LAB ASSIGNMENT

On

(KCA 351 "AI Lab")

G.L. BAJAJ COLLEGE OF TECHNOLOGY & MANAGEMENT

GL BAJAJ, GREATER NOIDA

Session: 2024 - 2025

Submitted To:

Dr. Bishwajeet Pandey

Submitted By:

Student Name : Pranav Verma

Roll. No. : 2312000140123

Program : MCA (3rd Sem.)

Section : B2

Department of Master of Computer Applications

GL BAJAJ
College of Technology & Management
FIND YOUR SPARK

TABLE OF CONTENTS

GL BAJAJ
College of Technology & Management
FIND YOUR SPARK

Syllabus

KCA351: Artificial Intelligence Lab		
Course Outcome (CO)		**Bloom's Knowledge Level (KL)**
At the end of course, the student will be able to understand		
CO 1	Study and understand AI tools such as Python / MATLAB.	K_1,K_2
CO 2	Apply AI tools to analyze and solve common AI problems.	K_3, K_4
CO 3	Implement and compare various AI searching algorithms.	K_6
CO 4	Implement various machine learning algorithms.	K_6
CO 5	Implement various classification and clustering techniques.	K_6

DETAILED SYLLABUS

1. Installation and working on various AI tools such as Python / MATLAB.
2. Programs to solve basic AI problems.
3. Implementation of different AI searching techniques.
4. Implementation of different game playing techniques.
5. Implementation of various knowledge representation techniques.
6. Program to demonstrate the working of Bayesian network.
7. Implementation of pattern recognition problems such as handwritten character/ digit recognition, speech recognition, etc.
8. Implementation of different classification techniques.
9. Implementation of various clustering techniques.
10. Natural language processing tool development.

Note:
TheInstructormayadd/delete/modify/tuneexperiments,whereverhe/shefeelsinajustifiedmanner.

Evaluation Scheme

S. No.	Subject Code	Name of the Subject	Periods			Evaluation Scheme				Subject Total	Credit
			L	T	P	Sessional Assessment			ESE		
						C T	T A	Total			
1	KCA-351	Artificial Intelligence Lab	0	0	3	30	20	50	50	100	2

GL BAJAJ
College of Technology & Management
FIND YOUR SPARK

Program outcomes (POs)

1. Computational Knowledge: Apply knowledge of computing fundamentals, computing specialization, mathematics, and domain knowledge appropriate for the computing specialization to the abstraction and conceptualization of computing models from defined problems and requirements.

2. Problem Analysis: Identify, formulate, research literature, and solve complex computing problems reaching substantiated conclusions using fundamental principles of mathematics, computing sciences, and relevant domain disciplines.

3. Design /Development of Solutions: Design and evaluate solutions for complex computing problems, and design and evaluate systems, components, or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.

4. Conduct investigations of complex Computing problems**:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern Tool Usage: Create, select, adapt and apply appropriate techniques, resources, and modern computing tools to complex computing activities, with an understanding of the limitations.

6. Professional Ethics: Understand and commit to professional ethics and cyber regulations, responsibilities, and norms of professional computing practices.

7. Life-long Learning: Recognize the need, and have the ability, to engage in independent learning for continual development as a computing professional.

8. Project management and finance: Demonstrate knowledge and understanding of the computing and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

9. Communication Efficacy: Communicate effectively with the computing community, and with society at large, about complex computing activities by being able to comprehend and write effective reports, design documentation, make effective presentations, and give and understand clear instructions.

10. Societal and Environmental Concern: Understand and assess societal, environmental, health, safety, legal, and cultural issues within local and global contexts, and the consequential responsibilities relevant to professional computing practices.

11. Individual and Team Work: Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary environments.

12. Innovation and Entrepreneurship Identify a timely opportunity and using innovation to pursue that opportunity to create value and wealth for the betterment of the individual and society at large.

