**Slip-11**

Q1 Write a java program to implement Adapter pattern to design Heart Model to Beat Model

Q2 Write a python program to find all null values in a given dataset and remove them.

import pandas as pd

import numpy as np

dict={'first score':[100,90,np.nan,95],

'second score':[30,45,56,np.nan],

'third score':[np.nan,40,80,98]}

df=pd.DataFrame(dict)

df.head()

df.isnull()

df.notnull()

#df=pd.DataFrame(dict)

df.fillna(0)

df.fillna(method='pad')

df.fillna(method='bfill')

df.replace(to\_replace=np.nan,value=-99)

df.dropna()

df.dropna(axis=1)

new\_data=df.dropna(axis=0)

new\_data

Q3 Create a node.js file that Select all records from the "customers" table, and delete the specified record.

var mysql = require('mysql');

var con = mysql.createConnection({

host: "localhost",

user: "root",

password: "Root@123",

database: "node"

});

con.connect(function(err) {

if (err) throw err;

var sql = "DELETE FROM customers WHERE name = 'pooja'";

con.query(sql, function (err, result) {

if (err) throw err;

console.log("Number of records deleted: " + result.affectedRows);

});

});

Slip-12

Q1 Write a Java Program to implement Decorator Pattern for interface Car to define the assemble() method and then decorate it to Sports car and Luxury Car .

Car.java

package com.journaldev.design.decorator;

public interface Car {

public void assemble();

}

BasicCar.java

package com.journaldev.design.decorator;

public class BasicCar implements Car {

@Override

public void assemble() {

System.out.print("Basic Car.");

}

}

CarDecorator.java

package com.journaldev.design.decorator;

public class CarDecorator implements Car {

protected Car car;

public CarDecorator(Car c){

this.car=c;

}

@Override

public void assemble() {

this.car.assemble();

}

}

SportsCar.java

package com.journaldev.design.decorator;

public class SportsCar extends CarDecorator {

public SportsCar(Car c) {

super(c);

}

@Override

public void assemble(){

super.assemble();

System.out.print(" Adding features of Sports Car.");

}

}

LuxuryCar.java

package com.journaldev.design.decorator;

public class LuxuryCar extends CarDecorator {

public LuxuryCar(Car c) {

super(c);

}

@Override

public void assemble(){

super.assemble();

System.out.print(" Adding features of Luxury Car.");

}

}

package com.journaldev.design.test;

import com.journaldev.design.decorator.BasicCar;

import com.journaldev.design.decorator.Car;

import com.journaldev.design.decorator.LuxuryCar;

import com.journaldev.design.decorator.SportsCar;

DecoratorPatternTest.java

public class DecoratorPatternTest {

public static void main(String[] args) {

Car sportsCar = new SportsCar(new BasicCar());

sportsCar.assemble();

System.out.println("\n\*\*\*\*\*");

Car sportsLuxuryCar = new SportsCar(new LuxuryCar(new BasicCar()));

sportsLuxuryCar.assemble();

}

}

Q2 Write a python program to make Categorical values in numeric format for a given dataset

import pandas as pd

#load Iris data set

iris = pd.read\_csv('Iris.csv')

iris.head()

iris['code']=pd.factorize(iris.Species)[0]

iris.tail()

iris.code.value\_counts()

Q3 Create a Simple Web Server using node js.

var http = require('http'); // 1 - Import Node.js core module

var server = http.createServer(function (req, res) { // 2 - creating server

//handle incomming requests here..

});

server.listen(5000); //3 - listen for any incoming requests

console.log('Node.js web server at port 5000 is running..')

**Slip-13**

Q1 Write a Java Program to implement an Adapter design pattern in mobile charger. Define two classes – Volt (to measure volts) and Socket (producing constant volts of 120V). Build an adapter that can produce 3 volts, 12 volts and default 120 volts.

|  |  |  |
| --- | --- | --- |
| Implements Adapter pattern using Class Adapter |  |  |

Voltage.java

public class Voltage

{

private int voltage;

public Voltage(int v)

{

this.voltage = v;

}

public int getVolts()

{

return voltage;

}

public void setVolts(int voltage)

{

this.voltage = voltage;

}

}

SocketAdapter.java

public interface SocketAdapter

{

public Voltage get120Voltage();

public Voltage get12Voltage();

public Voltage get3VVoltage();

}

Socket.java

public class Socket

{

public Voltage getVoltage()

{

return new Voltage(120); //In India 240 is the default voltage

}

}

SocketAdapterImpl.java

public class SocketAdapterImpl extends Socket implements SocketAdapter

{

//Using Composition for adapter pattern

private Socket sock = new Socket();

private Voltage convertVolt(Voltage v, int i)

{

return new Voltage(v.getVolts() / i);

}

@Override

public Voltage get120Voltage()

{

return sock.getVoltage();

}

@Override

public Voltage get12Voltage()

{

Voltage v = sock.getVoltage();

return convertVolt(v, 20);

}

@Override

public Voltage get3VVoltage()

{

Voltage v = sock.getVoltage();

return convertVolt(v, 80);

}

}

AdapterEx.java

public class AdapterEx

{

public static void main(String[] args)

{

SocketAdapter socketAdapter = new SocketAdapterImpl();

Voltage voltage12 = socketAdapter.get12Voltage();

System.out.println(voltage12.getVolts());

Voltage voltage3 = socketAdapter.get3VVoltage();

System.out.println(voltage3.getVolts());

Voltage voltage120 = socketAdapter.get120Voltage();

System.out.println(voltage120.getVolts());

}

}

2) Write a Python program to prepare Scatter Plot for Iris Dataset

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

iris = pd.read\_csv("Iris.csv") # Reading the dataset “Iris.csv”.

print (iris.head(10)) # head() will display the top rows of the dataset, the default value of this function is 5,

#that is it will show top 5 rows when no argument is given to it.

plt.plot(iris.Id, iris["SepalLengthCm"],"r--")

plt.show #plt.show () will display the current figure that you are working on

iris.plot(kind ="scatter", x ='SepalLengthCm', y ='PetalLengthCm')

plt.grid() # grid () function to add grid lines to the plot

Q.3 Using node js create a User Login System.

**Slip-14**

Q1 Write a Java Program to implement Command Design Pattern for Command Interface with execute() .Use this to create variety of commands for LightOnCommand, LightOffCommand, GarageDoorUpCommand, StereoOnWithCDComman.

