Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the participants individually and don't necessarily reflect those of blibli.com.

Blibli.com does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Python, data science, and unsupervised learning

Hendri Karisma

hendri.karisma@gdn-commerce.com / situkangsayur@gmail.com

Hendri Karisma

- Sr. Research and Development Engineer at blibli.com (PT. Global Digital Niaga)
- Rnd Team for Machine Learning
- Working for Fraud Detection System. Current working in dynamic recommendation system project.

Definition of Informatics

"Automation of Information" –

Prof. Dr. Ing. Iping Supriana

Solution Approachment

- Analytical (Exact)

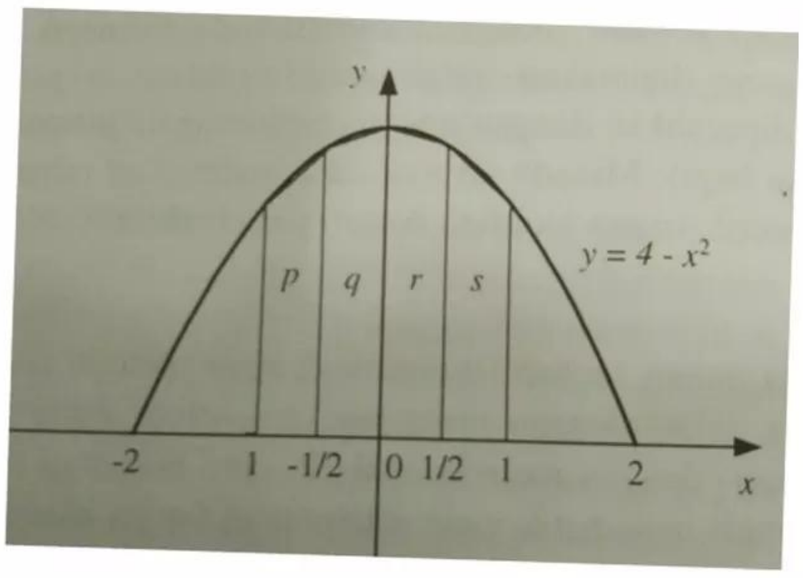
  Example :

  - analytics solution :

$$I = \int_{-1}^{1} (4 - x^2)\,dx = [\,4x - x^3/3\,]_{x=-1}^{x=1} = \{4(1) - (1)/3\} - \{4(-1) - (-1)/3\} = 22/3$$

  - Numerical solution

$$\begin{aligned} I &\approx p + q + r + s \\ &\approx \{[f(-1) + f(-1/2)] \times 0.5/2\} + \{[f(-1/2) + f(0)] \times 0.5/2\} + \\ &\quad \{[f(0) + f(1/2)] \times 0.5/2\} + \{[f(1/2) + f(1)] \times 0.5/2\} \\ &\approx 0.5/2\ \{f(-1) + 2f(-1/2) + 2f(0) + 2f(1/2) + f(1)\} \\ &\approx 0.5/2\ \{3 + 7.5 + 8 + 7.5 + 3\} \\ &\approx 7.25 \end{aligned}$$



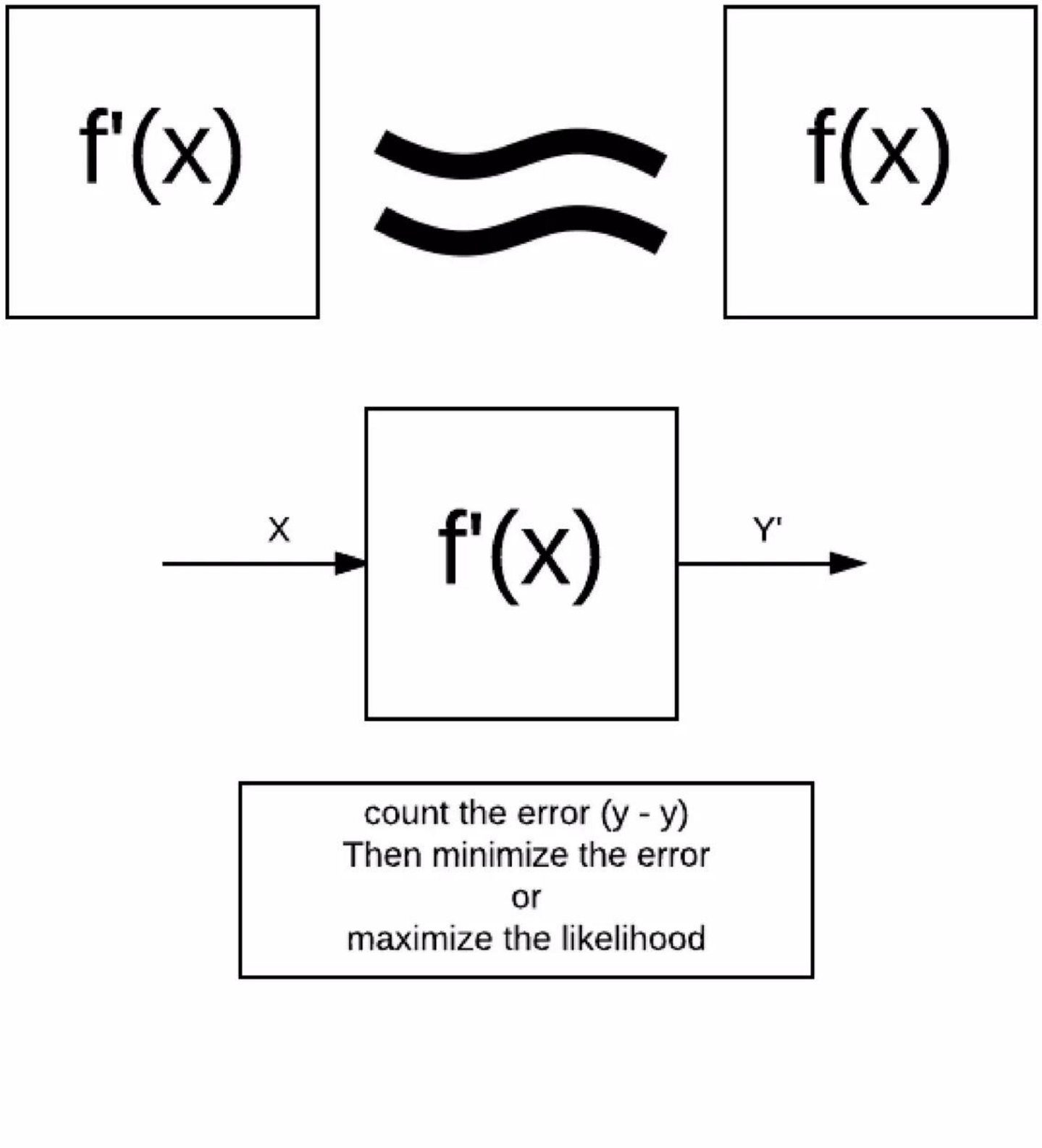
  - Error = | 7.25 – 22/3| = |7.25-7.33|=0.08333

- Numerical (Aprox)
  - Is numerical methods just about ML method that we know in the book?
  - Newton raphson, Gauss Elimination, Gauss-Jordan, Jacobi method, Gauss-Seidel, Lagrange, Newton Gregory, Richardson Interpolation, etc.

