**Slips1**

**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"]))

{

    $val=$\_SESSION["count"];

    $val++;

    $\_SESSION["count"]=$val;

    echo(" visit count=".$val);

}

else{

    $\_SESSION["count"]="1";

    echo"welcome 1st time visit";

}

?>

**2)      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

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

X = dataset.iloc[:, 1:-1].values

y = dataset.iloc[:, -1].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

print("Training set:")

print("X\_train:", X\_train)

print("y\_train:", y\_train)

print("Testing set:")

print("X\_test:", X\_test)

print("y\_test:", y\_test)

**Slips2**

1)      **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).

   <?php

// Check if form has been submitted

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

  // Set cookie with user preferences

  setcookie('font\_style', $\_POST['font\_style'], time() + (86400 \* 30), "/");

  setcookie('font\_size', $\_POST['font\_size'], time() + (86400 \* 30), "/");

  setcookie('font\_color', $\_POST['font\_color'], time() + (86400 \* 30), "/");

  setcookie('bg\_color', $\_POST['bg\_color'], time() + (86400 \* 30), "/");

  // Redirect to next page to display selected settings

  header('Location: selected\_settings.php');

  exit;

}

?>

<!DOCTYPE html>

<html>

<head>

  <title>Change Page Preferences</title>

</head>

<body>

  <h1>Change Page Preferences</h1>

  <form method="POST">

    <label>Font Style:</label>

    <select name="font\_style">

      <option value="Arial">Arial</option>

      <option value="Times New Roman">Times New Roman</option>

      <option value="Verdana">Verdana</option>

    </select>

    <br>

    <label>Font Size:</label>

    <select name="font\_size">

      <option value="12px">12px</option>

      <option value="14px">14px</option>

      <option value="16px">16px</option>

    </select>

    <br>

**2)      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 pandas as pd

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

from sklearn.linear\_model import LinearRegression

df = pd.DataFrame({

    'Position': ['Software Engineer', 'Product Manager', 'Data Scientist', 'Sales Executive', 'Marketing Manager', 'Intern'],

    'Level': [1, 2, 3, 4, 5, 6],

    'Salary': [5000, 8000, 11000, 15000, 20000, 25000]

})

X = df[['Level']]

y = df['Salary']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

lr = LinearRegression()

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

print('Training set:')

print('X\_train:', X\_train)

print('y\_train:', y\_train)

print('Testing set:')

print('X\_test:', X\_test)

print('y\_test:', y\_test)

**Slips3**

**1)      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]**

<html>

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

        enter userid:

        <input type="TExt" name="u"><br>

        enter password:

        <input type="TExt" name="p"><br>

        <input type="submit" value="login"><br>

    </form>

</html>

<?php

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

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

session\_start();

if(isset($\_SESSION[$u]))

{

    $val=$\_SESSION[$u];

    $val++;

    $\_SESSION[$u]=$val;

}

else{

    $\_SESSION[$u]=1;

}

if($\_SESSION[$u]>4)

die("attempt is over");

if($p==123)

echo"welcome";

else

echo"invalid info";

?>

**2)      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.**

  import pandas as pd

import numpy as np

user\_data = {'User ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

             'Age': [19, 35, 26, 27, 19, 32, 25, 29, 34, 25],

             'Estimated Salary': [19000, 20000, 43000, 57000, 76000, 58000, 84000, 15000, 43000, 22000],

             'Purchased': [0, 0, 0, 0, 0, 1, 1, 0, 1, 0]}

user\_df = pd.DataFrame(user\_data)

X = user\_df.iloc[:, 1:-1].values

y = user\_df.iloc[:, -1].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)

from sklearn.linear\_model import LogisticRegression

logistic\_model = LogisticRegression()

logistic\_model.fit(X\_train, y\_train)

y\_pred = logistic\_model.predict(X\_test)

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Precision:", precision\_score(y\_test, y\_pred))

print("Recall:", recall\_score(y\_test, y\_pred))

print("F1-score:", f1\_score(y\_test, y\_pred))

**Slips4**

**1)      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]**

page1.html

<html>

    <body>

        <form method="post" action="page2.php">

            enetr emp no:

            <input type="text" name="no"><br>

            enter name:

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

            enetr adrrsess:

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

            <input type="submit" value="next"><br>

        </form>

    </body>

</html>

page2.php

<html>

    <body>

        <form method="post" action="page3.php">

            enetr basic sal:

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

            enter DA:

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

            enetr HRA:

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

            <input type="submit" value="next"><br>

        </form>

    </body>

</html>

<?php

session\_start();

$\_SESSION["no"]=$\_POST['no'];

$\_SESSION["name"]=$\_POST["name"];

$\_SESSION["add"]=$\_POST["add"];

?>

page3.php

<?php

session\_start();

$no= $\_SESSION['no'];

$name= $\_SESSION["name"];

$address= $\_SESSION["add"];

$basic=$\_POST["bsal"];

$da=$\_POST["da"];

$hra=$\_POST["hra"];

echo"emp no=".$no."<br>name=".$name."<br>address=".$address.

"<br> basic sal=".$basic."<br>DA=".$da."<BR>HRA=".$hra;

?>

**2)      Build a simple linear regression model for Fish Species Weight Prediction.**

import pandas as pd

from sklearn.linear\_model import LinearRegression

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

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

X = fish\_data.iloc[:, 1:2].values

y = fish\_data.iloc[:, 2].values

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

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

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

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

r2 = r2\_score(y\_test, y\_pred)

print("Mean squared error:", mse)

print("R2 score:", r2)

new\_data = [[30]]

prediction = regressor.predict(new\_data)

print("Predicted weight:", prediction)

**Slips5**

**1)      Create XML file named “Item.xml”with item-name, item-rate, item quantity Store the details of 5 Items of different Type**

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

<item>

<items>

<itemname>pen</itemname>

<rate>10</rate>

<quantity>2</quantity>

</items>

<items>

<itemname>pencil</itemname>

<rate>5</rate>

<quantity>6</quantity>

</items>

<items>

<itemname>book</itemname>

<rate>100</rate>

<quantity>2</quantity>

</items>

<items>

<itemname>scale</itemname>

<rate>10</rate>

<quantity>87</quantity>

</items>

<items>

<itemname>notes</itemname>

<rate>0976</rate>

<quantity>2</quantity>

</items>

</item>

**2)      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 pandas as pd

from sklearn.datasets import load\_iris

iris = load\_iris()

iris\_df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

iris\_df['target'] = iris.target

iris\_df['target'] = iris\_df['target'].apply(lambda x: iris.target\_names[x])

print(iris\_df[iris\_df['target'] == 'setosa'].describe())

print(iris\_df[iris\_df['target'] == 'versicolor'].describe())

print(iris\_df[iris\_df['target'] == 'virginica'].describe())