Command.java

public interface Command {

public void execute();

}

LightOnCommand.java

public class LightOnCommand implements Command {

Light light;

public LightOnCommand(Light light) {

this.light = light;

}

public void execute() {

light.on();

}

}

LightOffCommand.java

public class LightOffCommand implements Command {

Light light;

public LightOffCommand(Light light) {

this.light = light;

}

public void execute() {

light.off();

}

}

Light.java

public class Light {

String location = "";

public Light(String location) {

this.location = location;

}

public void on() {

System.out.println(location + " light is on");

}

public void off() {

System.out.println(location + " light is off");

}

}

StereoOnWithCDCommand.java

public class StereoOnWithCDCommand implements Command {

Stereo stereo;

public StereoOnWithCDCommand(Stereo stereo) {

this.stereo = stereo;

}

public void execute() {

stereo.on();

stereo.setCD();

stereo.setVolume(11);

}

}

StereoOffCommand.java

public class StereoOffCommand implements Command {

Stereo stereo;

public StereoOffCommand(Stereo stereo) {

this.stereo = stereo;

}

public void execute() {

stereo.off();

}

}

RemoteControlWithUndo.java

public class RemoteControlWithUndo {

Command[] onCommands;

Command[] offCommands;

Command undoCommand;

public RemoteControlWithUndo() {

onCommands = new Command[7];

offCommands = new Command[7];

Command noCommand = new NoCommand();

for(int i=0;i<7;i++) {

onCommands[i] = noCommand;

offCommands[i] = noCommand;

}

undoCommand = noCommand;

}

public void setCommand(int slot, Command onCommand, Command offCommand) {

onCommands[slot] = onCommand;

offCommands[slot] = offCommand;

}

public void onButtonWasPushed(int slot) {

onCommands[slot].execute();

undoCommand = onCommands[slot];

}

public void offButtonWasPushed(int slot) {

offCommands[slot].execute();

undoCommand = offCommands[slot];

}

public void undoButtonWasPushed() {

undoCommand.undo();

}

public String toString() {

StringBuffer stringBuff = new StringBuffer();

stringBuff.append("\n------ Remote Control -------\n");

for (int i = 0; i < onCommands.length; i++) {

stringBuff.append("[slot " + i + "] " + onCommands[i].getClass().getName()

+ " " + offCommands[i].getClass().getName() + "\n");

}

stringBuff.append("[undo] " + undoCommand.getClass().getName() + "\n");

return stringBuff.toString();

}

}

NoCommand.java

public class NoCommand implements Command {

public void execute() { }

public void undo() { }

}

RemoteLoader.java

public class RemoteLoader {

public static void main(String[] args) {

RemoteControlWithUndo remoteControl = new RemoteControlWithUndo();

Light livingRoomLight = new Light("Living Room");

Light kitchenLight = new Light("Kitchen");

CeilingFan ceilingFan= new CeilingFan("Living Room");

GarageDoor garageDoor = new GarageDoor("");

Stereo stereo = new Stereo("Living Room");

LightOnCommand livingRoomLightOn =

new LightOnCommand(livingRoomLight);

LightOffCommand livingRoomLightOff =

new LightOffCommand(livingRoomLight);

LightOnCommand kitchenLightOn =

new LightOnCommand(kitchenLight);

LightOffCommand kitchenLightOff =

new LightOffCommand(kitchenLight);

CeilingFanOnCommand ceilingFanOn =

new CeilingFanOnCommand(ceilingFan);

CeilingFanOffCommand ceilingFanOff =

new CeilingFanOffCommand(ceilingFan);

GarageDoorUpCommand garageDoorUp =

new GarageDoorUpCommand(garageDoor);

GarageDoorDownCommand garageDoorDown =

new GarageDoorDownCommand(garageDoor);

StereoOnWithCDCommand stereoOnWithCD =

new StereoOnWithCDCommand(stereo);

StereoOffCommand stereoOff =

new StereoOffCommand(stereo);

remoteControl.setCommand(0, livingRoomLightOn, livingRoomLightOff);

remoteControl.setCommand(1, kitchenLightOn, kitchenLightOff);

remoteControl.setCommand(2, ceilingFanOn, ceilingFanOff);

remoteControl.setCommand(3, stereoOnWithCD, stereoOff);

System.out.println(remoteControl);

remoteControl.onButtonWasPushed(0);

remoteControl.offButtonWasPushed(0);

remoteControl.onButtonWasPushed(1);

remoteControl.offButtonWasPushed(1);

remoteControl.onButtonWasPushed(2);

remoteControl.offButtonWasPushed(2);

remoteControl.onButtonWasPushed(3);

remoteControl.offButtonWasPushed(3);

}

}

2) Write a python program to find all null values in a given dataset and remove them.

import pandas as pd

import numpy as np

dict={'first score':[100,90,np.nan,95],

'second score':[30,45,56,np.nan],

'third score':[np.nan,40,80,98]}

df=pd.DataFrame(dict)

df.head()

df.isnull()

df.notnull()

#df=pd.DataFrame(dict)

df.fillna(0)

df.fillna(method='pad')

df.fillna(method='bfill')

df.replace(to\_replace=np.nan,value=-99)

df.dropna()

df.dropna(axis=1)

new\_data=df.dropna(axis=0)

new\_data

Q3 Write node js script to interact with the filesystem, and serve a web page from a file .

**Slip-15**

Q1 Write a Java Program to implement Facade Design Pattern for HomeTheater.