Machine Learning Definition

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.” – Prof. Tom Mitchel

How it works

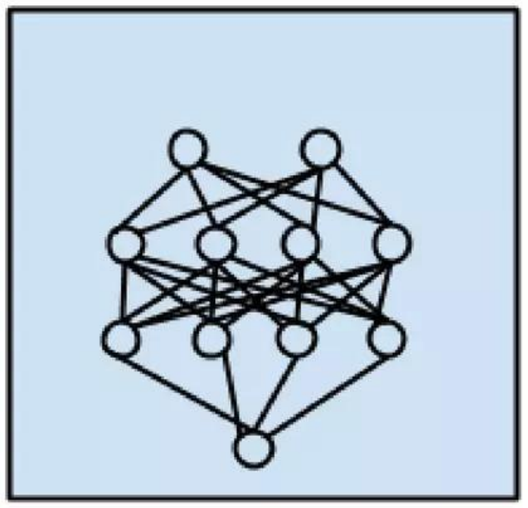

f'(x)
f(x)
X
f'(x)
Y'
count the error (y - y)
Then minimize the error
or
maximize the likelihood

Machine Learning Perspective

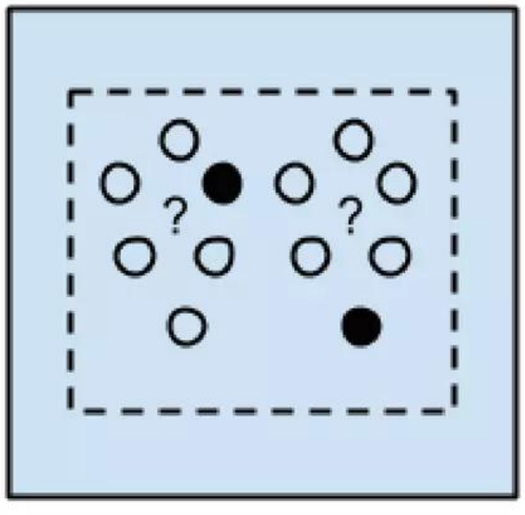
- Information Theory (Decission Tree : ID-Tree, C4.5, etc)
- **Probability (Bayessian : Naive Bayes, Belief Network, etc)**
- Graphical Model (Belief network, HMM, CRF, Neural Network, etc)
- Numerical Method or Regression (Stochastic Gradient Descent/Ascent: Linear Regression, Multiple Linear Regression, Neural Network, **E-M Algorithm**, HMM)

Machine Learning

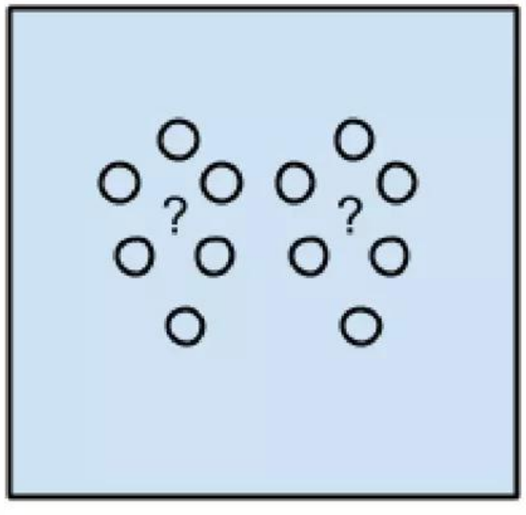
- Supervised
- **Unsupervised**
- Reinforcement Learning
- Semi-Supervised
- Deep Learning



