Expt. No. 1

Page No. 1

Expt. Name Techniques for Data Processing

Date : 30-11-2021

mean Removal Scaling Normalization.

Aim:- To perform techniques for data Pre-Processing mean Removal, Scaling normalization.

a) mean Removal:

Algorithm:

1. Read the input csv file and extract the column of numbers:
   a. open the input csv file in read mode using the open() function.
   b. create a csv reader object using the csv.reader() function.
   c. iterate over the csv file using a for loop extracting each row.
   d. convert each row's first element to a float value and appened it to data list.
2. Calculate the mean:
   a. import the statistics module to access the statistics functions.
   b. calculate the mean of the data list using the statistics.mean() function.
3. Remove the mean from each data point:
   a. create a newlist called mean removed data.
   b. iterate over the data list using for loop.
   c. subtract the mean value from each data point and append result to the mean removed data list.
4. write the mean removed data to a new csv file.
   a. Open the output csv file in write mode with newline = '' to prevent extra blank lines using the open() function.
   b. create a csv writer object using the csv.writer() function.
   c. Iterate over the mean removed data list using a for loop.
   d. write each value as a row in the csv file using the writer.writerow() method.
5. close the input and output csv file.

b. Scaling:

Expt. No. 1 Page No. 2

Expt. Name _____ Date : 30-11-2023

Algorithm:

1. Read the input csv file.
   a. Open the csv file input-file in read mode using the open() function.
   b. Create a csv file reader object using the csv.reader() function.
   c. Extract the data from the csv file by iterating over the rows and converting each row's first element to a float using a list comprehension.
2. Find the minimum and maximum values in the data:
   a. Determine the minimum value in the extracted data using the min() function.
   b. Determine the maximum value in the extracted data using the max() function.
3. Perform min-max scaling on each data point:
   a. Create an empty list to store the scaled data (scaled-data)
   b. Iterate over the original data points (data)
   c. For each data point x calculate the scaled value using the formula.
      Python scaled value = (x - current min) / (current - max - current min) * (new - max - new - min) + new min
   d. Append the calculated scaled value to the scaled data list
4. Write the scaled data to a new csv file.
   a. Open the output csv file output-file in write mode using the open() function
   b. Create a csv writer object using the csv writer() function.
   c. Iterate over the scaled data (scaled-data)
   d. For each scaled data point value write a new row containing the value to the csv file using the writer.write row() method.
5. Close the input and output csv files.

Expt. No. 1

Page No. 3

Expt. Name

Date : 30-11-2023

C. Normalization:

1. Read the csv file and extract the column of numbers
   Open the input csv file in read mode using open
   create a csv reader object using csv.reader(file)
   Extract the first column of numbers and convert each value to float
   using float(row[0]) for row in reader.
2. find the minimum and maximum values in the data.
   Determine the minimum value in the extracted data using max(data)
   Perform min-max scaling on each data point.
3. Apply the min-max scaling formula to each data point.
   Replace x with the current data point current - min with the minimum value
   current - max with the maximum value new-min with the desired
   minimum value and new-max with the desire maximum value.
4. write the normalized data to a new csv file.
   Open the output csv file in write mode with newline character
   handling using open(output file 'w' newline='')
   create a csv file writer object using csv.writeline
   iterate through the normalized data write each value as a one-
   element list using open(output file 'w' newline).
   Create a csv file call the min-max - scaling function.
   check if the code is executing as the main program using if.
   name == "main"
   Define the input and output file path in variables input file and
   output file. respectively.
   Call the min-max - scaling function with the input and output file
   Path as arguments.
   The provided code effectively normalizes the data in the
   input csv file using min-max scaling and saves the normalized

Expt. No. 9(a) Page No. 5

Expt. Name Naive Bayes classifer Date : 1-02-24

Aim: To perform Naive Bayes classification

Algorithm:-

Step-1: Import necessary libraries

Step-2: Create a data frame

Step-3: Encode categorical data.

Step-4: Separate features and target

Step-5: Split data into training and testing data.

Step-6: Create and train a Gaussian Naive Bayes model

Step-7: Make Predictions on the testing set

Step-8: Evaluate model Accuracy.

