**Slip no 1**

// Write a PHP script to keep track of number of times the web page has been accessed (Use Session Tracking).

<?php

    session\_start();

    if(!isset($\_SESSION['count']))

    {

        echo"Welcome you have viwed this page first time";

        $\_SESSION['count'] = 1;

    }

    else

    {

        $\_SESSION['count']++;

        echo"You have viewed this page ".$\_SESSION['count']." times";

    }

?>

#Create ‘Position\_Salaries’ Data set. Build a linear regression model by identifying independent and target variable. Split the variables into training and testing sets. then divide the training and testing sets into a 7:3 ratio, respectively and print them. Build a simple linear regression model.

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.metrics import r2\_score, mean\_squared\_error

salary = pd.read\_csv('CSV/Position\_Salaries.csv')

salary.sample(5)

new\_sal = salary[['Level', 'Salary']]

x = np.array(new\_sal[['Level']])

y = np.array(new\_sal[['Salary']])

x\_train, x\_test, y\_t/rain, y\_test = train\_test\_split(x,y, test\_size=0.70, random\_state=0)

regret = LinearRegression()

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

plt.scatter(x\_test, y\_test, color = 'green')

plt.plot(x\_train, regret.predict(x\_train), color='red',linewidth=3)

plt.title('Regression(Position\_salaries)')

plt.xlabel('Level')

plt.ylabel('Salary')

plt.show()

**Slip no 2**

//Write a PHP script to change the preferences of your web page like font style, font size, font color, background color using cookie. Display selected setting on next web page and actual implementation (with new settings) on third page (Use Cookies).

//html file

<html>

<body>

<form action="1.php" method="get">

<center>

<b>select font style:</b><input type=text name=s1><br>

<b>Enter font size:</b><input type=text name=s><br>

<b>Enter font color:</b><input type=text name=c><br>

<b>Enter background color:</b><input type=text name=b><br>

<input type=submit value="Next">

</center>

</form>

</body>

<html>

//php1 file

<?php

    echo "Style is ".$\_GET['s1']." Color is ".$\_GET['c']." Background color is ".$\_GET['b']." size is ".$\_GET['s'];

    setcookie("set1",$\_GET['s1'],time()+3600);

    setcookie("set2",$\_GET['c'],time()+3600);

    setcookie("set3",$\_GET['b'],time()+3600);

    setcookie("set4",$\_GET['s'],time()+3600);

?>

<a href="2.php"><br><br>Show</a>

//php2 file

<?php

$style=$\_COOKIE['set1'];

$color=$\_COOKIE['set2'];

$size=$\_COOKIE['set4'];

$b\_color=$\_COOKIE['set3'];

$msg="Hello php";

echo"<body bgcolor=$b\_color>";

echo"<font color=$color size=$size face=$style>$msg";

echo"</font></body>";

?>

//Create ‘Salary’ Data set . Build a linear regression model by identifying independent and target

variable. Split the variables into training and testing sets and print them. Build a simple linear regression model for predicting purchases.

import numpy as np

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

num\_samples = 1000

salary\_mean = 50000

salary\_std = 10000

purchases\_slope = 0.001

purchases\_intercept = 10

salary = np.random.normal(salary\_mean, salary\_std, num\_samples)

purchases = salary \* purchases\_slope + purchases\_intercept + np.random.normal(0, 5, num\_samples)

data = pd.DataFrame({'Salary': salary, 'Purchases': purchases})

X = data[['Salary']]

y = data['Purchases']

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

model = LinearRegression()

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

train\_rmse = np.sqrt(mean\_squared\_error(y\_train, model.predict(X\_train)))

test\_rmse = np.sqrt(mean\_squared\_error(y\_test, model.predict(X\_test)))

print("Training RMSE:", train\_rmse)

print("Testing RMSE:", test\_rmse)

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

plt.plot(X\_train, model.predict(X\_train), color='red',linewidth=3)

plt.title('Regression(Salary)')

plt.xlabel('Salary')

plt.ylabel('Purchases')

plt.show()

**Slip no 3**

//Write a PHP script to accept username and password. If in the first three chances, username and

password entered is correct then display second form with “Welcome message” otherwise display error message. [Use Session]

//php1

<?php

session\_start();

$correctUsername = "admin";

$correctPassword = "admin";

if (!isset($\_SESSION['login\_attempts']))

{

    $\_SESSION['login\_attempts'] = 0;

}

if ($\_SERVER['REQUEST\_METHOD'] === 'POST')

{

    $\_SESSION['login\_attempts']++;

    $enteredUsername = $\_POST['username'];

    $enteredPassword = $\_POST['password'];

    if ($enteredUsername === $correctUsername && $enteredPassword === $correctPassword)

    {

        unset($\_SESSION['login\_attempts']);

        header("Location: php2.php");

        exit();

    }

    elseif ($\_SESSION['login\_attempts'] >= 3)

    {

        unset($\_SESSION['login\_attempts']);

        session\_destroy();

        exit();

    }

}

?>

<form method="post" action="">

    <label for="username">Username:</label>

    <input type="text" name="username" required><br>

    <label for="password">Password:</label>

    <input type="password" name="password" required><br>

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

</form>

<?php if ($\_SESSION['login\_attempts'] > 0 && $\_SESSION['login\_attempts'] < 3): ?>

    <p>Remaining login attempts: <?php echo 3 - $\_SESSION['login\_attempts']; ?></p>

<?php endif; ?>

//php2

<?php

echo "Welcome to my web page!";

?>

#Create ‘User’ Data set having 5 columns namely: User ID, Gender, Age, Estimated Salary and Purchased. Build a logistic regression model that can predict whether on the given parameter a person will buy a car or not.

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report

import pandas as pd

df = pd.read\_csv('./csv/User\_Data.csv')

# Perform one-hot encoding on the 'Gender' column

df = pd.get\_dummies(df, columns=['Gender'], drop\_first=True)

X = df[['Gender\_Male', 'Age', 'EstimatedSalary']]

y = df['Purchased']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=15)

model = LogisticRegression(max\_iter=1000)

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

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

df

def predict\_purchase(gender, age, salary, model):

    gender\_num = 1 if gender.lower() == 'male' else 0

    input\_data = pd.DataFrame([[gender\_num, age, salary]], columns=['Gender\_Male', 'Age', 'EstimatedSalary'])

    prediction = model.predict(input\_data)

    return "Will Purchase a car" if prediction[0] == 1 else "Will Not Purchase a car"

gender = "Male"

age = 42

salary = 149000

result = predict\_purchase(gender, age, salary, model)

print(result)

**Slip no 4**

//Write a PHP script to accept Employee details (Eno, Ename, Address) on first page. On second

page accept earning (Basic, DA, HRA). On third page print Employee information (Eno, Ename, Address, Basic, DA, HRA, Total) [ Use Session]

