**SOURCE CODE AND OUTPUT**

**7.1 CODING**

**Similarity calculator (psycho graphics based):**

***Ca.py:***

# Import Required Modules and library

import pandas as pd

import numpy as np

from scipy.spatial import distance

from pymongo import MongoClient

import os

# Read Dataset CSV File from the disk

df = pd.read\_csv('dataset.csv')

#Arrange a CSV File in a order

rows = rows = df['name'].values.tolist()

df['min'] = (df['o1'] + df['c1'] + df['e1'] + df['a1'] + df['n1']) /5

df['max'] = (df['o2'] + df['c2'] + df['e2'] + df['a2'] + df['n2']) /5

df.set\_index('name', inplace=True)

# Crate a new Empty datafram

newdf = pd.DataFrame(index=rows,columns=rows)

# Loop each record(min max) in dataframe and find eucledean distance of them

for index,rec in df.iterrows():

now = [rec['min'],rec['max']]

for sindex,srec in df.iterrows():

sub = [srec['min'],srec['max']]

new\_value = distance.euclidean(now,sub)

newdf.at[index,sindex] = new\_value

# Store the new Dataframe in mongodb

client = MongoClient(os.getenv("DB\_URI"))

db = client['ca\_db']

collection = db['ca\_scores']

# Convert a dataframe into JSON Format

data = newdf.to\_dict()

row = newdf.index.values.tolist()

# Iterate Each row and convert into Object

for key in row:

newobj = {

'name':key,

'similar':[data[key]]

}

# Insert the objects into MongoDB

rec\_id = collection.insert\_one(newobj).inserted\_id

# Print the ID of each Inserted Record

print(rec\_id)

print("Completed")

**(Collaborative filtering based)**

import pandas as pd

from scipy import sparse

from sklearn.metrics.pairwise import cosine\_similarity

# read the dataset

ratings = pd.read\_csv("cl-dataset.csv",index\_col=0)

ratings = ratings.fillna(0)

# standardize the data

def standardize(row):

new\_row = (row - row.mean())/(row.max()-row.min())

return new\_row

ratings\_std = ratings.apply(standardize)

# Find the cosine similarity

item\_similarity = cosine\_similarity(ratings\_std.T)

print(item\_similarity)

# Crete a DataFrame

item\_similarity\_df = pd.DataFrame(item\_similarity,index=ratings.columns,columns=ratings.columns)

# Get Similar Product

def get\_similar(product\_name,user\_rating):

similar\_score = item\_similarity\_df[product\_name]\*(user\_rating-2.5)

similar\_score = similar\_score.sort\_values(ascending=False)

return similar\_score

print(get\_similar("Product6",1))

# User based recommendation

testuser = [("Product1",5),("Product5",1),("Product6",1)]

similar\_product = pd.DataFrame()

for product,rating in testuser:

similar\_product = similar\_product.append(get\_similar(product,rating),ignore\_index=True)

similar\_product.head()

similar\_product.sum().sort\_values(ascending=False)

**Recommendation System Api**

***App.js***

// Load the required Modules

const express = require('express')

const mongoose = require('mongoose')

const dotenv = require('dotenv').config()

// Initialize the Express App Instance

const app = express()

// Assigns a PORT to Web Application

const port = process.env.PORT || 5000

// Get the DataBase URI From the System Environment Variables

let url = process.env.DBURI;

// Establish a Connection to MongoDB

mongoose.connect(url,

    {

        dbName:'ca\_db',

        useNewUrlParser: true,

        useUnifiedTopology: true

    }

);

mongoose.set('useCreateIndex', true);

const db = mongoose.connection;

db.on('error', ()=>console.log("DB Connection Error"));

db.once('open',()=>console.log('Connction DB Done'));

// initialize a Schema for MongoDB Collections

const caschema = mongoose.Schema({

    name:{

        type:String,

        required:true,

    },

    similar:{

        type:Array,

        required:true,

    }

});

// Bind the schema to MongoDB Collection

model = mongoose.model('ca\_scores', caschema);

// Create Index Route, that send a response about the instance

app.get('/',(req,res)=>{

    res.json({

        Name: "Cognitive-Analytica",

        InstanceId : "7E3AL83Z",

        status : "Active"

    })

})

