**Slip 1**

* 1. Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters. [20 M]

import java.io.FilterInputStream;

import java.io.IOException;

import java.io.InputStream;

public class InputStreamDecorator extends FilterInputStream {

protected InputStreamDecorator(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 &lt; offset + result; i++) {

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

}

return result;

}

}

Client

import java.io.BufferedInputStream;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStream;

public class DecoratorClient {

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

int c;

try {

InputStream in = new InputStreamDecorator(new BufferedInputStream(

new FileInputStream(“decorator.txt”)));

while ((c = in.read()) &gt;= 0) {

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

}

in.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

* 1. Write a Python program to prepare Scatter Plot for Iris Dataset [20 M]

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

iris = load\_iris()

df= pd.DataFrame(data= np.c\_[iris['data'], iris['target']],

columns= iris['feature\_names'] + ['target'])

# select setosa and versicolor

y = df.iloc[0:100, 4].values

y = np.where(y == 'Iris-setosa', 0, 1)

# extract sepal length and petal length

X = df.iloc[0:100, [0, 2]].values

# plot data

plt.scatter(X[:50, 0], X[:50, 1],

color='blue', marker='o', label='Setosa')

plt.scatter(X[50:100, 0], X[50:100, 1],

color='green', marker='s', label='Versicolor')

plt.xlabel('Sepal length [cm]')

plt.ylabel('Petal length [cm]')

plt.legend(loc='upper left')

# plt.savefig('images/02\_06.png', dpi=300)

plt.show()

* 1. Create an HTML form that contain the Student Registration details and write a JavaScript to validate Student first and last name as it should not contain other than alphabets and age should be between 18 to 50.

[20 M]

Studentform.html

<html>

<head>

<title style="color: blue;">Student Form</title>

</head>

<body>

<div id="error"></div>

<form id="form1" onsubmit="validation()">

<table align="center" border="3" cellspacing ="10" style="color: blue;

bordercolor: crimson;">

<tr><td>Your First name: </td><td><input type="text" id="fname"

name="Fname"></td></tr><br>

<tr><td>Enter Last name: </td><td><input type ="text" id="lname"

name="lname"></td></tr><br>

<tr><td>Enter age: </td><td><input type ="text" id="age"

name="age"></td></tr><br>

<tr><td>Enter mobile: </td><td><input type ="text" id="mobile"

name="mobile"></td></tr><br>

<tr><td>Enter Address : </td><td><input type ="text" id="address"

name="address"></td></tr><br>

<tr><td>Select Subject :</td><td><Select type="text" name="select" value="-

1">

<option >select subject</option>

<option name ="BSC">BSC</option>

<option name ="BSC">BSC(CS)</option>

<option name ="BSC">BSC(CA)</option>

</Select></td></tr>

<tr><td style="text-align: center;">

<input type = "submit" value="Register"></td></tr>

</table>

</form>

<script src="validateJS.js" type="text/javascript">

</script>

</body>

</html>

ValidateJS.js

function validation(){ const

fname=document.getElementById("fname") const

lname=document.getElementById("lname") const form

= document.getElementById("form1") const age =

document.getElementById("age") const error =

document.getElementById("error") const mobile =

document.getElementById("mobile") const address =

document.getElementById("address") const pattern

= /^[A-Z a-z]+$/; const mpatrn = /^([9]{1})-([0-

9]{9})$/ const addpatrn = /^[A-Z a-z 0-9]+$/ if(!

pattern.test(fname.value))

{

alert("first name should contain alphabate only!!")

return false

}if(!pattern.test(lname.value))

{

alert("Last name should contain alphabates only")

return false

}if(age.value <= 18 || age.value > 50)

{

alert("Age should be between 18 to 50 ")

return false

}if(mobile.value.length != 10)

{

alert("Mobile number should be of ten numbers")

return false

}if(!addpatrn.test(address.value))

{

alert("Address does not contains special

character")

return false

}}

* 1. Viva [10 M]

**Slip 2**

* 1. Write a Java Program to implement Singleton pattern for multithreading

Step 1:

public class SingletonDesignPatternWithEagerInitialization {

// Step 1: private static variable of INSTANCE variable

private static SingletonDesignPatternWithEagerInitialization

INSTANCE = new SingletonDesignPatternWithEagerInitialization();

// Step 2: private constructor

private SingletonDesignPatternWithEagerInitialization() {

}

// Step 3: Provide public static getInstance() method

// returning same INSTANCE same time

public static SingletonDesignPatternWithEagerInitialization

getInstance() {

return INSTANCE;

}

}

Step 2:

public class SingletonDesignPatternWithLazyInitialization {

// Step 1: private static variable of INSTANCE variable

private static SingletonDesignPatternWithLazyInitialization

INSTANCE;

// Step 2: private constructor

private SingletonDesignPatternWithLazyInitialization() {

}

// Step 3: Provide public static getInstance() method

// returning INSTANCE after checking

public static SingletonDesignPatternWithLazyInitialization

getInstance() {

if(null == INSTANCE){

INSTANCE = new

SingletonDesignPatternWithLazyInitialization();

}

return INSTANCE;

}

}

Step 3:

public class SingletonDesignPatternInMultiThreadedEnvironment {

// Step 1: private static variable of INSTANCE variable

private static volatile

SingletonDesignPatternInMultiThreadedEnvironment INSTANCE;

// Step 2: private constructor

private SingletonDesignPatternInMultiThreadedEnvironment() {

}

// Step 3: Provide public static getInstance() method

// returning INSTANCE after checking

public static SingletonDesignPatternInMultiThreadedEnvironment

getInstance() {

// synchronized block

synchronized

(SingletonDesignPatternInMultiThreadedEnvironment.class){

if(null == INSTANCE){

INSTANCE =new

SingletonDesignPatternInMultiThreadedEnvironment();

}

return INSTANCE;

}

}

}

Step 4:

package in.bench.resources.singleton.design.pattern;

public class SingletonDesignPatternWithDCL {

// Step 1: private static variable of INSTANCE variable

private static volatile SingletonDesignPatternWithDCL

INSTANCE;

// Step 2: private constructor

private SingletonDesignPatternWithDCL() {

}

// Step 3: Provide public static getInstance() method

// returning INSTANCE after checking

public static SingletonDesignPatternWithDCL getInstance() {

// double-checking lock

if(null == INSTANCE){

// synchronized block

synchronized (SingletonDesignPatternWithDCL.class) {

if(null == INSTANCE){

INSTANCE = new SingletonDesignPatternWithDCL();

}}}

return INSTANCE;

}}

* 1. Write a python program to find all null values in a given dataset and remove them. [20 M]
  2. Create an HTML form that contain the Employee Registration details and write a JavaScript to validate DOB, Joining Date, and Salary. [20 M]

Employeeform.html

<html>

<head>

</head>

<center><h2>Employee Registration Form</h2></center>

<body>

<div id="error"></div>

<form id="form1" onsubmit="validation()">

<table align="center" border="3" cellspacing ="10">

<tr><td>Your First name: </td><td><input type="text" id="fname"

name="Fname"></td></tr><br>

<tr><td>Enter Last name: </td><td><input type ="text" id="lname"

name="lname"></td></tr><br>

<tr><td>Enter age: </td><td><input type ="text" id="age"

name="age"></td></tr><br>

<tr><td>Enter mobile: </td><td><input type ="text" id="mobile"

name="mobile"></td></tr><br>

<tr><td>Enter Address : </td><td><input type ="text" id="address"

name="address"></td></tr><br>

<tr><td>Select Designation :</td><td><Select type="text" id="desig"

name="designation" >

<option value="null" >select designation</option>

<option value="Employee">Employee</option>

<option value ="Employee ">Fresher</option>

<option value ="Employee ">Manager</option>

<option value ="Employee ">Assistant</option>

<option value ="Employee ">Technical support</option>

<option value ="Employee ">Accountant</option>

</Select></td></tr>

<tr><td>Date OF Birth(DOB) </td><td><input type ="text" id="dob"

name="dob"></td></tr><br>

<tr><td>Date OF Joining </td><td><input type ="text" id="doj"

name="doj"></td></tr><br>

<tr><td>Salary </td><td><input type ="text" id="sal" name="sal"></td></tr><br>

<tr><td><input type = "submit" value="Register"></td></tr>

</table>

</form>

<script src="Assignment2.js" type="text/javascript"></script></body></html>

Assignment2.js

function validation(){

const fname=document.getElementById("fname")

const lname=document.getElementById("lname")

const form = document.getElementById("form1")

const age = document.getElementById("age") const

error = document.getElementById("error") const

mobile = document.getElementById("mobile") const

address = document.getElementById("address")

const dob = document.getElementById("dob") const

desig = document.getElementById("desig") const

dojoin = document.getElementById("doj") const sal

= document.getElementById("sal")

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

dobPattern=/(((0|1)[0-9]|2[0-9]|3[0-1])\/(0[1-9]|1[0-

2])\/((19|20)\d\d))$/; const pattern

= /^[A-Z a-z]+$/; const mpatrn

= /^([9]{1})-([0-9]{9})$/ const

addpatrn = /^[A-Z a-z 0-9]+$/

if(!

pattern.test(fname.value))

{

alert("first name should contain alphabate only or it can't be null")

return false

}

if(!pattern.test(lname.value))

{

alert("Last name should contain alphabates only")

return false

} if(age.value <= 18 || age.value >

50)

{

alert("Age should be between 18 to 50

")

return false

} if(mobile.value.length !=

10)

{

alert("Mobile number should be of ten

numbers")

return false

} if(!

addpatrn.test(address.value))

{

alert("Address does not contains special

character")

return false

} if(!

dobPattern.test(dob.value))

{

alert("Enter birth date in [dd/mm/yyyy]

format")

return false

} if(desig.value ==

"")

{

alert("Select your

designation")

return false

} if(!

dobPattern.test(dojoin.value))

{

alert("Enter join date in [dd/mm/yyyy] format

")

return false

} if(!

salpattern.test(sal.value))

{

alert("Something is wrong while entering salary!")

return false

}

}

Q.4 Viva [10 M]

**Slip 3**

Q.1 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() [20 M]

Step-1

public interface Observer {

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

}

Another interface

public interface DisplayElement {

public void display();

}

Step-2

public interface Subject {

public void registerObserver(Observer o);

public void removeObserver(Observer o);

public void notifyObservers();

}

Step-3

import java.util.\*;

public class WeatherData implements Subject {

private ArrayList<Observer> observers;

private float temperature;

private float humidity;

private float pressure;

public WeatherData() {

observers = new ArrayList<Observer>();

}

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;

}}

Step-4

public 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, rainy weather");

}

}

}

Step-5

public 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);

}

}

Step-6

public 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("Avg/Max/Min temperature = " + (tempSum / numReadings)

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

}

}

Step 7: Create fourth Observer CurrentConditionsDisplay class.

public 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");

}

}

Step 8: Create WeatherStation class to test observer design pattern.

public class WeatherStation {

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(80, 65, 30.4f);

weatherData.setMeasurements(82, 70, 29.2f);

weatherData.setMeasurements(78, 90, 29.2f);

} }

Q. 2. Write a python program to make Categorical values in numeric format for a given dataset [20 M]

import pandas as pd

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

df

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

label=le.fit\_transform(df['Purchased'])

label

df.drop("Purchased", axis=1, inplace=True)

df["Purchased"] = label

df

Q. 3 Create an HTML form for Login and write a JavaScript to validate email ID using Regular Expression. [20 M]

Loginform.html

<!DOCTYPE html>

<html>

<head>

</head>

<center><h2>Login Form</h2></center>

<body>

<div id="error1"></div>

<form id="form1" onsubmit="validation()">

<table align="center" border="3" cellspacing="10">

<tr><td>Your First name: </td><td><input type="text" id="fname"

name="Fname"></td></tr>

<br>

<tr><td>Enter Last name: </td><td><input type="text" id="lname"

name="lname"></td></tr>

<br>

<tr><td>Enter mobile: </td><td><input type="text" id="mobile"

name="mobile"></td></tr>

<br>

<tr><td>Enter Address : </td><td><input type="text" id="address"

name="address"></td></tr>

<br>

<tr><td>Enter email\_id : </td><td><input type="text" id="email"

name="email" /></td></tr

<br />

<tr><td>Date OF Birth(DOB) </td><td><input type="text" id="dob"

name="dob"></td></tr>

<br>

<tr><td><input type="submit" value="Login"></td></tr>

</table>

</form>

<script src="Valid.js" type="text/javascript">

</script>

</body>

</html>

Valid.js

function validation() {

const fname = document.getElementById("fname")

const lname = document.getElementById("lname")

const form =

document.getElementById("form1")

const error1 =

document.getElementById("error1")

const mobile =

document.getElementById("mobile")

const address =

document.getElementById("address")

const email =

document.getElementById("email")

const pattern = /^[A-Z a-z]+$/;

const mpatrn = /^([9]{1})-([0-9]{9})$/ const addpatrn = /^[A-Z a-z 0-

9]+$/

const emailpattern = /^[a-zA-Z0-9.\_-]+@ [ a-zA-Z0-9.-]+\.[a-zA-Z]

{2,4}$/

if (!pattern.test(fname.value))

{

alert("first name should contain alphabate

only!!")

return false

}

if (!

pattern.test(lname.value))

{

only")

}

!= 10)

{

numbers")

}

alert("Last name should contain alphabates

return false

if (mobile.value.length

alert("Mobile number should be of ten

return false

if (!

addpatrn.test(address.value))

{

alert("Address does not contains special

character")

return false

}

if (!emailpattern.test(email.value))

{

alert("Enter the correct email address")

return false

}

}

Q.4 Viva [10 M]

**Slip 4**

Q.1 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. [20 M]

Step 1 :-

import java.util.ArrayList;

abstract public class Pizza {

String name;

String dough;

String sauce;

ArrayList toppings = new ArrayList();

public String getName() {

return name;

}

public void prepare() {

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

}

public void bake() {

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

}public void cut() {

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

}

public void box() {

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

}

public String toString() {

// code to display pizza name and ingredients

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();

}

}

Step2:-

public class CheesePizza extends Pizza {

public CheesePizza() {

name = "Cheese Pizza";

dough = "Regular Crust";

sauce = "Marinara Pizza Sauce";

toppings.add("Fresh Mozzarella");

toppings.add("Parmesan");

}

}public class ClamPizza extends Pizza{

public ClamPizza() {

name="Clam Pizza";

dough="Thin crust";

sauce = "White garlic sauce";

toppings.add("Clams");

toppings.add("Grated parmesan cheese");

}}

public class PepperoniPizza extends Pizza {

public PepperoniPizza() {

name = "Pepperoni Pizza";

dough = "Crust";

sauce = "Marinara sauce";

toppings.add("Sliced Pepperoni");

toppings.add("Sliced Onion");

toppings.add("Grated parmesan cheese");

}}

public class VeggiePizza extends Pizza {

public VeggiePizza() {

name = "Veggie Pizza";

dough = "Crust";

sauce = "Marinara sauce";

toppings.add("Shredded mozzarella");

toppings.add("Grated parmesan");

toppings.add("Diced onion");

toppings.add("Sliced mushrooms");

toppings.add("Sliced red pepper");

toppings.add("Sliced black olives");

}

}

Step 3:-

public class SimplePizzaFactory {

public Pizza createPizza(String type) {

Pizza pizza = null;

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

pizza = new CheesePizza();

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

pizza = new PepperoniPizza();

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

pizza = new ClamPizza();

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

pizza = new VeggiePizza();

}

return pizza;

}

}

Step 4:-

public class PizzaStore {

SimplePizzaFactory factory;

public PizzaStore(SimplePizzaFactory factory) {

this.factory = factory;

}

public Pizza orderPizza(String type) {

Pizza pizza;

pizza = factory.createPizza(type);

pizza.prepare();

pizza.bake();

pizza.cut();

pizza.box();return pizza;

}}

Step 5:-

public class PizzaTestDrive {

public static void main(String[] args) {

SimplePizzaFactory factory = new SimplePizzaFactory();

PizzaStore store = new PizzaStore(factory);

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

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

pizza = store.orderPizza("veggie");

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

}

}

Q. 2 Write a python program to Implement Simple Linear Regression for predicting house price. [20 M]

# Simple Linear Regression

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('Salary\_Data.csv')

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

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

# Splitting the dataset into the Training set and Test set

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 1/3, random\_state = 0)

# Training the Simple Linear Regression model on the Training set

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

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

# Predicting the Test set results

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

# Visualising the Training set results

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

plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')

plt.title('Salary vs Experience (Training set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()

# Visualising the Test set results

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

plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')

plt.title('Salary vs Experience (Test set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()

Q. 3 Create a Node.js file that will convert the output "Hello World!" into upper-case letters. [20 M]

File.js

var http = require('http'); var uc = require('upper-case');

http.createServer(function (req, res) {

res.writeHead(200, {'Content-Type':

'text/html'});

/\*Use our upper-case module to upper case a string:\*/

res.write(uc.upperCase("Hello World!")); res.end();

}).listen(8080);

Q.4 Viva [10 M]

**Slip 5**

* 1. Write a Java Program to implement Adapter pattern for Enumeration iterator [20 M]

import java.util.\*;

public class EnumerationIterator implements Iterator {

Enumeration enumeration;

public EnumerationIterator(Enumeration enumeration) {

this.enumeration = enumeration;

}

public boolean hasNext() {

return enumeration.hasMoreElements();

}

public Object next() {

return enumeration.nextElement();

}

public void remove() {

throw new UnsupportedOperationException();

}

}

Test Drive the Enumeration Adapter

import java.util.\*;

public class EnumerationIteratorTestDrive {

public static void main (String args[]) {

Vector v = new Vector(Arrays.asList(args));

Iterator iterator = new

EnumerationIterator(v.elements());

while (iterator.hasNext()) {

System.out.println(iterator.next());

}

}

}

* 1. Write a python program to implement Multiple Linear Regression for given dataset.

[20 M]

# Multiple Linear Regression

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

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

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

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

print(X)

# Encoding categorical data

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)

# Splitting the dataset into the Training set and Test set

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, random\_state = 0)

# Training the Multiple Linear Regression model on the Training set

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

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

# Predicting the Test set results

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

np.set\_printoptions(precision=2)

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

Q. 3 Using nodejs create a web page to read two file names from user and append contents of first file into second file. [20 M]

var fs = require('fs');

var file1 ='input.txt';

var file2 = 'output.txt';

function

fileValidation()

{

fs.open(file1,'r',function(err,fd){

if(err)

{

return console.error(err);

}

var buffer = new Buffer.alloc(30);

fs.read(fd,buffer,0,buffer.length,0,function(err,bytes){

if(err) throw err;

console.log(buffer.toString());

});

fs.appendFile(file2,buffer,function(err){

if(err) throw err;

console.log("data

saved !!!");

fs.close(fd,function(err)

{

if(err)

throw err;});

});

});

}

fileValidation();

A screenshot of a computer program

Description automatically generated

Q.4 Viva [10 M]

**Slip 6**

* 1. Write a Java Program to implement command pattern to test Remote Control

[20 M]

Step 1:

public class Car {

public void move()

{

System.out.println("Car is moving");

}

public void stop()

{

System.out.println("Car has stopped");

}

}

Step 2

public class RotatingTop {

public void startRotating(){

System.out.println("Top has start rotating");

}

public void stopRotating(){

System.out.println("Top has stopped rotating");

}

}

Step 3:

public interface CommandBase {

void execute();

void undo();

}

Step 4:

import guru.springframework.gof.command.receiver.Car;

2.3. public class CarMoveCommand implements CommandBase {

private Car car;

public CarMoveCommand(Car car){

this.car=car;

}

@Override

public void execute(){

System.out.println("CarMoveCommand.execute(): Invoking move() on

Car");

car.move();

}

@Override

public void undo(){

System.out.println("CarMoveCommand.undo(): Undoing previous action-

>Invoking stop() on Car");

car.stop();

}

}

Step5

CarStopCommand.java

package guru.springframework.gof.command.commandobjects;

import guru.springframework.gof.command.receiver.Car;

public class CarStopCommand implements CommandBase{

private Car car;

public CarStopCommand(Car car){

this.car=car;

}

@Override

public void execute(){

System.out.println("CarStopCommand.execute(): Invoking stop() on Car");14. car.stop();

}

@Override

public void undo()

{

System.out.println("CarStopCommand.undo(): Undoing previous action->

Invoking move() on Car");

car.move();

}

}

Step 6

TopRotateCommand.java

package guru.springframework.gof.command.commandobjects;

import guru.springframework.gof.command.receiver.RotatingTop;

public class TopRotateCommand implements CommandBase{

RotatingTop rotatingTop;

public TopRotateCommand(RotatingTop rotatingTop){

this.rotatingTop=rotatingTop;

}

@Override

public void execute(){

System.out.println("TopRotateCommand.execute(): Invoking startRotating() on

RotatingTop");

rotatingTop.startRotating();

}

@Override

public void undo(){

System.out.println("TopRotateCommand.undo(): Undoing previous action-

>Invoking stopRotating() on

RotatingTop");

rotatingTop.stopRotating();20. }

}

Step 7

TopStopRotateCommand.java

package guru.springframework.gof.command.commandobjects;

import guru.springframework.gof.command.receiver.RotatingTop;

public class TopStopRotateCommand implements CommandBase{

RotatingTop rotatingTop;

public TopStopRotateCommand(RotatingTop rotatingTop){

this.rotatingTop=rotatingTop;

}

@Override

public void execute(){

System.out.println("TopStopRotateCommand.execute(): Invoking stopRotating() on

RotatingTop");

rotatingTop.stopRotating();

}

@Override

public void undo(){

System.out.println("TopStopRotateCommand.undo(): Undoing previous action-

>Invoking startRotating() on

RotatingTop");

rotatingTop.startRotating();

}

}

Step 8:

RemoteControl.java

package guru.springframework.gof.command.invoker;

import guru.springframework.gof.command.commandobjects.CommandBase;

public class RemoteControl {7. CommandBase onCommand, offCommand, undoCommand;

public void onButtonPressed(CommandBase onCommand){

this.onCommand=onCommand;

onCommand.execute();

undoCommand=onCommand;

}

public void offButtonPressed(CommandBase offCommand){

this.offCommand=offCommand;

offCommand.execute();

undoCommand=offCommand;

}

public void undoButtonPressed(){

undoCommand.undo();

}

}

Step 9

RemoteControlTest.java

package guru.springframework.gof.command.invoker;

import guru.springframework.gof.command.commandobjects.\*;

import guru.springframework.gof.command.receiver.Car;

import guru.springframework.gof.command.receiver.RotatingTop;

import org.junit.Test;

import static org.junit.Assert.\*;

public class RemoteControlTest {

@Test

public void testRemoteControlButtonPressed() throws Exception {

RemoteControl remoteControl=new RemoteControl();

System.out.println("-----Testing onButtonPressed on RemoteControl for Car-----

");

Car car=new Car();

CommandBase carMoveCommand=new CarMoveCommand(car);

remoteControl.onButtonPressed(carMoveCommand);

System.out.println("-----Testing offButtonPressed on RemoteControl for Car-----

");

CommandBase carStopCommand=new CarStopCommand(car);

remoteControl.offButtonPressed(carStopCommand);

System.out.println("-----Testing undoButtonPressed() on RemoteControl for Car----

-");

remoteControl.undoButtonPressed();

System.out.println("-----Testing onButtonPressed on RemoteControl for

RotatingTop-----");

RotatingTop top=new RotatingTop();

CommandBase topRotateCommand=new TopRotateCommand(top);

remoteControl.onButtonPressed(topRotateCommand);

System.out.println("-----Testing offButtonPressed on RemoteControl for

RotatingTop-----");

CommandBase topStopRotateCommand=new TopStopRotateCommand(top);

remoteControl.offButtonPressed(topStopRotateCommand);

System.out.println("-----Testing undoButtonPressed on RemoteControl for

RotatingTop-----");

remoteControl.undoButtonPressed();

}

}

* 1. Write a python program to implement Polynomial Linear Regression for given dataset

[20 M]

# Polynomial Regression

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

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

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

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

# Training the Linear Regression model on the whole dataset

from sklearn.linear\_model import LinearRegression

lin\_reg = LinearRegression()

lin\_reg.fit(X, y)

# Training the Polynomial Regression model on the whole dataset

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree = 4)

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

lin\_reg\_2 = LinearRegression()

lin\_reg\_2.fit(X\_poly, y)

# Visualising the Linear Regression results

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

plt.plot(X, lin\_reg.predict(X), color = 'blue')

plt.title('Truth or Bluff (Linear Regression)')

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()

# Visualising the Polynomial Regression results

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

plt.plot(X, lin\_reg\_2.predict(poly\_reg.fit\_transform(X)), color = 'blue')

plt.title('Truth or Bluff (Polynomial Regression)')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

# Visualising the Polynomial Regression results (for higher resolution and smoother curve)

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.plot(X\_grid, lin\_reg\_2.predict(poly\_reg.fit\_transform(X\_grid)), color = 'blue')

plt.title('Truth or Bluff (Polynomial Regression)')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()# Predicting a new result with Linear Regression

lin\_reg.predict([[6.5]])

# Predicting a new result with Polynomial Regression

lin\_reg\_2.predict(poly\_reg.fit\_transform([[6.5]]))

Q.3. Create a Node.js file that opens the requested file and returns the content to the client. If anything goes wrong, throw a 404 error. [20 M]

var http = require('http'); var url = require('url'); var fs =

require('fs'); http.createServer(function (req, res) {

var

pathname = url.parse(req.url, true).pathname;

console.log("Request for" + pathname + "received.");

fs.readFile(pathname.substr(1), function (err, data) {

if (err) {

console.log(err);

res.writeHead(404, { 'content-type': 'text/html' });

res.end('<html><body><h1>404 Not found</h1></body></html>');

}

else {

'content-type': 'text/html' });

res.writeHead(200, {

res.write(data);

res.end();

}

});

}).listen(9030); console.log('server is

running on port 8080');

index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>Sample Page</title>

</head>

<body>

Hello World! Welcome to web module.

</body>

</html>

Q.4 Viva [10 M]

**Slip 7**

|  |  |  |
| --- | --- | --- |
| Q.1 | Write a Java Program to implement undo command to test Ceiling fan. | [20 M] |

public interface Command

{ public void execute();

public void undo();

}

public class LightOnCommand implements Command

{ Light light; public LightOnCommand(Light light)

{ this.light = light;

} public void execute()

{ light.on(); } public void undo() { light.off();

}}

public class LightOffCommand implements Command

{ Light light; public LightOffCommand(Light light)

{ this.light = light; } public void execute()

{ light.off(); } public void undo() { light.on();

}}

public class RemoteControl

{ Command[] onCommands; Command[] offCommands; public RemoteControl()

{ 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; }

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

{ onCommands[slot] = onCommand; offCommands[slot] = offCommand;

} public void onButtonWasPushed(int slot)

{ onCommands[slot].execute(); } public void offButtonWasPushed(int slot) {

offCommands[slot].execute();} 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”);

}

return stringBuff.toString();

}}

|  |  |  |
| --- | --- | --- |
| Q.2. | Write a python program to implement Naive Bayes. | [20 M] |

# Naive Bayes

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

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

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

# Splitting the dataset into the Training set and Test set

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)

print(X\_train)

print(y\_train)

print(X\_test)

print(y\_test)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

print(X\_train)

print(X\_test)

# Training the Naive Bayes model on the Training set

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

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

# Predicting a new result

print(classifier.predict(sc.transform([[30,87000]])))

# Predicting the Test set results

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

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

# Visualising the Training set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = sc.inverse\_transform(X\_train), y\_trainX1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = X\_set[:, 0].max() + 10,

step = 0.25),

np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step

= 0.25))

plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(),

X2.ravel()]).T)).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'green'))(i),

label = j)

plt.title('Naive Bayes (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = sc.inverse\_transform(X\_test), y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = X\_set[:, 0].max() + 10,

step = 0.25),

np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step

= 0.25))

plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(),

X2.ravel()]).T)).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'green'))(i),

label = j)

plt.title('Naive Bayes (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

|  |  |  |
| --- | --- | --- |
| Q. 3 | Create a Node.js file that writes an HTML form, with an upload field. | [20 M] |

File.js

var http = require('http');

http.createServer(function (req, res) {

res.writeHead(200, {'Content-Type':

'text/html'});

res.write('<form action="fileupload" method="post"

enctype="multipart/formdata">');

res.write('<input type="file"

name="filetoupload"><br>');

res.write('<input type="submit">');

res.write('</form>');

return res.end();

}).listen(8080);

Q.4 Viva [10 M]

**Slip 8**

|  |  |  |
| --- | --- | --- |
| Q. 1 | Write a Java Program to implement State Pattern for Gumball Machine. |  |

|  |  |  |
| --- | --- | --- |
| Q.2. | Write a python program to implement Decision Tree whether or not to play Tennis. |  |

# Decision Tree Classification

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

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

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

# Splitting the dataset into the Training set and Test set

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)

print(X\_train)

print(y\_train)

print(X\_test)

print(y\_test)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

print(X\_train)

print(X\_test)

# Training the Decision Tree Classification model on the Training set

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion = 'entropy', random\_state = 0)

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

# Predicting a new result

print(classifier.predict(sc.transform([[30,87000]])))

# Predicting the Test set results

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

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

# Visualising the Training set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = sc.inverse\_transform(X\_train), y\_trainX1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = X\_set[:, 0].max() + 10,

step = 0.25),

np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step

= 0.25))

plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(),

X2.ravel()]).T)).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'green'))(i),

label = j)

plt.title('Decision Tree Classification (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = sc.inverse\_transform(X\_test), y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = X\_set[:, 0].max() + 10,

step = 0.25),

np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step

= 0.25))

plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(),

X2.ravel()]).T)).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'green'))(i),

label = j)

plt.title('Decision Tree Classification (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

|  |  |  |
| --- | --- | --- |
| Q. 3 | Create a Node.js file that demonstrates create database and table in MySQL. [20 M] |  |

Createdatabase.js

var mysql= require('mysql'); var

con= mysql.createConnection({

host: "localhost",

user:

"root",

password: "satara@123"

}); con.connect(function(err){

if (err) throw err;

console.log("Connected!!");

con.query("create

DATABASE mydb",function(err,result){

if (err)

throw err;

console.log("Database created");

});

});

A screenshot of a computer

Description automatically generated

Q.4 Viva [10 M]

**Slip 9**

|  |  |  |
| --- | --- | --- |
| Q.1 | Design simple HR Application using Spring Framework |  |

|  |  |  |
| --- | --- | --- |
| Q 2. | Write a python program to implement Linear SVM. |  |

# Support Vector Machine (SVM)

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

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

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

# Splitting the dataset into the Training set and Test set

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)

print(X\_train)

print(y\_train)

print(X\_test)

print(y\_test)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

print(X\_train)

print(X\_test)

# Training the SVM model on the Training set

from sklearn.svm import SVC

classifier = SVC(kernel = 'linear', random\_state = 0)

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

# Predicting a new result

print(classifier.predict(sc.transform([[30,87000]])))

# Predicting the Test set results

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

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

# Visualising the Training set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = sc.inverse\_transform(X\_train), y\_trainX1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = X\_set[:, 0].max() + 10,

step = 0.25),

np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step

= 0.25))

plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(),

X2.ravel()]).T)).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'green'))(i),

label = j)

plt.title('SVM (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = sc.inverse\_transform(X\_test), y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = X\_set[:, 0].max() + 10,

step = 0.25),

np.arange(start = X\_set[:, 1].min() - 1000, stop = X\_set[:, 1].max() + 1000, step

= 0.25))

plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(),

X2.ravel()]).T)).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'green'))(i),

label = j)

plt.title('SVM (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

|  |  |  |
| --- | --- | --- |
| Q. 3 | Create a node.js file that Select all records from the "customers" table, and display the result object on console. |  |

Db.js

var mysql= require('mysql'); var

con= mysql.createConnection({

host:"localhost",

user:"root",

password:"satara@123",

database: "mydb"

});

con.connect(function(err){

if (err) throw err;

con.query("select name , address

from

Customer",function(err,result,fields){

if (err) throw err;

console.log(result);

});

});

A computer screen with white text

Description automatically generated

Q.4 Viva [10 M]

**Slip 10**

|  |  |  |
| --- | --- | --- |
| Q.1 | Write a Java Program to implement Strategy Pattern for Duck Behavior. Create instance variable that holds current state of Duck from there, we just need to handle all Flying Behaviors and Quack Behavior |  |

|  |  |  |
| --- | --- | --- |
| Q. 2 | Write a Python program to prepare Scatter Plot for Iris Dataset. |  |

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

iris = load\_iris()

df= pd.DataFrame(data= np.c\_[iris['data'], iris['target']],

columns= iris['feature\_names'] + ['target'])

# select setosa and versicolor

y = df.iloc[0:100, 4].values

y = np.where(y == 'Iris-setosa', 0, 1)

# extract sepal length and petal length

X = df.iloc[0:100, [0, 2]].values

# plot data

plt.scatter(X[:50, 0], X[:50, 1],

color='blue', marker='o', label='Setosa')

plt.scatter(X[50:100, 0], X[50:100, 1],

color='green', marker='s', label='Versicolor')

plt.xlabel('Sepal length [cm]')

plt.ylabel('Petal length [cm]')

plt.legend(loc='upper left')

# plt.savefig('images/02\_06.png', dpi=300)

plt.show()

|  |  |  |
| --- | --- | --- |
| Q. 3 | Create a node.js file that Insert Multiple Records in "student" table, and display the result object on console. [20 M] |  |

Student.js

var mysql=require('mysql');

var

con=mysql.createConnection({

host:"localhost",

user:"root",

password:"satara@123",

database:"mydb"

}); con.connect(function(err){

if (err) throw err;

console.log("Connected!!!");

var sql="INSERT INTO

Student (name,address) Values ?";

var values=[

['Pritam','Highway 71'],

['Sneha','Lowstreet 4'],

['Sid','Apple st 652'],

['Sam','Valley 345'],

['Michael','Green Greass 1'],

['Griss','One way 98'],

['Richard','Sky st 331']

];

con.query(sql,

[values],function(err,result){

if (err)

throw err;

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

});

});

A screenshot of a computer

Description automatically generated

Q.4 Viva [10 M]