K-means
↙ optimize
↘ Lloyd's

Topics:

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

→ probabilistic ✓

- Low-voice
- breaks as needed

- GMMs
- Misc Topics ...
- Anamoly detection ....
- t-SNE & UMAP ←
- Rec Sys
- Time-series

12

- SQL, ✓
- Intro to Tableau

- DL - MLP .. (wlock)
  - DL - CV (1m)
  - DL - NLP (1m)
  - ⋮

```
[ ]  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

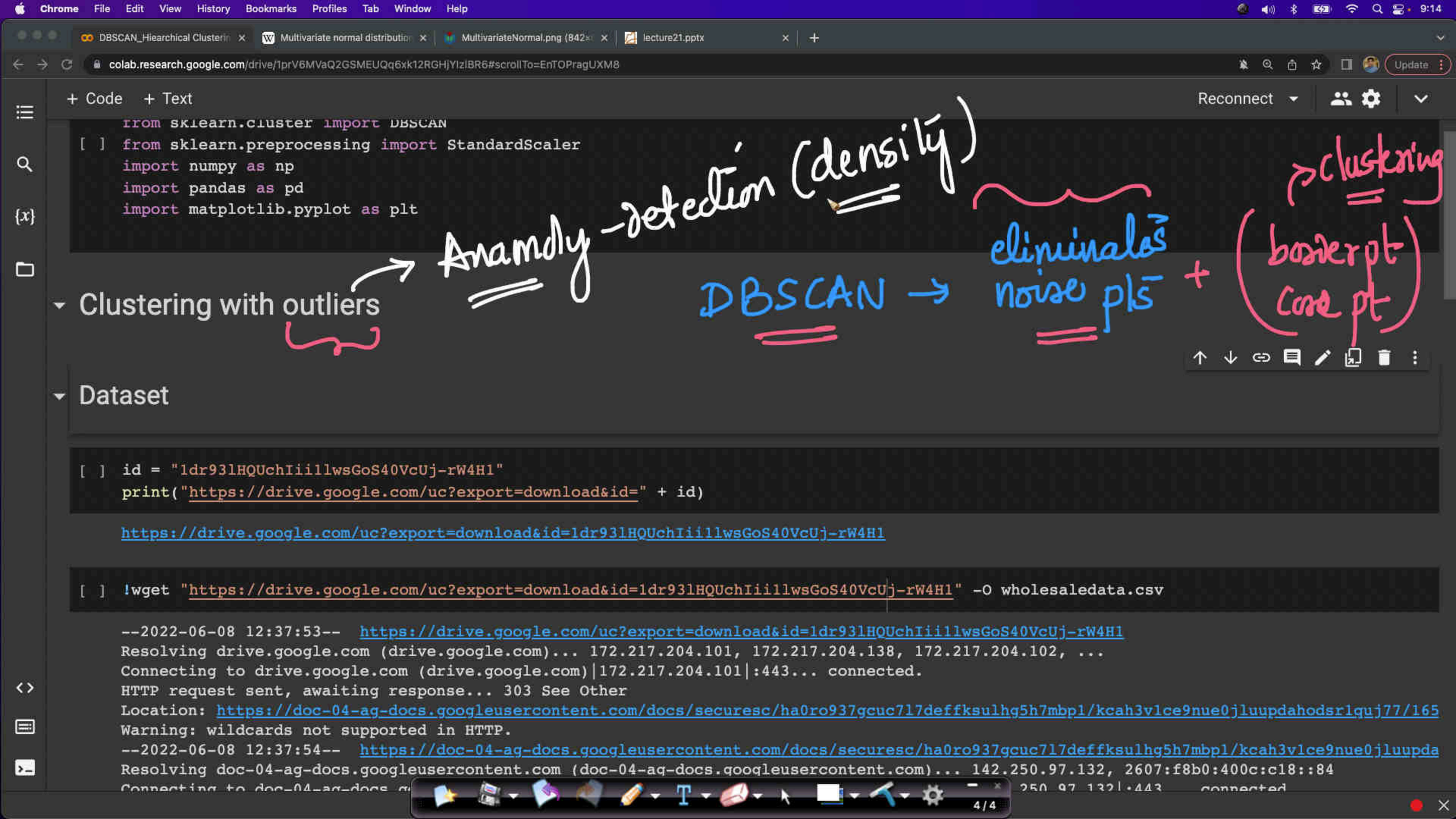
[ ] ↳ 5 cells hidden

Simple Visualization

[ ] ↳ 2 cells hidden

DBSCAN

[ ] ↳ 6 cells hidden


Chrome File Edit View History Bookmarks Profiles Tab Window Help
DBSCAN_Hiearchical Clusterin
Multivariate normal distributio
MultivariateNormal.png (842×
lecture21.pptx
colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzlBR6#scrollTo=EnTOPragUXM8
Update
+ 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
Anamdy-detection (density)
Clustering with outliers
DBSCAN → eliminates noise pts +
clustering
boundary pt
Core pt
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/kcah3v1ce9nue0jluupdahodsr1quj77/165
Warning: wildcards not supported in HTTP.
--2022-06-08 12:37:54-- https://doc-04-ag-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc717deffksulhg5h7mbp1/kcah3v1ce9nue0jluupda
Resolving doc-04-ag-docs.googleusercontent.com (doc-04-ag-docs.googleusercontent.com)... 142.250.97.132, 2607:f8b0:400c:c18::84
4/4

```
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)
```

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

```
df = pd.read_csv('./wholesaledata.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
```

Visualize
2-Variables

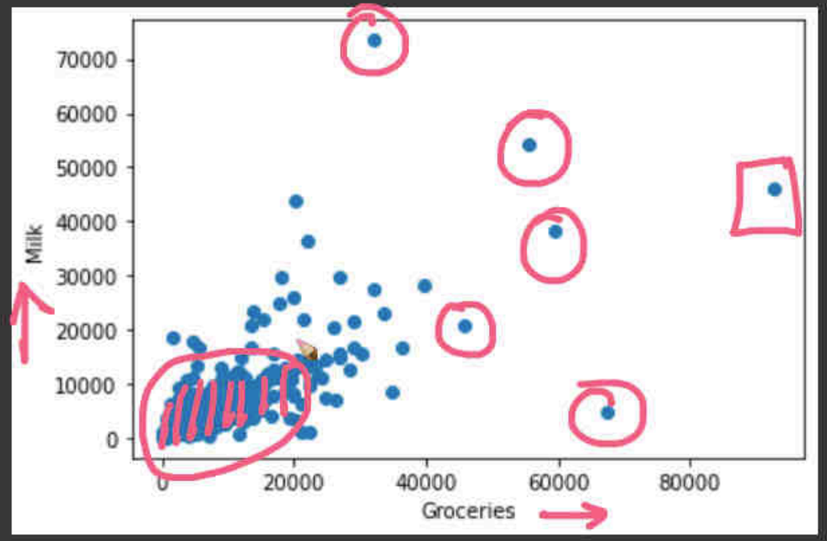
```
df.head()
```

	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

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

Simple Visualization

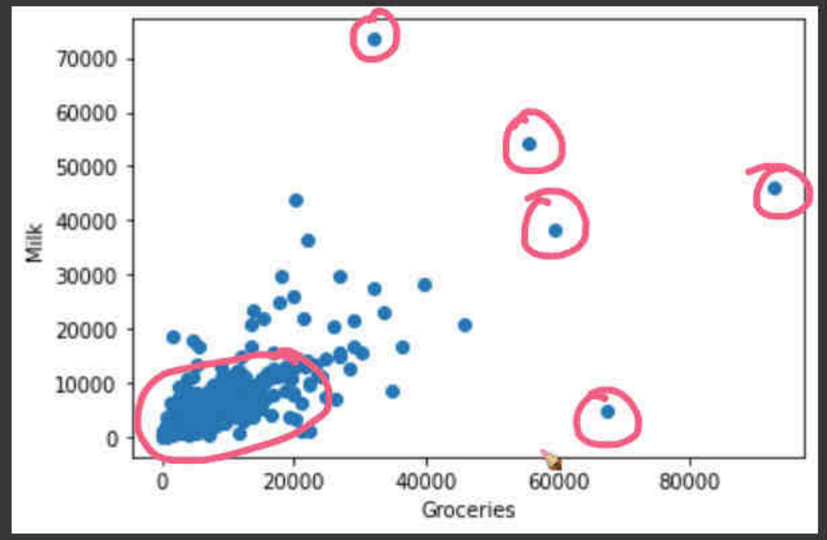
```
# Let's plot two features data now
x = df['Grocery']
y = df['Milk']

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



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

Simple Visualization

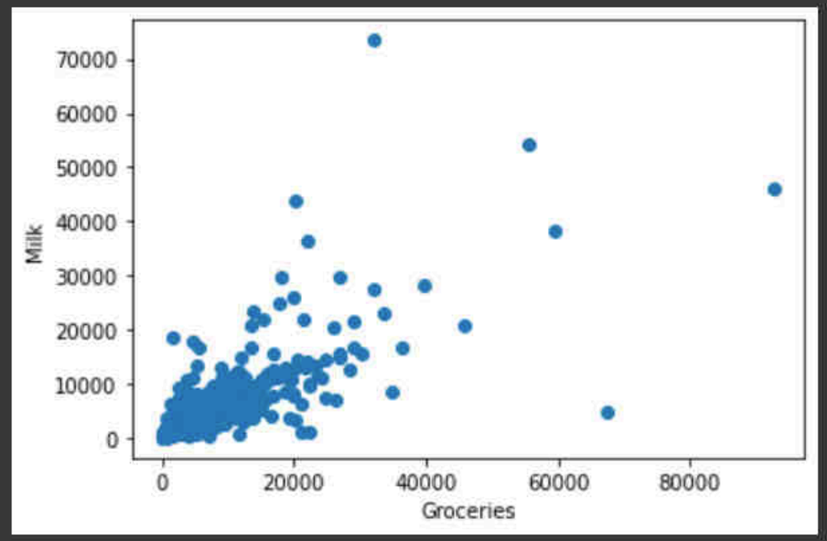
```
# Let's plot two features data now
x = df['Grocery']
y = df['Milk']

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



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

Simple Visualization

```
# 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()
```



curse of dim

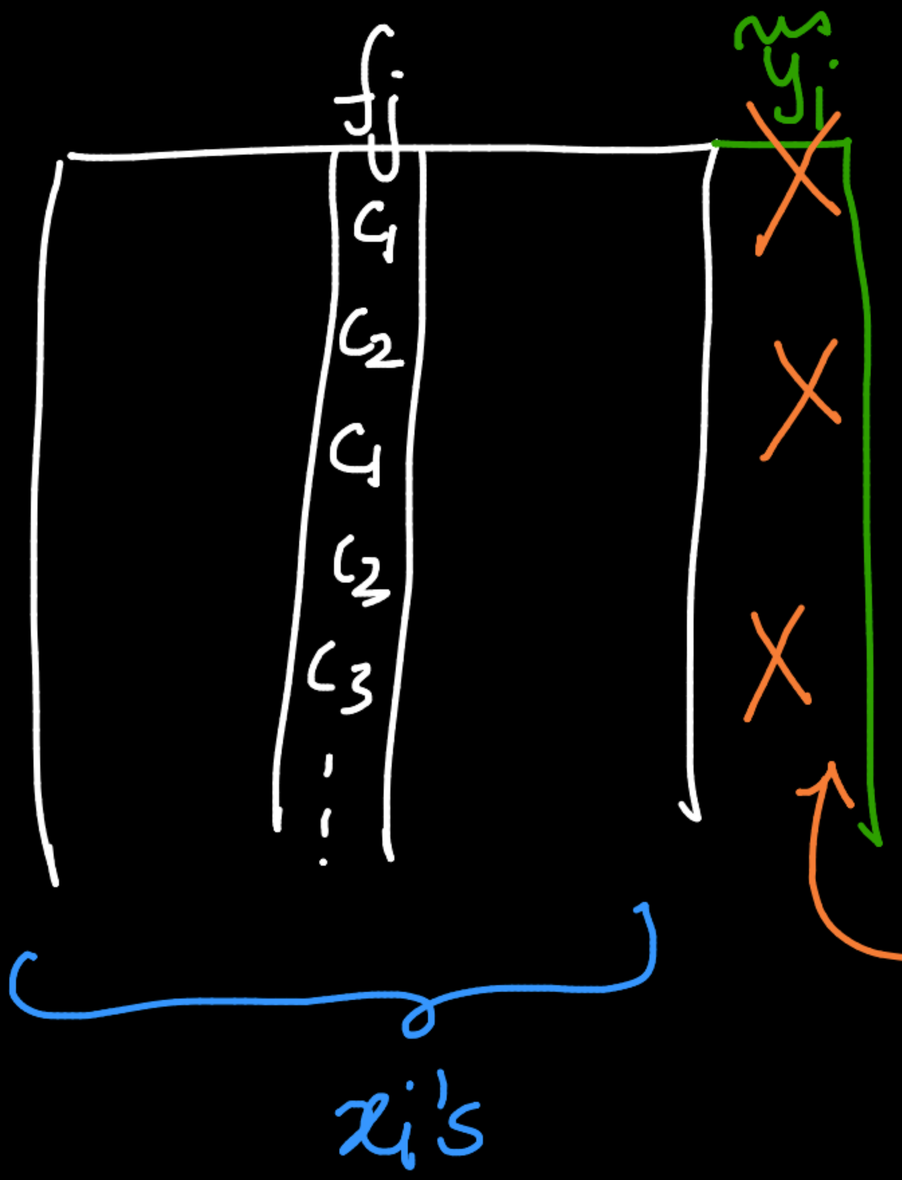
- OHE
- Label encoding

0, 1, 2, ... k-1

Ordering

$C_1, C_2, C_3 \rightarrow 1, 2, 3$

$C_1, C_2 \rightarrow 0, 1$



Target encoding

cannot do Target encoding

y_i's not present

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:26

DBSCAN_Hiearchical Clusterin | Multivariate normal distributio | MultivariateNormal.png (842× | lecture21.pptx

colab.research.google.com/drive/1prV6MVaQ2GSMEUQq6xk12RGHjYIzlBR6#scrollTo=7BQNvrhbWP2P

Update

+ Code + Text

Reconnect

[ ] ↳ 5 cells hidden

Simple Visualization

[ ] ↳ 2 cells hidden

DBSCAN

eps

Range Query

(?) Yes (why?)

No ....

f2: salary → 120,000

f1: age → 29...

```
[ ] df = df[['Grocery', 'Milk']]
    std_scaler = StandardScaler().fit(df)
    std_df = 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,  0, -1,  0,  0,  0,  0, -1,  0,  0,  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,  0, -1,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,
       -1, -1,  0,  0,  0,  0,
```

11 / 11

Simple Visualization

[ ] ↳ 2 cells hidden

DBSCAN

```
[ ] df = df[['Grocery', 'Milk']]
    std_scaler = StandardScaler().fit(df)
    std_df = 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,  0, -1,  0,  0,  0,  0, -1,  0,  0,  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,  0, -1,  0,  0,  0,  0,  0, -1,  0,  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,  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,
```

scikit learn

Prev Up Next

scikit-learn 1.1.1
Other versions

Please cite us if you use the software.

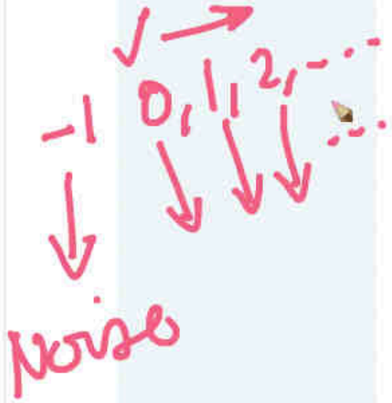
sklearn.cluster.DBSCAN
Examples using
sklearn.cluster.DBSCAN

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.

```
       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, -1,  0,  0,  0,
       0,  0, -1,  0,  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,
      -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, -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, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1,  0,  0])
```

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

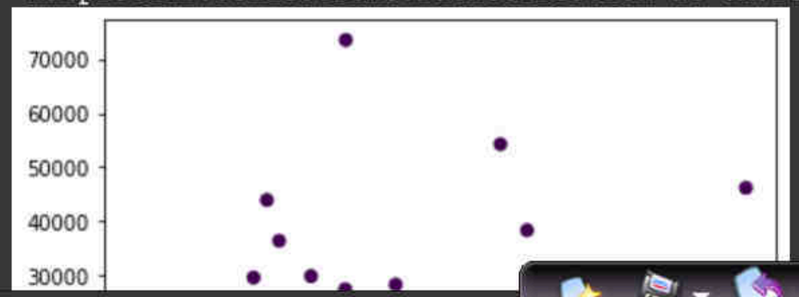
```
[ ] out_df['label'].value_counts()
```

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

0.5 15
Eps, Minpts

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

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



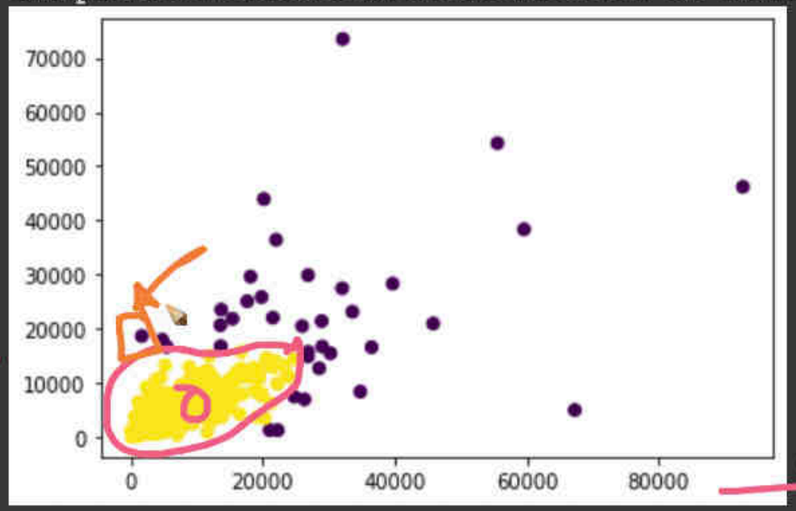
```
[ ] out_df = df.copy()
    out_df['label'] = dbsc.labels_
```

```
[ ] 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>
```



Milk

Grocery

{ eps = 0.5
  MinPts = 15

0.5

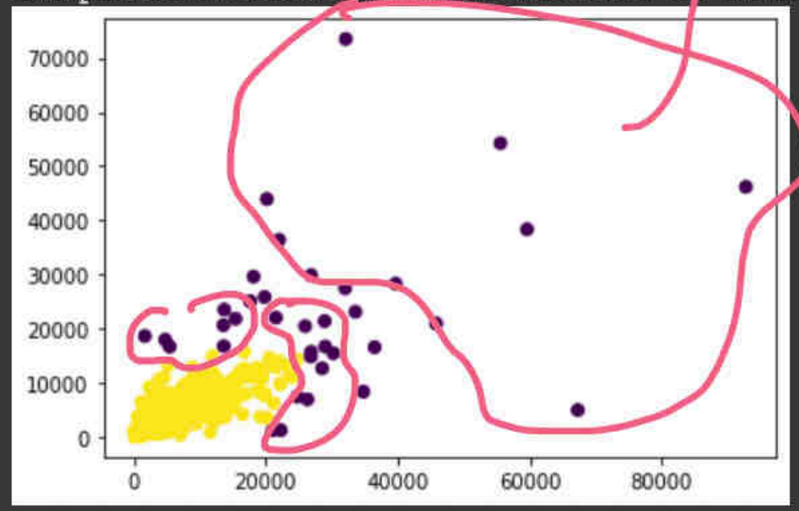
```
[ ] out_df = df.copy()
    out_df['label'] = dbsc.labels_
```