// Get Route for all product simlilarity

app.get("/all",(req,res)=>{

    // select the all record

    model.find({},(err,result)=>{

        if(err){

            // send a errorr esponse

            res.json({

                status:"fail",

                data:err

            })

        }

        else{

            // Get a success response

            res.json({

                status:"success",

                data:result

            })

        }

    })

})

// Initiallize a End point that taking a product name on the query

app.get('/item/:productname',(req,res)=>{

    // fetch the Product name parameter from request object

    let query = req.params.productname

    // Query the Database and get the result

    model.find({"name":query},(err,result)=>{

        // If the Query returns error

        if(err){

            res.json(err)

        }

        // If the Query returns a successfull result

        else{

            // Extract the Required Data from the Query result

            raw\_data = result[0]['similar'][0]

            // Initialize the Empty Array

            let sortable = [];

            // Loop every object in the fetched data

            for (let value of Object.keys(raw\_data)) {

                 // Push the Every Object into the Array

               sortable.push([value, raw\_data[value]]);

                }

            // Sort the Every Elements in the Array

            let similar = sortable.sort(function(a, b) {

                    // Return the Array in ascending

                 return a[1] - b[1];

            });

            // Send a resonse in the JSON Format

            res.json({similar})

        }

    })

})

// Listen a Application on a specified Port

app.listen(port,()=>{

    // Print the debug line

    console.log(`Server Listening in ${port} `)

})

**Sample Data Set:**

name,o1,o2,c1,c2,e1,e2,a1,a2,n1,n2

product0,4,7,3,7,2,7,2,5,3,8

product1,3,5,2,2,1,7,1,2,3,9

product2,3,5,2,2,1,6,4,8,4,6

product3,4,5,1,3,4,6,2,3,3,3

product4,4,5,2,7,1,5,2,3,3,5

**Dependencies And Run Scripts:**

{

"name": "cognitive-analytica-recommendation-system",

"version": "1.0.0",

"description": "",

"main": "app.js",

"scripts": {

"start": "node app.js",

"dev": "nodemon app.js"

},

"author": "Naveen Micheal",

"license": "ISC",

"dependencies": {

"dotenv": "^8.2.0",

"express": "^4.17.1",

"mongodb": "^3.5.5",

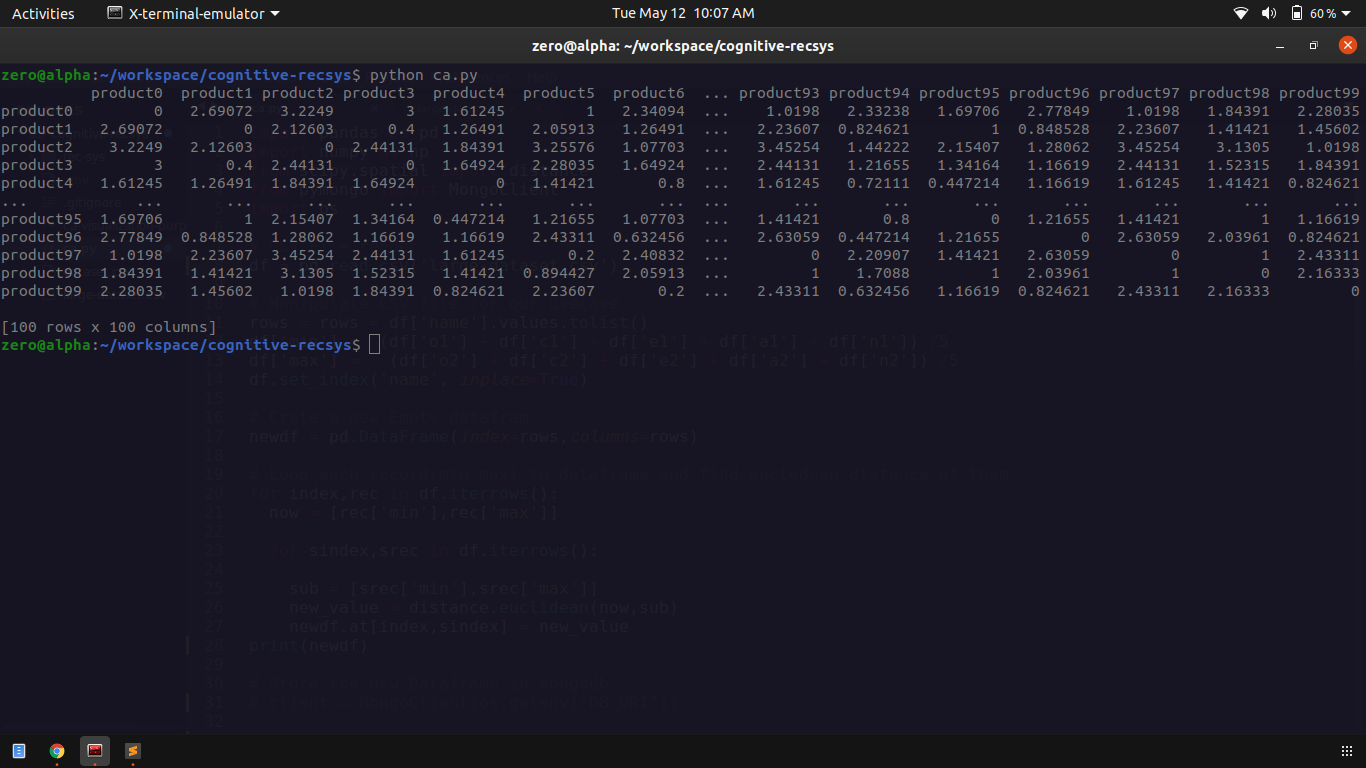
"mongoose": "^5.9.7"