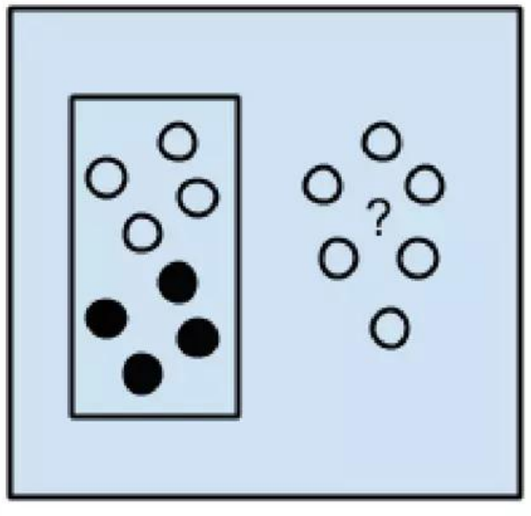
Deep Learning Algorithms



Semi-supervised Learning Algorithms

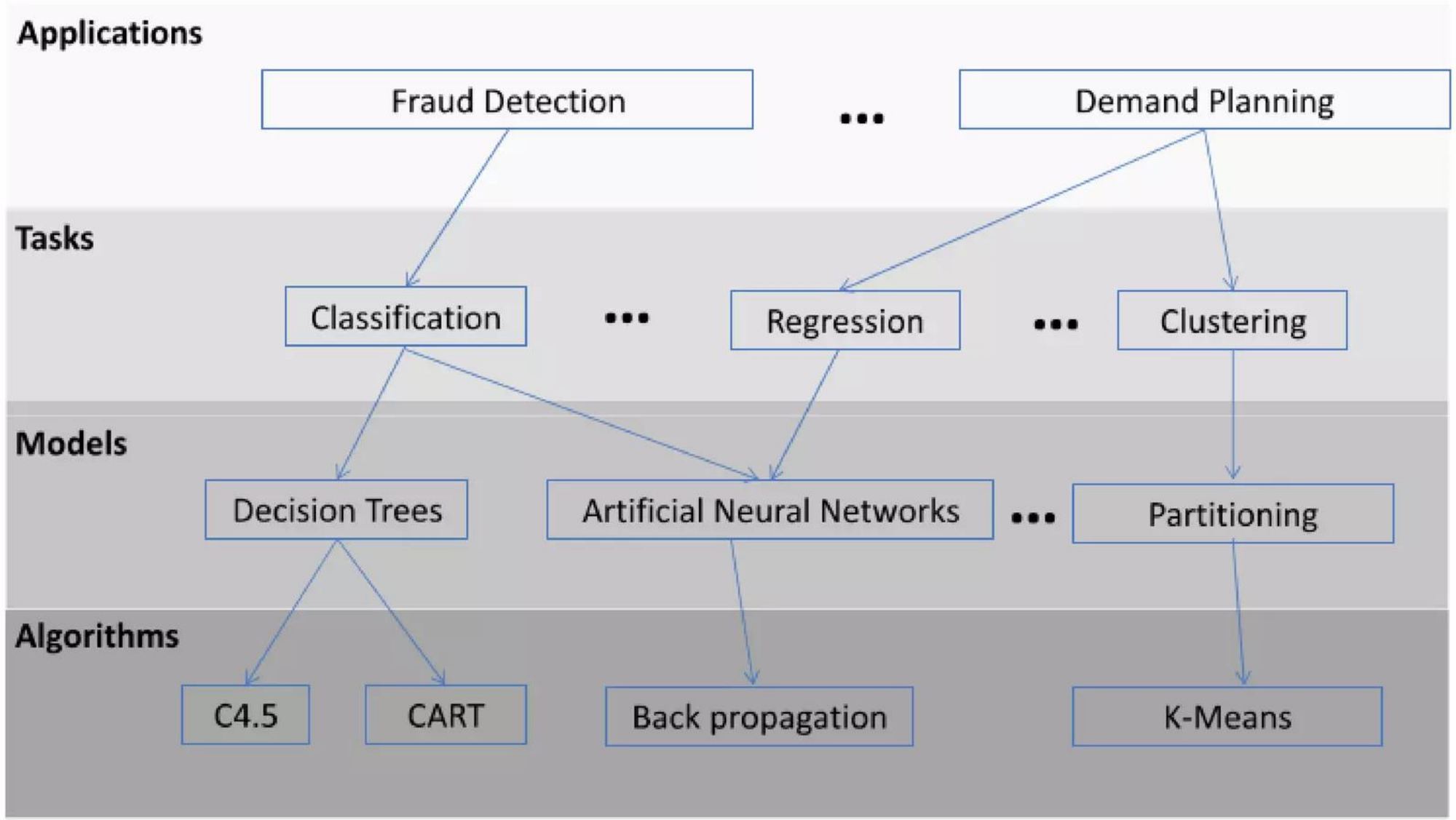


Unsupervised Learning Algorithms



Supervised Learning Algorithms

The four layer of data mining



Tools/libs in python

- Numpy
- Scipy
- Pandas
- Scikit-learn
- Matplotlib
- seaborn
- Tensorflow

*pydata.org
*anaconda

- Other Tech (to support ML) :
  - Apache Kafka
  - Apache Spark
  - Db : mongo, postgre
  - elasticsearch
  - CUDA/OpenCL

Numpy, scipy, padas, and sk-learn

- **Numpy & scipy**: Arrays, Indexing, Slicing, and Iterating, Reshaping, Shallow vs deep copy, Broadcasting, Indexing (advanced), Matrices, Matrix decompositions, Scipy on top numpy
- **Pandas** : Reading data, Selecting columns and rows, Filtering, Vectorized string operations, Missing values, Handling time, Time series, On top numpy.
- **SK-Learn** : Feature extraction, Classification, Regression, Clustering, Dimension reduction, Model selection

What we do in blibli using python

- **Data flow**
- **Data pooling**
- Data preprocessing
- Machine Learning Service/app

Our system that using python for ML

- Personalize recommendation system
- *Data engineering (especially the data flow for ML engine)*
- Machine learning engine
- **Fraud detection experiments**

EM Algorithms

Repeat until convergence{

(E-step) For each i, set

$$Q_i(z^{(i)}) := p(z^{(i)}|x^{(i)};\theta).$$

(M-step) Set

$$\theta := \arg\max_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}.$$

}

What??

EM Algorithms

There are 3 keys that (as far as I know) almost always used in EM-Algorithm :

- Data Distribution
- Maximum Likelihood Estimation (MLE)
- Estimation-Maximization (EM)

*Today we will use the **Gaussian distribution** for sample case

EM Algorithms

The algorithm has 2 main steps just like the name of the algorithm:

- Expectation :

$$Q_i(z^{(i)}) := p(z^{(i)}|x^{(i)};\theta).$$

- Maximization:

$$\theta := \arg\max_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}.$$

*repeat until get maximum likelihood :

$$\begin{aligned} \ell(\theta) &= \sum_{i=1}^{m} \log p(x;\theta) \\ &= \sum_{i=1}^{m} \log \sum_z p(x,z;\theta). \end{aligned}$$

Gaussian Distribution

Google

gaussian distribution

Semua Gambar Video Buku Berita Lainnya Setelan Alat

Sekitar 704.000 hasil (0,53 detik)

Normal distribution - Wikipedia

https://en.wikipedia.org/wiki/Normal_distribution

In probability theory, the normal distribution is a very common continuous probability distribution. **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. A random variable with a **Gaussian distribution** ...

Normal distribution · Multivariate normal distribution · Log-normal distribution

Normal Distributions: Definition, Word Problems - Statistics How To

www.statisticshowto.com/probability.../normal-distributions/ Terjemahkan halaman ini

A **normal distribution**, sometimes called the bell curve, is a distribution that occurs naturally in many situations. For example, the bell curve is seen in tests like the SAT and GRE. The bulk of students will score the average (C), while smaller numbers of students will score a B or D. An even smaller percentage of students ...

Normal Distribution - Math is Fun

https://www.mathsisfun.com/.../standard-normal-distribution.... Terjemahkan halaman ini

But there are many cases where the data tends to be around a central value with no bias left or right, and it gets close to a "**Normal Distribution**" like this: bell curve. A **Normal Distribution**. The "Bell Curve" is a **Normal Distribution**. And the yellow histogram shows some data that follows it closely, but not perfectly (which is ...

Distribusi normal

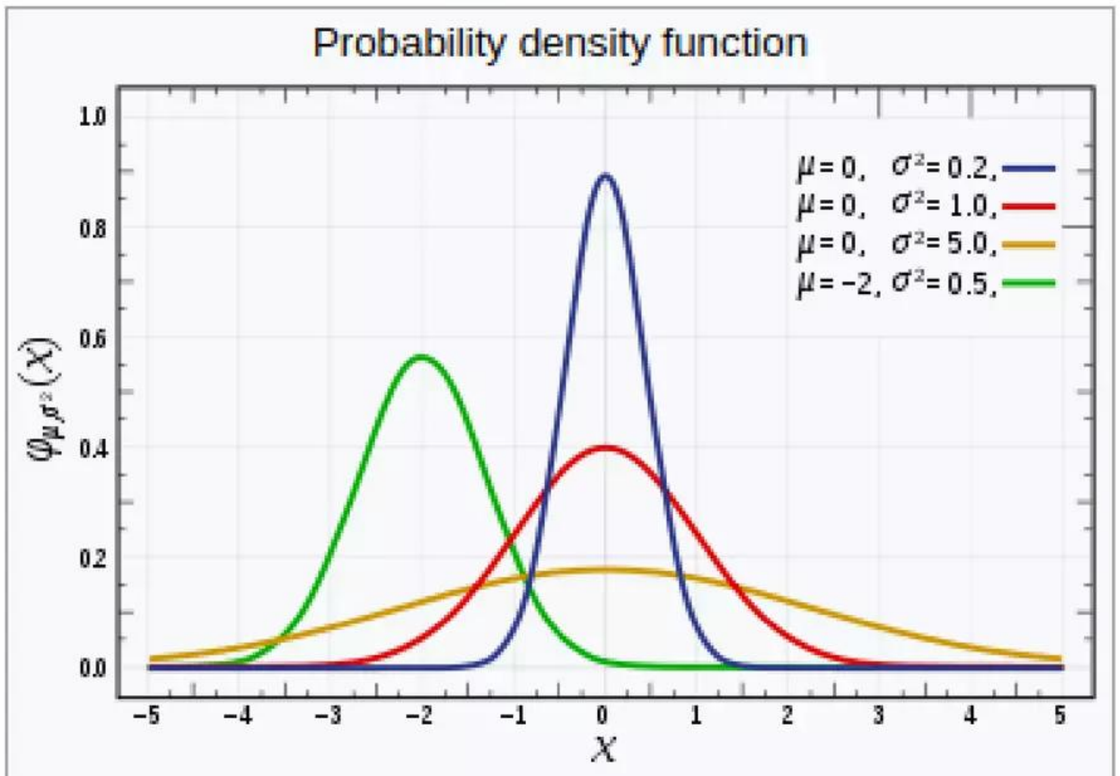
Distribusi normal, disebut pula distribusi Gauss, adalah distribusi probabilitas yang paling banyak digunakan dalam berbagai analisis statistika. Distribusi normal baku adalah distribusi normal yang memiliki rata-rata nol dan simpangan baku satu. Wikipedia

