# Machine Learning

**Pembelajaran mesin adalah cabang kecerdasan buatan (AI) yang difokuskan pada pembangunan aplikasi yang belajar dari data dan meningkatkan ketepatannya dari masa ke masa tanpa diprogramkan untuk melakukannya.**

## Supervised machine learning

**Pembelajaran mesin yang diselia melatih dirinya pada set data berlabel. Artinya, data dilabel dengan informasi yang model pembelajaran mesin sedang dibangun untuk menentukan dan bahkan mungkin diklasifikasikan dengan cara model tersebut seharusnya mengklasifikasikan data. Sebagai contoh, model penglihatan komputer yang direka untuk mengenal pasti anjing Gembala Jerman yang tulen mungkin dilatih menggunakan kumpulan data pelbagai gambar anjing berlabel.**

## Unsupervised machine learning

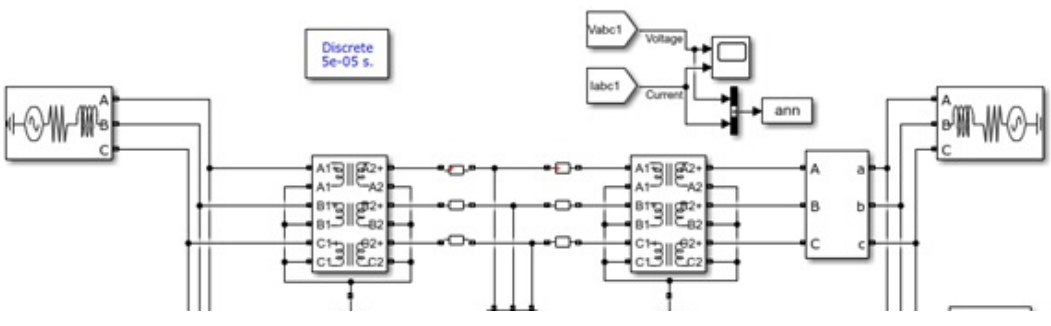
**Pembelajaran mesin yang tidak diawasi memakan data yang tidak berlabel — banyak dan banyak — dan menggunakan algoritma untuk mengekstrak ciri-ciri bermakna yang diperlukan untuk melabel, menyusun dan mengklasifikasikan data dalam masa nyata, tanpa campur tangan manusia. Pembelajaran tanpa pengawasan lebih kurang mengotomatisasi keputusan dan ramalan, dan lebih banyak mengenai mengenal pasti corak dan hubungan dalam data yang akan dilewatkan oleh manusia. Sebagai contoh, ikuti pengesanan spam - orang menghasilkan lebih banyak e-mel daripada yang mungkin diharapkan oleh pasukan saintis data untuk melabel atau mengklasifikasikannya sepanjang hayat mereka. Algoritma pembelajaran yang tidak diawasi dapat menganalisis sejumlah besar e-mel dan mengungkap ciri dan corak yang menunjukkan spam (dan terus menjadi lebih baik dalam menandai spam dari masa ke masa).**

# Pernyataan Masalah

**Saluran penghantaran adalah bahagian terpenting dalam sistem kuasa. Keperluan kuasa dan kesetiaannya telah berkembang secara eksponensial dalam era moden, dan peranan utama saluran transmisi adalah untuk menghantar kuasa elektrik dari kawasan sumber ke rangkaian pengedaran. Sistem kuasa elektrik terdiri daripada begitu banyak elemen dinamik dan interaksi yang kompleks yang selalu terdedah kepada gangguan atau kerosakan elektrik. Loji janakuasa elektrik berkapasiti tinggi dan konsep grid, iaitu, loji janakuasa elektrik yang disegerakkan dan grid yang dipindahkan secara geografi, diperlukan pengesanan kesalahan dan pengoperasian peralatan perlindungan dalam masa minimum yang mungkin agar tetap stabil. Kesalahan pada talian penghantaran sistem kuasa elektrik sepatutnya pertama kali dikesan dan dikelaskan dengan betul dan harus diselesaikan dalam masa yang paling sedikit. Sistem perlindungan yang digunakan untuk saluran transmisi juga dapat memulai relay lain untuk melindungi sistem kuasa dari pemadaman. Sistem pengesanan kerosakan yang teladan memberikan cara praktikal, boleh dipercayai, cepat, dan selamat dalam operasi penyampaian. Penerapan teknologi pengecaman corak dapat membantu membezakan sistem elektrik yang rosak dan sihat. Ini juga membolehkan kita membezakan antara tiga fasa sistem kuasa tiga fasa yang mengalami kerosakan.**