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

iris = load\_iris()

X = iris.data

y = iris.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = LogisticRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

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

print('Accuracy:', accuracy)

**Slips6**

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

book.xml

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

<book>

<info>

<bno>1</bno>

<name>c</name>

<author>xyz</author>

</info>

<info>

<bno>2</bno>

<name>php</name>

<author>abc</author>

</info>

<info>

<bno>3</bno>

<name>java</name>

<author>pqr</author>

</info>

</book>

book.php

<?php

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

var\_dump($ob);

?>

**2)      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 pandas as pd

from mlxtend.preprocessing import TransactionEncoder

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

data = [['Milk', 'Egg', 'Bread'],

        ['Milk', 'Bread'],

        ['Milk', 'Egg', 'Bread', 'Cheese'],

        ['Milk', 'Egg'],

        ['Bread', 'Cheese']]

te = TransactionEncoder()

te\_ary = te.fit\_transform(data)

tid = pd.DataFrame(te\_ary, columns=te.columns\_)

items = tid.astype('int')

items = items.replace({True: 1, False: 0})

min\_sup\_values = [0.4, 0.6, 0.8]

for min\_sup in min\_sup\_values:

    frequent\_itemsets = apriori(items, min\_support=min\_sup, use\_colnames=True)

    rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.7)

    print('Min support:', min\_sup)

    print('Frequent itemsets:')

    print(frequent\_itemsets)

    print('Association rules:')

    print(rules)

    print()

**Slips7**

**1)      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 version="1.0" encoding="UTF-8"?>

<movies>

<movie>

<mno>65464</mno>

<title>dilwale</title>

<actor>varun</actor>

<year>2015</year>

</movie>

</movies>

<?php

$doc = new DOMDocument();

$doc->load('movie.xml');

$movieInfoList = $doc->getElementsByTagName('movie');

foreach ($movieInfoList as $movieInfo) {

    $movieTitle = $movieInfo->getElementsByTagName('title')->item(0)->nodeValue;

    $actorName = $movieInfo->getElementsByTagName('actor')->item(0)->nodeValue;

    echo "MovieTitle: " . $movieTitle . "<br/>";

    echo "ActorName: " . $actorName . "<br/>";

}

?>

**2)      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.preprocessing import TransactionEncoder

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

import urllib.request

url = "http://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx"

filename = "Online Retail.xlsx"

urllib.request.urlretrieve(url, filename)

df = pd.read\_excel(filename)

print("Dataset information:")

print(df.info())

df = df.dropna()

df = df[df['Quantity'] > 0]

df['StockCode'] = pd.to\_numeric(df['StockCode'], errors='coerce')

transactions = df.groupby(['InvoiceNo'])['StockCode'].apply(list).values.tolist()

te = TransactionEncoder()

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

tid = pd.DataFrame(te\_ary, columns=te.columns\_)

items = tid.astype('int')

min\_sup = 0.03

frequent\_itemsets = apriori(items, min\_support=min\_sup, use\_colnames=True)

rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.7)

print('Min support:', min\_sup)

print('Frequent itemsets:')

print(frequent\_itemsets)

print('Association rules:')

print(rules)

**Slips8**

**1)      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>

    <script>

alert("Exams are near, have you started preparing for?");

var num1 = prompt("Enter the first number:");

var num2 = prompt("Enter the second number:");

num1 = parseInt(num1);

num2 = parseInt(num2);

var sum = num1 + num2;

confirm("The sum of " + num1 + " and " + num2 + " is " + sum + ".");

</script>

</html>

**2)      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.preprocessing import TransactionEncoder

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

df = pd.read\_csv('Downloads/archive (3)/Groceries\_dataset.csv')

print(df.info())

df.dropna(inplace=True)  # Drop rows with missing values

df = df.apply(lambda x: pd.factorize(x)[0])  # Convert categorical values to numeric format

te = TransactionEncoder()

te\_ary = te.fit\_transform(df.values)

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

frequent\_itemsets = apriori(df, min\_support=0.01, use\_colnames=True)

association\_rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.5)

print("Frequent Itemsets:")

print(frequent\_itemsets)

print("Association Rules:")

print(association\_rules)

**Slips9**

**1)      Write a JavaScript function to validate username and password for a membership form**

 <html>

    <form>

        enter name:

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

        enter pass:

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

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

    </form>

</html>

<script>

function validateForm() {

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

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

if(u=="")

alert("enter username");

if(p=="")

alert("enter password");

if(p.length<8)

alert(" password lenghth must be greater than 8" );

}

</script>

**2)      Create your own transactions dataset and apply the above process on your dataset.**

import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

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

dataset = [['beer', 'chips', 'soda'],

           ['beer', 'soda', 'pizza', 'wings'],

           ['soda', 'pizza'],

           ['beer', 'chips', 'soda', 'pizza'],

           ['beer', 'chips', 'wings'],

           ['chips', 'soda', 'pizza']]

df = pd.DataFrame(dataset)

print("Dataset information:")

print(df.info())

te = TransactionEncoder()

te\_ary = te.fit\_transform(df.values)

tid = pd.DataFrame(te\_ary, columns=te.columns\_)

items = tid.astype('int')

min\_sup = 0.5

frequent\_itemsets = apriori(items, min\_support=min\_sup, use\_colnames=True)

rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.5)

print('Min support:', min\_sup)

print('Frequent itemsets:')

print(frequent\_itemsets)

print('Association rules:')

print(rules)

**Slips10**

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

  <html>

    <head>

        <script src="jquery-3.6.4.js">

        </script>

        <script>

            $(document).ready(function(){

                $('#bt1').click(function(){

                $('p').before("good morning");

            });

            $('#bt2').click(function(){

                $('p').after("how are you");

            });

        });

        </script>

    </head>

    <body>

        <button id="bt1">before</button><br><br>

       <p> hello evryone</p>

        <button id="bt2">after</button>

    </body>

</html>

**2)      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 pandas as pd

from mlxtend.preprocessing import TransactionEncoder

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

data = [['Milk', 'Egg', 'Bread'],

        ['Milk', 'Bread'],

        ['Milk', 'Egg', 'Bread', 'Cheese'],

        ['Milk', 'Egg'],

        ['Bread', 'Cheese']]

te = TransactionEncoder()

te\_ary = te.fit\_transform(data)

tid = pd.DataFrame(te\_ary, columns=te.columns\_)

items = tid.astype('int')

items = items.replace({True: 1, False: 0})

min\_sup\_values = [0.4, 0.6, 0.8]

for min\_sup in min\_sup\_values:

    frequent\_itemsets = apriori(items, min\_support=min\_sup, use\_colnames=True)

    rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.7)

    print('Min support:', min\_sup)

    print('Frequent itemsets:')

    print(frequent\_itemsets)

    print('Association rules:')

    print(rules)

    print()