Masukan

Gaussian Distribution

In probability theory, the **normal** (or **Gaussian** or **Gauss** or **Laplace-Gauss**) **distribution** is a very common continuous probability distribution. 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.[1][2] A random variable with a Gaussian distribution is said to be **normally distributed** and is called a **normal deviate**.

Normal Distribution



The red curve is the *standard normal distribution*

	x
Notation	$\mathcal{N}(\mu, \sigma^2)$
Parameters	$\mu \in \mathbb{R}$ = mean (location) $\sigma^2 > 0$ = variance (squared scale)
Support	$x \in \mathbb{R}$
PDF	$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

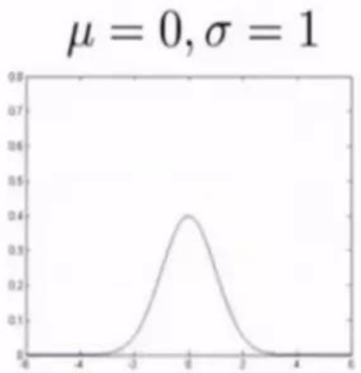
Gaussian Multivariate

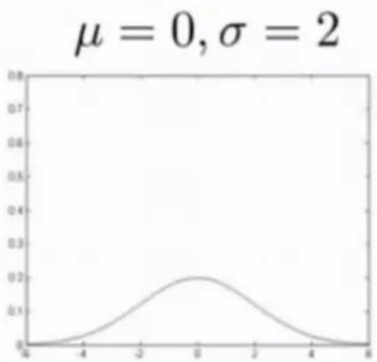
- Gaussian Distribution :

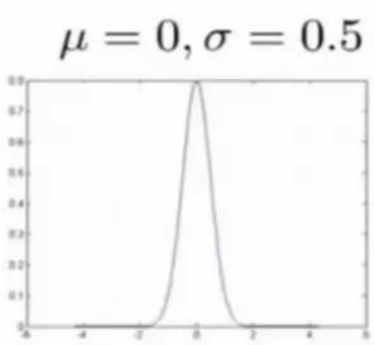
$$p(x_j; \mu_j, \sigma_j^2) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

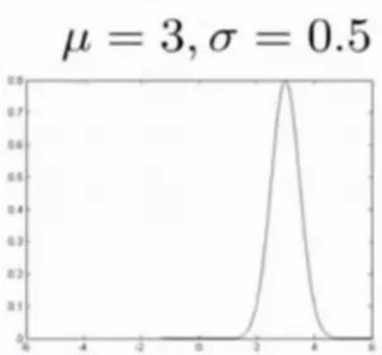
- Gaussian Distribution Multivariate :

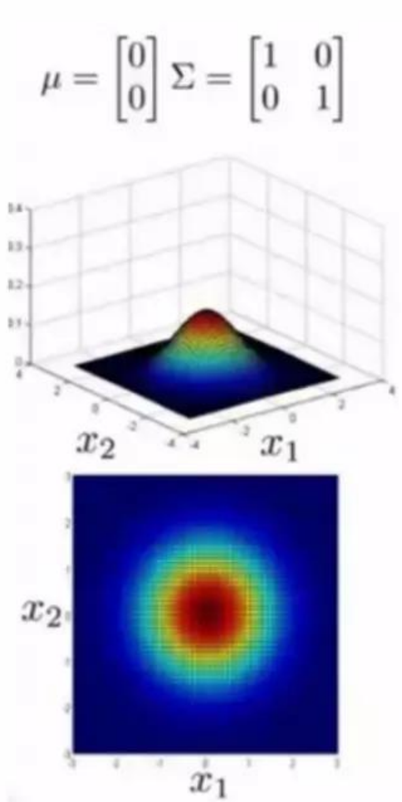
$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right).$$

