Machine learning

Anomaly Detection(Outliers)

Morteza khorsand

- **Statistical population**

  a set of similar items or events which is of interest for some question or experiment

- **Statistical population**

  A subset of a population that shares one or more additional properties is called a *sub population*.

  population mean

  The population mean, or population expected value, is a measure of the central tendency.

- **Non- parametric methods**

  a statistical method in which the data are not assumed to come from prescribed models that are determined by a small number of parameters. examples of such as linear regression model. data can be collected from a sample that does not follow a specific distribution.

- **parametric methods**

  assumes that sample data comes from a population that can be adequately modeled by a probability distribution that has a fixed set of parameters

Anomaly Detection (Outliers)

- **Range**

Max $(x_i) \; - \boldsymbol{min}(x_i)$

- **The number of classes**

K = 1 + 3.32 log n
K= $\sqrt{n}$

- **Mode**

The mode is the value that appears most often in a set of data values.
Sometimes mode is the criterion in a population like voting.

2,2,9,9,9,5 : 9 is the mode

- **Median**

the middle" value

Anomaly Detection (Outliers)

- **Gaussian Mixture Model (GMM) for Anomaly Detection (Z-score method) – normal distribution**

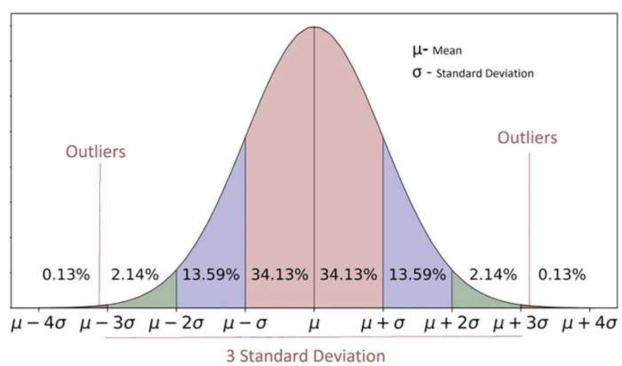
$x \sim N\;(\mu, \sigma^2)$

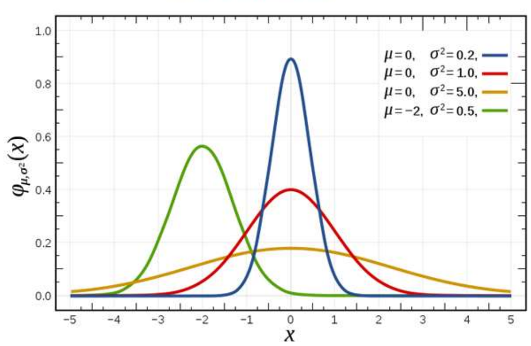
$$p(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \; e^{(-\frac{(x-\mu)^2}{2\,\sigma^2})}$$

$p(\mu - \sigma < x < \mu + \sigma)$ = 0.6827

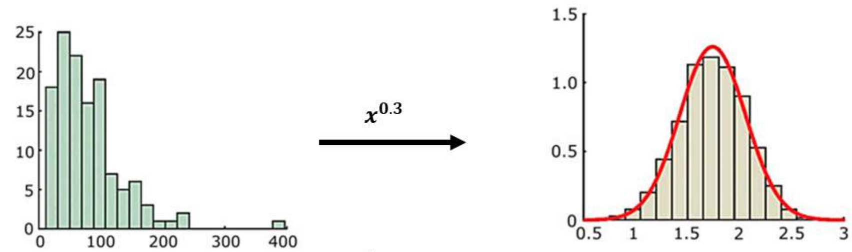
$p(\mu - 2\sigma < x < \mu + 2\sigma)$ = 0.9544