//html1

<html>

<body>

<form action="php1.php" method="post">

<center><h2>Enter employee details</h2><br>

<table>

<tr>

<td><br>Emp no</b></td>

<td><input type=text name=eno></td>

</tr>

<tr>

<td><b>Name</b></td>

<td><input type=text name=enm></td>

</tr>

<tr>

<td><b>Address</b></td>

<td><input type=text name=eadd></td>

</tr>

</table>

<br><input type=submit value=show name=submit>

</center>

</form>

</body>

</html>

//php1

<?php

session\_start();

$eno=$\_POST['eno'];

$enm=$\_POST['enm'];

$eadd=$\_POST['eadd'];

$\_SESSION['eno']=$eno;

$\_SESSION['enm']=$enm;

$\_SESSION['eadd']=$eadd;

?>

<a href="p2.html"><br>Show</a>

//html2

<html>

<body>

<form action="php2.php" method="post">

<center><h2>Enter earnings of employee</h2><br>

<table>

<tr>

<td>Basic:</td>

<td><input type="text" name="e1"></td></tr>

<tr>

<td>DA:</td>

<td><input type="text" name="e2"></td></tr>

<tr>

<td>HRA:</td>

<td><input type="text" name="e3"></td></tr>

<tr><td></td><td><input type=submit value=Next></td></tr>

</table>

</center>

</form>

</body>

</html>

//php2

<?php

session\_start();

$e1=$\_POST['e1'];

$e2=$\_POST['e2'];

$e3=$\_POST['e3'];

echo"Employee Details";

echo "<br>Eno:".$\_SESSION['eno']."<br>";

echo "Name:".$\_SESSION['enm']."<br>";

echo "Address:".$\_SESSION['eadd']."<br>";

echo "class:".$\_SESSION['eadd']."<br>";

echo"basic:".$e1."<br>";

echo"DA:".$e2."<br>";

echo"HRA:".$e3."<br>";

$total=$e1+$e2+$e3;

echo"<h2>total of earnings is".$total."</h2>";

?>

#Build a simple linear regression model for Fish Species Weight Prediction.

import numpy as np

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

fish\_data = pd.read\_csv('CSV/Fish.csv')

X = fish\_data[['Width']]  # Features

y = fish\_data['Weight']  # Target variable

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

model = LinearRegression()

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

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

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print("Mean Squared Error:", mse)

print("R-squared:", r2)

print("\nCoefficients:", model.coef\_)

print("Intercept:", model.intercept\_)

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

plt.plot(X\_train, model.predict(X\_train), color='red',linewidth=3)

plt.title('Regression(Salary)')

plt.xlabel('Features')

plt.ylabel('Weight')

plt.show()

**Slip no 5**

//Create XML file named “Item.xml”with item-name, item-rate, item quantity Store the details of 5 Items of different Types

<?xml version="1.0" encoding="UTF-8"?>

<ItemDetails>

<Item>

<ItemName>Shoes</ItemName>

<ItemPrice>1000</ItemPrice>

<Quantity>3</Quantity>

</Item>

<Item>

<ItemName>Jeans</ItemName>

<ItemPrice>900</ItemPrice>

<Quantity>1</Quantity>

</Item>

<Item>

<ItemName>Watch</ItemName>

<ItemPrice>2000</ItemPrice>

<Quantity>1</Quantity>

</Item>

<ItemName>Phone</ItemName>

<ItemPrice>20000</ItemPrice>

<Quantity>1</Quantity>

</Item>

<ItemName>Laptop</ItemName>

<ItemPrice>40000</ItemPrice>

<Quantity>1</Quantity>

</Item>

</ItemDetails>

#Use the iris dataset. Write a Python program to view some basic statistical details like percentile, mean, std etc. of the species of 'Iris-setosa', 'Iris-versicolor' and 'Iris-virginica'. Apply logistic regression on the dataset to identify different species (setosa, versicolor, verginica) of Iris flowers given just 4 features: sepal and petal lengths and widths.. Find the accuracy of the model.

import numpy as np

import pandas as pd

from sklearn.datasets import load\_iris

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

from scipy import stats

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

X = iris\_data[['SepalLenghCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]

y = iris\_data['Species']

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

model = LogisticRegression(max\_iter=1000)

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

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

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

print("\nAccuracy of the Logistic Regression model:", accuracy)

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

plt.plot(X\_train, model.predict(X\_train), color='red',linewidth=3)

plt.title('Regression(Salary)')

plt.xlabel('Features')

plt.ylabel('Species')

plt.show()

**Slip no 6**

Write PHP script to read “book.xml” file into simpleXML object. Display attributes and elements . ( simple\_xml\_load\_file() function )

//xml

<?xml version="1.0" encoding="UTF-8"?>

<BookInfo>

<book>

<bookno>1</bookno>

<bookname>java</bookname>

<authorname>Balguru Swami</authorname>

<price>250</price>

<year>2006</year>

</book>

<book>

<bookno>2</bookno>

<bookname>c</bookname>

<authorname>Denic Ritchie</authorname>

<price>500</price>

<year>1971</year>

</book>

</BookInfo>

//php1

<?php

$xml=simplexml\_load\_file("book.xml");

foreach($xml->book as $bk)

{

  echo"Book Number=$bk->bookno<br>";

  echo"Book Name=$bk->bookname<br>";

  echo"AuthorNumber=$bk->authorname<br>";

  echo"price=$bk->price<br>";

  echo"year=$bk->year<br>";

}

?>

#Create the following dataset in python & Convert the categorical values into numeric format.Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules. Repeat the process with different min\_sup values.

import numpy as np

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

transactions = [['Bread', 'Milk'], ['Bread','Diaper','Beer','Eggs'],['Milk','Diaper','Beer','Coke'],['Bread','Milk','Diaper','Beer'],['Bread','Milk','Diaper','Coke']]

from mlxtend.preprocessing import TransactionEncoder

te=TransactionEncoder()

te\_array=te.fit(transactions).transform(transactions)

df=pd.DataFrame(te\_array, columns=te.columns\_)

df = df.astype(int)

df

freq\_items = apriori(df, min\_support = 0.5, use\_colnames = True)

print(freq\_items)

rules = association\_rules(freq\_items, metric ='support', min\_threshold=0.05)

rules = rules.sort\_values(['support', 'confidence'], ascending =[False,False])

rules

**Slip no 7**

//Write a PHP script to read “Movie.xml” file and print all MovieTitle and ActorName of file using DOMDocument Parser. “Movie.xml” file should contain following information with at least 5 records with values. MovieInfoMovieNo, MovieTitle, ActorName ,ReleaseYear

//xml

<?xml version="1.0" encoding="UTF-8"?>

<MovieInfo>

<Movie>

<Movieno>1</Movieno>

<Movietitle>war</Movietitle>

<Actorname>Hritik Roshan</Actorname>

<year>2019</year>

</Movie>

<Movie>

<Movieno>2</Movieno>

<Movietitle>Pathan</Movietitle>

<Actorname>Sharukh Khan</Actorname>

<year>2023</year>

</Movie>

</MovieInfo>

//php1

<?php

$dom = new DomDocument();

