LAB-03

PAGE NO : DATE:

(a) Lineau Regnesion Deswing following linear Ragonesian problem using on mouti'x approach. 2 x: (week)

Q 4 y: (Salus in thousand)

import numpy as np

implementation

X=np. array ([[1,1],

[1,2]

[1,3].

Y=np, array ([2,4,5,9])

XT = X.T XTX = np. dot (XT.X)

XTX_inv = np.linalg.inv(XTX)

XTY = np, dot (XT, Y) beta = np. dot (XTX-inv, XTY)

beta-0, beta-1 = beta print (f"Intercept (BO): & beta_0: 0 2 f g")

Print(+" Shape (B1); { beta= 1:, 2 fg")

Y_pred = np. dot (x, beta)

print ('Predicted sales; ", y pred)

PAGE NO : DATE: output ? Intercept (Ba): -0.50 SLOPE (B1): 2.20 predicted sales : [1,7,3.9, 6.1, 8.3] Drubtiple Linear Regnesion The predict choose were of diabetes double osu" usin MLP. diabetes_docter. CSV .. Insulin D. Pedigne Fini cylucose. Age BIMI BP 150 25.3 50 81. 195 12.67 160. 29.1 implement ation import numpy as no impost pandas as pa.

dt = pd. read_csyl" diabetis_deda esv"). X = del [['Age', 'BMi', 'BP', 'Insulin', 'O. Pediqueefin']]. Values y = df [' Gucose 1]. Values x = np. c. Inp. ones (x. sharpe to]), x]

	PAGE NO: DATE:
	train_size = int (0.8 + len (x2)
	v to in v 10H - X Sedrun 5,2 ft x to
	V-train, V-test & Ylstran, Size) MArcin, Size;
	thata: npolinaly, inv (x-train of 6) x-train (a
	N-train , t a Y-train.
	Viprod = X-test @ theta.
	mse = np, mean ((y-test y_pred) + 2)
	79 = 1 - (nPosum((y-test - y-pred) # 12)/nPosum
	((Y-+43+ - np, mean(Y-+43+)) + 2))
-	Prind (" Multiple Lineau Regun model (Age, BMI, I, Dr. >61)
	Print (f" Mean squaed tror : Imsey")
	print (f" K12 Scote : 1 22 3")
	output!
	MLP 8-
	Mean Squared trop : 41. 2351233 65 46309

Mean Squand tres : 41.23512336576399 2^2 Score; 0.9312747 943903934

. W. K.