Code:

```
import pandas as pd
f = pd.data frame ({'weather': ['Sunny', 'Rainy', 'Sunny', 'wind': ['
                    'mild', 'High', 'mild',],
                    'Temp': ['moderate', 'mild', 'moderate', 'm
                    'go': ['Yes', 'No', 'Yes', 'Yes']})
print(f.columns)
from sklearn.naive_bayes import GaussianNB as g
from sklearn.preprocessing import label encode as le
from sklearn.model_selection import train_test_split as tt
l = le()
for i in f columns
f[i] = l fit transform (f[i])
```

Expt. No. 2b Page No. 7

Expt. Name Support vector machine Date : 1-02-24

Aim: To demonstrate Support vector machine

Algorithm:

Step-1: Import necessary libraries

Step-2: load the dataset

Step-3: extract the features and target variable

Step-4: Split the data and into training and testing sets

Step-5: scale features

Step-6: create and train a Svm model

Step-7: make predictions on the testing set

Step-8: Evaluate model performance

Step-9: visualize the decision boundary

Programme:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv(r"c:/users/online/desktop/social_network_Ads
                      csv")
x = dataset.iloc[:,[2,3]].values
y = dataset.iloc[:,4].values
from sklearn.model_selection import train_test_split
x_train x_test y_train y_test = train_test_split
                      (x,y,test_size=0.25, random_state=0)
from sklearn.preprocessing import standardScaler
sc = StandardScaler()
x_train = sc.fit_transform (x_train)
x_test = sc.transform (x_test)
```

Expt. No. 2c Page No. 10

Expt. Name Logistic Regression Date : 1-8-24

Aim: To Perform logistic Regression

Algorithm:

Step-1: Import necessary libraries

Step-2: Load the dataset

Step-3: Extract features and target variables

Step-4: Split data into training and testing sets

Step-5: Scale features

Step-6: Create and train a logistic regression model

Step-7: make predictions on the testing set

Step-8: Evaluate model Performance

Step-9: Visualize the decision boundary.

Programme:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from matplotlib.colors import ListedColormap
dataset = pd.read_csv("C:/users/online/Downloads/diabetes.csv")
    x = dataset.iloc[:,[4,7]].values
    y = dataset.iloc[:,8].values
xtrain xtest ytrain ytest = train_test_split
                    (x,y,test_size=0.25, random_state=0)
    Sc_x = StandardScaler()
```

Expt. No. 2d Page No. 13

Expt. Name Decision Tree Date : 1-02-24

Aim: To perform the decision tree Algorithm

Algorithm:

Step-1: Import necessary libraries

Step-2: load the dataset

Step-3: define a Node class

Step-4: calculate entropy

Step-5: calculate information gain

Step-6: implement the ID3 Algorithm

Step-7: Print the Decision tree

Step-8: Classify a new example

Step-9: Train and use the model

Programme:

```
import pandas as pd
import math
import numpy as np
data = pd.read_csv(r"c:\users\online\downloads\weather.csv")
feature = [feat for feat in data]
feature.remove("Answers")
class Node:
    def __init__(self):
        self.children = []
        self.value = ""
        self.isleaf = False
        self.pred = ""
    def entropy(examples):
        pos = 0.0
```

Expt. No. 2c Page No. 17

Expt. Name Random forest Date : 1-02-24

Aim: To perform the Random forest Algorithm

Algorithm:

Step-1: Import necessary libraries

Step-2: load the dataset

Step-3: extract features and target variables

Step-4: Split data into training and testing sets

Step-5: Create and train a Random forest model

Step-6: make Predictions on the testing set

Step-7: Evaluate model performance

Step-8: Visualize the confusion matrix.

Programme:

```
import Pandas as Pd
data = Pd.read_csv("Heart Disease1.csv")
x = data.iloc[:,[1,2,3,4,5,6,7,8,9,10,11,12]] values
y = data.iloc[:,13] value
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.
                                                    random_state=1)
from sklearn.ensemble import Random forest Classifier
rc = Random forest Classifier()
rc.fit(x_train, y_train)
y_pred = RFC.Predict(x_test)
from sklearn import metrics
print("Classification Accuracy", metrics.accuracy_score(y_test, y_pred)*100)
```

Expt. No. 3 Page No. 19

Expt. Name K-means clustering Date : 1-02-24

Aim: To Perform the k-means Algorithm

Algorithm:

Step-1: Import the necessary libraries

Step-2: Load the dataset

Step-3: extract features and target variables

Step-4: Split the dataset into it explanations

Step-5: Create and train k-means model

Step-6: Evaluate the model performance

Step-7: Visualize the k-means boundary

Programme:-

```
import pandas as pd
import numpy as ny
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn cluster import kmeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import minmax scaler
iris = pd.read_csv("iris.csv")
x = iris.iloc[:,[1,2,3,4]].values
from sklearn.cluster import kmeans
wcss = []
for i in range(1,11)
k means = k means (n_clusters = i, init = 'k-means++')
                  max_iter = 300, n_init = 10, random_state = 0)
k-means.fit(x)
```

Expt. No. 4 Page No. 21

Expt. Name NLTK Date : 22-02-24

Aim: To Perform the NLTK Programme.