$dom->load("movie.xml");

echo "<h2>Names of the Movies and their Actors:</h2>";

$movies = $dom->getElementsByTagName("Movie");

foreach ($movies as $movie) {

    $title = $movie->getElementsByTagName("Movietitle")->item(0)->textContent;

    $actor = $movie->getElementsByTagName("Actorname")->item(0)->textContent;

    echo "<b>Movie Title:</b> $title<br>";

    echo "<b>Actor Name:</b> $actor<br><br>";

}

?>

#Download the Market basket dataset. Write a python program to read the dataset and display its

#information. Preprocess the data (drop null values etc.) Convert the categorical values into numeric format. Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules

import pandas as pd

from mlxtend.frequent\_patterns import apriori,association\_rules

df=pd.read\_csv('CSV/Market\_Basket\_Optimisation.csv')

df=df.sample(50)

df

transactions=[]

for i in range(0,len(df)):

      transactions.append([str(df.values[i,j])for j in range(0,len(df.columns))if(str(df.values[i,j])!='nan')])

transactions

df.dropna(inplace=True,axis=0)

from mlxtend.preprocessing import TransactionEncoder

te=TransactionEncoder()

te\_array=te.fit\_transform(transactions)

te\_array = te\_array.astype('int')

df1=pd.DataFrame(te\_array,columns=te.columns\_)

df1

freq\_items=apriori(df1,min\_support=0.005,use\_colnames=True)

freq\_items

rules=association\_rules(freq\_items,metric='support',min\_threshold=0.005)

rules=rules.sort\_values(['support','confidence'])

rules.tail()

**Slip no  8**

//Write a JavaScript to display message ‘Exams are near, have you started preparing for?’ (usealert box ) and Accept any two numbers from user and display addition of two number .(Use Prompt and confirm box)

<html>

<head>

    <title>Exam Preparation</title>

</head>

<body>

<script>

    var userInputPrompt = confirm('Exams are near, have you started preparing for?');

        if (userInputPrompt == true)

        {

            alert('Great! Keep up the good work!');

        }

    else

        {

            prompt('You should start preparing!');

        }

    var num1 = prompt('Enter num1: ');

    var num2 = prompt('Enter num2: ');

    var addn = Number(num1) + Number(num2);

    alert("Addition of num1 and num2 = " + addn);

</script>

</body>

</html>

//Download the groceries dataset. Write a python program to read the dataset and display its information. Preprocess the data (drop null values etc.) Convert the categorical values into numeric format. Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules.

import pandas as pd

from mlxtend.frequent\_patterns import apriori,association\_rules

df=pd.read\_csv('CSV/groceries.csv')

df=df.sample(50)

df

transactions=[]

for i in range(0,len(df)):

      transactions.append([str(df.values[i,j])for j in range(0,len(df.columns))if(str(df.values[i,j])!='nan')])

transactions

df.dropna(inplace=True,axis=0)

from mlxtend.preprocessing import TransactionEncoder

te=TransactionEncoder()

te\_array=te.fit\_transform(transactions)

te\_array = te\_array.astype('int')

df1=pd.DataFrame(te\_array,columns=te.columns\_)

df1

freq\_items=apriori(df1,min\_support=0.005,use\_colnames=True)

freq\_items

rules=association\_rules(freq\_items,metric='support',min\_threshold=0.005)

rules=rules.sort\_values(['support','confidence'])

rules.tail()

**Slip no 9**

//Write a JavaScript function to validate username and password for a membership form

<script>

    function validateform() {

        var name = document.myform.name.value;

        var password = document.myform.password.value;

        if (name == null || name == "") {

            alert("Name can't be blank");

            return false;

        } else if (password.length < 6) {

            alert("Password must be at least 6 characters long.");

            return false;

        }

    }

</script>

<body>

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

        Name: <input type="text" name="name"><br />

        Password: <input type="password" name="password"><br />

        <input type="submit" value="register">

    </form>

</body>

#Create your own transactions dataset and apply the above process on your dataset.

import pandas as pd

from mlxtend.frequent\_patterns import apriori,association\_rules

transaction=[['sugar','tea'],['coffee','tea','sugar'],['tea','coffee'],['coffee','suagr','tea','milk']]

from mlxtend.preprocessing import TransactionEncoder

te=TransactionEncoder()

te\_array=te.fit(transaction).transform(transaction)

df=pd.DataFrame(te\_array,columns=te.columns\_)

df = df.astype(int)

df

freq\_items=apriori(df,min\_support=0.5,use\_colnames=True)

rules=association\_rules(freq\_items,metric='support',min\_threshold=0.05)

rules=rules.sort\_values(['support','confidence'],ascending=[False,False])

rules

**Slip no 10**

//Create a HTML fileto insert text before and after a Paragraph using jQuery. [Hint : Use before( ) and after( )]

<html>

    <body>

        <p> This is some text. </p>

        <script src="jquery-3.7.1.min.js"></script>

        <script>

            $("p").before("<input type='text' />");

            $("p").after("<input type='text'  />");

        </script>

    </body>

</html>

#Create the following dataset in python & Convert the categorical values into numeric format.Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules. Repeat the process with different min\_sup values.

import numpy as np

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

transactions = [['eggs','milk','bread'],['eggs','apple'],['milk','bread'],['apple', 'milk'],['milk','apple','bread']]

from mlxtend.preprocessing import TransactionEncoder

te=TransactionEncoder()

te\_array=te.fit(transactions).transform(transactions)

df=pd.DataFrame(te\_array, columns=te.columns\_)

df = df.astype(int)

df

freq\_items = apriori(df, min\_support = 0.5, use\_colnames = True)

print(freq\_items)

rules = association\_rules(freq\_items, metric ='support', min\_threshold=0.05)

rules = rules.sort\_values(['support', 'confidence'], ascending =[False,False])

rules

**Slip no 11**

//Write a Javascript program to accept name of student, change font color to red, font size to 18 if student name is present otherwise on clicking on empty text box display image which changes its size (Use onblur, onload, onmousehover, onmouseclick, onmouseup)

<html>

<body>

  <label for="studentName">Enter Student Name:</label>

  <input type="text" id="studentName" onblur="changeStyle()" onmouseover="displayImage()" onmouseout="hideImage()" onclick="changeImageSize()" onmouseup="hideImage()">

  <div id="outputImage">

    <img src="img.png" alt="Image" id="studentImage" onload="changeImageSize()">

  </div>

  <script>

    function changeStyle() {

      var studentNameInput = document.getElementById('studentName');

      if (studentNameInput.value.trim() !== '') {

        studentNameInput.style.color = 'red';

        studentNameInput.style.fontSize = '18px';

