

WT and DA

@Slip – 1

Q. 1) Write a PHP script to keep track of number of times the web page has been accessed (Use Session Tracking).

Ans:

```
<?php
Session_start();
If(isset($_SESSION['pcount'])) {
    $_SESSION['pcount'] += 1;
} else {
    $_SESSION['pcount'] = 1;
}
Echo "You have visited this page ".$_SESSION['pcount']." Time(s).";
?>
```

Q. 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.

Ans:

```
Import numpy as np
Import pandas as pd
From sklearn.model_selection import train_test_split
From sklearn.linear_model import LinearRegression
# Create the Position_Salaries dataset
Data = {'Position': ['CEO', 'charman', 'director', 'Senior Manager', 'Junior Manager', 'Intern'],
        'Level': [1, 2, 3, 4, 5, 6],
        'Salary': [50000, 80000, 110000, 150000, 200000, 250000]}
Df = pd.DataFrame(data)
# Identify the independent and target variables
X = df.iloc[:, 1:2].values
Y = df.iloc[:, 2].values
```

```

# Split the variables into training and testing sets with a 7:3 ratio
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Print the training and testing sets
Print("X_train:\n", X_train)
Print("y_train:\n", y_train)
Print("X_test:\n", X_test)
Print("y_test:\n", y_test)

# Build a simple linear regression model
Regressor = LinearRegression()
Regressor.fit(X_train, y_train)

# Print the coefficients and intercept
Print("Coefficients:", regressor.coef_)
Print("Intercept:", regressor.intercept_)

```

@Slip-2

Q. 1Write 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).

Ans :

Fristpage.html

```

<!DOCTYPE html>

<html>

<head>

    <title>Change preferences</title>

</head>

<body>

    <h1>Change preferences</h1>

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

        <label for="fontstyle">Font Style:</label>

        <select name="fontstyle" id="fontstyle">

```

```

        <option value="Arial">Arial</option>
        <option value="Times New Roman">Times New Roman</option>
        <option value="Verdana">Verdana</option>
    </select><br><br>
    <label for="fontsize">Font Size:</label>
    <select name="fontsize" id="fontsize">
        <option value="12">12</option>
        <option value="14">14</option>
        <option value="16">16</option>
    </select><br><br>
    <label for="fontcolor">Font Color:</label>
    <input type="color" name="fontcolor" id="fontcolor"><br><br>
    <label for="bgcolor">Background Color:</label>
    <input type="color" name="bgcolor" id="bgcolor"><br><br>
    <input type="submit" name="submit" value="Save">
</form>
</body>
</html>

```

Secondpage.php

```
<?php
```

```

If(isset($_POST['submit'])) {
    $fontstyle = $_POST['fontstyle'];
    $fontsize = $_POST['fontsize'];
    $fontcolor = $_POST['fontcolor'];
    $bgcolor = $_POST['bgcolor'];

    Setcookie('fontstyle', $fontstyle, time()+86400);
    Setcookie('fontsize', $fontsize, time()+86400);
    Setcookie('fontcolor', $fontcolor, time()+86400);
    Setcookie('bgcolor', $bgcolor, time()+86400);
}

```

```
Header('Location: thirdpage.php');  
Exit();  
}  
?>
```

Thirdpage.php

```
<?php  
$fontstyle = isset($_COOKIE['fontstyle']) ? $_COOKIE['fontstyle'] : 'Arial';  
$fontsize = isset($_COOKIE['fontsize']) ? $_COOKIE['fontsize'] : '12';  
$fontcolor = isset($_COOKIE['fontcolor']) ? $_COOKIE['fontcolor'] : '#000000';  
$bgcolor = isset($_COOKIE['bgcolor']) ? $_COOKIE['bgcolor'] : '#FFFFFF';  
?>
```