```
[ ] 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 observe

(2) all-dimensions

Categorical → OHE / LE

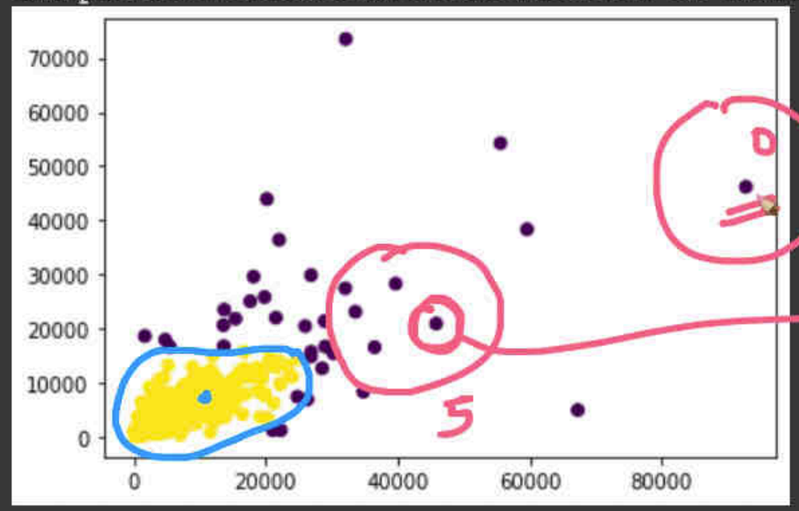
```
[ ] out_df = df.copy()
    out_df['label'] = dbsc.labels_
```

```
[ ] 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>
```



Polar plots ✓

Eps ; pts ✓

t-SNE / UMAP ✓

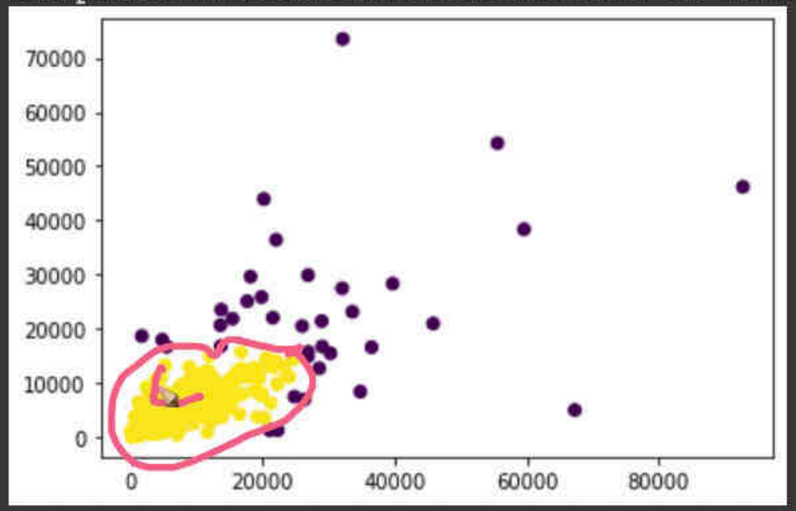
```
out_df = df.copy()
out_df['label'] = dbsc.labels_
```

```
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>
```



KMeans

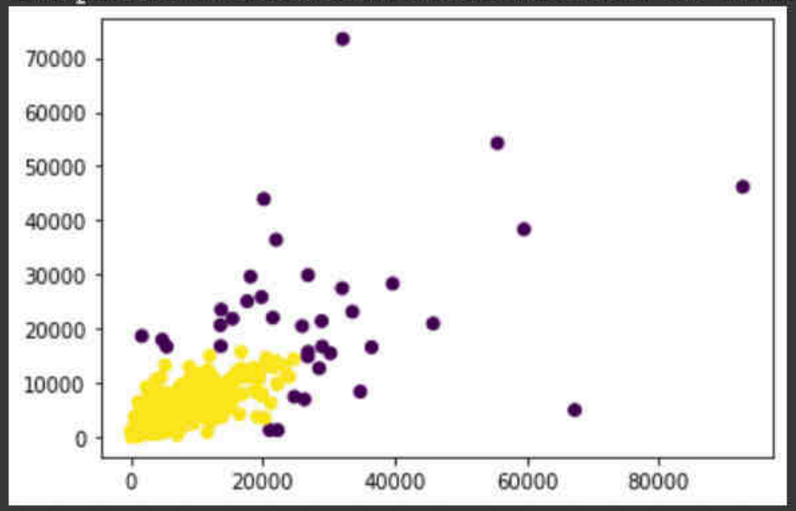
```
out_df = df.copy()
out_df['label'] = dbsc.labels_
```

```
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>
```



{ Eps = 0.5
{ MinPts = 15

KMeans

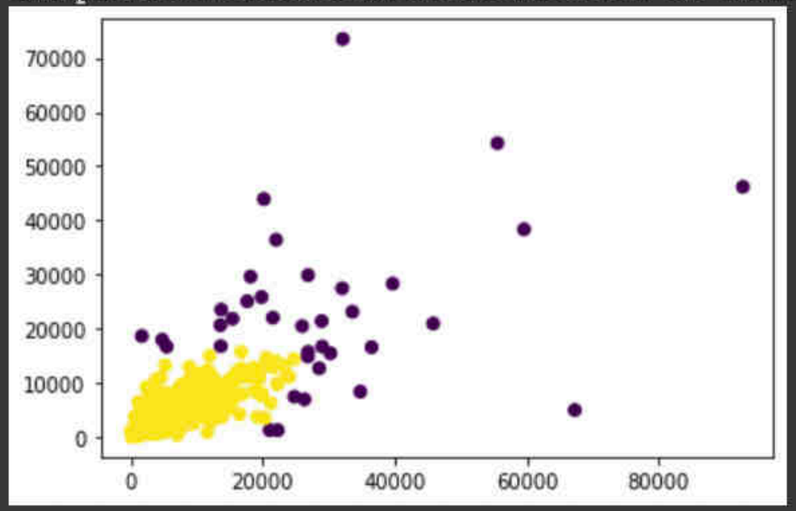
```
out_df = df.copy()
out_df['label'] = dbsc.labels_
```

```
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>
```



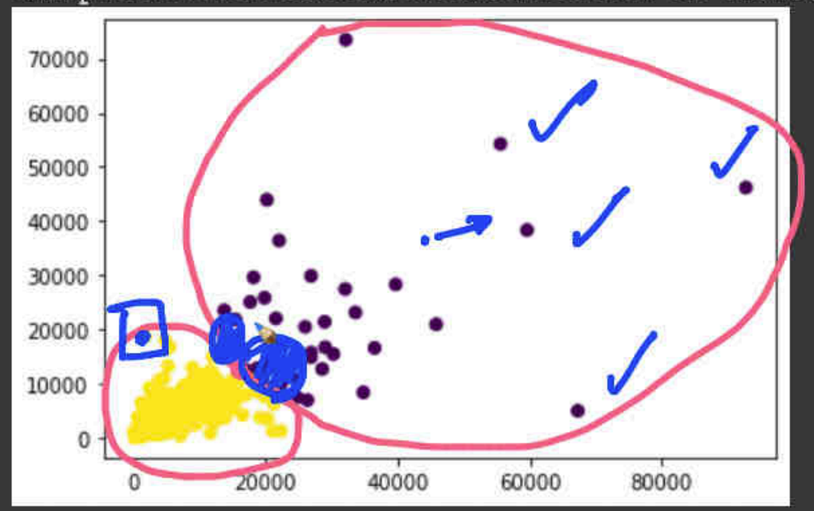
KMeans

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

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

```
[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_symobols))
yf_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

```
[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_symobols))
yf_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',
```

```
[ ]  import yfinance as yf


     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'])
```

ROCE, RoI . .

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

```
[ ]  import yfinance as yf


     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
```

```
    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	-0.235
1	2596018061312	1742929	-0.002	143.991	2.898187e+11	0.11496	0.10457	-0.002

```
[ ]     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	-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()
```

```
 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_symobols
```

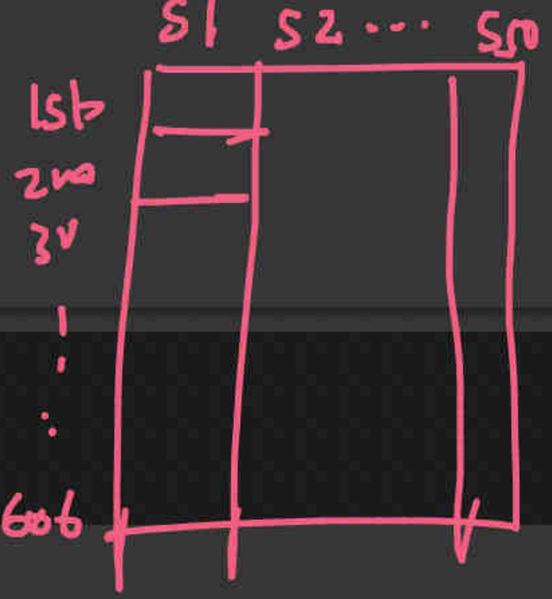
```
[**********100%**********]  50 of 50 completed
```

```
stock_prices.shape
```

```
(606, 50)
```

```
stock_prices.head()
```

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



```
[ ] stock_prices.columns = list_of_symobols
```

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

```
[ ] stock_prices.shape
```

```
(606, 50)
```

```
[ ] stock_prices.head()
```

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

```
[ ] stock_prices.columns = list_of_symobols
```

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

```
[ ] stock_prices.shape
```

```
(606, 50)
```


```
[ ] stock_prices.head()
```

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

30 / 30

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

```
##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	1768.3						397.889709	2896.747559	469.682129	...	422.8171

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

```
##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	1768.3						397.889709	2896.747559	469.682129	...	422.8171

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

$\left(\frac{end}{start} - 1\right) + 100 = Y.age$

```
##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	1768.3						397.889709	2896.747559	469.682129	...	422.8171

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
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 × 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	earningsGro
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

[ ]

```
# 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
```

Hiearchical Clustering

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)
```

```
earningsQuarterlyGrowth     2
bookValue                   1
totalRevenue                1
returnOnAssets             28
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)
```

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

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

→ scipy
→ sklearn

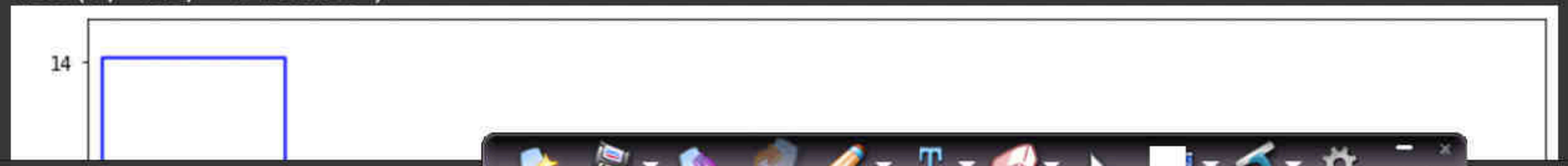
```
# import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

# 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')
```



```
scaler.fit(df)
X = scaler.transform(df)
```

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

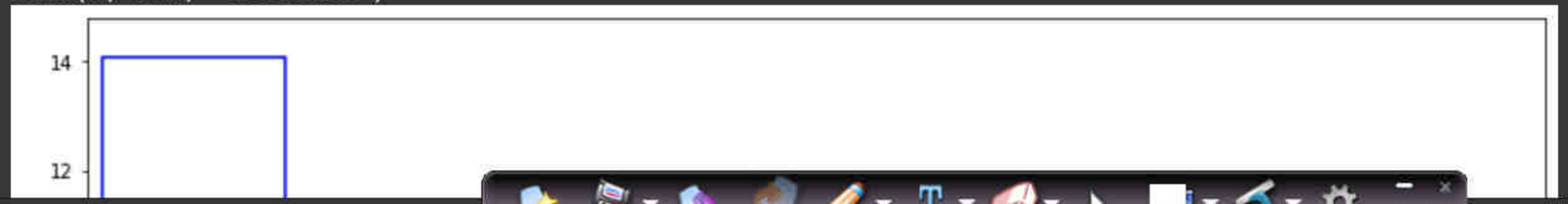
```
# import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

# 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)
ax.set_ylabel('distance')
```

```
Text(0, 0.5, 'distance')
```



```
scaler.fit(df)
X = scaler.transform(df)
```

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

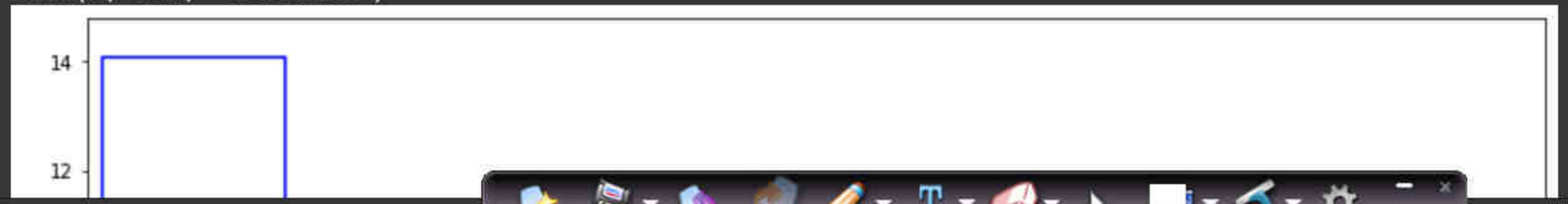
```
# import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

# 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)
ax.set_ylabel('distance')
```

```
Text(0, 0.5, 'distance')
```



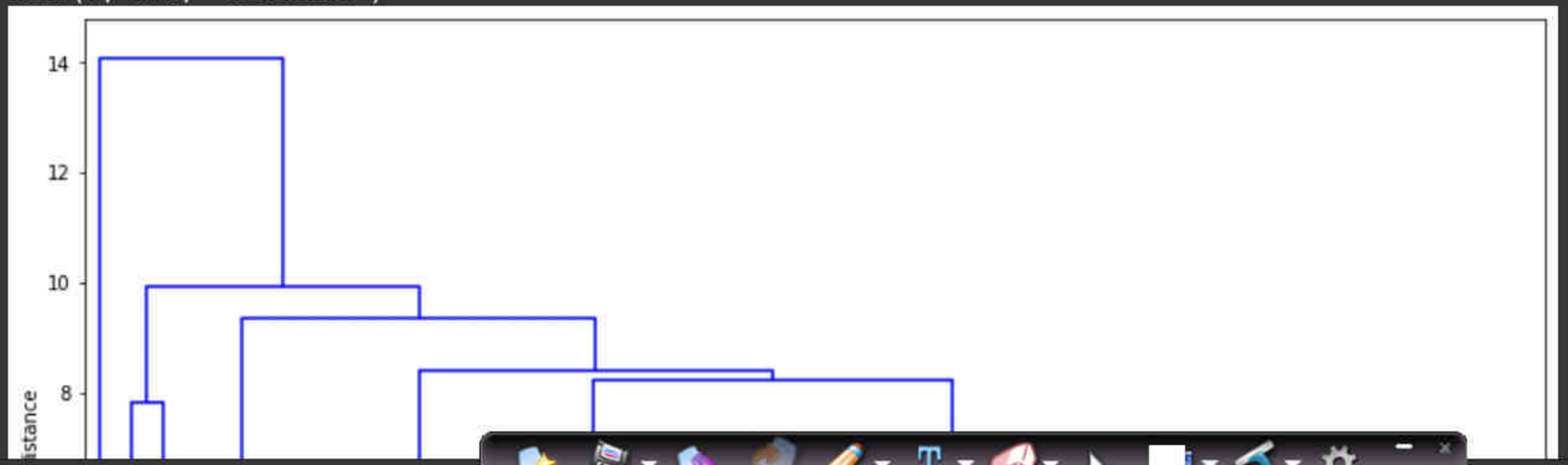
```
# import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

# 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')
```



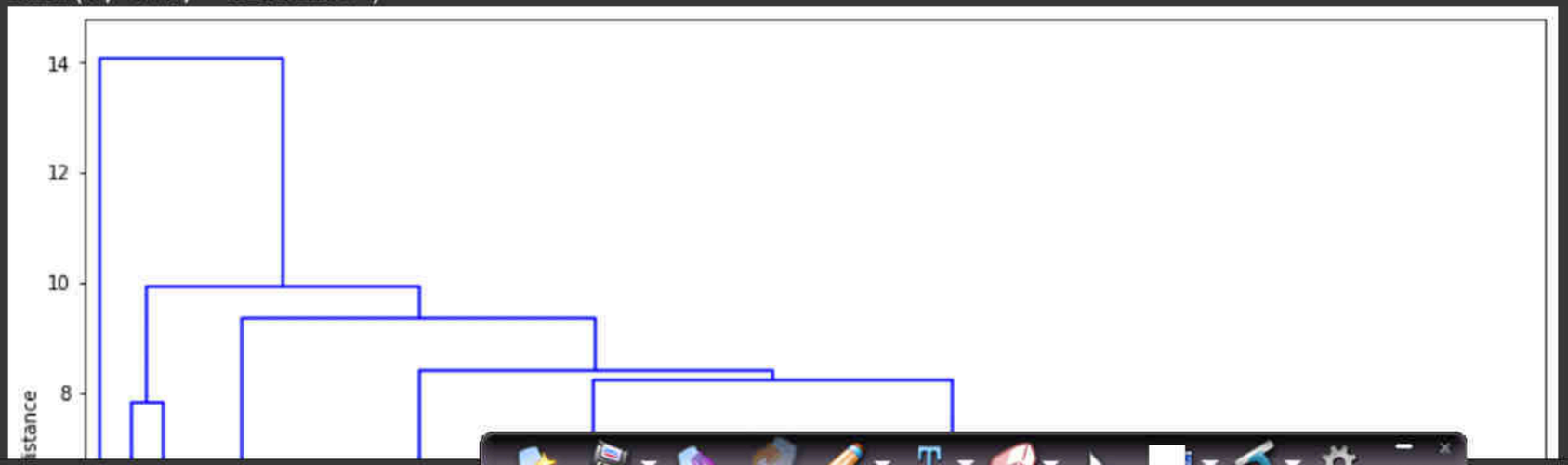
```
# import hierarchical clustering libraries
import scipy.cluster.hierarchy as sch

# 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
```