      }

    }

    function displayImage() {

      var studentNameInput = document.getElementById('studentName');

      if (studentNameInput.value.trim() === '') {

        var outputImage = document.getElementById('outputImage');

        outputImage.style.display = 'block';

      }

    }

    function hideImage() {

      var outputImage = document.getElementById('outputImage');

      outputImage.style.display = 'none';

    }

    function changeImageSize() {

      var studentImage = document.getElementById('studentImage');

      studentImage.style.width = '200px';

    }

  </script>

</body>

</html>

#Create the following dataset in python & Convert the categorical values into numeric format.Apply the apriori algorithm on the above dataset to generate the frequent itemsets and associationrules. Repeat the process with different min\_sup values.

import numpy as np

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

transactions = [['eggs','milk','bread'],['eggs','apple'],['milk','bread'],['apple', 'milk'],['milk','apple','bread']]

from mlxtend.preprocessing import TransactionEncoder

te=TransactionEncoder()

te\_array=te.fit(transactions).transform(transactions)

df=pd.DataFrame(te\_array, columns=te.columns\_)

df = df.astype(int)

df

freq\_items = apriori(df, min\_support = 0.5, use\_colnames = True)

print(freq\_items)

rules = association\_rules(freq\_items, metric ='support', min\_threshold=0.05)

rules = rules.sort\_values(['support', 'confidence'], ascending =[False,False])

rules= rules.sort\_values(['support', 'confidence'], ascending =[False,False])

**Slip no 12**

//Write AJAX program to read contact.dat file and print the contents of the file in a tabular format when the user clicks on print button. Contact.dat file should contain srno, name, residence number, mobile number, Address.

//html

<html>

<head>

    <script type="text/javascript">

        function print() {

            var ob = false;

            ob = new XMLHttpRequest();

            ob.open("GET", "1.php?");

            ob.send();

            ob.onreadystatechange = function () {

                if (ob.readyState == 4 && ob.status == 200) {

                    document.getElementById("i").innerHTML = ob.responseText;

                }

            }

        }

    </script>

</head>

<body>

    <center>

        <h3>Display the contents of a contact.dat file </h3>

        <br><input type="button" value=Print onclick="print()">

        <span id="i"></span>

    </center>

</body>

</html>

//php

<?php

    $fp = fopen('contacts.dat','r');

    echo "<table border=1>";

    echo "<tr><th>Sr. No.</th><th>Name</th><th>Residence No.</th><th>Mob. no.</th><th>Address</th></tr>";

    while($row =  fscanf($fp,"%s %s %s %s %s"))

    {

        echo "<tr>";

        foreach($row as $r)

        {

            echo "<td>$r</td>";

        }

        echo "</tr>";

    }

    echo "</table>";

    fclose($fp);

?>

#Create ‘heights-and-weights’ Data set . Build a linear regression model by identifying independent and target variable. Split the variables into training and testing sets and print them. Build a simple linear regression model for predicting purchases.

import numpy as np

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

num\_samples = 1000

heights = np.random.normal(170, 10, num\_samples)

weights = 0.5 \* heights + 30 + np.random.normal(0, 5, num\_samples)

data = pd.DataFrame({'Height': heights, 'Weight': weights})

X = data[['Height']]

y = data['Weight']

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

model = LinearRegression()

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

train\_rmse = np.sqrt(mean\_squared\_error(y\_train, model.predict(X\_train)))

test\_rmse = np.sqrt(mean\_squared\_error(y\_test, model.predict(X\_test)))

print("Training RMSE:", train\_rmse)

print("Testing RMSE:", test\_rmse)

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

plt.plot(X\_train, model.predict(X\_train), color='red',linewidth=3)

plt.title('Regression(Height Weight)')

plt.xlabel('Height')

plt.ylabel('Weight')

plt.show()

**Slip no 13**

//Write AJAX program where the user is requested to write his or her name in a text box, and the server keeps sending back responses while the user is typing. If the user name is not entered then the message displayed will be, “Stranger, please tell me your name!”. If the name is Rohit, Virat, Dhoni, Ashwin or Harbhajan , the server responds with “Hello, master !”. If the name is anything else, the message will be “, I don’t know you!”

//html

<html>

<body>

<label for="nameInput">Enter your name:</label>

<input type="text" id="nameInput">

<p id="response">Stranger, please tell me your name!</p>

<script>

document.addEventListener('DOMContentLoaded', function() {

  var nameInput = document.getElementById('nameInput');

  var responseElement = document.getElementById('response');

  function checkName() {

    var name = nameInput.value.trim();

    if (name === '') {

      responseElement.textContent = 'Stranger, please tell me your name!';

      return;

    }

    var xhr = new XMLHttpRequest();

    xhr.open('POST', '1.php', true);

    xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

    xhr.onreadystatechange = function() {

      if (xhr.readyState === 4 && xhr.status === 200) {

        responseElement.textContent = xhr.responseText;

      }

    };

    xhr.send('name=' + encodeURIComponent(name));

  }

  nameInput.addEventListener('input', function() {

    checkName();

  });

});

</script>

</body>

</html>

//1.php

<?php

if (isset($\_POST['name'])) {

    $name = $\_POST['name'];

    if (empty($name)) {

        echo 'Stranger, please tell me your name!';

    } elseif (in\_array($name, ['Rohit', 'Virat', 'Dhoni', 'Ashwin', 'Harbhajan'])) {

        echo 'Hello, master!';

    } else {

        echo "I don't know you!";

    }

} else {

    echo 'Invalid request';

}

?>

#download nursery dataset from UCI. Build a linear regression model by identifying independent

#d target variable. Split the variables into training and testing sets and print them. Build a simple linear regression model for predicting purchases.

import numpy as np

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

nursery\_data = pd.read\_csv("CSV/nursery.csv")

X = nursery\_data[['social']]

y = nursery\_data[['health']]

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

model = LinearRegression()

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

train\_rmse = np.sqrt(mean\_squared\_error(y\_train, model.predict(X\_train)))

test\_rmse = np.sqrt(mean\_squared\_error(y\_test, model.predict(X\_test)))

print("Training RMSE:", train\_rmse)

print("Testing RMSE:", test\_rmse)

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

plt.plot(X\_train, model.predict(X\_train), color='red',linewidth=3)

plt.title('Regression(Height Weight)')

plt.xlabel('Height')

plt.ylabel('Weight')

plt.show()

**Slip no 14**

//Create TEACHER table as follows TEACHER(tno, tname, qualification, salary). Write Ajax program to select a teachers name and print the selected teachers details

//html

<html>

<script>

    function dis() {

        var ob = false;

        ob = new XMLHttpRequest();

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

        ob.open("GET", "1.php?tname=" + name);

        ob.send();

        ob.onreadystatechange = function () {

            if (ob.readyState == 4 && ob.status == 200) {

                document.getElementById("snamed").innerHTML = ob.responseText;