Practical Outcomes:

	At the end of the course , the student will be able to	Bloom's Level
PrO1	Students will be able to study and understand AI tools such as Python ,Google Colab and Jupyter Notebook	K1,K2
PrO2	Students will be able to apply AI tools to analyze and solve common AI problems	K3, K4
PrO3	Students will be able to implement and compare various AI searching algorithms.	K6
PrO4	Students will be able to implement various machine learning algorithms.	K6
PrO5	Students will be able to Implement various classification and clustering techniques.	K6

GL BAJAJ
College of Technology & Management
FIND YOUR SPARK

Mapping of Program Outcomes with Practical Objectives

Practical Outcomes	PO1	PO 2	PO 3	PO 4	PO5	PO6	PO 7	PO8	PO9	PO10	PO11	PO1 2	PSO 1	PS02
PrO1	2	2	1	1	1	2	2	-	2	1	1	-	2	1
PrO2	3	2	1	2	1	2	2	1	1	2	2	2	2	-
PrO3	2	2	1	1	1	2	2	1	1	-	-	1	1	1
PrO4	3	3	2	2	1	1	3	2	2	2	2	2	2	2
PrO5	2	2	2	-	1	2	2	1	2	1	1	1	2	2

(Put 1,2,3 for mapping)

1-> slight(low) 2-> Moderate(Medium) 3: Substantial(High)

GL BAJAJ College of Technology & Management
FIND YOUR SPARK

Session Plan

COURSE:	MCA
TITLE:	Artificial Intelligence Lab
CREDIT:	2
USED TOOLS:	Python , Google Colab and Jupyter Notebook
PREREQUISITES COURSES:	Fundamental of Mathematics, Calculus, Probability and Linear Algebra
TEXT BOOK(S) AND/OR REQUIRED MATERIALS:	1. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Geron Aurelien 2. Artificial Intelligence: A Modern Approach : Peter Norvig · Stuart J. Russell 3. Machine Learning: Tom Mitchell
WEB RESOURCES:	https://scikit-learn.org/stable/ https://www.javatpoint.com/python-tutorial

GL BAJAJ
College of Technology & Management
FIND YOUR SPARK

LABORATORY POLICIES AND REPORT FORMAT

1. **Lab reports should be submitted on A4 paper.** Your report is a professional presentation of your work in the lab. Neatness, organization, and completeness will be rewarded. Points will be deducted for any part that is not clear.

2. The lab reports will be written individually. Please use the following format for your lab reports.

a.	**Cover Page**	Include your name, Subject Code, Subject title, Name of the College
b.	**Evaluation Sheet**	Gives your internal mark split –up
c.	**Index Sheet**	Includes the name of all the experiments
d.	**Experiment documentation**	It includes experiment name, date, objective, flowchart, algorithm, formulae used, Model calculation, problem solution, simulated output and print-outs
e.	**Post Lab question**	Should be written after completing the experiments**.**

3. Your work must be original and prepared independently. However, if you need any guidance or have any questions or problems, please do not hesitate to approach your staff in-charge. **The students should follow the dress code in the Lab session.**

4. Labs will be graded as per the following grading/marks policy:

5. **Reports Due Dates**: Reports should be submitted immediately after next week of the experiment. A late lab report will have 20% of the points deducted for being one day late .If a report is 3 days late, a grade/marks of D/0 will be assigned.

.

GL BAJAJ
College of Technology & Management
FIND YOUR SPARK

List of Experiments

Minimum Ten out of 15 to be added in Lab File

1.	Write a program in Python to implement Breadth First Search.
2.	Write a program in Python to implement Depth First Search.
3.	Write a program in Python to implement Best First Search.
4.	Implement Android Malware Detection using Logistic Regression and XGBoost Classifier
5.	Implement ANN using FASHION MNIST Dataset
6.	To create a WordNET using Natural Language Processing and Python
7.	To Read SQL Database in Jupyter Notebook
8.	Exploratory Data Analysis or Iris Dataset
9.	Write a program in python to implement A* Algorithm.
10.	Write a program in python to implement Tic Tac Toe game.
11.	Data visualization using Iris dataset using python.
12.	Write a program in python to implement Linear Regression.
13.	Implementation of digit recognition using MNIST Data set.
14.	Implementation of Decision Tree classifier using python and either of Diabetes Dataset or Car Evaluation Dataset
15.	Implementation of K-Means clustering.