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>Page with new settings</title>  
    <style type="text/css">  
        Body {  
            Font-family: <?php echo $fontstyle ?>;  
            Font-size: <?php echo $fontsize ?>px;  
            Color: <?php echo $fontcolor ?>;  
            Background-color: <?php echo $bgcolor ?>;  
        }  
    </style>  
</head>  
<body><h1>Page with new settings</h1>  
    <p>This is the page with the new settings. The font style is <?php echo $fontstyle ?>, the  
font size is <?php echo $fontsize ?>px, the font color is <?php echo $fontcolor ?>, and the  
background color is <?php echo $bgcolor ?>.</p>  
</body>  
</html>
```

Q. 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.

Ans:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

Data = {'YearsExperience': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        'Salary': [50000, 60000, 70000, 80000, 90000, 100000, 110000, 120000, 130000, 140000]}

Df = pd.DataFrame(data)
X = df.iloc[:, 0:1].values
Y = df.iloc[:, 1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

Print("X_train:\n", X_train)
Print("y_train:\n", y_train)
Print("X_test:\n", X_test)
Print("y_test:\n", y_test)

Regressor = LinearRegression()
Regressor.fit(X_train, y_train)

Print("Coefficients:", regressor.coef_)
Print("Intercept:", regressor.intercept_)
```

@Slip-3

Q. 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]

Ans:

```
<?php
Session_start();
```

```

If(isset($_POST['submit'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];
    $correct_username = 'myusername';
    $correct_password = 'mypassword';
    If($username == $correct_username && $password == $correct_password) {
        $_SESSION['loggedin'] = true;
        Header('Location: welcome.php');
        Exit;
    } else {
        If(isset($_SESSION['attempts'])) {
            $_SESSION['attempts']--;
        } else {
            $_SESSION['attempts'] = 3;
        }
        If($_SESSION['attempts'] <= 0) {
            Echo "Maximum login attempts exceeded. Please try again later.";
        } else {
            Echo "Invalid username or password. You have ".$_SESSION['attempts']." Attempt(s) left.";
        }
    }
}
?>

```

<!--HTML form for user input →

```

<form method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username" required><br><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required><br><br>

```

```
<input type="submit" name="submit" value="Log In">
</form>
```

Q. 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.

Ans:

Import pandas as pd

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

 'Gender': ['Male', 'Male', 'Female', 'Female', 'Male', 'Male', 'Female', 'Female', 'Male',
 'Female'],

 'Age': [19, 35, 26, 27, 19, 27, 32, 25, 33, 45],

 'Estimated Salary': [19000, 20000, 43000, 57000, 76000, 58000, 82000, 32000, 69000, 65000],

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

Df = pd.DataFrame(data)

From sklearn.model_selection import train_test_split

X = df.iloc[:, 1:4].values

Y = df.iloc[:, 4].values

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

Lr=LogisticRegression(random_state=0)

Lr.fit(X_train, y_train)

Observation = [[0, 30, 87000]]

Prediction = Lr.predict(observation)

Print(prediction)

Observations = [[0, 30, 87000], [1, 50, 45000], [1, 22, 30000]]

Predictions = Lr.predict(observations)

Print(predictions)

@Slip-4

Q. 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]

Ans:

Firstpage.php

```
<?php
```

```
Session_start();
```

```
?>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>Employee Details</title>
```

```
</head>
```

```
<body>
```

```
    <h1>Employee Details</h1>
```

```
    <form method="POST" action="Secondpage.php">
```

```
        <label for="eno">Employee No:</label>
```

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

```
        <label for="ename">Employee Name:</label>
```

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

```
        <label for="address">Address:</label>
```

```
        <textarea id="address" name="address"></textarea><br><br>
```

```
        <input type="submit" value="Next">
```

```
    </form>
```

```
</body>
```

```
</html>
```

```
<?php
```

```
If(isset($_POST['eno']) && isset($_POST['ename']) && isset($_POST['address'])) {
```

```
    $_SESSION['eno'] = $_POST['eno'];
```

```
    $_SESSION['ename'] = $_POST['ename'];
```

```
    $_SESSION['address'] = $_POST['address'];
```

```
}
```

```
?>
```

Secondpage.php

```
<?php
Session_start();

?>

<!DOCTYPE html>

<html>

<head>

    <title>Earnings</title>

</head>

<body>

    <h1>Earnings</h1>

    <form method="POST" action="thirdpage.php">

        <label for="basic">Basic:</label>

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

        <label for="da">DA:</label>

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

        <label for="hra">HRA:</label>

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

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

    </form>

</body>

</html>

<?php
If(isset($_POST['basic']) && isset($_POST['da']) && isset($_POST['hra'])) {

    $_SESSION['basic'] = $_POST['basic'];

    $_SESSION['da'] = $_POST['da'];

    $_SESSION['hra'] = $_POST['hra'];

}

?>
```