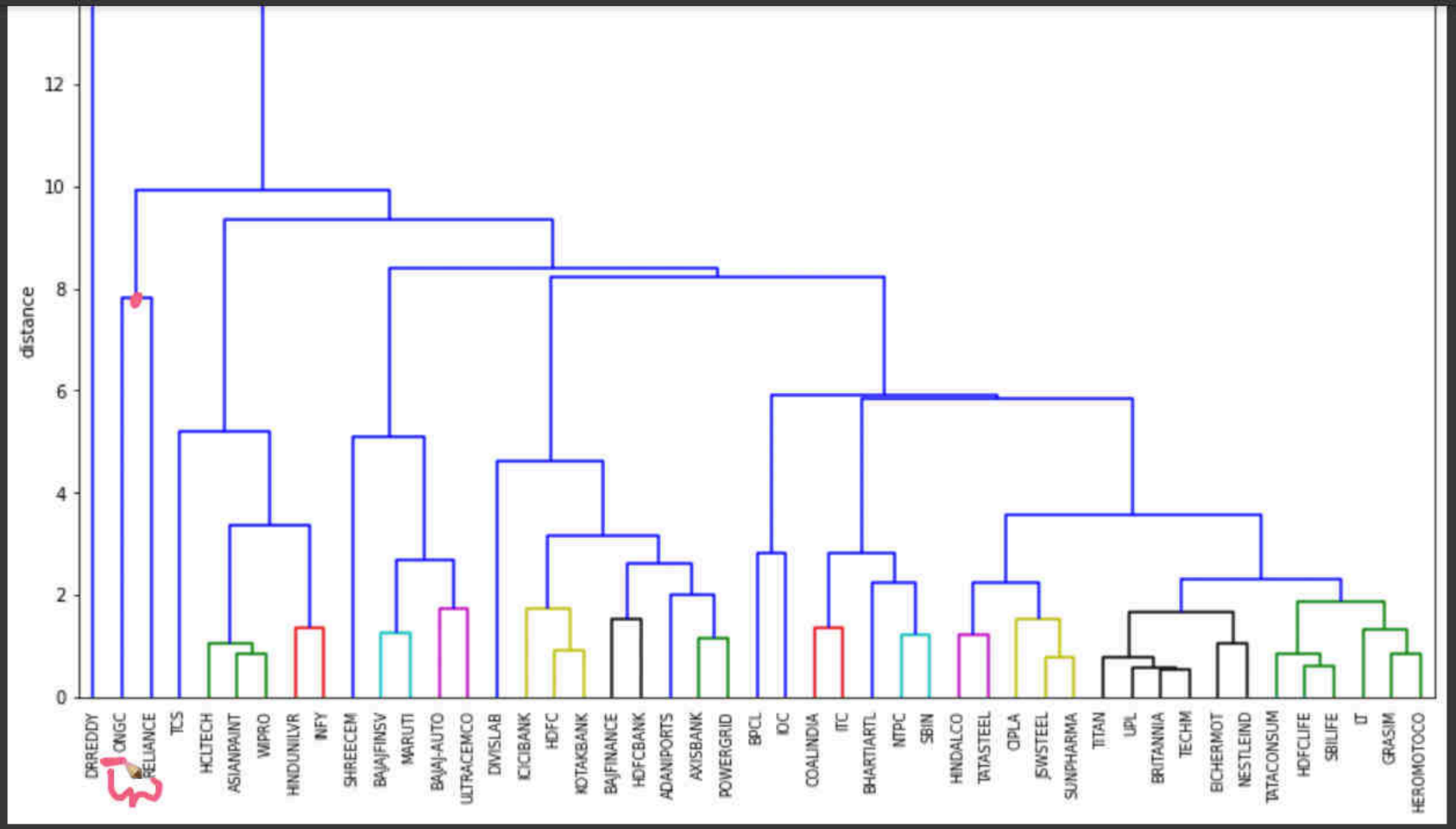
```
(46, 4)
```

```
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')
```

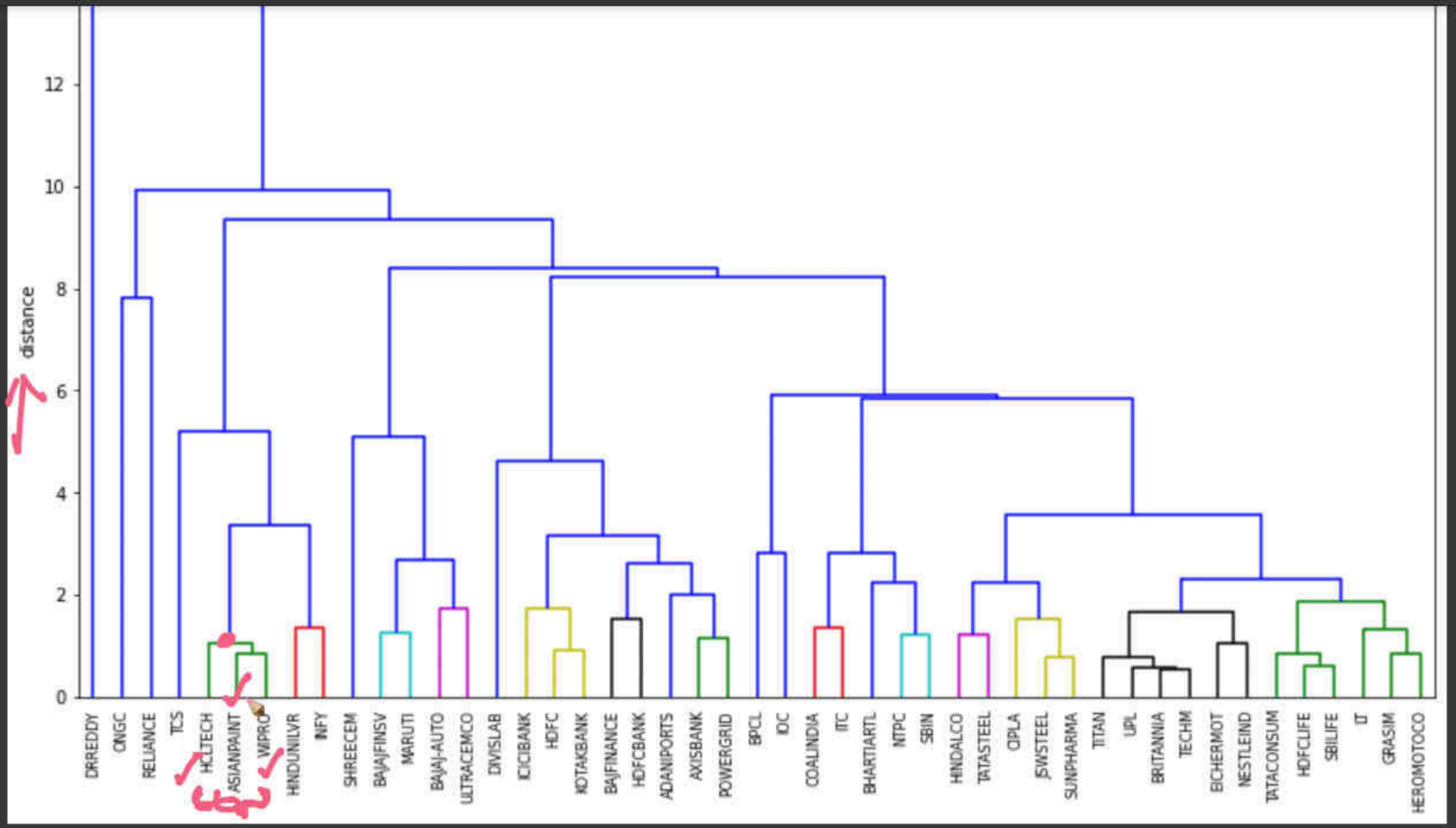
```
Text(0, 0.5, 'distance')
```



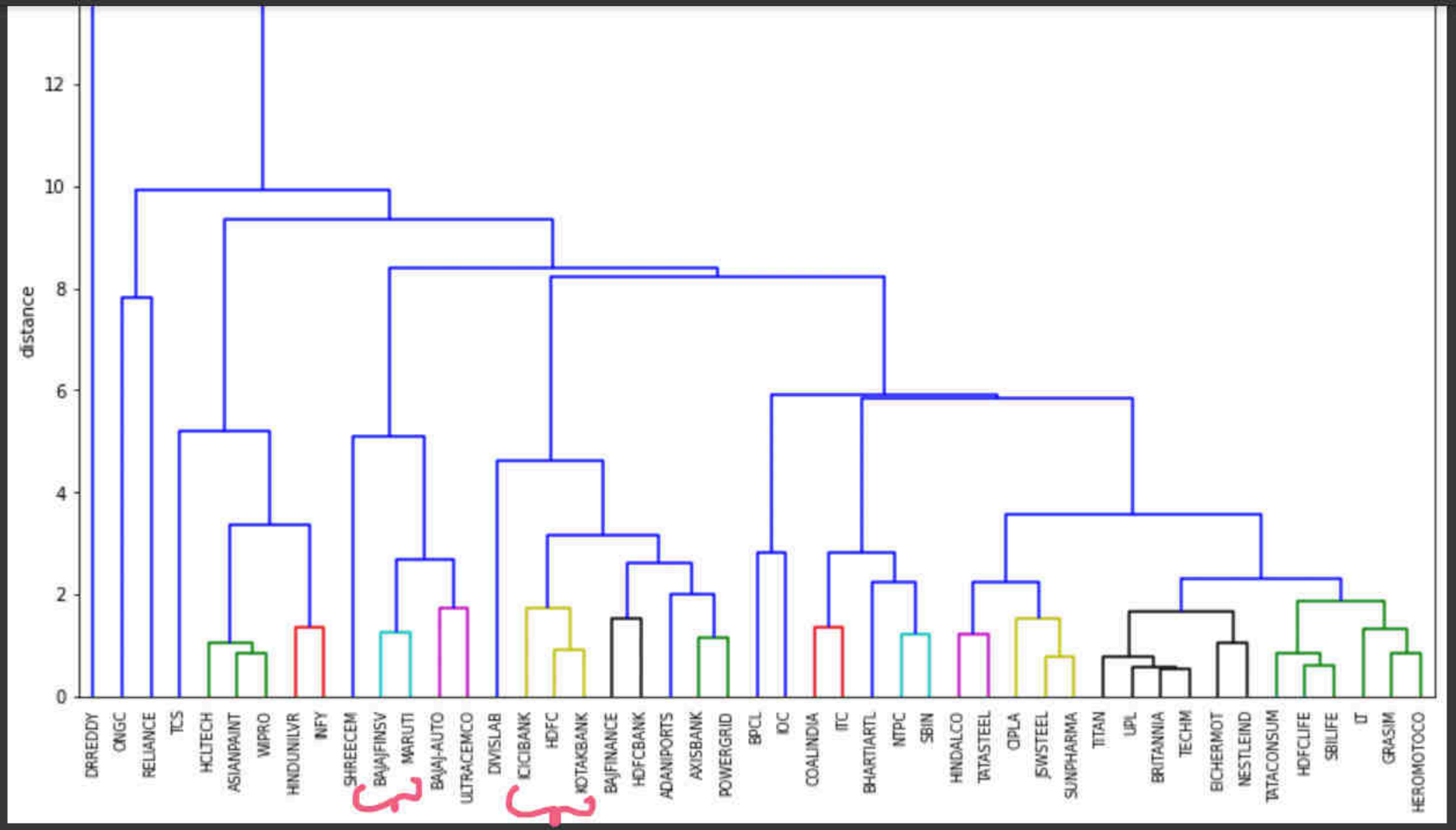
[ ]




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


     # create clusters
```




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


     # create clusters
```



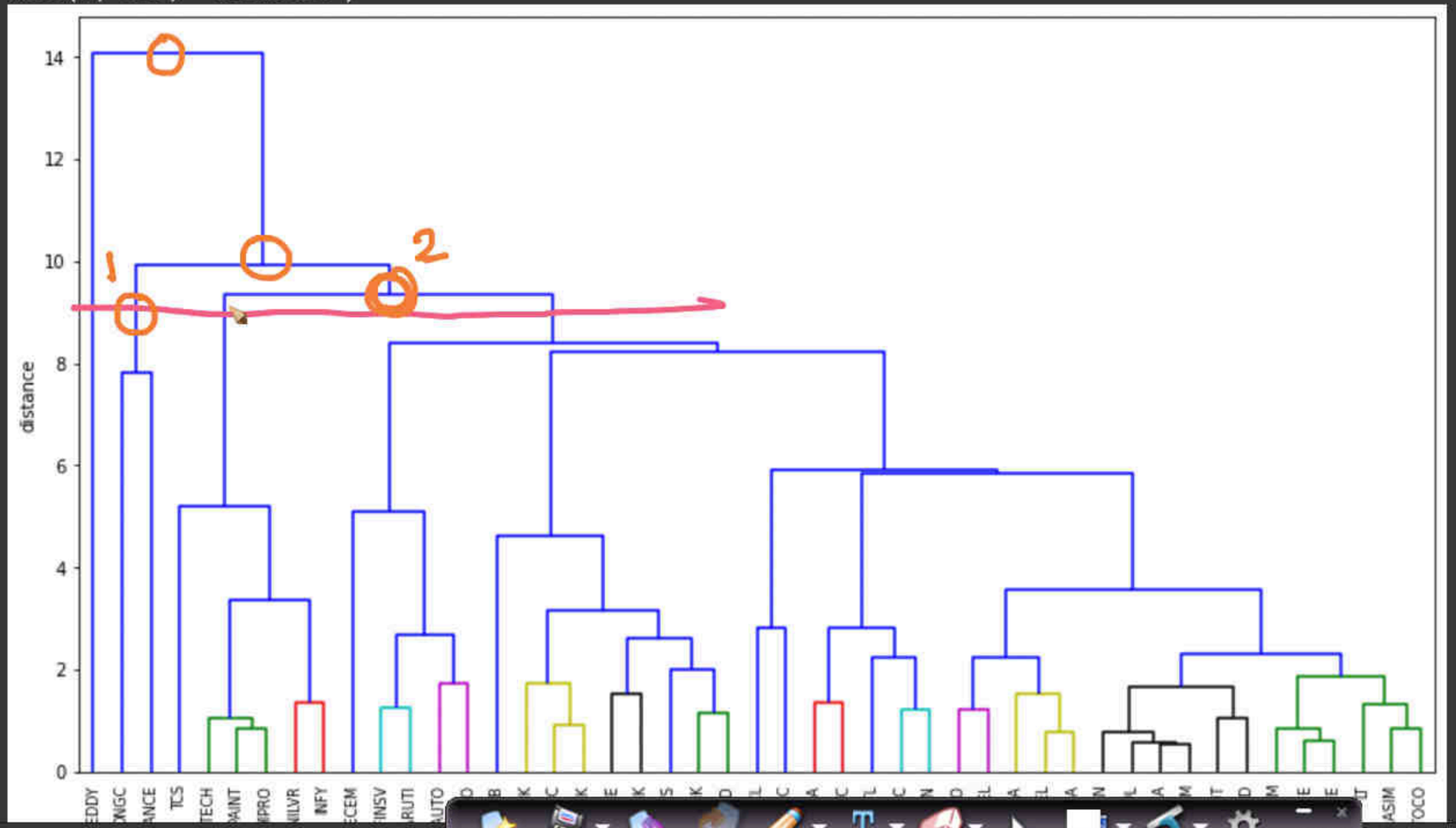
financial metric

8-dim

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


     # create clusters
```

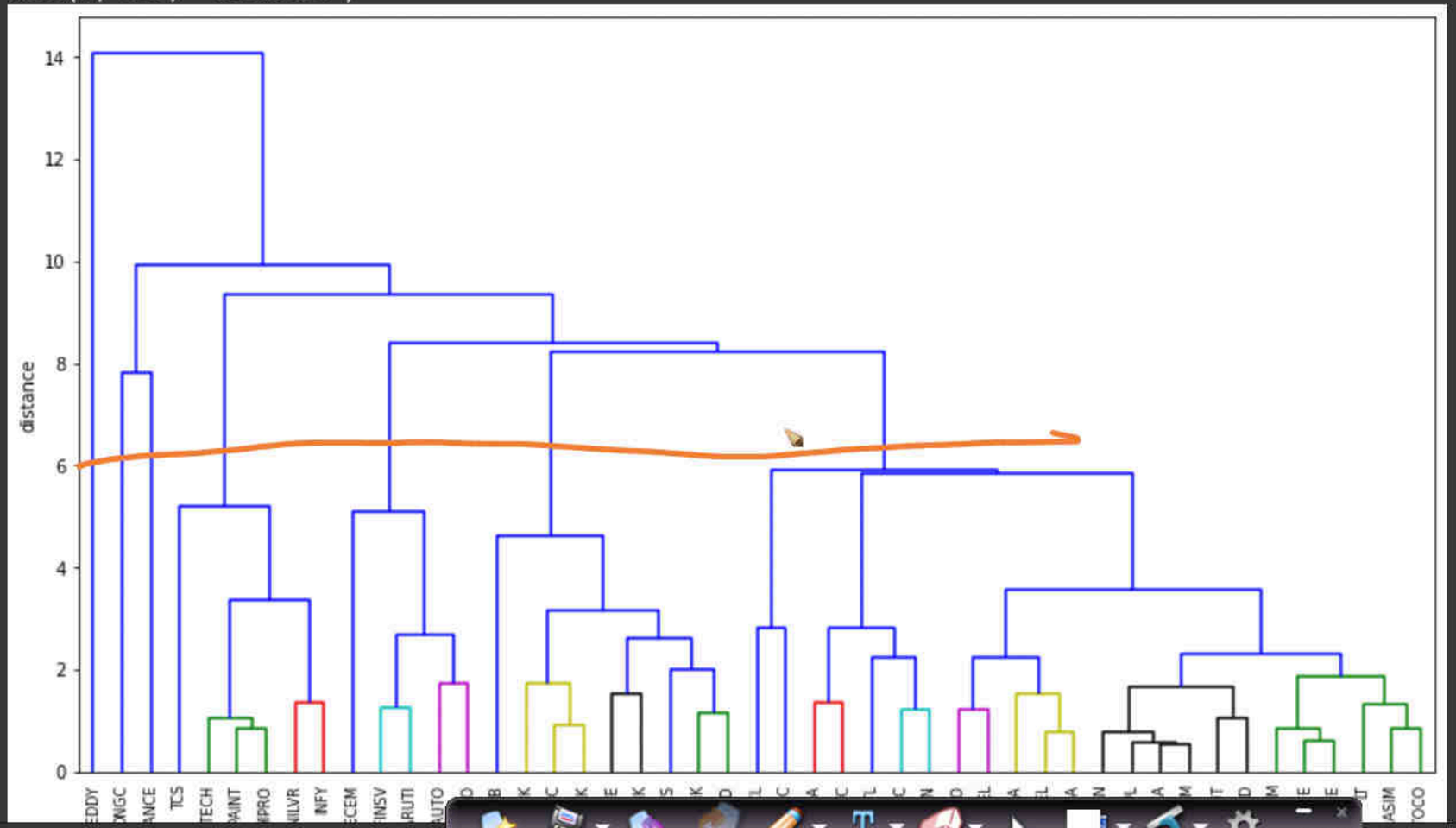
```
fig, ax = plt.subplots(figsize=(14, 8))
sch.dendrogram(Z, labels=scaled_df.index, ax=ax, color_threshold=2)
plt.xticks(rotation=90)
ax.set_ylabel('distance')
```