Algorithm:

Step-1: Import NLTK and Download Resources

Step-2: Import the necessary modules from NLTK such as word tokens PorterStemer, word Net lemmatizer and stop words.

Step-3: Download required NLTK Resources using nltk.download()

Define Text:

Input the text you want to Process

Tokenization:

* use sent_tokenize to split the text into Sentence
* for each sentence use word_tokenize to split into words.

Stemming:

* create an instance of PorterStemmer
* for each word token Apply Stemming using the Stem method of the Stemmer

lemmatization:

* create an instance of word Net lemmatizer

Programme:-

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import stopwords
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
text = "The quick brown foxes are jumping over the lazy
```

Expt. No. 5 Page No. 29

Expt. Name Building Bag of words model using NLTK Date : 22-02-24

Aim:- To implement bag of words model using NLTK.

Algorithm:

1. Import libraries

import necessary NLTK modules for tokenization, stopwords, and frequency distribution

2. Preprocessing function:

Tokenizes convert to lowercase and remove stopwords from the input text

3. Bow model function (create_bow_model)

utilizes preprocess_text for each text in the corpus

Counts word frequencies using freqdist and builds a Bag of words model.

4. Example usage:

An example list of texts is provided in the text variable.

The create_bow_model function is called and the resulting model is printed.

Code:-

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
nltk.download('punkt')
nltk.download('stopwords')
def preprocess_text(text):
    stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(text)
    filtered_words = [word.lower() for word in word_tokens if word
```

Expt. No. 6 Page No. 25

Expt. Name Identifying patterns in text data Date : 29-08-24

Aim:- To write a Python code for Identifying patterns in text data.

Algorithm:-

Step-1:- Import libraries

Step-2:- Define function

Step-3:- Intalize Pattern dictionary

Step-4:- Open csv file

Step-5:- Read csv file

Step-6:- Interate over rows

Step-7:- Extract text

Step-8:- find patter matches

Step-9:- Update patterns dictionary

Step-10:- Return Patterns.

Code:-

```
import csv
import re
def identify Patterns (csv_file_Path, Column_name)
        Patterns = { }
with open (csv_file_Path 'r') as csv file :
    reader = csv.DictReader file (csv file)
    for row in reader
    text = row [column_name]
Pattern_matches = refind all (r'female' text flags = re.IGNO
        for match in Pattern_matches:
            if match in Pattern
            Patterns [match] += 1
    else
```

Expt. No. 7 Page No. 27

Expt. Name Analysing Sequential Data by HMM Date : 24-02-24

Aim:- To analyze Sequential data by Hidden markov model

Algorithm:

Step-1:- Initialization

~~Step-01~~ choose the number of hidden States

Initalize the transition probabilities (trans-matrix) Emission probabilities (emission matrix) and initial state probabilities

Step-2:- Training

* Provide a sequence of observed data
* Update the model parameters using training Algorithms

step-3 - Prediction.

* Given a new sequence of observed data use the viterbi algorithm to find the most likely sequence of hidden states.

step-4:- Results

* output the predicted sequence of hidden states based on the observed data

Code:

```
import numpy as np
from hmmlearn import hmm
n_states = 2
trans_matrix = np.array([[0.7, 0.3], [0.4, 0.6]])
Emission_matrix = np.array([[0.1, 0.4, 0.5], [0.6, 0.3, 0.1]])
initial_probs = np.array([0.6, 0.4])
model = hmm.MultinomialHMM(n_components = n_states, startprob
_prior = initial_probs, transmat_prior = trans_matrix, n_iter=100)
train_data = np.array([[0, 1, 2, 0, 1, 2, 0, 2, 1]])
train_data = train_data.reshape(-1, 1)
model.fit(train_data)
```

Expt. No. 8

Page No. 29

Expt. Name Heuristic Search Technique.

Date : 07-03-24

Concept of Heuristic Search technique:

A Heuristic is a technique to solve a problem faster than classic methods or to find an Approximate solution when classic method cannot this is a kind of a shortcut as we often trade one of optimality completeness, accuracy or precision for speed.

Heuristic Search techniques in AI

Other names for these are informed search Heuristic other names for these are effective if applied correctly to the right types of tasks and usually demand domain-specific information.

Before move on to describe certain techniques let's first take a look at ones we generally observe Below we name a few.

