Expt. No. Page. No. def iddles (groph, stant, god):
def des (node, god, depth, path): if node = = goal: if depth == 0: return None. for neighbor in graph get (node, []):
result = dls treighbor, goal,
alepth -1, path + [node]) if result: return result return None · sfor depth in range (len (graph)):
result = dist start, goal, depth, [3)
if result: if result: return result return None graph = & BFS 9 path = iddles (graph 'A', 16') if path:
print (f' Path Jand: 5 path 3')
else:
print ("No path Jourd") 000

graph = s'A': [(B',1), (c',3), (0,7)], B': [(E'S) , (F,12)] 'C': [(G,2)], D: CJ, `F': [('G',3)],
`G': [] } del heuristic (n): H= { 'A': 10 , B': 6, 'L': A, 'D': 7,

E': 5, 'F': 3, G': 0 3 return H(n) print (a-star (graph, 'A', 'G', Leuristic)) O/P: ['A', 'c', 'G'] 1) Memory Bounded A* import heapy del mb-a-star (graph, start, goal, h, memory-limit): open-list=[] Leapq. Leappush (open-list > 10+h (start), 0,

Page. No. import heapay del dijikstra (graph, start, gad):
que = [(0, start)] distance = & vertere: gloat ('infinity')
distance Estart] = 0 came - from = f Start: None 3 cohile quene: current-distance, current-verten= if current distance > distance (current verton 3: continue. bor reighbor, weight in graph (current vertex J. items 1):

distance = current_distance + weight if distance a distances Enrighbour]: distances Enrighbour] = distance Come - from [neighbour]= correct vertex heapq. heappush (quene, reighbour)) path=[]
current = goal

while current is not None: path appoint (current) coverent = came - from [current] path reversel) return path, distances [goal] graph = f'A': { 'B': 1, 'C': 4'9, 'B': f'A'; 1 , 'C': 2 , D': 59, 'C': {'A': 4 , 'B': 2, 'D':13, 3'D': { 'B':5, 'C':13 path, cost = dijilostra (graph, 'A', 'D')
print (f' Path: f path y, cost = scost y') and the back of O(p) + ... , Water transport to the Path: ['A', 18', 'C', 'D'], cost =4 was fath - E man to purit a gratuly colonia tracina e supply habity of the party quadrich acostock r 100 p = 4000 100

from tensory law Kerns models import from tensory law Kerns models import from tensory low Kerns layers import Dense impart mumpy as up

1 = np. random. random (1000, 10)

1 = np. dot(. 11, np. corray (11.5, 1.0, 3.0,

-0.5, 2.5, -1.5, 2.0, -2.5, 1.0, 0.53))+0.1* np. random. rand n (1000) model = Sequential ([
Dense (64, activation = 'relu',
input_Shape (10,1),
Dense (64, activation = 'relu'),
Dense (1)]) model · Compile (optinizer = 'adam', loss = 'mse') model. fit (n, y, epochs = co, both-size=3 print (model predict (x(:53))

To and the sold for the sold fo EC 2.60 - = = the he will and of heart may E 3) 1 Se Agricult Hopes mobile of the section who the west to the same of the same Constitute of the second male complete topical a mideni, south of the mid opposite in a habitant ((ca) e) Liberg- Kelmil Lings

print(& Decision Tree Accuracy: focuracy-tree= 100: -2531. printfl f 'Rardon Forest Acuracy: { accuracy print (1 Decision Tree Classifier . Report") print (classification - report (oy-test, y-pred-tree, larget-name = ivis . tariget - names. 1) print ("Random barest Classifier. Report") -print (classification_ropart (ry-test, ry-pred-bonest, target-name = iris. target-names)) print ('Dec Tree Confusion matrix") print (confusion - mot vix (y-test oy-pred-tree) print (Rand for confusion matrix") print (composion - motrix (y - test , y-pred - for) print l'Docision Tree prediction for 1st 10 hest print ("Predicted:", y-pred-tree [:10]) print (" Actual: ", y-test [:10]) print ("Random Forest practición for 1st 10 test print (" Borticalled: ", y-prod-bort: 103) print (" Actual: ", y- test [: (0])

Random forest by compare decision troo a wondon from from sklearn datasets import lond-ivis from sklearn model-selection import from sklearn model-selection import train-test-split from sklearn tree import Decision Tree Classifier from SKlearn ensemble impart Rondom Farest Classifier from SKlearn metrics import accuracy-scare, classification report , confusion - matrix ivis=load ivis() n= iris · data my : iris · canget split (n, y, test-size = 0.3, random_ State = 42) decision-tree - Doxision Tree Classifier () decision-tree · fit (x-train, y-train) nandon_forest = Random Forest Classifier (n_estimator = 100g rand om_ state = 42) random-farest-bit (n-train, y-train) y-pred-tree = decision-tree · predict (x-test)
y-pred-for= random-forest-predict (x-test)