            }

        }

    }

</script>

<body>

    Enter teacher name Name<input type="text" name=tname id=tname>

    <input type="submit" value=submit>

    <span id="snamed"></span>

</body>

</html>

//php

<?php

$tname=$\_GET['tname'];

$db = pg\_connect("host=localhost dbname= user= password= ");

$query = "select \*from teacher where tname='$tname'";

$result=pg\_query($db,$query);

while($row=pg\_fetch\_array($result))

{

    echo $row['tno']."--".$row['tname']."--".$row['qualifications']."--".$row['salary']."<br>";

}

?>

#Create the following dataset in python & Convert the categorical values into numeric format.Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules. Repeat the process with different min\_sup values.

import numpy as np

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

transactions = [['eggs','milk','bread'],['eggs','apple'],['milk','bread'],['apple', 'milk'],['milk','apple','bread']]

from mlxtend.preprocessing import TransactionEncoder

te=TransactionEncoder()

te\_array=te.fit(transactions).transform(transactions)

df=pd.DataFrame(te\_array, columns=te.columns\_)

df = df.astype(int)

df

freq\_items = apriori(df, min\_support = 0.5, use\_colnames = True)

print(freq\_items)

rules = association\_rules(freq\_items, metric ='support', min\_threshold=0.05)

rules = rules.sort\_values(['support', 'confidence'], ascending =[False,False])

print(rules)

**Slip no 15**

//Write Ajax program to fetch suggestions when is user is typing in a textbox. (eg like google suggestions. Hint create array of suggestions and matching string will be displayed)

//html

<html>

<head>

    <script type="text/javascript">

        function m1(str) {

            var ob = false;

            ob = new XMLHttpRequest();

            ob.open("GET", "1.php?q=" + str);

            ob.send();

            ob.onreadystatechange = function () {

                if (ob.readyState == 4 && ob.status == 200) {

                    document.getElementById("a").innerHTML = ob.responseText;

                }

            }

        }

    </script>

</head>

<body>

    <form>

        Search<input type=text name=search size="20" onkeyup="m1(form.search.value)">

        <input type=button value="submit" onclick="m1(form.search.value)">

    </form>

    suggestions :<span id="a"></span><br>

</body>

</html>

//php

<?php

$a=array("pune","satara","nashik","sangli","mumbai","murud","akola","dound","dhule","ratnagiri","rajpur");

$q=$\_GET['q'];

if(strlen($q)>0)

{

    $match="";

    for($i=0;$i<count($a);$i++)

    {

    if(strtolower($q)==strtolower(substr($a[$i],0,strlen($q))))

    {

        if($match=="")

            $match=$a[$i];

        else

            $match=$match.",".$a[$i];

    }

    }

if($match=="")

    echo "No Suggestios";

else

    echo $match;

}

?>

#Create the following dataset in python & Convert the categorical values into numeric format.Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules. Repeat the process with different min\_sup values

import numpy as np

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

transactions = [['eggs','milk','bread'],['eggs','apple'],['milk','bread'],['apple', 'milk'],['milk','apple','bread']]

from mlxtend.preprocessing import TransactionEncoder

te=TransactionEncoder()

te\_array=te.fit(transactions).transform(transactions)

df=pd.DataFrame(te\_array, columns=te.columns\_)

df = df.astype(int)

df

freq\_items = apriori(df, min\_support = 0.5, use\_colnames = True)

print(freq\_items)

rules = association\_rules(freq\_items, metric ='support', min\_threshold=0.05)

rules = rules.sort\_values(['support', 'confidence'], ascending =[False,False])

**Slip no 16**

//Write Ajax program to get book details from XML file when user select a book name. Create XML file for storing details of book(title, author, year, price).

//html

<html>

<body>

<select id="bookSelect" onchange="titles()">

    <option value="">Select a book</option>

</select>

<div id="bookDetails"></div>

<script>

function titles() {

    var title = document.getElementById("bookSelect").value;

    var xhttp = new XMLHttpRequest();

    xhttp.onreadystatechange = function() {

        if (this.readyState == 4 && this.status == 200) {

            document.getElementById("bookDetails").innerHTML = this.responseText;

        }

    };

    xhttp.open("GET", "1.php?title=" + title, true);

    xhttp.send();

}

window.onload = function() {

    var xhttp = new XMLHttpRequest();

    xhttp.onreadystatechange = function() {

        if (this.readyState == 4 && this.status == 200) {

            document.getElementById("bookSelect").innerHTML = this.responseText;

        }

    };

    xhttp.open("GET", "2.php", true);

    xhttp.send();

};

</script>

</body>

</html>

//1.php

<?php

if(isset($\_GET['title'])) {

    $selectedTitle = $\_GET['title'];

    $xml = simplexml\_load\_file('books.xml');

    $bookDetails = "";

    foreach($xml->children() as $book) {

        if($book->title == $selectedTitle) {

            $bookDetails .= "<p><strong>Title:</strong> " . $book->title . "</p>";

            $bookDetails .= "<p><strong>Author:</strong> " . $book->author . "</p>";

            $bookDetails .= "<p><strong>Year:</strong> " . $book->year . "</p>";

            $bookDetails .= "<p><strong>Price:</strong> $" . $book->price . "</p>";

        }

    }

    echo $bookDetails;

}

?>

//2.php

<?php

$xml = simplexml\_load\_file('books.xml');

$bookOptions = "<option value=''>Select a book</option>";

foreach($xml->children() as $book) {

    $title = $book->title;

    $bookOptions .= "<option value='$title'>$title</option>";

}

echo $bookOptions;

?>

#Consider any text paragraph. Preprocess the text to remove any special characters and digits. Generate the summary using extractive summarization process

import re

from nltk.tokenize import sent\_tokenize

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

text="Text summarization 09 is an NLP technique that extracts@ text , from a large amount of data .Its is a process of identifying the most important meaningful information in a document."

def preprocess\_text(text):

    text = re.sub(r'[^a-zA-Z\s]', '', text)

    text = re.sub(r'\d+', '', text)

    return text.lower()

def tokenize\_sentences(text):

    return sent\_tokenize(text)

preprocessed\_text = preprocess\_text(text)

sentences = tokenize\_sentences(preprocessed\_text)

stop\_words = set(stopwords.words("english"))

stemmer = PorterStemmer()

def preprocess\_sentence(sentence):

    words = sentence.split()

    words = [stemmer.stem(word) for word in words if word not in stop\_words]

    return ' '.join(words)

preprocessed\_sentences = [preprocess\_sentence(sentence) for sentence in sentences]

vectorizer = TfidfVectorizer()

tfidf\_matrix = vectorizer.fit\_transform(preprocessed\_sentences)

cosine\_sim\_matrix = cosine\_similarity(tfidf\_matrix, tfidf\_matrix)

sentence\_scores = cosine\_sim\_matrix.sum(axis=1)

sorted\_indices = sentence\_scores.argsort()[::-1]

num\_sentences\_summary = 2

summary\_sentences = [sentences[idx] for idx in sorted\_indices[:num\_sentences\_summary]]

summary = ' '.join(summary\_sentences)

print("Original Text:\n", text)

print("\nExtractive Summary:\n", summary)

**Slip no 17**

//Write a Java Script Program to show Hello Good Morning message onload event using alert box and display the Student registration form.

