K.M.S.P.Mandal's	Date :- / / 202	
Sant Rawool Maharaj Mahavidyalaya,Kudal	Roll No :- 01	
Department of Computer Science	Expt No :- 01	Signature

Title _____

- BFS - Breadth first Search.
  - It is a vertex-based technique for finding the shortest path in the graph.
  - In BFS, one vertex is selected at a time when it is visited and marked then its adjacent are visited and stored inqueue

- DFS - Depth first Search
  - It is an-edge-based technique. It uses a stack data Structure and performs two stages.
  - First visted vertices are pushed into stack. and second if there are no vertices then visited vertices are popped.

- Example.

(1) we have 8 taluka in Sindhudurg and they are connected with other taluka's. I want shortest distance between source to destination using bfs and dfs.

For this problam statement I am using following matrix.

	Devgad	Malvan	Kankavli	Kudal	Vengurla	Dodamarg	Vai	San.
Devgad	0	1	1	0	0	0	0	0
Malvan	1	0	1	1	1	0	0	0
Kankavli	1	1	0	1	0	0	1	0
Kudal	0	1	1	0	1	0	0	1
Vengurla	0	1	0	1	0	0	0	1
Dodamarg	0	0	0	0	0	0	0	1
Vaibhavadi	0	0	1	0	0	0	0	0
Sawantwadi	0	0	0	1	1	1	0	0

0 - means no route

1 - mean route are available.

I am ~~us~~ implement BFS or DFS ~~der~~ technique on above matrix and try to find out shortest path between source and destination.

⊙ DFS code / Algorithm for above problem statement

- I am using SQLite database named "visited place.db" all visited place are stored into that database.
- The DFS algorithm is implemented using recursive function named
- function take graph start vertex, end vertex, set of visited vertex, current path as parameter
- During the DFS traversal, it adds visited place to database
- The "graph" represented adjacency matrix and their value indicates connection between two place (1 connected, 0 not connected)
- User is prompted to input the start and end indices the vertices they want to find path between
- DFS function called with provided graph, start and end vertices
- If path found, it print the path from start to end. and otherwise print no path found.

Ex. If user want to go Vengurla → Kankavli

① Vengurla —go to→ Malvan → Devgad → Kankavli

* then user have path Devgad to Malvan but malvan are already visited therefore it go to kankavli
* then kankavli → Devgad or Malvan But both are visted also have path kankavli → kudal but we are not choose because direct path kankavli → vaibhavwadi

∴ final path is Ven → Mal → Dev → Kan → Vai.

⊙ BFS Algorith for same problem.

Here I am not using Database.

- A deque is used as a queue for implementing BFS
- bfs function takes parameter as graph start and end vertex.
- It initialize a set to keep track of visited vertices and deque list containing the start vertex as initial path
- function enter while loop continue as long as queue is not empty if he got destination the returns the path
- while loop complete without path finding then return 'None'
- The graph (matrix) is defined, and user prompted to input start and end vertices.
- B function called with these input and if path found it print path otherwise 'No path found' will be printed

Ex. vengurla → vaibhavadi

1) He go ven → Malvan | kudal | Sawantwadi

Here check all path if Malvan have Malvan → Vaibavwadi direct if it is found then it will print same for kudal or Sawantwadi otherwise check Malvan —have path→ Devgad | Kankavli · same method follow. but Here kankavli have direct path to vaibhavwadi

therefore final output is Ven → Malvan → Kankavli → Vaibhav

K.M.S.P.Mandal's
Sant Rawool Maharaj Mahavidyalaya,Kudal
Department of Computer Science

Date :- / / 202
Roll No :-
Expt No :- 02

Signature

Title _____

_____

ey

⊙A* Algorithm.

A* is a popular pathfinding algorithm that intelligently explores path in a graph by combining cost of the actual distance traveled with a heuristic estimate of the remaing distance to the goal. The priority queue ensure that node with lower estimated total distance are explored first. The algorithm terminates when the goal is reached or all possible paths are explored.