(E) togistic pagmesson. X = al [['Age', 'BML', 'BP', 'Insuliv', 'OPF']]. Value X = nP. C-InPrones(X. shapeto]), X] theta > nP. 2eros(X. shapeto]), X] theta > nP. 2eros(X. shapeto]) def compute cost (X, Y. theta); m = (enty) n = signord(n) + nP. sun(y + nP. log(n)+(1-y) and prodiend - descend (X, Y, theta) m = (enty) n = signord(X) n = signord(X) theta - descend (X, Y, theta) gradiend = (Ym) + (X, T) & (n-y)) theta - alpha + gradiend cost-nistory. apparal compute. cost (X, Y, theta)] resurn thuka, cost history alpha > 0.01 Sterations = 1000 theta, Cost-nistory : gradient, descent (X, Y, theta), alpha, ithata, alpha, ithata, itha		PAGE NO: DATE:
X = df[['Age', 'Bmc', 'Bp!, 'Engulin', 'OPF']]. value V = df[' Diabetus ']. Values X = np. C-Enp. ones(X. shapet of), X] theta > np. 2 eros(X. shapet of), X] theta > np. 2 eros(X. shapet of), X] def compute cost (X, Y, theta); m = (enty) n = sigmoid(x @) theta) vetern (-1/m) ones sun(y opply(n) + (1-y) s np. 100 (1-h)) ded gradiend-des and (X, Y, theta, alpha, ifrah); m = centy) n = sigmoid(X @) theta) gradiend = cym) o(Y, T @) (h-y) theta - alpha of gradiend cost-nistory, append (compute. cost (X, Y, theta)) referr ones append (compute. cost (X, Y, theta))		C Logistic Rognesion.
X = df ['Bmt', 'Bpt', 'Inalin', 'DPF']]. Value Y = df ['Dicubetus ']. Values X = np. C- Enp. ones (X. sneepet of), X] theta > np. 2 eros (X. sneepet of), X] theta > np. 2 eros (X. sneepet of) def (ompute cost (X, Y, theta); m = (enty) n = sigmord (a @) theta) vetern (-1/m) a np. sum (y a np. lbg(n)+(1-y) and produced descard (X, Y, theta, alpha, itrah); m = (enty) n = sigmord (X @) theta) gradient = cym) a (X. T @) (n-y) theta - alpha a gradient cost-nistory, append (compute cost (X, Y, theta)) referr theta, cost history alpha > 0.01 Pferations: 1000 theta, Cost-nistory: gradient, descent (Fiy,		
X = df ['Bmt', 'Bpt', 'Ingully', 'OPF']]. Value Y = df ['Dicubetus ']. Values X = np. C- Enp. ones (X. sneepet of), X] theta > np. 2 eros (X. sneepet of), X] theta > np. 2 eros (X. sneepet of) def compute cost (X, Y. theta); m = lenty) n = sigmord (a @ theta) vetern (-1/m) a np. sum (y a np. lbg(n)+(1-y) and produced descard (X, Y. theta, alpha, if rah); m = lenty) n = sigmord (X @ theta) gradient = cym) a (X. T @ (n-y)) theta - alpha a gradient cost-nistory, append (compute cost (X, Y. theta)) referr thuta, cost history alpha > 0.01 Pferations: 1000 theta, (ost-nistory: gradient, descent (X, Y.		
y = df [piabetes]. Nature x = np. C-Inp. ones (x. snapet of), x] theta > np. zeros (x. snapet of), x] def signord(2): return 1/(14 np. exp(-2)) def (ompute cost (x,y, theta): m = len(y) n=signord(x a) theta) return (-1/m) = np. log(n)+(1-y) snp. log(1-h)) ded gradiend-des and (x,y, theta, alpha, itrah): m = cen(y) n = signord(x a) theta) gradiend = cym) = (x, T a) (n-y) theta - alpha = gradiend cost-vistory, appard (compute-cost (x,y, theta)) return theta, cost history alpha > 0.01 Pserations= 1000 thata, Cost-history: gradiend desacrd (x,y, theta)		X = df [['Age', 'BME', 'BP!, 'Insulin', 'DPF']]. value
theta > np. C-Inp. ones (x. shapet of), x] theta > np. 2 eros (x. shapet of), x] def sigmoid (2): return 1/(1 + np. exp(-2)) def compute cost (x, y, theta); m = (ency) n = sigmoid (a a) theta) return (-1/m) * np. sum (y * np. log(n)+(1-y) Anp. 100 (1-h)) def gradient-des and (x, y, theta, alpha, ifrah); m = cen(y) n = sigmoid (x a) theta) gradient = cym) * (x, T a) (h-y) theta - alpha * gradient cost-nistory, appard (compute cost (x, y, theta)) return thata, cost history alpha > 0.01 sterations = 1000 thata, Cost-nistory = gradient desant (x, y,		
theta > NP. 2 eros (x. sneepecis) def sigmoid(2): return 1/11 + NP. exp(-2)) def compute cost (x, y, theta): m = (enly) N = sigmoid(n @ 4meta) return (-1/m) * nP. sum (y + nP. log(n) + (1-y) & nP. log (1-h)) def gradiend-descend (x, y, theta, alpha, itrah): m = cen(y) n = sigmoid (x @ + meta) gradiend = cym) + (x, T @ (n-y)) theta - alpha & gradiend cost-nistory, appard (compute cost (x, y, theda)) return + muta, cost history alpha > 0.01 sterations: 4000 thata, Cost-nistory : gradient descent (x, y, theda)	-	
def sigmoid(2): return 1/(1+np.exp(-2)) def (compute cost (x,y, theta): m = (en (y)) n = sigmoid(x @ theta) return (-1/m) & np. sum (y + np. log(n)+(1-y) & np. 10g(1-h)) def gradiend-descent (x,y, theta, alpha, ifrah): m = cen(y) n = sigmoid(x @ theta) gradient = (ym) + (x, T @ (h-y)) thefa - alpha & gradient cost-vistory, appard (computer cost (x,y, theta)) return thata, cost history alpha > 0.01 isterations: 1000 theta, (ost-history = gradient, descent (x,y, theta))	1	X=np.c-[np.ones(X.shapetoJ), X]
def sigmoid(2): return 1/(1+np.exp(-2)) def (compute cost (x,y, theta): m = (en (y)) n = sigmoid(x @ theta) return (-1/m) & np. sum (y + np. log(n)+(1-y) & np. 10g(1-h)) def gradiend-descent (x,y, theta, alpha, ifrah): m = cen(y) n = sigmoid(x @ theta) gradient = (ym) + (x, T @ (h-y)) thefa - alpha & gradient cost-vistory, appard (computer cost (x,y, theta)) return thata, cost history alpha > 0.01 isterations: 1000 theta, (ost-history = gradient, descent (x,y, theta))		
return 1/(1+ np. exp(-2)) def (ompute cost (x,y, theta); m = (enty) n = sigmord(r @) theta) return (-1/m) onlosum(y + np. log(n)+(1-y) sup. 10g(1-h)) def gradient-descent (x,y, theta, alpha, ifrah); m = centy) n = sigmoid(x@) theta) gradient = cym) + (x, t @) (h-y) theta - alpha of gradient cost-nistory, appural (computer cost (x,y, theta)) return oneta, cost history alpha > 0.01 sterations: tooo theta, Cost-nistory: gradient, descent(x,y,		theta >np. 2eros (x. snape (ij))
return 1/(1+ np. exp(-2)) def (ompute cost (x,y, theta); m = (enty) n = sigmord(r @) theta) return (-1/m) onlosum(y + np. log(n)+(1-y) sup. 10g(1-h)) def gradient-descent (x,y, theta, alpha, ifrah); m = centy) n = sigmoid(x@) theta) gradient = cym) + (x, t @) (h-y) theta - alpha of gradient cost-nistory, appural (computer cost (x,y, theta)) return oneta, cost history alpha > 0.01 sterations: tooo theta, Cost-nistory: gradient, descent(x,y,		
def (ompute cost (x, y, threta); m = len (y) n = sigmord (a @ threta) return (-1/m) & no sum (y + no, log(n) + (1-y) s no, log (1-h)) def gradiend-des and (x, y, threta, alpha, itrah); m = cen (y) n = sigmoid (x @ threta) gradient = (ym) + (x, t @ (h-y)) threta - alpha & gradient cost-nistory, appur of compute ~ cost (x, y, threta)] return threta, cost history alpha > 0.01 ? Jerahions = 1000 threta, Cost-nistory = gradient - desart (x, y,	-	-
m=(en(y) n=sigmord(r@)+ne+al) return (-1/m) + ne, sum (y + ne, log(n)+(1-y) & ne, log(1-h)) ded gradiend-des cent (x, y, theta, alpha, itrah): m=cen(y) n=sigmoid(x@)+(x, T@)(h-y)) thefa-alpha+gradient cost-nistory, append (computer cost (x, y, theta)) return + hata, cost history alpha > 0.01 ? derations: 1000 theta, Cost-nistory: gradient, desart (r, y,	-	return 1/11+ np. exp(-2)
m=(en(y) n=sigmord(r@)+ne+al) return (-1/m) + ne, sum (y + ne, log(n)+(1-y) & ne, log(1-h)) ded gradiend-des cent (x, y, theta, alpha, itrah): m=cen(y) n=sigmoid(x@)+(x, T@)(h-y)) thefa-alpha+gradient cost-nistory, append (computer cost (x, y, theta)) return + hata, cost history alpha > 0.01 ? derations: 1000 theta, Cost-nistory: gradient, desart (r, y,	+	