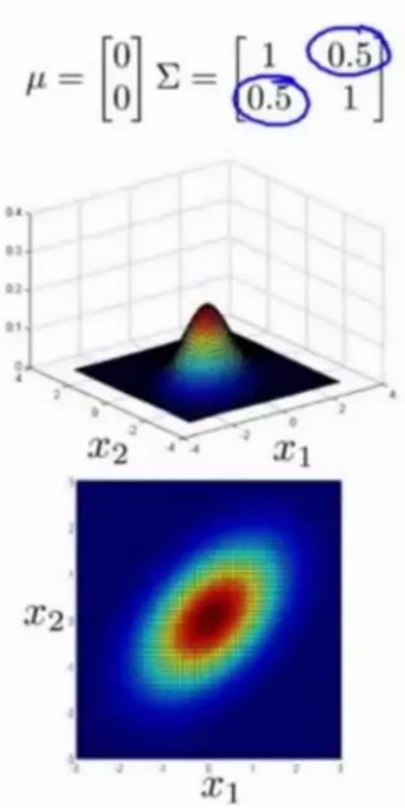


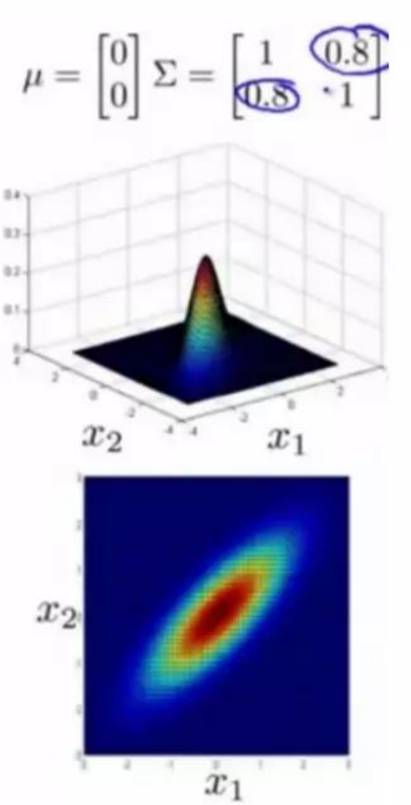




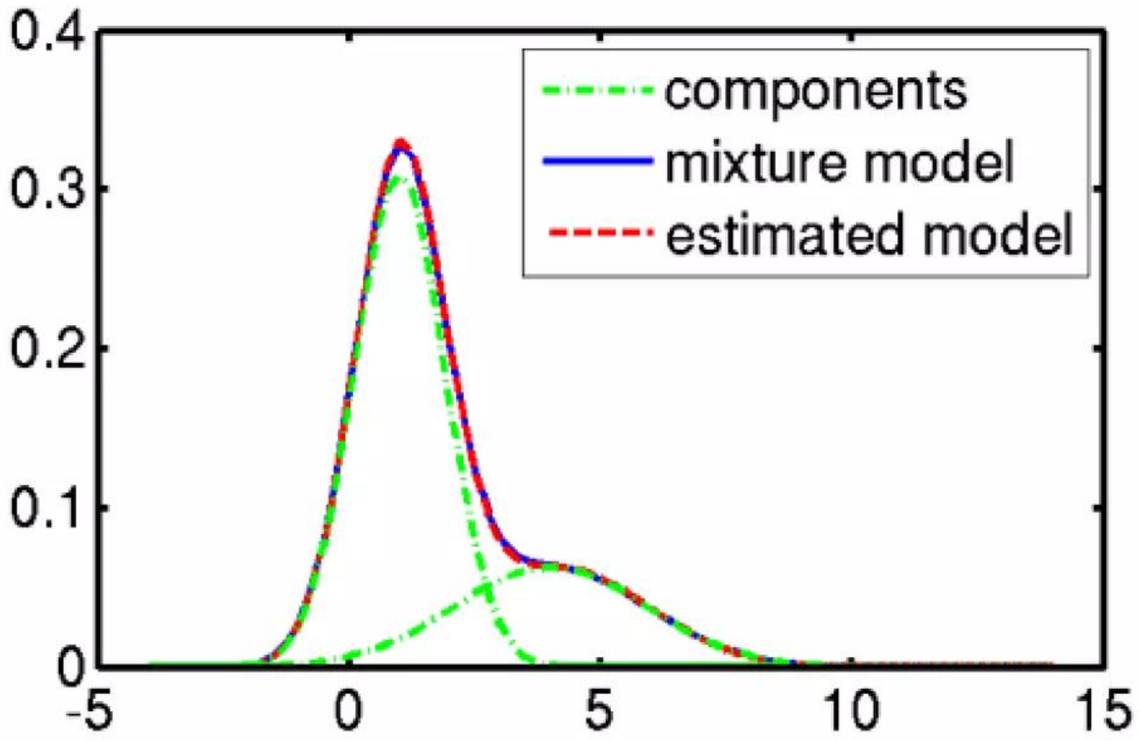


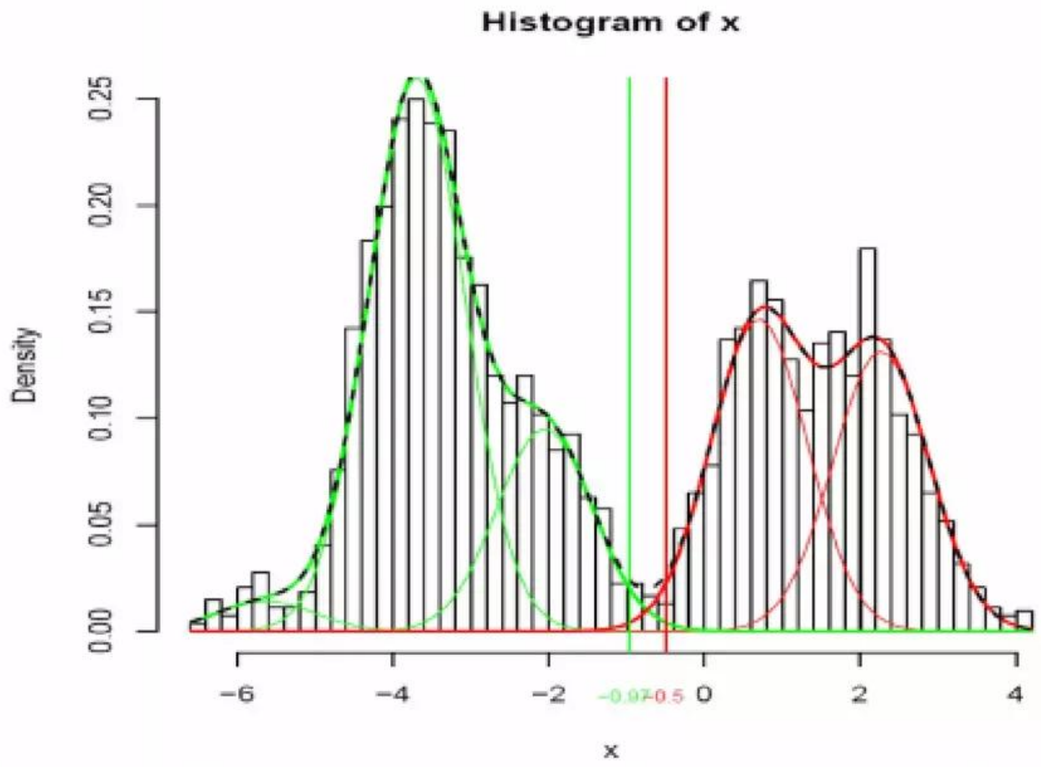






Mixture Gaussian





EM-Algorithm for Mixture Gaussian

- Expectation : $w_j^{(i)} := p(z^{(i)} = j|x^{(i)}; \phi, \mu, \Sigma)$

$$p(z^{(i)} = j|x^{(i)}; \phi, \mu, \Sigma) = \frac{p(x^{(i)}|z^{(i)} = j; \mu, \Sigma)p(z^{(i)} = j; \phi)}{\sum_{l=1}^{k} p(x^{(i)}|z^{(i)} = l; \mu, \Sigma)p(z^{(i)} = l; \phi)}$$

- Maximization :

$$\phi_j := \frac{1}{m}\sum_{i=1}^{m} w_j^{(i)},$$

$$\mu_j := \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}},$$

$$\Sigma_j := \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$$

*Log likelihood :

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}|z^{(i)}; \mu, \Sigma) + \log p(z^{(i)}; \phi).$$

Fraud – without target class/labels

- These are anomalous data
- Anomaly data usually have one or some small group of data
- A lot of features without labels

- We need unsupervised algorithm (EM-Algorithm)

Case Anomaly Detection

- Credit Card data with fraudulant data.