Thirdpage.php

```
<?php
```

```
Session_start();
```

```
$total = $_SESSION['basic'] + $_SESSION['da'] + $_SESSION['hra'];
```

```
?>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>Employee Information</title>
```

```
</head>
```

```
<body>
```

```
    <h1>Employee Information</h1>
```

```
    <p><strong>Employee No:</strong> <?php echo $_SESSION['eno']; ?></p>
```

```
    <p><strong>Employee Name:</strong> <?php echo $_SESSION['ename']; ?></p>
```

```
    <p><strong>Address:</strong> <?php echo $_SESSION['address']; ?></p>
```

```
    <p><strong>Basic:</strong> <?php echo $_SESSION['basic']; ?></p>
```

```
    <p><strong>DA:</strong> <?php echo $_SESSION['da']; ?></p>
```

```
    <p><strong>HRA:</strong> <?php echo $_SESSION['hra']; ?></p>
```

```
    <p><strong>Total Earnings:</strong> <?php echo $total; ?></p>
```

```
</body>
```

```
</html>
```

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

Ans:

Import pandas as pd

Import random

From sklearn.linear_model import LinearRegression

Fish_species = ['Tuna', 'Salmon', 'Trout', 'Bass', 'Sardine', 'Cod', 'Mackerel']

Weights = []

```

For i in range(50):
    Fish_weight = []
    For j in range(7):
        Weight = random.randint(1, 20)
        Fish_weight.append(weight)
    Weights.append(fish_weight)
Df = pd.DataFrame(weights, columns=fish_species)
X = df.iloc[:, :-1] # independent variables
Y = df.iloc[:, -1] # target variable
Model = LinearRegression()
Model.fit(X, y)
New_fish = [[10, 12, 15, 7, 4, 8]] # example input
Predicted_weight = model.predict(new_fish)
Print("Predicted weight:", predicted_weight)

```

@Slip-5

Q. 1) Create XML file named "Item.xml" with item-name, item-rate, item quantity Store the details of 5 Items of different Types.

Ans:

Item.xml

```

<items>
  <item type="Electronics">
    <name>Television</name>
    <rate>500</rate>
    <quantity>10</quantity>
  </item>
  <item type="Clothing">
    <name>Shirt</name>
    <rate>50</rate>
    <quantity>20</quantity>
  </item>
  <item type="Grocery">

```

```

<name>Rice</name>

<rate>40</rate>

<quantity>30</quantity>
</item>

<item type="Books">

<name>Harry Potter and the Philosopher's Stone</name>

<rate>20</rate>

<quantity>50</quantity>
</item>

<item type="Sports">

<name>Football</name>

<rate>100</rate>

<quantity>5</quantity>
</item>
</items>

```

Q. 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.

Ans:

```

Import pandas as pd

From sklearn.datasets import load_iris

From sklearn.linear_model import LogisticRegression

From sklearn.model_selection import train_test_split

From sklearn.metrics import accuracy_score

Iris = load_iris()

Df = pd.DataFrame(iris.data, columns=iris.feature_names)

Df['target'] = iris.target

Print("Statistical details of Iris-setosa:")

Print(df[df['target']==0].describe())

```

```

Print("Statistical details of Iris-versicolor:")
Print(df[df['target']==1].describe())
Print("Statistical details of Iris-virginica:")
Print(df[df['target']==2].describe())
X = df.iloc[:, :-1]
Y = df.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
Logreg = LogisticRegression()
Logreg.fit(X_train, y_train)
Y_pred = logreg.predict(X_test)
Accuracy = accuracy_score(y_test, y_pred)
Print("Accuracy of the logistic regression model:", accuracy)

```

@Slip-6

Q. 1) Write PHP script to read "book.xml" file into simpleXML object. Display attributes and elements (simple_xml_load_file() function)