<html>

    <body onload="alert('Hello Good Morning')">

        <form>

            Name : <input type="text"><br>

            Class: <input type="text"><br>

            Roll no: <input type="text"><br>

            <input type="submit" value="SUBMIT">

        </form>

    </body>

</html>

#Consider text paragraph.So, keep working. Keep striving. Never give up. Fall down seven times, get up eight. Ease is a greater threat to progress than hardship. Ease is a greater threat to progress than hardship. So, keep moving, keep growing, keep learning. See you at work.Preprocess the text to remove any special characters and digits. Generate the summary using extractive summarization process.

import re

from nltk.tokenize import sent\_tokenize

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

text = "So, keep working. Keep striving. Never give up. Fall down seven times, get up eight. Ease is a greater threat to progress than hardship. Ease is a greater threat to progress than hardship. So, keep moving, keep growing, keep learning. See you at work."

def preprocess\_text(text):

    text = re.sub(r'[^a-zA-Z\s]', '', text)

    text = re.sub(r'\d+', '', text)

    return text.lower()

def tokenize\_sentences(text):

    return sent\_tokenize(text)

preprocessed\_text = preprocess\_text(text)

sentences = tokenize\_sentences(preprocessed\_text)

stop\_words = set(stopwords.words("english"))

stemmer = PorterStemmer()

def preprocess\_sentence(sentence):

    words = sentence.split()

    words = [stemmer.stem(word) for word in words if word not in stop\_words]

    return ' '.join(words)

preprocessed\_sentences = [preprocess\_sentence(sentence) for sentence in sentences]

vectorizer = TfidfVectorizer()

tfidf\_matrix = vectorizer.fit\_transform(preprocessed\_sentences)

cosine\_sim\_matrix = cosine\_similarity(tfidf\_matrix, tfidf\_matrix)

sentence\_scores = cosine\_sim\_matrix.sum(axis=1)

sorted\_indices = sentence\_scores.argsort()[::-1]

num\_sentences\_summary = 2

summary\_sentences = [sentences[idx] for idx in sorted\_indices[:num\_sentences\_summary]]

summary = ' '.join(summary\_sentences)

print("Original Text:\n", text)

print("\nExtractive Summary:\n", summary)

**Slip no 18**

//Write a Java Script Program to print Fibonacci numbers on onclick event.

<html>

<head>

    <title>Fibonacci Numbers</title>

    <script>

        function fibo() {

            var num = parseInt(prompt("Enter the number of Fibonacci numbers to generate: "));

            if (isNaN(num) || num <= 0) {

                alert("Please enter a valid positive number.");

                return;

            }

            var fib = [];

            fib.push(0);

            fib.push(1);

            for (var i = 2; i < num; i++) {

                fib.push(fib[i - 1] + fib[i - 2]);

            }

            document.getElementById("p1").innerHTML = "Fibonacci Numbers: " + fib.join(", ");

        }

    </script>

</head>

<body>

    <button onclick="fibo()">Generate Fibonacci Numbers</button>

    <p id="p1"></p>

</body>

</html>

#Consider any text paragraph. Remove the stopwords. Tokenize the paragraph to extract words and sentences. Calculate the word frequency distribution and plot the frequencies. Plot the wordcloud of the text.

import re

import matplotlib.pyplot as plt

from wordcloud import WordCloud

from nltk.tokenize import word\_tokenize, sent\_tokenize

from nltk.corpus import stopwords

from collections import Counter

text = """

Hello world this is 4 and Here to summarize text

"""

stop\_words = set(stopwords.words('english'))

filtered\_text = ' '.join([word for word in re.findall(r'\b\w+\b', text.lower()) if word not in stop\_words])

words = word\_tokenize(filtered\_text)

sentences = sent\_tokenize(text)

word\_freq = Counter(words)

plt.figure(figsize=(10, 6))

plt.bar(word\_freq.keys(), word\_freq.values())

plt.xlabel('Words')

plt.ylabel('Frequency')

plt.title('Word Frequency Distribution')

plt.xticks(rotation=45)

plt.show()

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(filtered\_text)

plt.figure(figsize=(10, 6))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.title('Wordcloud')

plt.show()

**Slip no 19**

//Write a Java Script Program to validate user name and password on onSubmit event

<html>

<head>

    <title>User Authentication</title>

    <script>

        function validateForm() {

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

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

            if (username.trim() === "") {

                alert("Please enter a username.");

                return false;

            }

            if (password.trim() === "") {

                alert("Please enter a password.");

                return false;

            }

            if (password.length < 6) {

                alert("Password must be at least 6 characters long.");

                return false;

            }

            return true;

        }

    </script>

</head>

<body>

    <h2>User Authentication</h2>

    <form onsubmit="return validateForm()">

        <label for="username">Username:</label><br>

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

        <label for="password">Password:</label><br>

        <input type="password" id="password" name="password"><br><br>

        <input type="submit" value="Submit">

    </form>

</body>

</html>

#Download the movie\_review.csv dataset from Kaggle by using the following link :https://www.kaggle.com/nltkdata/movie-review/version/3?select=movie\_review.csv to perform sentiment analysis on above dataset and create a wordcloud

import pandas as pd

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

from nltk.stem import WordNetLemmatizer

from wordcloud import WordCloud

import matplotlib.pyplot as plt

nltk.download('wordnet')

df = pd.read\_csv('CSV/movie\_review.csv')

df

stop\_words = set(stopwords.words('english'))

lemmatizer = WordNetLemmatizer()

def preprocess\_text(text):

    words = word\_tokenize(text)

    words = [word.lower() for word in words if word.isalpha()]

    words = [lemmatizer.lemmatize(word) for word in words if word not in stop\_words]

    return ' '.join(words)

df['clean\_text'] = df['text'].apply(preprocess\_text)

all\_text = ' '.join(df['clean\_text'])

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(all\_text)

plt.figure(figsize=(10, 6))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.title('Word Cloud of Movie Reviews')

plt.show()

**Slip no 20**

//create a student.xml file containing at least 5 student information

<?xml version="1.0" encoding="UTF-8"?>

<Students>

    <Student>

        <name>s1</name>

        <rno>1</rno>

    </student>

</Students>

#Consider text paragraph."""Hello all, Welcome to Python Programming Academy. Python

#Programming Academy is a nice platform to learn new programming skills. It is difficult to get enrolled in this Academy."""Remove the stopwords.