```
Text(0, 0.5, 'distance')
```



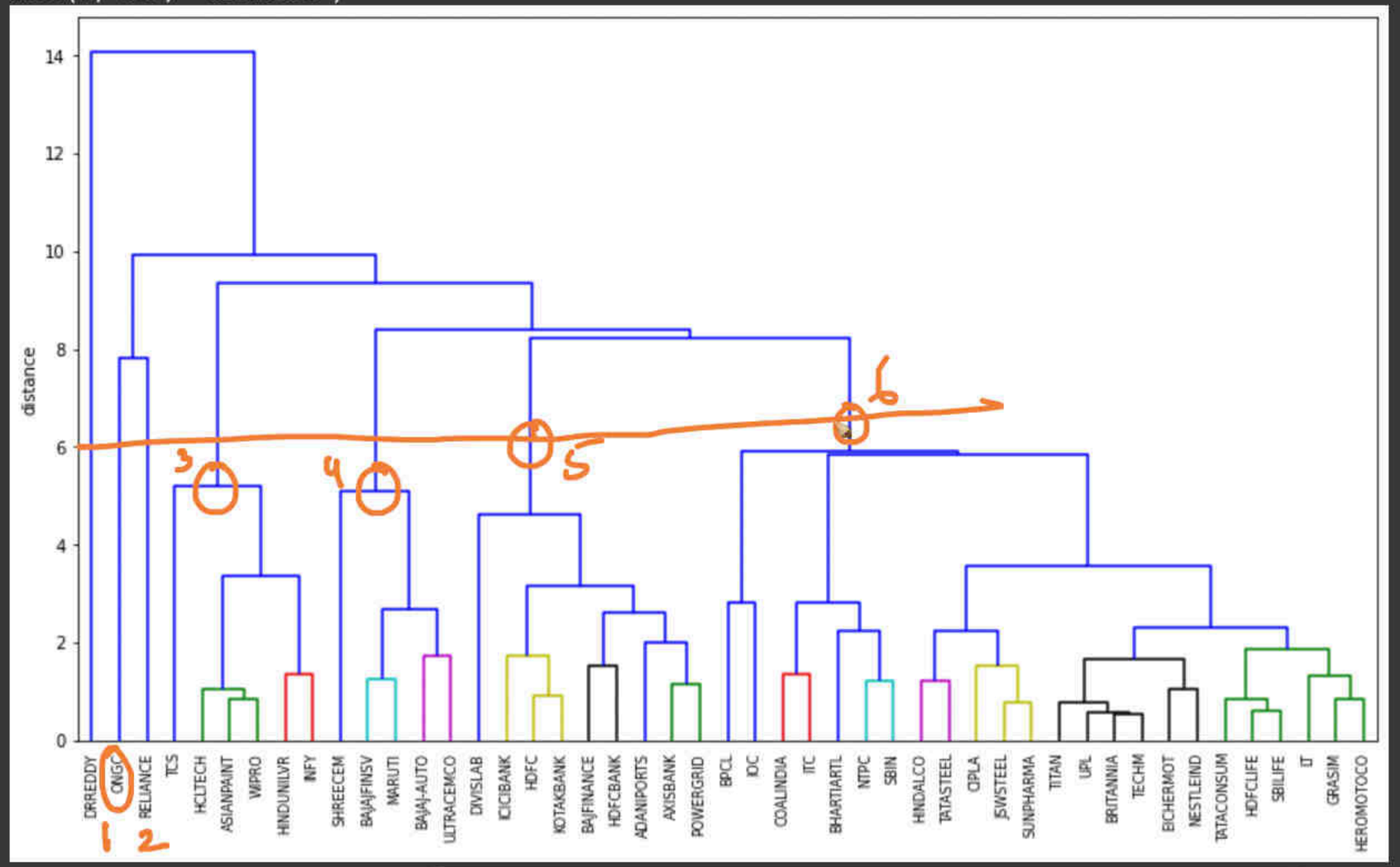
```
fig, ax = plt.subplots(figsize=(14, 8))
sch.dendrogram(Z, labels=scaled_df.index, ax=ax, color_threshold=2)
plt.xticks(rotation=90)
ax.set_ylabel('distance')
```

```
Text(0, 0.5, 'distance')
```



```
ax.set_ylabel('distance')
```

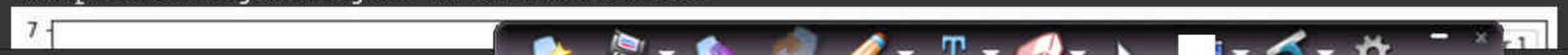
```
Text(0, 0.5, 'distance')
```




```
# 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>
```

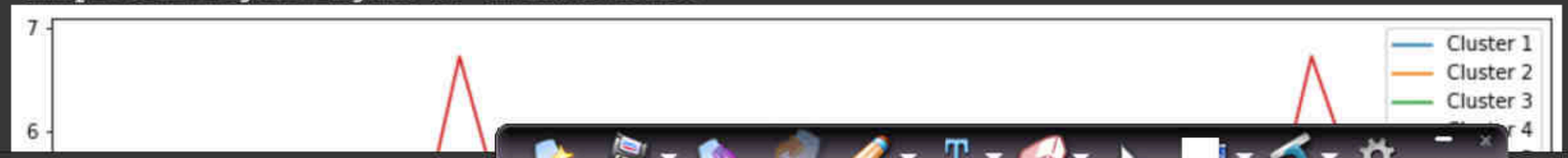


```
# import hierarchical clustering libraries
from sklearn.cluster import AgglomerativeClustering


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

y_pred = hc_2020.fit_predict(scaled_df)
```

each cluster
↓
{ mean of each feature

```
#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)
```

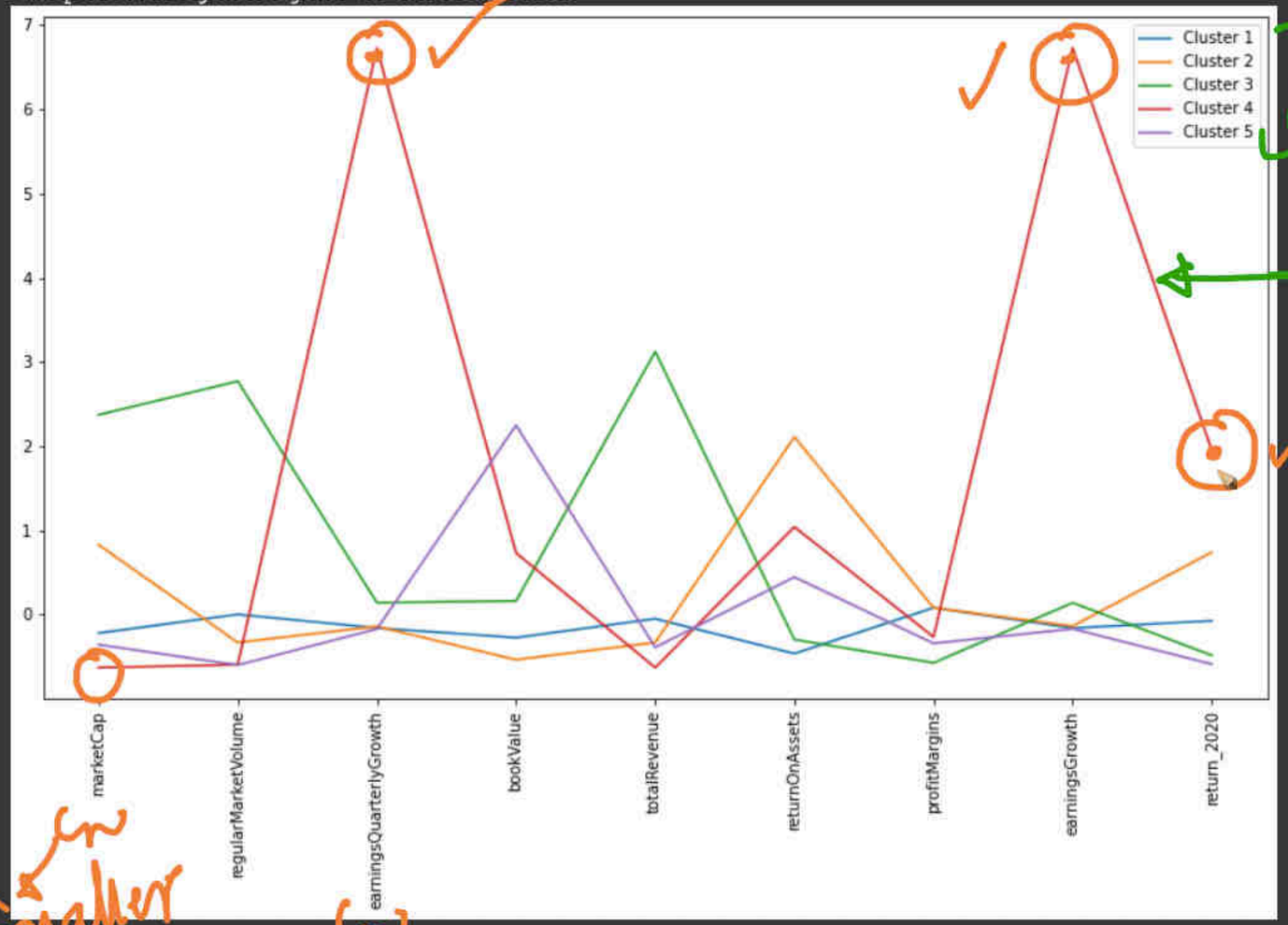
```
<matplotlib.legend.Legend at 0x7f36d94d4490>
```



50 / 50

```
plt.legend(labels)
```

```
<matplotlib.legend.Legend at 0x7f36d9[illegible]d4490>
```

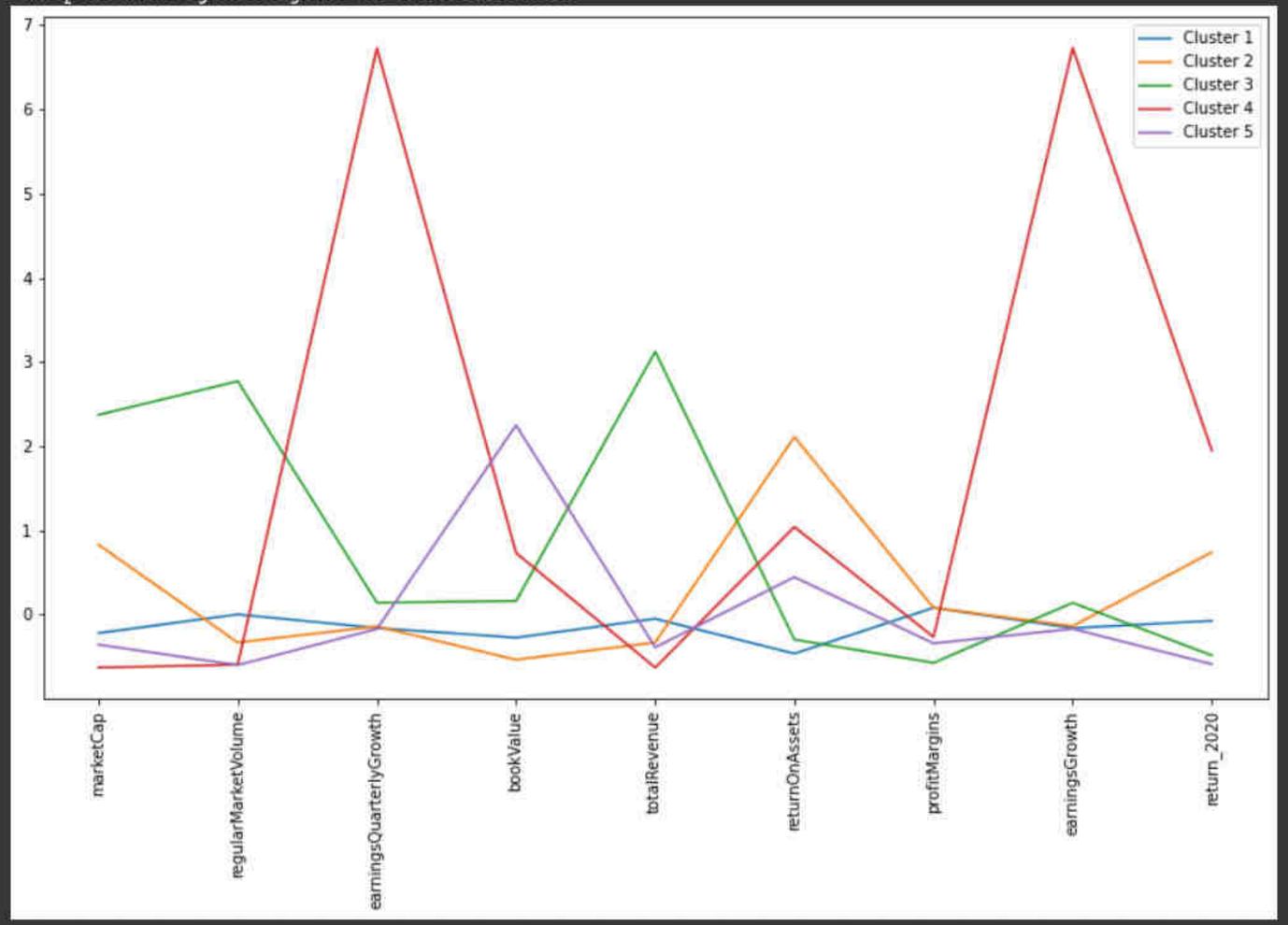


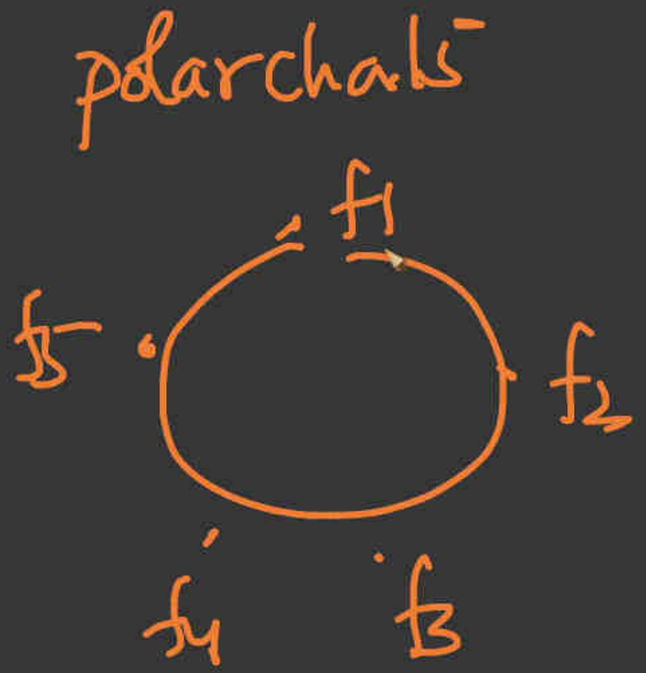
CL 4

Smaller in NIFTY 50

```
plt.legend(labels)
```

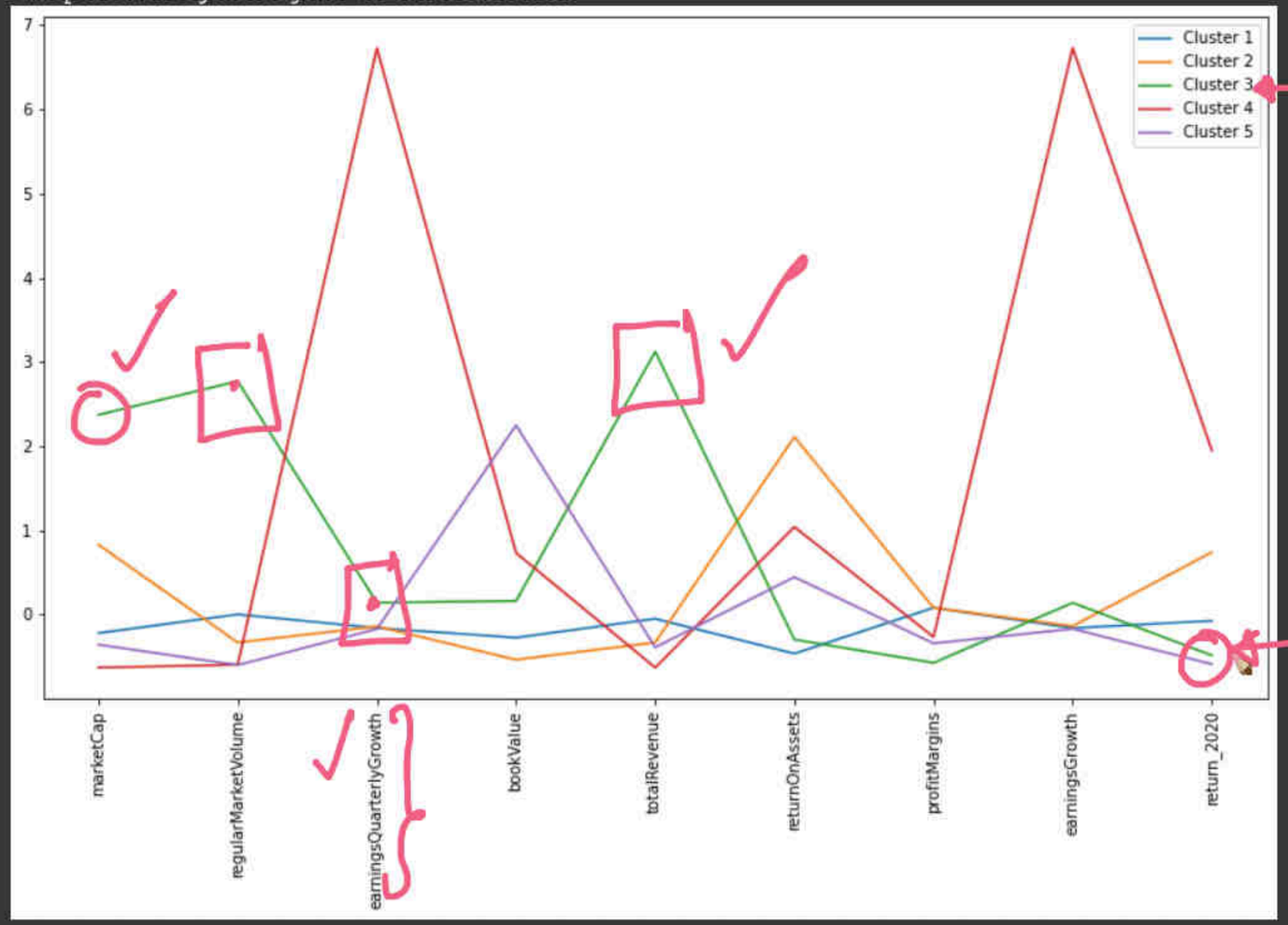
```
<matplotlib.legend.Legend at 0x7f36d94d4490>
```





```
plt.legend(labels)
```

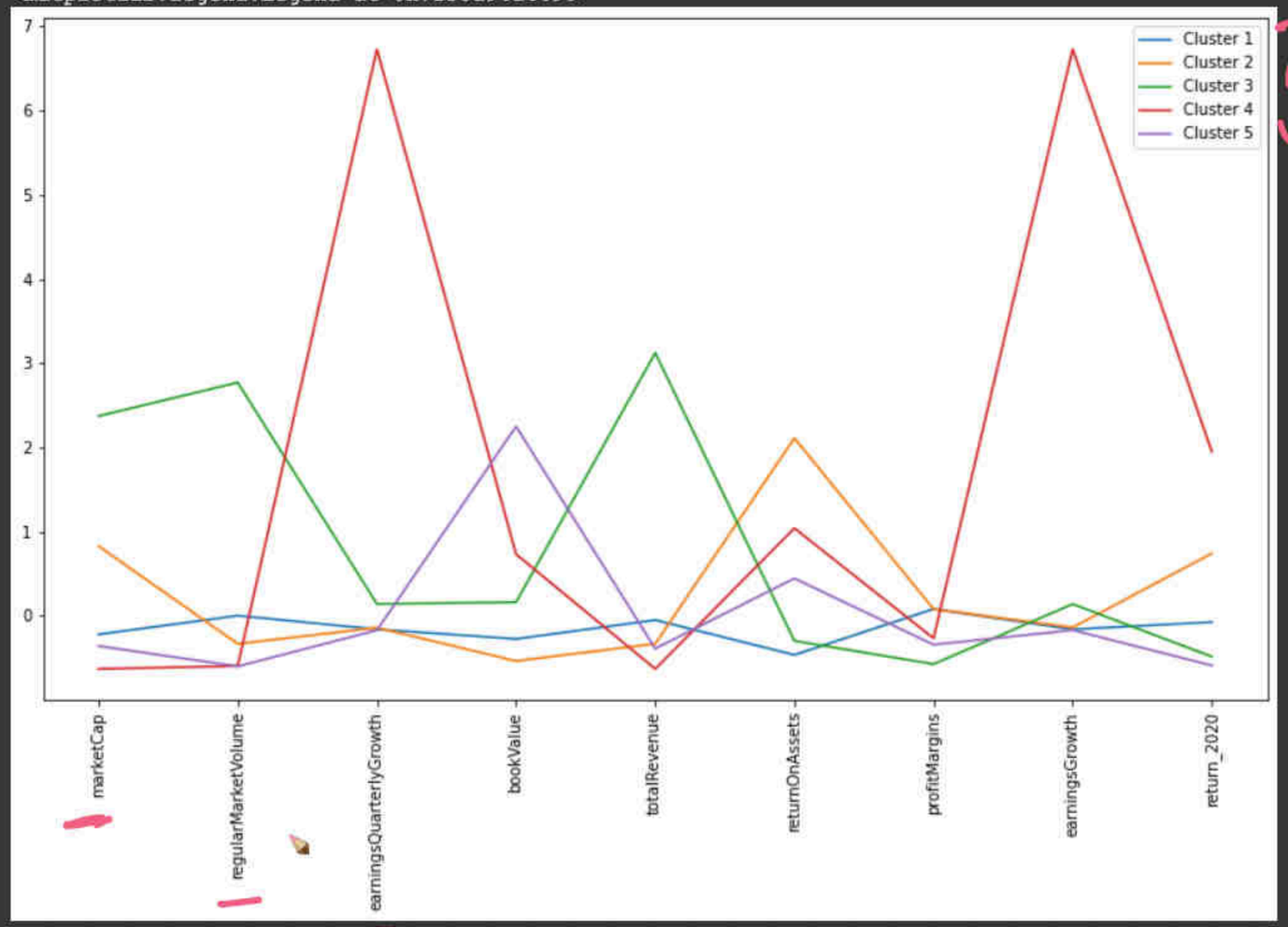
```
<matplotlib.legend.Legend at 0x7f36d94d4490>
```



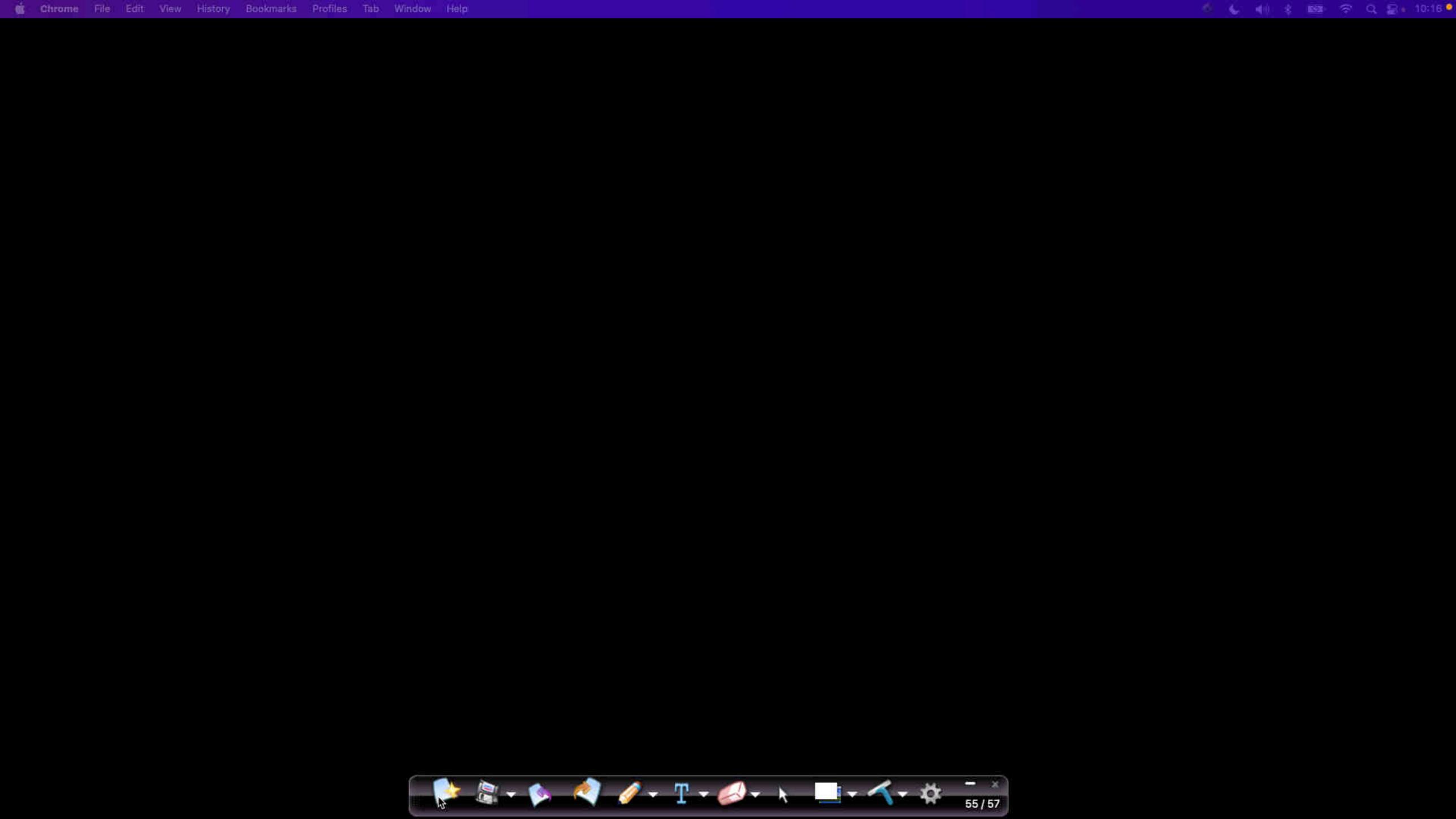

```
[ ]
```

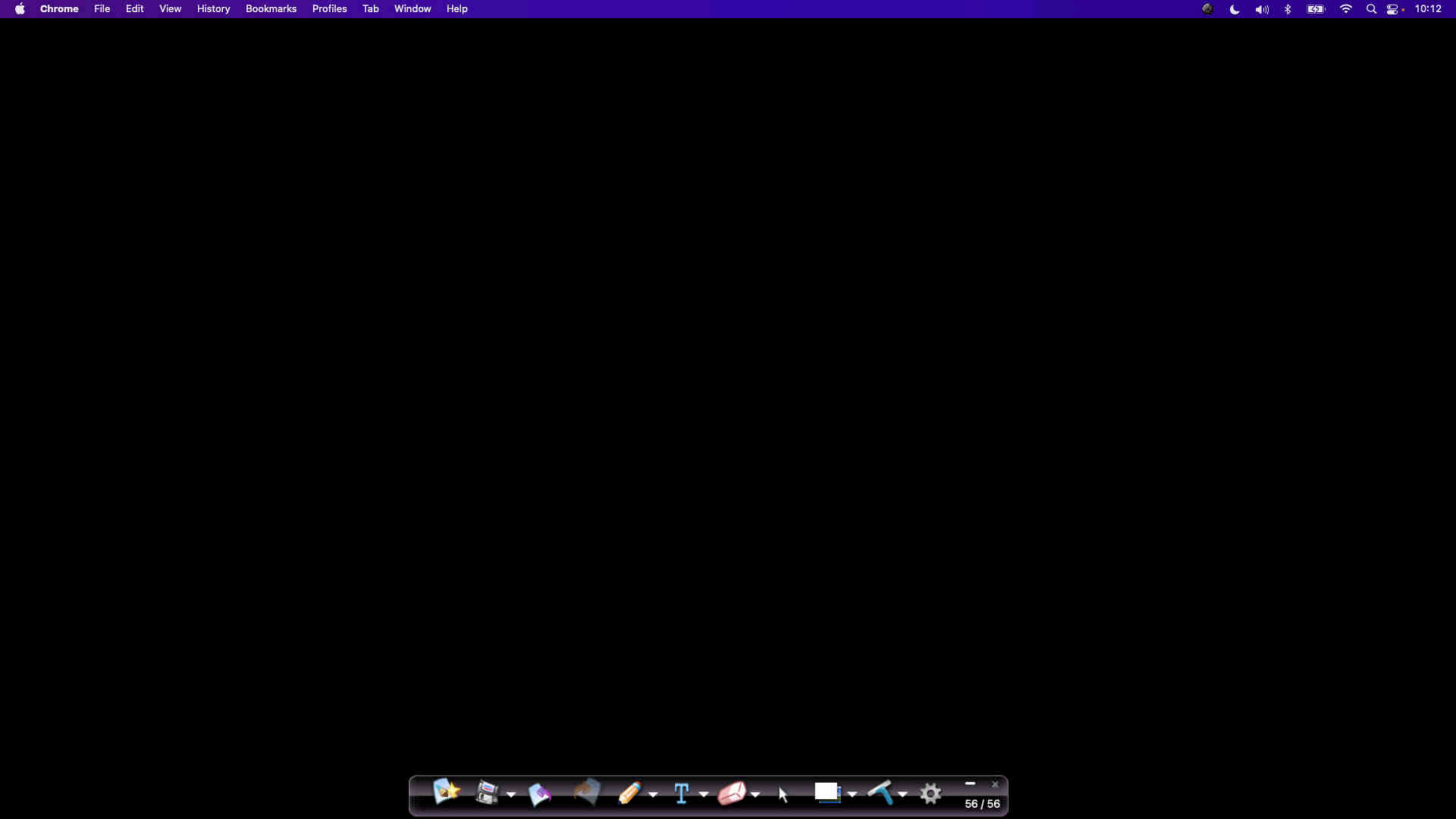
```
plt.legend(labels)
```

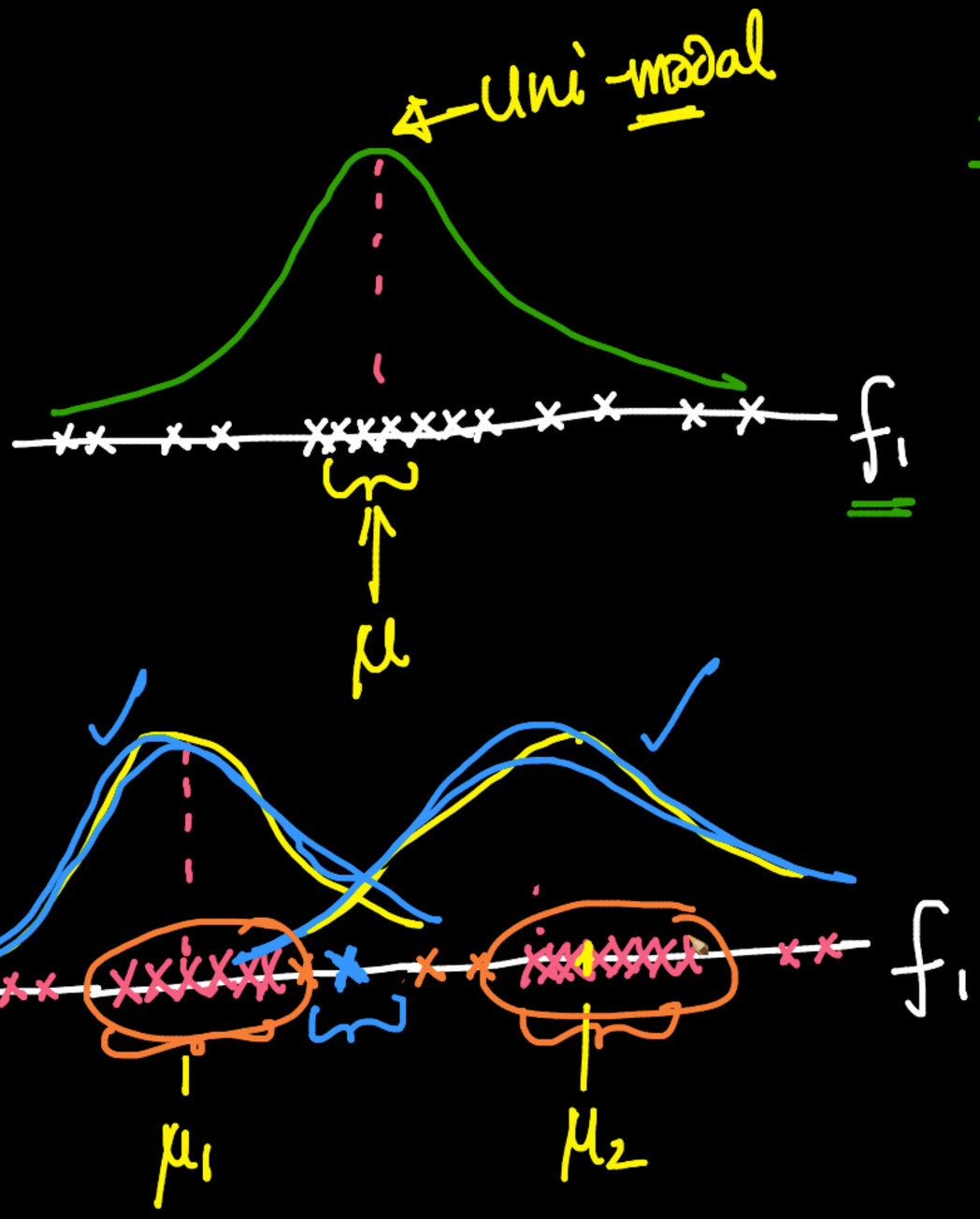
```
<matplotlib.legend.Legend at 0x7f36d94d4490>
```

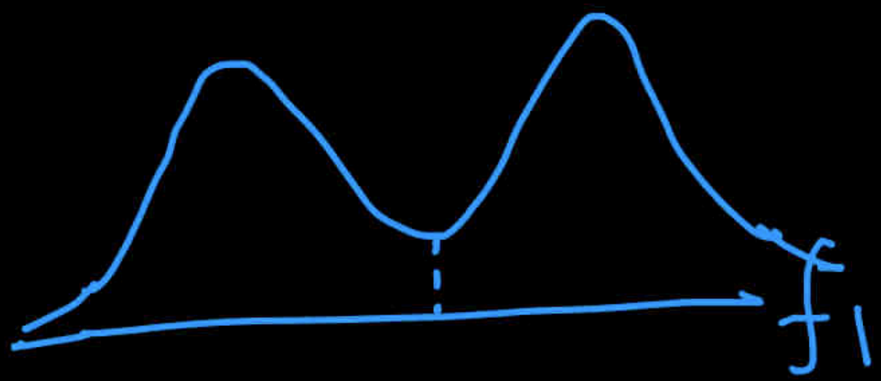