	A	B	C	D	E	F	G	H	I	J	K
1	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
2	1	PAYMENT	9839.64	C1231006815	170136	160296.36	M1979787155	0	0	0	0
3	1	PAYMENT	1864.28	C1666544295	21249	19384.72	M2044282225	0	0	0	0
4	1	TRANSFER	181	C1305486145	181	0	C553264065	0	0	1	0
5	1	CASH_OUT	181	C840083671	181	0	C38997010	21182	0	1	0
6	1	PAYMENT	11668.14	C2048537720	41554	29885.86	M1230701703	0	0	0	0
7	1	PAYMENT	7817.71	C90045638	53860	46042.29	M573487274	0	0	0	0
8	1	PAYMENT	7107.77	C154988899	183195	176087.23	M408069119	0	0	0	0
9	1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0	0	0	0
10	1	PAYMENT	4024.36	C1265012928	2671	0	M1176932104	0	0	0	0
11	1	DEBIT	5337.77	C712410124	41720	36382.23	C195600860	41898	40348.79	0	0
12	1	DEBIT	9644.94	C1900366749	4465	0	C997608398	10845	157982.12	0	0
13	1	PAYMENT	3099.97	C249177573	20771	17671.03	M2096539129	0	0	0	0
14	1	PAYMENT	2560.74	C1648232591	5070	2509.26	M972865270	0	0	0	0
15	1	PAYMENT	11633.76	C1716932897	10127	0	M801569151	0	0	0	0
16	1	PAYMENT	4098.78	C1026483832	503264	499165.22	M1635378213	0	0	0	0
17	1	CASH_OUT	229133.94	C905080434	15325	0	C476402209	5083	51513.44	0	0
18	1	PAYMENT	1563.82	C761750706	450	0	M1731217984	0	0	0	0
19	1	PAYMENT	1157.86	C1237762639	21156	19998.14	M1877062907	0	0	0	0
20	1	PAYMENT	671.64	C2033524545	15123	14451.36	M473053293	0	0	0	0
21	1	TRANSFER	215310.3	C1670993182	705	0	C1100439041	22425	0	0	0
22	1	PAYMENT	1373.43	C20804602	13854	12480.57	M1344519051	0	0	0	0
23	1	DEBIT	9302.79	C1566511282	11299	1996.21	C1973538135	29832	16896.7	0	0
24	1	DEBIT	1065.41	C1959239586	1817	751.59	C515132998	10330	0	0	0
25	1	PAYMENT	3876.41	C504336483	67852	63975.59	M1404932042	0	0	0	0
26	1	TRANSFER	311685.89	C1984094095	10835	0	C932583850	6267	2719172.89	0	0
27	1	PAYMENT	6061.13	C1043358826	443	0	M1558079303	0	0	0	0
28	1	PAYMENT	9478.39	C1671590089	116494	107015.61	M58488213	0	0	0	0
29	1	PAYMENT	8009.09	C1053967012	10968	2958.91	M295304806	0	0	0	0
30	1	PAYMENT	8901.99	C1632497828	2958.91	0	M33419717	0	0	0	0
31	1	PAYMENT	9920.52	C764826684	0	0	M1940055334	0	0	0	0

Case Anomaly Detection

```
import pandas as pd
import pandas_datareader.data as web
import numpy as np
import scipy.stats as scs
from scipy.stats import multivariate_normal as mvn
import sklearn.mixture as mix
import copy
from pandas.plotting import scatter_matrix

import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import random
```

:

```
f = '/home/hendri/python_projects/samples/Untitled Folder/fraud_cc.csv'
# Take every N-th (in this case 10th) row
n = 500

# Count the lines or use an upper bound
num_lines = sum(1 for l in open(f))

# The row indices to skip - make sure 0 is not included to keep the header!
skip = [x for x in range(1, num_lines) if x % n != 0]


data = pd.read_csv(filepath_or_buffer = f, skiprows=skip)
# data.info()
data
```

Case Anomaly Detection

```
import pandas as pd
import pandas_datareader.data as web
import numpy as np
import scipy.stats as scs
from scipy.stats import multivariate_normal as mvn
import sklearn.mixture as mix
import copy
from pandas.plotting import scatter_matrix

import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import random
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	5882.32	C278476563	10252.00	4369.68	M1479909053	0.00	0.00	0	0
1	1	PAYMENT	2284.54	C49318987	539.00	0.00	M1058650291	0.00	0.00	0	0
2	1	DEBIT	8648.45	C397318359	6734.00	0.00	C171493374	11010.00	0.00	0	0
3	1	PAYMENT	7363.65	C1687087217	1574226.59	1566862.94	M928734079	0.00	0.00	0	0
4	1	PAYMENT	7677.95	C1150436743	45881.29	38203.34	M407241000	0.00	0.00	0	0
5	2	PAYMENT	12710.27	C33238366	325690.00	312979.73	M387377349	0.00	0.00	0	0
6	2	PAYMENT	5374.49	C957608454	695819.48	690444.99	M65567135	0.00	0.00	0	0
7	3	DEBIT	5922.61	C512734831	17149.68	11227.07	C461160828	18493.88	24416.50	0	0
8	4	CASH_IN	64846.34	C394645908	6758299.24	6823145.58	C11003494	11364261.67	10946411.29	0	0
9	5	CASH_IN	50535.87	C189993695	4875810.94	4926346.82	C187649742	70183.75	19647.88	0	0
10	5	CASH_OUT	297734.01	C1301228797	0.00	0.00	C1477979030	355036.00	9113630.99	0	0
11	6	PAYMENT	35236.35	C902556500	21026.00	0.00	M175825728	0.00	0.00	0	0
12	6	PAYMENT	487.15	C1268892066	0.00	0.00	M861310502	0.00	0.00	0	0
13	6	PAYMENT	13407.42	C1637577782	29802.00	16394.58	M1794130044	0.00	0.00	0	0
14	7	PAYMENT	4607.26	C1491860427	48639.00	44031.74	M1755838330	0.00	0.00	0	0
15	7	CASH_OUT	498747.45	C1525969940	0.00	0.00	C991505714	4391509.82	5352935.74	0	0
16	7	PAYMENT	13223.51	C1686896318	165.00	0.00	M943346272	0.00	0.00	0	0
17	7	CASH_OUT	267174.86	C1511538598	0.00	0.00	C1636786811	6325419.04	6779300.28	0	0
18	7	PAYMENT	8012.89	C636910314	106992.00	98979.11	M1335046207	0.00	0.00	0	0
19	7	PAYMENT	5096.16	C299358529	60472.00	55375.84	M176755220	0.00	0.00	0	0

:
```
f = '/home/hendri/python_projects/samples/Untitled Folder/fraud_cc.csv'
# Take every N-th (in this case 10th) row
n = 500

# Count the lines or use an upper bound
num_lines = sum(1 for l in open(f))

# The row indices to skip - make sure 0 is not included to keep the header!
skip = [x for x in range(1, num_lines) if x % n != 0]


data = pd.read_csv(filepath_or_buffer = f, skiprows=skip)
# data.info()
data
```

```
data[data.isnull().any(axis=1)]
```

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud

Case Anomaly Detection

```
flags = data.loc[:, data.columns == 'isFraud']


data  = data.drop(['isFraud','isFlaggedFraud'], axis=1)
columns = data.columns.values
print(columns)
data
```

