**LAPORAN TUGAS 1 : [INDIVIDU] LINEAR (POLYNOMIAL) REGRESSION**

Linear Regression adalah salah satu algoritma yang digunakan data science dan tergolong pada algoritma Supervised Learning. Algoritma ini menggunakan prinsip regresi. Regresi membuat model prediksi untuk target variabel berdasarkan dari variabel bebasnya. Jenis algoritma ini sering digunakan untuk mencari hubungan antara variabel-variabel yang ada dan prediksinya. Sehingga pada linear regression, ditujukan untuk melakukan prediksi pada variabel terkait (y) berdasarkan variabel bebas yang diberikan (x). hasil akhirnya berupa hubungan linear antara variabel input (x) dengan variabel output (y).

Linear Regression dibedakan menjadi dua jenis berdasarkan jumlah variabel terikatnya. Pertama **Univariate Linear Regression** yaitu jumlah variabel terikatnya hanya ada satu (satu atribut/fitur). Kedua **Multivariate Linear Regression** yaitu jumlah variabel terikatnya lebih dari satu (beberapa atribut/fitur) menjadi satu nilai output (target) berupa nilai real.

Berikut Model yang dihasilkan dari kedua jenis Linear Regression:

1. **Univariate Linear Regression** (satu atribut/fitur)
2. **Multivariate Linear Regression** (beberapa atribut/fitur)

Keterangan:

* atau adalah variabel output
* adalah variabel input atau nilai atribut/fitur
* adalah bobot atau *coefficient*
* adalah bias atau intercept

Proses Training model Linear Regression dibedakan menjadi dua jenis yaitu **Direct/Matrix Equation** dan **Gradient Descent (Stochastic/Online Learning atau Batch/Offline Learning)**

1. **Direct/Matrix Equation**
2. Model Univariate Linear Regression

Linear Regression :

Bobot dan Bias dapat dihitung melalui persamaan **variants(x)** dan **covariants(x, y)**

Coefficient dan intercept dari model regresi linear adalah:

* Contoh Perhitungan Manual



* Hasil uji coba dan Analisa
* Mean x = 3
* Mean y = 2.8
* Variants(x) = 2.5
* Covariant(x, y) = 2
* W0 = 0.4
* W1 = 0.8
* Contoh Implementasi Python dan Hasil

import numpy as np

import matplotlib.pyplot as plt

class LinearRegressionDirectUnivariate:

def \_\_init\_\_(self):

self.weights = None

self.bias = None

def fit(self, X, y):

meanX = np.mean(X)

meanY = np.mean(y)

varX = np.var(X, ddof=1)

covXY = np.cov(X, y)

dataCov = covXY[0,1]

self.weights = dataCov / varX

self.bias = meanY - self.weights \* meanX

def predict(self, X):

return self.bias + self.weights \* X

def display\_model(self, X, y):

plt.plot(X, y, 'o')

dataX = np.linspace(np.min(X), np.max(X), 20)

dataY = self.bias + self.weights \* dataX

plt.plot(dataX, dataY, 'r-')

plt.show()

X = np.array([1, 2, 3, 4, 5])

Y = np.array([1, 3, 2, 3, 5])

linearRegressionModel = LinearRegressionDirectUnivariate()

linearRegressionModel.fit(X, Y)

print("Intercept:", linearRegressionModel.bias)

print("Koefisien:", linearRegressionModel.weights)

print("Mean X:", np.mean(X))

print("Covariance:", np.cov(X, Y)[0,1])

print("Variance:", np.var(X, ddof=1))

linearRegressionModel.display\_model(X, Y)

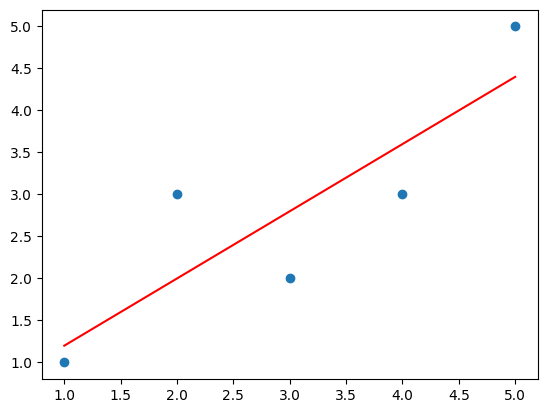
Intercept: 0.39999999999999947

Koefisien: 0.8

Mean X: 3.0

Covariance: 2.0

Variance: 2.5



1. **Gradient Descent**
2. Stochastic Gradient Descent (online learning)

* Contoh Perhitungan Manual



* Hasil uji coba dan Analisa
* Pada Epoch 1 data ke-1 diperoleh w0 = 0.01, w1 = 0.01 dan w2 = 0.02
* Pada Epoch 1 data ke-2 diperoleh w0 = 0.0391, w1 = 0.0682 dan w2 = 0.1073
* Contoh Implementasi Python dan Hasil

import numpy as np

import matplotlib.pyplot as plt

class LinearRegression:

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):

y\_predicted = np.dot(X[i], self.weights) + self.bias

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], ' Error:', error, ' w:', w)

def predict(self, X):

y\_predicted = np.dot(X, self.weights) + self.bias

return y\_predicted

def r\_squared(self, X, y):

y\_mean = np.mean(y)

y\_predicted = self.predict(X)

ss\_tot = np.sum((y - y\_mean)\*\*2)

ss\_res = np.sum((y - y\_predicted)\*\*2)

r2 = 1 - (ss\_res / ss\_tot)

return r2

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([[1, 2], [2, 3], [4, 5], [3, 4], [5, 1]])

Y = np.array([1, 3, 3, 2, 5])

linearRegressionModel = LinearRegression(epochs=2, learning\_rate=0.01)

linearRegressionModel.fit(X, Y)

print("Weights:", linearRegressionModel.weights)

print("Bias:", linearRegressionModel.bias)

print("R2 score:", linearRegressionModel.r\_squared(X, Y))

linearRegressionModel.display\_model(X, Y)

Epoch- 0 Data: [1 2] Error: -1.0 w: [0.01 0.01 0.02]

Epoch- 0 Data: [2 3] Error: -2.91 w: [0.0391 0.0682 0.1073]

Epoch- 0 Data: [4 5] Error: -2.1516 w: [0.060616 0.154264 0.21488 ]

Epoch- 0 Data: [3 4] Error: -0.6170719999999998 w: [0.06678672 0.17277616 0.23956288]

Epoch- 0 Data: [5 1] Error: -3.8297695999999997 w: [0.10508442 0.36426464 0.27786058]

Epoch- 1 Data: [1 2] Error: 0.025070208000000038 w: [0.10483371 0.36401394 0.27735917]

Epoch- 1 Data: [2 3] Error: -1.33506089472 w: [0.11818432 0.39071516 0.317411 ]

Epoch- 1 Data: [4 5] Error: 0.2680999395328003 w: [0.11550332 0.37999116 0.304006 ]

Epoch- 1 Data: [3 4] Error: 0.4715008049909759 w: [0.11078832 0.36584613 0.28514597]

Epoch- 1 Data: [5 1] Error: -2.7748350446559233 w: [0.13853667 0.50458789 0.31289432]

Weights: [0.50458789 0.31289432]

Bias: 0.13853666586852148

R2 score: 0.2786911841095945

A red line with blue dots

Description automatically generated

1. Batch Gradient Descent (batch learning

* Contoh Perhitungan Manual



* Hasil uji coba dan Analisa
* Pada Epoch 1 diperoleh w0 = 0.125, w1 = 0.36 dan w2 = 0.48
* Pada Epoch 2 diperoleh w0 = 0.22085, w1 = 0.6376625 dan w2 = 0.8380125
* Contoh Implementasi Python dan Hasil

import numpy as np

import matplotlib.pyplot as plt

class LinearRegression:

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):

y\_predicted = np.dot(X, self.weights) + self.bias

error = y\_predicted - y

dw = (1/n\_samples) \* np.dot(X.T, error)

db = (1/n\_samples) \* np.sum(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:', np.concatenate(X), ' Error:', error, ' w:', w)

def predict(self, X):

y\_predicted = np.dot(X, self.weights) + self.bias

return y\_predicted

def score(self, X, y):

y\_mean = np.mean(y)

y\_predicted = self.predict(X)

ss\_tot = np.sum((y - y\_mean)\*\*2)

ss\_res = np.sum((y - y\_predicted)\*\*2)

r2 = 1 - (ss\_res / ss\_tot)

return r2

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([[2, 4], [2, 5], [3, 1], [4, 5]])

Y = np.array([10, 12, 12, 16])

linearRegressionModel = LinearRegression(epochs=2, learning\_rate=0.01)

linearRegressionModel.fit(X, Y)

print("Weights:", linearRegressionModel.weights)

print("Bias:", linearRegressionModel.bias)

print("R2 score:", linearRegressionModel.score(X, Y))

linearRegressionModel.display\_model(X, Y)

Epoch- 0 Data: [2 4 2 5 3 1 4 5] Error: [-10. -12. -12. -16.] w: [0.125 0.36 0.48 ]

Epoch- 1 Data: [2 4 2 5 3 1 4 5] Error: [ -7.235 -8.755 -10.315 -12.035] w: [0.22085 0.6376625 0.8380125]

Weights: [0.6376625 0.8380125]

Bias: 0.22085

R2 score: -11.084485653733553

A graph with a red line and blue dots

Description automatically generated

Keterangan:

* adalah jumlah data dalam data training
* adalah prediksi data ke-i
* adalah target output untuk data ke-i
* adalah jumlah fitur
* adalah fitur ke-j
* adalah 1 bias/intercept
* adalah bobot untuk fitur ke-j

**Akurasi**

Perhitungan kinerja untuk mengukur seberapa akurat model yang dihasilkan untuk proses generalisasi. *R-Square* untuk mengukur model linear regresi.

Nilai *r-squared* antara 0 s.d 1, nilai 1 berarti ouput dapat diprediksi secara benar tanpa ada error.

Keterangan:

* adalah target data ke-i
* adalah mean target
* adalah jumlah data
* adalah data ke-i
* adalah prediksi untuk data ke-i