Ans:

```

<?php
$xml = simplexml_load_file("book.xml");
Echo "Book attributes: <br>";
Echo "ISBN: " . $xml['isbn'] . "<br>";
Echo "Publisher: " . $xml['publisher'] . "<br>";
Echo "<br>";
Echo "Book elements: <br>";
Echo "Title: " . $xml->title . "<br>";
Echo "Author: " . $xml->author . "<br>";
Echo "Description: " . $xml->description . "<br>";
?>

Book.xml file
<?xml version="1.0"?>

```

```

<book isbn="978-3-16-148410-0" publisher="Example Publisher">

<title>Example Book</title>

<author>John Doe</author>

<description>This is an example book.</description>

</book>

```

Q. 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.
TID={1:["bread","milk"],2:["bread","diaper","beer","eggs"],3:["milk","diaper","beer","coke"],4:["bread","milk","diaper","beer"],5:["bread","milk","diaper","coke"]}

Ans:

```

Import pandas as pd

From mlxtend.preprocessing import TransactionEncoder

From mlxtend.frequent_patterns import apriori, association_rules

TID =
{1:["bread","milk"],2:["bread","diaper","beer","eggs"],3:["milk","diaper","beer","coke"],4:["bread",
"milk","diaper","beer"],5:["bread","milk","diaper","coke"]}

Transactions = []

For key, value in TID.items():
    Transactions.append(value)

Te = TransactionEncoder()

Te_ary = te.fit_transform(transactions)

Df = pd.DataFrame(te_ary, columns=te.columns_)

Min_sup_values = [0.2, 0.4, 0.6]

For min_sup in min_sup_values:
    Frequent_itemsets = apriori(df, min_support=min_sup, use_colnames=True)
    Rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
    Print("Min_sup:", min_sup)
    Print("Frequent Itemsets:")
    Print(frequent_itemsets)
    Print("Association Rules:")
    Print(rules)

```

@Slip-7

Q. 1) Write a PHP script to read "Movie.xml" file and print all MovieTitle and ActorName of file using OMDocument Parser. "Movie.xml" file should contain following information with at least 5 records With values. MovieNo, MovieTitle, ActorName, ReleaseYear.

Ans:

Php file

```
<?php
$xml = new DOMDocument();
$xml->load('Movie.xml');
$movies = $xml->getElementsByTagName('MovieInfo');
Foreach ($movies as $movie) {
    Echo "Movie Title: " . $movie->getElementsByTagName('MovieTitle')[0]->textContent . " ";
    Echo "Actor Name: " . $movie->getElementsByTagName('ActorName')[0]->textContent . "<br>";
}
?>
```

XML file

```
<?xml version="1.0"?>
<MovieList>
    <MovieInfo>
        <MovieNo>1</MovieNo>
        <MovieTitle>The Shawshank Redemption</MovieTitle>
        <ActorName>Tim Robbins</ActorName>
        <ReleaseYear>1994</ReleaseYear>
    </MovieInfo>
    <MovieInfo>
        <MovieNo>2</MovieNo>
        <MovieTitle>The Godfather</MovieTitle>
        <ActorName>Marlon Brando</ActorName>
        <ReleaseYear>1972</ReleaseYear>
    </MovieInfo>
```

```

<MovieInfo>
  <MovieNo>3</MovieNo>
  <MovieTitle>The Dark Knight</MovieTitle>
  <ActorName>Christian Bale</ActorName>
  <ReleaseYear>2008</ReleaseYear>
</MovieInfo>
<MovieInfo>
  <MovieNo>4</MovieNo>
  <MovieTitle>The Godfather: Part II</MovieTitle>
  <ActorName>Al Pacino</ActorName>
  <ReleaseYear>1974</ReleaseYear>
</MovieInfo>
<MovieInfo>
  <MovieNo>5</MovieNo>
  <MovieTitle>12 Angry Men</MovieTitle>
  <ActorName>Henry Fonda</ActorName>
  <ReleaseYear>1957</ReleaseYear>
</MovieInfo>
</MovieList>

```

Q. 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. .

Ans:

```

Import pandas as pd

From mlxtend.preprocessing import TransactionEncoder

From mlxtend.frequent_patterns import apriori, association_rules

Df = pd.read_csv('Market_Basket_Optimisation.csv', header=None)

Df.dropna(inplace=True)

Te = TransactionEncoder()

```

```

Te_ary = te.fit(df.values).transform(df.values)
Df = pd.DataFrame(te_ary, columns=te.columns_)
Frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)
Rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
Print("Original Dataset:\n")
Print(df.head())
Print("\nFrequent Itemsets:\n")
Print(frequent_itemsets)
Print("\nAssociation Rules:\n")
Print(rules)

```

@Slip-8

Q. 1) Write a JavaScript to display message 'Exams are near, have you started preparing for?' (use alertBox) and Accept any two numbers from user and display addition of two number .(Use Prompt and Confirm box)

Ans:

```

Alert('Exams are near, have you started preparing for?');
Let num1 = prompt('Enter first number:');
Let num2 = prompt('Enter second number:');
Let confirmMsg = `Are you sure you want to add ${num1} and ${num2}?`;
Let confirmResult = confirm(confirmMsg);
If (confirmResult) {
    Num1 = parseInt(num1);
    Num2 = parseInt(num2);
    Let sum = num1 + num2;
    Alert(`The sum of ${num1} and ${num2} is ${sum}.`);
}

```

Q. 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.

Ans:

```

Import pandas as pd

From mlxtend.preprocessing import TransactionEncoder

From mlxtend.frequent_patterns import apriori, association_rules

Df = pd.read_csv('market_basket.csv')

Df.dropna(inplace=True)

Te = TransactionEncoder()

Te_ary = te.fit(df.values).transform(df.values)

Df = pd.DataFrame(te_ary, columns=te.columns_)

Frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)

Rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

Print("Dataset information:")

Print(df.info())

Print("\nFrequent itemsets:")

Print(frequent_itemsets)Print("\nAssociation rules:")

Print(rules)

```

@Slip-9

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

Ans:

```

Function validateForm() {

    Var username = document.forms["membershipForm"]["username"].value;

    Var password = document.forms["membershipForm"]["password"].value;

    If (username == "") {

        Return false;

    }

    If (password == "") {

        Return false;

    }

    Return true;

}

```

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

Ans:

```
Items=['item1','item2','item3','item4']
```

```
Transactions = [ ['item1', 'item2', 'item3'],  
                 ['item2', 'item3'],  
                 ['item1', 'item2', 'item4'],  
                 ['item1', 'item4'],  
                 ['item2', 'item3', 'item4'],  
                 ['item1', 'item3', 'item4'],  
                 ['item1', 'item2'],  
                 ['item1', 'item3'],  
                 ['item3', 'item4'],  
                 ['item2', 'item4']  
]
```

```
From mlxtend.preprocessing import TransactionEncoder
```

```
From mlxtend.frequent_patterns import apriori, association_rules
```

```
Te = TransactionEncoder()
```

```
Te_ary = te.fit_transform(transactions)
```

```
Df = pd.DataFrame(te_ary, columns=te.columns_)
```

```
Frequent_itemsets = apriori(df, min_support=0.3, use_colnames=True)
```

```
Association_rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

```
Print(frequent_itemsets)
```

```
Print(association_rules)
```

@Slip-11

Q. 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, onmouseover, onclick, onmouseup)

Ans:

```
<!DOCTYPE html>

<html>

<head>

    <title>JavaScript Example</title>

    <style>

        #name {

            Font-size: 14px;

            Color: black;

        }

    </style>

</head>

<body>

    <input type="text" id="name" onblur="changeStyle()" onmouseover="changeSize()"
onmouseout="resetSize()" onmousedown="changeColor()" onmouseup="resetColor()">

    <img id="img" src=https://via.placeholder.com/150 onload="changeImageSize()">

    <script>

        Function changeStyle() {

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

            If (name) {

                Document.getElementById("name").style.fontSize = "18px";

                Document.getElementById("name").style.color = "red";

            } else {

                Document.getElementById("img").style.display = "block";

            }

        }

        Function changeSize() {

            Document.getElementById("name").style.fontSize = "16px";

        }

        Function resetSize() {

            Document.getElementById("name").style.fontSize = "14px";

        }

    </script>

</body>

</html>
```

```

        Function changeColor() {
            Document.getElementById("name").style.color = "blue";
        }
        Function resetColor() {
            Document.getElementById("name").style.color = "red";
        }
        Function changelImageSize() {
            Document.getElementById("img").style.width = "200px";
            Document.getElementById("img").style.height = "200px";
        }
    </script>
</body>
</html>

```

Q 2).Create the above 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.TID={1:["butter","bread","milk"],2=["butter","flour","milk","suger"],3=["butter","eggs","milk","salt"],4=["eggs"],5=["butter","flour","milk","salt"]}

Ans:

```
Import pandas as pd
```

```
From mlxtend.preprocessing import TransactionEncoder
```

```
From mlxtend.frequent_patterns import apriori, association_rules
```

```
Dataset = [['butter', 'bread', 'milk'], ['butter', 'flour', 'milk', 'sugar'], ['butter', 'eggs', 'milk', 'salt'],
['eggs'], ['butter', 'flour', 'milk', 'salt']]
```

```
Df = pd.DataFrame(dataset)
```

```
Te = TransactionEncoder()
```

```
Te_ary = te.fit(dataset).transform(dataset)
```

```
Df = pd.DataFrame(te_ary, columns=te.columns_)
```

```
Min_sup_values = [0.4, 0.3, 0.2]
```

```
For min_sup in min_sup_values:
```

```

Frequent_itemsets = apriori(df, min_support=min_sup, use_colnames=True)
Print("Frequent Itemsets with minimum support of", min_sup)
Print(frequent_itemsets)
Rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
Print("Association Rules with minimum support of", min_sup)
Print(rules)

```

@Slip-17

Q. 1) Write a Java Script Program to show Hello Good Morning message onload event using alert box And display the Student registration form.

Ans:

```

<!DOCTYPE html>

<html>

<head>

<title>Student Registration Form</title>

    <script>

        Window.onload = function() {

            Alert("Hello Good Morning!");

        };

    </script>

</head>

<body>

    <h1>Student Registration Form</h1>

    <form>

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

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

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

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

        <label for="phone">Phone:</label>

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

        <label for="address">Address:</label>
    
```

```

        <textarea id="address" name="address" required></textarea><br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>

```

Q. 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.

Ans:

Import re

From nltk.tokenize import sent_tokenize

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."

Text = re.sub('[^A-Za-z]+', '', text)

Sentences = sent_tokenize(text)

Scores = {}

For sentence in sentences:

Words = sentence.split()

Score = len(words)

Scores[sentence] = score

Sorted_sentences = sorted(scores.items(), key=lambda x: x[1], reverse=True)

Summary_sentences = [sentence[0] for sentence in sorted_sentences[:2]]

Summary = " ".join(summary_sentences)

Print(summary)

@Slip-18

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

Ans:

```

<!DOCTYPE html>

<html>

<head>

    <title>Fibonacci Numbers</title>

    <script>

        Function generateFibonacci() {

            Var input = document.getElementById("inputNumber").value;

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

            Var n = parseInt(input);

            Var fib = [];

            Fib[0] = 0;

            Fib[1] = 1;

            For (var i = 2; i <= n; i++) {

                Fib[i] = fib[i-1] + fib[i-2];

            }

            Output.innerHTML = "Fibonacci sequence up to " + n + ": " + fib.join(", ");

        }

    </script>

</head>

<body>

    <h1>Fibonacci Numbers</h1>

    <p>Enter a number:</p>

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

    <button onclick="generateFibonacci()">Generate Fibonacci</button>

    <p id="output"></p>

</body>

</html>

```

Q. 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 Txt.

Ans:

```

Import nltk

From nltk.corpus import stopwords

From nltk.tokenize import word_tokenize, sent_tokenize

From nltk.probability import FreqDist

Import matplotlib.pyplot as plt

From wordcloud import WordCloud

Nltk.download('stopwords')

Words = word_tokenize(text.lower())

Sentences = sent_tokenize(text)

Stop_words = set(stopwords.words('english'))

Filtered_words = [word for word in words if word.casefold() not in stop_words]

Fdist = FreqDist(filtered_words)

Fdist.plot(30, cumulative=False)

Plt.show()

