EXP NO: 01
DATE : 27/04/2023

FIND-S Algorithm

Aim :

To Implement and demonstrate the Find-S Algorithm for finding the most specific hypothesis based on a given set of training data samples.

Algorithm:

1. Import the dataset using csv.reader()
2. Find the total no. of attributes present in the dataset and initialize a variable with the same value.
3. Initialize an empty list to store the final hypothesis.
4. Implement the main algorithm -

   If an instance is positive -

   Update the value of the attribute in the hypothesis.

   If hypothesis value == "ϕ" or current instance's value

   hypothesis value ← current instance's value

   else

   hypothesis value ← '?'

   If an instance is negative - Ignore the instance
5. Display the hypothesis thus formed
6. END

EXP NO: 02
DATE: 28/04/2023

CANDIDATE ELIMINATION ALGORITHM

Aim:

For a given set of training data examples stored in a .csv file, implement and demonstrate the candidate-elimination algorithm in python.

Algorithm:

1. Import the dataset using .csv reader()
2. Find the total no. of attributes present in the dataset. Store this value in a variable (say, num_attributes)
3. Declare two lists to store specific & general hypothesis.
4. Initialize the specific hypothesis to '0's equal to the no. of attributes.
5. Similarly, initialize the general hypothesis to '?'s equal to the no. of attributes.
6. Implement the main algorithm.
   - Iterate through all instances
   - If an instance is positive:
     - generalize the specific hypothesis
   - If an instance is negative:
     - make the general hypothesis more specific
7. Display both general and specific hypothesis.
8. END

EXP NO: 03
Date: 29/04/2023

ID3 ALGORITHM

Aim:

To write a program to demonstrate the working of the decision tree based ID3 algorithm. To use an appropriate dataset for building the decision tree and apply this knowledge to classify a new example.

Algorithm:

1. Import all required libraries (pandas, math, numpy).
2. Read data using read_csv() function.
3. Remove the target from the data & store the attributes in a seperate variable.
4. Create a class named node with 4 members - 'Children', 'value', 'isleaf' and 'pred'.
5. Define a function called entropy to find the entropy of the dataset.
6. Define a function named printTree to draw the decision tree
7. Finally, call the ID3 printTree function to obtain the decision tree
8. END

EXP NO : 04
DATE : 02/05/2023

BACKPROPAGATION ALGORITHM

Aim:

To build an artificial neural network by implementing the backpropagation algorithm and test the same using appropriate datasets.

Algorithm:

1. Initialize the no. of input neurons, hidden-neurons and output neurons.
2. Initialize the weights and biases associated with an artificial neuron randomly using np.random.uniform()
3. Calculate the net input of every neuron

   Net input = sum of the product of each weight value & corresponding input value + bias.
4. Calculate the net output of every hidden neuron using sigmoid function

   $$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$
5. Calculate errors -

   $$Error_x = \frac{1}{2}(\text{target o/p}_x - \text{generated o/p}_x)^2$$

   and calculate total error

   $$Error(total) = \epsilon_1 + \epsilon_2 + \dots \epsilon_n$$
6. If Δ error is high, traverse back the network and update the weight values.
7. Repeat steps 1-6 after updating the weights till the error difference is minimum.

EXP NO: 10
DATE : 09/05

K-NN ALGORITHM

Aim:

To write a program to implement the k-NN (k-nearest neighbour) algorithm in Python.

Algorithm:

1. Import dataset using csv.reader() or create your own list of instances.
2. Get the query instance from the user or initialize it as a list.
3. Define the value of 'k' (or) get it from the user.
4. Based on the query instance, find the distance of each instance from the query point using -

$$d_x = \sqrt{(q_1 - x_1)^2 + (q_2 - x_2)^2 + \ldots + (q_n - x_n)^2}$$

(n = total no. of attributes)

5. Find the nearest distance by sorting the distance array.
6. Check the consequence of first k nearest distances.
7. Count the no. of positive and negative consequences
8. If +ve consequences > -ve consequences:

   The given query instance corresponds to a positive consequence.

   Else:

   The given query instance corresponds to a negative consequnes
9. END

EXP NO : 05
DATE : 03/05/2023

NAIVE BAYES ALGORITHM

Aim :

To write a program to implement Naive Bayes algorithm in python and to display the result.

Algorithm:

1. Import the dataset using CSV.reader() and append it to the list
2. Calculate the no. of attributes in the dataset and store it in a variable.
3. Recieve the query instance from the user or initialize it within the program as a list.
4. Count the total no. of positive and negative instances & store it in seperate variables.
5. Calculate the overall possibilities (both +ve & -ve) of the dataset.
6. Create two seperate variables to store the final Naive Bayes probability. Initialize them with the overall probability

   NB_pos ← overall_pos
   NB_neg ← overall_neg
7. For every attribute value of the given query instance, calculate the probability using Bayes theorem

   NB_pos ← NB_pos * value
   NB_neg ← NB_neg * value
8. Display the result

EXP NO: 07

DATE: 05/05/2023

LINEAR REGRESSION

Aim: To write a python program to implement linear regression.

Algorithm:

1. Load the desired dataset using csv.reader()
2. Identify the dependent and independent variable and store them seperately as lists.
3. Visualize the dependencies using scatterplot from the mathplotlib library.
4. Split the data into training and testing data.
5. Train the algorithm by importing 'Linear Regression' from 'sklearn.linearmodel'.
6. Visualize the training set using scatterplot from matplot lib library.
7. END.

EXP NO: 08

DATE: 06/05/2023

POLYNOMIAL REGRESSION

Aim: To write a program to implement polynomial regression in python.

Algorithm:

1. Import the desired dataset using CSV.Reader() and store it as a list.
2. Visualize the dataset and identify the dependent and independent variables.
3. Seggregate the independent and dependent variable as X and Y.
4. Divide the dataset into training and testing data.
5. Import 'polynomial features' from sklearn preprocessing
6. Increment or decrement the degree to get a more accurate result.