**Slips11**

**1)      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)**

 <!DOCTYPE html>

<html>

<head>

    <title>Change Font and Image on Events</title>

    <script>

        function checkName() {

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

            if (name === "") {

                // display image on empty text box

                var img = document.getElementById("img");

                img.src = "image.jpg";

                img.style.width = "100px";

                img.style.height = "100px";

                img.onload = function() {

                    this.style.width = "200px";

                    this.style.height = "200px";

                };

                img.onmouseover = function() {

                    this.style.width = "300px";

                    this.style.height = "300px";

                };

                img.onmousedown = function() {

                    this.style.width = "200px";

                    this.style.height = "200px";

                };

                img.onmouseup = function() {

                    this.style.width = "300px";

                    this.style.height = "300px";

                };

            } else {

                // change font color and size

                var element = document.getElementById("name");

                element.style.color = "red";

                element.style.fontSize = "18px";

            }

        }

    </script>

</head>

<body>

    <label for="name">Name:</label>

    <input type="text" id="name" onblur="checkName()">

    <br><br>

    <img id="img" src="" alt="">

</body>

</html>

**2)      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.**

**please refer slips number 10**

**Slips12**

**1)      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. [Enter at least 3 record in contact.dat file]**

 <!DOCTYPE html>

<html>

<head>

    <title>Contacts</title>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

    <script src="script.js"></script>

</head>

<body>

    <h1>Contacts</h1>

    <button id="printBtn">Print Contacts</button>

    <table id="contactsTable">

        <thead>

            <tr>

                <th>Sr No</th>

                <th>Name</th>

                <th>Residence Number</th>

                <th>Mobile Number</th>

                <th>Address</th>

            </tr>

        </thead>

        <tbody></tbody>

    </table>

</body>

</html>

<script >

    $(document).ready(function() {

    $('#printBtn').on('click', function() {

        $.ajax({

            url: 'contact.dat',

            type: 'GET',

            dataType: 'text',

            success: function(data) {

                let contacts = data.split('\n');

                let tbody = $('#contactsTable tbody');

                tbody.empty(); // Clear any existing rows from the table body

                for (let i = 0; i < contacts.length; i++) {

                    let fields = contacts[i].split(',');

                    if (fields.length === 5) { // Only process lines with 5 fields

                        let srno = fields[0].trim();

                        let name = fields[1].trim();

                        let residenceNumber = fields[2].trim();

                        let mobileNumber = fields[3].trim();

                        let address = fields[4].trim();

                        let row = '<tr><td>' + srno + '</td><td>' + name + '</td><td>' + residenceNumber + '</td><td>' + mobileNumber + '</td><td>' + address + '</td></tr>';

                        tbody.append(row);

                    }

                }

            },

            error: function(jqXHR, textStatus, errorThrown) {

                alert('Error retrieving contacts: ' + textStatus + ', ' + errorThrown);

            }

        });

    });

});

</script>

**2)      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

np.random.seed(0)

heights = np.random.normal(loc=170, scale=10, size=100)

weights = 0.7 \* heights + np.random.normal(loc=0, scale=5, size=100)

data = np.column\_stack((heights, weights))

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data[:,0], data[:,1], test\_size=0.2, random\_state=0)

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

model.fit(X\_train.reshape(-1, 1), y\_train.reshape(-1, 1))

y\_pred = model.predict(X\_test.reshape(-1, 1))

from sklearn.metrics import r2\_score

r2 = r2\_score(y\_test, y\_pred)

print("R-squared:", r2)

**Slips13**

**1)      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!”**

 <!DOCTYPE html>

<html>

<head>

    <title>AJAX Example</title>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

</head>

<body>

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

    <div id="response"></div>

    <script>

        $(document).ready(function() {

            $("#nameInput").on("input", function() {

                var name = $(this).val();

                if (name === "") {

                    $("#response").html("Stranger, please tell me your name!");

                } else if (["Rohit", "Virat", "Dhoni", "Ashwin", "Harbhajan"].includes(name)) {

                    $("#response").html("Hello, master!");

                } else {

                    $("#response").html(name + ", I don't know you!");

                }

            });

        });

    </script>

</body>

</html>

**2)      Download nursery dataset from UCI. 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 pandas as pd

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/nursery/nursery.data"

names = ["parents", "has\_nurs", "form", "children", "housing", "finance", "social", "health", "target"]

df = pd.read\_csv(url, names=names)

X = df.drop("target", axis=1)

y = df["target"]

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

from sklearn.linear\_model import LinearRegression

lr = LinearRegression()

lr.fit(X\_train, y\_train)

print("X\_train:", X\_train.shape)

print("y\_train:", y\_train.shape)

print("X\_test:", X\_test.shape)

print("y\_test:", y\_test.shape)

**Slips14**

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

  <!DOCTYPE html>

<html>

<head>

  <meta charset="UTF-8">

  <title>Teacher Details</title>

  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

  <script>

    $(document).ready(function() {

      // Bind an event handler to the select element

      $("#teacher-select").change(function() {

        // Get the selected teacher's name

        var tname = $(this).val();

        // Send an Ajax request to get the teacher's details

        $.ajax({

          type: "POST",

          url: "get-teacher-details.php",

          data: { tname: tname },

          dataType: "json",

          success: function(data) {

            // Update the details section with the teacher's details

            $("#qualification").text(data.qualification);

            $("#salary").text(data.salary);

          }

        });

      });

    });

  </script>

</head>

<body>

  <h1>Select a Teacher</h1>

  <select id="teacher-select">

    <option value="">-- Select a Teacher --</option>

    <?php

      // Connect to the database

      $conn = new mysqli("localhost", "username", "password", "database");

      // Get the list of teachers

      $result = $conn->query("SELECT tname FROM TEACHER");

      while ($row = $result->fetch\_assoc()) {

        // Output an option for each teacher

        echo "<option value=\"{$row["tname"]}\">{$row["tname"]}</option>";

      }

      // Close the database connection

      $conn->close();

    ?>

  </select>

  <h2>Details</h2>

  <p>Qualification: <span id="qualification"></span></p>

  <p>Salary: <span id="salary"></span></p>

</body>

</html>

<?php

  // Connect to the database

  $conn = new mysqli("localhost", "username", "password", "database");

  // Get the selected teacher's name

  $tname = $\_POST["tname"];

  // Query the database for the teacher's details

  $result = $conn->query("SELECT qualification, salary FROM TEACHER WHERE tname = '{$tname}'");

  $row = $result->fetch\_assoc();

  // Close the database connection

  $conn->close();