## Data

**Kami telah memodelkan sistem kuasa dalam MATLAB untuk mensimulasikan analisis kesalahan. Sistem kuasa terdiri daripada 4 penjana 11 × 10 ^ 3 V, setiap pasangan terletak di setiap hujung talian penghantaran. Transformer hadir untuk mensimulasikan dan mengkaji pelbagai kesalahan di titik tengah talian penghantaran.**



In [1]:

```
# !pip install lightgbm==2.0.4
# Mulakan dengan memuat turun data daripada kaggle https://www.kaggle.com/esathyaprakash/electrical-fault-detection-and-classification?select=detect_dataset.csv
```

# Jenis Masalah Yang Akan Diselesaikan

**Untuk bahagian 2 ini, kita akan menggunakan data saluran penghantaran untuk meramal sama ada saluran adalah rosak (1) atau tidak rosak (0). Berdasarkan penerangan yang diberikan di atas, ini merupakan classification problem di mana objektif penciptaan machine learning model ini adalah untuk meramal sama ada saluran adalah rosak atau tidak rosak berdasarkan pemboleh ubah yang lain seperti garis arus A -> C dan voltan talian A -> C.**

**Maka, untuk membina model ini, kita harus menggunakan algoritma yang dicipta untuk classification seperti Logistic Regression, Random Forest Classifier, Support Vector Machine dan Gradient Boosting Model.**

**Dalam tutorial ini, kita akan menggunakan Logistic Regression, Random Forest Classifier (Bagging) dan Gradient Boosting Model (Boosting). Di akhir tutorial kita akan bandingkan hasil ketiga-tiga model.**

In [2]:

```
# Mengimport kesemua modul yang diperlukan terdahulu
import pandas as pd # Untuk kegunaan manipulasi data
import numpy as np # Untuk kes penggunaan matematik
import seaborn as sns # Untuk data visualization (EDA)
import matplotlib.pyplot as plt # # Untuk data visualization (EDA)

from sklearn.linear_model import LogisticRegression # Linear classification model
from sklearn.ensemble import RandomForestClassifier # Bagging model
import lightgbm as lgb # Boosting model

from sklearn.metrics import classification_report # Untuk kegunaan penilaian ketepatan model

from scipy.stats import skew, kurtosis # Untuk pengiraan statistik

plt.rcParams['figure.figsize'] = (12,6)
```

In [3]:

```
# Membaca data ke dalam pemboleh ubah jenis DataFrame dengan menggunakan modul pandas dan method read_csv()
data = pd.read_csv("data_machine_learning_part2.csv")
```

In [4]:

```
# Melihat 5 barisan pertama di dalam data secara rawak
data.sample(5)
```

Out[4]:

| | Output (S) | Ia | Ib | Ic | Va | Vb | Vc | Unnamed: 7 | Unnamed: 8 |
|---|---|---|---|---|---|---|---|---|---|
| **667** | 1 | 362.402321 | 64.591400 | -45.048523 | -0.369445 | 0.435665 | -0.066220 | NaN | NaN |
| **10950** | 0 | 51.570789 | -14.527929 | -40.214858 | 0.301153 | 0.310917 | -0.612070 | NaN | NaN |
| **1219** | 1 | 291.733304 | -27.486758 | 47.649241 | 0.337104 | -0.605854 | 0.268750 | NaN | NaN |
| **233** | 0 | 35.698487 | 27.555589 | -63.254076 | -0.416508 | 0.589368 | -0.172860 | NaN | NaN |

| | Output (S) | Ia | Ib | Ic | Va | Vb | Vc | Unnamed: 7 | Unnamed: 8 |
|---|---|---|---|---|---|---|---|---|---|
| 804 | 1 | 477.305898 | -11.496187 | 42.852636 | 0.285488 | -0.591154 | 0.305666 | NaN | NaN |