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

text = """Hello all, Welcome to Python Programming Academy. Python Programming Academy is a nice platform to learn new programming skills. It is difficult to get enrolled in this Academy."""

nltk.download('stopwords')

nltk.download('punkt')

words = word\_tokenize(text)

stop\_words = set(stopwords.words('english'))

filtered\_words = [word for word in words if word.lower() not in stop\_words]

filtered\_text = ' '.join(filtered\_words)

print("Original Text:\n", text)

print("\nText after removing stopwords:\n", filtered\_text)

**Slip no 21**

//Add a JavaScript File in Codeigniter. The Javascript code should check whether a number is positive or negative.

welcome\_message.php/

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

$this->load->helper('url');

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Q1</title>

</head>

<body>

    <input type="text" id="input" />

    <p id="output"></p>

    <button id="btn">check</button>

<script src="<?php echo base\_url('public/js/new.js'); ?>">

</script>

</body>

</html>

public/

....js/

........new.js/

let button = document.getElementById("btn");

let input = document.getElementById("input");

let output = document.getElementById("output");

button.addEventListener("click", function () {

    let current = +input.value;

    if (current < 0) {

        output.innerText = "Negative";

    } else if (current > 0) {

        output.innerText = "Positive";

    } else {

        output.innerText = "Zero";

    }

});

#Build a simple linear regression model for User Data.

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

from sklearn.linear\_model import LinearRegression

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('./csv/User\_Data.csv')

df = pd.get\_dummies(df, columns=['Gender'], drop\_first=True)

X = df[['EstimatedSalary']]

y = df['Purchased']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=15)

model = LinearRegression()

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

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

plt.plot(X\_train, model.predict(X\_train), color='red',linewidth=3)

plt.title('Linear Regression')

plt.xlabel('Salary')

plt.ylabel('Purchases')

plt.show()

**Slip no 22**

//Create a table student having attributes(rollno, name, class). Using codeigniter, connect to the database and insert 5 recodes in it.

view/welcome\_message.php/

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

$this->load->helper('url');

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Q2</title>

</head>

<body>

    <?php echo validation\_errors(); ?>

    <?php echo form\_open('welcome/add'); ?>

        <label>Rollno: </label>

        <input type="number" name="rollno">

        <label>Name: </label>

        <input type="text" name="name">

        <label>Class: </label>

        <input type="text" name="class">

        <button>submit</button>

    </form>

</body>

</html>

controller/Welcome.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Welcome extends CI\_Controller {

    public function \_\_construct() {

        parent::\_\_construct();

        $this->load->library('form\_validation');

        $this->load->model('Model');

        $this->load->helper('url');

    }

    public function index() {

        $this->load->view('welcome\_message.php');

    }

    public function add() {

        $this->form\_validation->set\_rules('rollno', 'Rollno', 'required');

        $this->form\_validation->set\_rules('name', 'Name', 'required');

        $this->form\_validation->set\_rules('class', 'Class', 'required');

        if ($this->form\_validation->run() == FALSE) {

            $this->load->view('welcome\_message');

        } else {

            $data['rollno'] = $this->input->post('rollno');

            $data['name'] = $this->input->post('name');

            $data['class'] = $this->input->post('class');

            $this->Model->save\_form\_data($data);

            redirect('welcome');

        }

    }

}

models/Model.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Model extends CI\_Model {

    public function save\_form\_data($data) {

        $this->db->insert('student', $data);

    }

}

config/autoload.php

$autoload['libraries'] = array('database');

#Consider any text paragraph. Remove the stopwords.

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

text = """Hello all, Welcome to Python Programming Academy. Hello all, Welcome to Python Programming Academy. Hello all, Welcome to Python Programming Academy. Hello all, Welcome to Python Programming Academy. Hello all, Welcome to Python Programming Academy. Hello all, Welcome to Python Programming Academy. Python Programming Academy is a nice platform to learn new programming skills. It is difficult to get enrolled in this Academy."""

nltk.download('stopwords')

nltk.download('punkt')

words = word\_tokenize(text)

stop\_words = set(stopwords.words('english'))

filtered\_words = [word for word in words if word.lower() not in stop\_words]

filtered\_text = ' '.join(filtered\_words)

print("Original Text:\n", text)

print("\nText after removing stopwords:\n", filtered\_text)

**Slip no 23**

//Create a table student having attributes(rollno, name, class) containing atleast 5 recodes. Using codeigniter, display all its records.

views/welcome\_message.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

$this->load->helper('url');

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Q2</title>

</head>

<body>

    <?php echo validation\_errors(); ?>

    <?php echo form\_open('welcome/add'); ?>

        <label>Rollno: </label>

        <input type="number" name="rollno">

        <label>Name: </label>

        <input type="text" name="name">

        <label>Class: </label>

        <input type="text" name="class">

        <button>submit</button>

    </form>

    <br>

    <?php echo form\_open('welcome/show'); ?>

        <button>view data</button>

    </form>

</body>

</html>

views/data\_view.php

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="utf-8">

    <title>Uploaded Data</title>

</head>

<body>

<h2>Data</h2>

<?php if (empty($records)): ?>

    <p>No data available.</p>

<?php else: ?>

    <table border="1">

        <tr><th>Roll</th><th>Name</th><th>Class</th></tr>

        <?php foreach ($records as $record): ?>

            <tr>

                <td><?php echo $record['rollno']; ?></td>

                <td><?php echo $record['name']; ?></td>

                <td><?php echo $record['class']; ?></td>

            </tr>

        <?php endforeach; ?>

    </table>

<?php endif; ?>

</body>

</html>

controllers/Welcom.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Welcome extends CI\_Controller {

public function \_\_construct() {

        parent::\_\_construct();

        $this->load->library('form\_validation');

        $this->load->model('Model');

        $this->load->helper('url');

    }

    public function index() {

        $this->load->view('welcome\_message.php');

    }

public function add() {

        $this->form\_validation->set\_rules('rollno', 'Rollno', 'required');

        $this->form\_validation->set\_rules('name', 'Name', 'required');

        $this->form\_validation->set\_rules('class', 'Class', 'required');

        if ($this->form\_validation->run() == FALSE) {

            $this->load->view('welcome\_message');

        } else {

            $data['rollno'] = $this->input->post('rollno');

            $data['name'] = $this->input->post('name');

            $data['class'] = $this->input->post('class');

            $this->Model->save\_form\_data($data);

            redirect('welcome');

        }

    }

    public function show() {

        $data['records'] = $this->Model->get\_uploaded\_data();

        $this->load->view('data\_view', $data);

    }

}

models/Model.php

<?php

defined('BASEPATH') OR exit('No direct script access allowed');

class Model extends CI\_Model {

    public function save\_form\_data($data) {

        $this->db->insert('student', $data);

    }

    public function get\_uploaded\_data() {

        $query = $this->db->get('student');

        return $query->result\_array();

    }

}

#Consider any text paragraph. Preprocess the text to remove any special characters and digits.

import re

text="hello 1234 this is @"

def preprocess\_text(text):

    text = re.sub(r'[^a-zA-Z\s]', '', text)

    text = re.sub(r'\d+', '', text)

    return text.lower()

preprocessed\_text = preprocess\_text(text)

print("Original Text:\n", text)

print("\nAfter processing text:\n", preprocessed\_text)

**Slip no 24**

//Write a PHP script to create student.xml file which contains student roll no, name, address, college and course. Print students detail of specific course in tabular format after accepting course as input.