10) Dosision I Yee Expt. No. Page. No. from sklearn datesets import

from sklearn datesets import

load-ixis

from sklearn tree import Decision Tree Classi free from sklearn import tree m, y = ixis. data , ivis. target df = Decision Tree Classifier () df. fit(2,y) pl. figure (figsize = (20, 101) træ . plot-træ (df. filled = True,

feature - names = ivis,

feature - names, class_names =

ivis target - names) plt. show () (010)

d1= Discrete Distribution (1'7':0.6, d2 = Conditional Probability Table ([[T, 7, 0.8], [T, F, 0.2], [F7T, 0.4], [F, F, 0.6]], [di]) SI= State (di, name = "A") 52 - State (d2 gname = "B") net: Bayesian Network ("simple Network"
net-add States (S1, S2) not. bake () data = np. avray [[[1,1], [1, F] [チャナン・(チャチング) net. fit (data, algorithm 'em') print (net. probability ([['T', 'T'])) OYP 0-69

import numpy as up import matplotlib pyplot as plt from Stelearn datasets import load-ixis from sklearn cluster import K Means ivis = load-ivis/) n= ivis.data of = iris. target Kneans = KHeans (n-clusters = 3, K means tabels = Kmeans · fit-predict (n) plt. figure (figsize = (8,6)) plt. Scatter (x[:,0], n[:,1], (= kmeans-labels, cmap = 'viridis', marker='o', edge color='K') plt. title ('K-means dustering') plt. glabel 1' y')

K-means dustering retails myster Took to many the was in the first of the top of the privatal man-41) did to (1 V 1 Jodal) +

o print ("In Predictions for 1st 10 heart samples:") 24 print ("Prodicted:", y-pred [:10]) 21 print "Actual: ", y-best [: 103) a right of the property and some some first transmisting 12) support vector Alletant township on the Direct Sugar and 東國 東烈 77 4. from sklearn. svm impart svc Suit of a soul of the way with the Colonia 一年一年一年一年十二年十二年十二年 co. model= SUC (Kanol = 'linear') come and a second - money of the best of the second Some as noise Brayes mold () segas verteralient ! I thing ! print (Clearpan report of 1981 ;

a source of the said to

magnific Confusion in the

Come - grad in

import numpy as no import matplotlib. pepplot as plt from sklearn. model-selection import from sklearn. linear-model import Linear Rograssion from sklearn. metrics import mean-squarederror, 12-Score np.random.seed(0) y = 4+3 *x + np. random . random . rando (100,1) 2 train-test-split (2, y , test-size = 0.2, random_stake = 0) model = Linean Regression () model · fit (n-train, y-train) y-pred = model . predict (a-test) mse = mean_squared - error (y-test, y-pred) print (f' R^2 score: (r2)') plt. scatter (n-test, y-test, color='black',

plt. plot (x test, y - pred, color = 1 blue, linewidth = 2, label = 'Predicted line') plt. xlakel ('x') plt. ylabel (V') plt. tible ('Linear Rogression') plt. legerd () to you with a world and ple · show() 10(p: Mean. Squared error = 1, 0434 3381 R^2 Scare: 0.74244523... Linear Regression To Actual data [-Predicted Lata] 0.00 0.05 0.50 0.15 1.00 1.25 . - .

Expt. No. Page No. 2) from siclearn datasets import lord-ivis 3. from sklearn model selection import train-test-spli 4. from sklearn naive bayes import Granssan NB 5. from sklearn metrics import accuracy scare, classification report, confusion matrix ixis = load ixis() x= ixis.data y = iris target 9. x-train, x-test, y-train, y-test = train-test.

Splif (x, y, test-size = 0.3) random-state = 42) 10. model = Craussian NB()

11. model · fit (n. train, y train)

12 y prod = model · predict (n. test) 13. accuracy = accuracy - score (y-test, y-pred)
10. print(f' Accuracy: { accuracy * 100: 26 y/2 15. print ("Classification report")

16. print (Classification-report ly test,

y-pred, target-names:

iris: target-names)) 17. print ("Confusion matrix")

6 print (confusion-matrix (y-test, y-pred.))

graph = f'A': ['B', 1), ('c', 3), ('D', 7)], 'B': [(E',5),('F',12)], C': [['G',2)], , CJ : CJ E': [], 111 - 111 40 111 F': [(6',3)], G: [] 3 def heuristic (n): H= { 'A1:10, 'B':6, 'C':4. 'D':7, 1 E 1:5, F1:3, 161:03 networn HED print (mb-a-stan (graph, 'A', 'G', heuristic , memory - limit = 5)) ['A','(', 16'] all the particular the second of a belief I fellipsent one our in 130 and the grante had a skewn) the rage therp is present Me allowed a colonial in

about the block

t. No. Page. No. g_score = fstart: None]
closed - list = set() while opent-list:

if len (closed-list) > momory-limit:

closed-list pop() closed-list-add (consent) if current = = g ad:

path=[]

while current: part. append (current) current = came - from [current] return path [::-1] for neighbor, cost in graph [current] tentalive -g = current -g + cost if neighbor not in g-score or tentative - g c 9 - score [neighbox came -from Eneighbor J= current g-score [neighbor] = tentative-g f-scare = tentative -g + h (neighbor) heapy heappush topen list, (f-score, tentative-g) neighbor) Metalen. None

del a star (graph, start, goal, h):

open list=[]

heapy heapp wh (open list, (o + h (start),

o, Start)) g-score = & start: None 3 while open-list: current g, current - heap heappop (open-list) if current == goal: while current: path append (current)
awrent = came-from [current] return path [::-i] for reighbor, cost in graph courrents: tentative g = current-g + cost if neighbor not in g-score or tontative g = g-scare [neighbor]: came _from [neighbor] = current g_score [neighbor] = tentative_g

g_score = tentative_g + hcnaghbor)

hospy.hosp push (open_list, l. f-score
tentative_g, neighbor)) E- mahan None