* Best first Search.
* Bi directional search
* Tabu Search
* Simulated Annealing
* Hill climbing
* Constraint satisfication problems (CSP)

let's talk of magic square this is a square of numbers usually integers - arranges in a square grid. the members in each row each column and each diagonal all add up to constant which we call the magic constant let's implement this with Python.

A* Search Algorithm and it's Basic Concepts

A* algorithm works based on heuristic nodes and this helps achieve optimality. A* is a different from of the best - first algorithm when A* enters into a problem firstly it calculates the cost to travel into the neighbour nodes and chooses the node with the lowest cost (f the fn) denotes the cost A* chooses the node with the lowest f(n) value. Herein denotes the neighbouring nodes the calculation of the value can be

Expt. No. 8
Expt. Name
Page No. 80
Date : 07-03-24

shown below

$f(n) = g(n) + h(n)$ $f(n) = g(n) + h(n)$

$g(n)$ = Shows the shortest path's value from the starting node to n

$h(n)$ = the heuristic approximation of the value of the node

The heuristic value has an important role in the efficiency of the A* algorithm to find the best solution you might have to use different heuristic functions according to the type of the problem

However the creation of these function is a diffcult task and this is the basic problem we face in AI

what is a heuristic function:

Essentially a heuristic function helps algorithm to make the best faster and more efficiently the ranking is based on the best available information and helps the algorithm decide the best possible branch to follow Admissibility and consistency are the two fundamental properties of a heuristic function.

Admissibility:

A heuristic function is admissible if it can effectively estimate the real distance between a node 'n' and the end node. it never overestimates if it ever does it will be denoted by 'd' which also denotes the accuracy of the solution

Consistency:

A heuristic function is called consistent if the estimate of a given heuristic function turns out to be equal to or less than the distance between the goal(n) and a neighbour and the cost calculated to reach that neighbour

A* is indeed a very powerful algorithm used to increase the performance of artificial intelligence it is one of the most popular search algorithm in AI the sky is the limit when comes the potential of this algorithm However the efficiency of an A* algorithm highly depends on the quality of its

Expt. No. 8
Page No. 31
Expt. Name
Date : 07-03-24

heuristic function.

Implementation with Python:

In this section we are going to find out how the A* search algorithm can be used to find the most cost-effective path in a graph. Consider the following graph. The numbers written on edges represent the distance between the nodes while the numbers written on nodes represent to heuristic values. Let us find the most cost effective path to reach from start state to final state using the A* algorithm.

Let's start with node A. Since A is a starting node therefore the values g(x) for A is zero and from the graph we get the heuristic of A is 11 therefore

$$g(x) + h(x) = f(x)$$

$$0 + 11 = 11$$

Thus for A, we can write

A = 11

Now from A, we can go to Point B or Point E. So we compute for each of them A -> B = 2+6 = 8

A -> E = 3+6 = 9

Since the cost for A -> B is less, we move for forward the this path and compute the f(x) for the children nodes of B

A -> B -> C = (2+1) + 99 = 102

A -> B -> G = (2+9) + 0 = 11

Here the pattern A -> B -> G has the least cost but it is still more then cost of A -> E we explore this path further

A -> E -> D = (3+6) + 1 = 10

Comparing the cost of A -> E -> D with all paths we got so far and as this cost is least of all we move forward with this path and compute f(x) for the children of D

Expt. No. 8

Expt. Name

Page No. 32

Date : 07-03-24

A -> E -> D -> G = (3+6+1)+0 = 10

Now Comparing all the Paths that lead us to the goal, we conclude that A -> E -> D -> G is the most cost-effective Path to get from A to G.

Programme:

```
def aStarAlgo(Start node, Stop node)
    open_set = set(Start node)
    closedSet = set()
    g = {}
    Parents = {}
    g[Start node] = 0
    while len(open_set) > 0:
        n = None
        for v in open_set:
            if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
                n = v
        if n == Stop node or Graph nodes[n] == None:
            pass
    else:
        for (m, weight) in get_neighbors(n)
            if g[m] > g[n] + weight
if n is in closed_set
    closed_set.remove(n)
    open_set.add(n)
```

H-dist = {
'A' : 11,
'B' : 6,
'C' : 99,
'D' : 1,
'E' : 7,
'G' : 0,

o/p ✓

Result: Thus the Heuristic techniques has been successfully implemented the graph.