In [5]:

```
# Hitung jumlah semua lajur untuk mendapatkan jumlah baris kosong
data.isna().sum()
```

Out[5]:

```
Output (S)         0
Ia                 0
Ib                 0
Ic                 0
Va                 0
Vb                 0
Vc                 0
Unnamed: 7     12001
Unnamed: 8     12001
dtype: int64
```

In [6]:

```
# Keluarkan lajur Unnamed: 7 dan Unnamed: 8 daripada data kerana ianya lajur yang tidak m
empunyai info
del data['Unnamed: 7']
del data['Unnamed: 8']
```

In [7]:

```
data.isna().sum()
```

Out[7]:

```
Output (S)     0
Ia             0
Ib             0
Ic             0
Va             0
Vb             0
Vc             0
dtype: int64
```

In [8]:

```
data.describe()
```

Out[8]:

| | Output (S) | Ia | Ib | Ic | Va | Vb | Vc |
|---|---|---|---|---|---|---|---|
| count | 12001.000000 | 12001.000000 | 12001.000000 | 12001.000000 | 12001.000000 | 12001.000000 | 12001.000000 |
| mean | 0.457962 | 6.709369 | -26.557793 | 22.353043 | 0.010517 | -0.015498 | 0.004980 |
| std | 0.498250 | 377.158470 | 357.458613 | 302.052809 | 0.346221 | 0.357644 | 0.349272 |
| min | 0.000000 | -883.542316 | -900.526952 | -883.357762 | -0.620748 | -0.659921 | -0.612709 |
| 25% | 0.000000 | -64.348986 | -51.421937 | -54.562257 | -0.237610 | -0.313721 | -0.278951 |
| 50% | 0.000000 | -3.239788 | 4.711283 | -0.399419 | 0.002465 | -0.007192 | 0.008381 |
| 75% | 1.000000 | 53.823453 | 69.637787 | 45.274542 | 0.285078 | 0.248681 | 0.289681 |
| max | 1.000000 | 885.738571 | 889.868884 | 901.274261 | 0.609864 | 0.627875 | 0.608243 |

# EDA

**Di fasa ini kita akan menggunakan method seperti spearman correlation dan data visualization untuk mencari hubungan antara lajur-lajur dan lajur output yang akan digunakan sebagai `label` untuk latihan model**

In [9]:

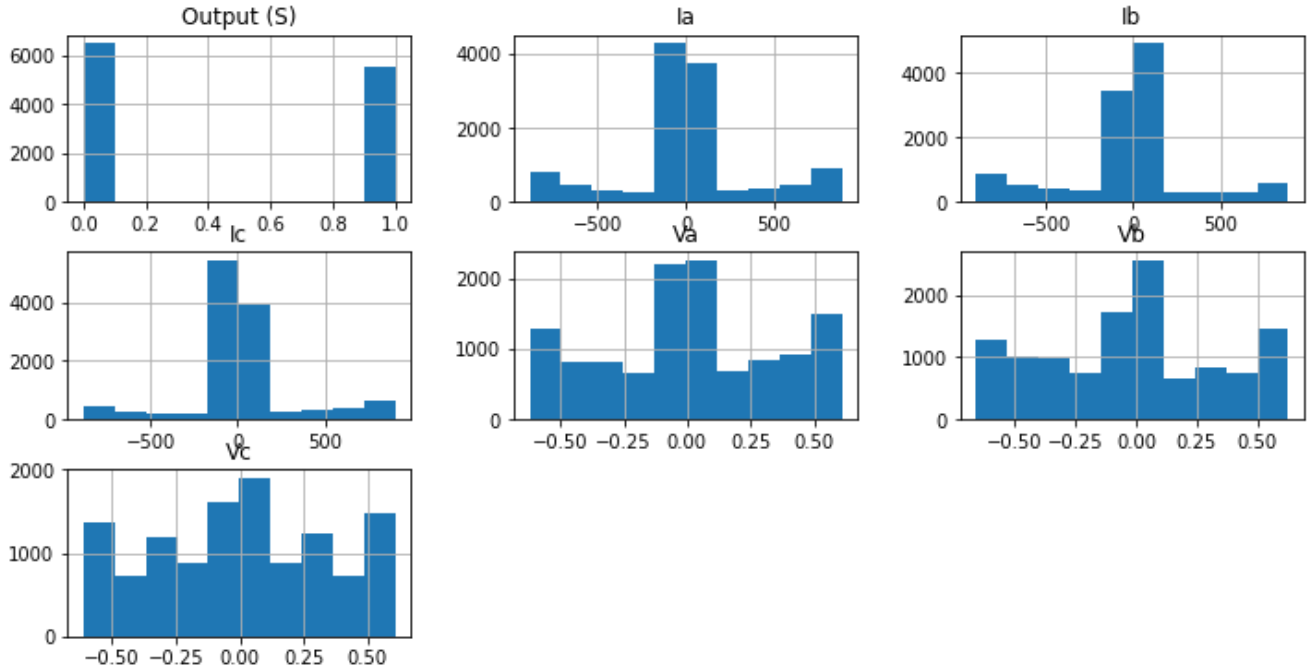
```
sns.heatmap(data.corr(), annot=True)
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1b632007d0>
```