Wordcloud = WordCloud(width = 800, height = 800,
                        Background_color = 'white',
                        Stopwords = stop_words,
                        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()

```

@Slip-19

Q. 1) Write a Java Script Program to validate use name and password on submit event.

Ans:

```

<!DOCTYPE html>

<html>

<head>

<title>Validate User Name and Password</title>

```

```

<script>
Function validateForm() {
    Var username = document.forms["myForm"]["username"].value;
    Var password = document.forms["myForm"]["password"].value;

    If (username == "") {
        Alert("Username must be filled out");
        Return false;
    }

    If (password == "") {
        Alert("Password must be filled out");
        Return false;
    }
}
</script>
</head>
<body>
<h2>Validate User Name and Password</h2>
<form name="myForm" onsubmit="return validateForm()" method="post">
    <label for="username">Username:</label>
    <input type="text" id="username" name="username"><br><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password"><br><br>
    <input type="submit" value="Submit">
</form>
</body>
</html>

```

Q. 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.

Ans:

Import pandas as pd

From textblob import TextBlob

From wordcloud import WordCloud, STOPWORDS

Import matplotlib.pyplot as plt

Df = pd.read_csv('movie_review.csv')

Df['Sentiment'] = df['Review'].apply(lambda x: TextBlob(x).sentiment.polarity)

Pos_df = df[df['Sentiment'] > 0.2]

Wordcloud = WordCloud(width = 800, height = 800,

Background_color = 'white',

Stopwords = STOPWORDS,

Min_font_size = (10).generate(' '.join(pos_df['Review']))Plt.figure(figsize = (8, 8), facecolor = None)

Plt.imshow(wordcloud)

Plt.axis("off")

Plt.tight_layout(pad = 0)

Plt.show()

@Slip-20

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

Ans:

```
<?xml version="1.0"?>
```

```
<students>
```

```
<student>
```

```
<name>John Doe</name>
```

```
<age>21</age>
```

```
<gender>Male</gender>
```

```
<major>Computer Science</major>
```

```
<gpa>3.8</gpa>
```

```
</student>
```

```
<student>
```

```
<name>Jane Smith</name>
<age>19</age>
<gender>Female</gender>
<major>Business</major>
<gpa>3.5</gpa>
</student>
<student>
  <name>Tom Johnson</name>
  <age>20</age>
  <gender>Male</gender>
  <major>Engineering</major>
  <gpa>3.2</gpa>
</student>
<student>
  <name>Sara Lee</name>
  <age>22</age>
  <gender>Female</gender>
  <major>Psychology</major>
  <gpa>3.6</gpa>
</student>
<student>
  <name>Mike Brown</name>
  <age>18</age>
  <gender>Male</gender>
  <major>Education</major>
  <gpa>3.4</gpa>
</student>
</students>
```

Q. 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.

Ans:

```
Import nltk
```

```
From nltk.corpus import stopwords
```

```
Nltk.download('stopwords')
```

```
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."
```

```
Tokens = nltk.word_tokenize(text)
```

```
Stop_words = set(stopwords.words('english'))
```

```
Filtered_tokens = [word for word in tokens if not word.lower() in stop_words]
```

```
Print(filtered_tokens)
```

@Slip-25

Q. 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".

Ans:

```
<?php
```

```
$doc = new DOMDocument();
```

```
$cricketTeam = $doc->createElement("CricketTeam");
```

```
$teamAustralia = $doc->createElement("Team");
```

```
$teamAustralia->setAttribute("country", "Australia");
```

```
$player1 = $doc->createElement("player", "Steve Smith");
```

```
$teamAustralia->appendChild($player1);
```

```
$runs1 = $doc->createElement("runs", "7090");
```

```

$teamAustralia->appendChild($runs1);
$wicket1 = $doc->createElement("wicket", "17");
$teamAustralia->appendChild($wicket1);
$cricketTeam->appendChild($teamAustralia);
$teamIndia = $doc->createElement("Team");
$teamIndia->setAttribute("country", "India");
$player2 = $doc->createElement("player", "Virat Kohli");
$teamIndia->appendChild($player2);
$runs2 = $doc->createElement("runs", "12169");
$teamIndia->appendChild($runs2);
$wicket2 = $doc->createElement("wicket", "4");
$teamIndia->appendChild($wicket2);
$category = $doc->createElement("category", "Captain");
$teamIndia->appendChild($category);
$cricketTeam->appendChild($teamIndia);
$doc->appendChild($cricketTeam);
$doc->save("cricket.xml");
Echo "Elements added successfully!";
?>

```

Q. 2) Consider the following dataset : https://www.kaggle.com/datasets/seungguini/youtube-commentsfor-covid19-relatedvideos?select=covid_2021_1.csv

- i. Write a Python script for the following :
- ii. Read the dataset and perform data cleaning operations on it.
- iii. Tokenize the comments in words. lii. Perform sentiment analysis and find the percentage of positive, negative and neutral comments..

Ans:

```
Import pandas as pd
```

```
Import nltk
```

```
From nltk.sentiment.vader import SentimentIntensityAnalyzer
```

```

Df = pd.read_csv('covid_2021_1.csv')
Df.dropna(inplace=True)
Df.drop_duplicates(subset='Comment', inplace=True)
Nltk.download('punkt')
Df['tokens'] = df['Comment'].apply(nltk.word_tokenize)
Nltk.download('vader_lexicon')
Sia = SentimentIntensityAnalyzer()
Df['sentiment'] = df['Comment'].apply(lambda x: sia.polarity_scores(x)['compound'])
Total_comments = len(df)
Positive_comments = len(df[df['sentiment'] > 0])
Negative_comments = len(df[df['sentiment'] < 0])
Neutral_comments = len(df[df['sentiment'] == 0])
Positive_percentage = (positive_comments / total_comments) * 100
Negative_percentage = (negative_comments / total_comments) * 100
Neutral_percentage = (neutral_comments / total_comments) * 100
Print('Total Comments:', total_comments)
Print('Positive Comments:', positive_comments, '(', positive_percentage, '%')
Print('Negative Comments:', negative_comments, '(', negative_percentage, '%')
Print('Neutral Comments:', neutral_comments, '(', neutral_percentage, '%')

```

@Slip-30

Q. 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"

Ans:

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

<Bookstore>

<Yoga>

<Book>

<Book_Title>Light on Yoga</Book_Title>

<Book_Author>B.K.S. Iyengar</Book_Author>

<Book_Price>20.99</Book_Price>

</Book>

<Book>

<Book_Title>The Yoga Bible</Book_Title>

<Book_Author>Christina Brown</Book_Author>

<Book_Price>15.50</Book_Price>

</Book>

</Yoga>

<Story>

<Book>

<Book_Title>The Alchemist</Book_Title>

<Book_Author>Paulo Coelho</Book_Author>

<Book_Price>12.99</Book_Price>

</Book>

<Book>

<Book_Title>The Da Vinci Code</Book_Title>

<Book_Author>Dan Brown</Book_Author>

<Book_Price>14.75</Book_Price>

```

</Book>
</Story>
<Technical>
  <Book>
    <Book_Title>Python for Data Science Handbook</Book_Title>
    <Book_Author>Jake VanderPlas</Book_Author>
    <Book_Price>28.99</Book_Price>
  </Book>
  <Book>
    <Book_Title>Cracking the Coding Interview</Book_Title>
    <Book_Author>Gayle Laakmann McDowell</Book_Author>
    <Book_Price>23.50</Book_Price>
  </Book>
</Technical>
</Bookstore>

```

Q. 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.

Ans:

```
Transactions = [['eggs', 'milk', 'bread'], ['eggs', 'apple'], ['milk', 'bread'], ['apple', 'milk'], ['milk', 'apple', 'bread']]
```

```
Item_to_num = {'eggs': 1, 'milk': 2, 'bread': 3, 'apple': 4}
```

```
Numeric_transactions = []
```

For transaction in transactions:

```
    Numeric_transaction = [item_to_num[item] for item in transaction]
```

```
    Numeric_transactions.append(numeric_transaction)
```

```
Print(numeric_transactions)
```

```
From mlxtend.frequent_patterns import apriori, association_rules
```

```
Frequent_itemsets = apriori(numeric_transactions, min_support=0.4, use_colnames=True)
```

```
Rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)
```

```
Print(frequent_itemsets)
```

Print(rules)