<?php

$course = $\_GET['course'];

$xml = simplexml\_load\_file('student.xml');

if ($xml) {

    $students = array();

    foreach ($xml->student as $student) {

        $studentDetails = array(

            'roll\_no' => (string)$student->roll\_no,

            'name' => (string)$student->name,

            'address' => (string)$student->address,

            'college' => (string)$student->college,

            'course' => (string)$student->course

        );

        $students[] = $studentDetails;

    }

    function printStudents($students) {

        echo "<table border='1'>

                <tr>

                    <th>Roll No</th>

                    <th>Name</th>

                    <th>Address</th>

                    <th>College</th>

                    <th>Course</th>

                </tr>";

        foreach ($students as $student) {

            echo "<tr>

                    <td>{$student['roll\_no']}</td>

                    <td>{$student['name']}</td>

                    <td>{$student['address']}</td>

                    <td>{$student['college']}</td>

                    <td>{$student['course']}</td>

                  </tr>";

        }

        echo "</table>";

    }

    function filterStudentsByCourse($students, $course) {

        $filteredStudents = array();

        foreach ($students as $student) {

            if ($student['course'] == $course) {

                $filteredStudents[] = $student;

            }

        }

        return $filteredStudents;

    }

    if (!empty($course)) {

        $filteredStudents = filterStudentsByCourse($students, $course);

        if (!empty($filteredStudents)) {

            echo "<h2>Students in $course</h2>";

            printStudents($filteredStudents);

        } else {

            echo "<p>No students found in $course.</p>";

        }

    }

} else {

    echo "Failed to load student.xml file.";

}

?>

#Consider the following dataset : https://www.kaggle.com/datasets/datasnaek/youtube-new?select=INvideos.csv Write a Python script for the following : i. Read the dataset and perform data cleaning operations on it. ii. Find the total views, total likes, total dislikes and comment count.

import pandas as pd

import matplotlib.pyplot as plt

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv("CSV/INvideos.csv")

data.dropna(inplace=True)

total\_views = data['views'].sum()

total\_likes = data['likes'].sum()

total\_dislikes = data['dislikes'].sum()

total\_comments = data['comment\_count'].sum()

print("Total Views:", total\_views)

print("Total Likes:", total\_likes)

print("Total Dislikes:", total\_dislikes)

print("Total Comments:", total\_comments)

least\_liked\_video = data.loc[data['likes'].idxmin()]

top\_liked\_video = data.loc[data['likes'].idxmax()]

least\_commented\_video = data.loc[data['comment\_count'].idxmin()]

top\_commented\_video = data.loc[data['comment\_count'].idxmax()]

**Slip no 25**

Write a script to create “cricket.xml” file with multiple elements as shown below: Write a script to add multiple elements in “cricket.xml” file of category, country=”India”.

<?php

function createCricketXML() {

    $doc = new DOMDocument();

    $cricketTeam = $doc->createElement('CricketTeam');

    $doc->appendChild($cricketTeam);

    $teamAustralia = $doc->createElement('Team');

    $teamAustralia->setAttribute('country', 'Australia');

    $cricketTeam->appendChild($teamAustralia);

    $playerAustralia = $doc->createElement('player', 'Player\_Aus');

    $teamAustralia->appendChild($playerAustralia);

    $runsAustralia = $doc->createElement('runs', '100');

    $teamAustralia->appendChild($runsAustralia);

    $wicketsAustralia = $doc->createElement('wicket', '5');

    $teamAustralia->appendChild($wicketsAustralia);

    $doc->formatOutput = true;

    $doc->save('cricket.xml');

}

function addElementsForIndia() {

    $doc = new DOMDocument();

    $doc->load('cricket.xml');

    $cricketTeam = $doc->documentElement;

    $teamIndia = $doc->createElement('Team');

    $teamIndia->setAttribute('country', 'India');

    $cricketTeam->appendChild($teamIndia);

    $players = array('Player1', 'Player2', 'Player3');

    $runs = array(50, 60, 70);

    $wickets = array(2, 3, 1);

    foreach ($players as $key => $player) {

        $playerIndia = $doc->createElement('player', $player);

        $teamIndia->appendChild($playerIndia);

        $runsIndia = $doc->createElement('runs', $runs[$key]);

        $teamIndia->appendChild($runsIndia);

        $wicketsIndia = $doc->createElement('wicket', $wickets[$key]);

        $teamIndia->appendChild($wicketsIndia);

    }

    $doc->formatOutput = true;

    $doc->save('cricket.xml');

}

if (!file\_exists('cricket.xml')) {

    createCricketXML();

}

addElementsForIndia();

echo "Elements added successfully to cricket.xml";

?>

#Consider the following dataset : https://www.kaggle.com/datasets/seungguini/youtube-comments-

#for-covid19-relatedvideos?select=covid\_2021\_1.csv Write a Python script for the following : i. Read the dataset and perform data cleaning operations on it. ii. Tokenize the comments in words. iii. Perform sentiment analysis and find the percentage of positive, negative and neutral comments..

import pandas as pd

import re

from textblob import TextBlob

data=pd.read\_csv('CSV/covid\_2021\_1.csv')

data=data.dropna(subset=['comment\_text'])

data['clean\_comment']=data['comment\_text'].apply(lambda x:re.sub(r'[^a-zA-Z\s]', '', str(x)))

data['clean\_comment']=data['comment\_text'].apply(lambda x:re.sub(r'\s+','',str(x)))

data['tokenized\_comment']=data['clean\_comment'].apply(lambda x:x.split())

positive\_comments=0

negative\_comments=0

neutral\_comments=0

for comment in data['clean\_comment']:

   analysis=TextBlob(comment)

   if analysis.sentiment.polarity > 0:

     positive\_comments+=1

   elif analysis.sentiment.polarity <0:

     negative\_comments+=1

   else:

     neutral\_comments+=1

total\_comments=len(data)

ps\_per=(positive\_comments/total\_comments)\*100

neg\_per=(negative\_comments/total\_comments)\*100

neut\_per=(positive\_comments/total\_comments)\*100

print("percentage of positive comment: ",format(ps\_per))

print("percentage of negative comment: ",format(neg\_per))

print("percentage of neutral comment: ",format(neut\_per))