```
[ ]
```

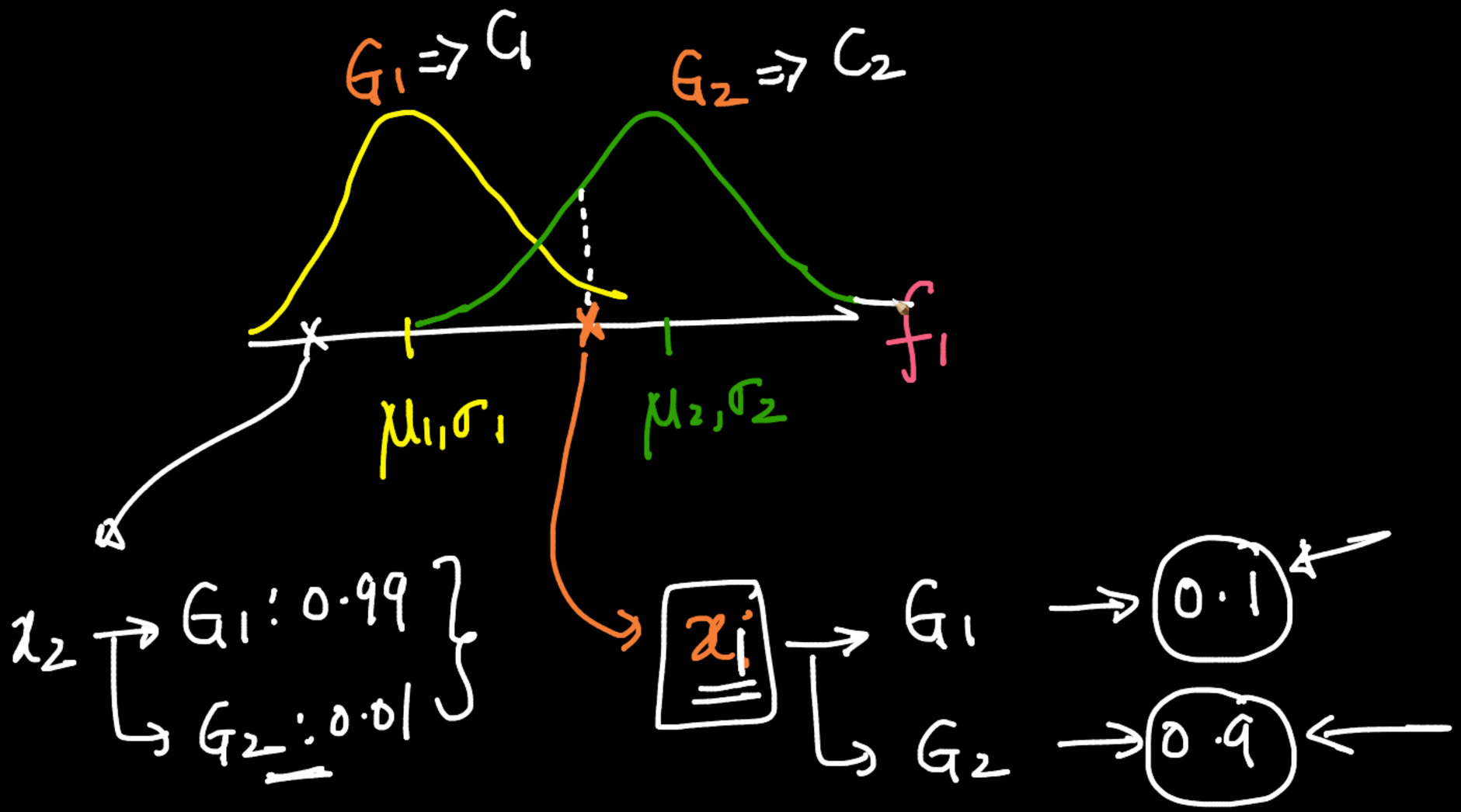

Chrome File Edit View History Bookmarks Profiles Tab Window Help
10:16
55 / 57


Chrome File Edit View History Bookmarks Profiles Tab Window Help
10:12
56 / 56



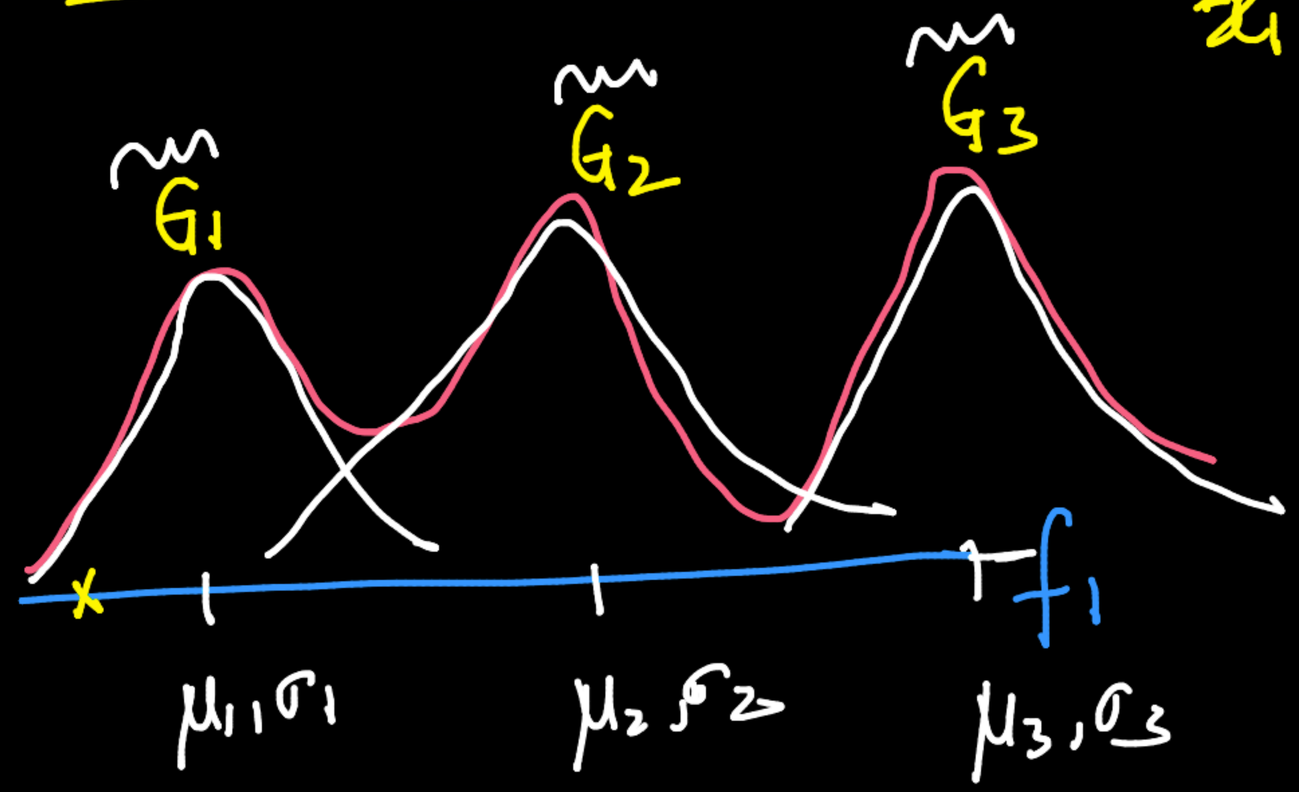


mixture of Gaussians


G1 => C1
G2 => C2
f1
μ1,σ1
μ2,σ2
x2
G1 : 0.99
G2 : 0.01
xi
G1
G2
0.1
0.9

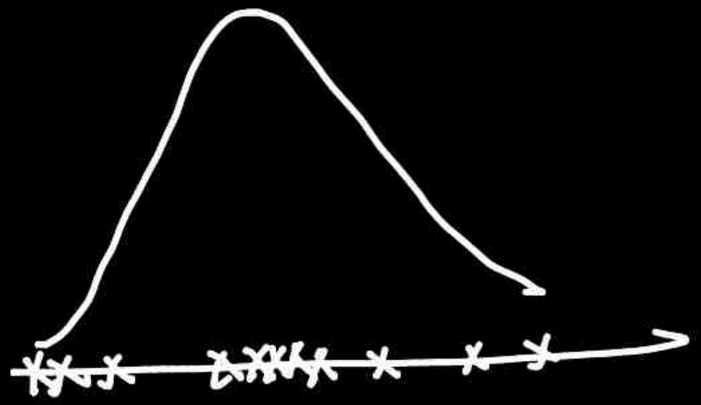
Gaussian Mixture Model

1D - GMM



x_i → G_1: $P(y_i=1)$
G_2: $P(y_i=2)$
G_3: $P(y_i=3)$

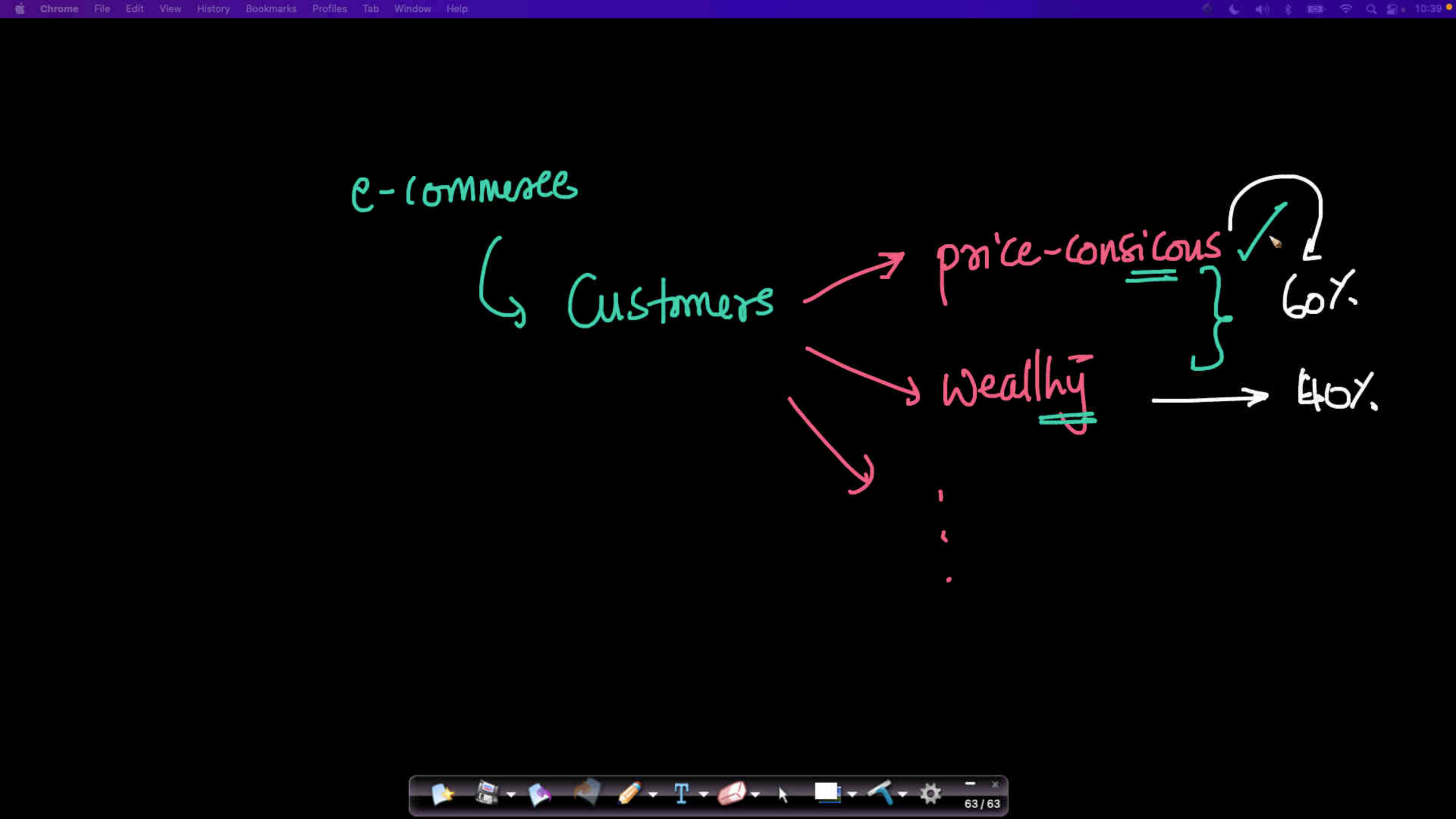
$\sum = 1$

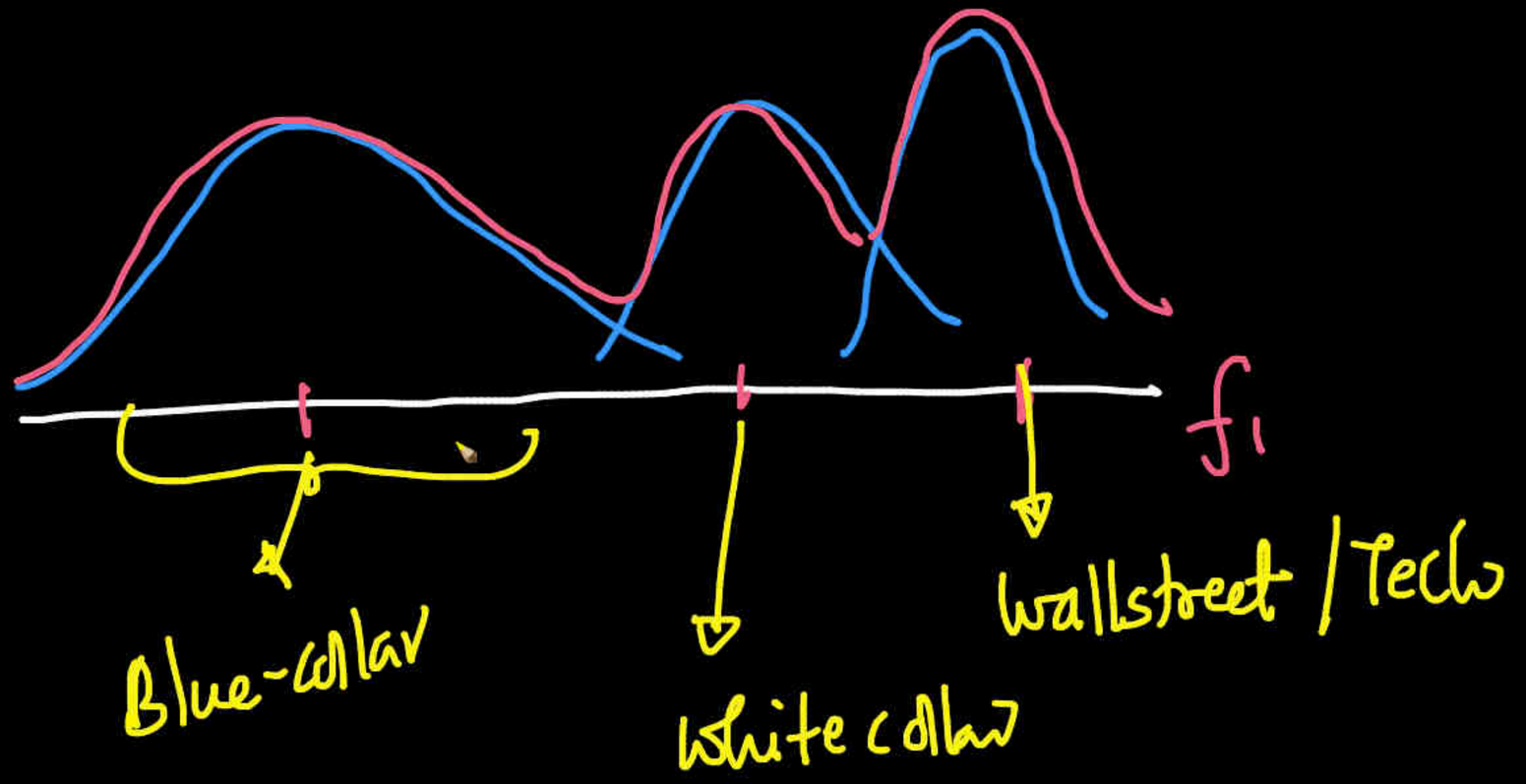


GMM :- Gi ✓

log-normal Mixture-model → Combination of Various dist

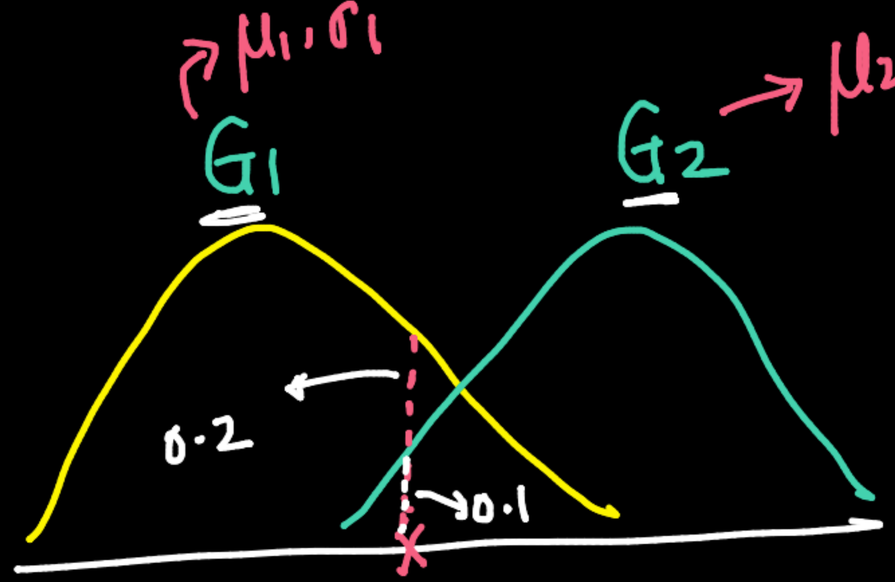
Theoritically →


e-commerce
Customers
price-conscious
wealthy
60%
40%


fi
Blue-collar
White collar
Wallstreet / Tech

GMM:

Soft-assign...



$x_1 \rightarrow G_1 : \frac{0.2}{0.3} = 2/3$

$G_2 : \frac{0.1}{0.3} = 1/3$

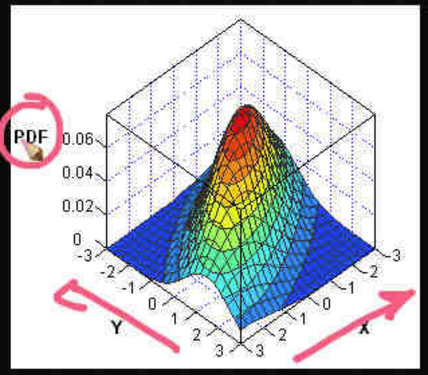
$P(x_1 | G_1)$ → μ_1, σ_1

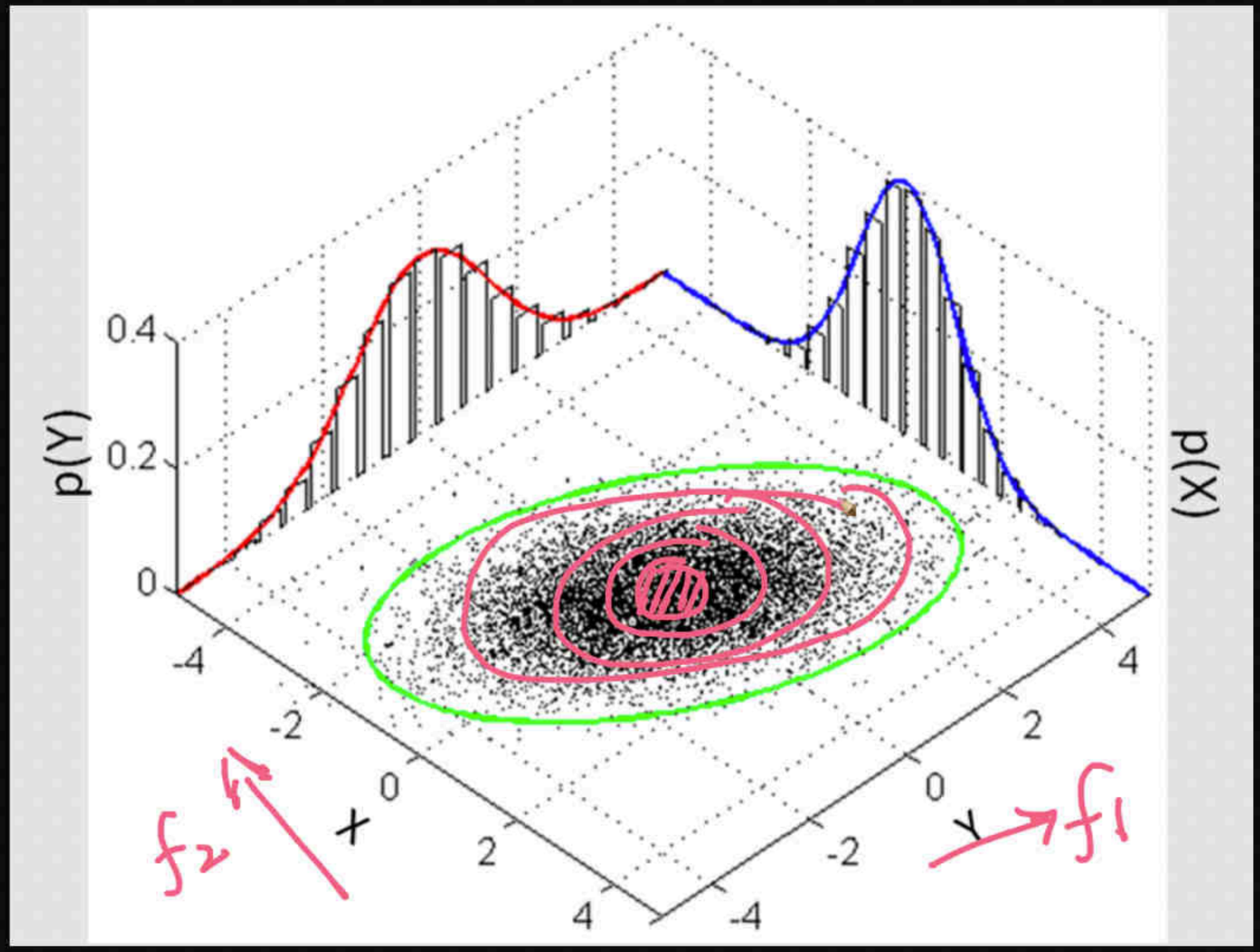
$P(x_1 | G_2)$ → μ_2, σ_2

K-means

$x_i \rightarrow C_1, C_2$

hard assignment

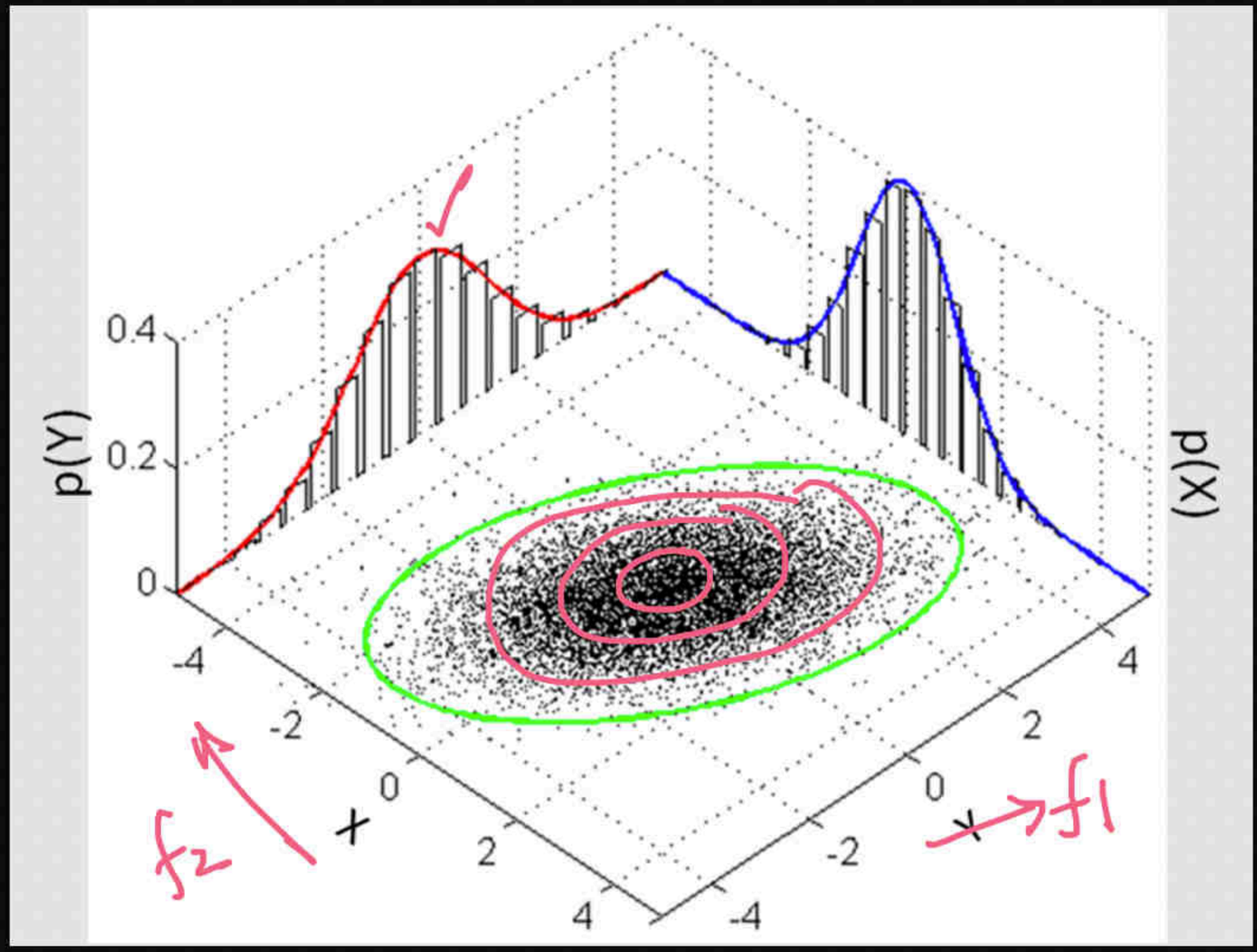



0.4
0.2
0
p(Y)
-4
-2
0
2
4
X
f2
4
2
0
-2
-4
Y
f1
p(X)

d-dim Gaussian

↓

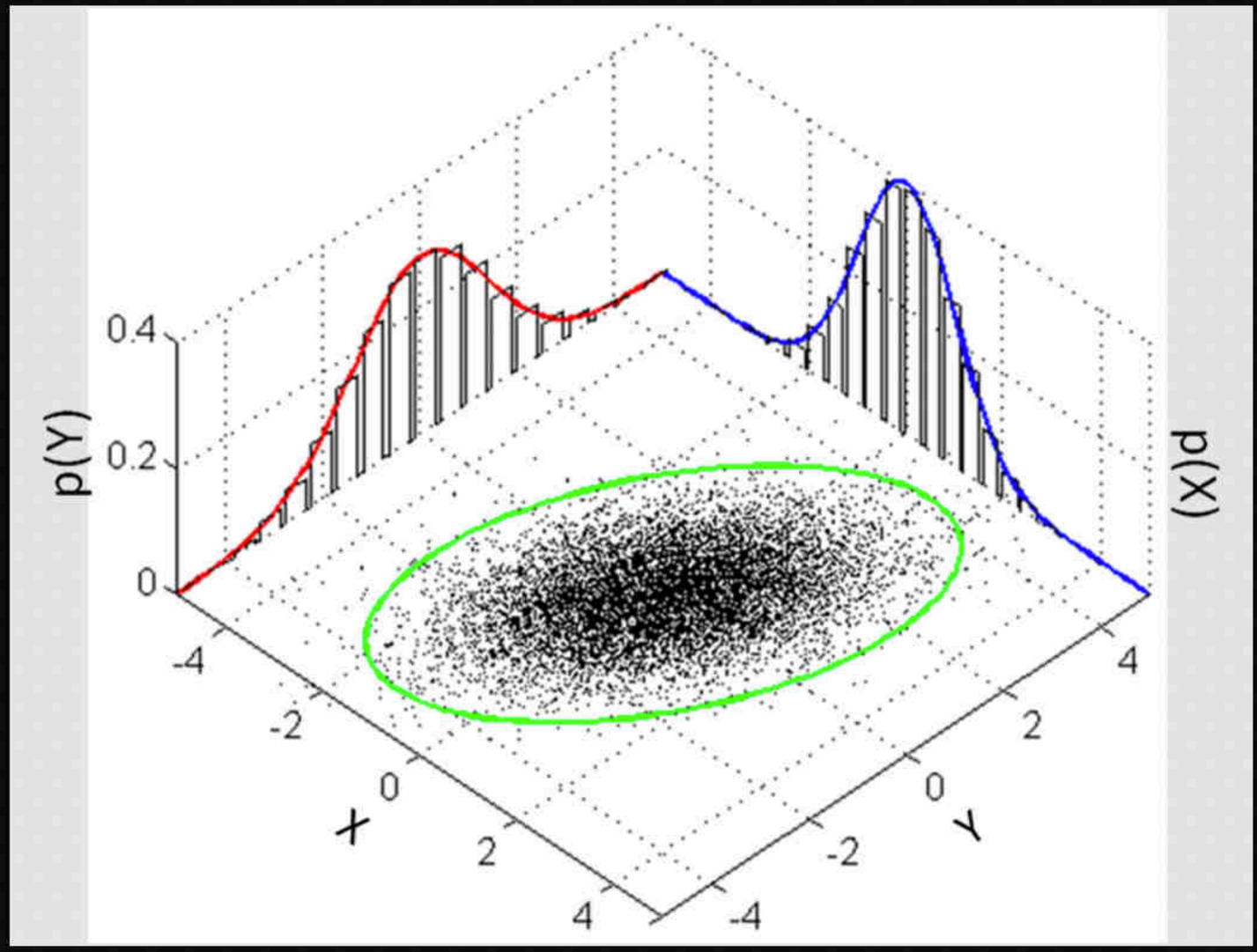
{ hill in d+1-dim space



2D - Gaussian

↓

3D-hill


0.4
0.2
0
p(Y)
-4
-2
0
2
4
X
4
2
0
-2
-4
Y
p(X)

1D:

$N(\mu, \sigma)$

d-dim:

$N_d(\mu, \Sigma)$

μ: d-dim

Σ: d x d

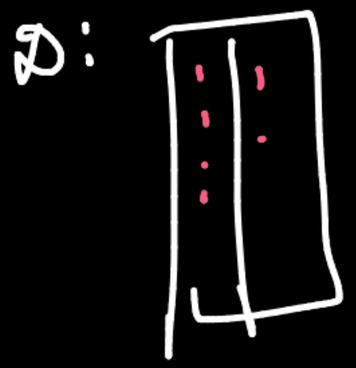
2D:

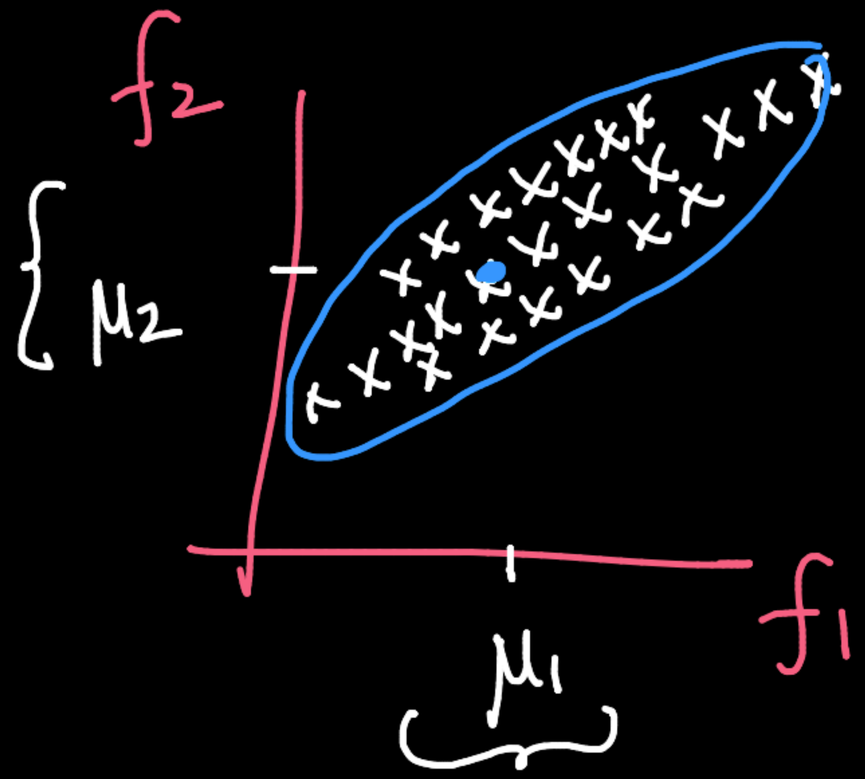
f_1, f_2

$N_2(\mu, \Sigma)$ 2×2

$[\mu_1, \mu_2]$

cov. matrix of 2





Σ

if

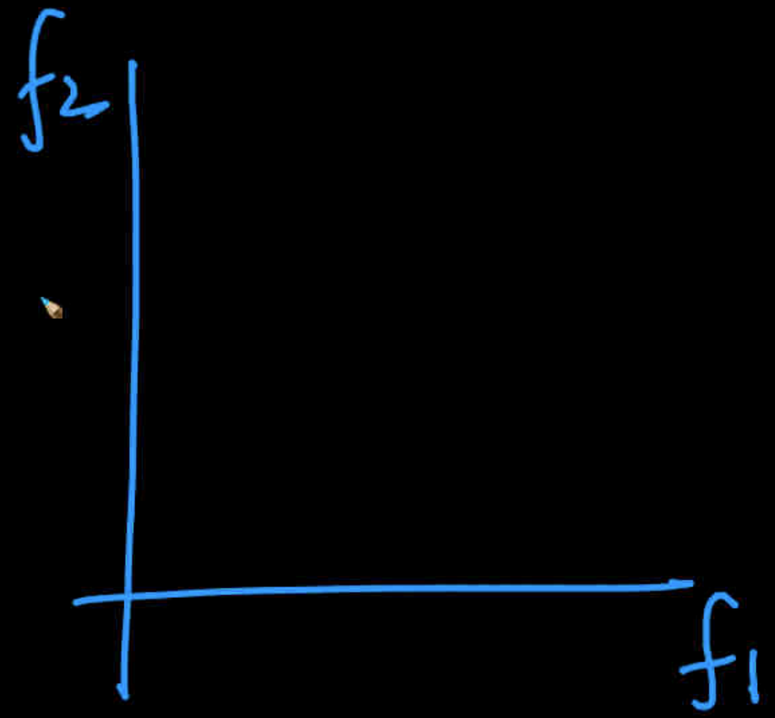
$N_2(\vec{\mu}, \Sigma)$

$[\mu_1, \mu_2]$

$[\sigma_1, \sigma_2]$

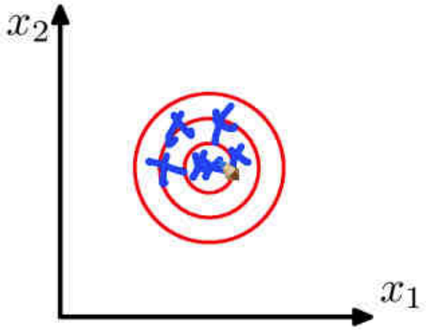
$\Sigma =$

	1	2
1	σ_1	$cov(2,1)$
2	$cov(1,2)$	σ_2


f2
f1

Multivariate Gaussians

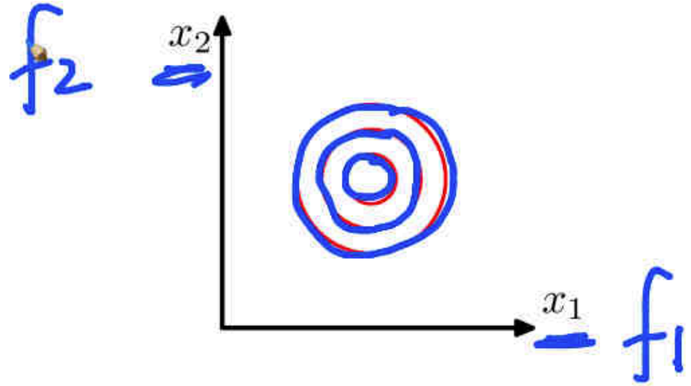
$$P(X=\mathbf{x}_j) = \frac{1}{(2\pi)^{m/2} \|\Sigma\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}_j - \mu)^T \Sigma^{-1} (\mathbf{x}_j - \mu)\right]$$



$\Sigma \propto$ identity matrix

Multivariate Gaussians

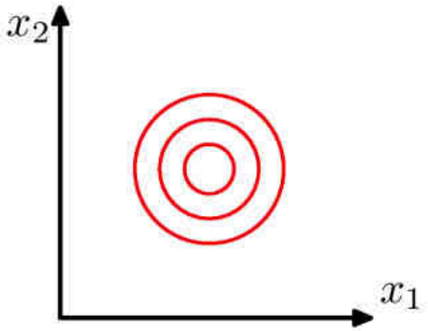
$$P(X=\mathbf{x}_j) = \frac{1}{(2\pi)^{m/2} \|\Sigma\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}_j - \mu)^T \Sigma^{-1} (\mathbf{x}_j - \mu)\right]$$



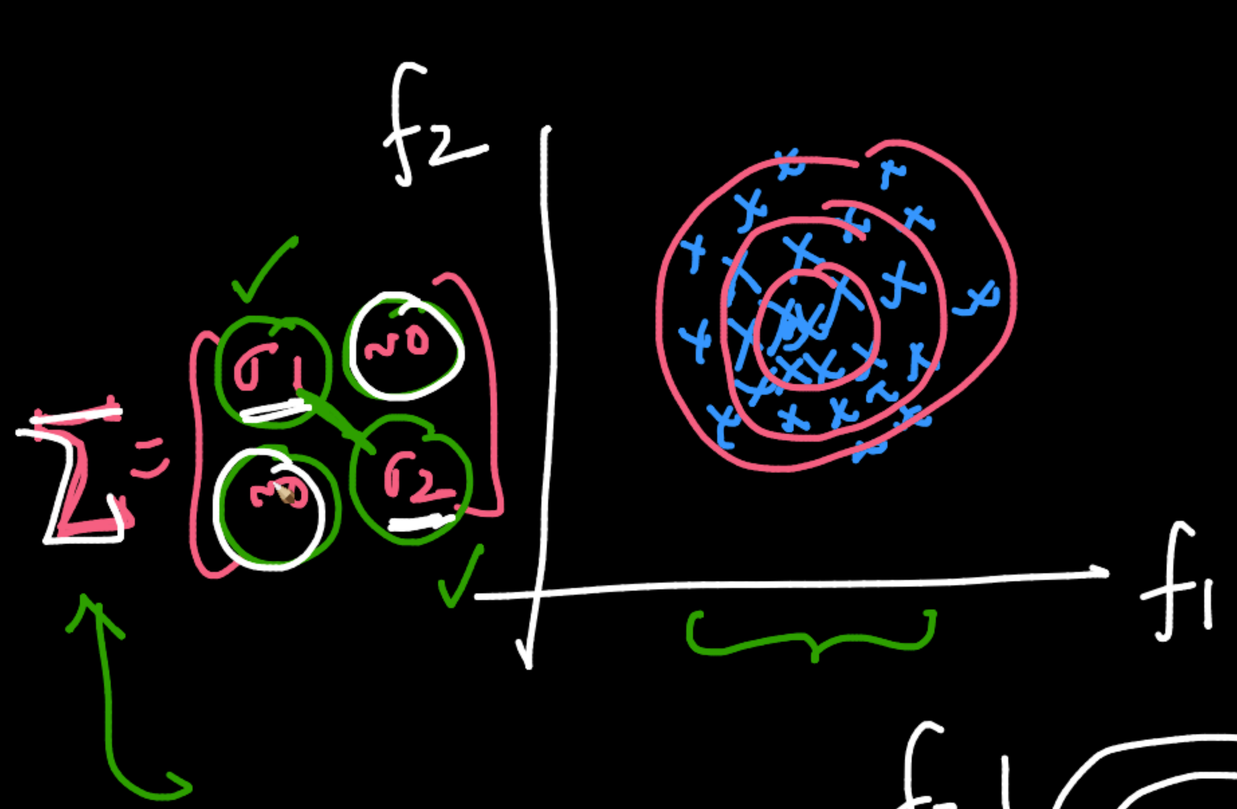
$\Sigma \propto$ identity matrix