```
['step' 'type' 'amount' 'nameOrig' 'oldbalanceOrg' 'newbalanceOrig'
 'nameDest' 'oldbalanceDest' 'newbalanceDest']
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest
0	1	PAYMENT	5882.32	C278476563	10252.00	4369.68	M1479909053	0.00	0.00
1	1	PAYMENT	2284.54	C49318987	539.00	0.00	M1058650291	0.00	0.00
2	1	DEBIT	8648.45	C397318359	6734.00	0.00	C171493374	11010.00	0.00
3	1	PAYMENT	7363.65	C1687087217	1574226.59	1566862.94	M928734079	0.00	0.00
4	1	PAYMENT	7677.95	C1150436743	45881.29	38203.34	M407241000	0.00	0.00
5	2	PAYMENT	12710.27	C33238366	325690.00	312979.73	M387377349	0.00	0.00
6	2	PAYMENT	5374.49	C957608454	695819.48	690444.99	M65567135	0.00	0.00
7	3	DEBIT	5922.61	C512734831	17149.68	11227.07	C461160828	18493.88	24416.50
8	4	CASH_IN	64846.34	C394645908	6758299.24	6823145.58	C11003494	11364261.67	10946411.29
9	5	CASH_IN	50535.87	C189993695	4875810.94	4926346.82	C187649742	70183.75	19647.88
10	5	CASH_OUT	297734.01	C1301228797	0.00	0.00	C1477979030	355036.00	9113630.99
11	6	PAYMENT	35236.35	C902556500	21026.00	0.00	M175825728	0.00	0.00

```
datacp = copy.deepcopy(data)
cat_col =['type', 'nameOrig','nameDest']
for temp in cat_col:
    datacp[temp] = datacp[temp].astype('category')
print(datacp.dtypes)
datacp
```

```
for temp in cat_col:
    datacp[temp] = datacp[temp].cat.codes

datacp
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest
0	1	3	5882.32	7902	10252.00	4369.68	9361	0.00	0.00
1	1	3	2284.54	9350	539.00	0.00	8422	0.00	0.00
2	1	2	8648.45	8715	6734.00	0.00	2963	11010.00	0.00
3	1	3	7363.65	4481	1574226.59	1566862.94	12439	0.00	0.00
4	1	3	7677.95	995	45881.29	38203.34	11306	0.00	0.00
5	2	3	12710.27	8272	325690.00	312979.73	11270	0.00	0.00
6	2	3	5374.49	12464	695819.48	690444.99	11831	0.00	0.00
7	3	2	5922.61	9496	17149.68	11227.07	5921	18493.88	24416.50
8	4	0	64846.34	8691	6758299.24	6823145.58	405	11364261.67	10946411.29
9	5	0	50535.87	5861	4875810.94	4926346.82	3680	70183.75	19647.88
10	5	1	297734.01	1922	0.00	0.00	1964	355036.00	9113630.99
11	6	3	35236.35	12082	21026.00	0.00	9991	0.00	0.00
12	6	3	487.15	1729	0.00	0.00	12271	0.00	0.00
13	6	3	13407.42	4149	29802.00	16394.58	10073	0.00	0.00
14	7	3	4607.26	3238	48639.00	44031.74	9985	0.00	0.00
15	7	1	498747.45	3457	0.00	0.00	8279	4391509.82	5352935.74
16	7	3	13223.51	4480	165.00	0.00	12471	0.00	0.00

Case Anomaly Detection

```
data.shape
```

```
(12725, 9)
```

```
gmm = mix.GaussianMixture( n_components=2,max_iter=5000, covariance_type='tied')
gmm.fit(datacp)


classes = gmm.predict(datacp)
pca_c = PCA(n_components=2)
comp_data = pca_c.fit(datacp).transform(datacp)
print(comp_data)

colors = ['navy', 'turquoise']
lw = 2
print('-----------------------')
tempdf = pd.DataFrame(data = classes, columns=['classes'])
list_color=[[0,'red'],[1,'blue']]
dfcolor = pd.DataFrame(data = list_color, columns=['classes','color'])
mergeddf = pd.merge(tempdf,dfcolor)

#Then we do the graph
plt.scatter(comp_data[:,0],comp_data[:,1],color=mergeddf['color'])
plt.show()
```

```
y = pd.Series(flags['isFraud'])
print('Jumlah cluster original label 0 : ',len(y[y==0].index))
print('Jumlah cluster from EM label 0 : ',len(result[result==0].index))

print('Jumlah cluster original label 1 : ',len(y[y==1].index))
print('Jumlah cluster from EM label 1 : ',len(result[result==1].index))

evaluation = (result == y)
print('True : ', len(evaluation[evaluation == True]))
print('False : ', len(evaluation[evaluation == False]))
false_idx = evaluation[evaluation == False].index
print(false_idx)
false_pos = [y[i] for i in false_idx if y[i] == 0]
false_neg = [y[i] for i in false_idx if y[i] == 1]

print()
print('false positive : ' , len(false_pos))
print('false negative : ' , len(false_neg))
```

```
Jumlah cluster original label 0 :  12709
Jumlah cluster from EM label 0 :  12712
Jumlah cluster original label 1 :  16
Jumlah cluster from EM label 1 :  13
True :  12696
False :  29
Int64Index([ 2060,  2109,  2118,  5129,  5699,  5750,  6515,  7221,  7337,
             7679,  7751,  7954,  8072,  8172,  8282,  8298,  8328,  8355,
             8357,  8358,  8367,  9185,  9198, 11975, 12129, 12336, 12525,
            12562, 12724],
           dtype='int64')

false positive :  13
false negative :  16
```

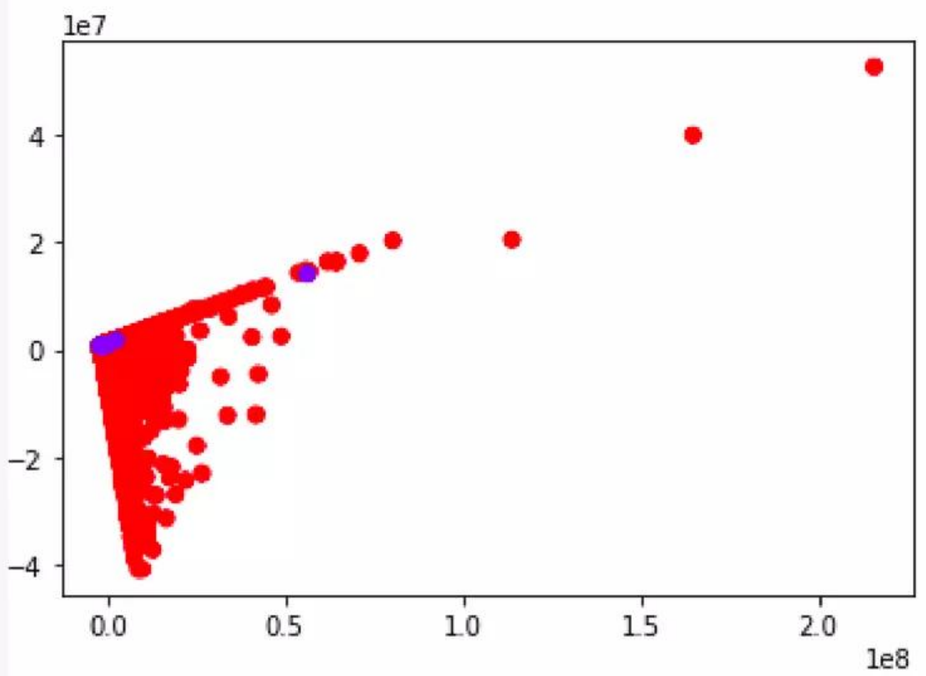
```
gmm = mix.GaussianMixture( n_components=2,max_iter=5000, covariance_type='tied')
gmm.fit(datacp)