Amplifier.java

public class Amplifier {

String description;

Tuner tuner;

DvdPlayer dvd;

CdPlayer cd;

public Amplifier(String description) {

this.description = description;

}

public void on() {

System.out.println(description + " on");

}

public void off() {

System.out.println(description + " off");

}

public void setStereoSound() {

System.out.println(description + " stereo mode on");

}

public void setSurroundSound() {

System.out.println(description + " surround sound on (5 speakers, 1 subwoofer)");

}

public void setVolume(int level) {

System.out.println(description + " setting volume to " + level);

}

public void setTuner(Tuner tuner) {

System.out.println(description + " setting tuner to " + dvd);

this.tuner = tuner;

}

public void setDvd(DvdPlayer dvd) {

System.out.println(description + " setting DVD player to " + dvd);

this.dvd = dvd;

}

public void setCd(CdPlayer cd) {

System.out.println(description + " setting CD player to " + cd);

this.cd = cd;

}

public String toString() {

return description;

}

}

CdPlayer.java

public class CdPlayer {

String description;

int currentTrack;

Amplifier amplifier;

String title;

public CdPlayer(String description, Amplifier amplifier) {

this.description = description;

this.amplifier = amplifier;

}

public void on() {

System.out.println(description + " on");

}

public void off() {

System.out.println(description + " off");

}

public void eject() {

title = null;

System.out.println(description + " eject");

}

public void play(String title) {

this.title = title;

currentTrack = 0;

System.out.println(description + " playing \"" + title + "\"");

}

public void play(int track) {

if (title == null) {

System.out.println(description + " can't play track " + currentTrack +

", no cd inserted");

} else {

currentTrack = track;

System.out.println(description + " playing track " + currentTrack);

}

}

public void stop() {

currentTrack = 0;

System.out.println(description + " stopped");

}

public void pause() {

System.out.println(description + " paused \"" + title + "\"");

}

public String toString() {

return description;

}

}

DvdPlayer.java

public class DvdPlayer {

String description;

int currentTrack;

Amplifier amplifier;

String movie;

public DvdPlayer(String description, Amplifier amplifier) {

this.description = description;

this.amplifier = amplifier;

}

public void on() {

System.out.println(description + " on");

}

public void off() {

System.out.println(description + " off");

}

public void eject() {

movie = null;

System.out.println(description + " eject");

}

public void play(String movie) {

this.movie = movie;

currentTrack = 0;

System.out.println(description + " playing \"" + movie + "\"");

}

public void play(int track) {

if (movie == null) {

System.out.println(description + " can't play track " + track + " no dvd inserted");

} else {

currentTrack = track;

System.out.println(description + " playing track " + currentTrack + " of \"" + movie + "\"");

}

}

public void stop() {

currentTrack = 0;

System.out.println(description + " stopped \"" + movie + "\"");

}

public void pause() {

System.out.println(description + " paused \"" + movie + "\"");

}

public void setTwoChannelAudio() {

System.out.println(description + " set two channel audio");

}

public void setSurroundAudio() {

System.out.println(description + " set surround audio");

}

public String toString() {

return description;

}

}

Projector.java

public class Projector {

String description;

DvdPlayer dvdPlayer;

public Projector(String description, DvdPlayer dvdPlayer) {

this.description = description;

this.dvdPlayer = dvdPlayer;

}

public void on() {

System.out.println(description + " on");

}

public void off() {

System.out.println(description + " off");

}

public void wideScreenMode() {

System.out.println(description + " in widescreen mode (16x9 aspect ratio)");

}

public void tvMode() {

System.out.println(description + " in tv mode (4x3 aspect ratio)");

}

public String toString() {

return description;

}

}

TheaterLights.java

public class TheaterLights {

String description;

public TheaterLights(String description) {

this.description = description;

}

public void on() {

System.out.println(description + " on");

}

public void off() {

System.out.println(description + " off");

}

public void dim(int level) {

System.out.println(description + " dimming to " + level + "%");

}

public String toString() {

return description;

}

}

Screen.java

public class Screen {

String description;

public Screen(String description) {

this.description = description;

}

public void up() {

System.out.println(description + " going up");

}

public void down() {

System.out.println(description + " going down");

}

public String toString() {

return description;

}

}

PopcornPopper.java

public class PopcornPopper {

String description;

public PopcornPopper(String description) {

this.description = description;

}

public void on() {

System.out.println(description + " on");

}

public void off() {

System.out.println(description + " off");

}

public void pop() {

System.out.println(description + " popping popcorn!");

}

public String toString() {

return description;

}

}

HomeTheaterFacade.java

public class HomeTheaterFacade {

Amplifier amp;

Tuner tuner;

DvdPlayer dvd;

CdPlayer cd;

Projector projector;

TheaterLights lights;

Screen screen;

PopcornPopper popper;

public HomeTheaterFacade(Amplifier amp,

Tuner tuner,

DvdPlayer dvd,

CdPlayer cd,

Projector projector,

Screen screen,

TheaterLights lights,

PopcornPopper popper) {

this.amp = amp;

this.tuner = tuner;

this.dvd = dvd;

this.cd = cd;

this.projector = projector;

this.screen = screen;

this.lights = lights;

this.popper = popper;

}

public void watchMovie(String movie) {

System.out.println("Get ready to watch a movie...");

popper.on();

popper.pop();

lights.dim(10);

screen.down();

projector.on();

projector.wideScreenMode();

amp.on();

amp.setDvd(dvd);

amp.setSurroundSound();

amp.setVolume(5);

dvd.on();

dvd.play(movie);

}

public void endMovie() {

System.out.println("Shutting movie theater down...");

popper.off();

lights.on();

screen.up();

projector.off();

amp.off();

dvd.stop();

dvd.eject();

dvd.off();

}

public void listenToCd(String cdTitle) {

System.out.println("Get ready for an audiopile experence...");

lights.on();

amp.on();

amp.setVolume(5);

amp.setCd(cd);

amp.setStereoSound();

cd.on();

cd.play(cdTitle);

}

public void endCd() {

System.out.println("Shutting down CD...");

amp.off();

amp.setCd(cd);

cd.eject();

cd.off();

}

public void listenToRadio(double frequency) {

System.out.println("Tuning in the airwaves...");

tuner.on();

tuner.setFrequency(frequency);

amp.on();

amp.setVolume(5);

amp.setTuner(tuner);

}

public void endRadio() {

System.out.println("Shutting down the tuner...");

tuner.off();

amp.off();

}

}

HomeTheaterTestDrive.java

public class HomeTheaterTestDrive {

public static void main(String[] args) {

Amplifier amp = new Amplifier("Top-O-Line Amplifier");

Tuner tuner = new Tuner("Top-O-Line AM/FM Tuner", amp);

DvdPlayer dvd = new DvdPlayer("Top-O-Line DVD Player", amp);

CdPlayer cd = new CdPlayer("Top-O-Line CD Player", amp);

Projector projector = new Projector("Top-O-Line Projector", dvd);

TheaterLights lights = new TheaterLights("Theater Ceiling Lights");

Screen screen = new Screen("Theater Screen");

PopcornPopper popper = new PopcornPopper("Popcorn Popper");