In [10]:

```
data.hist()
plt.show()
```




In [11]:

```
data['Output (S)'].value_counts().plot(kind='bar') # Binary Classification
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1b47238150>
```


# Recap Skewness and Kurtosis

**Skewness adalah ukuran asimetri sebaran. Nilai ini boleh menjadi positif atau negatif.**

- **Kecenderungan negatif menunjukkan bahawa ekor berada di sebelah kiri taburan, yang memanjang ke arah nilai yang lebih negatif.**
- **Kecenderungan positif menunjukkan bahawa ekor berada di sebelah kanan taburan, yang memanjang ke arah nilai yang lebih positif.**
- **Nilai sifar menunjukkan bahawa tidak ada kecenderungan dalam pengedaran sama sekali, yang bermaksud pembahagiannya simetri dengan sempurna.**

**Kurtosis adalah ukuran sama ada atau tidak pengedaran adalah berat atau ekor ringan berbanding dengan taburan normal.**

- **Kurtosis taburan normal adalah 3.**
- **Sekiranya sebaran yang diberikan mempunyai kurtosis kurang dari 3, ia dikatakan playkurtic, yang bermaksud cenderung menghasilkan outliers yang lebih sedikit dan lebih ekstrem daripada taburan normal.**
- **Sekiranya sebaran yang diberikan mempunyai kurtosis lebih besar daripada 3, dikatakan leptokurtik, yang bermaksud cenderung menghasilkan lebih banyak penyimpangan daripada taburan normal.**

In [12]:

```
for col in data.columns:
  col_skew = skew(data[col].values, bias=False)
  col_kur = kurtosis(data[col].values, bias=False)
  print("Column {} : Skewness is {} and Kurtosis is {} \n".format(col, col_skew, col_kur)
)
```

```
Column Output (S) : Skewness is 0.16877124002514493 and Kurtosis is -1.9718449097542423

Column Ia : Skewness is 0.08245054541564444 and Kurtosis is 0.9026659342618357

Column Ib : Skewness is -0.2024416824307686 and Kurtosis is 1.25406115106827

Column Ic : Skewness is 0.30131107027235504 and Kurtosis is 2.795724135019788

Column Va : Skewness is -0.04369258404786632 and Kurtosis is -0.8404250600089109

Column Vb : Skewness is 0.08125974158015871 and Kurtosis is -0.9126162256402304

Column Vc : Skewness is -0.02737848111659492 and Kurtosis is -0.967196816066648
```

# Normalize Data

**Machine Learning berkerja lebih baik atau berkumpul lebih cepat apabila pelbagai ciri (pemboleh ubah) berada pada skala yang lebih kecil. Oleh itu, adalah amalan biasa untuk menormalkan data sebelum melatih model pembelajaran mesin di dalamnya.**

**Normalisasi juga menjadikan proses latihan kurang peka terhadap skala ciri. Ini menghasilkan pekali yang lebih baik selepas latihan.**

**Proses menjadikan ciri-ciri yang lebih sesuai untuk latihan dengan melakukan penyelamatan disebut sebagai skala fitur.**