(p and q
ing

or

- ⊙ imports 'hepq' module which provide an implementation of the 'hepq' heap queue algorithm. used to efficiently manage the prority queue in A* algorithm.

on

- ⊙ Assign the 'graph' of cities and their distances. It is weighted undirected graph represented as a dictionary of dictionaries. Each city is the key and corresponding value is another dictionary containing neighbouring another cities or distance

he

sen
key.

- ⊙ then define heuristic function it assign estimated distance from each city to goal. it is used to guide A* algorithm.
  - These value are arbitrary and should be choosen based on domain knowledge or experimentation.
  - The purpose of heuristic function is to provide an optimistic estimate of remaining distance to the goal.
- ⊙ Define function named 'astar' with two arguments 'start' and 'goal (starting and destination)
- ⊙ Initialize priority queue ('open_list') with tuple startcity and cost
- ⊙ Create empty set ('closed_list') to keep track of visited cities
- ⊙ Initializing dictionary ('g') to store actual distance from the start to each city.

I

- Initialize dictionary ('f') to store the estimated total distance start to goal city via each city. All values are set to infinity initially, except for the starting city is set to the heuristic estimate
- Initialize ('path') to store the path taken to reach each city.
- pop the node with lowest total estimated distance from 'open_list' current_distance is total estimated distance, and 'current_city' is current
- Iterate over the neighbours of current city and distance
- Calculate actual distance
- print available city in graph
- when users give input start and goal city 'astar' function
- 'shortest_distance' function print result, either the shortest distance and path or message indicating that no path is found.

Ex. Suppose we have graph like

	Mumbai	Pune	Nashik	Sambhajinagar
Mumbai	0	150	170	0
Pune	150	0	0	260
Nashik	170	0	0	180
Sambhajinagar	0	260	180	0

I want go Mumbai ⟶ Sambhajinagar

for that we have two path

1) Mumbai $\xrightarrow{150}$ Pune $\xrightarrow{260}$ Sambhajinagar
2) Mumbai $\xrightarrow{170}$ Nashik $\xrightarrow{180}$ Sambhajinagar

150 + 260 = 310 - This is shortest path.

170 + 180 = 350

Mumbai ⟶ Pune ⟶ Sambhajinagar

will be return with total distance (310)

K.M.S.P.Mandal's Sant Rawool Maharaj Mahavidyalaya,Kudal **Department of Computer Science**	Date :- / / 202 Roll No :- Expt No :- 05	Signature

Title _____

⊙ Decision Tree is supervised learning techniques that can be used for both classification and Regression problems.
- It is a tree structured classifier, where, internal nodes represents features of dataset, branches represent the decision rule and each leaf node represents outcomes.
- There are two Node Decision Node and Leaf Node.
  - Decision node - used to make any decision
  - Leaf node are the output of those decision and do not contain any further branches.

- Steps.

⊙ import all required libraries
- 'pyplot' from 'matplotlib' for plotting and
- 'datasets' and 'decisionTreeClassifier' from 'sklearn' for working with the Iris datasets.

⊙ Load ~~using~~ Iris dataset using 'datasets.load_iris()' data set contain sepal length, sepal width, petal length petal width for 150 iris flowers.
X contain features value Y contains target labels

⊙ Create DecisionTree classifier ('DecisionTreeClassifier) with default hyper-parameters 'random_state' is set to reproducibility fit the model to the training data.

⊙ Create figure for plotting with specified size
use 'tree.plot_tree to visualize trained decision tree.
function take trained classifier ('clf')
features name(iris.feature_names'), target class (iris.target.names) and sets 'filled-True'

K.M.S.P.Mandal's
Sant Rawool Maharaj Mahavidyalaya,Kudal
Department of Computer Science

Date :- / / 202
Roll No :- 01
Expt No :- 04

Signature

Title _____

Support Vector Machine

Support vector machine (SVM) is a supervised machine learning algorithm used for the classification and regression tasks. The primary objective of SVM is to find hyperplane that best separates the data into different classes. In the context of classification, this hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class. The data points nearest that are closed to the hyperplane are known as Support vector.

