

LAPORAN TUGAS 2 : [INDIVIDU] LOGISTIC REGRESSION

Logistic Regression merupakan jenis supervised learning yang biasa digunakan untuk membuat sebuah model prediksi yang sama halnya dengan Linear Regresi. Bedanya ada pada nilai variabelnya jadi biasanya nilainya adalah ya/tidak, benar/salah, ataupun dalam bentuk bilangan biner 0/1.

Logistic Function

Output dari fungsi logistic turun menjadi $0 \leq y \leq 1$ Fungsi Logistik:

$$y = f(x) = \frac{1}{1 + e^{-x}}$$

Logistic Regression:

$$f(x; w) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d)}}$$

Accuracy

Ukuran kinerja yang paling sederhana untuk klasifikasi adalah menghitung prediksi yang benar dari semua data uji

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Confusion Matrix

Menghitung TP, TN, FP, dan FN

Positive – Negative adalah kelas

True – False adalah hasil prediksi yang sama dengan kelas target

- True Positive : jika data diprediksi kelas positif, dan targetnya adalah kelas positif
- True Negative : jika data diprediksi kelas negatif, dan targetnya adalah kelas negatif
- False Positive : jika data diprediksi kelas positif, dan targetnya adalah kelas negatif
- False Negative : jika data diprediksi kelas negatif, dan targetnya adalah kelas positif

Precision dan Recall

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{T}{TP + FN}$$

- Contoh Perhitungan Manual

Logistic Regression				
Data:				
No	Feature (x1)	Feature (x2)	Target (y)	
1	3	3	0	
2	1	2	0	
3	3	4	0	
4	1	2	0	
5	3	3	0	
6	8	3	1	
7	5	2	1	
8	7	2	1	
9	9	0	1	
10	8	4	1	
Epoch 1				
epoch = 1, += 1, w0 = 0, w1 = 0, w2 = 0, learnRate = 0.01, f(x) = 1/1+e-(w0+w1x+w2x) = ?, error = f(x)-y = -y				
	Ket :			w0 = w0 - learnRate . SUM(error)
	- epoch = 1	- learnRate = 0.01		w1 = w1 - learnRate . error . x =
	- w0 = 0	- f(x) = 1/1+e-(w0+w1x+w2x)		w2 = w2 - learnRate . error . x =
	- w1 = 0	- error = f(x) - y		
	- w2 = 0			
epoch = 1, += 2, w0 = -0.005, w1 = -0.015, w2 = -0.015, learnRate = 0.01, f(x) = 1/1+e-(w0+w1x+w2x) = ?, error = f(x)-y = -y				
	Ket :			w0 = w0 - learnRate . SUM(error)
	- epoch = 1	- learnRate = 0.01		w1 = w1 - learnRate . error . x =
	- w0 = -0.005	- f(x) = 1/1+e-(w0+w1x+w2x)		w2 = w2 - learnRate . error . x =
	- w1 = -0.015	- error = f(x) - y		
	- w2 = -0.015			

- Hasil uji coba dan Analisa

- Pada Epoch 1 data ke-1 diperoleh $w_0=-0.005$, $w_1=-0.015$ dan $w_2=-0.015$
- Pada Epoch 1 data ke-2 diperoleh $w_0=-0.009875$, $w_1=-0.019875$ dan $w_2=-0.247501$
- Begitu juga seterusnya sampai dengan epoch yang diinginkan

- Contoh Implementasi Python dan Hasil

```
import numpy as np
import matplotlib.pyplot as plt
```

```

class LogisticRegression:
    def __init__(self, learning_rate=0.01, epochs=1000):
        self.learning_rate = learning_rate
        self.epochs = epochs
        self.weights = None
        self.bias = None

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)
        self.bias = 0

        for epoch in range(self.epochs):
            for i in range(n_samples):
                linear_model = np.dot(X[i], self.weights) +
self.bias

                y_predicted = self._sigmoid(linear_model)

                error = y_predicted - y[i]
                dw = X[i] * error
                db = error

                self.weights -= self.learning_rate * dw
                self.bias -= self.learning_rate * db

                w = np.concatenate((np.array([self.bias]),
self.weights))

                print('Epoch-', epoch, ' Data:', X[i], ' Output:',
y_predicted, ' Error:', error, ' w:', w)

    def predict(self, X):
        linear_model = np.dot(X, self.weights) + self.bias
        y_predicted = self._sigmoid(linear_model)
        return np.round(y_predicted)

    def _sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    def _confusion_matrix(self, y_true, y_pred):
        y_pred = self.predict(X)
        tp = np.sum((y_true == 1) & (y_pred == 1))
        tn = np.sum((y_true == 0) & (y_pred == 0))
        fp = np.sum((y_true == 0) & (y_pred == 1))
        fn = np.sum((y_true == 1) & (y_pred == 0))
        return tp, tn, fp, fn

```

```

def precision_recall(self, X, y_true):
    y_pred = self.predict(X)
    tp, tn, fp, fn = self._confusion_matrix(y_true, y_pred)
    precision = tp / (tp + fp)
    recall = tp / (tp + fn)
    return precision, recall

def accuracy(self, X, y_true):
    y_pred = self.predict(X)
    tp, tn, fp, fn = self._confusion_matrix(y_true, y_pred)
    return (tp + tn) / (tp + tn + fp + fn)

def display_model(self, X, y):
    if X.shape[1] > 2:
        print("Tidak dapat menampilkan model dengan lebih dari 2
fitur")
        return

    x_values = X[:, 0]

    plt.scatter(x_values, y)
    plt.plot(x_values, self.predict(X), color='red')
    plt.show()

X = np.array([[3,3], [1,2], [3,4], [1,2], [3,3], [8,3], [5,2],
[7,2], [9,0], [8,4]])
Y = np.array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1])
logisticRegressionModel = LogisticRegression(epochs=2,
learning_rate=0.01)
logisticRegressionModel.fit(X, Y)
print("Weights:", logisticRegressionModel.weights)
print("Bias:", logisticRegressionModel.bias)
print("Predict:", logisticRegressionModel.predict(X))
print("Accuracy:", logisticRegressionModel.accuracy(X, Y))
print("Confusion Matrix:",
logisticRegressionModel._confusion_matrix(Y, X))
print("Precision Recall:",
logisticRegressionModel.precision_recall(X, Y))
logisticRegressionModel.display_model(X, Y)

```

```

Epoch- 0 Data: [3 3] Output: 0.5 Error: 0.5 w: [-0.005 -0.015 -
0.015]
Epoch- 0 Data: [1 2] Output: 0.4875026035157896 Error:
0.4875026035157896 w: [-0.00987503 -0.01987503 -0.02475005]
Epoch- 0 Data: [3 4] Output: 0.4579743089029386 Error:
0.4579743089029386 w: [-0.01445477 -0.03361426 -0.04306902]
Epoch- 0 Data: [1 2] Output: 0.4664985010531951 Error:
0.4664985010531951 w: [-0.01911975 -0.03827924 -0.05239899]
Epoch- 0 Data: [3 3] Output: 0.4277212600548746 Error:
0.4277212600548746 w: [-0.02339697 -0.05111088 -0.06523063]

```

```

Epoch- 0 Data: [8 3] Output: 0.3479696405155403 Error: -
0.6520303594844596 w: [-0.01687666 0.00105155 -0.04566972]
Epoch- 0 Data: [5 2] Output: 0.47428312525340877 Error: -
0.5257168747465912 w: [-0.01161949 0.02733739 -0.03515538]
Epoch- 0 Data: [7 2] Output: 0.5273306057840257 Error: -
0.4726693942159743 w: [-0.0068928 0.06042425 -0.025702 ]
Epoch- 0 Data: [9 0] Output: 0.6310969136775831 Error: -
0.36890308632241686 w: [-0.00320377 0.09362553 -0.025702 ]
Epoch- 0 Data: [8 4] Output: 0.6554296020881527 Error: -
0.3445703979118473 w: [ 0.00024193 0.12119116 -0.01191918]
Epoch- 1 Data: [3 3] Output: 0.5812867538956875 Error: -
0.5812867538956875 w: [-0.00557093 0.10375256 -0.02935778]
Epoch- 1 Data: [1 2] Output: 0.5098652346112613 Error: -
0.5098652346112613 w: [-0.01066959 0.09865391 -0.03955509]
Epoch- 1 Data: [3 4] Output: 0.5317252679725956 Error: -
0.5317252679725956 w: [-0.01598684 0.08270215 -0.0608241 ]
Epoch- 1 Data: [1 2] Output: 0.48627023093315014 Error: -
0.48627023093315014 w: [-0.02084954 0.07783945 -0.0705495 ]
Epoch- 1 Data: [3 3] Output: 0.5002550724460065 Error: -
0.5002550724460065 w: [-0.02585209 0.06283179 -0.08555715]
Epoch- 1 Data: [8 3] Output: 0.5548115415302464 Error: -
0.44518845846975363 w: [-0.02140021 0.09844687 -0.0722015 ]
Epoch- 1 Data: [5 2] Output: 0.5808907668600172 Error: -
0.41910923313998283 w: [-0.01720911 0.11940233 -0.06381932]
Epoch- 1 Data: [7 2] Output: 0.6661823574508454 Error: -
0.3338176425491546 w: [-0.01387094 0.14276957 -0.05714296]
Epoch- 1 Data: [9 0] Output: 0.7809233219014717 Error: -
0.21907667809852827 w: [-0.01168017 0.16248647 -0.05714296]
Epoch- 1 Data: [8 4] Output: 0.7426216895968462 Error: -
0.25737831040315384 w: [-0.00910639 0.18307673 -0.04684783]
Weights: [ 0.18307673 -0.04684783]
Bias: -0.009106387980436363
Predict: [1. 1. 1. 1. 1. 1. 1. 1. 1.]
Accuracy: 0.5
Confusion Matrix: (5, 0, 5, 0)
Precision Recall: (0.5, 1.0)

```