}

}

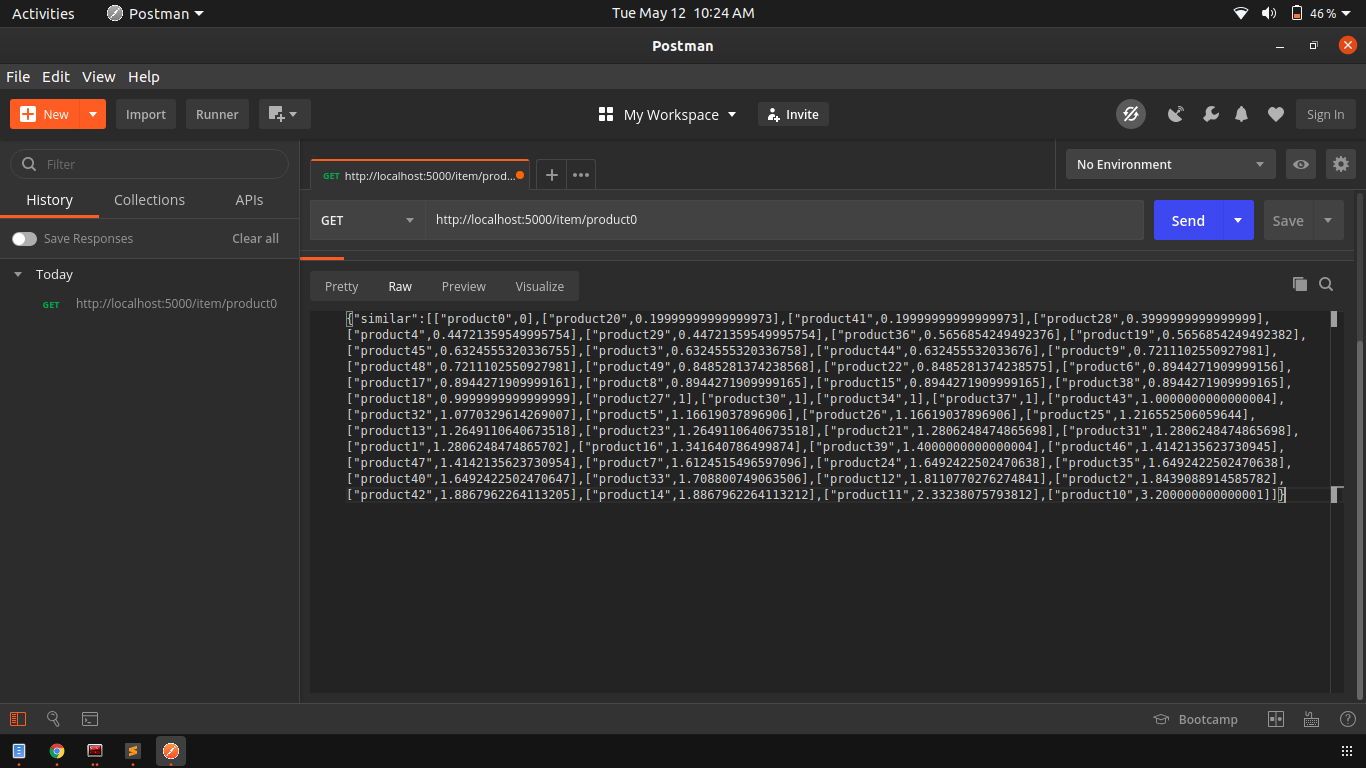
**7.2 OUTPUT**

**50 products psychographics score calculation:**



**Description:**

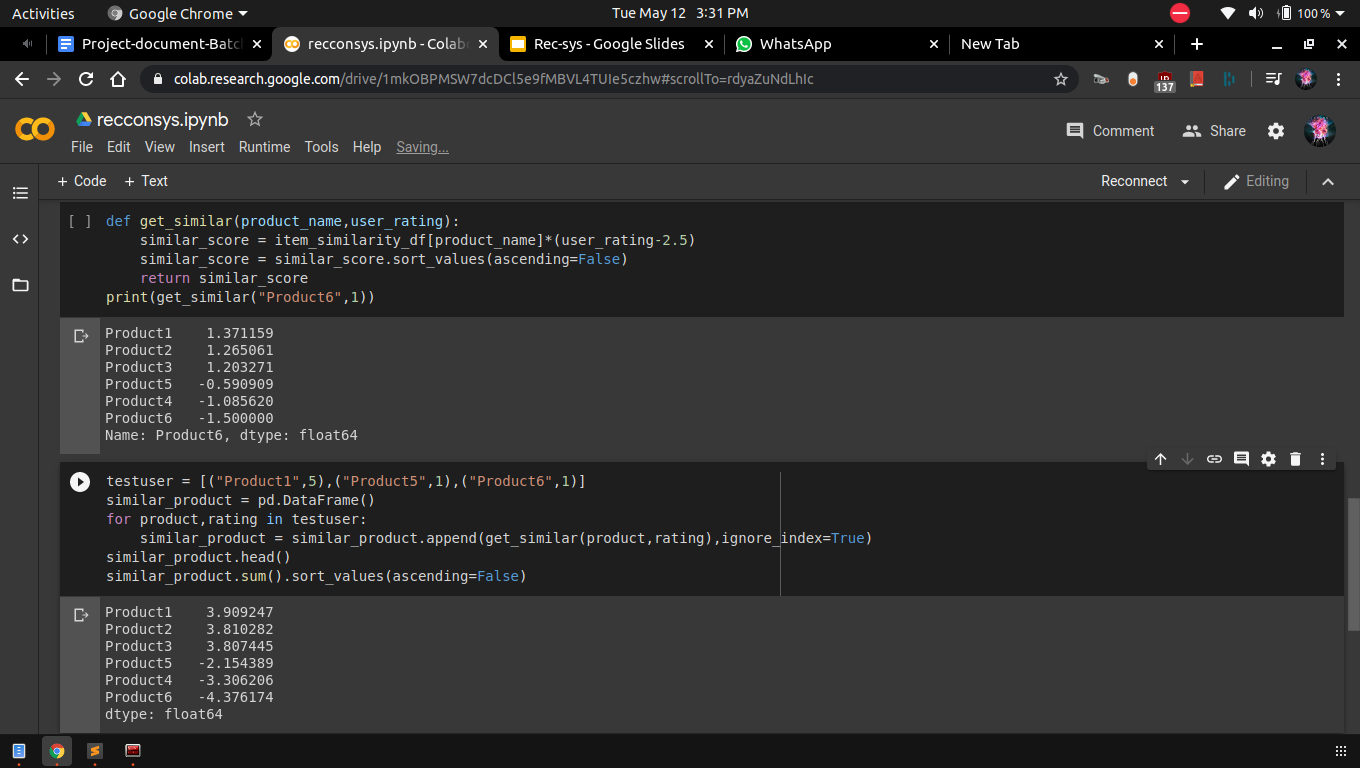
Here the python application reads the data sets and produce the similarity matrix.



**Description:**

Here the API will fetch the similar products of the given product which is passed in URL parameters or input. The fetched results are converted to JSON array of objects then return to the user

**Collaborative filtering output:**



**Description:**

Here the similarity score has been calculated by using collaborative based filtering algorithm.

