Build KNN Classification model and SVM for a given dataset

Cab G		function kuncs
of knn Algorium		train, test = spit dolaset datases, 0.2)
the part opposition of the first terms of the first		E-3
THE CARE AND FOR SELECT OF SELECT	250 //	Inadichim c 07
-Tunchion split-daloset (dout, +es2 x	cho).	for each terr instance in test set
SHUSFIE stall randomly		predicted class = prodict (train,
Spit Indext leight (data) x1. 4	estarato)	testi, K)
train_sate first spil-make ele test set a remains elemants	water of	prediction appeal (predicted closs)
Penum fraisel, restest;	of data	Carried transfel
Control of the Control of the Control		Distriction policy
and a little of the part of the beautiful for	-	SVM
Junision get neighbours (How, tes	(1)	Initally anply list dataset
0 0		
olifere & EMPTY LIST	. 3	open file. at file name of c susceder
for gary trains in hair.		skip ku header row
APPENP (now i, dier) to die	2 stures	
APPENP (howi, dist) to dish	us .	for each row in coverador
SORT duranus by durana Case	custing 1	Extract label from last column:
		Extract labet from last course.
intern cirrof class labels from	7	the , set -1
neighbors.		the het - 1 Jeanures for simplicity appeared of features 1 Jeanure 2 graded to descent
at the state of the state of		the only first I dock as 2 (146el) to
· · · · · · · · · · · · · · · · · · ·		Coast general
Flenting fredict Chown, training):	return date set
and a contract to		
heighbors + get heighborschau,		Tunching JRAIN TEST SPLITE data, test ratio)
1e11: 1/1		surgle data randomly
Tate 4 IDUNT Occuration	7	
Each class 12 heighters		tano) dol) w(+-ters_
lehrn dass with highest	ble	1-64 Far Polit indep) as
		remus data Laiz Phitra inder as
10.1		Corte C
12 NOV CONTRACT		
not product ocums.		8 - first 2 Jeanns
		The state of the s
Junction TRAIN SUM C train, loan, 6, 10 mg 6, 10 mg weight weeker w= 10,00		preducted a crediction, w, 6)
Junchin TRAIN SUM Chain, loan, 6	nto	predicted a reduction
Initalize weight vector W= ED, O.	1 1000 10/16	buil achiet a predicted
inhalize bids 6=0.	1	· (nim of
The state of the s	M	
TO TO AND AND A STATE OF THE PARTY OF THE PA	1	The Property of the Party of th
for epoch in I be epochs:	0	
for gen gample in their:		The second of the first section
(etr= [fife)		colo la lastante
ial label	(1)	SULAT Nowell governot as 1 "
MVE.	LANGER	and the state of t
		special system to organ
compute = y = (POT- Product (w, x).	16)	terms of the second
y duian >=1:		a some of hard and while without
updict WCiJ=WEIT- Corn &	1 1000	a your by the task some
and the second of the second o	(5 X	
louble & Wijj	n war a say	when I have district pringers
elu:	" Unit live	citized point about the property
updata w ti 3 muti J-learn & (2	×	0
· · · · · · · · · · · · · · · · · · ·		
a lace de cont lambel will - 4 4 4 100	D T	A constant manual of Arabit
: a min update 6 = 6 + learn xy	179	Marinday Bridge at the
to along and a lotel ly		1 1000 2000 11
return w, b. 1 = 112 113		i souther to lost of the same
		Top some series and
The second second to the second second		driving of respond strengers
err liengnes Iris. CSV	The second second	
and (dataset 10AD (literant)		
10.114	(3) (3)	is low trestoy the year?
have lest star (datiset Li		No. of the contract of the con
call hain ter still (datiset, test		1817 A. 1821 A. 1821 A. 1821
call frain, lett- still (datiset, tes		a love all a division of the same
con hain, terr star (datiset, ten	Jugia-	a love state a start of the
COLL bain, wir Star (datiset, test	20-U1	But ald a series of the series
call frain, lett- still (datiset, tes	20-U1	