HomeTheaterFacade homeTheater =

new HomeTheaterFacade(amp, tuner, dvd, cd,

projector, screen, lights, popper);

homeTheater.watchMovie("Raiders of the Lost Ark");

homeTheater.endMovie();

}

}

Tuner.java

public class Tuner {

String description;

Amplifier amplifier;

double frequency;

public Tuner(String description, Amplifier amplifier) {

this.description = description;

}

public void on() {

System.out.println(description + " on");

}

public void off() {

System.out.println(description + " off");

}

public void setFrequency(double frequency) {

System.out.println(description + " setting frequency to " + frequency);

this.frequency = frequency;

}

public void setAm() {

System.out.println(description + " setting AM mode");

}

public void setFm() {

System.out.println(description + " setting FM mode");

}

public String toString() {

return description;

}

}

2) Write a python program to make Categorical values in numeric format for a given dataset

import pandas as pd

#load Iris data set

iris = pd.read\_csv('Iris.csv')

iris.head()

iris['code']=pd.factorize(iris.Species)[0]

iris.tail()

iris.code.value\_counts()

3) Write node js script to build Your Own Node.js Module. Use require (‘http’) module is a built-in Node module that invokes the functionality of the HTTP library to create a local server. Also use the export statement to make functions in your module available externally. Create a new text file to contain the functions in your module called, “modules.js” and add this function to return today’s date and time.

var http = require('http');

var dt = require('./myfirstmodule17');

http.createServer(function (req, res) {

res.writeHead(200, {'Content-Type': 'text/html'});

res.write("The date and time are currently: " + dt.myDateTime());

res.end();

}).listen(8080);

Myfirstmodule:

exports.myDateTime = function () {

return Date();

};

**Slip-16**

Q1 Write a Java Program to implement Observer Design Pattern for number conversion. Accept a number in Decimal form and represent it in Hexadecimal, Octal and Binary. Change the Number and it reflects in other forms also

Subject.java

import java.util.ArrayList;

import java.util.List;

public class Subject {

private List<Observer> observers = new ArrayList<Observer>();

private int state;

public int getState() {

return state;

}

public void setState(int state) {

this.state = state;

notifyAllObservers();

}

public void attach(Observer observer){

observers.add(observer);

}

public void notifyAllObservers(){

for (Observer observer : observers) {

observer.update();

}

}

}

Observer.java

public abstract class Observer {

protected Subject subject;

public abstract void update();

}

BinaryObserver.java

public class BinaryObserver extends Observer{

public BinaryObserver(Subject subject){

this.subject = subject;

this.subject.attach(this);

}

@Override

public void update() {

System.out.println( "Binary String: " + Integer.toBinaryString( subject.getState() ) );

}

}

OctalObserver.java

public class OctalObserver extends Observer{

public OctalObserver(Subject subject){

this.subject = subject;

this.subject.attach(this);

}

@Override

public void update() {

System.out.println( "Octal String: " + Integer.toOctalString( subject.getState() ) );

}

}

HexaObserver.java

public class HexaObserver extends Observer{

public HexaObserver(Subject subject){

this.subject = subject;

this.subject.attach(this);

}

@Override

public void update() {

System.out.println( "Hex String: " + Integer.toHexString( subject.getState() ).toUpperCase() );

}

}

ObserverPatternDemo.java

public class ObserverPatternDemo {

public static void main(String[] args) {

Subject subject = new Subject();

new HexaObserver(subject);

new OctalObserver(subject);

new BinaryObserver(subject);

System.out.println("First state change: 15");

subject.setState(15);

System.out.println("Second state change: 10");

subject.setState(10);

}

}

2) Write a python program to Implement Simple Linear Regression for predicting house price.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_predict

data = pd.read\_csv(r'kc\_house\_data.csv')

data.head(5)

print(data.shape)

# Make a list of importatnt feature which is needed to be incuding in training data

f = ['price', 'bedrooms', 'bathrooms', 'sqft\_living', 'floors', 'condition', 'sqft\_above', 'sqft\_basement', 'yr\_built',

'yr\_renovated']

data = data[f]

print(data.shape)

# Drop the missing values

data = data.dropna()

print(data.shape)

# Get the statictial information of the dataset

data.describe()

# Now,Divide the dataset into two parts:independent variable and dependent variable

X = data[f[1:]]

y = data['price']

# Split the dataset into training data and testing data

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.2, random\_state=42)

print(X\_train.shape)

print(X\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

# Fit the regression model

lr = LinearRegression() # Create object of linear regression class

lr.fit(X\_train,y\_train) #fit training data

print(lr.coef\_)

# Create the Prediction

y\_test\_predict = lr.predict(X\_test)

print(y\_test\_predict.shape)

# Plot the error

g=plt.plot((y\_test - y\_test\_predict),marker='o',linestyle='')

# # Fit the regression model without b(w0)

lr = LinearRegression(fit\_intercept=False)

lr.fit(X\_train,y\_train)

y\_test\_predict = lr.predict(X\_test)

g=plt.plot((y\_test - y\_test\_predict),marker='o',linestyle='')

3) Create a js file named main.js for event-driven application. There should be a main loop that listens for events, and then triggers a callback function when one of those events is detected.

// Import events module

var events = require('events');

// Create an eventEmitter object

var eventEmitter = new events.EventEmitter();

// Create an event handler as follows

var connectHandler = function connected() {

console.log('connection succesful.');

// Fire the data\_received event

eventEmitter.emit('data\_received');

}

// Bind the connection event with the handler

eventEmitter.on('connection', connectHandler);

// Bind the data\_received event with the anonymous function

eventEmitter.on('data\_received', function() {

console.log('data received succesfully.');

});

// Fire the connection event

eventEmitter.emit('connection');

console.log("Program Ended.");

**Slip-17**

Q1 Write a Java Program to implement Abstract Factory Pattern for Shape interface.

Shape.java

public interface Shape {

void draw();

}

public class RoundedRectangle implements Shape {

@Override

public void draw() {

System.out.println("Inside RoundedRectangle::draw() method.");

}

}

RoundedSquare.java

public class RoundedSquare implements Shape {

@Override

public void draw() {

System.out.println("Inside RoundedSquare::draw() method.");

}

}

Rectangle.java

public class Rectangle implements Shape {

@Override

public void draw() {

System.out.println("Inside Rectangle::draw() method.");

}

}

Step 3

Create an Abstract class to get factories for Normal and Rounded Shape Objects.