$p(\mu - 3\sigma < x < \mu + 3\sigma)$ = 0.9973

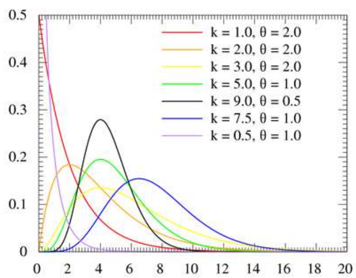




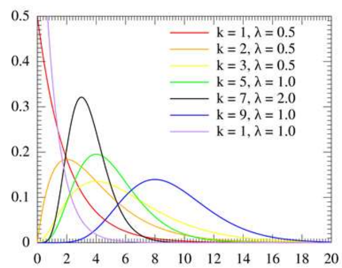
Other distributions



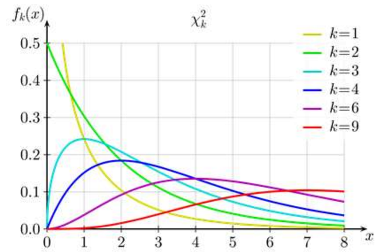
Gamma distribution Normal distribution



Gamma distribution



Erlang distribution

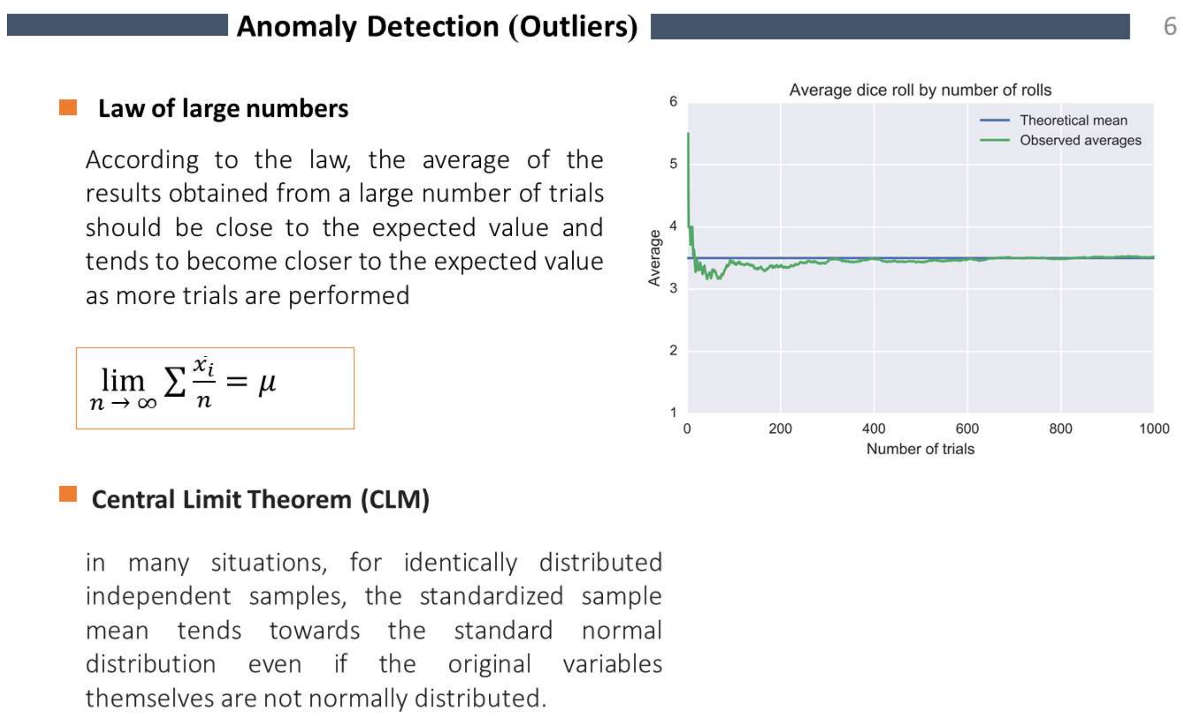


Chi-squared distribution

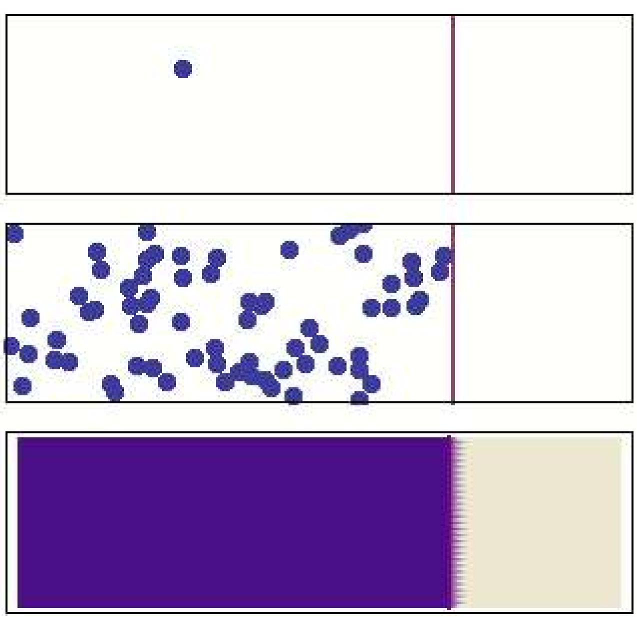
In [6]:
```
%matplotlib nbagg

import warnings
import pandas as pd
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
import ipywidgets as widgets

from mpl_toolkits.mplot3d import Axes3D
from numpy.linalg import pinv
from scipy.stats import multivariate_normal

import seaborn as sns; sns.set()

plt.rcParams['figure.figsize'] = (4, 4)
np.set_printoptions(precision=2)
warnings.filterwarnings('ignore')
```

In [ ]:
```
X1 = np.random.gamma(1, 2, size=(10000, 1))

plt.figure(figsize=(8, 4))
plt.subplot(121)
plt.hist(X1, bins=50)