GL BAJAJ
College of Technology & Management
FIND YOUR SPARK

Internal Evaluation Method

S.No.	Item	%
1.	Attendance	10
2.	Lab Performance	30
3.	Record	10
4.	Post lab Viva-voce	30
5.	Lab Test	20

GL BAJAJ College of Technology & Management
FIND YOUR SPARK

Experiment No.:1

Objective: Python Program to implement Breadth First Search

Code:

```
graph = {
    'A' : ['B', 'C'],
    'B' : ['D', 'E'],
    'C' : ['F', 'G'],
    'D' : [],
    'E' : [],
    'F' : [],
    'G' : []
}

visited = []  # List for visited nodes.
queue = []    # Initialize a queue

def bfs(visited, graph, node):
    visited.append(node)
    queue.append(node)

    while queue:
        s = queue.pop(0)

        if s == 'F':  # Stop the search when 'F' is found
            print(s)
            break

        print(s, end=" --> ")

        for neighbour in graph[s]:
            if neighbour not in visited:
                visited.append(neighbour)
                queue.append(neighbour)

# Driver Code
print("Breadth-First Search (stops when 'F' is found):")
bfs(visited, graph, 'A')
```

Result:

```
Breadth-First Search (stops when 'F' is found):
A --> B --> C --> D --> E --> F
```

GL BAJAJ College of Technology & Management
FIND YOUR SPARK

Experiment No.:2

Objective: Python Program to implement Depth First Search

Code:

```
# Define the graph as a dictionary where each key is a node
and the value is a list of its neighbors.
GRAPH = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F', 'G'],
    'D': ['H'],
    'E': [],
    'F': [],
    'G': [],
    'H': []
}

# Depth-First Search (DFS) function
def dfs(graph, node):
    visited = set()  # Set to keep track of visited nodes
    path = []        # List to store the order of nodes
visited

    def dfs_recursive(current_node):
        visited.add(current_node)  # Mark the current node as
visited
        path.append(current_node)  # Add the current node to
the path

        # Iterate over neighbors of the current node
        for neighbor in graph.get(current_node, []):
            if neighbor not in visited:
                dfs_recursive(neighbor)  # Recursively visit
unvisited neighbors

    # Start the DFS traversal from the initial node
    dfs_recursive(node)
    return path

# Execute DFS starting from node 'A' and print the result
result = dfs(GRAPH, 'A')
print(result)
```

Result:

```
['A', 'B', 'D', 'H', 'E', 'C', 'F', 'G']
```

GL BAJAJ College of Technology & Management
FIND YOUR SPARK

Experiment No.:3

Objective: Python Program to implement Best First Search

Code:

```
graph = {
    'A': [('B', 12), ('C', 4)],
    'B': [('D', 7), ('E', 3)],
    'C': [('F', 8), ('G', 2)],
    'D': [],
    'E': [('H', 0)],
    'F': [('H', 0)],
    'G': [('H', 0)]
}

def bfs(start, target, graph):
    queue = [(start, 0)]  # Queue stores nodes along with
their weights
    visited = set()  # Using a set for fast lookup

    while queue:
        # Dequeue the next node
        current, _ = queue.pop(0)

        if current not in visited:
            print(current)
            visited.add(current)

            # If the target is found, we exit
            if current == target:
                return

            # Enqueue all the neighbors of the current node
            for neighbor, weight in graph[current]:
                if neighbor not in visited:
                    queue.append((neighbor, weight))

            # Sort the queue based on weights
            queue.sort(key=lambda x: x[1])

bfs('A', 'H', graph)
```

Result:

A
C
G
H

GL BAJAJ
College of Technology & Management
FIND YOUR SPARK

Experiment No.:4

Objective: To Implement Decision Tree classifier using python and either of Diabetes Dataset or Car Evaluation Dataset

