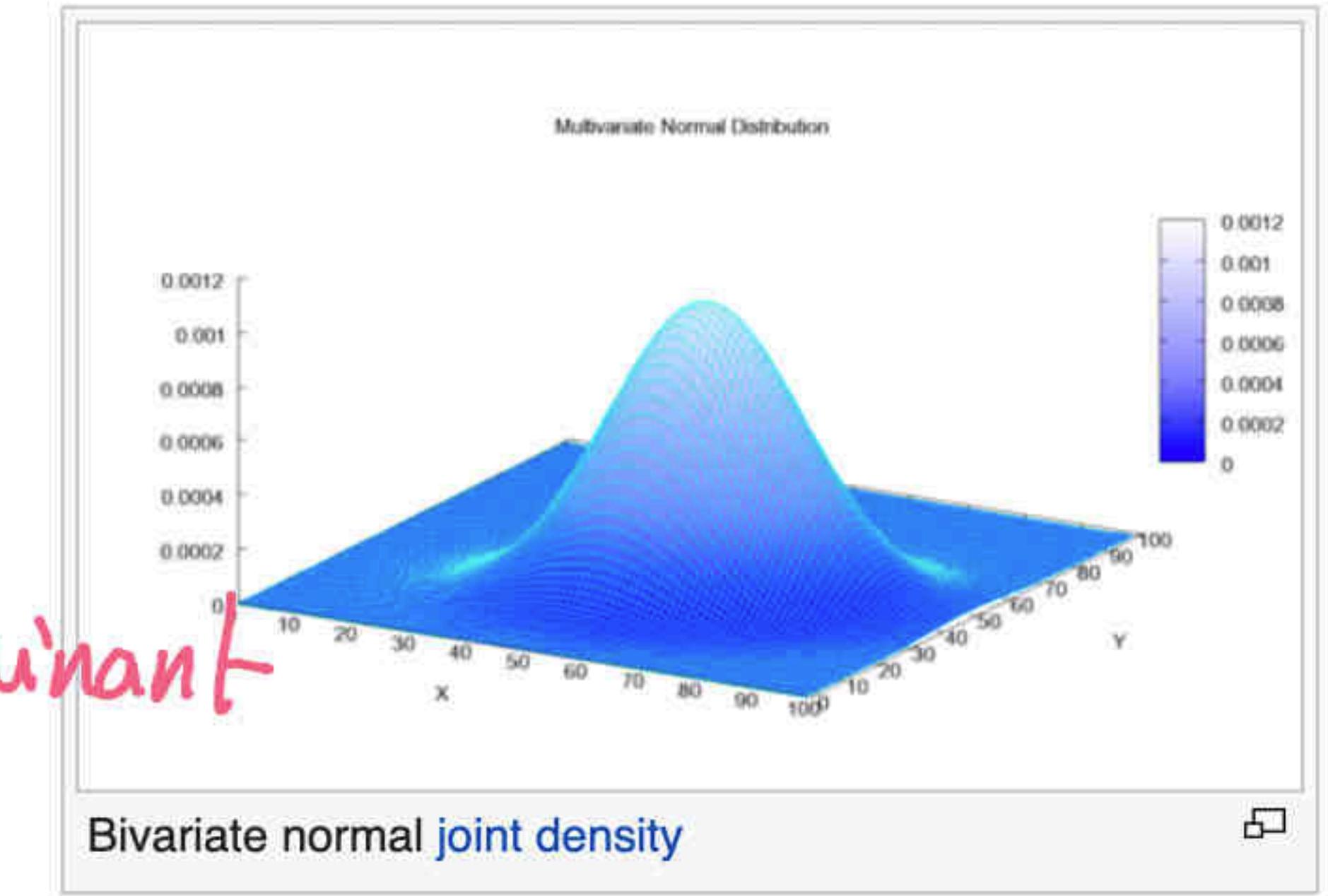
where \mathbf{x} is a real k -dimensional column vector and $|\boldsymbol{\Sigma}| \equiv \det \boldsymbol{\Sigma}$ is the determinant of $\boldsymbol{\Sigma}$, also known as the generalized variance.

The equation above reduces to that of the univariate normal distribution if $\boldsymbol{\Sigma}$ is a 1×1 matrix (i.e. a single real number).