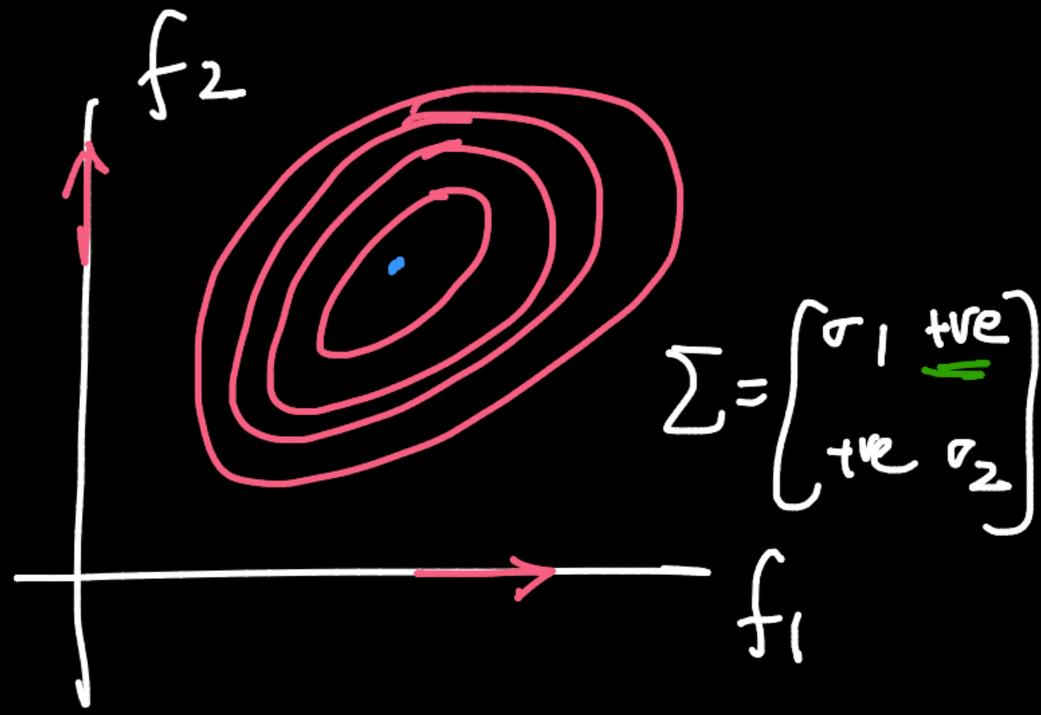
Multivariate Gaussians

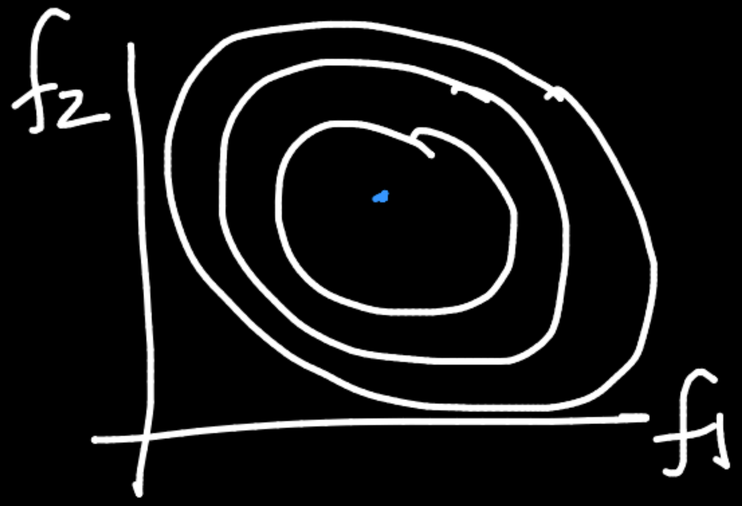
$$P(X=\mathbf{x}_j) = \frac{1}{(2\pi)^{m/2} \|\Sigma\|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}_j - \mu)^T \Sigma^{-1} (\mathbf{x}_j - \mu)\right]$$



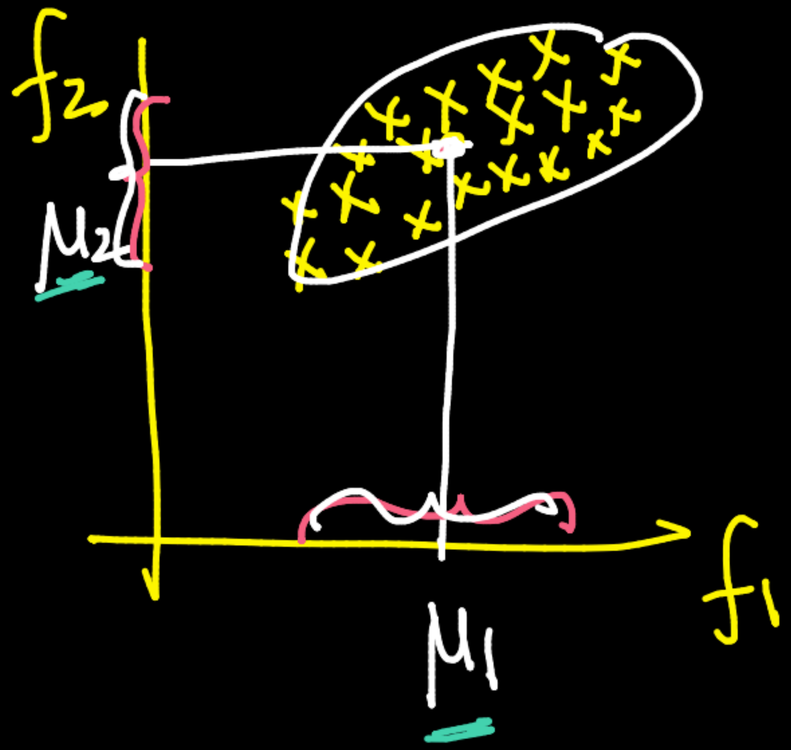
$\Sigma \propto$ identity matrix







$$\Sigma = \begin{bmatrix} \sigma_1 & -ve \\ -ve & \sigma_2 \end{bmatrix}$$



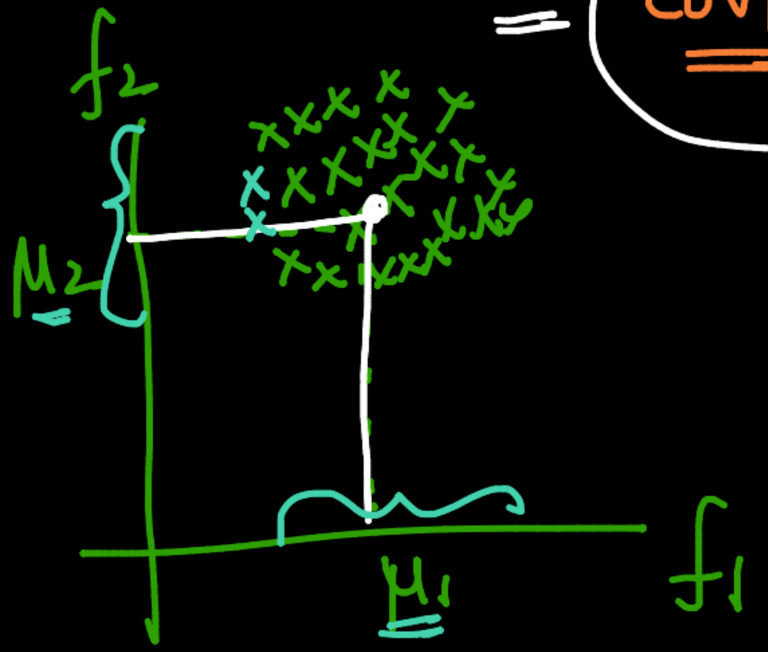
✓ $\mu = [\mu_1, \mu_2]$

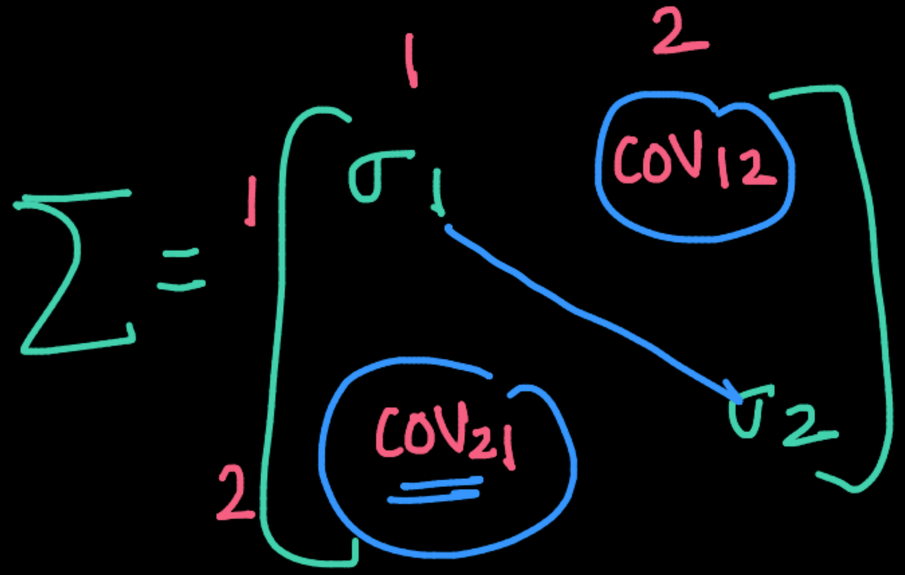
✓ $\underset{2\times 2}{\Sigma} = \begin{bmatrix} \sigma_1 & Cov_{12} \\ Cov_{21} & \sigma_2 \end{bmatrix}$: Symm - matrix

✓ ✓ $[\mu_1, \mu_2] = \mu_{2\text{-dim}}$

✓ ✓ ✓ σ_1, σ_2

o.g $Cov_{12} = Cov_{21}$





$\Sigma =$

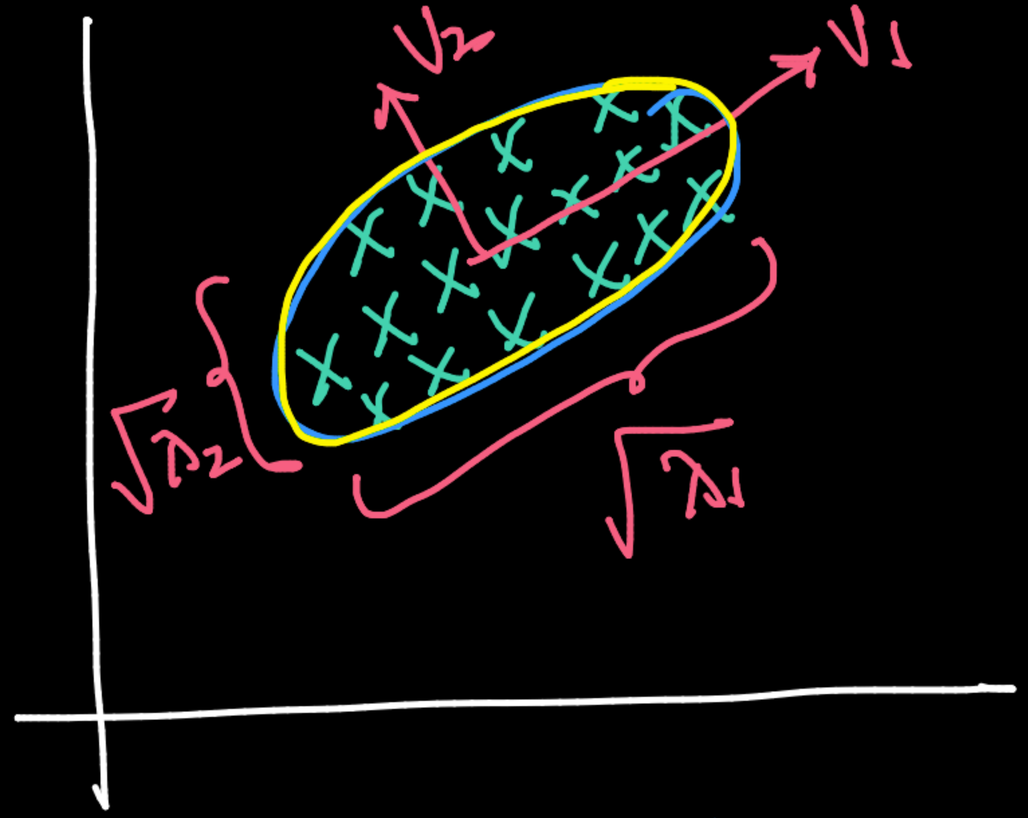
: why Symmetric

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

$Cov(X,Y) = Cov(Y,X)$

$Cov(X,X) = Var(X)$ } PCA

PCA



eig vec of Σ

1 d-dim Gaussian

$\tilde{\Sigma} = \text{cov}(\mathcal{D})$

$\mathcal{D} \in \mathbb{R}^2$

PCA :-

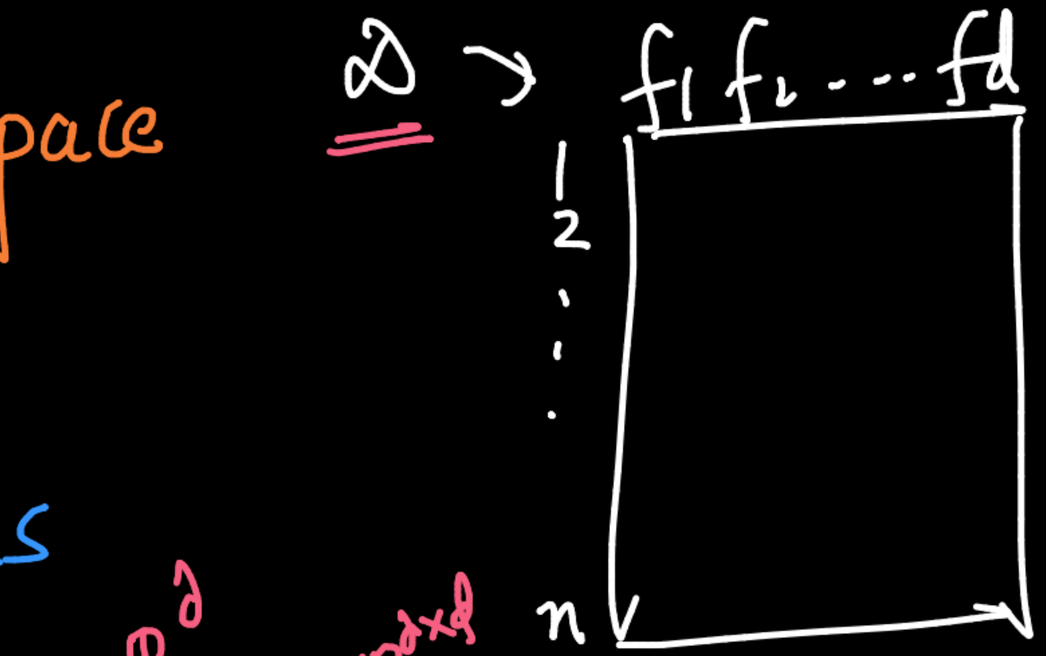
↓

eig vec of $\tilde{S}$

↓

cov($\mathcal{D}$)

d-dim space



3 - Gaussians

Find
- G_1 : μ_1, Σ_1 ($\mu_1 \in \mathbb{R}^d$, $\Sigma_1 \in \mathbb{R}^{d \times d}$)
- G_2 : μ_2, Σ_2
- G_3 : μ_3, Σ_3

Params

x_i → G_1 : p_{i1}
G_2 : p_{i2}
G_3 : p_{i3}

$p_{i1} + p_{i2} + p_{i3} = 1$

1D:

3- Gaussian

G_1: μ_1, σ_1

G_2: μ_2, σ_2

G_3: μ_3, σ_3

pdf of 1D- Gaussian

$x_i = P(x_i \mid G_j)$

$P(x_i \mid G_1) = 0.2/0.8$

$P(x_i \mid G_2) = 0.1/0.8$

$P(x_i \mid G_3 = 0.5/0.8$

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
मराठी
Português
தமிழ்
اردو
中文
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.

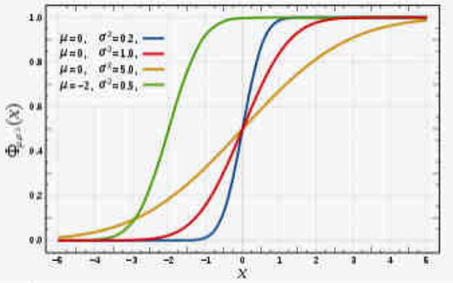
A normal distribution is sometimes informally called a **bell curve**.[5] However, many other distributions are bell-shaped (such as the Cauchy, Student's *t*, and logistic distributions).

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

Contents [hide]

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	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$
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. kurtosis	3
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

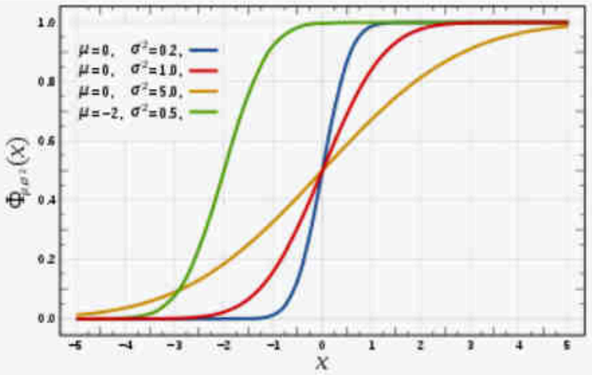