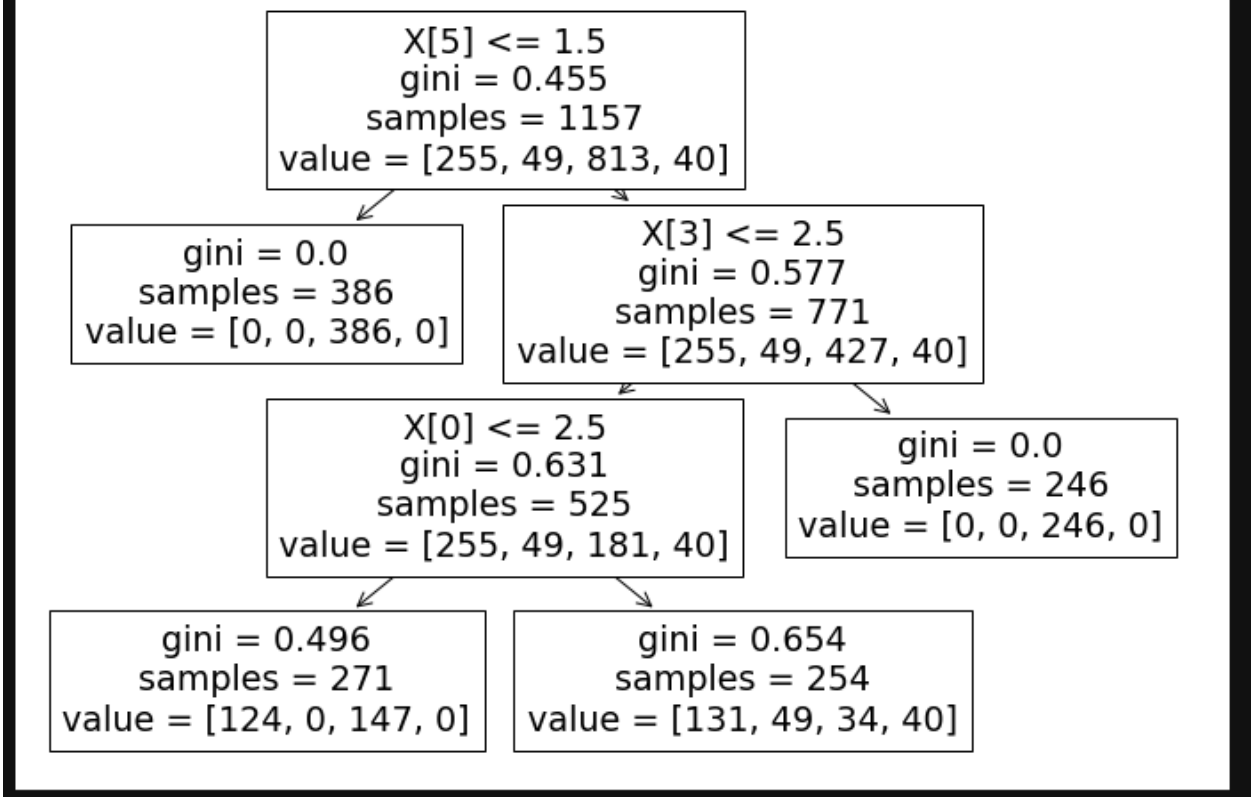
Code:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)
import matplotlib.pyplot as plt # data visualization
import seaborn as sns # statistical data visualization
%matplotlib inline

data = 'car_evaluation.csv'

df = pd.read_csv(data, header=None)
col_names = ['buying', 'maint', 'doors', 'persons',
'lug_boot', 'safety', 'class']
df.columns = col_names
col_names
df.head()
X = df.drop(['class'], axis=1) #Declare feature vector and
target variable

y = df['class']

# split X and y into training and testing sets

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.33, random_state = 42)
# check the shape of X_train and X_test

X_train.shape, X_test.shape
# import category encoders

import category_encoders as ce
# encode variables with ordinal encoding
encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors',
'persons', 'lug_boot', 'safety'])
X_train = encoder.fit_transform(X_train)
X_test = encoder.transform(X_test)
```

```
# import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
# instantiate the DecisionTreeClassifier model
clf_gini = DecisionTreeClassifier(criterion='gini',
max_depth=3, random_state=0)
# fit the model
clf_gini.fit(X_train, y_train)
#Predict the Test set results
y_pred_gini = clf_gini.predict(X_test)
#Check accuracy score
from sklearn.metrics import accuracy_score
print('Model accuracy score with criterion gini index:
{0:0.4f}'.format(accuracy_score(y_test, y_pred_gini)))
#Compare the train-set and test-set accuracy
y_pred_train_gini = clf_gini.predict(X_train)

y_pred_train_gini
print('Training-set accuracy score: {0:0.4f}'.
format(accuracy_score(y_train, y_pred_train_gini)))
# print the scores on training and test set

print('Training set score:
{:.4f}'.format(clf_gini.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(clf_gini.score(X_test,
y_test)))
#Visualize decision-trees
plt.figure(figsize=(12,8))
from sklearn import tree
tree.plot_tree(clf_gini.fit(X_train, y_train))
```

Result:

```
[Text(0.4, 0.875, 'X[5] <= 1.5\ngini = 0.455\nsamples = 1157\nvalue = [255, 49, 813, 40]'),
 Text(0.2, 0.625, 'gini = 0.0\nsamples = 386\nvalue = [0, 0, 386, 0]'),
 Text(0.6, 0.625, 'X[3] <= 2.5\ngini = 0.577\nsamples = 771\nvalue = [255, 49, 427, 40]'),
 Text(0.4, 0.375, 'X[0] <= 2.5\ngini = 0.631\nsamples = 525\nvalue = [255, 49, 181, 40]'),
 Text(0.2, 0.125, 'gini = 0.496\nsamples = 271\nvalue = [124, 0, 147, 0]'),
 Text(0.6, 0.125, 'gini = 0.654\nsamples = 254\nvalue = [131, 49, 34, 40]'),
 Text(0.8, 0.375, 'gini = 0.0\nsamples = 246\nvalue = [0, 0, 246, 0]')]
```



GL BAJAJ College of Technology & Management
FIND YOUR SPARK

Experiment No.:5

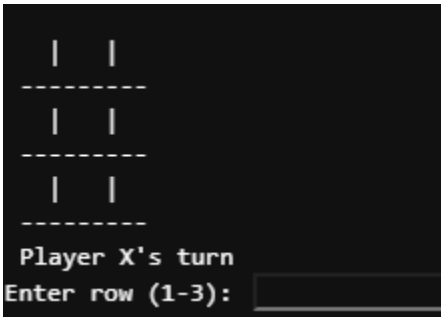
Objective: To Implement Tic Tac Toe game in python.

Code:

```
# Tic Tac Toe Game in Python

# Display the board
def display_board(board):
    print("\n")
    for row in board:
        print(" | ".join(row))
        print("-" * 9)

# Check if there's a winner
def check_winner(board, player):
    # Check rows, columns, and diagonals for a win
    for row in board:
        if all([cell == player for cell in row]):
            return True
    for col in range(3):
        if all([board[row][col] == player for row in
range(3)]):
            return True
    if all([board[i][i] == player for i in range(3)]) or
all([board[i][2 - i] == player for i in range(3)]):
        return True
    return False

# Check if the board is full
def check_full(board):
    return all([cell != " " for row in board for cell in
row])

# Main game function
def play_game():
    # Initialize the game board
    board = [[" " for _ in range(3)] for _ in range(3)]
    current_player = "X"

    # Main game loop
```

```
    while True:
        display_board(board)
        print(f"Player {current_player}'s turn")

        # Get player move
        try:
            row = int(input("Enter row (1-3): ")) - 1
            col = int(input("Enter column (1-3): ")) - 1
            if board[row][col] != " ":
                print("Cell is already taken! Choose
another.")
                continue
        except (ValueError, IndexError):
            print("Invalid input! Please enter numbers
between 1 and 3.")
            continue

        # Update the board with the current player's move
        board[row][col] = current_player

        # Check for a winner
        if check_winner(board, current_player):
            display_board(board)
            print(f"Player {current_player} wins!")
            break

        # Check for a tie
        if check_full(board):
            display_board(board)
            print("It's a tie!")
            break