classes = gmm.predict(datacp)
pca_c = PCA(n_components=2)
comp_data = pca_c.fit(datacp).transform(datacp)
print(comp_data)

colors = ['navy', 'turquoise']
lw = 2

tempdf = pd.DataFrame(data = classes, columns=['classes'])
list_color=[[0,'red'],[1,'blue']]
dfcolor = pd.DataFrame(data = list_color, columns=['classes','color'])
mergeddf = pd.merge(tempdf,dfcolor)

#Then we do the graph
plt.scatter(comp_data[:,0],comp_data[:,1],color=mergeddf['color'])
plt.show()
```



Problem Performance

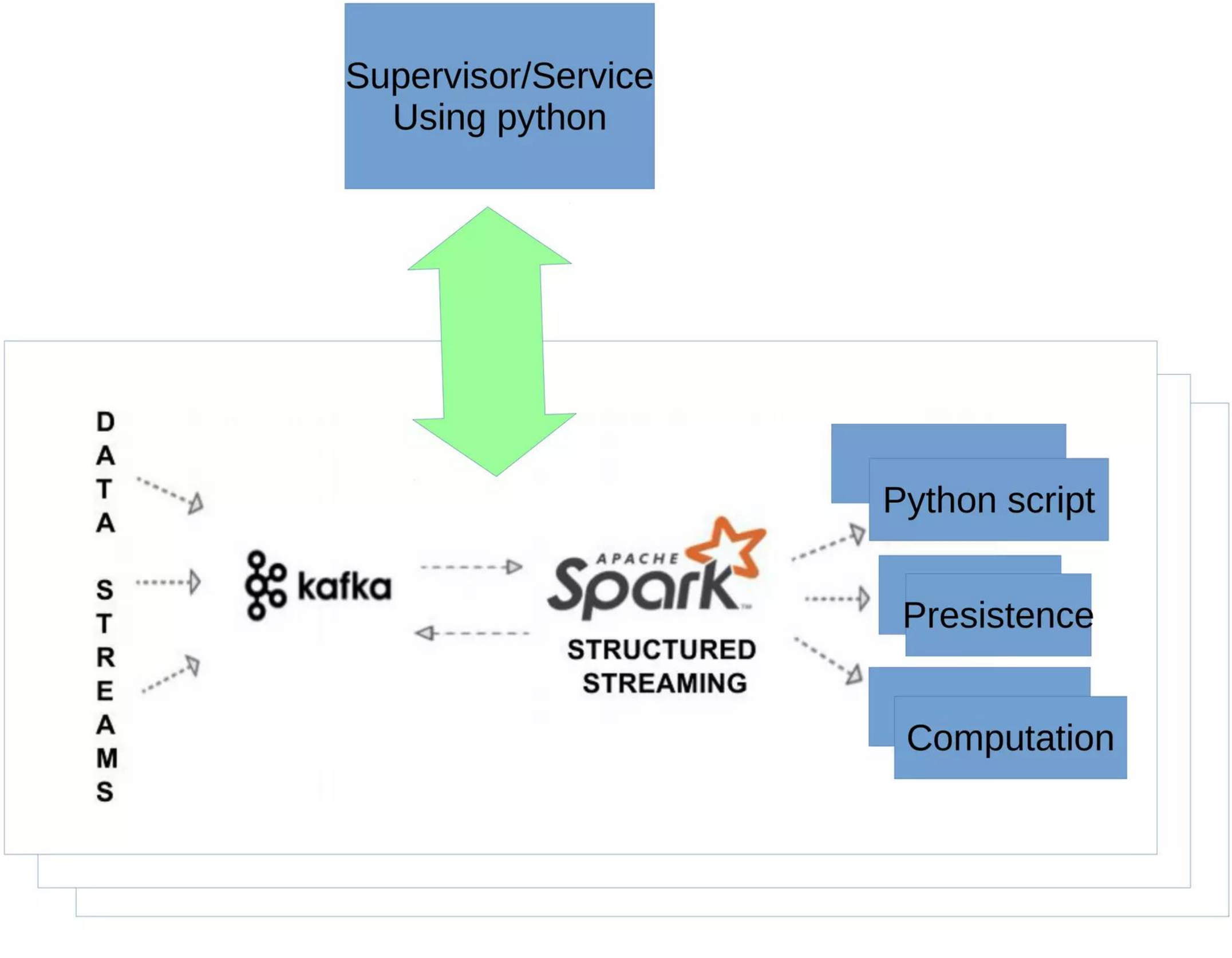
dri@astrajingga: ~

hendri@astrajingga: ~ × | hendri@astrajingga: ~

```
  1  [|||||||||||||||||                                    36.9%]   Tasks: 25
  2  [|||||||||||||||||||                                  40.6%]   Load aver
  3  [|||||||||||||||||||                                  41.6%]   Uptime: 1
  4  [|||||||||||||||||||                                  41.4%]
Mem[|||||||||||||||||||||||||||||||||||||||||||||||||15.0G/15.4G]
Swp[|||||||||||||||||||||||||||||||||||||||||||||||||12.2G/15.1G]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Comma
 1761 hendri     20   0 1577M  234M 22456 S 19.7  1.5 19:44.47 compi
10469 hendri     20   0 2989M  230M 30348 R 50.6  1.5  1:24.29 /opt/
 4824 hendri     20   0 1606M  330M 46104 S 23.0  2.1  1h35:36 /opt/
 2302 hendri     20   0 2182M  326M 67448 S 15.1  2.1 45:49.06 /opt/
    ri           20   0 1329M  115M 12412 S  0.0  0.7  0:56.32 /opt/
                 20   0  766M 81620 55340 S 13.1  0.5 20:03.21 /usr/
    ri           20   0 1606M  330M 46104 S  9.2  2.1 33:44.01 /opt/
    ri           20   0 1606M  330M 46104 S  8.5  2.1 32:00.36 /opt/
    ri           20   0 1275M  126M 20800 S  7.2  0.8 36:09.41 /opt/
    ri           20   0 1246M  122M 24480 S  5.3  0.8 14:32.17 /opt/
```

Distributed System/Scale Out



THANK YOU

Any question?

we are hiring