(and also its median and mode), while the parameter σ is its standard deviation. The variance of the distribution is σ^2.[1] A random variable with a Gaussian distribution is said to be **normally distributed**, and is called a **normal deviate**.

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

(μ,σ)

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

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

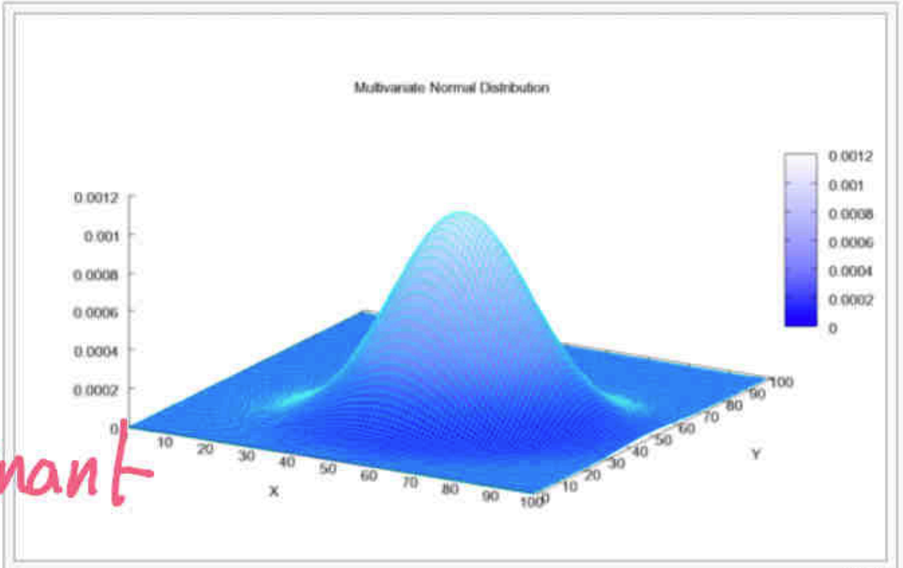
$P(x_i \mid \text{Normal } d) = \ldots$

Non-degenerate case [edit]

The multivariate normal distribution is said to be "non-degenerate" when the symmetric covariance matrix $\boldsymbol{\Sigma}$ is positive definite. In this case the distribution has density[5]

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

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).



Bivariate normal joint density

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 $\sqrt{(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$ 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

recap:

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

$P(x_i | G_j)$ → 1D dim PDF function

→ find k - Gaussians

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

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

μ_i: d-dim

Σ_i: $d \times d$ matrix

→ find $\underline{k\text{-Gaussians}}$

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

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

μ_i ↓ d-dim

Σ_i ↓ dxd matrix

→ find $\underline{k\text{-Gaussians}}$

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

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

μ_i ↓ d-dim

Σ_i ↓ dxd matrix

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

→ find

K - Gaussians

$\tilde{\mu}_i, \tilde{\Sigma}_i \quad \forall i : 1 \to k$

↓ d-dim

↓ dxd matrix

Probabilistic

Optimization → EM (coordinate ascent)

↓ GD

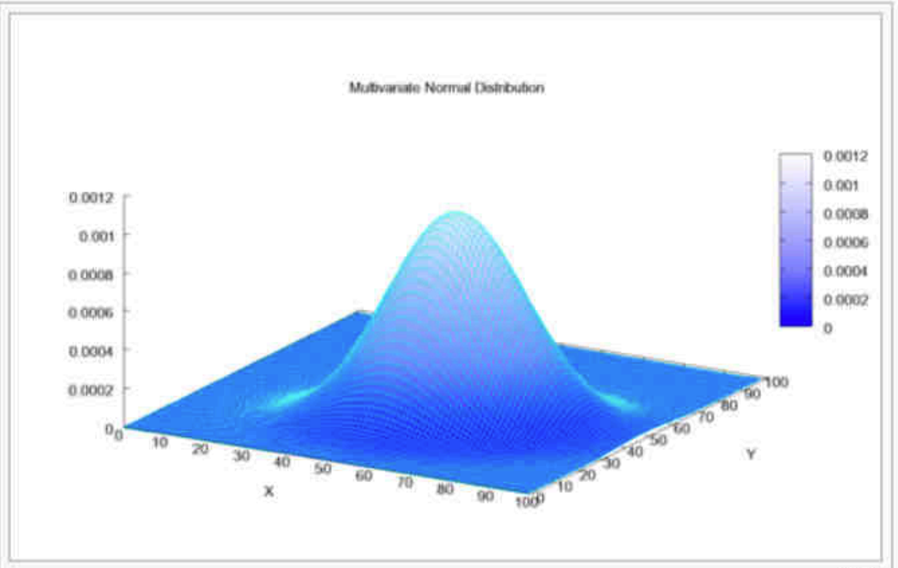
↓ code

Non-degenerate case [edit]

The multivariate normal distribution is said to be "non-degenerate" when the symmetric covariance matrix $\boldsymbol{\Sigma}$ is positive definite. In this case the distribution has density[5]

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

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).



Bivariate normal joint density

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 $\sqrt{(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$ 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