Code:

```
import matplotlib.pyplot as plt
#AGE, RESTING HEARTRATE, isAthlete
data = [
   [35, 85, 0],
   [40, 90, 0],
   [45, 88, 0],
   [25, 60, 1],
   [31, 70, 0],
    [19, 50, 1],
    [40, 80, 0]
def euclidean distance(p1, p2):
    return math.sqrt((p1[0]-p2[0])**2 + (p1[1]-p2[1])**2)
def knn_predict(data, test_point, k=3):
   distances = []
    for point in data:
        dist = euclidean distance(point, test point)
        distances.append((dist, point[2]))
   distances.sort(key=lambda x: x[0])
    k nearest = distances[:k]
   votes = \{0: 0, 1: 0\}
       votes[label] += 1
   predicted class = max(votes, key=votes.get)
    return predicted class
# Step 4: Predict test point
test point = [41, 81]
prediction = knn predict(data, test point, k=3)
print(f"Predicted Class for point {test point}: {prediction}")
```

```
class 0 = [point for point in data if point[2] == 0]
class 1 = [point for point in data if point[2] == 1]
# Extract values for plotting
age 0 = [point[0] for point in class 0]
hr 0 = [point[1] for point in class 0]
age 1 = [point[0] for point in class 1]
hr 1 = [point[1] for point in class 1]
# Plot with color grading
plt.figure(figsize=(8,6))
scatter0 = plt.scatter(age 0, hr 0, c='lightcoral', s=120,
label='Non-Athlete (Class 0)', edgecolors='black')
scatter1 = plt.scatter(age 1, hr 1, c='skyblue', s=120, label='Athlete
(Class 1)', edgecolors='black')
plt.xlabel("Age")
plt.ylabel("Resting Heart Rate (bpm)")
plt.title("Athlete Classification Based on Age and Resting HR")
plt.legend()
plt.grid(True)
plt.show()
```

import csv

```
import random
import matplotlib.pyplot as plt

# Load dataset

def load_dataset(filename):
    dataset = []
    with open(filename, 'r') as file:
        csv_reader = csv.reader(file)
        next(csv_reader) # skip header
        for row in csv_reader:
            features = list(map(float, row[1:-1])) # Assuming format: ID,

f1, f2, f3, f4, label
            label = 1 if row[-1] == 'Iris-setosa' else -1 # Binary

classification
            dataset.append(features[:2] + [label]) # Only use first 2

features for plotting
```

```
return dataset
# Split into training and test
def train test split(data, test ratio=0.2):
             random.shuffle(data)
             split_index = int(len(data) * (1 - test_ratio))
             return data[:split index], data[split index:]
def train svm(train set, learning rate=0.01, lambda param=0.01,
epochs=1000):
            w = [0, 0]
             for epoch in range (epochs):
                           for features in train set:
                                        x = features[:2]
                                        y = features[2]
                                         condition = y * (dot_product(w, x) + b) >= 1
                                         if condition:
                                                       w = [w[i] - learning rate * (2 * lambda param * w[i]) for
i in range(len(w))]
                                                       w = [w[i] - learning rate * (2 * lambda param * w[i] - y * lambda pa
x[i]) for i in range(len(w))]
def dot product(a, b):
             return sum([a[i] * b[i] for i in range(len(a))])
# Prediction
def predict(x, w, b):
             return 1 if dot_product(w, x) + b >= 0 else -1
# Plotting
def plot svm(train set, w, b):
             for point in train_set:
```

```
color = 'blue' if label == 1 else 'red'
       plt.scatter(x1, x2, c=color)
    x_{vals} = [min(p[0] for p in train_set), max(p[0] for p in train_set)]
   plt.plot(x vals, y vals, '--k', label='Decision Boundary')
   plt.xlabel("Sepal Length")
   plt.ylabel("Sepal Width")
   plt.title("Linear SVM Classifier")
   plt.grid(True)
   plt.legend()
   plt.show()
# MAIN
filename = "/content/Iris.csv" # Replace with your actual file path
dataset = load dataset(filename)
train set, test set = train test split(dataset)
w, b = train svm(train set)
plot svm(train_set, w, b)
# Testing
for point in test set:
   x = point[:2]
   actual = point[2]
   pred = predict(x, w, b)
    print(f"Actual: {actual:<2} Predicted: {pred}")</pre>
```

Output:



