PYTHON PSEUDOKOD ZA LOGISTICKU REGRESIJU

```
eta = 0.1
train = df(open("train.csv"))
# tu jos morate odvojiti training data ili input vektore od targeta
test = df(open("test.csv"))
number_of_weights = test.shape()[1]
W=[]
no_of_animals = train.shape[0]
for n in range(number_of_weights):
	W.append(0.5)
#dakle pocetni W ce biti [0.5,0.5,...,0.5]
outputs = [] # ovo su svi y-kapice
errors = []

def sigma(z):
	return 1 / (1+(2.71**(-z)))

def forwardPass(tezine, inputVektor, target):
	res = sigma(dot(tezine, inputVektor))
	err = 0.5*((target - res)**2)
	outputs.append(res)
	errors.append(err)
	return [res, err]

def BackProp(tezine, inputVektor, y_kapica, target):
	updateaneTezine = []
	for m in range(len(tezine)):
		curr_w = tezine[m]
		parDer = - inputVektor[m] * y_kapica * (1 - y_kapica) * (target - y_kapica)
		updateanWeight = curr_w - eta * parDer
		updateaneTezine.append(updateanWeight)
	return updateaneTezine

def Train(trainSet, weights, targets, N=2): #N je broj epoha
	W = []
	for n in range N:
		for k in range(no_of_animals):
			r = forwardPass(weights, trainSet[k], targets[k])
			W.append(BackProp(weights, trainSet[k], r[0], tragets[k]))
	return W     #W je lista listi tezina kroz epohe. Na zadnju referiramo s W[-1]

def Predict(tezine, inputVektori):
	predicitions = []
	for k in inputVektori:
		res = sigma(dot(tezine, inputVektori[k]))
		predictions.append(res)
	return predictions



FINAL_W = Train(inputVektori, W, targeti, N=5)[-1]
PREDIKCIJE = Predict(FINAL_W,testVektori)
print(PREDIKCIJE)
```