  // Return the teacher's details as a JSON object

  echo json\_encode($row);

?>

**2)      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**

**please refer slips number 10**

**Slips15**

**1)      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>

    <head>

        <script language="javascript" src="sugg.js">

        </script>

    </head>

    <body>

        <form id="frm" name="frm">

            search:

            <input type="text" id="t1"onkeyup="javascript:search(document.getElementById('frm'));"><br>

        </form>

        <div name="txt" id="txt">

        </div>

    </body>

</html>

<script src="">

function search(obj)

{

    var XHRobj=false;

    if(window.XMLHttpRequest)

    {

        XHRobj=new XMLHttpRequest();

    }

    else if(window.ActiveXObject)

    {

        XHRobj=new ActiveXObject("Microsoft.XMLHTTP");

    }

    var str1="t1="+document.getElementById("t1").value;

    XHRobj.onreadystatechange=show;

    XHRobj.open('POST','sugg.php',true);

    XHRobj.setRequestHeader("content-type","application/x-www-form-urlencoded");

    XHRobj.send(str1);

    function show()

    {

        if(XHRobj.readyState==4)

        {

            if(XHRobj.status==200)

            {

                result=XHRobj.responseText;

                document.getElementById('txt').innerHTML=result;

            }

        }

    }

}

</script>

<?php

$name=$\_POST["t1"];

$a=array("sai","sham","soham","amna","apeksha","aarati");

foreach($a as $v)

{

if(strstr($v,$name))

echo"<br>".$v;

}

?>

**2)      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 pandas as pd

from mlxtend.preprocessing import TransactionEncoder

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

dataset = [['tata', 'nexon', '2017'],

           ['MG', 'astor', '2021'],

           ['KIA', 'seltos', '2019'],

            ['hyundai', 'creta', '2015']

          ]

te = TransactionEncoder()

te\_ary = te.fit\_transform(dataset)

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

for min\_sup in [0.2, 0.4]:

    frequent\_itemsets = apriori(df, min\_support=min\_sup, use\_colnames=True)

    association\_rules = association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.5)

    # Display the results

    print(f"Frequent Itemsets with minimum support of {min\_sup}:")

    print(frequent\_itemsets)

    print(f"Association Rules with minimum support of {min\_sup}:")

    print(association\_rules)

**Slips16**

**1)      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).**

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

<book>

<info>

<bno>1</bno>

<name>c</name>

<author>xyz</author>

</info>

<info>

<bno>2</bno>

<name>php</name>

<author>abc</author>

</info>

<info>

<bno>3</bno>

<name>java</name>

<author>pqr</author>

</info>

</book>

<html>

    <head>

        <script language="javascript" src="book.js">

        </script>

    </head>

    <body>

        <form id="frm" name="frm" action="javascript:search(document.getElementById('frm'));">

            enter book name:

            <input type="text" id="t1"><br>

            <input type="submit" value="search"><br>

        </form>

        <div name="txt" id="txt">

        </div>

    </body>

</html>

<script>

function search(obj)

{

    var XHRobj=false;

    if(window.XMLHttpRequest)

    {

        XHRobj=new XMLHttpRequest();

    }

    else if(window.ActiveXObject)

    {

        XHRobj=new ActiveXObject("Microsoft.XMLHTTP");

    }

    var str1="t1="+document.getElementById("t1").value;

    XHRobj.onreadystatechange=show;

    XHRobj.open('POST','book.php',true);

    XHRobj.setRequestHeader("content-type","application/x-www-form-urlencoded");

    XHRobj.send(str1);

    function show()

    {

        if(XHRobj.readyState==4)

        {

            if(XHRobj.status==200)

            {

                result=XHRobj.responseText;

                document.getElementById('txt').innerHTML=result;

            }

        }

    }

}

</script>

<?php

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

if($ob==false)

    die("file not found");

$c=$\_POST["t1"];

echo"<table border=1";

echo"<tr><th>bno<th>name<th>author</tr>";

foreach($ob->info as $a)

{

    if($a->name==$c)

    {

    echo"<tr><td>".$a->bno;

    echo"<td>".$a->name;

    echo"<td>".$a->author;

    echo"</tr>";

}

}

echo"</table>";

?>

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

  import re

import nltk

from nltk.corpus import stopwords

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

from heapq import nlargest

text = '''Extractive summarization is a process that involves automatically selecting sentences from a text

document to create a shorter version that conveys the most important information. It is commonly used in news articles,

scientific papers, and other types of text documents. The main advantage of extractive summarization is that it preserves

the original wording and context of the text, making it easier for readers to understand the main points. In this program,

we will demonstrate how to perform extractive summarization on a text paragraph using Python and NLTK.'''

text = re.sub(r'\d+', '', text)

text = re.sub(r'[^\w\s]', '', text)

sentences = sent\_tokenize(text)

words = word\_tokenize(text.lower())

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

word\_frequencies = {}

for word in words:

    if word not in stop\_words:

        if word not in word\_frequencies:

            word\_frequencies[word] = 1

        else:

            word\_frequencies[word] += 1

sentence\_scores = {}

for sentence in sentences:

    for word in word\_tokenize(sentence.lower()):

        if word in word\_frequencies:

            if len(sentence.split(' ')) < 30:  # Exclude long sentences

                if sentence not in sentence\_scores:

                    sentence\_scores[sentence] = word\_frequencies[word]

                else:

                    sentence\_scores[sentence] += word\_frequencies[word]

summary\_sentences = nlargest(2, sentence\_scores, key=sentence\_scores.get)

summary = ' '.join(summary\_sentences)

print("Original text:\n", text)

print("\nSummary:\n", summary)

**Slips17**

**1)         Write a Java Script Program to show Hello Good Morning message onload event using alert box and display the Student registration from.**

  <!DOCTYPE html>

<html>

  <body onload="showGreeting()">

    <script>

      function showGreeting() {

        alert("Hello, Good Morning!");

      }

    </script>

    <h1>Student Registration Form</h1>

    <form>

      Name:

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

      <br>

     Email:

      <input type="email" id="email" name="email" >

      <br>

      Phone:

      <input type="tel" id="phone" name="phone" >

      <br>

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

    </form>

  </body>

</html>

**2)      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

import nltk

from nltk.corpus import stopwords

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

from heapq import nlargest

# Sample text paragraph

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.'''

# Preprocess the text

text = re.sub(r'\d+', '', text)  # Remove digits

text = re.sub(r'[^\w\s]', '', text)  # Remove special characters

sentences = sent\_tokenize(text)  # Tokenize into sentences

words = word\_tokenize(text.lower())  # Tokenize into words and convert to lowercase

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

word\_frequencies = {}  # Calculate word frequencies

for word in words:

    if word not in stop\_words:

        if word not in word\_frequencies:

            word\_frequencies[word] = 1

        else:

            word\_frequencies[word] += 1

# Calculate sentence scores

sentence\_scores = {}

for sentence in sentences:

    for word in word\_tokenize(sentence.lower()):

        if word in word\_frequencies:

            if len(sentence.split(' ')) < 30:  # Exclude long sentences

                if sentence not in sentence\_scores:

                    sentence\_scores[sentence] = word\_frequencies[word]

                else:

                    sentence\_scores[sentence] += word\_frequencies[word]

# Generate summary

summary\_sentences = nlargest(2, sentence\_scores, key=sentence\_scores.get)

summary = ' '.join(summary\_sentences)

# Display the results

print("Original text:\n", text)

print("\nSummary:\n", summary)

**Slips18**

**1)      Write a Java Script Program to print Fibonacci numbers on onclick event.**

  <!DOCTYPE html>

<html>

<body>

<h1>Fibonacci Sequence</h1>

<p>Click the button to print the first 10 numbers in the Fibonacci sequence:</p>

<button onclick="fibonacci()">Print Fibonacci Sequence</button>

<p id="fib-sequence"></p>

<script>

function fibonacci() {

  var n = 10;

  var fib = [0, 1];

  for (var i = 2; i < n; i++) {

    fib[i] = fib[i - 1] + fib[i - 2];

  }

  document.getElementById("fib-sequence").innerHTML = fib.join(", ");

}

</script>

</body>

</html>

**2)      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 nltk

from nltk.corpus import stopwords

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

import matplotlib.pyplot as plt

from wordcloud import WordCloud

# Example text paragraph

text = "Natural Language Processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human–computer interaction. Many challenges in NLP involve natural language understanding, that is, enabling computers to derive meaning from human or natural language input, and others involve natural language generation."

# Remove stopwords

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

words = word\_tokenize(text.lower())

words = [w for w in words if not w in stop\_words]

# Tokenize into sentences

sentences = sent\_tokenize(text)

# Calculate word frequency distribution

freq\_dist = nltk.FreqDist(words)

# Plot word frequencies

freq\_dist.plot(30)

# Plot wordcloud

wordcloud = WordCloud(width = 800, height = 800, background\_color ='white', min\_font\_size = 10).generate(text)

plt.figure(figsize = (8, 8), facecolor = None)

plt.imshow(wordcloud)

plt.axis("off")

plt.tight\_layout(pad = 0)

plt.show()

**Slips19**

**1)      Write a Java Script Program to validate user name and password on onSubmit event.**

<!DOCTYPE html>

<html>

  <head>

    <meta charset="UTF-8">

    <title>Login Form</title>

  </head>

  <body>

    <form name="loginForm" onsubmit="return validateForm()">

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

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

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

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

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

    </form>

    <script>

      function validateForm() {

        // Get the values of the username and password input fields

        var username = document.forms["loginForm"]["username"].value;

        var password = document.forms["loginForm"]["password"].value;

        // Validate the username and password

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

          alert("Please enter a username.");

          return false;

        } else if (password == null || password == "") {

          alert("Please enter a password.");

          return false;

        } else {

          return true;

        }

      }

    </script>

  </body>

</html>

**2)      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

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

from textblob import TextBlob

def get\_sentiment(text):

    blob = TextBlob(text)

    sentiment = blob.sentiment.polarity

    if sentiment > 0:

        return 'positive'

    elif sentiment < 0:

        return 'negative'

    else:

        return 'neutral'

df['sentiment'] = df['review'].apply(get\_sentiment)

from wordcloud import WordCloud

import matplotlib.pyplot as plt

# Combine all reviews into a single string

text = ' '.join(df['review'])

# Create a word cloud

wordcloud = WordCloud(width=800, height=800, background\_color='white', min\_font\_size=10).generate(text)

# Plot the word cloud

plt.figure(figsize=(8, 8))

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

plt.axis('off')

plt.show()

**Slips20**

**1)      create a student.xml file containing at least 5 student information**

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

<students>

  <student>

    <id>001</id>

    <name>John Doe</name>

    <age>21</age>

    <major>Computer Science</major>

    <gpa>3.5</gpa>

  </student>

  <student>

    <id>002</id>

    <name>Jane Smith</name>

    <age>20</age>

    <major>Engineering</major>

    <gpa>3.8</gpa>

  </student>

  <student>

    <id>003</id>

    <name>Bob Johnson</name>

    <age>22</age>

    <major>Business Administration</major>

    <gpa>3.2</gpa>

  </student>

  <student>

    <id>004</id>

    <name>Sara Lee</name>

    <age>19</age>

    <major>Mathematics</major>

    <gpa>3.9</gpa>

  </student>

  <student>

    <id>005</id>

    <name>Mike Johnson</name>

    <age>20</age>

    <major>Psychology</major>

    <gpa>3.6</gpa>

  </student>

</students>

**2)      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

# Sample text paragraph

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."""

# Tokenize into words

words = word\_tokenize(text.lower())

# Load English stop words

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

# Remove stop words

filtered\_words = [word for word in words if word not in stop\_words]

# Join the filtered words into a string

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

# Display the results

print("Original text:\n", text)

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

**Slips21**

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

  <!DOCTYPE html>

<html>

<head>

    <title>Number Check</title>

    <script>

function checkNumber(number) {

  if (number > 0) {

    console.log(number + " is positive");

  } else if (number < 0) {

    console.log(number + " is negative");

  } else {

    console.log(number + " is neither positive nor negative");

  }

}

    </script>

</head>

<body>

    <input type="number" id="numberInput">

    <button onclick="check()">Check</button>

    <script>

        function check() {

            var number = document.getElementById("numberInput").value;

            checkNumber(number);

        }

    </script>

</body>

</html>

**2)      Build a simple linear regression model for User Data.**

 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

# load the user data into a pandas DataFrame

user\_data = pd.read\_csv("user\_data.csv")

# split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(user\_data[["independent\_var"]], user\_data["dependent\_var"], test\_size=0.2, random\_state=42)

# create a Linear Regression model and fit it to the training data

model = LinearRegression()

model.fit(X\_train, y\_train)

# make predictions on the testing data

y\_pred = model.predict(X\_test)

# evaluate the model's performance using mean squared error

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

print("Mean Squared Error: ", mse)

# use the model to make predictions on new data

new\_data = pd.DataFrame({"independent\_var": [1.5, 2.0, 3.5]})

predicted\_values = model.predict(new\_data)

print("Predicted Values: ", predicted\_values)

**Slips22**

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

 <?php

class Student\_model extends CI\_Model {

    function \_\_construct() {

        parent::\_\_construct();

    }

    function insert\_student($data) {

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

    }

}

class Student extends CI\_Controller {

    function \_\_construct() {

        parent::\_\_construct();

        $this->load->model('student\_model');

    }

    function insert() {

        $data = array(

            array('rollno' => '1', 'name' => 'John', 'class' => '10th'),

            array('rollno' => '2', 'name' => 'Mary', 'class' => '9th'),

            array('rollno' => '3', 'name' => 'Bob', 'class' => '11th'),

            array('rollno' => '4', 'name' => 'Alice', 'class' => '12th'),

            array('rollno' => '5', 'name' => 'Tom', 'class' => '8th')

        );

        foreach ($data as $row) {

            $this->student\_model->insert\_student($row);

        }

        echo "Records inserted successfully!";

    }

}

?>

**2)      Consider any text paragraph. Remove the stopwords.**

 import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

# Sample text paragraph

text = "This is a sample text paragraph. It contains some words that are stopwords, such as 'the', 'is', and 'a'."

# Tokenize the paragraph into words

words = word\_tokenize(text)

# Remove stopwords

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

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

# Print the filtered words

print(filtered\_words)

**Slips23**

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

  <html>

<head>

  <title>Student List</title>

</head>

<body>

  <h1>Student List</h1>

  <table>

    <thead>

      <tr>

        <th>Roll No</th>

        <th>Name</th>

        <th>Class</th>

      </tr>

    </thead>

    <tbody>

      <?php foreach ($students as $student): ?>

        <tr>

          <td><?php echo $student->rollno; ?></td>

          <td><?php echo $student->name; ?></td>

          <td><?php echo $student->class; ?></td>

        </tr>

      <?php endforeach; ?>

    </tbody>

  </table>

</body>

</html>

<?php

class Student\_model extends CI\_Model {

public function \_\_construct() {

  $this->load->database();

}

public function get\_all\_students() {

  $query = $this->db->get('student');

  return $query->result();

}

}

class Student extends CI\_Controller {

  public function \_\_construct() {

    parent::\_\_construct();

    $this->load->model('student\_model');

  }

  public function index() {

    $data['students'] = $this->student\_model->get\_all\_students();

    $this->load->view('student/index', $data);

  }

}

?>

sql

CREATE TABLE student (

  rollno INT PRIMARY KEY,

  name VARCHAR(50),

  class VARCHAR(50)

);

INSERT INTO student (rollno, name, class) VALUES

  (1, 'John', '12th'),

  (2, 'Mary', '11th'),

  (3, 'Bob', '10th'),

  (4, 'Alice', '9th'),

  (5, 'David', '12th');

**2)      Consider any text paragraph. Preprocess the text to remove any special characters and digits.**

   import re

# Example text paragraph

text = "Hello! This is an example paragraph. It contains special characters like % and digits like 123."

# Remove special characters and digits

processed\_text = re.sub(r'[^\w\s]', '', text)

processed\_text = re.sub(r'\d+', '', processed\_text)

print(processed\_text)

**Slips24**

**1)      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.**

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

<?xml-stylesheet type="text/css" href="stud.css"?>

<student>

<info>

<sno>101</sno>

<name>om</name>

<course>bcs</course>

<college>rbnb</college>

</info>

<info>

<sno>102</sno>

<name>sai</name>

<course>bcs</course>

<college>rbnb</college>

</info>

<info>

<sno>103</sno>

<name>ram</name>

<course>bca</course>

<college>cdj</college>

</info>

</student>

<html>

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

        enter course to search:

        <input type="text" name="t1"><br>

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

    </form>

</html>

stud.php

<?php

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

if($ob==false)

    die("file not found");

$c=$\_POST["t1"];

echo"<table border=1";

echo"<tr><th>rno<th>name<th>course<th>college</tr>";

foreach($ob->info as $a)

{

    if($a->course==$c)

    {

    echo"<tr><td>".$a->sno;

    echo"<td>".$a->name;

    echo"<td>".$a->course;

    echo"<td>".$a->college;

    echo"</tr>";

}

}

echo"</table>";

?>

**2)       Consider the following dataset : https://www.kaggle.com/datasets/datasnaek/youtubenew?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

# Read the dataset

df = pd.read\_csv('INvideos.csv')

# Drop unnecessary columns

df.drop(['video\_id', 'trending\_date', 'title', 'channel\_title', 'category\_id', 'publish\_time', 'tags', 'thumbnail\_link', 'comments\_disabled', 'ratings\_disabled', 'video\_error\_or\_removed'], axis=1, inplace=True)

# Drop duplicate rows

df.drop\_duplicates(inplace=True)

# Drop rows with missing values

df.dropna(inplace=True)

# Convert the data types of relevant columns

df['views'] = pd.to\_numeric(df['views'])

df['likes'] = pd.to\_numeric(df['likes'])

df['dislikes'] = pd.to\_numeric(df['dislikes'])

df['comment\_count'] = pd.to\_numeric(df['comment\_count'])

# Calculate total views, likes, dislikes, and comment count

total\_views = df['views'].sum()

total\_likes = df['likes'].sum()

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

total\_comment\_count = df['comment\_count'].sum()

# Display the results

print('Total views:', total\_views)

print('Total likes:', total\_likes)

print('Total dislikes:', total\_dislikes)

print('Total comment count:', total\_comment\_count)

**Slips25**

**1)      Write a script to create “cricket.xml” file with multiple elements as shown below:**

**< CricketTeam>**

**< Team country=”Australia”>**

**<player>\_\_\_\_\_</player>**

**<runs>\_\_\_\_\_\_\_</runs>**

**<wicket>\_\_\_\_\_</wicket>**

**</Team>**

**</ CricketTeam>**

**Write a script to add multiple elements in “cricket.xml” file of category, country=”India”.**

 <CricketTeam>

    <Team country="Australia">

    <player>xyz</player>

    <runs>12</runs>

    <wicket>2</wicket>

    </Team>

    <Team country="india">

    <player>xyz</player>

    <runs>12</runs>

    <wicket>2</wicket>

    </Team>

    <Team country="india">

    <player>xyz</player>

    <runs>12</runs>

    <wicket>2</wicket>

    </Team>

    <Team country="”Australia”">

    <player>xyz</player>

    <runs>12</runs>

    <wicket>2</wicket>

    </Team>

    </CricketTeam>

**2)** **Consider the following dataset : https://www.kaggle.com/datasets/seungguini/youtube-commentsfor-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.                  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

import nltk

from nltk.sentiment import SentimentIntensityAnalyzer

# Read the dataset

df = pd.read\_csv('Downloads/archive (4)/covid\_2021\_1.csv')

# Drop unnecessary columns

df.drop(['comment\_id', 'author', 'comment\_date'], axis=1, inplace=True)

# Drop duplicate rows

df.drop\_duplicates(inplace=True)

# Drop rows with missing values

df.dropna(inplace=True)

# Tokenize comments into words

df['comment'] = df['comment'].apply(lambda x: re.findall(r'\b\w+\b', x.lower()))

# Perform sentiment analysis

nltk.download('vader\_lexicon')

sia = SentimentIntensityAnalyzer()

df['sentiment\_score'] = df['comment'].apply(lambda x: sia.polarity\_scores(' '.join(x))['compound'])

# Classify comments as positive, negative, or neutral

df['sentiment'] = df['sentiment\_score'].apply(lambda x: 'positive' if x > 0 else 'negative' if x < 0 else 'neutral')

# Calculate percentage of positive, negative, and neutral comments

positive\_percent = (df['sentiment'] == 'positive').sum() / len(df) \* 100

negative\_percent = (df['sentiment'] == 'negative').sum() / len(df) \* 100

neutral\_percent = (df['sentiment'] == 'neutral').sum() / len(df) \* 100

# Display the results

print('Percentage of positive comments:', positive\_percent)

print('Percentage of negative comments:', negative\_percent)

print('Percentage of neutral comments:', neutral\_percent)

.

**Slips26**

**1)      Create employee table as follows EMP (eno, ename, designation, salary). Write Ajax program to select the employees name and print the selected employee’s details**

 <!DOCTYPE html>

<html>

<head>

  <title>Employee Details</title>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

  <script>

    $(document).ready(function() {

      $('#select-button').click(function() {

        var ename = $('#ename-input').val();

        $.ajax({

          url: 'get\_employee\_details.php',

          type: 'post',

          data: { ename: ename },

          success: function(response) {

            $('#employee-details').html(response);

          }

        });

      });

    });

  </script>

</head>

<body>

  <h1>Select Employee</h1>

  <input type="text" id="ename-input" placeholder="Enter employee name">

  <button id="select-button">Select</button>

  <div id="employee-details"></div>

</body>

</html>

<?php

$ename = $\_POST['ename'];

$conn = new mysqli('localhost', 'username', 'password', 'database\_name');

if ($conn->connect\_error) {

  die('Connection failed: ' . $conn->connect\_error);

}

$sql = "SELECT \* FROM EMP WHERE ename = '$ename'";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

  while ($row = $result->fetch\_assoc()) {

    echo '<p><strong>Employee Number:</strong> ' . $row['eno'] . '</p>';

    echo '<p><strong>Employee Name:</strong> ' . $row['ename'] . '</p>';

    echo '<p><strong>Designation:</strong> ' . $row['designation'] . '</p>';

    echo '<p><strong>Salary:</strong> $' . $row['salary'] . '</p>';

  }

} else {

  echo 'No employee found with name ' . $ename;

}

$conn->close();

?>

sql:-

CREATE TABLE EMP (

  eno INT PRIMARY KEY,

  ename VARCHAR(50),

  designation VARCHAR(50),

  salary DECIMAL(10, 2)

);

**2)      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.""" Preprocess the text to remove any special characters and digits. Generate the summary using extractive summarization process.**

   import re

from nltk.corpus import stopwords

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

from collections import defaultdict

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."

# Preprocess the text

text = re.sub(r'[^\w\s]', '', text)  # remove special characters

text = re.sub(r'\d+', '', text)  # remove digits

text = text.lower()  # convert to lowercase

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

words = word\_tokenize(text)

words = [word for word in words if word not in stop\_words]

# Generate word frequency

word\_freq = defaultdict(int)

for word in words:

    word\_freq[word] += 1

# Generate sentence score based on word frequency

sent\_scores = defaultdict(int)

for sentence in sent\_tokenize(text):

    for word in word\_tokenize(sentence.lower()):

        if word in word\_freq:

            sent\_scores[sentence] += word\_freq[word]

# Get the top 2 sentences based on sentence score

summary\_sentences = sorted(sent\_scores, key=sent\_scores.get, reverse=True)[:2]

# Join the summary sentences

summary = ' '.join(summary\_sentences)

print(summary)

**Slips27**

**1)      Create web Application that contains Voters details and check proper validation for (name, age, and nationality), as Name should be in upper case letters only, Age should not be less than 18 yrs and Nationality should be Indian.(use HTML-AJAX-PHP**

          <!DOCTYPE html>

<html>

<head>

    <title>Voter Registration</title>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

    <script src=>

        $(document).ready(function(){

    $("#voter-form").submit(function(event){

        event.preventDefault();

        var name = $("#name").val();

        var age = $("#age").val();

        var nationality = $("#nationality").val();

        if(validateForm(name, age, nationality)){

            $.ajax({

                url: "submit.php",

                type: "POST",

                data: {

                    name: name,

                    age: age,

                    nationality: nationality

                },

                success: function(data){

                    $("#result").html(data);

                }

            });

        }

    });

    function validateForm(name, age, nationality){

        if(name == "" || age == "" || nationality == ""){

            alert("All fields are required!");

            return false;

        }

        if(!/^[A-Z]+$/.test(name)){

            alert("Name should be in uppercase letters only!");

            return false;

        }

        if(age < 18){

            alert("Age should not be less than 18 years!");

            return false;

        }

        if(nationality.toLowerCase() != "indian"){

            alert("Nationality should be Indian!");

            return false;

        }

        return true;

    }

});

    </script>

</head>

<body>

    <h1>Voter Registration</h1>

    <form id="voter-form" method="post">

        <label for="name">Name:</label>

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

        <label for="age">Age:</label>

        <input type="number" id="age" name="age"><br>

        <label for="nationality">Nationality:</label>

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

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

    </form>

    <div id="result"></div>

</body>

<?php

    $name = $\_POST['name'];

    $age = $\_POST['age'];

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

    // Here you can write code to store the data in a database or perform any other action you want.

    echo "Thank you for registering, $name!";

?>

</html>

**2)      Create your own transactions dataset and apply the above process on your dataset**

import csv

# create a sample transactions dataset

transactions = [

    ["John Doe", 25, "US", 100.00, "Credit Card"],

    ["Jane Smith", 30, "UK", 50.00, "PayPal"],

    ["David Lee", 20, "India", 75.00, "Debit Card"],

    ["Emily Chen", 18, "Canada", 200.00, "Credit Card"]

]

# write the transactions dataset to a CSV file

with open("transactions.csv", "w", newline="") as csvfile:

    writer = csv.writer(csvfile)

    writer.writerow(["Name", "Age", "Nationality", "Amount", "Payment Method"])

    for transaction in transactions:

        writer.writerow(transaction)

# read the transactions dataset from the CSV file

with open("transactions.csv", "r") as csvfile:

    reader = csv.reader(csvfile)

    next(reader)  # skip the header row

    for row in reader:

        name = row[0]

        age = int(row[1])

        nationality = row[2]

        amount = float(row[3])

        payment\_method = row[4]

        # validate the transaction

        if not name.isupper():

            print(f"Invalid name: {name}. Name should be in uppercase letters only.")

            continue

        if age < 18:

            print(f"Invalid age: {age}. Age should not be less than 18 years.")

            continue

        if nationality.lower() != "india":

            print(f"Invalid nationality: {nationality}. Nationality should be Indian.")

            continue

        # process the transaction

        print(f"Processing transaction for {name}. Amount: {amount}. Payment method: {payment\_method}.")

**Slips28**

**1)      Write a PHP script using AJAX concept, to check user name and password are valid or Invalid (use database to store user name and password)**

<?php

// database connection information

$servername = "localhost";

$username = "root";

$password = "password";

$dbname = "myDB";

// get the username and password from the AJAX request

$username = $\_POST['username'];

$password = $\_POST['password'];

// create a connection to the database

$conn = new mysqli($servername, $username, $password, $dbname);

// check if the connection was successful

if ($conn->connect\_error) {

    die("Connection failed: " . $conn->connect\_error);

}

// prepare the SQL statement to check if the username and password are valid

$stmt = $conn->prepare("SELECT \* FROM users WHERE username = ? AND password = ?");

$stmt->bind\_param("ss", $username, $password);

$stmt->execute();

$result = $stmt->get\_result();

// check if the username and password are valid

if ($result->num\_rows > 0) {

    echo "valid"; // return 'valid' if the username and password are valid

} else {

    echo "invalid"; // return 'invalid' if the username and password are invalid

}

// close the database connection

$conn->close();

?>

**2)      Build a simple linear regression model for Car Dataset**

   import pandas as pd

from sklearn.linear\_model import LinearRegression

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

# load the car dataset into a Pandas DataFrame

car\_data = pd.read\_csv("car\_data.csv")

# extract the input features (i.e., horsepower) and the output variable (i.e., price)

X = car\_data["horsepower"].values.reshape(-1, 1)

y = car\_data["price"].values.reshape(-1, 1)

# split the data into training and testing sets (80% for training, 20% for testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# create a linear regression model and fit it to the training data

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

# predict the prices for the testing data

y\_pred = regressor.predict(X\_test)

# evaluate the performance of the model using the mean squared error (MSE)

from sklearn.metrics import mean\_squared\_error

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

print("Mean Squared Error:", mse)

# plot the training data and the linear regression line

import matplotlib.pyplot as plt

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

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

plt.title("Linear Regression - Car Dataset")

plt.xlabel("Horsepower")

plt.ylabel("Price")

plt.show()

**Slips29**

**1)      Write a PHP script for the following: Design a form to accept a number from the user. Perform the operations and show the results.**

**1) Fibonacci Series.**

**2) To find sum of the digits of that number. (Use the concept of self processing page.)**

      <?php

// check if the form has been submitted

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

  // get the number from the user input

  $number = $\_POST["number"];

  // function to calculate the Fibonacci series

  function fibonacci($n) {

    if ($n == 0) {

      return 0;

    } elseif ($n == 1) {

      return 1;

    } else {

      return fibonacci($n - 1) + fibonacci($n - 2);

    }

  }

  // function to calculate the sum of digits in a number

  function sumOfDigits($number) {

    $sum = 0;

    while ($number != 0) {

      $sum += $number % 10;

      $number = (int)($number / 10);

    }

    return $sum;

  }

  // calculate the Fibonacci series and sum of digits for the input number

  $fibonacci = fibonacci($number);

  $sum = sumOfDigits($number);

  // display the results to the user

  echo "The Fibonacci series of $number is: $fibonacci<br>";

  echo "The sum of digits in $number is: $sum<br>";

}

?>

<!DOCTYPE html>

<html>

<head>

  <title>Number Operations</title>

</head>

<body>

  <h1>Number Operations</h1>

  <form method="POST" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>">

    Enter a number: <input type="text" name="number"><br><br>

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

  </form>

</body>

</html>

**2)      Build a logistic regression model for Student Score Dataset.**

import pandas as pd

from sklearn.linear\_model import LogisticRegression

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

from sklearn.metrics import accuracy\_score

# load the student score dataset

df = pd.read\_csv("student\_scores.csv")

# split the dataset into input features (X) and output variable (y)

X = df.iloc[:, :-1].values

y = df.iloc[:, -1].values

# split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# create a logistic regression model and fit it to the training data

logreg = LogisticRegression()

logreg.fit(X\_train, y\_train)

# make predictions on the testing data

y\_pred = logreg.predict(X\_test)

# evaluate the accuracy of the model

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

print("Accuracy:", accuracy)

**Slips30**

**1)      Create a XML file which gives details of books available in “Bookstore” from following categories.**

**1) Yoga**

**2) Story**

**3) Technical and elements in each category are in the following format**

**<Book>**

**<Book\_Title> --------------</Book\_Title>**

**<Book\_Author> ---------------</Book\_Author>**

**<Book\_Price> --------------</Book\_Price>**

**<Book>**

**Save the file as “Bookcategory.xml”**

<books>

<Book categorie="yoga">

<Book\_Title>abc</Book\_Title>

<Book\_Author> abc</Book\_Author>

<Book\_Price> 67</Book\_Price>

</Book>

<Book categorie="story">

<Book\_Title>abc</Book\_Title>

<Book\_Author> abc</Book\_Author>

<Book\_Price> 67</Book\_Price>

</Book>

<Book categorie="technical">

<Book\_Title>abc</Book\_Title>

<Book\_Author> abc</Book\_Author>

<Book\_Price> 67</Book\_Price>

</Book>

</books>

**2)      Create the dataset . transactions = [['eggs', 'milk','bread'], ['eggs', 'apple'], ['milk', 'bread'], ['apple', 'milk'], ['milk', 'apple', 'bread']] .**

**Convert the categorical values into numeric format.Apply the apriori algorithm on the above dataset to generate the frequent itemsets and association rules.**

from mlxtend.preprocessing import TransactionEncoder

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

# create dataset

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

                ['eggs', 'apple'],

                ['milk', 'bread'],

                ['apple', 'milk'],

                ['milk', 'apple', 'bread']]

# convert categorical values to numeric format

te = TransactionEncoder()

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

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

# apply apriori algorithm to generate frequent itemsets

frequent\_itemsets = apriori(df, min\_support=0.4, use\_colnames=True)

# generate association rules

rules = association\_rules(frequent\_itemsets, metric='lift', min\_threshold=1)

# print results

print("Frequent Itemsets:\n", frequent\_itemsets)

print("\nAssociation Rules:\n", rules)