n=sigmoid(& a) theta) return (-1/m) & npo sum (y * npo log(n)+(1-y) A npolog (1-h)) et gradient-descent (x, y, theta, alpha, itrah); m=cen(y) n=sigmoid (x a) theta) gradient=(ym) + (x, ta) (h-y) theta-alpha & gradient cost-vistory, append (computer cost (x, y, theta)) return thata, cost history alpha > 0.01 sterations=too0 theta, cost-history=gradient-descent(x, y,	-	
return (-1/m) and sum (y and 10g(n)+(1-y) And, 10g(1-h)) def gradiend-descent (x, y, theta, alpha, itrah): m = con(y) n = sigmoid (x a) theta) gradient = (ym) + (x, T a) (h-y) thefa - alpha + gradient cost-nistory, appural (computer cost (x, y, theta)) return theta, cost history alpha > 0.01 iferations = tooo thata, Cost-nistory = gradient, descent (x, y,	-	A
and gradiend-descent (X, Y, theta, alpha, itrah): m = cen(4) n = sigmoid (X @ theta) gradient = Cym) + (X, T @ Ch-y) thefa - = alpha + gradient cost-nistory, append (computer cost (X, Y, thuta)) referr thata, cost history alpha > 0,01 ? Serations: 1000 theta, Cost-nistory: gradient, descent (X, Y,	-	n=sigmoid(ra) theta)
act gradiend-descent (x, y, theta, alpha, itrah)? m = cen(y) n = sigmoid (x a) theta) gradient = cym) + (x, T a) (h-y) theta - alpha + gradient cost-nistory, appural (computer cost (x, y, theta)) refer thata, cost history alpha > 0.01 ? Serations = 1000 thata, Cost-nistory = gradient descent (x, y,	+	
m = cen(4) n = sigmoid (x a theta) gradient = Cym) + (x, T a) (n-y) thefa - alpha + gradient cost-nistory, appard (computer cost (x, y, thuta)) referr thata, cost history alpha > 0.01 iferations= 4000 theta, Cost-nistery = gradient - desart (x, y,	+	rp, log(1-hJ)
m = cen(4) n = sigmoid (x a theta) gradient = Cym) + (x, T a) (n-y) thefa - alpha + gradient cost-nistory, appard (computer cost (x, y, thuta)) referr thata, cost history alpha > 0.01 iferations= 4000 theta, Cost-nistery = gradient - desart (x, y,	+	8/ 0
n = Sigmoid (x @ +heta) gradient = Cym) + (x, T @ (h-y)) +nefa = alpha + gradient cost-nistory, append (computer cost (x, y, thutal)) refurn +nefa, cost history alpha > 0.01 iferations= 4000 thata, Cost-nistery = gradient desart (x, y,		
gradient = (ym) + (x, T @ (h-y)) + nefa - alpha + gradient cost-nistory. append (computer cost (x, y, thutal)) refur + mutal, cost history alpha > 0.01 ? Serations = 1000 thata, Cost-nistery = gradient descent (x, y,		
thefa - alpha & gradient cost-nistory. append (computer cost (x14, threta)) referr under , cost history alpha > 0.01 isterations: 4000 theta, Cost-nistery = gradient. descent (x14,	-	n = sigmoid (x a) theta)
cost-nistory, append (computer cost (x14, thuted)) referr thata, cost history alpha > 0.01 isterations= tooo thata, Cost-nistery = graduent_desant(x14,	-	
reform 4nded, cost history alpha > 0.01 isterations= 4000 thata, cost-nistery = graduent_desant (r.y.	-	
alpha > 0.01 Prevations= 4000 thata, Cost-nistery = graduent descent (Fiy,	-	cost-nistory, appural computer cost (KIY, thutter)
alpha > 0.01 Prevations= 4000 thata, Cost-nistery = graduent descent (Fiy,	#	CP CLOCK MARK
isterations= 4000 thata, Cost-nistery = graduent_desant (r.y,	+	resure multa, cost history
isterations= 4000 thata, Cost-nistery = graduent_desant (r.y,	-	
thata, Cost-nistery : gradient desant (r. Y,		
	-	
frota, ayma, it was and I.	+	
	+	freta, ayma, it was as 1.

	PAGE NO : DATE :
	print (" Final fueta;", fineta)
	del predict (x, theta); return (sigmoid (x @) theta):>0.5) astype
	refer (5. green)
	predictions = predict (x, theta)
	accordey: information (Predictions = >4) \$ 100
	print(f" L P.A: Lacorancy; , 2f) vo ").
	oeuthers:
	To.0046,3.82019, -0.0030, +00843, -1.8403, -0.019
	0,00000 (3,0000)
	Accuracy = 100.00 y.
,	
	•
	•
1	