⊙ Import Libraries:

- Sklearn :- Scikit-learn or sklearn is an open-source machine learning library for an python. It provides simple and efficient tools for data analysis and a wide range of machine learning algorithm.
- 'matplotlib' for plotting, 'datasets' for loading datasets 'svm' for support Vector Machine, 'train_test_split' for splitting data into training and ~~training~~ testing set 'svc' for the SVM classifier and 'accuracy_score' for evaluating model accuracy.

⊙ Load the digit datasets ('digits') using ~~'datasets.load~~ 'dataset.load_digits()'.

- This dataset consist of 8×8 pixel images of handwritten digits (0 through 9)

x - pixel values

y - target labels

Printing x and

⊙ Split the datasets into training and testing sets using 'train_test_split'.
80% data used for training ('X_train', 'y_train')
20% data used for testing ('X_test', 'y_test')

⊙ Creating an SVM classifier ('svc') with a linear kernel. It is worth nothing that you have overwritten the assignment of 'X_train', 'y_train'.

⊙ Train SVM classifier on data using 'fit' method
cls.fit(X_train, y_train)

⊙ use the trained classifier to make prediction on the data ('X_test')

⊙ Display an image from dataset using 'imshow'. In this case, it show 10th image in dataset (digit). This likely done for visualization purpose.

- the SVM classifier is used to classify handwritten digits based on pixel value.

Sant Rawool Maharaj Mahavidyalaya,Kudal

Department of Computer Science

Date :- / / 202

Roll No :- 01

Expt No :- 05

Signature

Title _____

⊙K-NN Algorithm

- K-Nearest Neighbour is one of the most basic yet essential classification algorithms in Machine Learning.
- It belongs to supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection.
- In a nutshell, when you want to predict the class or value of new data point, K-NN looks at the labels or values of its nearest neighbours in training data.
- The term "K" represents the number of neighbors considered and it's crucial factor that you can adjust based on your data.

⊙ Step | Mechanism of program

- Import ~~dataset~~ libraries which is necessary for dataset handling
- we are using dataset 'digits', which contain images and handwritten digits and their corresponding labels.
- load digits dataset
- train_test_split function used for split dataset into training and testing sets.
- 80% for training and 20% for testing
- K-NN classifier is created ('knn') using KNeighborsClassifiers with k=9

n_neighbors=3 parameters to consider when making prediction. In this case, the value is set to 9, meaning that when algorithm needs to make a prediction for new data points, it looks at the labels of the three nearest neighbors in the training set and assign the most common lable among them to the new data point.

- Train the K-NN classifier using training set. ~~used~~ we ~~us~~ ('fit') function
- use the trained classifier ~~using the trainingset~~ to make prediction on testing set. predicted value printed.
- accuracy_score function evaluate accuracy using scikit-lea comparing predicted values ('y_pred') with actual values ('y

K.M.S.P.Mandal's
Sant Rawool Maharaj Mahavidyalaya, Kudal
Department of Computer Science

Date :
Roll No :
Expt No :
Signature

Title

- Naive Bayes algorithm is supervised learning algorithm, which is based on Bayes theorem and used for solving classification
- It is mainly used in text classification that include a high dimensional training dataset.
- It helps in building the fast machine learning models that can make quick predictions.

⊙ Advantage.
- Most popular choice for text classification.
- It perform well in Multi-class classification. prediction as compare to other algorithms

⊙ Disadvantage
- Navie Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features

⊙ Steps.
- import require packeges or dataset. we are using iris dataset
- These dataset consists measurment of 150 iris flowers from three different species.

- used 'train_test_split' function to split dataset into training and testing sets.
- 60% of data is used for training (X_train, y_train)
  40% of data is used for testing (X_test, y_test)
- import Gausssian Navie Bayes classifier from scikit-learn and create an instance ('gnb')
  GaussianNB() function / class choosen for classification tasks when assumption of normally distributed features holds and its used to create an instance of Gaussian Navie Baye classifier ('gnb')

- then used 'fit' method to train the model on the trainin
  ('x_train) and ('y_train)
- trained model ('gnb') is used to predict target values f
  testing set ('x_test')
- The accuracy_score function from 'metrics' module is
  to compare the actual target values (y_test) with predi
  values (y_pred)
- the accuracy will be print as percentage.