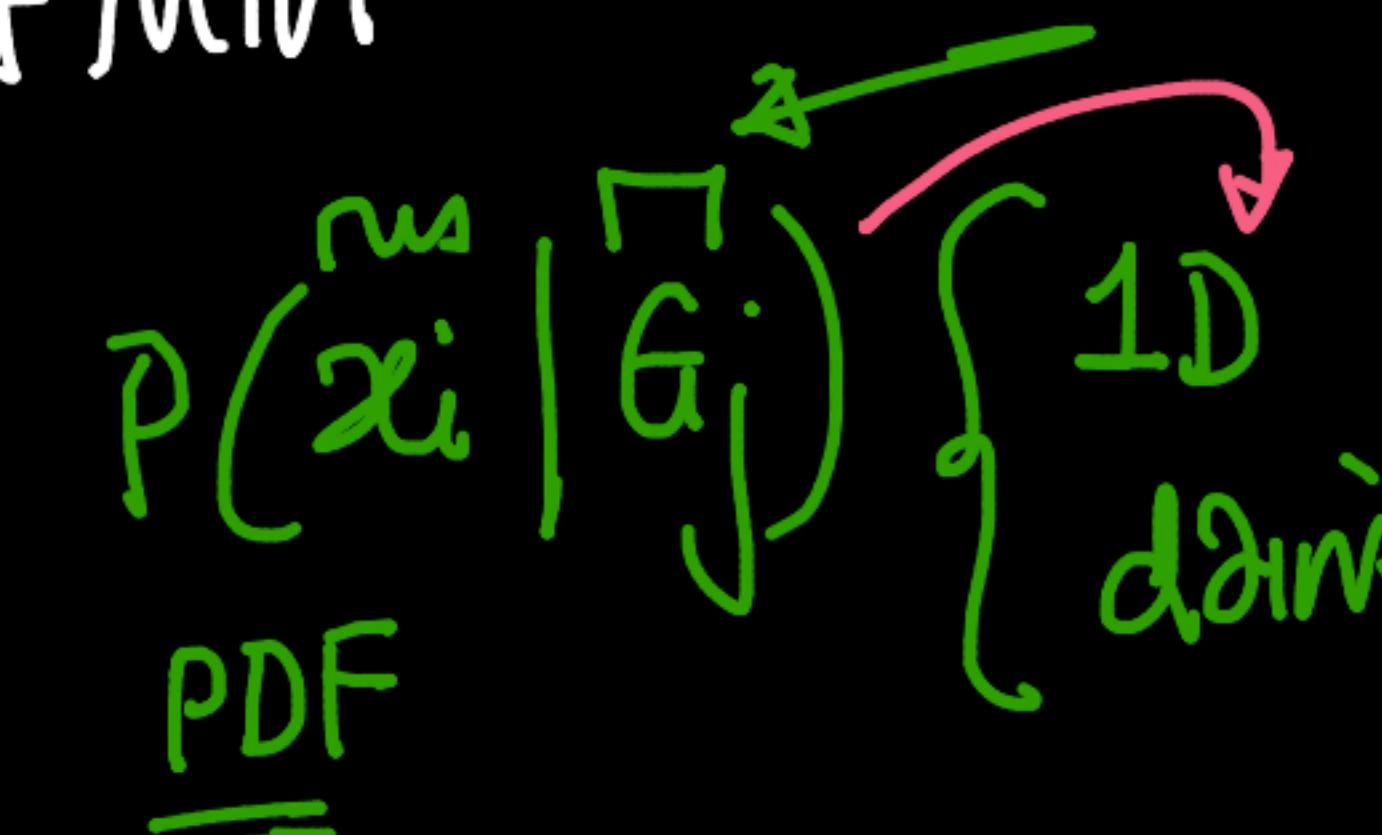
The circularly symmetric version of the complex normal distribution has a slightly different form.

Each iso-density locus — the locus of points in k -dimensional space each of which gives the same particular value of the density — is an ellipse or its higher-dimensional generalization; hence the multivariate normal is a special case of the elliptical distributions.

The quantity $\sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$ is known as the Mahalanobis distance, which represents the distance of the test point \mathbf{x} from the mean $\boldsymbol{\mu}$. Note that in the case when $\boldsymbol{\Sigma} = I_d$, the distribution reduces to a univariate normal distribution



Recap:

- 1D - GMM (intuition)
- d-dim Gaussian - dist ($\mu_d, \Sigma_{d \times d}$)
- d-dim GMM
- $x_i \rightarrow P(x_i | G_j)$  PDF fundim
" " "

→ find

k -Gaussians

given $\mathcal{D} \in \mathbb{R}^d$

$$\mu_i, \Sigma_i \quad \forall i: 1 \rightarrow k$$

\downarrow \downarrow
 d -dim dxd matrix

→ find
=

k-Gaussians

given $\mathcal{D} \in \mathbb{R}^d$

$$\mu_i, \Sigma_i \quad \forall i: 1 \rightarrow k$$

\downarrow \downarrow
 d -dim dxd matrix

→ find
=

k-Gaussians

given $\mathcal{D} \in \mathbb{R}^d$

$$\mu_i, \Sigma_i \quad \forall i: 1 \rightarrow k$$

\downarrow \downarrow
 d -dim dxd matrix

→ find

k -Gaussians

$\mu_i, \Sigma_i \quad \forall i: 1 \rightarrow k$

d -dim

given $\tilde{D} \in \mathbb{R}^d$

Probabilistic

optimization

EM (coordinate ascent)

Code



Non-degenerate case [edit]

The multivariate normal distribution is said to be "non-degenerate" when the symmetric covariance matrix Σ is positive definite. In this case the distribution has density^[5]

$$f_{\mathbf{X}}(x_1, \dots, x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}}$$

where \mathbf{x} is a real k -dimensional column vector and $|\boldsymbol{\Sigma}| \equiv \det \boldsymbol{\Sigma}$ is the determinant of $\boldsymbol{\Sigma}$, also known as the generalized variance.

The equation above reduces to that of the univariate normal distribution if $\boldsymbol{\Sigma}$ is a 1×1 matrix (i.e. a single real number).

The circularly symmetric version of the complex normal distribution has a slightly different form.

Each iso-density locus — the locus of points in k -dimensional space each of which gives the same particular value of the density — is an ellipse or its higher-dimensional generalization; hence the multivariate normal is a special case of the elliptical distributions.



The quantity $\sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$ is known as the Mahalanobis distance, which represents the distance of the test point \mathbf{x} from the mean $\boldsymbol{\mu}$. Note that in the case when $k = 1$, the distribution reduces to a univariate normal distribution