AbstractFactory.java

public abstract class AbstractFactory {

abstract Shape getShape(String shapeType) ;

}

Step 4

Create Factory classes extending AbstractFactory to generate object of concrete class based on given information.

ShapeFactory.java

public class ShapeFactory extends AbstractFactory {

@Override

public Shape getShape(String shapeType){

if(shapeType.equalsIgnoreCase("RECTANGLE")){

return new Rectangle();

}else if(shapeType.equalsIgnoreCase("SQUARE")){

return new Square();

}

return null;

}

}

RoundedShapeFactory.java

public class RoundedShapeFactory extends AbstractFactory {

@Override

public Shape getShape(String shapeType){

if(shapeType.equalsIgnoreCase("RECTANGLE")){

return new RoundedRectangle();

}else if(shapeType.equalsIgnoreCase("SQUARE")){

return new RoundedSquare();

}

return null;

}

}

Step 5

Create a Factory generator/producer class to get factories by passing an information such as Shape

FactoryProducer.java

public class FactoryProducer {

public static AbstractFactory getFactory(boolean rounded){

if(rounded){

return new RoundedShapeFactory();

}else{

return new ShapeFactory();

}

}

}

Step 6

Use the FactoryProducer to get AbstractFactory in order to get factories of concrete classes by passing an information such as type.

AbstractFactoryPatternDemo.java

public class AbstractFactoryPatternDemo {

public static void main(String[] args) {

//get shape factory

AbstractFactory shapeFactory = FactoryProducer.getFactory(false);

//get an object of Shape Rectangle

Shape shape1 = shapeFactory.getShape("RECTANGLE");

//call draw method of Shape Rectangle

shape1.draw();

//get an object of Shape Square

Shape shape2 = shapeFactory.getShape("SQUARE");

//call draw method of Shape Square

shape2.draw();

//get shape factory

AbstractFactory shapeFactory1 = FactoryProducer.getFactory(true);

//get an object of Shape Rectangle

Shape shape3 = shapeFactory1.getShape("RECTANGLE");

//call draw method of Shape Rectangle

shape3.draw();

//get an object of Shape Square

Shape shape4 = shapeFactory1.getShape("SQUARE");

//call draw method of Shape Square

shape4.draw();

}

}

2) Write a python program to implement Multiple Linear Regression for a given dataset.

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('50\_Startups.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])],

remainder='passthrough')

X = np.array(ct.fit\_transform(X))

print(X)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

df = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred})

df

3) Write node js application that transfer a file as an attachment on web and enables browser to prompt the user to download file using express js.

var express = require('express');

var app = express();

var PORT = 3000;

app.get('/', function(req, res){

res.download('hello.txt');

});

app.listen(PORT, function(err){

if (err) console.log(err);

console.log("Server listening on PORT", PORT);

});

**Slip-18**

Q1 Write a JAVA Program to implement built-in support (java.util.Observable) Weather station with members temperature, humidity, pressure and methods mesurmentsChanged(), setMesurment(), getTemperature(), getHumidity(), getPressure().

import java.util.\*;

interface Observer {

public void update(float temp, float humidity, float pressure);

}

interface DisplayElement {

public void display();

}

interface Subject {

public void registerObserver(Observer o);

public void removeObserver(Observer o);

public void notifyObservers();

}

class WeatherData implements Subject {

private ArrayList<Observer> observers;

private float temperature;

private float humidity;

private float pressure;

public WeatherData() {

observers = new ArrayList<>();

}

public void registerObserver(Observer o) {

observers.add(o);

}

public void removeObserver(Observer o) {

int i = observers.indexOf(o);

if (i >= 0) {

observers.remove(i);

}

}

public void notifyObservers() {

for (int i = 0; i < observers.size(); i++) {

Observer observer = (Observer) observers.get(i);

observer.update(temperature, humidity, pressure);

}

}

public void measurementsChanged() {

notifyObservers();

}

public void setMeasurements(float temperature, float humidity, float pressure) {

this.temperature = temperature;

this.humidity = humidity;

this.pressure = pressure;

measurementsChanged();

}

public float getTemperature() {

return temperature;

}

public float getHumidity() {

return humidity;

}

public float getPressure() {

return pressure;

}

}

class ForecastDisplay implements Observer, DisplayElement {

private float currentPressure = 29.92f;

private float lastPressure;

private WeatherData weatherData;

public ForecastDisplay(WeatherData weatherData) {

this.weatherData = weatherData;

weatherData.registerObserver(this);

}

public void update(float temp, float humidity, float pressure) {

lastPressure = currentPressure;

currentPressure = pressure;

display();

}

public void display() {

System.out.print("Forecast: ");

if (currentPressure > lastPressure) {

System.out.println("Improving weather on the way!");

} else if (currentPressure == lastPressure) {

System.out.println("More of the same");

} else if (currentPressure < lastPressure) {

System.out.println("Watch out for cooler weather, rainy weather");

}

}

}

class HeatIndexDisplay implements Observer, DisplayElement {

float heatIndex = 0.0f;

private WeatherData weatherData;

public HeatIndexDisplay(WeatherData weatherData) {

this.weatherData = weatherData;

weatherData.registerObserver(this);

}

public void update(float t, float rh, float pressure) {

heatIndex = computeHeatIndex(t, rh);

display();

}

private float computeHeatIndex(float t, float rh) {

float index = (float) ((16.923 + (0.185212 \* t) + (5.37941 \* rh) - (0.100254 \* t \* rh)

+ (0.00941695 \* (t \* t)) + (0.00728898 \* (rh \* rh))

+ (0.000345372 \* (t \* t \* rh)) - (0.000814971 \* (t \* rh \* rh)) +

(0.0000102102 \* (t \* t \* rh \* rh)) - (0.000038646 \* (t \* t \* t)) + (0.0000291583 \*

(rh \* rh \* rh))

+ (0.00000142721 \* (t \* t \* t \* rh)) +

(0.000000197483 \* (t \* rh \* rh \* rh)) - (0.0000000218429 \* (t \* t \* t \* rh \* rh)) +

0.000000000843296 \* (t \* t \* rh \* rh \* rh)) -

(0.0000000000481975 \* (t \* t \* t \* rh \* rh \* rh)));

return index;

}

public void display() {

System.out.println("Heat index is " + heatIndex);

}

}

class StatisticsDisplay implements Observer, DisplayElement {

private float maxTemp = 0.0f;

private float minTemp = 200;

private float tempSum = 0.0f;

private int numReadings;

private WeatherData weatherData;

public StatisticsDisplay(WeatherData weatherData) {

this.weatherData = weatherData;

weatherData.registerObserver(this);

}

public void update(float temp, float humidity, float pressure) {

tempSum += temp;

numReadings++;

if (temp > maxTemp) {

maxTemp = temp;

}

if (temp < minTemp) {

minTemp = temp;

}

display();

}

public void display() {

System.out.println("Avgerage/Maximum/Minimum temperature = " + (tempSum / numReadings)

+ "/" + maxTemp + "/" + minTemp);

}

}

class CurrentConditionsDisplay implements Observer, DisplayElement {

private float temperature;

private float humidity;

private Subject weatherData;

public CurrentConditionsDisplay(Subject weatherData) {

this.weatherData = weatherData;

weatherData.registerObserver(this);

}

public void update(float temperature, float humidity, float pressure) {

this.temperature = temperature;

this.humidity = humidity;

display();

}

public void display() {

System.out.println("Current conditions: " + temperature

+ "F degrees and " + humidity + "% humidity");

}

}

class weather {

public static void main(String[] args) {

WeatherData weatherData = new WeatherData();

CurrentConditionsDisplay currentDisplay = new CurrentConditionsDisplay(weatherData);

StatisticsDisplay statisticsDisplay = new StatisticsDisplay(weatherData);

ForecastDisplay forecastDisplay = new ForecastDisplay(weatherData);

weatherData.setMeasurements(70, 55, 40.4f);

weatherData.setMeasurements(72, 60, 39.2f);

weatherData.setMeasurements(68, 80, 39.2f);

}

}

2) Write a python program to implement Polynomial Linear Regression for given dataset.

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Position\_Salaries.csv')

X = dataset.iloc[:, 1:-1].values

y = dataset.iloc[:, -1].values

dataset.head(5)

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

poly\_reg = PolynomialFeatures(degree = 4)

X\_poly = poly\_reg.fit\_transform(X)

lin\_reg = LinearRegression()

lin\_reg.fit(X\_poly, y)

y\_pred = lin\_reg.predict(X\_poly)

df = pd.DataFrame({'Real Values':y, 'Predicted Values':y\_pred})

df

X\_grid = np.arange(min(X), max(X), 0.1)

X\_grid = X\_grid.reshape((len(X\_grid), 1))

plt.scatter(X, y, color = 'red')

plt.scatter(X, y\_pred, color = 'green')

plt.plot(X\_grid,

lin\_reg.predict(poly\_reg.fit\_transform(X\_grid)), color =

'black')

plt.title('Polynomial Regression')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

Q3 Create your Django app in which after running the server, you should see on the browser, the text “Hello! I am learning Django”, which you defined in the index view.

<html>

<head>

<script type="text/javascript">

function validate()

{

var regName=/^[a-zA-z]+[a-zA-Z]+$/;

var fname=document.getElementById("txtfname").value;

var lname=document.getElementById("txtlname").value;

var age=document.getElementById("txtage").value;

var mobno=document.getElementById("txtmobno").value;

if(age<18||age>50)

alert("student age must be 18 to 50");

if(!regName.test(fname))

alert("invalid name is given");

else

alert("valid name is given");

}

</script>

</head>

<body>

<form>

enter student first name

<input type="text" name="txtfname" id="txtfname"><br>

enter student last name

<input type="text" name="txtlname" id="txtlname"><br>

enter student age

<input type="text" name="txtage" id="txtage"><br>

enter mobile no

<input type="text" name="txtmobno" id="txtmobno"><br>

<input type="button" value="validate" onclick="validate()">

</form>

</body>

</html>

**Slip-19**

Q1 Write a Java Program to implement Factory method for Pizza Store with createPizza(), orederPizza(), prepare(), Bake(), cut(), box(). Use this to create variety of pizza’s like NyStyleCheesePizza, ChicagoStyleCheesePizza etc.

import java.util.ArrayList;

class ChicagoPizzaStore extends PizzaStore

{Pizza createPizza(String item)

{if (item.equals("cheese"))

{return new ChicagoStyleCheesePizza();

}

else if (item.equals("veggie"))

{return new ChicagoStyleVeggiePizza();

}

else if (item.equals("clam"))

{return new ChicagoStyleClamPizza();

}

else if (item.equals("pepperoni"))

{return new ChicagoStylePepperoniPizza();

}

else return null;

}}

class ChicagoStyleCheesePizza extends Pizza

{public ChicagoStyleCheesePizza()

{

name = "Chicago Style Deep Dish Cheese Pizza";

dough = "Extra Thick Crust Dough";

sauce = "Plum Tomato Sauce";

toppings.add("Shredded Mozzarella Cheese");

}

void cut()

{System.out.println("Cutting the pizza into square slices");

}}

class ChicagoStyleClamPizza extends Pizza

{public ChicagoStyleClamPizza()

{

name = "Chicago Style Clam Pizza";

dough = "Extra Thick Crust Dough";

sauce = "Plum Tomato Sauce";

toppings.add("Shredded Mozzarella Cheese");

toppings.add("Frozen Clams from Chesapeake Bay");

}

void cut()

{System.out.println("Cutting the pizza into square slices");

}}

class ChicagoStylePepperoniPizza extends Pizza

{public ChicagoStylePepperoniPizza()

{

name = "Chicago Style Pepperoni Pizza";

dough = "Extra Thick Crust Dough";

sauce = "Plum Tomato Sauce";

toppings.add("Shredded Mozzarella Cheese");

toppings.add("Black Olives");

toppings.add("Spinach");

toppings.add("Eggplant");

toppings.add("Sliced Pepperoni");

}

void cut()

{System.out.println("Cutting the pizza into square slices");

}}

class ChicagoStyleVeggiePizza extends Pizza

{public ChicagoStyleVeggiePizza()

{name = "Chicago Deep Dish Veggie Pizza";

dough = "Extra Thick Crust Dough";

sauce = "Plum Tomato Sauce";

toppings.add("Shredded Mozzarella Cheese");

toppings.add("Black Olives");

toppings.add("Spinach");

toppings.add("Eggplant");

}

void cut()

{System.out.println("Cutting the pizza into square slices");

}}

class DependentPizzaStore

{public Pizza createPizza(String style, String type)

{ Pizza pizza = null;

if (style.equals("NY"))

{if (type.equals("cheese"))

{pizza = new NYStyleCheesePizza();

}

else if (type.equals("veggie"))

{pizza = new NYStyleVeggiePizza();

}

else if (type.equals("clam"))

{pizza = new NYStyleClamPizza();

}

else if (type.equals("pepperoni"))

{pizza = new NYStylePepperoniPizza();

}}

else if (style.equals("Chicago"))

{if (type.equals("cheese"))

{pizza = new ChicagoStyleCheesePizza();

}

else if (type.equals("veggie"))

{pizza = new ChicagoStyleVeggiePizza();

}

else if (type.equals("clam"))

{pizza = new ChicagoStyleClamPizza();

}

else if (type.equals("pepperoni"))

{pizza = new ChicagoStylePepperoniPizza();

}}

else

{System.out.println("Error: invalid type of pizza");

return null;

}

pizza.prepare();

pizza.bake();

pizza.cut();

pizza.box();

return pizza;

}}

class NYPizzaStore extends PizzaStore

{Pizza createPizza(String item)

{if (item.equals("cheese"))

{return new NYStyleCheesePizza();

}

else if (item.equals("veggie"))

{return new NYStyleVeggiePizza();

}

else if (item.equals("clam"))

{return new NYStyleClamPizza();

}

else if (item.equals("pepperoni"))

{return new NYStylePepperoniPizza();

}

else return null;

}}

class NYStyleCheesePizza extends Pizza

{public NYStyleCheesePizza()

{

name = "NY Style Sauce and Cheese Pizza";

dough = "Thin Crust Dough";

sauce = "Marinara Sauce";

toppings.add("Grated Reggiano Cheese");

}}

class NYStyleClamPizza extends Pizza

{public NYStyleClamPizza()

{

name = "NY Style Clam Pizza";

dough = "Thin Crust Dough";

sauce = "Marinara Sauce";

toppings.add("Grated Reggiano Cheese");

toppings.add("Fresh Clams from Long Island Sound");

}}

class NYStylePepperoniPizza extends Pizza

{public NYStylePepperoniPizza()

{

name = "NY Style Pepperoni Pizza";

dough = "Thin Crust Dough";

sauce = "Marinara Sauce";

toppings.add("Grated Reggiano Cheese");

toppings.add("Sliced Pepperoni");

toppings.add("Garlic");

toppings.add("Onion");

toppings.add("Mushrooms");

toppings.add("Red Pepper");

}}

class NYStyleVeggiePizza extends Pizza

{public NYStyleVeggiePizza()

{

name = "NY Style Veggie Pizza";

dough = "Thin Crust Dough";

sauce = "Marinara Sauce";

toppings.add("Grated Reggiano Cheese");

toppings.add("Garlic");

toppings.add("Onion");

toppings.add("Mushrooms");

toppings.add("Red Pepper");

}}

abstract class Pizza

{

String name;

String dough;

String sauce;

ArrayList toppings = new ArrayList();

void prepare()

{

System.out.println("Preparing " + name);

System.out.println("Tossing dough...");

System.out.println("Adding sauce...");

System.out.println("Adding toppings: ");

for (int i = 0; i < toppings.size(); i++)

{System.out.println(" " + toppings.get(i));

}}

void bake()

{System.out.println("Bake for 25 minutes at 350");

}

void cut()

{System.out.println("Cutting the pizza into diagonal slices");

}

void box()

{System.out.println("Place pizza in official PizzaStore box");

}

public String getName()

{return name;

}

public String toString()

{StringBuffer display = new StringBuffer();

display.append("---- " + name + " ----\n");

display.append(dough + "\n");

display.append(sauce + "\n");

for (int i = 0; i < toppings.size(); i++)

{display.append((String )toppings.get(i) + "\n");

}

return display.toString();

}}

abstract class PizzaStore

{abstract Pizza createPizza(String item);

public Pizza orderPizza(String type)

{Pizza pizza = createPizza(type);

System.out.println("--- Making a " + pizza.getName() + " ---");

pizza.prepare();

pizza.bake();

pizza.cut();

pizza.box();

return pizza;

}}

public class Main

{public static void main(String[] args)

{

PizzaStore nyStore = new NYPizzaStore();

PizzaStore chicagoStore = new ChicagoPizzaStore();

Pizza pizza = nyStore.orderPizza("cheese");

System.out.println("Ethan ordered a " + pizza.getName() + "\n");

pizza = chicagoStore.orderPizza("cheese");

System.out.println("Joel ordered a " + pizza.getName() + "\n");

pizza = nyStore.orderPizza("clam");

System.out.println("Ethan ordered a " + pizza.getName() + "\n");

pizza = chicagoStore.orderPizza("clam");

System.out.println("Joel ordered a " + pizza.getName() + "\n");

pizza = nyStore.orderPizza("pepperoni");

System.out.println("Ethan ordered a " + pizza.getName() + "\n");

pizza = chicagoStore.orderPizza("pepperoni");

System.out.println("Joel ordered a " + pizza.getName() + "\n");

pizza = nyStore.orderPizza("veggie");

System.out.println("Ethan ordered a " + pizza.getName() + "\n");

pizza = chicagoStore.orderPizza("veggie");

System.out.println("Joel ordered a " + pizza.getName() + "\n");

}}

2) Write a python program to implement Naive Bayes.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

data=pd.read\_csv('pima-indians-diabetes.csv')

data.shape

data.isnull().sum()

data.isnull().values.any()

data.dtypes

#visualisation

data.hist()

columns=list(data)[0:-1]

data[columns].hist()

#identifty the correlation

data.corr()

sns.heatmap(data.corr(),annot=True)

sns.pairplot(data)

#calculate diabetes ratio of true or false target varible

n\_true=len(data.loc[data['class']==True])

n\_false=len(data.loc[data['class']==False])

print("No.of true cases:{0} {1}%".format(n\_true,(n\_true/(n\_true+n\_false))\*100))

print("No.of false cases:{0} {1}%".format(n\_false,(n\_false/(n\_true+n\_false))\*100))

#split the data

from sklearn.model\_selection import train\_test\_split

x=data.drop('class',axis=1)

y=data['class']

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(x,y,test\_size=0.30,random\_state=1)

from sklearn.impute import SimpleImputer

rep\_0=SimpleImputer(missing\_values=0,strategy='mean')

cols=X\_train.columns

X\_train=pd.DataFrame(rep\_0.fit\_transform(X\_train))

X\_test=pd.DataFrame(rep\_0.fit\_transform(X\_test))

X\_train.columns=cols

X\_test.columns=cols

X\_train.head()

from sklearn.naive\_bayes import GaussianNB

diab\_mode=GaussianNB()

diab\_mode.fit(X\_train,Y\_train)

diab\_train\_predict=diab\_mode.predict(X\_train)

from sklearn import metrics

print("Model Accuracy:{0}".format(metrics.accuracy\_score(Y\_train,diab\_train\_predict)))

diab\_train\_predict=diab\_mode.predict(X\_test)

from sklearn import metrics

print("Model Accuracy:{0}".format(metrics.accuracy\_score(Y\_test,diab\_train\_predict)))

cm1=metrics.confusion\_matrix(Y\_test,diab\_train\_predict,labels=[1,0])

df\_cm1=pd.DataFrame(cm1,index=[i for i in['1','0']],columns=[i for i in['predict 1','predict o']] )

df\_cm1

3) Design a Django application that adds web pages with views and templates.

<html>

<head>

<script type="text/javascript">

function validate()

{

var regName=/^[a-zA-z]+[a-zA-Z]+$/;

var dateformatdob = /^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-9]|1[012])[\/\-]\d{4}$/;

var dateformatjdate = /^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-9]|1[012])[\/\-]\d{4}$/;

//Max six digits, a dot, max two digits after dot

var salaryformat=/^\d{1,6}(?:\.\d{0,2})?$/

var name=document.getElementById("txtname").value;

var dob=document.getElementById("txtdob").value;

var jdate=document.getElementById("txtjdate").value;

var salary=document.getElementById("txtsalary").value;

if(!regName.test(name))

alert("invalid name is given");

else

alert("valid name is given");

if(!dateformatjdate.test(jdate))

alert("invalid joining date is given");

else

alert("valid joining date is given");

if(!dateformatdob.test(dob))

alert("invalid date of birth is given");

else

alert("valid date of birth is given is given");

if(!salaryformat.test(salary))

alert("invalid salary");

else

alert("salary is valid");

}

</script>

</head>

<body>

<form>

<h1>Employee Rsgistration Details</h1>

enter employee first name

<input type="text" name="txtfname" id="txtname"><br>

enter date of birth

<input type="text" name="txtdob" id="txtdob"><br>

enter joining date

<input type="text" name="txtjdate" id="txtjdate"><br>

enter salary

<input type="text" name="txtsalary" id="txtsalary"><br>

<input type="button" value="validate" onclick="validate()">

</form>

</body>

</html>

**Slip-20**

Q1 Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters.

Lower.java

import java.io.\*;

import java.util.\*;

class LowerCaseInputStream extends FilterInputStream

{

public LowerCaseInputStream(InputStream in)

{

super(in);

}

public int read() throws IOException

{

int c=super.read();

return (c==-1?c:Character.toLowerCase((char)c));

}

public int read(byte[] b,int offset,int len) throws IOException

{

int result =super.read(b,offset,len);

for (int i=offset;i<offset+result;i++)

{

b[i]=(byte)Character.toLowerCase((char)b[i]);

}

return result;

}

}

class Lower

{

public static void main(String[] args) throws IOException

{

int c;

try

{

InputStream in =new LowerCaseInputStream(new BufferedInputStream(new FileInputStream("data.txt")));

while((c = in.read()) >= 0)

{

System.out.print((char)c);

}

in.close();

}

catch(IOException e)

{

e.printStackTrace();

}

}

}

2) Write a python program to implement Decision Tree whether or not to play Tennis.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from scipy.stats import zscore

import seaborn as sns

data=pd.read\_csv('play\_tennis.csv')

data.isnull().sum()

data.dtypes

data.head()

data.outlook.value\_counts()

from sklearn.preprocessing import LabelEncoder

l=LabelEncoder()

for i in data.columns:

if data[i].dtypes=='object' or data[i].dtypes=='bool':

data[i]=pd.Categorical(data[i])

for i in data.columns:

data[i]=l.fit\_transform(data[i])

data.dtypes

data.head()

x=data.drop(['play'],axis=1)

y=data['play']

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.30,random\_state=1)

dtree=DecisionTreeClassifier(criterion='gini',random\_state=1)

dtree.fit(x\_train,y\_train)

print(dtree.score(x\_train,y\_train)) #data is over fitted so we use max\_depth =5 means prunning technique

print(dtree.score(x\_test,y\_test))

dtree1=DecisionTreeClassifier(criterion='gini',max\_depth=5,random\_state=1)

dtree1.fit(x\_train,y\_train)

print(dtree1.score(x\_train,y\_train)) #data is over fitted so we use max\_depth =5 means prunning technique

print(dtree1.score(x\_test,y\_test))

y\_predict=dtree.predict(x\_test)

from sklearn import metrics

cm=metrics.confusion\_matrix(y\_test,y\_predict,labels=[1,0])

df\_cm=pd.DataFrame(cm,index=[i for i in['1','0']],columns=[i for i in['predicted 1','predicted 0']] )

df\_cm

sns.heatmap(df\_cm,annot=True)

from sklearn.metrics import classification\_report

m=classification\_report(y\_test,y\_predict)

print(m)

3) Develop a basic poll application (app).It should consist of two parts:

a) A public site in which user can pick their favourite programming language and vote.

b) An admin site that lets you add, change and delete programming languages.

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

</style>

</head>

<body>

<script>

function validateform(){

var email = document.getElementById("email").value;

var password = document.getElementById("psw").value;

if (/^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$/.test(email))

{

alert("Valid Email Id..")

return (true)

}

else{ alert("You have entered an invalid email address!")

return (false)

}

}

</script>

<form name="myform" onsubmit="return validateform()">

<div class="container">

<p>Please fill in this form to Login.</p>

<hr>

<label for="email"><b>Email</b></label>

<input type="text" autocomplete="off" placeholder="Enter Email" name="email" id="email" required>

<label for="psw"><b>Password</b></label>

<input type="password" autocomplete="off" placeholder="Enter Password" name="psw" id="psw" required>

<hr>

<button type="submit" class="registerbtn">Register</button>

</div>

</form>

</body>

</html>