        # Switch players
        current_player = "O" if current_player == "X" else
"X"

# Run the game
play_game()
```

Result:


| |

| |

| |

Player X's turn
Enter row (1-3):

GL BAJAJ
College of Technology & Management
FIND YOUR SPARK

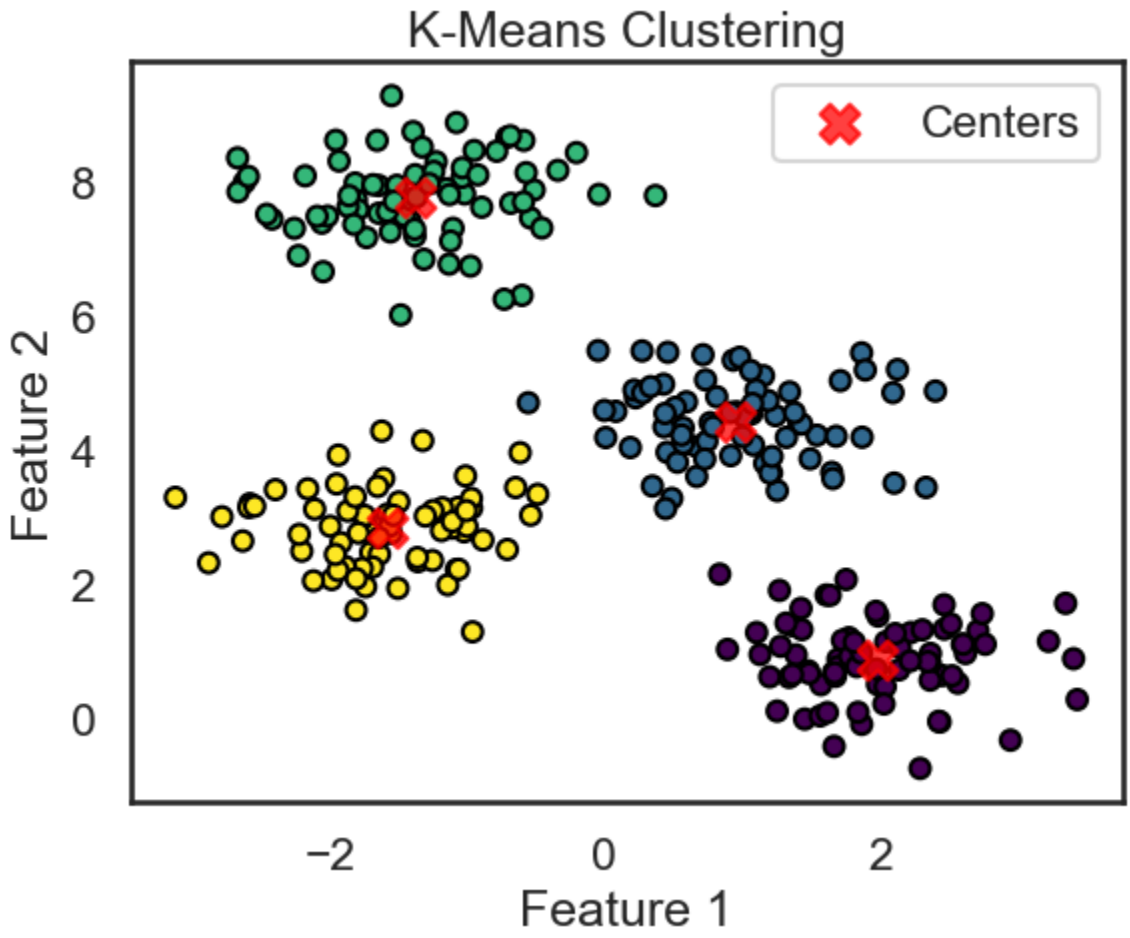
Experiment No.:6

Objective: To Implement K-Means clustering in python.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

# Generate synthetic data
np.random.seed(0)
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.6,
random_state=0)

# Define the K-Means model with number of clusters
kmeans = KMeans(n_clusters=4, random_state=0)
# Fit the model to the data
kmeans.fit(X)

# Get the cluster centers and labels
centers = kmeans.cluster_centers_
labels = kmeans.labels_

# Plot the clusters and their centers
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis',
marker='o', edgecolor='k', s=50)
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200,
alpha=0.75, marker='X', label="Centers")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()
plt.title("K-Means Clustering")
plt.show()
```

Result:



GL BAJAJ
College of Technology & Management
FIND YOUR SPARK

Experiment No.:7

Objective: To Read SQL Database in Jupyter Notebook

Code:

```
# Imports
import sqlite3 as sq3
import pandas.io.sql as pds
import pandas as pd
# Initialize path to SQLite database
path = 'data/classic_rock.db'
con = sq3.Connection(path)

# We now have a live connection to our SQL database
# Write the query
query = '''
SELECT *
FROM rock_songs where PlayCount>100 ;
'''

# Execute the query
observations = pds.read_sql(query, con)

observations.head(50)
```

Result:

	Song	Artist	Release_Year	PlayCount
0	You Shook Me All Night Long	AC/DC	1980.0	138
1	Dream On	Aerosmith	1973.0	142
2	Sweet Emotion	Aerosmith	1975.0	141
3	Walk This Way	Aerosmith	1975.0	106
4	Paranoid	Black Sabbath	1970.0	105
5	Burnin' for You	Blue Oyster Cult	1981.0	107
6	More Than a Feeling	Boston	1976.0	134
7	Peace of Mind	Boston	1976.0	132
8	I Want You to Want Me	Cheap Trick	1977.0	110
9	Hotel California	Eagles	1976.0	109
10	Cold As Ice	Foreigner	1977.0	102
11	Barracuda	Heart	1977.0	113
12	Crazy On You	Heart	1976.0	125

GL BAJAJ College of Technology & Management
FIND YOUR SPARK

Experiment No.:8

Objective: To Implement Exploratory Data Analysis or Iris Dataset

Code:

```
import os
import numpy as np
import pandas as pd
filepath = "iris_data.csv"
data = pd.read_csv(filepath)
data.head()
### BEGIN SOLUTION
# Number of rows
print(data.shape[0])

# Column names
print(data.columns.tolist())

# Data types
print(data.dtypes)
### END SOLUTION
```

Result:

```
150
['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

GL BAJAJ College of Technology & Management
FIND YOUR SPARK

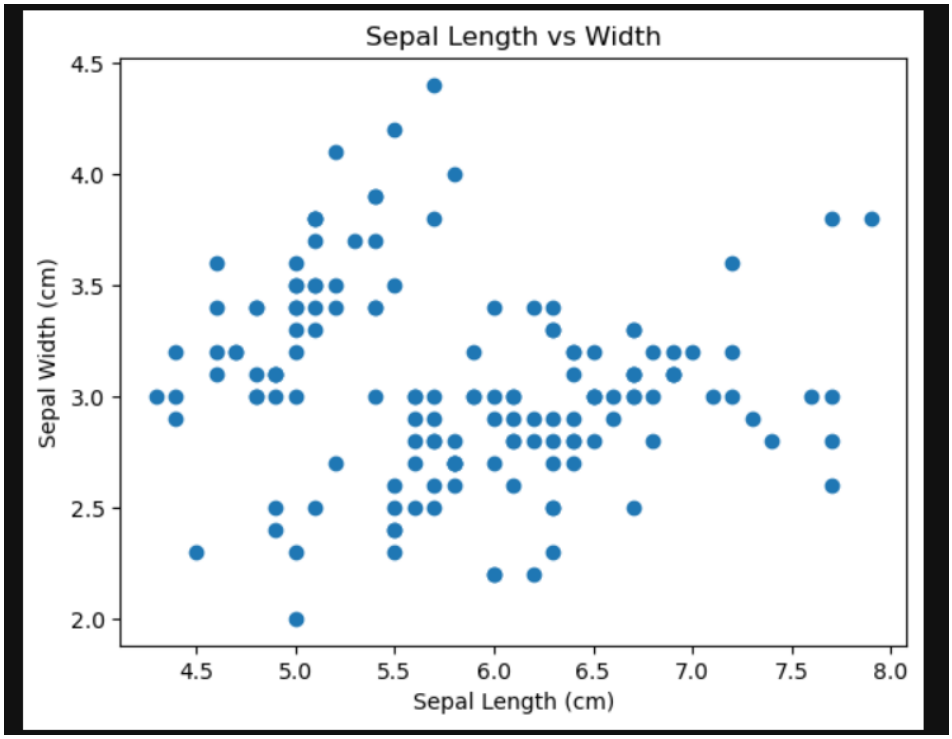
Experiment No.:9

Objective: To Implement Data visualization using Iris dataset using python

Code:

```
import os
### BEGIN SOLUTION
import matplotlib.pyplot as plt
%matplotlib inline
# A simple scatter plot with Matplotlib
ax = plt.axes()

ax.scatter(data.sepal_length, data.sepal_width)

# Label the axes
ax.set(xlabel='Sepal Length (cm)',
       ylabel='Sepal Width (cm)',
       title='Sepal Length vs Width');
### END SOLUTION
```

Result:



GL BAJAJ College of Technology & Management
FIND YOUR SPARK

Experiment No.:10

Objective: To implement Linear Regression

Code:

```
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Generate some synthetic data
np.random.seed(0)
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)  # y = 4 + 3*X +
noise

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Create and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared: {r2:.2f}")

# Get the model parameters
print(f"Intercept: {model.intercept_[0]:.2f}")
print(f"Slope: {model.coef_[0][0]:.2f}")

# Plot the results
plt.scatter(X, y, color="blue", label="Data points")
plt.plot(X_test, y_pred, color="red", linewidth=2,
label="Regression Line")
plt.xlabel("X")
```

```
plt.ylabel("y")
plt.legend()
plt.show()
```

Result:

```
Mean Squared Error: 0.92
R-squared: 0.65
Intercept: 4.21
Slope: 2.99
```