Expt. No. 9th Page No. 33

Expt. Name Tic-Tac-Toe Date : 7-03-24

Aim: To play a bot to play Tic Tac Toe

Algorithm:

1. Initialization

Create a 3×3 board represented as a list of lists where each inner list represents a row initialize all elements in the board to empty spaces (" ")

Define two players with their symbols (eg 'x' and 'o')

Choose a starting player (eg player with "x")

2. Game loop:

* Display the board: Call the print board function to show the current state of the game.

Get player move: o Prompt the current player to enter the desire row and column coordinates (0, 1 or 2)

o validate the input: Check if the entered row and column are within the board's boundaries (0-2)

Ensure the choosen position is empty (" ")

o if the input is invalid prompt the player to enter again until a valid move is made.

Make the move: Place the current player's symbol ("x" or "o") at the chosen position on the board

Checks for winner: Call the check winner function with the current board and player symbol o if the function returns true the current player has won

Display the board and announce the winning player

Exit the game loop

Check for tie: Call the is board full function with the current board

o if the function returns True all position are filled and if so the display the board and announce a tie

Expt. No. 9 Page No. 34

Expt. Name Date : 07-03-24

Exit the game loop

Switch Player: Change the current Player to the other Player

3. Repeat:

Continue iterating through the game loop until win or tie is declared

Programme:

```
def Print_board(board):
    for row in board
    print ("|" join (row))
    Print ("-" *5)
def check_winner (board, Player):
    for i range (3):
    if all (board [i] [j] == Player for j in range (3) or all (board [j] [i]
    Player for i in range (3):
    return true
if all (board [i] [i] == Player for i in range (3)) or all (board [i] [2-i] ==
    Player for i range (3)):
        return true
        return false
def is board full (board)
    return all (board [i] [j] != " " for i in range (3) for j in range (3):
    def tic tac toe ():
board = [[" " for in range (3)] for i in range (3)]
    Player = ["x" "o"]
    current_Player = Player [0]
    while true:
        Print_board (board)
        while (true):
```

Expt. No. 10

Expt. Name Single layer perceptron

Page No. 36

Date : 14-03-24

Aim:- To Perform the single layer perceptron

Algorithm:-

Step-1: import necessary libraries

Step-2:- Now load the dataset using keras import version of tensorflow

Step-3: Now display the shape and image of single image in dataset.

Step-4:- Now normalize the dataset in order to compute calculations in fast and accurate manner.

Step-5: Building a neural network with single layer perception.

Step-6:- validate and display the result.

Code:-

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
plt.matshow(x_train[0])
x_train = x_train/255
x_test = x_test/255
x_train_flatten = x_train.reshape(len(x_train), 28*28)
x_test_flatten = x_test.reshape(len(x_test), 28*28)
model = keras.Sequential([keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')])
model.fit(x_train_flatten, y_train, epochs=5)
model.evaluate(x_test_flatten, y_test)
```

Expt. No. 11

Expt. Name Building linear Regression using ANN

Page No. 40

Date : 14-04-24

Aim: To perform Building linear Regression using ANN

Algorithm:

Data Preparation: Prepare your dataset with input features (x) and target variables (y)

Split the dataset into training and testing

model architecture:

Create a Sequential model.

Add a Dense layer with one neuron as the output layer

use a linear Activation function for the output layer

Compile the model.

Compile the model using Appropriate optimize and loss function

model training:

Train the model using the training data use fit

Model evaluation:

Evulate the model use the testing data

Use the evaluate method to calculate the model performance metrics

model prediction.

use the trained model to make predictions on new data

use the predict method to get prediction.

Expt. No. 12

Expt. Name Image classifier on application of DL

Page No. 42

Date : 14-03-24

Aim:- To Perform the image classification using Deep learning

Algorithm:-

Step-1:- Import required libraries using tensorflow and keras for building and training CNN model

Step-2:- load the MNIST dataset tensorflow keras provides easy Access to the MNIST dataset.

Step-3:- Preprocess Data:- The involve normalize the pixel value and reshaping the data to fit the CNN input requirements

Step-4:- Define the CNN model:- A basic CNN model involve convolution layers

Step-5:- Complete and train the model. Use the Adam optimizer and sparse categorical cross entropy as the loss function

Step-6:- Evaluate the model Asses the model performance of the test set

Code:-

```
import tensorflow as tf
from tensorflow keras models import Sequential
from tensorflow keras layer import Conv2D, Dense,
                    Flatten, max Pooling 2D
minist = tf.keras.datasets.mnist
(x-train, y-train), (x-test, y-test) = minist.load_data()
x-train, x-test = x-train/255.0, x-test/255.0
model = Sequential
Conv 2D(32, kernel = (3,3), activation = 'relu',
                    input Shape = (28,28))
```