**Formula untuk Normalisasi diberikan di bawah:**

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

In [13]:

```
# Menyatakan salinan data daripada pemboleh ubah data untuk menghasilkan data yang dinorm
alisasikan, apabila kita akan lakukan latihan model
# Kita akan menggunakan kedua-dua data ini untuk melihat adakah normalisasi akan menambah
baik prestasi model

data_norm = data.copy() # Salinan data

# Melakukan normalisasi ke atas semua lajur kerana ketiga-tiga lajur ini mempunyai varian
ce yang tinggi
data_norm['Ia_norm'] = ( data_norm['Ia'] - data_norm['Ia'].min() ) / ( data_norm['Ia'].m
ax() - data_norm['Ia'].min() )
data_norm['Ib_norm'] = ( data_norm['Ib'] - data_norm['Ib'].min() ) / ( data_norm['Ib'].m
ax() - data_norm['Ib'].min() )
data_norm['Ic_norm'] = ( data_norm['Ic'] - data_norm['Ic'].min() ) / ( data_norm['Ic'].m
ax() - data_norm['Ic'].min() )
data_norm['Va_norm'] = ( data_norm['Va'] - data_norm['Va'].min() ) / ( data_norm['Va'].m
ax() - data_norm['Va'].min() )
data_norm['Vb_norm'] = ( data_norm['Vb'] - data_norm['Vb'].min() ) / ( data_norm['Vb'].m
ax() - data_norm['Vb'].min() )
data_norm['Vc_norm'] = ( data_norm['Vc'] - data_norm['Vc'].min() ) / ( data_norm['Vc'].m
ax() - data_norm['Vc'].min() )

data_norm.describe()
```

Out[13]:

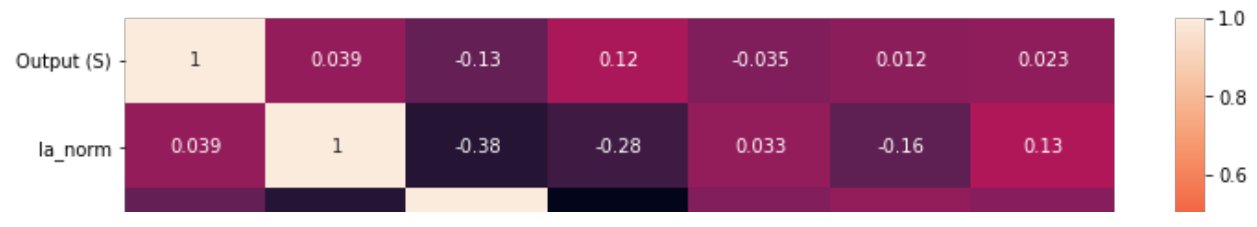
| | Output (S) | Ia | Ib | Ic | Va | Vb | Vc | Ia_norm | |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 12001.000000 | 12001.000000 | 12001.000000 | 12001.000000 | 12001.000000 | 12001.000000 | 12001.000000 | 12001.000000 | 120 |
| **mean** | 0.457962 | 6.709369 | -26.557793 | 22.353043 | 0.010517 | -0.015498 | 0.004980 | 0.503171 | |
| **std** | 0.498250 | 377.158470 | 357.458613 | 302.052809 | 0.346221 | 0.357644 | 0.349272 | 0.213170 | |
| **min** | 0.000000 | -883.542316 | -900.526952 | -883.357762 | -0.620748 | -0.659921 | -0.612709 | 0.000000 | |
| **25%** | 0.000000 | -64.348986 | -51.421937 | -54.562257 | -0.237610 | -0.313721 | -0.278951 | 0.463009 | |
| **50%** | 0.000000 | -3.239788 | 4.711283 | -0.399419 | 0.002465 | -0.007192 | 0.008381 | 0.497548 | |
| **75%** | 1.000000 | 53.823453 | 69.637787 | 45.274542 | 0.285078 | 0.248681 | 0.289681 | 0.529800 | |
| **max** | 1.000000 | 885.738571 | 889.868884 | 901.274261 | 0.609864 | 0.627875 | 0.608243 | 1.000000 | |

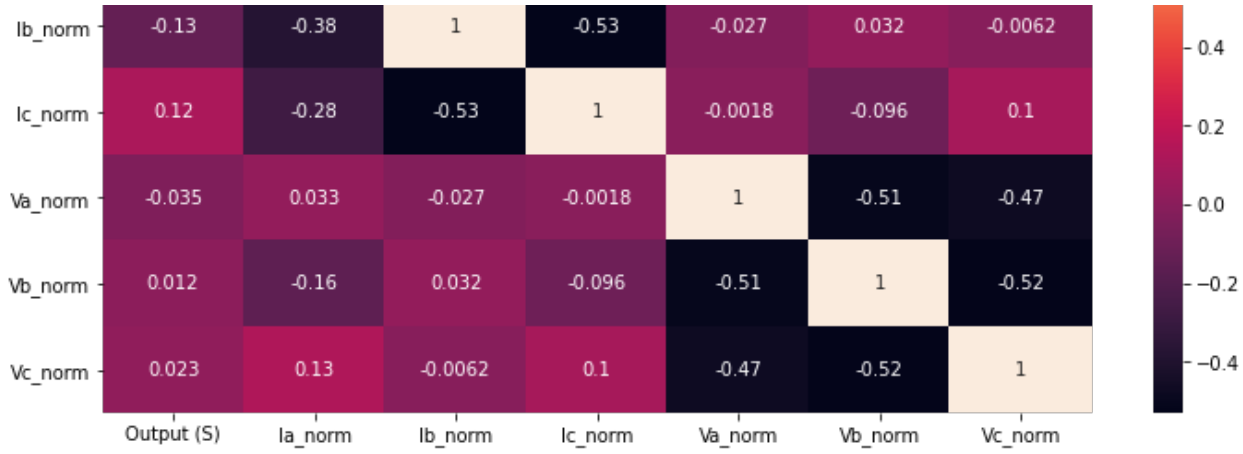
In [14]:

```
sns.heatmap(data_norm.drop(['Ia', 'Ib', 'Ic', 'Va', 'Vb', 'Vc'], axis=1).corr(), annot=T
rue)
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1b471f1050>
```






# Pembuatan Machine Learning

**Dalam fasa ini, kita akan melatih 3 algoritma berlainan untuk meramal apabila ada kerosakan pada saluran penghantaran.**

**Langkah pertama dalam pembinaan machine learning model adalah dengan mengasingkan data kepada train(latihan) dan test(ujian) set. Motivasi disebalik konsep ini adalah untuk melatih algoritma ke atas satu set data yang lain dan mengujinye ke atas set data yang model tidak pernah lihat.**

**Jika model berjaya untuk mendapatkan keputusan ramalan yang baik di kedua-dua set data, makan model tersebut boleh di gunakan dalam persekitaran pengeluaran.**

In [15]:

```
from sklearn.model_selection import train_test_split # Modul untuk mencipta train dan tes
t set
```

In [16]:

```
# Cara untuk membahagikan data kepada train dan test set
X = data[['Ia', 'Ib', 'Ic', 'Va', 'Vb', 'Vc']] # Pembolehubah bebas
y = data['Output (S)'] # Pembolehubah bersandar (To be predicted)

# Bermaksud 80% data akan digunakan untuk latihan dan 20% untuk ujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=4
2, shuffle=True)

print(X_train.shape, X_test.shape)

# Cara untuk membahagikan data kepada train dan test set
X_norm = data_norm[['Ia_norm', 'Ib_norm', 'Ic_norm', 'Va_norm', 'Vb_norm', 'Vc_norm']] #
Pembolehubah bebas
y_norm = data_norm['Output (S)'] # Pembolehubah bersandar (To be predicted)

# Bermaksud 80% data akan digunakan untuk latihan dan 20% untuk ujian
X_train_norm, X_test_norm, y_train_norm, y_test_norm = train_test_split(X_norm, y_norm,
test_size=0.20, random_state=42, shuffle=True)

print(X_train_norm.shape, X_test_norm.shape)
```

```
(9600, 6) (2401, 6)
(9600, 6) (2401, 6)
```

1. **(Accuracy) Ketepatan: bahagian jumlah ramalan yang betul.**
2. **(Precision) Nilai Ramalan Positif atau Ketepatan: bahagian kes positif yang dikenal pasti dengan betul.**
3. **(Negative Predictive Value) Nilai Ramalan Negatif: bahagian kes negatif yang dikenal pasti dengan betul.**
4. **(Recall) Sensitiviti atau Ingat semula: bahagian kes positif sebenar yang dikenal pasti dengan betul.**
5. **(Specificity)Kekhususan: bahagian kes negatif sebenar yang dikenal pasti dengan betul.**

In [17]:

```
# Model pertama : Logistic Regression

lr = LogisticRegression()
lr.fit(X_train, y_train)
pred = lr.predict(X_test)
print("Not Normalized Data")
print(classification_report(y_test, pred))

lr_norm = LogisticRegression()
lr_norm.fit(X_train_norm, y_train_norm)
pred_norm = lr_norm.predict(X_test)
print("Normalized Data")
print(classification_report(y_test, pred_norm))
```

```
Not Normalized Data
              precision    recall  f1-score   support

           0       0.67      1.00      0.81      1306
           1       1.00      0.42      0.60      1095

    accuracy                           0.74      2401
   macro avg       0.84      0.71      0.70      2401
weighted avg       0.82      0.74      0.71      2401

Normalized Data
              precision    recall  f1-score   support

           0       0.62      0.56      0.59      1306
           1       0.53      0.60      0.56      1095

    accuracy                           0.57      2401
   macro avg       0.58      0.58      0.57      2401
weighted avg       0.58      0.57      0.58      2401
```

In [18]:

```
# Model Kedua : Random Forest Regression (Bagging)

rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
pred = rfc.predict(X_test)
print("Not Normalized Data")
print(classification_report(y_test, pred))

rfc_norm = RandomForestClassifier()
rfc_norm.fit(X_train_norm, y_train_norm)
pred_norm = rfc_norm.predict(X_test)
print("Normalized Data")
print(classification_report(y_test, pred_norm))
```

```
Not Normalized Data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1306
           1       1.00      1.00      1.00      1095

    accuracy                           1.00      2401
   macro avg       1.00      1.00      1.00      2401
weighted avg       1.00      1.00      1.00      2401

Normalized Data
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1306
           1       0.46      1.00      0.63      1095

    accuracy                           0.46      2401
   macro avg       0.23      0.50      0.31      2401
weighted avg       0.21      0.46      0.29      2401
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
```

```
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [19]:

```
# Model Ketiga : Gradient Boosting (Boosting)

# Cara untuk menyatakan model Light Gradient Boosting Model
params={}
params['learning_rate']=0.03
params['boosting_type']='gbdt' #GradientBoostingDecisionTree
params['objective']='binary' #Binary target feature
params['metric']='binary_logloss' #metric for binary classification
params['max_depth']=10

train_data = lgb.Dataset(X_train, label=y_train)

# Cara untuk melatih model Light Gradient Boosting Model
num_round = 1000
model = lgb.train(params, train_data, num_round)

# Cara untuk mendapatkan ramalan
pred = model.predict(X_test)
pred=pred.astype(int) # Menjadikan output ramalan kepada 0 dan 1 (0.5 < akan menjadi 0 da
n 0.5 > akan menjadi 1)

print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       0.54      1.00      0.70      1306
           1       0.00      0.00      0.00      1095

    accuracy                           0.54      2401
   macro avg       0.27      0.50      0.35      2401
weighted avg       0.30      0.54      0.38      2401
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Undefined
MetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

# Kesimpulan

**Melalui latihan ini, kita telah mendapati bahawa untuk objektif ini, algoritma terbaik adalah Random Forest Classifier (Bagging) model. Ini kerana model tersebut berjaya memperoleh ketepatan yang terbaik di antara ketiga-tiga model yang telah dilatih. Kadang-kadang data yang di normalisasikan akan memberikan impak yang negative terhadap latihan model, hal ini kerana, normalisasi boleh menjadi punca hilangnya info di dalam data, tetapi 80% masa data yang di normalisasi akan memberikan impak positive.**

**Dalam activity ini, kite telah berlajar cara untuk membaca data, membuat analisa asas, membuat EDA, dan kemudian melatih dan menilai tahap kecergasan model. Dengan ini, anda susah mempunyai kemahiran asas untuk membuat data analytics dan machine learning development.**

In [19]: