Slip 1:

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

```
import java.io.*;

class LowercaseDecorator extends FilterReader {

    public LowercaseDecorator(Reader in) {
        super(in);
    }

    @Override
    public int read() throws IOException {
        int c = super.read();
        if (c != -1) {
            return Character.toLowerCase((char) c);
        }
        return -1;
    }

    @Override
    public int read(char[] cbuf, int off, int len) throws IOException {
        int bytesRead = super.read(cbuf, off, len);
        if (bytesRead != -1) {
            for (int i = off; i < off + bytesRead; i++) {
                cbuf[i] = Character.toLowerCase(cbuf[i]);
            }
        }
        return bytesRead;
    }
}

public class IODecoratorExample {
    public static void main(String[] args) {
        try {
            // Create a FileReader for the input file
            FileReader fileReader = new FileReader("input.txt");

            // Wrap it with LowercaseDecorator
```

```
            LowercaseDecorator lowercaseDecorator = new
LowercaseDecorator(fileReader);

            // Create a BufferedReader for reading lines
            BufferedReader bufferedReader = new
BufferedReader(lowercaseDecorator);

            // Read and print lines
            String line;
            while ((line = bufferedReader.readLine()) != null) {
                System.out.println(line);
            }

            // Close readers
            bufferedReader.close();
            fileReader.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Q2) iris

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("Iris.csv")
print (data.head(10))
x=data["sepal_length"]
y=data["petal_length"]
plt.scatter(x,y)
plt.show()
```

Q3) HTML FORM

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Registration Form</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        .error {
            color: red;
        }
    </style>
</head>
<body>

<h2>Student Registration Form</h2>

<form id="registrationForm" onsubmit="return validateForm()">
    <label for="firstName">First Name:</label>
    <input type="text" id="firstName" name="firstName" required>
    <span id="firstNameError" class="error"></span>

    <br>

    <label for="lastName">Last Name:</label>
    <input type="text" id="lastName" name="lastName" required>
    <span id="lastNameError" class="error"></span>

    <br>

    <label for="age">Age:</label>
    <input type="number" id="age" name="age" required>
    <span id="ageError" class="error"></span>

    <br>

    <input type="submit" value="Register">
</form>

<script>
```

```
    function validateForm() {
        var firstName = document.getElementById('firstName').value;
        var lastName = document.getElementById('lastName').value;
        var age = document.getElementById('age').value;

        // Regular expression to check if the name contains only alphabets
        var nameRegex = /^[a-zA-Z]+$/;

        // Validate First Name
        if (!nameRegex.test(firstName)) {
            document.getElementById('firstNameError').innerHTML = 'First
name should contain only alphabets.';
            return false;
        } else {
            document.getElementById('firstNameError').innerHTML = '';
        }

        // Validate Last Name
        if (!nameRegex.test(lastName)) {
            document.getElementById('lastNameError').innerHTML = 'Last name
should contain only alphabets.';
            return false;
        } else {
            document.getElementById('lastNameError').innerHTML = '';
        }

        // Validate Age
        if (age < 18 || age > 50 || isNaN(age)) {
            document.getElementById('ageError').innerHTML = 'Age should be
between 18 and 50.';
            return false;
        } else {
            document.getElementById('ageError').innerHTML = '';
        }

        // If all validations pass, the form is submitted
        return true;
    }
</script>
```

```
</body>
</html>
```

Slip 11:

Q1 Heart beat

```
// Existing BeatModel
interface BeatModel {
    void beat();
}

// HeartModel (Adapter) implementing BeatModel
class HeartModelAdapter implements BeatModel {
    private HeartModel heartModel;

    public HeartModelAdapter(HeartModel heartModel) {
        this.heartModel = heartModel;
    }

    @Override
    public void beat() {
        heartModel.heartbeat();
    }
}

// Existing HeartModel
class HeartModel {
    void heartbeat() {
        System.out.println("Heart is beating!");
    }
}

// Client code using BeatModel
class Client {
    public static void main(String[] args) {
        // Use the existing HeartModel with the help of the adapter
        HeartModel heartModel = new HeartModel();
```

```
        BeatModel adapter = new HeartModelAdapter(heartModel);

        // Use the adapted interface
        adapter.beat();
    }
}
```

Q2) dataset null remove

```
import pandas
# reading the CSV file
csvFile = pandas.read_csv('employees.csv')
# displaying the contents of the CSV file
print(csvFile)
count=csvFile.isnull()
#displaying NULL content
print(count)
newdf = csvFile.dropna()
print(newdf)
```

Q3)

npm install mysql

```
const mysql = require('mysql');

// Create a connection to the database
const connection = mysql.createConnection({
   host: 'your_host',
   user: 'your_user',
   password: 'your_password',
   database: 'your_database',
});

// Connect to the database
connection.connect();

// Select all records from the "customers" table
const selectQuery = 'SELECT * FROM customers';
```

```
connection.query(selectQuery, (error, results) => {
   if (error) throw error;

   console.log('All records from "customers" table:', results);

   // Specify the record to delete (replace 'your_condition' with your
specific condition)
   const deleteQuery = 'DELETE FROM customers WHERE your_condition';

   // Delete the specified record
   connection.query(deleteQuery, (deleteError, deleteResults) => {
       if (deleteError) throw deleteError;

       console.log('Record deleted successfully');

       // Close the connection
       connection.end();
   });
});
```

Slip 2

Q1) Write a Java Program to implement Singleton pattern for multithreading

```
public class Singleton {

   // Volatile keyword ensures that multiple threads handle the
uniqueInstance variable correctly
   private static volatile Singleton uniqueInstance;

   // Private constructor to prevent instantiation from outside
   private Singleton() {
       // Initialization code if needed
   }

   // Public method to get the singleton instance
```

```
    public static Singleton getInstance() {
        if (uniqueInstance == null) {
            // Double-check locking for thread safety
            synchronized (Singleton.class) {
                if (uniqueInstance == null) {
                    uniqueInstance = new Singleton();
                }
            }
        }
        return uniqueInstance;
    }

    // Other methods or properties can be added here

    public void displayMessage() {
        System.out.println("Singleton instance is created!");
    }

    public static void main(String[] args) {
        // Example usage
        Singleton singleton1 = Singleton.getInstance();
        singleton1.displayMessage();

        Singleton singleton2 = Singleton.getInstance();
        singleton2.displayMessage();

        // Both instances should be the same
        System.out.println("Are instances equal? " + (singleton1 ==
singleton2));
    }
}
```

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

```
import pandas
# reading the CSV file
csvFile = pandas.read_csv('employees.csv')
```

```
# displaying the contents of the CSV file
print(csvFile)
count=csvFile.isnull()
#displaying NULL content
print(count)
newdf = csvFile.dropna()
print(newdf)
```

Q3)
Create an HTML form that contain the Employee Registration details and write a JavaScript to validate DOB, Joining Date, and Salary.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Employee Registration Form</title>
   <style>
       body {
           font-family: Arial, sans-serif;
       }
       .error {
           color: red;
       }
   </style>
</head>
<body>

<h2>Employee Registration Form</h2>

<form id="employeeForm" onsubmit="return validateForm()">
   <label for="firstName">First Name:</label>
   <input type="text" id="firstName" name="firstName" required>
   <span id="firstNameError" class="error"></span>

   <br>

   <label for="lastName">Last Name:</label>
   <input type="text" id="lastName" name="lastName" required>
   <span id="lastNameError" class="error"></span>
```

```
    <br>

    <label for="dob">Date of Birth:</label>
    <input type="date" id="dob" name="dob" required>
    <span id="dobError" class="error"></span>

    <br>

    <label for="joiningDate">Joining Date:</label>
    <input type="date" id="joiningDate" name="joiningDate" required>
    <span id="joiningDateError" class="error"></span>

    <br>

    <label for="salary">Salary:</label>
    <input type="number" id="salary" name="salary" required>
    <span id="salaryError" class="error"></span>

    <br>

    <input type="submit" value="Register">
</form>

<script>
    function validateForm() {
        var dob = new Date(document.getElementById('dob').value);
        var joiningDate = new
Date(document.getElementById('joiningDate').value);
        var salary = document.getElementById('salary').value;

        // Validate Date of Birth
        if (isNaN(dob) || dob >= new Date()) {
            document.getElementById('dobError').innerHTML = 'Invalid Date
of Birth.';
            return false;
        } else {
            document.getElementById('dobError').innerHTML = '';
        }
```

```
        // Validate Joining Date
        if (isNaN(joiningDate) || joiningDate > new Date()) {
            document.getElementById('joiningDateError').innerHTML =
'Invalid Joining Date.';
            return false;
        } else {
            document.getElementById('joiningDateError').innerHTML = '';
        }

        // Validate Salary
        if (isNaN(salary) || salary <= 0) {
            document.getElementById('salaryError').innerHTML = 'Invalid
Salary.';
            return false;
        } else {
            document.getElementById('salaryError').innerHTML = '';
        }

        // If all validations pass, the form is submitted
        return true;
    }
</script>

</body>
</html>
```

Slip 3

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.Observable;
import java.util.Observer;

// WeatherData class represents the concrete subject that extends
java.util.Observable
```

```
class WeatherData extends Observable {
   private float temperature;
   private float humidity;
   private float pressure;

   public void measurementsChanged() {
       setChanged();
       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;
   }
}

// DisplayElement interface represents the Observer
interface DisplayElement {
   void display();
}

// CurrentConditionsDisplay is a concrete Observer that implements
DisplayElement
class CurrentConditionsDisplay implements Observer, DisplayElement {
   private float temperature;
```

```
    private float humidity;
    private Observable weatherData;

    public CurrentConditionsDisplay(Observable weatherData) {
        this.weatherData = weatherData;
        weatherData.addObserver(this);
    }

    @Override
    public void update(Observable o, Object arg) {
        if (o instanceof WeatherData) {
            WeatherData weatherData = (WeatherData) o;
            this.temperature = weatherData.getTemperature();
            this.humidity = weatherData.getHumidity();
            display();
        }
    }

    @Override
    public void display() {
        System.out.println("Current conditions: " + temperature + "F
degrees and " + humidity + "% humidity");
    }
}

public class WeatherStation {
    public static void main(String[] args) {
        WeatherData weatherData = new WeatherData();

        // Create an observer (display) and register it with the subject
(weatherData)
        CurrentConditionsDisplay currentConditionsDisplay = new
CurrentConditionsDisplay(weatherData);

        // Simulate changes in weather conditions
        weatherData.setMeasurements(80, 65, 30.4f);
        weatherData.setMeasurements(82, 70, 29.2f);
        weatherData.setMeasurements(78, 90, 29.2f);
    }
}
```

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

```
import pandas as pd
cars = pd.read_csv('data.csv')
print(cars.to_string())
ohe_cars = pd.get_dummies(cars[['Car']])
print(ohe_cars.to_string())
```

Q3)Create an HTML form for Login and write a JavaScript to validate email ID using Regular Expression.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Login Form</title>
   <style>
       body {
           font-family: Arial, sans-serif;
       }
       .error {
           color: red;
       }
   </style>
</head>
<body>

<h2>Login Form</h2>

<form id="loginForm" onsubmit="return validateForm()">
   <label for="email">Email:</label>
   <input type="text" id="email" name="email" required>
   <span id="emailError" class="error"></span>

   <br>

   <label for="password">Password:</label>
   <input type="password" id="password" name="password" required>
```

```
    <br>

    <input type="submit" value="Login">
</form>

<script>
    function validateForm() {
        var email = document.getElementById('email').value;

        // Regular expression for a simple email validation
        var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

        // Validate email using regular expression
        if (!emailRegex.test(email)) {
            document.getElementById('emailError').innerHTML = 'Invalid
email address.';
            return false;
        } else {
            document.getElementById('emailError').innerHTML = '';
        }

        // If email validation passes, the form is submitted
        return true;
    }
</script>

</body>
</html>
```

==========

Slip 4:

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.

```
// Product: Pizza interface
interface Pizza {
```

```
    void prepare();
    void bake();
    void cut();
    void box();
}

// Concrete Product: NyStyleCheesePizza
class NyStyleCheesePizza implements Pizza {
    @Override
    public void prepare() {
        System.out.println("Preparing New York Style Cheese Pizza");
    }

    @Override
    public void bake() {
        System.out.println("Baking New York Style Cheese Pizza");
    }

    @Override
    public void cut() {
        System.out.println("Cutting New York Style Cheese Pizza");
    }

    @Override
    public void box() {
        System.out.println("Boxing New York Style Cheese Pizza");
    }
}

// Concrete Product: ChicagoStyleCheesePizza
class ChicagoStyleCheesePizza implements Pizza {
    @Override
    public void prepare() {
        System.out.println("Preparing Chicago Style Cheese Pizza");
    }

    @Override
    public void bake() {
        System.out.println("Baking Chicago Style Cheese Pizza");
    }
```

```
    @Override
    public void cut() {
        System.out.println("Cutting Chicago Style Cheese Pizza");
    }

    @Override
    public void box() {
        System.out.println("Boxing Chicago Style Cheese Pizza");
    }
}

// Creator: PizzaStore abstract class
abstract class PizzaStore {
    // Factory Method
    abstract Pizza createPizza(String type);

    // Other methods for pizza ordering
    public Pizza orderPizza(String type) {
        Pizza pizza = createPizza(type);

        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();

        return pizza;
    }
}

// Concrete Creator: NYPizzaStore
class NYPizzaStore extends PizzaStore {
    // Factory Method implementation
    @Override
    Pizza createPizza(String type) {
        if (type.equals("cheese")) {
            return new NyStyleCheesePizza();
        }
        // Add more pizza types as needed
        return null;
```

```
    }
}

// Concrete Creator: ChicagoPizzaStore
class ChicagoPizzaStore extends PizzaStore {
    // Factory Method implementation
    @Override
    Pizza createPizza(String type) {
        if (type.equals("cheese")) {
            return new ChicagoStyleCheesePizza();
        }
        // Add more pizza types as needed
        return null;
    }
}

public class PizzaStoreApp {
    public static void main(String[] args) {
        PizzaStore nyPizzaStore = new NYPizzaStore();
        nyPizzaStore.orderPizza("cheese");

        System.out.println();

        PizzaStore chicagoPizzaStore = new ChicagoPizzaStore();
        chicagoPizzaStore.orderPizza("cheese");
    }
}
```

Q2)Write a python program to Implement Simple Linear Regression for predicting house price.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
df = pd.read_csv('Salary_data.csv')
print(df.to_string())
```

```
des=df.describe()
print(des)
x=df['YearsExp']
y=df['salary']
#plt.scatter(x,y)
#plt.show()
x=df['YearsExp'].values.reshape(-1,1)
y=df['salary'].values.reshape(-1,1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
ate=0)
sc=StandardScaler()
sc.fit(x_train)
x_train=sc.transform(x_train)
x_test=sc.transform(x_test)
y_train=sc.transform(y_train)
y_test=sc.transform(y_test)
LR=LinearRegression()
LR.fit(x_train,y_train)
print("Intercept",LR.intercept_)
print("Coefficient",LR.coef_)
y_pred=LR.predict(x_test)
plt.scatter(x_train,y_train)
plt.plot(x_test,y_pred,color='red')
plt.title("Simple Regression")
plt.xlabel("YearExperience")
plt.ylabel("Salary")
plt.show()
data=pd.DataFrame({'Actual':y_test.flatten(),'predicted':y_pred.flatten()}
)
print(data)
y_pred2=LR.predict([[3]])
print(y_pred2)
```

Q3) Create a Node.js file that will convert the output "Hello World!" into upper-case letters.

```
// Import the built-in 'readline' module
const readline = require('readline');

// Create an interface to read input from the console
const rl = readline.createInterface({
```

```
 input: process.stdin,
 output: process.stdout
});

// Prompt the user with a question
rl.question('Enter a string: ', (inputString) => {
 // Convert the input string to uppercase
 const uppercasedString = inputString.toUpperCase();

 // Print the result
 console.log('Uppercase Output:', uppercasedString);

 // Close the readline interface
 rl.close();
});
```

++

Slip 5:

Q1) Write a Java Program to implement Adapter pattern for Enumeration iterator

```
import java.util.Enumeration;
import java.util.Iterator;

// Enumeration interface (existing interface)
interface MyEnumeration {
   boolean hasMoreElements();
   Object nextElement();
}

// Concrete implementation of Enumeration
class MyConcreteEnumeration implements MyEnumeration {
   private String[] elements;
   private int index;

   public MyConcreteEnumeration(String[] elements) {
       this.elements = elements;
       this.index = 0;
   }
```

```
    @Override
    public boolean hasMoreElements() {
        return index < elements.length;
    }

    @Override
    public Object nextElement() {
        if (hasMoreElements()) {
            return elements[index++];
        }
        return null;
    }
}

// Adapter class that adapts Enumeration to Iterator
class EnumerationAdapter<T> implements Iterator<T> {
    private MyEnumeration enumeration;

    public EnumerationAdapter(MyEnumeration enumeration) {
        this.enumeration = enumeration;
    }

    @Override
    public boolean hasNext() {
        return enumeration.hasMoreElements();
    }

    @Override
    public T next() {
        return (T) enumeration.nextElement();
    }

    // Optional: Implement remove() method if needed
    @Override
    public void remove() {
        throw new UnsupportedOperationException("remove() method is not
supported.");
    }
}
```

```
public class AdapterPatternExample {
   public static void main(String[] args) {
       String[] elements = {"A", "B", "C", "D"};

       // Using the existing Enumeration
       MyEnumeration myEnumeration = new MyConcreteEnumeration(elements);

       // Using the Adapter to adapt Enumeration to Iterator
       Iterator<String> iterator = new
EnumerationAdapter<>(myEnumeration);

       // Iterating through the elements using Iterator
       while (iterator.hasNext()) {
           System.out.println(iterator.next());
       }
   }
}
```

Q2)Write a python program to implement Multiple Linear Regression for given dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
df = pd.read_csv("data.csv")
data=df.head()
print(data)
x= df[['Weight','Volume']]
y= df['CO2']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
ate=0)
MLR=LinearRegression()
MLR.fit(x_train,y_train)
print("Intercept",MLR.intercept_)
print("Coefficient",MLR.coef_)
```

```
#predict the CO2 emission of a car where the weight is 2300g, and the
volume is 1300ccm:
predictedCO2 = MLR.predict([[1500, 1140]])
print(predictedCO2)
```

Q3) Using nodejs create a web page to read two file names from user and append contents of first file into second file.

```
#.js file

const express = require('express');
const fs = require('fs');
const path = require('path');
const bodyParser = require('body-parser');

const app = express();
const port = 3000;

app.use(bodyParser.urlencoded({ extended: true }));

app.get('/', (req, res) => {
   res.sendFile(path.join(__dirname, 'index.html'));
});

app.post('/appendFiles', (req, res) => {
   const { sourceFileName, destinationFileName } = req.body;

   // Read the contents of the source file
   fs.readFile(sourceFileName, 'utf8', (err, data) => {
       if (err) {
           return res.status(500).send('Error reading source file');
       }

       // Append the contents to the destination file
       fs.appendFile(destinationFileName, data, (err) => {
           if (err) {
               return res.status(500).send('Error appending to destination
file');
           }
```

```
            res.status(200).send('Contents appended successfully!');
        });
    });
});

app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});

#index.html file

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>File Appender</title>
</head>
<body>
    <h2>File Appender</h2>
    <form action="/appendFiles" method="post">
        <label for="sourceFileName">Source File Name:</label>
        <input type="text" id="sourceFileName" name="sourceFileName"
required>
        <br>
        <label for="destinationFileName">Destination File Name:</label>
        <input type="text" id="destinationFileName"
name="destinationFileName" required>
        <br>
        <button type="submit">Append Files</button>
    </form>
</body>
</html>
```

==========

Slip 6:

Q1 Write a Java Program to implement command pattern to test Remote Control

```
// Command interface
interface Command {
```

```
    void execute();
}

// Concrete Command: TurnOnCommand
class TurnOnCommand implements Command {
    private Light light;

    public TurnOnCommand(Light light) {
        this.light = light;
    }

    @Override
    public void execute() {
        light.turnOn();
    }
}

// Concrete Command: TurnOffCommand
class TurnOffCommand implements Command {
    private Light light;

    public TurnOffCommand(Light light) {
        this.light = light;
    }

    @Override
    public void execute() {
        light.turnOff();
    }
}

// Receiver class: Light
class Light {
    public void turnOn() {
        System.out.println("Light is ON");
    }

    public void turnOff() {
        System.out.println("Light is OFF");
    }
```

```
}

// Invoker class: RemoteControl
class RemoteControl {
    private Command command;

    public void setCommand(Command command) {
        this.command = command;
    }

    public void pressButton() {
        command.execute();
    }
}

public class CommandPatternExample {
    public static void main(String[] args) {
        // Receiver
        Light light = new Light();

        // Concrete Commands
        Command turnOnCommand = new TurnOnCommand(light);
        Command turnOffCommand = new TurnOffCommand(light);

        // Invoker
        RemoteControl remoteControl = new RemoteControl();

        // Testing the remote control with turn on command
        remoteControl.setCommand(turnOnCommand);
        remoteControl.pressButton();

        // Testing the remote control with turn off command
        remoteControl.setCommand(turnOffCommand);
        remoteControl.pressButton();
    }
}
```

Q2) 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
# Importing the dataset
datas = pd.read_csv('data.csv')
datas
# Dividing the dataset into 2 components
X = datas.iloc[:, 1:2].values
y = datas.iloc[:, 2].values
# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin = LinearRegression()
lin.fit(X, y)
# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree = 4)
X_poly = poly.fit_transform(X)
poly.fit(X_poly, y)
lin2 = LinearRegression()
lin2.fit(X_poly, y)
# Visualising the Linear Regression results
plt.scatter(X, y, color = 'blue')
plt.plot(X, lin.predict(X), color = 'red')
plt.title('Linear Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')
plt.show()
# Visualising the Polynomial Regression results
plt.scatter(X, y, color = 'blue')
plt.plot(X, lin2.predict(poly.fit_transform(X)), color = 'red')
plt.title('Polynomial Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')
plt.show()
```

Q3) 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.

```
const express = require('express');
const fs = require('fs');
const path = require('path');

const app = express();
const port = 3000;

app.get('/:filename', (req, res) => {
   const filename = req.params.filename;
   const filePath = path.join(__dirname, filename);

   // Read the file
   fs.readFile(filePath, 'utf8', (err, data) => {
       if (err) {
           // If there's an error, send a 404 response
           res.status(404).send('File not found!');
       } else {
           // If successful, send the file content
           res.send(data);
       }
   });
});

app.listen(port, () => {
   console.log(`Server is running on http://localhost:${port}`);
});
```

==========

Slip 7:

Q1) Write a Java Program to implement undo command to test Ceiling fan.

```
import java.util.Stack;

// Command interface
interface Command {
   void execute();
   void undo();
}
```

```
// Receiver class: CeilingFan
class CeilingFan {
    private String location;
    private int speed;

    public CeilingFan(String location) {
        this.location = location;
        this.speed = 0;
    }

    public void turnOn() {
        System.out.println(location + " Ceiling Fan is ON");
    }

    public void turnOff() {
        System.out.println(location + " Ceiling Fan is OFF");
    }

    public void increaseSpeed() {
        if (speed < 3) {
            speed++;
            System.out.println(location + " Ceiling Fan speed increased to
" + speed);
        }
    }

    public void decreaseSpeed() {
        if (speed > 0) {
            speed--;
            System.out.println(location + " Ceiling Fan speed decreased to
" + speed);
        }
    }

    public int getSpeed() {
        return speed;
    }
}

// Concrete Command: CeilingFanOnCommand
```

```
class CeilingFanOnCommand implements Command {
    private CeilingFan ceilingFan;

    public CeilingFanOnCommand(CeilingFan ceilingFan) {
        this.ceilingFan = ceilingFan;
    }

    @Override
    public void execute() {
        ceilingFan.turnOn();
    }

    @Override
    public void undo() {
        ceilingFan.turnOff();
    }
}

// Concrete Command: CeilingFanOffCommand
class CeilingFanOffCommand implements Command {
    private CeilingFan ceilingFan;

    public CeilingFanOffCommand(CeilingFan ceilingFan) {
        this.ceilingFan = ceilingFan;
    }

    @Override
    public void execute() {
        ceilingFan.turnOff();
    }

    @Override
    public void undo() {
        ceilingFan.turnOn();
    }
}

// Concrete Command: CeilingFanIncreaseSpeedCommand
class CeilingFanIncreaseSpeedCommand implements Command {
    private CeilingFan ceilingFan;
```

```
    public CeilingFanIncreaseSpeedCommand(CeilingFan ceilingFan) {
        this.ceilingFan = ceilingFan;
    }

    @Override
    public void execute() {
        ceilingFan.increaseSpeed();
    }

    @Override
    public void undo() {
        ceilingFan.decreaseSpeed();
    }
}

// Concrete Command: CeilingFanDecreaseSpeedCommand
class CeilingFanDecreaseSpeedCommand implements Command {
    private CeilingFan ceilingFan;

    public CeilingFanDecreaseSpeedCommand(CeilingFan ceilingFan) {
        this.ceilingFan = ceilingFan;
    }

    @Override
    public void execute() {
        ceilingFan.decreaseSpeed();
    }

    @Override
    public void undo() {
        ceilingFan.increaseSpeed();
    }
}

// Invoker class: RemoteControl
class RemoteControl {
    private Command command;

    public void setCommand(Command command) {
```

```
        this.command = command;
    }

    public void pressButton() {
        command.execute();
    }

    public void pressUndoButton() {
        command.undo();
    }
}

public class CeilingFanTest {
    public static void main(String[] args) {
        CeilingFan ceilingFan = new CeilingFan("Living Room");

        // Concrete Commands
        CeilingFanOnCommand fanOnCommand = new
CeilingFanOnCommand(ceilingFan);
        CeilingFanOffCommand fanOffCommand = new
CeilingFanOffCommand(ceilingFan);
        CeilingFanIncreaseSpeedCommand increaseSpeedCommand = new
CeilingFanIncreaseSpeedCommand(ceilingFan);
        CeilingFanDecreaseSpeedCommand decreaseSpeedCommand = new
CeilingFanDecreaseSpeedCommand(ceilingFan);

        // Invoker
        RemoteControl remoteControl = new RemoteControl();

        // Testing the remote control with different commands
        remoteControl.setCommand(fanOnCommand);
        remoteControl.pressButton();

        remoteControl.setCommand(increaseSpeedCommand);
        remoteControl.pressButton();

        remoteControl.setCommand(decreaseSpeedCommand);
        remoteControl.pressButton();

        remoteControl.setCommand(fanOffCommand);
```

```
        remoteControl.pressButton();

        // Undo the last command
        remoteControl.pressUndoButton();
    }
}
```

Q2) Write a python program to implement Naive Bayes.

```
import math
import random
import csv
# the categorical class names are changed to numberic data
# eg: yes and no encoded to 1 and 0
def encode_class(mydata):
    classes = []
    for i in range(len(mydata)):
        if mydata[i][-1] not in classes:
            classes.append(mydata[i][-1])
    for i in range(len(classes)):
        for j in range(len(mydata)):
            if mydata[j][-1] == classes[i]:
                mydata[j][-1] = i
    return mydata

# Splitting the data
def splitting(mydata, ratio):
    train_num = int(len(mydata) * ratio)
    train = []
    # initially testset will have all the dataset
    test = list(mydata)
    while len(train) < train_num:
        # index generated randomly from range 0
        # to length of testset
        index = random.randrange(len(test))
        # from testset, pop data rows and put it in train
        train.append(test.pop(index))
    return train, test

# Group the data rows under each class yes or
```

```
# no in dictionary eg: dict[yes] and dict[no]
def groupUnderClass(mydata):
   dict = {}
   for i in range(len(mydata)):
       if (mydata[i][-1] not in dict):
           dict[mydata[i][-1]] = []
           dict[mydata[i][-1]].append(mydata[i])
   return dict

# Calculating Mean
def mean(numbers):
   return sum(numbers) / float(len(numbers))

# Calculating Standard Deviation
def std_dev(numbers):
   avg = mean(numbers)
   variance = sum([pow(x - avg, 2) for x in numbers]) / float(len(numbers)
- 1)
   return math.sqrt(variance)

def MeanAndStdDev(mydata):
   info = [(mean(attribute), std_dev(attribute)) for attribute in
zip(*mydata)]
   # eg: list = [ [a, b, c], [m, n, o], [x, y, z]]
   # here mean of 1st attribute =(a + m+x), mean of 2nd attribute = (b +
n+y)/3
   # delete summaries of last class
   del info[-1]
   return info

# find Mean and Standard Deviation under each class
def MeanAndStdDevForClass(mydata):
   info = {}
   dict = groupUnderClass(mydata)
   for classValue, instances in dict.items():
       info[classValue] = MeanAndStdDev(instances)
   return info

# Calculate Gaussian Probability Density Function
def calculateGaussianProbability(x, mean, stdev):
```

```
    expo = math.exp(-(math.pow(x - mean, 2) / (2 * math.pow(stdev, 2))))
    return (1 / (math.sqrt(2 * math.pi) * stdev)) * expo

# Calculate Class Probabilities
def calculateClassProbabilities(info, test):
    probabilities = {}
    for classValue, classSummaries in info.items():
        probabilities[classValue] = 1
        for i in range(len(classSummaries)):
            mean, std_dev = classSummaries[i]
            x = test[i]
            probabilities[classValue] *= calculateGaussianProbability(x,
mean, std_dev)
    return probabilities

# Make prediction - highest probability is the prediction
def predict(info, test):
    probabilities = calculateClassProbabilities(info, test)
    bestLabel, bestProb = None, -1
    for classValue, probability in probabilities.items():
        if bestLabel is None or probability > bestProb:
            bestProb = probability
            bestLabel = classValue
    return bestLabel

# returns predictions for a set of examples
def getPredictions(info, test):
    predictions = []
    for i in range(len(test)):
        result = predict(info, test[i])
        predictions.append(result)
    return predictions

# Accuracy score
def accuracy_rate(test, predictions):
    correct = 0
    for i in range(len(test)):
        if test[i][-1] == predictions[i]:
            correct += 1
    return (correct / float(len(test))) * 100.0
```

```
# driver code
# add the data path in your system
filename = r'E:\user\MACHINE LEARNING\machine learning algos\Naive
bayes\filedata.csv'

# load the file and store it in mydata list
mydata = csv.reader(open(filename, "rt"))
mydata = list(mydata)
mydata = encode_class(mydata)
for i in range(len(mydata)):
   mydata[i] = [float(x) for x in mydata[i]]
# split ratio = 0.7
# 70% of data is training data and 30% is test data used for testing
ratio = 0.7
train_data, test_data = splitting(mydata, ratio)
print('Total number of examples are: ', len(mydata))
print('Out of these, training examples are: ', len(train_data))
print("Test examples are: ", len(test_data))
# prepare model
info = MeanAndStdDevForClass(train_data)
# test model
predictions = getPredictions(info, test_data)
accuracy = accuracy_rate(test_data, predictions)
print("Accuracy of your model is: ", accuracy)
```

Q3) Create a Node.js file that writes an HTML form, with an upload field.

```
const express = require('express');
const multer = require('multer');
const path = require('path');

const app = express();
const port = 3000;

// Set up storage for file uploads
const storage = multer.diskStorage({
 destination: './uploads',
 filename: function (req, file, cb) {
   cb(null, file.fieldname + '-' + Date.now() +
path.extname(file.originalname));
```

```
 }
});

// Initialize multer with the storage configuration
const upload = multer({ storage: storage });

// Set up static file serving for uploaded files
app.use('/uploads', express.static('uploads'));

// Set up route for the form
app.get('/', (req, res) => {
 res.sendFile(path.join(__dirname, 'index.html'));
});

// Set up route to handle form submission (file upload)
app.post('/upload', upload.single('file'), (req, res) => {
 res.send('File uploaded successfully!');
});

app.listen(port, () => {
 console.log(`Server is running on http://localhost:${port}`);
});
```

indexedDB.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>File Upload Form</title>
</head>
<body>
   <h2>File Upload Form</h2>
   <form action="/upload" method="post" enctype="multipart/form-data">
       <label for="file">Choose a file:</label>
       <input type="file" name="file" id="file" required>
       <br>
       <input type="submit" value="Upload">
```

```
    </form>
</body>
</html>
```

==========

Slip 8:

Q1) Write a Java Program to implement State Pattern for Gumball Machine. Create instance variable that holds current state from there, we just need to handle all actions, behaviors and state transition that can happen

```
// GumballMachine class - Context
class GumballMachine {
    private State currentState;
    private int gumballsCount;

    public GumballMachine(int numGumballs) {
        gumballsCount = numGumballs;
        if (gumballsCount > 0) {
            currentState = new NoQuarterState(this);
        } else {
            currentState = new SoldOutState(this);
        }
    }

    public void insertQuarter() {
        currentState.insertQuarter();
    }

    public void ejectQuarter() {
        currentState.ejectQuarter();
    }

    public void turnCrank() {
        currentState.turnCrank();
        currentState.dispense();
    }

    public void setState(State state) {
        this.currentState = state;
    }
```

```
    public void releaseGumball() {
        System.out.println("A gumball comes rolling out of the slot...");
        if (gumballsCount > 0) {
            gumballsCount--;
        }
    }

    public int getGumballsCount() {
        return gumballsCount;
    }

    public void refill(int numGumballs) {
        gumballsCount += numGumballs;
        if (gumballsCount > 0) {
            currentState = new NoQuarterState(this);
        } else {
            currentState = new SoldOutState(this);
        }
    }
}

// State interface
interface State {
    void insertQuarter();

    void ejectQuarter();

    void turnCrank();

    void dispense();
}

// Concrete State: NoQuarterState
class NoQuarterState implements State {
    private GumballMachine gumballMachine;

    public NoQuarterState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }
```

```
    @Override
    public void insertQuarter() {
        System.out.println("You inserted a quarter.");
        gumballMachine.setState(new HasQuarterState(gumballMachine));
    }

    @Override
    public void ejectQuarter() {
        System.out.println("You haven't inserted a quarter.");
    }

    @Override
    public void turnCrank() {
        System.out.println("You turned, but there's no quarter.");
    }

    @Override
    public void dispense() {
        System.out.println("You need to pay first.");
    }
}

// Concrete State: HasQuarterState
class HasQuarterState implements State {
    private GumballMachine gumballMachine;

    public HasQuarterState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

    @Override
    public void insertQuarter() {
        System.out.println("You can't insert another quarter.");
    }

    @Override
    public void ejectQuarter() {
        System.out.println("Quarter returned.");
        gumballMachine.setState(new NoQuarterState(gumballMachine));
    }
```

```
    @Override
    public void turnCrank() {
        System.out.println("You turned...");
        gumballMachine.setState(new SoldState(gumballMachine));
    }

    @Override
    public void dispense() {
        System.out.println("No gumball dispensed.");
    }
}

// Concrete State: SoldState
class SoldState implements State {
    private GumballMachine gumballMachine;

    public SoldState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

    @Override
    public void insertQuarter() {
        System.out.println("Please wait, we're already giving you a
gumball.");
    }

    @Override
    public void ejectQuarter() {
        System.out.println("Sorry, you already turned the crank.");
    }

    @Override
    public void turnCrank() {
        System.out.println("Turning twice doesn't get you another
gumball!");
    }

    @Override
    public void dispense() {
```

```
        gumballMachine.releaseGumball();
        if (gumballMachine.getGumballsCount() > 0) {
            gumballMachine.setState(new NoQuarterState(gumballMachine));
        } else {
            System.out.println("Oops, out of gumballs!");
            gumballMachine.setState(new SoldOutState(gumballMachine));
        }
    }
}

// Concrete State: SoldOutState
class SoldOutState implements State {
    private GumballMachine gumballMachine;

    public SoldOutState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

    @Override
    public void insertQuarter() {
        System.out.println("Sorry, the machine is sold out.");
    }

    @Override
    public void ejectQuarter() {
        System.out.println("You can't eject, you haven't inserted a quarter
yet.");
    }

    @Override
    public void turnCrank() {
        System.out.println("You turned, but there are no gumballs.");
    }

    @Override
    public void dispense() {
        System.out.println("No gumball dispensed.");
    }
}
```

```
// Client code
public class GumballMachineTest {
   public static void main(String[] args) {
       GumballMachine gumballMachine = new GumballMachine(5);

       System.out.println("Initial Gumball Machine State:");
       System.out.println("Gumballs count: " +
gumballMachine.getGumballsCount());

       gumballMachine.insertQuarter();
       gumballMachine.turnCrank();

       System.out.println("\nGumball Machine State after inserting quarter
and turning crank:");
       System.out.println("Gumballs count: " +
gumballMachine.getGumballsCount());

       gumballMachine.ejectQuarter();

       System.out.println("\nGumball Machine State after ejecting
quarter:");
       System.out.println("Gumballs count: " +
gumballMachine.getGumballsCount());

       gumballMachine.insertQuarter();
       gumballMachine.turnCrank();

       System.out.println("\nGumball Machine State after inserting quarter
and turning crank:");
       System.out.println("Gumballs count: " +
gumballMachine.getGumballsCount());

       gumballMachine.insertQuarter();
       gumballMachine.turnCrank();

       System.out.println("\nGumball Machine")
```

Q2) Write a python program to implement Decision Tree whether or not to play Tennis.

```
import numpy as np
import pandas as pd
#Loading the PlayTennis data
PlayTennis = pd.read_csv("PlayTennis.csv")
PlayTennis
from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()
PlayTennis['outlook'] = Le.fit_transform(PlayTennis['outlook'])
PlayTennis['temp'] = Le.fit_transform(PlayTennis['temp'])
PlayTennis['humidity'] = Le.fit_transform(PlayTennis['humidity'])
PlayTennis['windy'] = Le.fit_transform(PlayTennis['windy'])
PlayTennis['play'] = Le.fit_transform(PlayTennis['play'])
PlayTennis
#split the training data and its coresponding prediction values.
#y - holds all the decisions.
#X - holds the training data.
y = PlayTennis['play']
X = PlayTennis.drop(['play'],axis=1)
# Fitting the model
from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X, y)
# We can visualize the tree using tree.plot_tree
tree.plot_tree(clf)
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph
```

Q3) Create a Node.js file that demonstrates create database and table in MySQL

```
const mysql = require('mysql');

// MySQL database connection configuration
const connection = mysql.createConnection({
 host: 'localhost',
 user: 'your_username',
 password: 'your_password',
});
```

```
// Connect to MySQL
connection.connect((err) => {
 if (err) throw err;
 console.log('Connected to MySQL!');

 // Create a new database
 connection.query('CREATE DATABASE IF NOT EXISTS my_database', (err) => {
   if (err) throw err;
   console.log('Database created or already exists');

   // Use the created database
   connection.query('USE my_database', (err) => {
     if (err) throw err;
     console.log('Using my_database');

     // Create a new table within the database
     const createTableQuery = `
       CREATE TABLE IF NOT EXISTS users (
         id INT AUTO_INCREMENT PRIMARY KEY,
         username VARCHAR(50) NOT NULL,
         email VARCHAR(100) NOT NULL
       )
     `;

     connection.query(createTableQuery, (err) => {
       if (err) throw err;
       console.log('Users table created or already exists');

       // Close the connection to MySQL
       connection.end((err) => {
         if (err) throw err;
         console.log('MySQL connection closed');
       });
     });
   });
 });
});
```

++=======

Slip 9:

Q1) Design simple HR Application using Spring Framework

```
// Employee.java
@Entity
public class Employee {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private String firstName;
   private String lastName;
   private String email;
   // other fields, getters, and setters
}

// Department.java
@Entity
public class Department {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private String name;
   // other fields, getters, and setters
}

// EmployeeRepository.java
public interface EmployeeRepository extends JpaRepository<Employee, Long>
{
   // custom queries if needed
}

// DepartmentRepository.java
public interface DepartmentRepository extends JpaRepository<Department,
Long> {
   // custom queries if needed
}

// EmployeeService.java
public interface EmployeeService {
   List<Employee> getAllEmployees();
   Employee getEmployeeById(Long id);
```

```
    void saveEmployee(Employee employee);
    void deleteEmployee(Long id);
}

// EmployeeServiceImpl.java
@Service
public class EmployeeServiceImpl implements EmployeeService {
    // implementation using EmployeeRepository
}


// EmployeeController.java
@Controller
@RequestMapping("/employees")
public class EmployeeController {
    // inject EmployeeService
    // define methods for handling CRUD operations
}

// DepartmentController.java
@Controller
@RequestMapping("/departments")
public class DepartmentController {
    // inject DepartmentService
    // define methods for handling CRUD operations
}


// EmployeeController.java
@Controller
@RequestMapping("/employees")
public class EmployeeController {
    // inject EmployeeService
    // define methods for handling CRUD operations
}

// DepartmentController.java
@Controller
@RequestMapping("/departments")
public class DepartmentController {
```

```
   // inject DepartmentService
   // define methods for handling CRUD operations
}
```

Q2) Write a python program to implement Linear SVM.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
# Import some Data from the iris Data Set
iris = datasets.load_iris()
# Take only the first two features of Data.
# To avoid the slicing, Two-Dim Dataset can be used
X = iris.data[:, :2]
y = iris.target
# C is the SVM regularization parameter
C = 1.0
# Create an Instance of SVM and Fit out the data.
# Data is not scaled so as to be able to plot the support vectors
svc = svm.SVC(kernel ='linear', C = 1).fit(X, y)
# create a mesh to plot
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
np.arange(y_min, y_max, h))
# Plot the data for Proper Visual Representation
plt.subplot(1, 1, 1)
# Predict the result by giving Data to the model
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap = plt.cm.Paired, alpha = 0.8)
plt.scatter(X[:, 0], X[:, 1], c = y, cmap = plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')
plt.show()
```

Q3) Create a node.js file that Select all records from the "customers" table, and display the result object on console.

```
const mysql = require('mysql');

// MySQL database connection configuration
const connection = mysql.createConnection({
 host: 'localhost',
 user: 'your_username',
 password: 'your_password',
 database: 'your_database',
});

// Connect to MySQL
connection.connect((err) => {
 if (err) throw err;
 console.log('Connected to MySQL!');

 // Select all records from the "customers" table
 const selectQuery = 'SELECT * FROM customers';

 connection.query(selectQuery, (err, result) => {
   if (err) throw err;

   // Display the result object on the console
   console.log('Result:', result);

   // Close the connection to MySQL
   connection.end((err) => {
     if (err) throw err;
     console.log('MySQL connection closed');
   });
 });
});
```

==========

Slip 10:

Q1) 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

```
// FlyingBehavior interface
interface FlyingBehavior {
   void fly();
```

```
}

// QuackingBehavior interface
interface QuackingBehavior {
    void quack();
}

// Concrete implementations of FlyingBehavior

class FlyWithWings implements FlyingBehavior {
    @Override
    public void fly() {
        System.out.println("Flying with wings");
    }
}

class FlyNoWay implements FlyingBehavior {
    @Override
    public void fly() {
        System.out.println("Cannot fly");
    }
}

// Concrete implementations of QuackingBehavior

class Quack implements QuackingBehavior {
    @Override
    public void quack() {
        System.out.println("Quack");
    }
}

class MuteQuack implements QuackingBehavior {
    @Override
    public void quack() {
        System.out.println("<< Silence >>");
    }
}

// Duck class
```

```
class Duck {
    // Instance variables to hold current state
    private FlyingBehavior flyingBehavior;
    private QuackingBehavior quackingBehavior;

    // Constructor
    public Duck(FlyingBehavior flyingBehavior, QuackingBehavior
quackingBehavior) {
        this.flyingBehavior = flyingBehavior;
        this.quackingBehavior = quackingBehavior;
    }

    // Perform the fly behavior
    public void performFly() {
        flyingBehavior.fly();
    }

    // Perform the quack behavior
    public void performQuack() {
        quackingBehavior.quack();
    }

    // Set a new flying behavior at runtime
    public void setFlyingBehavior(FlyingBehavior flyingBehavior) {
        this.flyingBehavior = flyingBehavior;
    }

    // Set a new quacking behavior at runtime
    public void setQuackingBehavior(QuackingBehavior quackingBehavior) {
        this.quackingBehavior = quackingBehavior;
    }
}

// Example usage

public class DuckBehaviorExample {
    public static void main(String[] args) {
        // Create a duck with specific behaviors
        Duck mallardDuck = new Duck(new FlyWithWings(), new Quack());
```

```
        // Test the duck's behaviors
        System.out.println("Mallard Duck behaviors:");
        mallardDuck.performFly();
        mallardDuck.performQuack();

        // Change the duck's flying behavior dynamically
        mallardDuck.setFlyingBehavior(new FlyNoWay());
        System.out.println("Mallard Duck cannot fly anymore:");
        mallardDuck.performFly();

        // Change the duck's quacking behavior dynamically
        mallardDuck.setQuackingBehavior(new MuteQuack());
        System.out.println("Mallard Duck is now silent:");
        mallardDuck.performQuack();
    }
}
```

Q2) Write a Python program to prepare Scatter Plot for Iris Dataset.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("Iris.csv")
print (data.head(10))
x=data["sepal_length"]
y=data["petal_length"]
plt.scatter(x,y)
plt.show()
```

Q3) Create a node.js file that Insert Multiple Records in "student" table, and display the result object on console.

```
const mysql = require('mysql');

// MySQL database connection configuration
const connection = mysql.createConnection({
 host: 'localhost',
 user: 'your_username',
 password: 'your_password',
 database: 'your_database',
});
```

```
// Connect to MySQL
connection.connect((err) => {
 if (err) throw err;
 console.log('Connected to MySQL!');

 // Data to be inserted into the "student" table
 const studentsData = [
   { name: 'John Doe', age: 20, grade: 'A' },
   { name: 'Jane Smith', age: 22, grade: 'B' },
   { name: 'Bob Johnson', age: 21, grade: 'C' },
   // Add more student data as needed
 ];

 // SQL query to insert multiple records into the "student" table
 const insertQuery = 'INSERT INTO student (name, age, grade) VALUES ?';

 // Execute the insert query with the array of data
 connection.query(insertQuery, [studentsData.map(student => [student.name,
student.age, student.grade])], (err, result) => {
   if (err) throw err;

   // Display the result object on the console
   console.log('Result:', result);

   // Close the connection to MySQL
   connection.end((err) => {
     if (err) throw err;
     console.log('MySQL connection closed');
   });
 });
});
```

+==========

Slip 11:

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

```
// Target interface (BeatModel)
interface BeatModel {
   void start();
   void stop();
```

```
    void increaseBPM();
    void decreaseBPM();
    int getBPM();
}

// Adaptee class (HeartModel)
class HeartModel {
    // Heart-specific methods and properties
    void startHeart() {
        System.out.println("Heart started beating");
    }

    void stopHeart() {
        System.out.println("Heart stopped beating");
    }

    void increaseHeartRate() {
        System.out.println("Heart rate increased");
    }

    void decreaseHeartRate() {
        System.out.println("Heart rate decreased");
    }

    int getHeartRate() {
        return 75; // Example heart rate
    }
}

// Adapter class (HeartAdapter)
class HeartAdapter implements BeatModel {
    private HeartModel heartModel;

    // Constructor taking a HeartModel instance
    public HeartAdapter(HeartModel heartModel) {
        this.heartModel = heartModel;
    }

    // Implementing BeatModel interface by delegating to HeartModel
    @Override
```

```
    public void start() {
        heartModel.startHeart();
    }

    @Override
    public void stop() {
        heartModel.stopHeart();
    }

    @Override
    public void increaseBPM() {
        heartModel.increaseHeartRate();
    }

    @Override
    public void decreaseBPM() {
        heartModel.decreaseHeartRate();
    }

    @Override
    public int getBPM() {
        return heartModel.getHeartRate();
    }
}

// Client code using BeatModel interface
public class AdapterPatternExample {
    public static void main(String[] args) {
        // Create a HeartModel instance
        HeartModel heartModel = new HeartModel();

        // Create a HeartAdapter to adapt the HeartModel to the BeatModel
interface
        BeatModel beatModel = new HeartAdapter(heartModel);

        // Use the BeatModel interface to interact with the adapted
HeartModel
        beatModel.start();
        System.out.println("Current BPM: " + beatModel.getBPM());
        beatModel.increaseBPM();
```

```
        System.out.println("Current BPM: " + beatModel.getBPM());
        beatModel.stop();
    }
}
```

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

```
import pandas
# reading the CSV file
csvFile = pandas.read_csv('employees.csv')
# displaying the contents of the CSV file
print(csvFile)
count=csvFile.isnull()
#displaying NULL content
print(count)
newdf = csvFile.dropna()
print(newdf)
```

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

```
const mysql = require('mysql');

// MySQL database connection configuration
const connection = mysql.createConnection({
 host: 'localhost',
 user: 'your_username',
 password: 'your_password',
 database: 'your_database',
});

// Connect to MySQL
connection.connect((err) => {
 if (err) throw err;
 console.log('Connected to MySQL!');

 // Select all records from the "customers" table
 const selectQuery = 'SELECT * FROM customers';

 connection.query(selectQuery, (err, result) => {
   if (err) throw err;
```

```
   // Display the result object on the console
   console.log('All Records:', result);

   // Specify the record to delete (change 'your_customer_id' accordingly)
   const customerIdToDelete = 1; // Example customer ID

   // Delete the specified record from the "customers" table
   const deleteQuery = `DELETE FROM customers WHERE id =
${customerIdToDelete}`;

   connection.query(deleteQuery, (err, deleteResult) => {
     if (err) throw err;

     // Display the result of the delete operation on the console
     console.log('Deleted Record:', deleteResult);

     // Close the connection to MySQL
     connection.end((err) => {
       if (err) throw err;
       console.log('MySQL connection closed');
     });
   });
 });
});
```

==========

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 interface
interface Car {
   void assemble();
}

// Concrete implementation of Car - BasicCar
class BasicCar implements Car {
   @Override
   public void assemble() {
       System.out.println("Basic Car assembled");
```

```
    }
}

// Decorator abstract class
abstract class CarDecorator implements Car {
    protected Car decoratedCar;

    public CarDecorator(Car decoratedCar) {
        this.decoratedCar = decoratedCar;
    }

    @Override
    public void assemble() {
        decoratedCar.assemble();
    }
}

// Concrete decorator - SportsCar
class SportsCar extends CarDecorator {
    public SportsCar(Car decoratedCar) {
        super(decoratedCar);
    }

    @Override
    public void assemble() {
        super.assemble();
        System.out.println("Sports Car feature added");
    }
}

// Concrete decorator - LuxuryCar
class LuxuryCar extends CarDecorator {
    public LuxuryCar(Car decoratedCar) {
        super(decoratedCar);
    }

    @Override
    public void assemble() {
        super.assemble();
        System.out.println("Luxury Car feature added");
```

```
    }
}

// Client code
public class DecoratorPatternExample {
    public static void main(String[] args) {
        // Create a basic car
        Car basicCar = new BasicCar();

        // Decorate the basic car with SportsCar features
        Car sportsCar = new SportsCar(basicCar);

        // Decorate the sports car with LuxuryCar features
        Car luxuryCar = new LuxuryCar(sportsCar);

        // Assemble the final decorated car
        luxuryCar.assemble();
    }
}
```

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

```
import pandas as pd
cars = pd.read_csv('data.csv')
print(cars.to_string())
ohe_cars = pd.get_dummies(cars[['Car']])
print(ohe_cars.to_string())
```

Q3) Create a Simple Web Server using node js.

```
const http = require('http');

// Create a server
const server = http.createServer((req, res) => {
 // Set the response header
 res.writeHead(200, {'Content-Type': 'text/plain'});

 // Send a response
 res.end('Hello, this is a simple web server!');
});
```

```
// Set the server to listen on port 3000
const PORT = 3000;
server.listen(PORT, () => {
 console.log(`Server running at http://localhost:${PORT}/`);
});
```

==========

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

```
// Volt class representing volts
class Volt {
   private int volts;

   public Volt(int volts) {
       this.volts = volts;
   }

   public int getVolts() {
       return volts;
   }
}

// Socket class producing constant 120 volts
class Socket {
   public Volt getVolts() {
       return new Volt(120);
   }
}

// Adapter interface
interface SocketAdapter {
   Volt get3Volts();
   Volt get12Volts();
   Volt getDefaultVolts();
}

// Adapter class implementing the SocketAdapter interface (Class Adapter)
```

```
class SocketClassAdapter extends Socket implements SocketAdapter {

    @Override
    public Volt get3Volts() {
        // Adapt 120 volts to 3 volts
        Volt volt = getVolts();
        return convertVolt(volt, 40);
    }

    @Override
    public Volt get12Volts() {
        // Adapt 120 volts to 12 volts
        Volt volt = getVolts();
        return convertVolt(volt, 10);
    }

    @Override
    public Volt getDefaultVolts() {
        // Use the default 120 volts from the Socket class
        return getVolts();
    }

    // Helper method to convert volts
    private Volt convertVolt(Volt volt, int divisor) {
        return new Volt(volt.getVolts() / divisor);
    }
}

// Client code
public class AdapterPatternExample {
    public static void main(String[] args) {
        SocketAdapter socketAdapter = new SocketClassAdapter();

        System.out.println("Default Volts: " +
socketAdapter.getDefaultVolts().getVolts() + "V");
        System.out.println("3 Volts: " +
socketAdapter.get3Volts().getVolts() + "V");
        System.out.println("12 Volts: " +
socketAdapter.get12Volts().getVolts() + "V");
    }}
```

Q2) Write a Python program to prepare Scatter Plot for Iris Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("Iris.csv")
print (data.head(10))
x=data["sepal_length"]
y=data["petal_length"]
plt.scatter(x,y)
plt.show()
```

Q3) Using node js create a User Login System.

```
const express = require('express');
const session = require('express-session');
const bodyParser = require('body-parser');

const app = express();
const port = 3000;

// Use session middleware
app.use(session({
 secret: 'your-secret-key',
 resave: false,
 saveUninitialized: true
}));

// Use body-parser middleware to parse POST request bodies
app.use(bodyParser.urlencoded({ extended: true }));

// In-memory user data (replace with a database in a production
environment)
const users = [
 { id: 1, username: 'user1', password: 'password1' },
 { id: 2, username: 'user2', password: 'password2' }
];

// Middleware to check if the user is authenticated
function isAuthenticated(req, res, next) {
 if (req.session && req.session.user) {
   return next();
```

```
 } else {
   return res.redirect('/login');
 }
}

// Routes

// Home page (requires authentication)
app.get('/', isAuthenticated, (req, res) => {
 res.send(`Welcome, ${req.session.user.username}! <a
href="/logout">Logout</a>`);
});

// Login page
app.get('/login', (req, res) => {
 res.sendFile(__dirname + '/login.html');
});

// Login form submission
app.post('/login', (req, res) => {
 const { username, password } = req.body;

 // Validate credentials (replace with database authentication)
 const user = users.find(u => u.username === username && u.password ===
password);

 if (user) {
   // Set user information in the session
   req.session.user = { id: user.id, username: user.username };
   res.redirect('/');
 } else {
   res.send('Invalid username or password. <a href="/login">Try
again</a>');
 }
});

// Logout
app.get('/logout', (req, res) => {
 req.session.destroy((err) => {
   if (err) {
```

```
      console.error(err);
    }
    res.redirect('/login');
  });
});

// Start the server
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});


<!DOCTYPE html>
<html>
<head>
  <title>Login</title>
</head>
<body>
  <h2>Login</h2>
  <form action="/login" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required><br>

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required><br>

    <button type="submit">Login</button>
  </form>
</body>
</html>
```

==========

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 interface
interface Command {
    void execute();
}
```

```
// Receiver class - Light
class Light {
    void turnOn() {
        System.out.println("Light is ON");
    }

    void turnOff() {
        System.out.println("Light is OFF");
    }
}

// Concrete Command - LightOnCommand
class LightOnCommand implements Command {
    private Light light;

    public LightOnCommand(Light light) {
        this.light = light;
    }

    @Override
    public void execute() {
        light.turnOn();
    }
}

// Concrete Command - LightOffCommand
class LightOffCommand implements Command {
    private Light light;

    public LightOffCommand(Light light) {
        this.light = light;
    }

    @Override
    public void execute() {
        light.turnOff();
    }
}
```

```java
// Receiver class - GarageDoor
class GarageDoor {
    void up() {
        System.out.println("Garage Door is UP");
    }
}

// Concrete Command - GarageDoorUpCommand
class GarageDoorUpCommand implements Command {
    private GarageDoor garageDoor;

    public GarageDoorUpCommand(GarageDoor garageDoor) {
        this.garageDoor = garageDoor;
    }

    @Override
    public void execute() {
        garageDoor.up();
    }
}

// Receiver class - Stereo
class Stereo {
    void onWithCD() {
        System.out.println("Stereo is ON with CD");
    }
}

// Concrete Command - StereoOnWithCDCommand
class StereoOnWithCDCommand implements Command {
    private Stereo stereo;

    public StereoOnWithCDCommand(Stereo stereo) {
        this.stereo = stereo;
    }

    @Override
    public void execute() {
        stereo.onWithCD();
    }
```

```
}

// Invoker class - RemoteControl
class RemoteControl {
    private Command command;

    public void setCommand(Command command) {
        this.command = command;
    }

    public void pressButton() {
        command.execute();
    }
}

// Client code
public class CommandPatternExample {
    public static void main(String[] args) {
        // Create instances of receivers
        Light livingRoomLight = new Light();
        GarageDoor garageDoor = new GarageDoor();
        Stereo stereo = new Stereo();

        // Create instances of concrete commands
        LightOnCommand lightOn = new LightOnCommand(livingRoomLight);
        LightOffCommand lightOff = new LightOffCommand(livingRoomLight);
        GarageDoorUpCommand garageDoorUp = new
GarageDoorUpCommand(garageDoor);
        StereoOnWithCDCommand stereoOnWithCD = new
StereoOnWithCDCommand(stereo);

        // Create an invoker
        RemoteControl remoteControl = new RemoteControl();

        // Set commands for the remote control
        remoteControl.setCommand(lightOn);
        remoteControl.pressButton();

        remoteControl.setCommand(lightOff);
        remoteControl.pressButton();
```

```
        remoteControl.setCommand(garageDoorUp);
        remoteControl.pressButton();

        remoteControl.setCommand(stereoOnWithCD);
        remoteControl.pressButton();
    }
}
```

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

```
import pandas
# reading the CSV file
csvFile = pandas.read_csv('employees.csv')
# displaying the contents of the CSV file
print(csvFile)
count=csvFile.isnull()
#displaying NULL content
print(count)
newdf = csvFile.dropna()
print(newdf)
```

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

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Web Page</title>
</head>
<body>
    <h1>Hello, this is a simple web page served by Node.js!</h1>
</body>
</html>


const http = require('http');
const fs = require('fs');
const path = require('path');
```

```
const server = http.createServer((req, res) => {
   // Determine the file path based on the request URL
   const filePath = path.join(__dirname, req.url === '/' ? 'index.html' :
req.url);

   // Check if the file exists
   fs.access(filePath, fs.constants.F_OK, (err) => {
       if (err) {
           // File not found
           res.writeHead(404, { 'Content-Type': 'text/plain' });
           res.end('404 Not Found');
           return;
       }

       // Read the file content
       fs.readFile(filePath, 'utf8', (err, data) => {
           if (err) {
               // Error reading the file
               res.writeHead(500, { 'Content-Type': 'text/plain' });
               res.end('500 Internal Server Error');
               return;
           }

           // Serve the file content
           res.writeHead(200, { 'Content-Type': 'text/html' });
           res.end(data);
       });
   });
});

const PORT = 3000;

server.listen(PORT, () => {
   console.log(`Server is running at http://localhost:${PORT}/`);
});
```

==========

Slip 15:

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

```
// Subsystem components
class Amplifier {
    void on() {
        System.out.println("Amplifier is ON");
    }

    void off() {
        System.out.println("Amplifier is OFF");
    }
}

class DVDPlayer {
    void play(String movie) {
        System.out.println("Playing DVD: " + movie);
    }

    void stop() {
        System.out.println("DVD stopped");
    }
}

class Projector {
    void on() {
        System.out.println("Projector is ON");
    }

    void off() {
        System.out.println("Projector is OFF");
    }
}

class Lights {
    void dim() {
        System.out.println("Lights are dimmed");
    }

    void bright() {
```

```
        System.out.println("Lights are brightened");
    }
}

// Facade for the Home Theater System
class HomeTheaterFacade {
    private Amplifier amplifier;
    private DVDPlayer dvdPlayer;
    private Projector projector;
    private Lights lights;

    public HomeTheaterFacade(Amplifier amplifier, DVDPlayer dvdPlayer,
Projector projector, Lights lights) {
        this.amplifier = amplifier;
        this.dvdPlayer = dvdPlayer;
        this.projector = projector;
        this.lights = lights;
    }

    // Simplified methods for the user
    public void watchMovie(String movie) {
        System.out.println("Get ready to watch a movie...");
        amplifier.on();
        dvdPlayer.play(movie);
        projector.on();
        lights.dim();
    }

    public void endMovie() {
        System.out.println("Shutting down the movie...");
        amplifier.off();
        dvdPlayer.stop();
        projector.off();
        lights.bright();
    }
}

// Client code
public class FacadePatternExample {
    public static void main(String[] args) {
```

```
        // Create subsystem components
        Amplifier amplifier = new Amplifier();
        DVDPlayer dvdPlayer = new DVDPlayer();
        Projector projector = new Projector();
        Lights lights = new Lights();

        // Create facade for the Home Theater System
        HomeTheaterFacade homeTheater = new HomeTheaterFacade(amplifier,
dvdPlayer, projector, lights);

        // Watch a movie using the facade
        homeTheater.watchMovie("Inception");

        // End the movie using the facade
        homeTheater.endMovie();
    }
}
```

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

```
import pandas as pd
cars = pd.read_csv('data.csv')
print(cars.to_string())
ohe_cars = pd.get_dummies(cars[['Car']])
print(ohe_cars.to_string())
```

Q3) 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.

```
// modules.js

// Function to get today's date and time
function getCurrentDateTime() {
    const currentDate = new Date();
```

```
   const options = { weekday: 'long', year: 'numeric', month: 'long', day:
'numeric', hour: 'numeric', minute: 'numeric', second: 'numeric',
timeZoneName: 'short' };
   const formattedDate = currentDate.toLocaleDateString('en-US', options);
   return formattedDate;
}

// Export the function to make it available externally
module.exports = {
   getCurrentDateTime: getCurrentDateTime
};

// app.js

// Require the custom module
const myModule = require('./modules');

// Use the exported function
const currentDate = myModule.getCurrentDateTime();

// Display the result
console.log('Today\'s Date and Time:', currentDate);
```

==========

SLIPS 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

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

// Subject interface
interface Subject {
   void addObserver(Observer observer);
   void removeObserver(Observer observer);
   void notifyObservers();
```

```
}

// Concrete Subject
class NumberConverter implements Subject {
    private int decimalNumber;
    private List<Observer> observers;

    public NumberConverter() {
        this.observers = new ArrayList<>();
    }

    public void setDecimalNumber(int decimalNumber) {
        this.decimalNumber = decimalNumber;
        notifyObservers();
    }

    public int getDecimalNumber() {
        return decimalNumber;
    }

    @Override
    public void addObserver(Observer observer) {
        observers.add(observer);
    }

    @Override
    public void removeObserver(Observer observer) {
        observers.remove(observer);
    }

    @Override
    public void notifyObservers() {
        for (Observer observer : observers) {
            observer.update();
        }
    }
}

// Observer interface
interface Observer {
```

```
    void update();
}

// Concrete Observers
class HexObserver implements Observer {
    private NumberConverter subject;

    public HexObserver(NumberConverter subject) {
        this.subject = subject;
        subject.addObserver(this);
    }

    @Override
    public void update() {
        System.out.println("Hexadecimal: " +
Integer.toHexString(subject.getDecimalNumber()));
    }
}

class OctalObserver implements Observer {
    private NumberConverter subject;

    public OctalObserver(NumberConverter subject) {
        this.subject = subject;
        subject.addObserver(this);
    }

    @Override
    public void update() {
        System.out.println("Octal: " +
Integer.toOctalString(subject.getDecimalNumber()));
    }
}

class BinaryObserver implements Observer {
    private NumberConverter subject;

    public BinaryObserver(NumberConverter subject) {
        this.subject = subject;
        subject.addObserver(this);
```

```
    }

    @Override
    public void update() {
        System.out.println("Binary: " +
Integer.toBinaryString(subject.getDecimalNumber()));
    }
}

// Client Code
public class ObserverPatternExample {
    public static void main(String[] args) {
        NumberConverter numberConverter = new NumberConverter();
        HexObserver hexObserver = new HexObserver(numberConverter);
        OctalObserver octalObserver = new OctalObserver(numberConverter);
        BinaryObserver binaryObserver = new
BinaryObserver(numberConverter);

        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.print("Enter a decimal number (or 'exit' to quit):
");
            String input = scanner.nextLine();

            if (input.equalsIgnoreCase("exit")) {
                break;
            }

            try {
                int decimalNumber = Integer.parseInt(input);
                numberConverter.setDecimalNumber(decimalNumber);
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please enter a valid
decimal number.");
            }
        }

        scanner.close();
    }
```

```
}
```

Q2. Write a python program to Implement Simple Linear Regression for predicting house price.

```
import numpy as np
import matplotlib.pyplot as plt

# Function to perform simple linear regression
def simple_linear_regression(x, y):
    n = len(x)

    # Calculate the mean of x and y
    x_mean = np.mean(x)
    y_mean = np.mean(y)

    # Calculate the slope (m) and y-intercept (b) of the regression line
    numerator = np.sum((x - x_mean) * (y - y_mean))
    denominator = np.sum((x - x_mean)**2)

    m = numerator / denominator
    b = y_mean - m * x_mean

    return m, b

# Function to make predictions using the regression line
def predict(x, m, b):
    return m * x + b

# Sample data (replace with your actual data)
house_sizes = np.array([1400, 1600, 1700, 1875, 1100, 1550, 2350, 2450,
1425, 1700])
house_prices = np.array([245000, 312000, 279000, 308000, 199000, 219000,
405000, 324000, 319000, 255000])

# Perform simple linear regression
slope, intercept = simple_linear_regression(house_sizes, house_prices)
```

```
# Make predictions
predicted_prices = predict(house_sizes, slope, intercept)

# Plot the regression line and data points
plt.scatter(house_sizes, house_prices, label='Actual Prices')
plt.plot(house_sizes, predicted_prices, color='red', label='Regression
Line')
plt.xlabel('House Size (sqft)')
plt.ylabel('House Price ($)')
plt.title('Simple Linear Regression for House Price Prediction')
plt.legend()
plt.show()
```

Q3. Create a js file named main.js for event-driven application. There should be a mainloop that listens for events, and then triggers a callback function when one of those
events is detected.

```
// main.js

// Example event-driven application with a main loop

// Function to simulate an event (replace with your actual event source)
function simulateEvent() {
   console.log('Event occurred!');
}

// Callback function to be executed when the event is detected
function eventCallback() {
   console.log('Event detected! Callback executed.');
}

// Main loop to listen for events
function mainLoop() {
   console.log('Main loop started.');
```

```
    // Set up event listeners (replace with your actual event listeners)
    setInterval(simulateEvent, 2000); // Simulate an event every 2 seconds

    // Event detection and callback execution
    setInterval(() => {
        // Check for the occurrence of an event (replace with your actual
event check)
        if (Math.random() < 0.5) {
            eventCallback(); // Trigger the callback function
        }
    }, 1000); // Check for events every 1 second
}

// Start the main loop
mainLoop();
```

++

SLIP 17:

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

```
// Shape interface
interface Shape {
    void draw();
}

// Concrete implementations of Shape interface
class Circle implements Shape {
    @Override
    public void draw() {
        System.out.println("Drawing Circle");
    }
}

class Square implements Shape {
    @Override
```

```
    public void draw() {
        System.out.println("Drawing Square");
    }
}

// Abstract Factory interface
interface ShapeFactory {
    Shape createShape();
}

// Concrete implementations of ShapeFactory for creating specific shapes
class CircleFactory implements ShapeFactory {
    @Override
    public Shape createShape() {
        return new Circle();
    }
}

class SquareFactory implements ShapeFactory {
    @Override
    public Shape createShape() {
        return new Square();
    }
}

// Client Code using the Abstract Factory Pattern
public class AbstractFactoryPatternExample {
    public static void main(String[] args) {
        // Create a circle using the CircleFactory
        ShapeFactory circleFactory = new CircleFactory();
        Shape circle = circleFactory.createShape();
        circle.draw();

        // Create a square using the SquareFactory
        ShapeFactory squareFactory = new SquareFactory();
        Shape square = squareFactory.createShape();
        square.draw();
    }
}
```

Q2. Write a python program to implement Multiple Linear Regression for a given dataset.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
df = pd.read_csv("data.csv")
data=df.head()
print(data)
x= df[['Weight','Volume']]
y= df['CO2']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_st
ate=0)
MLR=LinearRegression()
MLR.fit(x_train,y_train)
print("Intercept",MLR.intercept_)
print("Coefficient",MLR.coef_)
```

Q3.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.

```
const express = require('express');
const fs = require('fs');

const app = express();
const port = 3000;

// Endpoint to trigger file download
app.get('/download', (req, res) => {
   const filePath = 'path/to/your/file/sample.txt'; // Replace with the
path to your file
   const fileName = 'sample.txt'; // Replace with the desired file name

   // Set headers to force download
```

```
   res.setHeader('Content-Disposition', `attachment;
filename=${fileName}`);
   res.setHeader('Content-Type', 'application/octet-stream');

   // Read the file and stream it to the response
   const fileStream = fs.createReadStream(filePath);
   fileStream.pipe(res);
});

// Start the server
app.listen(port, () => {
   console.log(`Server is running on http://localhost:${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.Observable;
import java.util.Observer;

// WeatherData class representing the weather station
class WeatherData extends Observable {
   private float temperature;
   private float humidity;
   private float pressure;

   // Method to notify observers about measurement changes
   public void measurementsChanged() {
       setChanged();
       notifyObservers();
   }

   // Setter method for updating measurements
```

```
    public void setMeasurements(float temperature, float humidity, float
pressure) {
        this.temperature = temperature;
        this.humidity = humidity;
        this.pressure = pressure;
        measurementsChanged();
    }

    // Getter methods for temperature, humidity, and pressure
    public float getTemperature() {
        return temperature;
    }

    public float getHumidity() {
        return humidity;
    }

    public float getPressure() {
        return pressure;
    }
}

// Display class representing an observer
class CurrentConditionsDisplay implements Observer {
    private float temperature;
    private float humidity;

    @Override
    public void update(Observable o, Object arg) {
        if (o instanceof WeatherData) {
            WeatherData weatherData = (WeatherData) o;
            this.temperature = weatherData.getTemperature();
            this.humidity = weatherData.getHumidity();
            display();
        }
    }

    public void display() {
        System.out.println("Current Conditions: " + temperature + "F
degrees and " + humidity + "% humidity");
```

```
    }
}

// Main class for testing the Weather Station
public class WeatherStation {
    public static void main(String[] args) {
        WeatherData weatherData = new WeatherData();
        CurrentConditionsDisplay currentConditionsDisplay = new
CurrentConditionsDisplay();

        // Register the observer (CurrentConditionsDisplay) to the subject
(WeatherData)
        weatherData.addObserver(currentConditionsDisplay);

        // Simulate measurements update
        weatherData.setMeasurements(80, 65, 30.4f);
        weatherData.setMeasurements(82, 70, 29.2f);
        weatherData.setMeasurements(78, 90, 29.2f);
    }
}
```

Q2. Write a python program to implement Polynomial Linear Regression for given dataset

```
# Importing the libraries
import numpy as npimport matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
datas = pd.read_csv('data.csv')
datas
# Dividing the dataset into 2 components
X = datas.iloc[:, 1:2].values
y = datas.iloc[:, 2].values
# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin = LinearRegression()
lin.fit(X, y)
# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree = 4)
```

```
X_poly = poly.fit_transform(X)
poly.fit(X_poly, y)
lin2 = LinearRegression()
lin2.fit(X_poly, y)
# Visualising the Linear Regression results
plt.scatter(X, y, color = 'blue')
plt.plot(X, lin.predict(X), color = 'red')
plt.title('Linear Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')
plt.show()
# Visualising the Polynomial Regression results
plt.scatter(X, y, color = 'blue')
plt.plot(X, lin2.predict(poly.fit_transform(X)), color = 'red')
plt.title('Polynomial Regression')
plt.xlabel('Temperature')
plt.ylabel('Pressure')
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.

```
django-admin startproject myproject
cd myproject
python manage.py startapp myapp
# myapp/views.py
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
   return HttpResponse("Hello! I am learning Django")
# myapp/urls.py
from django.urls import path
from .views import index

urlpatterns = [
   path('', index, name='index'),
]
```

```
# myapp/urls.py
from django.urls import path
from .views import index

urlpatterns = [
   path('', index, name='index'),
]
python manage.py runserver
```

++

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.

```
// Pizza interface
interface Pizza {
   void prepare();
   void bake();
   void cut();
   void box();
}

// Concrete implementation of Pizza
class NyStyleCheesePizza implements Pizza {
   @Override
   public void prepare() {
       System.out.println("Preparing NY style cheese pizza");
   }

   @Override
   public void bake() {
       System.out.println("Baking NY style cheese pizza");
   }

   @Override
   public void cut() {
```

```
        System.out.println("Cutting NY style cheese pizza");
    }

    @Override
    public void box() {
        System.out.println("Boxing NY style cheese pizza");
    }
}

// Concrete implementation of Pizza
class ChicagoStyleCheesePizza implements Pizza {
    @Override
    public void prepare() {
        System.out.println("Preparing Chicago style cheese pizza");
    }

    @Override
    public void bake() {
        System.out.println("Baking Chicago style cheese pizza");
    }

    @Override
    public void cut() {
        System.out.println("Cutting Chicago style cheese pizza");
    }

    @Override
    public void box() {
        System.out.println("Boxing Chicago style cheese pizza");
    }
}

// Pizza Store interface with a factory method
abstract class PizzaStore {
    abstract Pizza createPizza(String type);

    // Order pizza using the factory method
    public Pizza orderPizza(String type) {
        Pizza pizza = createPizza(type);
```

```
        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();

        return pizza;
    }
}

// Concrete implementation of PizzaStore with the factory method
class NyPizzaStore extends PizzaStore {
    @Override
    Pizza createPizza(String type) {
        if (type.equals("cheese")) {
            return new NyStyleCheesePizza();
        }
        // Add more pizza types as needed
        return null;
    }
}

// Concrete implementation of PizzaStore with the factory method
class ChicagoPizzaStore extends PizzaStore {
    @Override
    Pizza createPizza(String type) {
        if (type.equals("cheese")) {
            return new ChicagoStyleCheesePizza();
        }
        // Add more pizza types as needed
        return null;
    }
}

// Client code to test the Pizza Store
public class PizzaStoreApp {
    public static void main(String[] args) {
        PizzaStore nyPizzaStore = new NyPizzaStore();
        PizzaStore chicagoPizzaStore = new ChicagoPizzaStore();

        // Order NY style cheese pizza
```

```
        System.out.println("Ordering NY style cheese pizza");
        Pizza nyCheesePizza = nyPizzaStore.orderPizza("cheese");

        // Order Chicago style cheese pizza
        System.out.println("\nOrdering Chicago style cheese pizza");
        Pizza chicagoCheesePizza = chicagoPizzaStore.orderPizza("cheese");
    }
}
```

Q2. Write a python program to implement Naive Bayes.

```
# Importing library
import math
import random
import csv
# the categorical class names are changed to numberic data
# eg: yes and no encoded to 1 and 0
def encode_class(mydata):
classes = []
for i in range(len(mydata)):
if mydata[i][-1] not in classes:
classes.append(mydata[i][-1])
for i in range(len(classes)):
for j in range(len(mydata)):
if mydata[j][-1] == classes[i]:
mydata[j][-1] = i
return mydata
# Splitting the data
def splitting(mydata, ratio):
train_num = int(len(mydata) * ratio)
train = []
# initially testset will have all the dataset
test = list(mydata)
while len(train) < train_num:
# index generated randomly from range 0
# to length of testset
index = random.randrange(len(test))
# from testset, pop data rows and put it in train
train.append(test.pop(index))
```

```
return train, test
# Group the data rows under each class yes or
# no in dictionary eg: dict[yes] and dict[no]
def groupUnderClass(mydata):
dict = {}for i in range(len(mydata)):
if (mydata[i][-1] not in dict):
dict[mydata[i][-1]] = []
dict[mydata[i][-1]].append(mydata[i])
return dict
# Calculating Mean
def mean(numbers):
return sum(numbers) / float(len(numbers))
# Calculating Standard Deviation
def std_dev(numbers):
avg = mean(numbers)
variance = sum([pow(x - avg, 2) for x in numbers]) / float(len(numbers) -
1)
return math.sqrt(variance)
def MeanAndStdDev(mydata):
info = [(mean(attribute), std_dev(attribute)) for attribute in
zip(*mydata)]
# eg: list = [ [a, b, c], [m, n, o], [x, y, z]]
# here mean of 1st attribute =(a + m+x), mean of 2nd attribute = (b +
n+y)/3
# delete summaries of last class
del info[-1]
return info
# find Mean and Standard Deviation under each class
def MeanAndStdDevForClass(mydata):
info = {}
dict = groupUnderClass(mydata)
for classValue, instances in dict.items():
info[classValue] = MeanAndStdDev(instances)
return info
# Calculate Gaussian Probability Density Function
def calculateGaussianProbability(x, mean, stdev):
expo = math.exp(-(math.pow(x - mean, 2) / (2 * math.pow(stdev, 2))))
return (1 / (math.sqrt(2 * math.pi) * stdev)) * exp
# Calculate Class Probabilities
def calculateClassProbabilities(info, test):
```

```
probabilities = {}
for classValue, classSummaries in info.items():probabilities[classValue] =
1
for i in range(len(classSummaries)):
mean, std_dev = classSummaries[i]
x = test[i]
probabilities[classValue] *= calculateGaussianProbability(x, mean,
std_dev)
return probabilities
# Make prediction - highest probability is the prediction
def predict(info, test):
probabilities = calculateClassProbabilities(info, test)
bestLabel, bestProb = None, -1
for classValue, probability in probabilities.items():
if bestLabel is None or probability > bestProb:
bestProb = probability
bestLabel = classValue
return bestLabel
# returns predictions for a set of examples
def getPredictions(info, test):
predictions = []
for i in range(len(test)):
result = predict(info, test[i])
predictions.append(result)
return predictions
# Accuracy score
def accuracy_rate(test, predictions):
correct = 0
for i in range(len(test)):
if test[i][-1] == predictions[i]:
correct += 1
return (correct / float(len(test))) * 100.0
# driver code# add the data path in your system
filename = r'E:\user\MACHINE LEARNING\machine learning algos\Naive
bayes\filedata.csv'
# load the file and store it in mydata list
mydata = csv.reader(open(filename, "rt"))
mydata = list(mydata)
mydata = encode_class(mydata)
for i in range(len(mydata)):
```

```
mydata[i] = [float(x) for x in mydata[i]]
# split ratio = 0.7
# 70% of data is training data and 30% is test data used for testing
ratio = 0.7
train_data, test_data = splitting(mydata, ratio)
print('Total number of examples are: ', len(mydata))
print('Out of these, training examples are: ', len(train_data))
print("Test examples are: ", len(test_data))
# prepare model
info = MeanAndStdDevForClass(train_data)
# test model
predictions = getPredictions(info, test_data)
accuracy = accuracy_rate(test_data, predictions)
print("Accuracy of your model is: ", accuracy)
```

Q3. Design a Django application that adds web pages with views and templates.

```
django-admin startproject myproject
cd myproject
python manage.py startapp myapp
from django.shortcuts import render
from django.http import HttpResponse

def home(request):
   return HttpResponse("Welcome to my Django application!")

def about(request):
   return render(request, 'about.html')
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>About Us</title>
</head>
<body>
   <h1>About Us</h1>
   <p>This is our Django application.</p>
</body>
```

```
</html>
from django.urls import path
from .views import home, about

urlpatterns = [
   path('', home, name='home'),
   path('about/', about, name='about'),
]
# myproject/urls.py
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
   path('admin/', admin.site.urls),
   path('', include('myapp.urls')),
]
python manage.py makemigrations
python manage.py migrate
python manage.py runserver
```

++

SLIP 20

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

```
import java.io.*;

// Uppercase to Lowercase I/O Decorator
class LowercaseInputStream extends FilterInputStream {

   protected LowercaseInputStream(InputStream in) {
       super(in);
   }

   @Override
   public int read() throws IOException {
       int data = super.read();
```

```
        return (data == -1 ? data : Character.toLowerCase((char) data));
    }

    @Override
    public int read(byte[] b, int offset, int length) throws IOException {
        int result = super.read(b, offset, length);
        for (int i = offset; i < offset + result; i++) {
            b[i] = (byte) Character.toLowerCase((char) b[i]);
        }
        return result;
    }
}

// Client Code
public class IODecoratorExample {
    public static void main(String[] args) {
        try {
            InputStream inputStream = new LowercaseInputStream(
                    new BufferedInputStream(
                            new FileInputStream("input.txt")
                    )
            );

            int data;
            while ((data = inputStream.read()) != -1) {
                System.out.print((char) data);
            }

            inputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
import java.io.*;

// Uppercase to Lowercase I/O Decorator
class LowercaseInputStream extends FilterInputStream {

    protected LowercaseInputStream(InputStream in) {
```

```
        super(in);
    }

    @Override
    public int read() throws IOException {
        int data = super.read();
        return (data == -1 ? data : Character.toLowerCase((char) data));
    }

    @Override
    public int read(byte[] b, int offset, int length) throws IOException {
        int result = super.read(b, offset, length);
        for (int i = offset; i < offset + result; i++) {
            b[i] = (byte) Character.toLowerCase((char) b[i]);
        }
        return result;
    }
}

// Client Code
public class IODecoratorExample {
    public static void main(String[] args) {
        try {
            InputStream inputStream = new LowercaseInputStream(
                    new BufferedInputStream(
                            new FileInputStream("input.txt")
                    )
            );

            int data;
            while ((data = inputStream.read()) != -1) {
                System.out.print((char) data);
            }

            inputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

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

```
#numpy and pandas initialization
import numpy as np
import pandas as pd
#Loading the PlayTennis data
PlayTennis = pd.read_csv("PlayTennis.csv")
PlayTennis
from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()
PlayTennis['outlook'] = Le.fit_transform(PlayTennis['outlook'])
PlayTennis['temp'] = Le.fit_transform(PlayTennis['temp'])
PlayTennis['humidity'] = Le.fit_transform(PlayTennis['humidity'])
PlayTennis['windy'] = Le.fit_transform(PlayTennis['windy'])
PlayTennis['play'] = Le.fit_transform(PlayTennis['play'])
PlayTennis#split the training data and its coresponding prediction values.
#y - holds all the decisions.
#X - holds the training data.
y = PlayTennis['play']
X = PlayTennis.drop(['play'],axis=1)
# Fitting the model
from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion = 'entropy')
clf = clf.fit(X, y)
# We can visualize the tree using tree.plot_tree
tree.plot_tree(clf)
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph
```

Q3. 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

```
django-admin startproject poll_project
```

```
cd poll_project
python manage.py startapp polls
# polls/models.py
from django.db import models

class ProgrammingLanguage(models.Model):
    name = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)

    def __str__(self):
        return self.name
python manage.py makemigrations
python manage.py migrate
# polls/admin.py
from django.contrib import admin
from .models import ProgrammingLanguage

admin.site.register(ProgrammingLanguage)
# polls/views.py
from django.shortcuts import render
from .models import ProgrammingLanguage

def index(request):
    languages = ProgrammingLanguage.objects.all()
    return render(request, 'polls/index.html', {'languages': languages})
<!-- polls/templates/polls/index.html -->
<h2>Programming Language Poll</h2>

<form action="{% url 'polls:vote' %}" method="post">
    {% csrf_token %}
    {% for language in languages %}
        <input type="radio" name="language" value="{{ language.id }}">
        {{ language.name }}<br>
    {% endfor %}
    <input type="submit" value="Vote">
</form>
# polls/urls.py
from django.urls import path
from . import views
```

```
app_name = 'polls'
urlpatterns = [
    path('', views.index, name='index'),
]
# polls/views.py
from django.shortcuts import render, get_object_or_404
from django.http import HttpResponseRedirect
from django.urls import reverse
from .models import ProgrammingLanguage

def vote(request):
    language_id = request.POST['language']
    language = get_object_or_404(ProgrammingLanguage, pk=language_id)
    language.votes += 1
    language.save()
    return HttpResponseRedirect(reverse('polls:index'))
# polls/urls.py
from django.urls import path
from . import views

app_name = 'polls'
urlpatterns = [
    path('', views.index, name='index'),
    path('vote/', views.vote, name='vote'),
]
python manage.py runserver
```

++

SLIP 20

Q1.Write a Java Program to implement command pattern to test Remote Control

```
// Command interface
interface Command {
    void execute();
}

// Concrete command classes
```

```
class LightOnCommand implements Command {
    private Light light;

    public LightOnCommand(Light light) {
        this.light = light;
    }

    @Override
    public void execute() {
        light.turnOn();
    }
}

class LightOffCommand implements Command {
    private Light light;

    public LightOffCommand(Light light) {
        this.light = light;
    }

    @Override
    public void execute() {
        light.turnOff();
    }
}

// Receiver class
class Light {
    public void turnOn() {
        System.out.println("Light is ON");
    }

    public void turnOff() {
        System.out.println("Light is OFF");
    }
}

// Invoker class
class RemoteControl {
    private Command command;
```

```
    public void setCommand(Command command) {
        this.command = command;
    }

    public void pressButton() {
        command.execute();
    }
}

// Client code to test the Command Pattern
public class CommandPatternExample {
    public static void main(String[] args) {
        // Create the receiver
        Light light = new Light();

        // Create concrete command objects
        Command lightOnCommand = new LightOnCommand(light);
        Command lightOffCommand = new LightOffCommand(light);

        // Create the invoker
        RemoteControl remoteControl = new RemoteControl();

        // Set the commands for the invoker
        remoteControl.setCommand(lightOnCommand);

        // Press the button to turn the light on
        remoteControl.pressButton();

        // Set a different command for the invoker
        remoteControl.setCommand(lightOffCommand);

        // Press the button to turn the light off
        remoteControl.pressButton();
    }
}
```

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

```
# Import the Libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
# Import some Data from the iris Data Set
iris = datasets.load_iris()
# Take only the first two features of Data.
# To avoid the slicing, Two-Dim Dataset can be used
X = iris.data[:, :2]
y = iris.target
# C is the SVM regularization parameter
C = 1.0
# Create an Instance of SVM and Fit out the data.
# Data is not scaled so as to be able to plot the support vectors
svc = svm.SVC(kernel ='linear', C = 1).fit(X, y)
# create a mesh to plot
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
np.arange(y_min, y_max, h))
# Plot the data for Proper Visual Representation
plt.subplot(1, 1, 1)
# Predict the result by giving Data to the model
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap = plt.cm.Paired, alpha = 0.8)plt.scatter(X[:,
0], X[:, 1], c = y, cmap = plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')
# Output the Plot
plt.show()
```

Q3. Design a Django application: A public site in which user can pick their favourite programming language and vote

```
# polls/models.py
```

```
from django.db import models

class ProgrammingLanguage(models.Model):
    name = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)

    def __str__(self):
        return self.name
# polls/views.py
from django.shortcuts import render
from .models import ProgrammingLanguage

def index(request):
    languages = ProgrammingLanguage.objects.all()
    return render(request, 'polls/index.html', {'languages': languages})
<!-- polls/templates/polls/index.html -->
<h2>Vote for Your Favorite Programming Language</h2>

<form action="{% url 'polls:vote' %}" method="post">
    {% csrf_token %}
    {% for language in languages %}
        <input type="radio" name="language" value="{{ language.id }}">
        {{ language.name }}<br>
    {% endfor %}
    <input type="submit" value="Vote">
</form>
# polls/urls.py
from django.urls import path
from . import views

app_name = 'polls'
urlpatterns = [
    path('', views.index, name='index'),
    path('vote/', views.vote, name='vote'),
]
# programming_poll/urls.py
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
```

```
    path('admin/', admin.site.urls),
    path('polls/', include('polls.urls')),
]
# polls/views.py
from django.shortcuts import render, get_object_or_404
from django.http import HttpResponseRedirect
from django.urls import reverse
from .models import ProgrammingLanguage

def vote(request):
    language_id = request.POST['language']
    language = get_object_or_404(ProgrammingLanguage, pk=language_id)
    language.votes += 1
    language.save()
    return HttpResponseRedirect(reverse('polls:index'))
```

++

SLIP 22

Q1. Design simple HR Application using Spring Framework

```
// Employee.java
@Entity
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String firstName;
    private String lastName;
    private String email;

    // Getters and setters
}

// Department.java
@Entity
public class Department {
    @Id
```

```java
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;

    // Getters and setters
}
// EmployeeRepository.java
public interface EmployeeRepository extends JpaRepository<Employee, Long>
{
    // Additional query methods if needed
}

// DepartmentRepository.java
public interface DepartmentRepository extends JpaRepository<Department,
Long> {
    // Additional query methods if needed
}
// EmployeeService.java
@Service
public class EmployeeService {
    @Autowired
    private EmployeeRepository employeeRepository;

    public List<Employee> getAllEmployees() {
        return employeeRepository.findAll();
    }

    public void saveEmployee(Employee employee) {
        employeeRepository.save(employee);
    }

    public Employee getEmployeeById(Long id) {
        return employeeRepository.findById(id).orElse(null);
    }

    public void deleteEmployee(Long id) {
        employeeRepository.deleteById(id);
    }
}
```

```java
// DepartmentService.java
@Service
public class DepartmentService {
    @Autowired
    private DepartmentRepository departmentRepository;

    public List<Department> getAllDepartments() {
        return departmentRepository.findAll();
    }

    public void saveDepartment(Department department) {
        departmentRepository.save(department);
    }

    public Department getDepartmentById(Long id) {
        return departmentRepository.findById(id).orElse(null);
    }

    public void deleteDepartment(Long id) {
        departmentRepository.deleteById(id);
    }
}
// EmployeeController.java
@RestController
@RequestMapping("/employees")
public class EmployeeController {
    @Autowired
    private EmployeeService employeeService;

    @GetMapping
    public List<Employee> getAllEmployees() {
        return employeeService.getAllEmployees();
    }

    @GetMapping("/{id}")
    public Employee getEmployeeById(@PathVariable Long id) {
        return employeeService.getEmployeeById(id);
    }
```

```
    @PostMapping
    public void saveEmployee(@RequestBody Employee employee) {
        employeeService.saveEmployee(employee);
    }

    @DeleteMapping("/{id}")
    public void deleteEmployee(@PathVariable Long id) {
        employeeService.deleteEmployee(id);
    }
}

// DepartmentController.java
@RestController
@RequestMapping("/departments")
public class DepartmentController {
    @Autowired
    private DepartmentService departmentService;

    @GetMapping
    public List<Department> getAllDepartments() {
        return departmentService.getAllDepartments();
    }

    @GetMapping("/{id}")
    public Department getDepartmentById(@PathVariable Long id) {
        return departmentService.getDepartmentById(id);
    }

    @PostMapping
    public void saveDepartment(@RequestBody Department department) {
        departmentService.saveDepartment(department);
    }

    @DeleteMapping("/{id}")
    public void deleteDepartment(@PathVariable Long id) {
        departmentService.deleteDepartment(id);
    }
}
```

Q2. Write a Python program to prepare Scatter Plot for Iris Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv("Iris.csv")
print (data.head(10))
x=data["sepal_length"]
y=data["petal_length"]
plt.scatter(x,y)
plt.show()
```

Q3.Design a Django application: An admin site that lets you add, change and delete programming languages.

```
django-admin startproject programming_languages
cd programming_languages
python manage.py startapp languages
# languages/models.py
from django.db import models

class ProgrammingLanguage(models.Model):
    name = models.CharField(max_length=200)

    def __str__(self):
        return self.name
# languages/admin.py
from django.contrib import admin
from .models import ProgrammingLanguage

admin.site.register(ProgrammingLanguage)
# languages/urls.py
from django.urls import path
from . import views

app_name = 'languages'
urlpatterns = [
    # Define your views and URL patterns if needed
]
# programming_languages/urls.py
```

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
   path('admin/', admin.site.urls),
   path('languages/', include('languages.urls')),
]
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
python manage.py createsuperuser
```

++

SLIP 23

Write a Java Program to implement State Pattern for Gumball Machine. Create instance variable that holds current state from there, we just need to handle all actions, behaviors and state transition that can happen

```
// State interface
interface State {
   void insertQuarter();
   void ejectQuarter();
   void turnCrank();
   void dispense();
}

// GumballMachine class
class GumballMachine {
   private State hasQuarterState;
   private State noQuarterState;
   private State soldOutState;

   private State currentState;

   private int gumballCount;

   public GumballMachine(int initialGumballCount) {
```

```
        hasQuarterState = new HasQuarterState(this);
        noQuarterState = new NoQuarterState(this);
        soldOutState = new SoldOutState(this);

        this.gumballCount = initialGumballCount;
        if (gumballCount > 0) {
            currentState = noQuarterState;
        } else {
            currentState = soldOutState;
        }
    }

    public void setCurrentState(State currentState) {
        this.currentState = currentState;
    }

    public State getHasQuarterState() {
        return hasQuarterState;
    }

    public State getNoQuarterState() {
        return noQuarterState;
    }

    public State getSoldOutState() {
        return soldOutState;
    }

    public int getGumballCount() {
        return gumballCount;
    }

    public void insertQuarter() {
        currentState.insertQuarter();
    }

    public void ejectQuarter() {
        currentState.ejectQuarter();
    }
```

```
    public void turnCrank() {
        currentState.turnCrank();
        currentState.dispense();
    }

    public void releaseBall() {
        System.out.println("A gumball comes rolling out the slot...");
        if (gumballCount != 0) {
            gumballCount--;
        }
    }
}

// Concrete State classes
class HasQuarterState implements State {
    private GumballMachine gumballMachine;

    public HasQuarterState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

    @Override
    public void insertQuarter() {
        System.out.println("You can't insert another quarter");
    }

    @Override
    public void ejectQuarter() {
        System.out.println("Quarter returned");
        gumballMachine.setCurrentState(gumballMachine.getNoQuarterState());
    }

    @Override
    public void turnCrank() {
        System.out.println("You turned...");
        gumballMachine.setCurrentState(gumballMachine.getSoldOutState());
    }

    @Override
    public void dispense() {
```

```
        System.out.println("No gumball dispensed");
    }
}

class NoQuarterState implements State {
    private GumballMachine gumballMachine;

    public NoQuarterState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

    @Override
    public void insertQuarter() {
        System.out.println("You inserted a quarter");

gumballMachine.setCurrentState(gumballMachine.getHasQuarterState());
    }

    @Override
    public void ejectQuarter() {
        System.out.println("You haven't inserted a quarter");
    }

    @Override
    public void turnCrank() {
        System.out.println("You turned, but there's no quarter");
    }

    @Override
    public void dispense() {
        System.out.println("You need to pay first");
    }
}

class SoldOutState implements State {
    private GumballMachine gumballMachine;

    public SoldOutState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }
```

```
    @Override
    public void insertQuarter() {
        System.out.println("You can't insert a quarter, the machine is sold
out");
    }

    @Override
    public void ejectQuarter() {
        System.out.println("You can't eject, you haven't inserted a quarter
yet");
    }

    @Override
    public void turnCrank() {
        System.out.println("You turned, but there are no gumballs");
    }

    @Override
    public void dispense() {
        System.out.println("No gumball dispensed");
    }
}

// Client code to test the State Pattern
public class GumballMachineTest {
    public static void main(String[] args) {
        GumballMachine gumballMachine = new GumballMachine(5);

        System.out.println("Initial Gumball Machine State:");
        System.out.println(gumballMachine.getGumballCount() + " gumballs
left\n");

        gumballMachine.insertQuarter();
        gumballMachine.turnCrank();

        System.out.println("\nGumball Machine State After Inserting Quarter
and Turning Crank:");
        System.out.println(gumballMachine.getGumballCount() + " gumballs
left");
```

```
    }
}
```

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

Same as slip 11

Q3.Create your own blog using Django.

```
# Create a new project
django-admin startproject myblog
cd myblog

# Create a new app
python manage.py startapp blog
# blog/models.py
from django.db import models
from django.utils import timezone

class Post(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
    pub_date = models.DateTimeField(default=timezone.now)

    def __str__(self):
        return self.title

class Comment(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    author = models.CharField(max_length=100)
    text = models.TextField()
    pub_date = models.DateTimeField(default=timezone.now)

    def __str__(self):
        return f"{self.author} on {self.post.title}"
python manage.py makemigrations
python manage.py migrate
# blog/admin.py
from django.contrib import admin
```

```
from .models import Post, Comment

admin.site.register(Post)
admin.site.register(Comment)
# blog/urls.py
from django.urls import path
from . import views

app_name = 'blog'
urlpatterns = [
    path('', views.post_list, name='post_list'),
    path('post/<int:pk>/', views.post_detail, name='post_detail'),
    path('post/new/', views.post_new, name='post_new'),
    path('post/<int:pk>/edit/', views.post_edit, name='post_edit'),
]
# myblog/urls.py
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('blog/', include('blog.urls')),
]
# blog/views.py
from django.shortcuts import render, get_object_or_404, redirect
from .models import Post
from .forms import PostForm

def post_list(request):
    posts = Post.objects.order_by('-pub_date')
    return render(request, 'blog/post_list.html', {'posts': posts})

def post_detail(request, pk):
    post = get_object_or_404(Post, pk=pk)
    return render(request, 'blog/post_detail.html', {'post': post})

def post_new(request):
    if request.method == 'POST':
        form = PostForm(request.POST)
        if form.is_valid():
```

```
            post = form.save(commit=False)
            post.save()
            return redirect('blog:post_detail', pk=post.pk)
    else:
        form = PostForm()
    return render(request, 'blog/post_edit.html', {'form': form})

def post_edit(request, pk):
    post = get_object_or_404(Post, pk=pk)
    if request.method == 'POST':
        form = PostForm(request.POST, instance=post)
        if form.is_valid():
            post = form.save(commit=False)
            post.save()
            return redirect('blog:post_detail', pk=post.pk)
    else:
        form = PostForm(instance=post)
    return render(request, 'blog/post_edit.html', {'form': form})
# blog/forms.py
from django import forms
from .models import Post

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = ['title', 'content']
# blog/forms.py
from django import forms
from .models import Post

class PostForm(forms.ModelForm):
    class Meta:
        model = Post
        fields = ['title', 'content']
```

==========

SLIP 24

Q1.Write a Java Program to implement Iterator Pattern for Designing Menu like Breakfast,
Lunch or Dinner Menu

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

// Iterator interface
interface Iterator<T> {
   boolean hasNext();
   T next();
}

// Menu interface
interface Menu {
   Iterator<String> createIterator();
}

// Concrete Iterator class
class ArrayIterator implements Iterator<String> {
   private String[] items;
   private int position = 0;

   public ArrayIterator(String[] items) {
       this.items = items;
   }

   @Override
   public boolean hasNext() {
       return position < items.length && items[position] != null;
   }

   @Override
   public String next() {
       return items[position++];
   }
}

// Concrete Menu class - Breakfast Menu
class BreakfastMenu implements Menu {
```

```
    private String[] items;

    public BreakfastMenu() {
        items = new String[]{"Pancakes", "Bacon and Eggs", "Toast with
Jam", "Omelette"};
    }

    @Override
    public Iterator<String> createIterator() {
        return new ArrayIterator(items);
    }
}

// Concrete Menu class - Lunch Menu
class LunchMenu implements Menu {
    private List<String> items;

    public LunchMenu() {
        items = new ArrayList<>();
        items.add("Chicken Sandwich");
        items.add("Caesar Salad");
        items.add("Spaghetti Bolognese");
    }

    @Override
    public Iterator<String> createIterator() {
        return new ListIterator(items);
    }
}

// Concrete Menu class - Dinner Menu
class DinnerMenu implements Menu {
    private String[] items;

    public DinnerMenu() {
        items = new String[]{"Steak", "Grilled Salmon", "Vegetarian
Lasagna"};
    }

    @Override
```

```
    public Iterator<String> createIterator() {
        return new ArrayIterator(items);
    }
}

// Concrete Iterator class for List
class ListIterator implements Iterator<String> {
    private List<String> items;
    private int position = 0;

    public ListIterator(List<String> items) {
        this.items = items;
    }

    @Override
    public boolean hasNext() {
        return position < items.size() && items.get(position) != null;
    }

    @Override
    public String next() {
        return items.get(position++);
    }
}

// Waitress class to demonstrate Iterator Pattern
class Waitress {
    private Menu breakfastMenu;
    private Menu lunchMenu;
    private Menu dinnerMenu;

    public Waitress(Menu breakfastMenu, Menu lunchMenu, Menu dinnerMenu) {
        this.breakfastMenu = breakfastMenu;
        this.lunchMenu = lunchMenu;
        this.dinnerMenu = dinnerMenu;
    }

    public void printMenus() {
        System.out.println("Breakfast Menu:");
        printMenu(breakfastMenu.createIterator());
```

```
        System.out.println("\nLunch Menu:");
        printMenu(lunchMenu.createIterator());

        System.out.println("\nDinner Menu:");
        printMenu(dinnerMenu.createIterator());
    }

    private void printMenu(Iterator<String> iterator) {
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}

// Client code to test the Iterator Pattern
public class IteratorPatternExample {
    public static void main(String[] args) {
        Menu breakfastMenu = new BreakfastMenu();
        Menu lunchMenu = new LunchMenu();
        Menu dinnerMenu = new DinnerMenu();

        Waitress waitress = new Waitress(breakfastMenu, lunchMenu,
dinnerMenu);
        waitress.printMenus();
    }
}
```

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

```
import pandas as pd
cars = pd.read_csv('data.csv')
print(cars.to_string())
ohe_cars = pd.get_dummies(cars[['Car']])
print(ohe_cars.to_string())
```

```
from sklearn.preprocessing import LabelEncoder
import pandas as pd
```

```
# Sample dataset with categorical values
data = {'Category': ['A', 'B', 'C', 'A', 'B', 'C', 'A']}
df = pd.DataFrame(data)

# Instantiate the LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform the categorical column
df['Category_numeric'] = label_encoder.fit_transform(df['Category'])

# Display the result
print(df)
```

Q.3 Implement Login System using Django.

```
django-admin startproject myloginproject
cd myloginproject
python manage.py startapp myloginapp
# myloginapp/models.py
from django.db import models
from django.contrib.auth.models import AbstractUser

class CustomUser(AbstractUser):
    # You can add additional fields if needed
    pass
# myloginproject/settings.py
AUTH_USER_MODEL = 'myloginapp.CustomUser'
python manage.py makemigrations
python manage.py migrate
# myloginapp/forms.py
from django import forms
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm

class UserRegistrationForm(UserCreationForm):
    email = forms.EmailField(required=True)

    class Meta:
        model = CustomUser
```

```
        fields = ['username', 'email', 'password1', 'password2']

class UserLoginForm(AuthenticationForm):
    class Meta:
        model = CustomUser
        fields = ['username', 'password']
# myloginapp/views.py
from django.shortcuts import render, redirect
from django.contrib.auth import login, authenticate
from django.contrib.auth.forms import AuthenticationForm
from .forms import UserRegistrationForm

def user_registration(request):
    if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('myloginapp:login')
    else:
        form = UserRegistrationForm()
    return render(request, 'myloginapp/registration.html', {'form': form})

def user_login(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect('myloginapp:home')
    else:
        form = AuthenticationForm()
    return render(request, 'myloginapp/login.html', {'form': form})

def user_logout(request):
    logout(request)
    return redirect('myloginapp:login')

def home(request):
    return render(request, 'myloginapp/home.html')
# myloginapp/urls.py
```

```
from django.urls import path
from . import views

app_name = 'myloginapp'

urlpatterns = [
   path('register/', views.user_registration, name='register'),
   path('login/', views.user_login, name='login'),
   path('logout/', views.user_logout, name='logout'),
   path('home/', views.home, name='home'),
]
# myloginproject/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
   path('admin/', admin.site.urls),
   path('myloginapp/', include('myloginapp.urls')),
]
<!-- myloginapp/templates/myloginapp/registration.html -->
{% extends 'myloginapp/base.html' %}

{% block content %}
 <h2>Register</h2>
 <form method="post" action="{% url 'myloginapp:register' %}">
   {% csrf_token %}
   {{ form.as_p }}
   <button type="submit">Register</button>
 </form>
{% endblock %}
<!-- myloginapp/templates/myloginapp/home.html -->
{% extends 'myloginapp/base.html' %}

{% block content %}
 <h2>Welcome, {{ user.username }}!</h2>
 <a href="{% url 'myloginapp:logout' %}">Logout</a>
{% endblock %}
<!-- myloginapp/templates/myloginapp/base.html -->
<!DOCTYPE html>
<html lang="en">
```

```
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Login System</title>
</head>
<body>
 <h1>Login System</h1>
 <div>
   {% block content %}
   {% endblock %}
 </div>
</body>
</html>
# myloginproject/settings.py
INSTALLED_APPS = [
   # ...
   'myloginapp',
   # ...
]
# myloginproject/settings.py
INSTALLED_APPS = [
   # ...
   'myloginapp',
   # ...
]
```

==========

SLIP 25

Q1. Write a Java Program to implement Singleton pattern for multithreading

Same as slip 2

Q2.Write a python program to Implement Simple Linear Regression for predicting house price.

Same as slip 4 and 16

Q.3 Create a Simple Web Server using node js.

Same as slip 12

```
// server.js
const http = require('http');

const server = http.createServer((req, res) => {
 res.writeHead(200, { 'Content-Type': 'text/plain' });
 res.end('Hello, this is a simple web server!');
});

const PORT = 3000;

server.listen(PORT, () => {
 console.log(`Server is running at http://localhost:${PORT}/`);
});
```

Node server.js

==========

SLIP 26

**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.**

Same as slip 10

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

Same as Slip 5, 17

Create a Node.js file that demonstrates create database and table in MySQL.

Same as slip 8

==========

SLIP 27

Q1. Write a Java Program to implement Abstract Factory Pattern for Shape interface.
Same as slip 17

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

Same as slip 18

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

Same as slip 18

==========

SLIP 28

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()

Same as slip 18 and 3

Q2. Write a python program to implement Naive Bayes.

Same as slip 18 and 3

Q3. Create your own blog using Django

Same as slip 23

==========

SLIP 29

Q1. Write a Java Program to implement State Pattern for Gumball Machine.

**Create instance variable that holds current state from there, we just need to handle all
actions, behaviors and state transition that can happen**

Same as slip 8 and 23

Q2. Write a python program to implement Decision Tree whether or not to play Tennis.

Same as slip 8 and 20

Q3.Create a clone of the "Hacker News" website.

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Hacker News Clone</title>
 <link rel="stylesheet" href="styles.css">
</head>
<body>
 <div id="app">
   <header>
     <h1>Hacker News Clone</h1>
   </header>
   <main>
     <ul id="news-list"></ul>
   </main>
 </div>
 <script src="app.js"></script>
</body>
</html>
```

```
body {
   font-family: Arial, sans-serif;
   margin: 0;
   padding: 0;
 }
```

```css
 #app {
  max-width: 800px;
  margin: 0 auto;
}
 header {
  background-color: #f60;
  color: white;
  padding: 10px;
}
 main {
  padding: 20px;
}
 ul {
  list-style-type: none;
  padding: 0;
}
 li {
  border-bottom: 1px solid #ddd;
  padding: 10px 0;
}
 a {
  text-decoration: none;
  color: #333;
}
 a:hover {
  text-decoration: underline;
}
```

```javascript
document.addEventListener('DOMContentLoaded', () => {
  const newsList = document.getElementById('news-list');
   // Simulated data (replace this with actual API calls)
  const fakeNews = [
    { title: 'Article 1', url: 'https://example.com/article1' },
    { title: 'Article 2', url: 'https://example.com/article2' },
    { title: 'Article 3', url: 'https://example.com/article3' },
  ];
   // Render news
  fakeNews.forEach((item, index) => {
    const li = document.createElement('li');
```

```
    const a = document.createElement('a');
    a.href = item.url;
    a.textContent = `${index + 1}. ${item.title}`;
    li.appendChild(a);
    newsList.appendChild(li);
  });
});
```

==========

SLIP 30

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.

Same as Slip 4 and 19

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

Same as slip 21 and 9

Q3. Implement Login System using Django.

Same as slip 24