plt.title('Original Data (Gamma Distribution)', size=10)

plt.subplot(122)
plt.hist(X1 ** 0.3, bins=50)
plt.title('Transformed Data (Normal Distribution)', size=10)
```

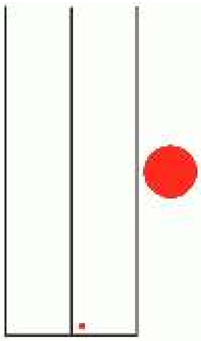
```
plt.show()
```



Diffusion



law of large numbers



- **Standard normal distribution**

$$x \sim N\,(\mu,\ \sigma)$$
$$\downarrow \quad \downarrow$$
$$Z \sim N\,(\,0,\ 1)$$

$$p(z\,;\mu = 0, \sigma^2 = 1) = \frac{1}{\sqrt{2\pi}}\, e^{\left(-\frac{(z)^2}{2}\right)}$$

z	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990

Example

$x \sim N\,(\mu = 10,\ \sigma = 2)$ calculate P(x<13)?

IQ level

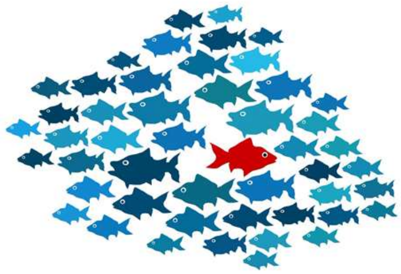
$x \sim N\,(\mu = 100,\ \sigma = 15)$ calculate P(x>125)?

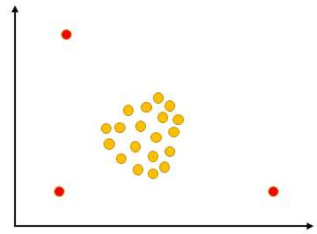
Anomaly Detection (Outliers)

Outliers(anomaly detection) – one class classification

Outliers are those data points that are *significantly* different from the rest of the dataset.

Fraud detection
Network Security





Reasons of Occurrence of Outliers:

· Data entry errors (human errors)
· Measurement errors (instrument errors)
· Experimental errors (data extraction or experiment planning/executing errors)
· Intentional (dummy outliers made to test detection methods)
· Data processing errors (data manipulation errors)
· Sampling errors (extracting or mixing data from wrong or various sources)
· **Natural** (not an error, novelties in data)

Anomaly Detection (Outliers)

Parameter evaluation

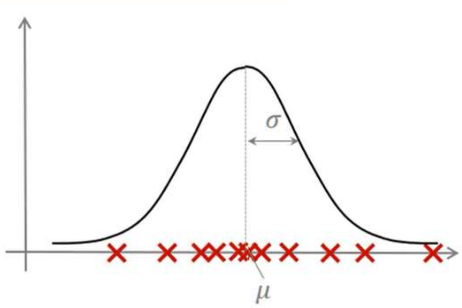
$data\ set \quad \{ x^{(1)}, x^{(2)}, x^{(3)} \ldots x^{(m)} \}$

$$\boxed{\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}}$$

$$\boxed{\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2}$$

$$p(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}$$

•if $p(x) < \varepsilon$, then x can be detected as an **anomalous data**.
•if $p(x) > \varepsilon$, then x can be regarded as a **normal data**




```
In [ ]: import numpy as np
        np.printoptions(precision=2)
```

```
import numpy as np
def read_dataset(fname, delimiter=','):
    return np.genfromtxt(fname, delimiter=delimiter)

X = read_dataset(r"F:\machine learning\jozavat\anomally\1dimensional.csv")
print(X.shape)
print(X[:10])
```

In [ ]:
```
mu=X.mean()
print(mu)
sigma=X.std()
print(sigma)
```

In [ ]:
```
z= (X-mu) / sigma
print(z.shape)
print(np.around(z[:10] , 3 ))
```

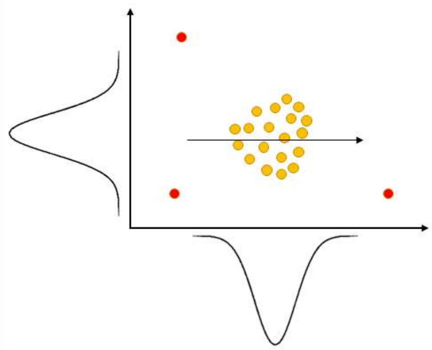
In [ ]:
```
import matplotlib.pyplot as plt
ax = fig.add_subplot(111)


outlier =z [p_z < 0.001]


plt.scatter(z , p_z +0.1, s=8, alpha=0.6, label='norm pdf' , c=p_z)
a=np.array([0 for i in  range(307)])
plt.scatter( z  , a , s= 8 , alpha = 0.6 , c=p_z)

b=a=np.array([0 for i in  range(outlier.shape[0])])
plt.scatter(outlier, b-.1, s=50, marker='x', c='blue', cmap='coolwarm')
plt.show()


np.argwhere(p_z < 0.001)
```

Algorithm

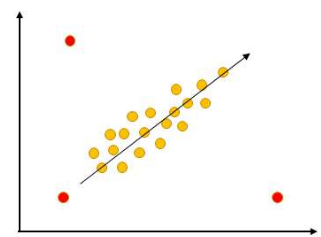
$data\ set \quad \{x^{(1)}, x^{(2)}, x^{(3)} \ldots x^{(m)}\} \qquad x^{(i)} \epsilon R^n$

- Features follow a normal distribution.
- Features do not follow a normal distribution

- correlation of x_1, x_2 are zero.(covariance matrix is diagonal)
- correlation of $x_1, x_2, \ldots x_n$ are not zero



no correlation



with correlation

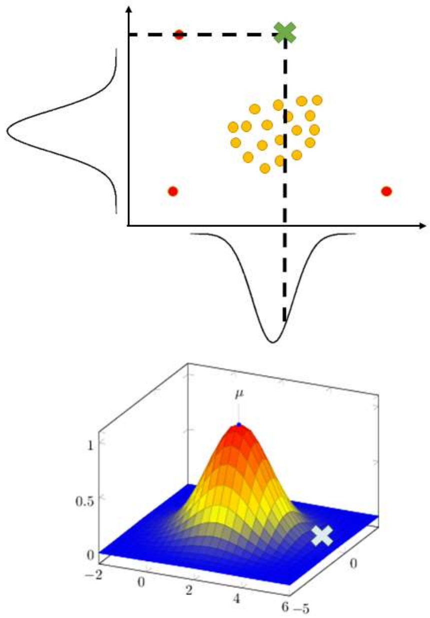
Algorithm

If $x_1, x_2, \ldots x_j$ are independent and $x_j \sim N(\mu_j, \sigma_j^2)$:

P(x) : $p(x_1; \mu_1, \sigma^2{}_1)p(x_2; \mu_2, \sigma^2{}_2) \;\ldots\ldots\; p(x_n; \mu_n, \sigma^2{}_n) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma^2{}_j)$

$$\prod_{j=1}^{n} \frac{1}{\sigma\sqrt{2\pi}}\, e^{\left(-\frac{(x_j-\mu_j)^2}{2\sigma^2{}_j}\right)}$$

Problem



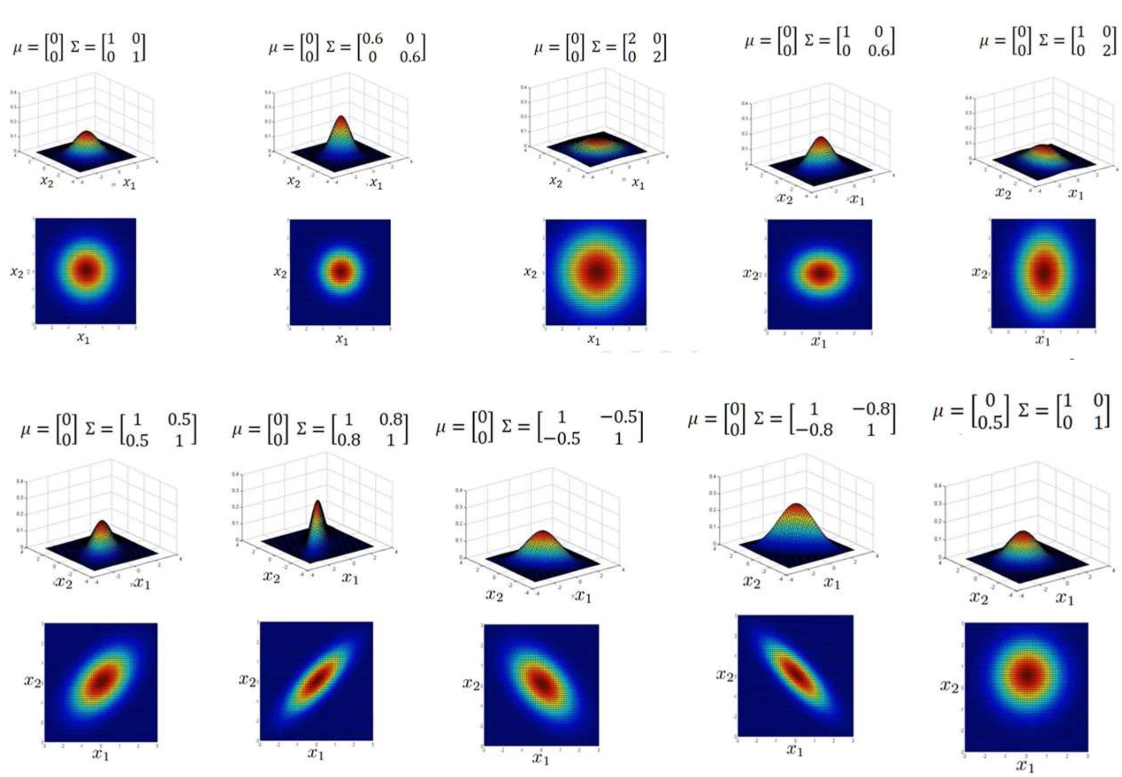
- **multivariate normal - multinomial**

$$P(x;\boldsymbol{\mu},\Sigma) = \frac{1}{|\Sigma|^{\frac{1}{2}}\,(2\pi)^{\frac{n}{2}}}\; e^{\left(-\frac{(x-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{\mathbf{2}}\right)}$$

Parameters : $\mu \in R^n, \quad \Sigma \in R^{n\times n}$

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)\,(x^{(i)} - \mu)^T$$



In [ ]:
```python
def read_dataset(fname, delimiter=','):
    return np.genfromtxt(fname, delimiter=delimiter)
```

```
X = read_dataset(r"F:\machine learning\jozavat\anomally\tr_server_data.csv")
print(X.shape)
print(X[:5])
```

In [ ]:
```
plt.figure(figsize=(6, 6))
plt.scatter(X[:, 0], X[:, 1], s=25, edgecolors='k', cmap='coolwarm', alpha=0.6)
plt.xlabel('Latency (ms)')
plt.ylabel('Throughput (Mb/s)')
plt.show()
```

In [ ]:
```
np.set_printoptions(precision=2)
mu = np.mean(X, axis=0)
m=X.shape[0]

Sigma= np.cov(X.T)     # sigma= ((X-mu).T @(X-mu)) * 1/m



print('mu =\n {}'.format(mu))
print('Sigma =\n {}'.format(Sigma))
```

In [ ]:
```
# defining the probabilistic model and computing
# the probabilty of observing each data in training data
p = multivariate_normal(mean=mu, cov=Sigma).pdf(X)

# print the probabilities for the first 10 data
print(p.shape)
print(p[:10])
```

In [ ]:
```
fig = plt.figure(figsize=(6, 6))
ax = fig.add_subplot(111)
normal = X[p >= 0.001]
outlier = X[p < 0.001]
ax.scatter(normal[:, 0], normal[:, 1], s=25, marker='o', c='b', edgecolors = "black",c
ax.scatter(outlier[:, 0], outlier[:, 1], s=50, marker='x', c='r',  cmap='coolwarm')
plt.show()

np.argwhere(p < 0.001)
```

Anomaly detection as an algorithm

- Anomaly detection as an algorithm

1. Evaluate parameters $$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \qquad \Sigma = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)\,(x^{(i)} - \mu)^T$$

2. Calculate p(x) for new x $$P(x; \boldsymbol{\mu}, \Sigma) = \frac{1}{|\Sigma|^{\frac{1}{2}}\,(2\pi)^{\frac{n}{2}}}\; e^{\left(-\frac{(x-\boldsymbol{\mu})^T \Sigma^{-1} (x-\boldsymbol{\mu})}{2}\right)}$$

3. Output : yes if $p(x) < \boldsymbol{\varepsilon}$

Normal : y = 0
Anomalous : y = 1

$data\ set$ $\{x^{(1)}, x^{(2)}, x^{(3)} \ldots x^{(m)}\}$ $x^{(i)} \epsilon R^n$

$validation\ data$ $\{(x_{cv}{}^{(1)}, y_{cv}{}^{(1)}), (x_{cv}{}^{(2)}, y_{cv}{}^{(2)}), \ldots (x_{cv}{}^{(m_{cv})}, y_{cv}{}^{(m_{cv})})\}$

$test\ data$ $\{(x_{test}{}^{(1)}, y_{test}{}^{(1)}), (x_{test}{}^{(2)}, y_{test}{}^{(2)}), \ldots (x_{test}{}^{(m_{test})}, y_{test}{}^{(m_{test})})\}$

- **Statistical hypothesis testing**

used to decide whether the data at hand sufficiently support a particular hypothesis.

$\boldsymbol{H_0}$: null hypothesis

$\boldsymbol{H_1}$: alternative hypotheses

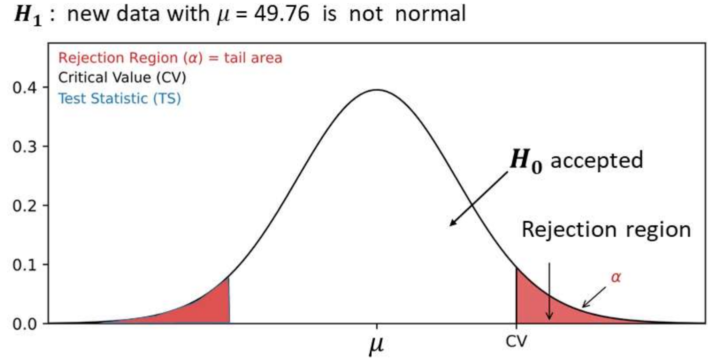
Example :

samples	kg
$x^{(1)}$	49.85
$x^{(2)}$	50.05
…	…
$x^{(16)}$	50.59
	$\bar{\mu} = 50$

$x \sim N(\mu,\ \sigma)$

$\boldsymbol{H_0}$: new data with μ = 49.76 is normal

$\boldsymbol{H_1}$: new data with μ = 49.76 is not normal

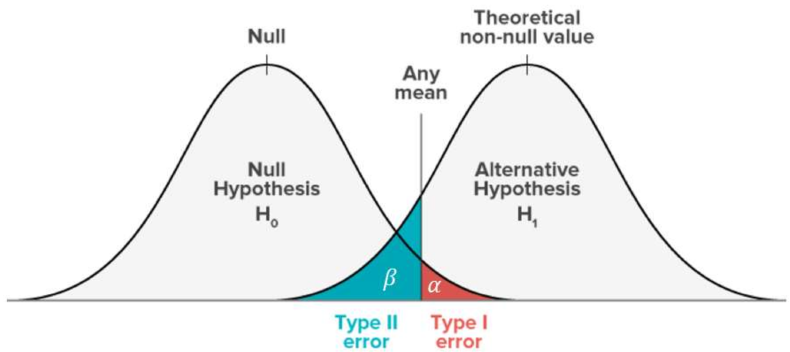


- **Errors hypothesis testing**

Error type I: H_0 is true but we reject it. **False-positive**

Error type II: H_0 is false but we accept it. **False-negative**

Actual / Prediction	True H_0	False H_0
Accept H_0	1-α	False-negative β
Reject H_0	False-positive α	1-β



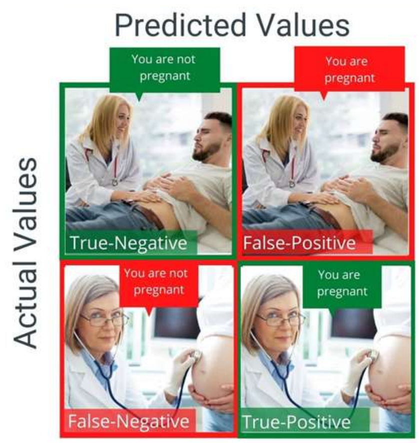
- **Criteria evaluation**

Confusion Matrix

- True positive
- False positive
- True negative
- False negative

- Precision
- Recall

- F_1score



		Actual: Positive	Actual: Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Criteria evaluation

- Precision

$$0 \leq \frac{ture\ positive}{true\ positive + false\ positive} \leq 1$$

- Recall

$$0 \leq \frac{ture\ positive}{true\ positive + false\ negative} \leq 1$$

- F_1score

$$F_1 = 2 * \frac{precision \times recal}{precision + recal}$$

In [3]:

```
import numpy as np
def read_dataset(fname, delimiter=','):
    return np.genfromtxt(fname, delimiter=delimiter)

X_train = read_dataset(r"F:\machine learning\jozavat\anomally\test train vald\train.cs
print(X_train.shape)
print(X_train[:5])

print("---------------------------------------")

data_validation = read_dataset(r"F:\machine learning\jozavat\anomally\test train vald\

print(data_validation.shape)
print(data_validation[:5])

print("---------------------------------------")

data_test = read_dataset(r"F:\machine learning\jozavat\anomally\test train vald\test.c

print(data_test.shape)
print(data_test[:5])
```

```
(245, 2)
[[13.047 14.741]
 [13.409 13.763]
 [14.196 15.853]
 [14.915 16.174]
 [13.577 14.043]]
---------------------------------------
(31, 3)
[[13.025 14.251  0.   ]
 [14.534 15.766  0.   ]
 [13.252 16.323  0.   ]
 [13.237 15.337  0.   ]
 [12.13  12.667  0.   ]]
---------------------------------------
(31, 3)
[[14.053 13.939  0.   ]
 [15.309 16.042  0.   ]
 [13.155 16.921  0.   ]
 [12.699 13.999  0.   ]
 [14.368 16.758  0.   ]]
```

In [7]:

```
#using data validation

def select_threshold(p_valid, y_valid):      #probs : p
    best_epsilon = 0
    best_f1 = 0


    stepsize = (max(p_valid) - min(p_valid)) / 1000;
    epsilons = np.arange(min(p_valid), max(p_valid), stepsize)     #arange return ever
    for epsilon in np.nditer(epsilons):                           #  عضای یک ارایه را برمیگرداند

        # PREDICT OUTLIERS
        y_pred = (p_valid < epsilon)                              #که احتمالشون کوچکتر تز اپسیلونه

        # calculate TP, FP and FN
        tp = np.sum((y_pred == 1) & (data_validation [: ,2] == 1)) * 1.0
        fp = np.sum((y_pred == 1) & (data_validation [: ,2] == 0)) * 1.0
        fn = np.sum((y_pred == 0) & (data_validation [: ,2] == 1)) * 1.0

        # calculate Precision, Recall and F1-score
        precision = tp / (tp + fp)
        recall    = tp / (tp + fn)
        f1 = (2 * precision * recall) / (precision + recall)

        if f1 > best_f1:
            best_f1 = f1
            best_epsilon = epsilon

    return best_f1, best_epsilon




# STEP 1: estimate parameters mu and sigma from X_val
mu_validation = np.mean((data_validation[ : , 0:2 ]), axis=0)
Sigma_val = np.cov((data_validation[ : , 0:2 ]).T)

# STEP 2: calculate probabilities
p_val = multivariate_normal(mean=mu_validation, cov=Sigma_val).pdf(data_validation[ :
```

```python
# STEP 3: choose best value for epsilon
f1, eps = select_threshold(p_val, data_validation [: ,2])
print("f1 = {:.2g}, epsilon = {}".format(f1, eps))
```

```
f1 = 0.57, epsilon = 0.022173060113477056
```

In [8]:
```python
np.argwhere(p_val<0.022173060113477056)
```

Out[8]:
```
array([[ 4],
       [ 6],
       [28],
       [29]], dtype=int64)
```

In [10]:
```python
plt.figure(figsize=(8, 4))

plt.subplot(121)
plt.scatter((data_validation[ : , 0:2 ])[p_val >= eps, 0], (data_validation[ : , 0:2 ]
plt.scatter((data_validation[ : , 0:2 ])[p_val <  eps, 0], (data_validation[ : , 0:2 ]
plt.title('Validation Data ($\epsilon$ = {:.5g})'.format(eps))

plt.subplot(122)
mu = np.mean(X_train, axis=0)
sigma = np.cov(X_train.T)
p = multivariate_normal(mean=mu, cov=sigma).pdf(X_train)
plt.scatter(X_train[p >= eps, 0], X_train[p >= eps, 1], s=15, marker='o', c='b', edgec
plt.scatter(X_train[p <  eps, 0], X_train[p <  eps, 1], s=50, marker='x', c='r', edgec
plt.title('Training Data ($\epsilon$ = {:.5g})'.format(eps))

plt.show()
```

Sklearn Novelty and Outlier Detection

In [34]:
```python
from sklearn.ensemble import IsolationForest
IF = IsolationForest()
IF.fit (X_train)      # b u i l d the t r e e s
a=IF.predict(data_test[ : , 0:2 ])
```

```
print(a)
print(data_test[ : , 2 ])
```

```
[ 1  1 -1 -1 -1  1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1 -1 -1 -1 -1 -1]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 1. 1. 1. 0.]
```

In [32]:

```
from sklearn.metrics import f1_score
f1_score(a, data_test[ : , 2 ] ,average= None )
```

Out[32]:

```
array([0., 0., 0.])
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: