

TUGAS 1

ALGORITMA MACHINE LEARNING

PRAKTIKUM KOMPUTASI STATISTIKA 2



Yogyakarta, 26 Mei 2022

Nama : Ahmad Habib Hasan Zein

NIM : 20/462305/PA/20277

Dosen Pengampu : 1. Prof. Dr.rer.nat. Dedi Rosadi, M.Sc.

Asisten Praktikum : 1. M. Adelft Ramadhan (18/430371/PA/18884)

LABORATORIUM KOMPUTASI MATEMATIKA DAN

STATISTIKA DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS GADJAH MADA

2022

## **BAB I**

### **LATAR BELAKANG**

Analisis Klasifikasi merupakan salah satu metode untuk mengelompokkan atau mengklasifikasikan data yang disusun secara sistematis, permasalahan klasifikasi sering dijumpai dalam sehari-hari baik di bidang akademik, sosial, pemerintah, dan lain-lain. Masalah ini muncul ketika ada sejumlah ukuran yang terdiri dari beberapa kategori yang tidak dapat diukur secara langsung tetapi harus melalui pengelompokan.

Regresi logistik merupakan bentuk klasifikasi yang paling dasar, dan sering digunakan. regresi logistik merupakan suatu teknik analisis data dalam statistika yang bertujuan untuk mengetahui hubungan antara beberapa variabel dimana variabel responnya adalah bersifat kategorik, baik nominal maupun ordinal dengan variabel penjelasnya dapat bersifat kategorik atau kontinu. Regresi logistik biner digunakan saat variabel respon merupakan variabel dikotomis (kategorik dengan 2 macam kategori). Dengan mempelajari algoritma regresi logistik biner, diharapkan dapat menjadi dasar ide algoritma klasifikasi lain. Sehingga mempermudah untuk mempelajari algoritma analisis klasifikasi lain

## BAB II

### PEMBAHASAN

1. Jelaskan cara kerja algoritma/model *machine learning* yang dipilih, penjelasan dapat dilakukan menggunakan tahapan-tahapan cara kerja maupun flow chart, kemudian jelaskan rinciannya setiap bagian-bagiannya.

**Jawab :**

**Cara kerja dari algoritma regresi logistik adalah**

#### **1) Melakukan Standarisasi**

Sebelum memulai algoritma machine learning ada baiknya untuk melakukan standarisasi dari masing masing variabel.

$$X_{i(std)} = \frac{X_i - \mu}{\sigma}$$

Dengan :

#### **2) Fungsi Aktivasi Sigmoid**

Dalam pembentukan algoritma *machine learning* untuk regresi logistik, hal yang pertama kali perlu diperhatikan adalah *activation function* yang digunakan. Karena kita akan melakukan klasifikasi biner menggunakan regresi logistik. Dimana nilai dari y atau variabel dependen nya adalah antara 0 atau 1. Maka fungsi aktivasinya akan berupa Sigmoid yang dituliskan dengan persamaan sebagai berikut :

$$g(z) = \frac{1}{1 + e^{-z}}$$

Dimana :

$$z = (C + W_i X_i)$$

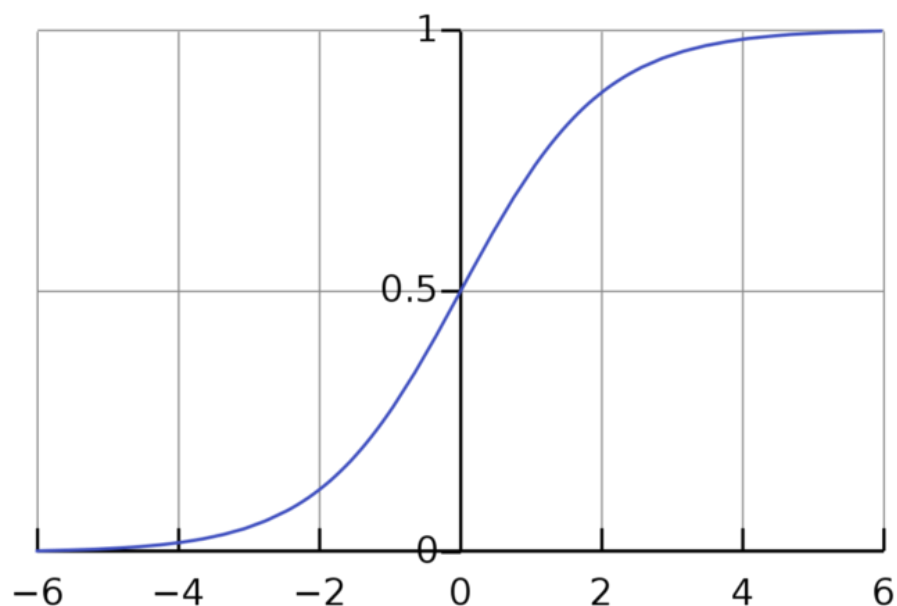
$$i = 1, 2, 3, \dots, n$$

Dengan :

C = konstanta persamaan

W<sub>i</sub> = koefisien dari variabel X ke i

Apabila dibentuk plot akan sebagai berikut :



### 3) Menghitung nilai logloss dari loss function

Untuk setiap algoritma *machine learning* kita membutuhkan *loss function*. Dimana kita ingin meminimalkan *loss function* tersebut untuk mendapatkan parameter yang optimal. Untuk permasalahan klasifikasi biner, kita harus dapat mencari tau peluang  $y$  bernilai 1, dan peluang  $y$  bernilai 0. Maka dari itu kita misalkan peluang  $y$  bernilai satu adalah  $y\_hat$  dan peluang  $y$  bernilai 0 adalah  $1-y\_hat$ . Dimana apabila dituliskan dalam persamaan akan menjadi :

$$P(y = 1|X ; w, c) = y\_hat$$

$$P(y = 0|X ; w, c) = 1 - y\_hat$$

Dari asumsi yang kita lakukan, kita dapat menghitung fungsi loglikelihood dari parameter menggunakan dua persamaan di atas dan dapat menentukan loss function yang harus kita minimalkan. Berikut ini adalah *Binary Cross-Entropy Loss* atau *Log Loss Function*

$$logloss_i = -[y_i \log(y\_hat_i) + (1 - y_i) \log(1 - y\_hat_i)]$$

$$Logloss = \frac{1}{m} \sum_{i=1}^m logloss_i$$

$$Logloss = \frac{1}{m} \sum_{i=1}^m -[y_i \log(y\_hat_i) + (1 - y_i) \log(1 - y\_hat_i)]$$

Dengan melihat nilai loss function kita dapat melihat loss mendekati 0 ketika hasil prediksi tepat, yaitu ketika  $y=0$  dan  $y\_hat=0$  atau,  $y=1$  and  $y\_hat=1$  dan loss mendekati tak hingga ketika hasil prediksi tidak tepat.

### 4) Menentukan gradien dari loss koefisien dan loss konstanta

Setelah mendapatkan loss function, yang harus kita lakukan selanjutnya adalah menentukan nilai parameter yang optimal menggunakan algoritma *Gradient Descent*.

$$w_{t+1} = w - lr * \frac{dLogloss}{dw}$$

$$c = c - lr * \frac{dLogloss}{dc}$$

Dimana :

$$\frac{dLogloss}{dw} = \frac{1}{m} * (y\_hat - y) * X$$

$$\frac{dLogloss}{dc} = \frac{1}{m} * (y\_hat - y)$$

### 5) Melakukan Iterasi

Setelah didapatkan fungsi untuk menghitung gradient loss dari koefisien dan konstanta, proses 3) dan 4) diulangi sebanyak  $n$ . Dilakukan perulangan sebanyak  $n$  kali untuk mengupdate parameter koefisien dan konstanta, serta mengupdate nilai loss function guna mendapatkan parameter koefisien dan konstanta yang optimal. Dalam machine learning juga dikenal dengan istilah batch size, yang mana menunjukkan berapa bagian data yang digunakan untuk setiap proses iterasi, misalkan di definisikan batch size 2 maka data akan dibagi menjadi 2. Misal ukuran data adalah 100 maka untuk setiap iterasi akan digunakan 50 data saja.

#### 6) Melakukan prediksi terhadap data

Setelah didapatkan nilai parameter koefisien dan konstanta yang optimal, Langkah selanjutnya yang perlu dilakukan adalah melakukan prediksi terhadap data variabel dependen ( $y_{\text{hat}}$ ) dengan menggunakan data variabel independent. Karena pada klasifikasi biner hanya terdapat dua output 0 dan 1 maka :

- Jika  $y_{\text{hat}} \geq 0.5$  maka akan dibulatkan menjadi 1
- Jika  $y_{\text{hat}} < 0.5$  maka akan dibulatkan menjadi 0

#### 7) Melakukan evaluasi dari hasil prediksi

Yang akan dilakukan selanjutnya adalah melakukan evaluasi dari hasil prediksi yang telah didapatkan yaitu dengan menghitung nilai prediksi yang sama dengan nilai sebenarnya dibagi dengan Panjang data.

$$\text{Akurasi} = \frac{\text{Jumlah prediksi benar}}{\text{jumlah data keseluruhan}}$$

2. Buat algoritma/model machine learning dari scratch, jelaskan bagian-bagian dari code tersebut.

- Fungsi Standarisasi

**Syntax :**

**# Membuat fungsi Standarisasi**

**def standarisasi(X):**

**# X = data variabel independen.**

**# m = jumlah data variabel independen**

**# n = jumlah variabel independen**

**m, n = X.shape**

**# iterasi untuk Standarisasi data variabel independen.**

**for i in range(n):**

**X = (X - X.mean(axis=0))/X.std(axis=0)**

**return X**

**Penjelasan :**

Pada fungsi ini setiap data pada variabel independent akan dilakukan standarisasi yaitu mengurangi dengan mean dari masing masing variabel, lalu membaginya dengan standar deviasi masing masing variabel. Proses ini dilakukan dengan iterasi

dari variabel independent pertama sampai variabel independen terakhir sampai seluruh data terstandarisasi

- Fungsi Aktivasi Sigmoid

**Syntax :**

**# Fungsi Aktivasi Sigmoid**

```
def sigmoid(z):  
    return 1.0/(1 + np.exp(-z))
```

**Penjelasan :**

Pada fungsi ini mendefinisikan fungsi aktivasi sigmoid yang digunakan dalam analisis regresi logistik, yaitu berdasarkan persamaan berikut dan mengembalikan nilainya :

$$g(z) = \frac{1}{1 + e^{-z}}$$

Dimana :

$$z = (C + W_i X_i)$$

$$i = 1, 2, 3, \dots, n$$

Dengan :

C = konstanta persamaan

W<sub>i</sub> = koefisien dari variabel X ke i

- Loss Function menghitung nilai los

**Syntax :**

**# Membuat Loss Function untuk menghitung nilai loss**

```
def loss(y, y_hat):  
    loss = -np.mean(y*(np.log(y_hat)) - (1-y)*np.log(1-y_hat))  
    return loss
```

**Penjelasan :**

pada fungsi ini menghitung nilai dari loss function yang akan diminimumkan dengan parameter y adalah data variabel dependen dan y\_hat adalah data prediksi variabel dependen, lalu mengembalikan nilainya. Loss function didapatkan melalui rumus yang telah dibahas diatas.

- Menghitung gradien dari loss parameter

**Syntax :**

**#menghitung gradien dari loss koefisien dan konstanta**

```
def gradients(X, y, y_hat):  
    # X = data variabel independen  
    # y = data variabel dependen  
    # y_hat = data prediksi variabel dependen  
    # w = koefisien variabel independen  
    # c = konstanta
```

```

# m = jumlah data variabel independen
m = X.shape[0]
# Gradien Loss Koefisien
dw = (1/m)*np.dot(X.T, (y_hat - y))
# Gradien Loss Konstanta
dc = (1/m)*np.sum((y_hat - y))
return dw, dc

```

### Penjelasan :

Pada fungsi ini dilakukan proses menghitung gradient loss dari parameter. Dw adalah gradient loss dari parameter koefisien dan dc adalah gradient loss dari parameter konstanta, yang didapatkan melalui rumus sebagai berikut :

$$w_{t+1} = w - lr * \frac{dLogloss}{dw}$$

$$c = c - lr * \frac{dLogloss}{dc}$$

Dimana :

$$dw = \frac{dLogloss}{dw} = \frac{1}{m} * (y_{hat} - y) * X$$

$$dc = \frac{dLogloss}{dc} = \frac{1}{m} * (y_{hat} - y)$$

dengan keterangan lainnya tertera pada *comment* di sintaks

- Fungsi melatih data dengan iterasi

### Syntax :

**# Membuat fungsi untuk melatih data**

```
def train(X, y, batch, iter, lr):
```

```
    # batch = Batch Size.
```

```
    # iter = jumlah iterasi.
```

```
    # lr = Learning rate.
```

```
    m, n = X.shape
```

```
    # Inisiasi Koefisien variabel dan Konstanta
```

```
    w = np.zeros((n,1))
```

```
    c = 0
```

```
    # Reshaping y
```

```
    y = y.reshape(m,1)
```

```
    # Standarisasi variabel independen
```

```
    x = standarisasi(X)
```

```
    # Empty list untuk menyimpan loss.
```

```
    losses = []
```

```
    # Training loop.
```

```
    for i in range(iter):
```

```

for i in range((m-1)//batch + 1):
    # Mendefinisikan Ukuran Data Yang digunakan setiap iterasi
    start_i = i*batch
    end_i = start_i + batch
    xb = X[start_i:end_i]
    yb = y[start_i:end_i]
    # menghitung data prediksi variabel dependen.
    y_hat = sigmoid(np.dot(xb, w) + c)
    # mendapatkan gradien loss dari koefisien dan konstanta.
    dw, dc = gradients(xb, yb, y_hat)
    # memperbaharui nilai parameter.
    w -= lr*dw
    c -= lr*dc
    # menghitung nilai loss dari setiap iterasi.
    l = loss(y, sigmoid(np.dot(X, w) + c))
    losses.append(l)
# mengembalikan nilai koefisien, konstanta and loss(List).
return w, c, losses

```

#### Penjelasan :

Pada fungsi ini layaknya algoritma machine learning lainnya, yaitu berguna untuk melatih data supaya didapatkan model yang optimal. Yaitu dengan mendefinisikan berapa banyak proses latih data diulang, atau yang biasa disebut dengan proses iterasi atau epoch, lalu Batch size yang digunakan, learning rate yang digunakan. Yang mana ketiga hal tersebut didapatkan melalui input parameter dari *user*. Setelah itu didefinisikan inisiasi koefisien dan konstanta sementara yaitu 0 dimana nanti akan diestimasi. Setelah itu dilakukan proses iterasi sebanyak *n* bilangan yang telah di definisikan sebelumnya dalam parameter dimana berfungsi untuk memperbarui nilai parameter sehingga didapatkan hasil yang optimal, lalu nilai loss di simpan dalam bentuk list berdasarkan banyak iterasi. Penjelasan lain fungsi ini sudah tertera dalam *comment* pada sintaks.

- Fungsi Memprediksi variabel dependen

#### Syntax :

# Mendefinisikan fungsi untuk memprediksi variabel dependen

def predict(X):

```

    # Standarisasi data variabel independen.
    x = standarisasi(X)
    # menghitung data prediksi variabel dependen.
    preds = sigmoid(np.dot(X, w) + c)
    # Empty List untuk menyimpan hasil prediksi.
    pred_class = []
    # jika y_hat >= 0.5 y dibulatkan ke 1
    # jika y_hat < 0.5 y dibulatkan ke 0
    pred_class = [1 if i > 0.5 else 0 for i in preds]
    # mengembalikan hasil prediksi

```



```
return np.array(pred_class)
```

**Penjelasan :**

Setelah didapatkan nilai parameter yang optimal selanjutnya dilakukan proses prediksi untuk variabel dependen, dimana prosesnya adalah menstandarisasi data terlebih dahulu, lalu memprediksi nilai berdasarkan fungsi aktivasi sigmoid, dengan nilai parameter koefisien dan konstanta telah didapatkan sebelumnya melalui proses latih data. Lalu melakukan pembulatan dimana jika  $y_{\text{hat}} \geq 0.5$  y dibulatkan ke 1 jika  $y_{\text{hat}} < 0.5$  y dibulatkan ke 0 dan mengembalikan nilainya. Penjelasan lain fungsi ini sudah tertera dalam *comment* pada sintaks.

- Fungsi Menghitung Akurasi

**Syntax :**

```
# Mendefinisikan fungsi menghitung akurasi
```

```
def accuracy(y, y_hat):
```

```
    # menghitung akurasi, yaitu banyaknya data benar dibagi panjang data
```

```
    accuracy = np.sum(y == y_hat) / len(y)
```

```
    return accuracy
```

**Penjelasan :**

Setelah dilakukan proses prediksi Langkah selanjutnya adalah menghitung akurasi dari model dan data prediksi yang telah didapatkan melalui algoritma regresi logistik. Untuk menghitung akurasi dari data prediksi digunakan rumus sebagai berikut :

$$\text{Akurasi} = \frac{\text{Jumlah prediksi benar}}{\text{jumlah data keseluruhan}}$$

3. Berikan contoh penggunaan dari algoritma/model yang telah dibuat, jelaskan hasilnya.

Untuk contoh penggunaan algoritma ini akan digunakan untuk memodelkan prediksi *breast cancer* dengan data set pada link : <https://drive.google.com/file/d/1iVkJQR1bxLFI-5bSa4sepgr8muCk6iAET/view?usp=sharing>

Langkah langkahnya akan dijelaskan sebagai berikut

- Menyiapkan data

**Sintaks :**

```
# Read data
```

```
data = pd.read_csv("/content/data.csv")
```

```
# Cek data
```

```
data.head()
```

```
# info dari data
```

```
data.info()
```

**Output :**

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030
5 rows x 33 columns							
	compactness_mean	concavity_mean	concave points_mean	...	texture_worst	perimeter_worst	area_worst
	0.27760	0.3001	0.14710	...	17.33	184.60	2019.0
	0.07864	0.0869	0.07017	...	23.41	158.80	1956.0
	0.15990	0.1974	0.12790	...	25.53	152.50	1709.0
	0.28390	0.2414	0.10520	...	26.50	98.87	567.7
	0.13280	0.1980	0.10430	...	16.67	152.20	1575.0
	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst	Unnamed: 32
	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890	NaN
	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902	NaN
	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758	NaN
	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300	NaN
	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678	NaN
<pre> &lt;class 'pandas.core.frame.DataFrame'&gt; RangeIndex: 569 entries, 0 to 568 Data columns (total 33 columns): #   Column                                Non-Null Count  Dtype ---  - 0   id                                    569 non-null    int64 1   diagnosis                            569 non-null    object 2   radius_mean                          569 non-null    float64 3   texture_mean                         569 non-null    float64 4   perimeter_mean                       569 non-null    float64 5   area_mean                           569 non-null    float64 6   smoothness_mean                      569 non-null    float64 7   compactness_mean                     569 non-null    float64 8   concavity_mean                       569 non-null    float64 9   concave points_mean                  569 non-null    float64 10  symmetry_mean                        569 non-null    float64 11  fractal_dimension_mean                569 non-null    float64 12  radius_se                             569 non-null    float64 13  texture_se                            569 non-null    float64 14  perimeter_se                          569 non-null    float64 15  area_se                              569 non-null    float64 16  smoothness_se                         569 non-null    float64 17  compactness_se                        569 non-null    float64 18  concavity_se                          569 non-null    float64 19  concave points_se                     569 non-null    float64 20  symmetry_se                           569 non-null    float64 21  fractal_dimension_se                  569 non-null    float64 22  radius_worst                          569 non-null    float64 23  texture_worst                         569 non-null    float64 24  perimeter_worst                       569 non-null    float64 25  area_worst                            569 non-null    float64 26  smoothness_worst                     569 non-null    float64 27  compactness_worst                     569 non-null    float64 28  concavity_worst                       569 non-null    float64 29  concave points_worst                  569 non-null    float64 30  symmetry_worst                        569 non-null    float64 31  fractal_dimension_worst               569 non-null    float64 32  Unnamed: 32                           0 non-null      float64 dtypes: float64(31), int64(1), object(1) memory usage: 146.8+ KB </pre>							

Terdapat 569 baris data, dengan jumlah variabel adalah 33, lalu karena variabel “Unnamed: 32” tidak berisi informasi (berisi NA), dan variabel “id” tidak berisi

informasi untuk memodelkan data maka langkah selanjutnya kita keluarkan data tersebut.

- Data Preprocessing

**Syntax :**

**# Membuang variabel bernama 'Unnamed: 32'**

**data = data.drop(['Unnamed: 32'], axis = 1)**

**# Membuang variabel bernama 'id'**

**data = data.drop(['id'], axis = 1)**

**# Check data**

**data.head()**

**Output :**

diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280

concavity_mean	concave points_mean	symmetry_mean	...	radius_worst	texture_worst	perimeter_worst
0.3001	0.14710	0.2419	...	25.38	17.33	184.60
0.0869	0.07017	0.1812	...	24.99	23.41	158.80
0.1974	0.12790	0.2069	...	23.57	25.53	152.50
0.2414	0.10520	0.2597	...	14.91	26.50	98.87
0.1980	0.10430	0.1809	...	22.54	16.67	152.20

area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst
2019.0	0.1622	0.6656	0.7119	0.2654	0.4601
1956.0	0.1238	0.1866	0.2416	0.1860	0.2750
1709.0	0.1444	0.4245	0.4504	0.2430	0.3613
567.7	0.2098	0.8663	0.6869	0.2575	0.6638
1575.0	0.1374	0.2050	0.4000	0.1625	0.2364

fractal_dimension_worst
0.11890
0.08902
0.08758
0.17300
0.07678

Dapat terlihat bahwa variabel “Unnamed: 32” dan “id” sudah dikeluarkan dari data, selanjutnya kita akan membuat variabel dependen nya dari variabel diagnosis dengan keterangan : M = malignant, B = benign, maka untuk M akan bernilai 1 dan B akan bernilai 0

**Syntax :**

```
# Membuat variabel data diagnosis sebagai variabel dependen
data.diagnosis = [1 if each == "M" else 0 for each in data.diagnosis]
# Check data.diagnosis
data.diagnosis.head()
```

**Output :**

```
0    1
1    1
2    1
3    1
4    1
Name: diagnosis, dtype: int64
```

Dapat terlihat bahwa data yang akan digunakan untuk variabel dependen sudah dalam bentuk nilai 0 dan 1 yaitu M bernilai 1 dan B bernilai 0. Selanjutnya kita akan membentuk variabel X sebagai variabel prediktor dan y sebagai variabel dependen yang akan kita gunakan untuk melatih data

**Syntax :**

```
# Membuat data X berisi variabel prediktor
X = data.loc[:, data.columns != 'diagnosis']
# Membuat variabel dependen
y = data.diagnosis
# Check data X
X.head()
# Check data y
y.head()
```

**Output :****Variabel X**

```
# Check data X
X.head()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	11.42	20.38	77.58	386.1	0.14250	0.28390
4	20.29	14.34	135.10	1297.0	0.10030	0.13280

concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean	...	radius_worst
0.3001	0.14710	0.2419	0.07871	...	25.38
0.0869	0.07017	0.1812	0.05667	...	24.99
0.1974	0.12790	0.2069	0.05999	...	23.57
0.2414	0.10520	0.2597	0.09744	...	14.91
0.1980	0.10430	0.1809	0.05883	...	22.54

texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst
17.33	184.60	2019.0	0.1622	0.6656	0.7119
23.41	158.80	1956.0	0.1238	0.1866	0.2416
25.53	152.50	1709.0	0.1444	0.4245	0.4504
26.50	98.87	567.7	0.2098	0.8663	0.6869
16.67	152.20	1575.0	0.1374	0.2050	0.4000

concave points_worst	symmetry_worst	fractal_dimension_worst
0.2654	0.4601	0.11890
0.1860	0.2750	0.08902
0.2430	0.3613	0.08758
0.2575	0.6638	0.17300
0.1625	0.2364	0.07678

## Variabel y

```
# Check data y
y.head()
```

```
0    1
1    1
2    1
3    1
4    1
Name: diagnosis, dtype: int64
```

Dapat terlihat bahwa telah terbentuk variabel X sebagai variabel prediktor dan y sebagai variabel dependen yang akan kita gunakan untuk melatih data. Selanjutnya kita mengubah format data menjadi numpy karena algoritma yang telah dibuat bekerja dalam format numpy

## Syntax :

# Merubah ke dalam bentuk array

**X = X.values**

**y = y.values**

# Check x

**X**

# Check y

y

Output :

```
# Check x
x
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]])

# Check y
y
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       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, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 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, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 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, 1, 1, 1, 1, 1, 0])
```

Dapat terlihat bahwa format data telah menjadi numpy karena algoritma yang telah dibuat bekerja dalam format numpy

- **Latih model dan akurasi**

Setelah dilakukan data preprocessing selanjutnya kita akan melatih algoritma untuk membentuk model yang optimal, dengan proses iterasi sebanyak 10000, batch size = 1, learning rate = 0.0001. lalu mengecek nilai koefisien, konstanta dan akurasi dari model dengan cara membandingkannya dengan data prediksi

Sintaks :

```

# melatih Algoritma terhadap data
w, c, l = train(X, y, batch=1, iter=10000, lr=0.0001)
# Check koefisien
print(w)
# Check Konstanta
print(c)
# Check Akurasi
accuracy(y, predict(X))

```

Output :

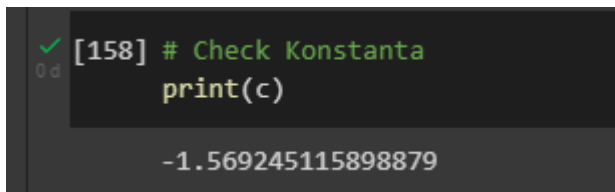
```

# Check koefisien
print(w)

[[-11.68863368]
 [ 0.34627353]
 [-16.4899809 ]
 [ 0.59176407]
 [ 0.33732876]
 [ 1.63602918]
 [ 2.32996945]
 [ 0.92926489]
 [ 0.50302526]
 [ 0.11775072]
 [-0.29540927]
 [-0.65797062]
 [ 2.08238987]
 [ 1.8453409 ]
 [ 0.04065618]
 [ 0.34475706]
 [ 0.50406588]
 [ 0.11481808]
 [ 0.14610503]
 [ 0.02850465]
 [-12.46740961]
 [ 5.95937391]
 [ 8.94152246]
 [ 0.32882636]
 [ 0.58593786]
 [ 4.92137936]
 [ 6.21971691]
 [ 1.66600738]
 [ 1.47951845]
 [ 0.46283829]]

```

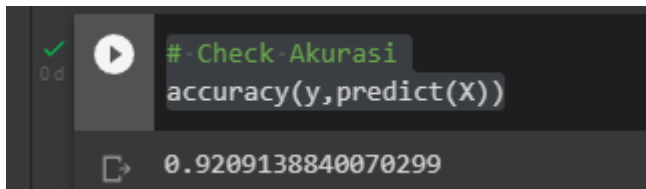
Didapatkan koefisien untuk setiap variabel nya seperti diatas,urut dari paling atas berarti Koefisien dari variabel  $x_1 = -116.31116007811198$ ,  $x_2 = 3.9633357161688125$ ,  $x_3 = -162.65943001992892$ ,  $x_4 = 5.373640940345474$ ,  $x_5 = 3.375005811777408$ , dst



```
[158] # Check Konstanta
print(c)

-1.569245115898879
```

Didapatkan nilai konstanta dari model regresi logistik yang diperoleh adalah sebesar = -1.569245115898879



```
# Check Akurasi
accuracy(y, predict(X))

0.9209138840070299
```

Dengan akurasi dari model sebesar = 0.9209138840070299 atau 92,09138840070299 % yang menunjukkan model memiliki akurasi yang tinggi untuk memprediksi *breast cancer* sehingga model layak digunakan

Karena output kurang menarik untuk dilihat maka akan dibuat fungsi untuk menampung fungsi untuk melatih algoritma dan mencari akurasinya lalu mengeluarkan output nya lebih terstruktur.

#### Syntax :

##### # Fungsi RegresiLogistik dan Output

##### def RegresiLogistik(X,y):

```
    w, c, l = train(X, y, batch=1, iter=10000, lr=0.0001)
    print("Koefisien dari variabel x1 = {0}, x2 = {1}, x3 = {2}, x4 = {3}, x5 = {4}"
          .format(w[0,0],w[1,0],w[2,0],w[3,0],w[4,0]))
    print("Koefisien dari variabel x5 = {0}, x7 = {1}, x8 = {2}, x9 = {3}, x10 = {4}"
          .format(w[5,0],w[6,0],w[7,0],w[8,0],w[9,0]))
    print("Koefisien dari variabel x11 = {0}, x12 = {1}, x13 = {2}, x14 = {3}, x15 = {4}"
          .format(w[10,0],w[11,0],w[12,0],w[13,0],w[14,0]))
    print("Koefisien dari variabel x16 = {0}, x17 = {1}, x18 = {2}, x19 = {3}, x20 = {4}"
          .format(w[15,0],w[16,0],w[17,0],w[18,0],w[19,0]))
    print("Koefisien dari variabel x21 = {0}, x22 = {1}, x23 = {2}, x24 = {3}, x25 = {4}"
          .format(w[20,0],w[21,0],w[22,0],w[23,0],w[24,0]))
    print("Koefisien dari variabel x26 = {0}, x27 = {1}, x28 = {2}, x29 = {3}, x30 = {4}"
          .format(w[25,0],w[26,0],w[27,0],w[28,0],w[29,0]))
    print("Konstanta dari persamaan logistik = {0}".format(c))
    print("Accuracy dari model = ",accuracy(y,predict(X)))
```

##### # Panggil Fungsinya

##### RegresiLogistik(X,y)

Output :



```
Koefisien dari variabel x1 = -11.68863367556556, x2 = 0.3462735255731345, x3 = -16.48998090273542, x4 = 0.5917640695320338, x5 = 0.33732876121901934
Koefisien dari variabel x6 = 1.636029175835698, x7 = 2.3299694454356263, x8 = 0.9292648927769442, x9 = 0.50302525806211, x10 = 0.11775071731644662
Koefisien dari variabel x11 = -0.2954092682850008, x12 = -0.6579706167054378, x13 = 2.082389874616541, x14 = 1.845340901513078, x15 = 0.04065617922751525
Koefisien dari variabel x16 = 0.34475706421046126, x17 = 0.504065877845447, x18 = 0.11481808087885616, x19 = 0.1461050273094117, x20 = 0.028504653849867948
Koefisien dari variabel x21 = -12.46740960666672, x22 = 5.9593739080797565, x23 = 8.941522456947931, x24 = 0.3288263583123243, x25 = 0.5859378564491273
Koefisien dari variabel x26 = 4.92137935869433, x27 = 6.219716914355003, x28 = 1.666007377005377, x29 = 1.4795184526488516, x30 = 0.462838287514781
Konstanta dari persamaan logistik = -1.569245115898879
Accuracy dari model = 0.9209138840070299
```

Dari data yang kita gunakan didapatkan persamaan regresi logistik dengan masing masing koefisien sebagai berikut :

- Koefisien dari variabel x1 = -11.68863367556556,
- Koefisien dari variabel x2 = 0.3462735255731345,
- Koefisien dari variabel x3 = -16.48998090273542,
- Koefisien dari variabel x4 = 0.5917640695320338,
- Koefisien dari variabel x5 = 0.33732876121901934
- Koefisien dari variabel x6 = 1.636029175835698,
- Koefisien dari variabel x7 = 2.3299694454356263,
- Koefisien dari variabel x8 = 0.9292648927769442,
- Koefisien dari variabel x9 = 0.50302525806211,
- Koefisien dari variabel x10 = 0.11775071731644662
- Koefisien dari variabel x11 = -0.2954092682850008,
- Koefisien dari variabel x12 = -0.6579706167054378,
- Koefisien dari variabel x13 = 2.082389874616541,
- Koefisien dari variabel x14 = 1.845340901513078,
- Koefisien dari variabel x15 = 0.04065617922751525
- Koefisien dari variabel x16 = 0.34475706421046126,
- Koefisien dari variabel x17 = 0.504065877845447,
- Koefisien dari variabel x18 = 0.11481808087885616,
- Koefisien dari variabel x19 = 0.1461050273094117,
- Koefisien dari variabel x20 = 0.028504653849867948
- Koefisien dari variabel x21 = -12.46740960666672,
- Koefisien dari variabel x22 = 5.9593739080797565,
- Koefisien dari variabel x23 = 8.941522456947931,
- Koefisien dari variabel x24 = 0.3288263583123243,
- Koefisien dari variabel x25 = 0.5859378564491273
- Koefisien dari variabel x26 = 4.92137935869433,
- Koefisien dari variabel x27 = 6.219716914355003,
- Koefisien dari variabel x28 = 1.666007377005377,
- Koefisien dari variabel x29 = 1.4795184526488516,
- Koefisien dari variabel x30 = 0.462838287514781
- Konstanta dari persamaan logistik = -1.569245115898879

Degan accuracy dari model adalah sebesar = 0.9209138840070299 atau 92,09138840070299 %

**BAB III**  
**LAMPIRAN**

**Link google colab :**

**<https://colab.research.google.com/drive/19PnE8yXF7kPh1890G323mqwTEKzV3mmD?usp=sharing>**

**Link dataset :**

**<https://drive.google.com/file/d/1iVkQR1bxLFl-5bSa4sepgr8muCk6iAET/view?usp=sharing>**