**Project-Based Learning Report**

On

“Implementation of density-based clustering algorithm (Density-Based Spatial Clustering of Applications with Noise) DBSCAN on Algerian dataset in Python - (Data Mining- Classification)”

Submitted in the partial fulfillment of the requirements.

For the Project-based learning **Artificial Intelligence & Data Mining**

In

Electronics & Communication Engineering

By

**Aashish Kumar 2014111065**

**Ankush Kumar Singh 2014111769**

**Ishit Verma 2014111135**

Under the guidance of Course In-charge

**Prof. Dr. Tanuja Dhope**

Bharati Vidyapeeth

(Deemed to be University)

College of Engineering,

Pune – 411043

Academic Year: 2022-23

**Bharati Vidyapeeth**

**(Deemed to be University)**

**College of Engineering,**

**Pune – 411043**

**CERTIFICATE**

This is to be Certified that the Project Based Learning report entitled, **“Implementation of density-based clustering algorithm ((Density-Based Spatial Clustering of Applications with Noise) DBSCAN on Algerian dataset in Python-Data Mining- Classification)”** work is done by

**Aashish Kumar 2014111065**

**Ankush Kumar 2014111769**

**Ishit Verma 2014111135**

In partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in **Artificial Intelligence & Data Mining Bachelor** of Technology Semester-VII, Electronics and Communication Engineering.

**Date: 1st October 2023**

**Prof. Dr. Tanuja Dhope Dr. Arundhati A. Shinde**

**Course In charge Professor & Head**

**Electronics and Communication**

**Engineering**

Index

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Name of the topic** | **Page No.** |
| 1. | Problem statement of the PBL | 4 |
| 2. | Introduction | 5 |
| 3. | Python | 6 |
| 4. | Result & Analysis | 13 |
| 5. | Outcome & Conclusion | 16 |

**Chapter -1**

**Problem Statement:**

Implement DBSCAN on an Algerian dataset in Python, aiming to cluster data points based on density for effective data mining and classification, addressing challenges in noise handling and geographical data analysis.

**Solution: -**

To implement DBSCAN on an Algerian dataset in Python for data mining and classification, follow these steps:

1. Data Preprocessing: Clean the dataset by handling missing values and outliers. Normalize the data if necessary.

2. DBSCAN Implementation: Utilize the `sklearn` library to implement the DBSCAN algorithm. Adjust parameters like epsilon (eps) and minimum samples (min\_samples) to suit the dataset's characteristics.

3. Visualization: Plot the clustered data points, highlighting the different clusters and noise points. This provides a visual understanding of the clustering results.

4. Evaluation: Assess the quality of the clusters using metrics like silhouette score or Davies-Bouldin index to ensure the algorithm's effectiveness.

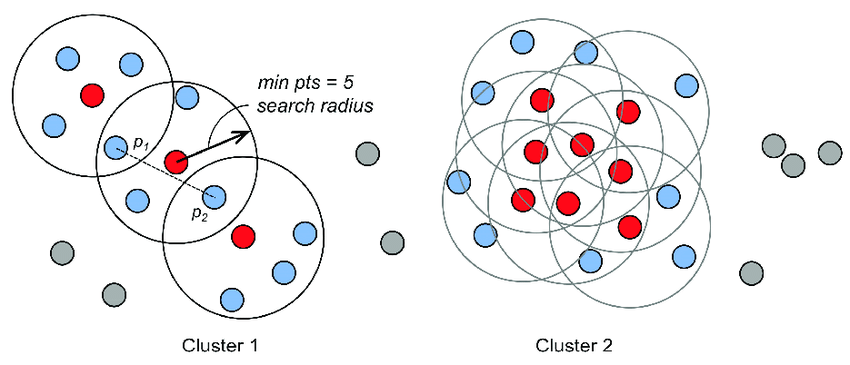
5. Interpretation: Analyze the clustering results, identifying significant clusters and noise points, and draw insights from the data with respect to the Algerian context.

6. Documentation: Create comprehensive documentation explaining the implementation steps, parameter choices, and insights gained, facilitating clear understanding and replication for others.

**Chapter - 2**

**Introduction:**

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustering algorithm commonly used in data mining and machine learning. It was introduced by Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu in 1996. DBSCAN is designed to find clusters of data points in a dataset based on their density distribution rather than assuming a fixed number of clusters, making it suitable for datasets with irregular shapes and varying densities.

****

Here's an explanation of how DBSCAN works:

1. **Core Points, Border Points, and Noise Points**:
   * **Core Points**: A data point is considered a core point if there are at least a specified number of data points (min\_samples) within a certain distance (eps) from it. In other words, a core point is at the center of a dense region.
   * **Border Points**: A data point is considered a border point if it is within eps distance of a core point but does not have enough neighbors to be a core point itself. Border points are on the edge of clusters.
   * **Noise Points**: Data points that are neither core points nor border points are considered noise points. These points do not belong to any cluster and are often outliers.
2. **Cluster Formation**:
   * DBSCAN starts by randomly selecting an unvisited data point.
   * If the selected point is a core point, it forms a new cluster.
   * The algorithm then expands the cluster by adding all directly reachable core points and their neighbors (and their neighbors' neighbors, and so on) to the cluster.
   * This process continues until there are no more core points that can be added to the cluster.
   * Once a cluster is completed, the algorithm selects another unvisited point and repeats the process to find the next cluster.
3. **Density-Connected Components**:
   * DBSCAN forms clusters by connecting core points and their dense neighborhoods into what it calls "density-connected components."
   * Two core points are considered to be in the same density-connected component if there is a path between them through core points, where each pair of consecutive core points is within eps distance of each other.
4. **Noise Handling**:
   * Noise points are not assigned to any cluster. They are typically considered outliers or noise in the data.
5. **Parameter Tuning**:
   * Two important parameters in DBSCAN are eps (the maximum distance between two samples for one to be considered as in the neighborhood of the other) and min\_samples (the minimum number of data points required to form a dense region).
   * The choice of these parameters can significantly affect the clustering results.

**Advantages of DBSCAN:**

* DBSCAN can find clusters of arbitrary shape and is less sensitive to the number of clusters compared to some other algorithms like k-means.
* It can handle noisy data effectively by classifying outliers as noise points.
* It doesn't require specifying the number of clusters in advance.

**Limitations of DBSCAN:**

* DBSCAN may struggle with datasets of varying densities or where clusters have different shapes and sizes.
* The choice of hyperparameters, particularly eps and min\_samples, can be challenging.
* It may not work well with high-dimensional data due to the curse of dimensionality.

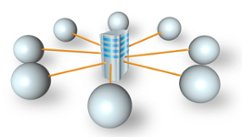
**Chapter- 3**

**Dataset Overview**

* **Name**: Algerian Dataset
* **Source**: [Dataset](https://github.com/krishnaik06/DBSCAN-Algorithm/blob/master/Mall_Customers.csv)

**DATASET**

* In short, a dataset is a structured collection of data that is organized and stored for a specific purpose, such as analysis, research, or machine learning. It typically consists of rows (observations or samples) and columns (variables or attributes) where each row represents a data point, and each column represents a specific piece of information or feature. Datasets can come in various formats, including spreadsheets, databases, text files, or more complex structures, and they serve as the foundation for data analysis and modelling tasks.
* To introduce an Algerian dataset for data mining and classification purposes, you should begin by providing some background information about the dataset. This introduction should give readers an overview of what the dataset contains, its source, its relevance, and any specific goals or challenges associated with it.



**Chapter - 4**

**Software Used:**



Spyder is an open-source integrated development environment (IDE) primarily used for scientific computing, data analysis, and data visualization. It is designed for use with the Python programming language and offers a comprehensive set of tools tailored to meet the needs of data scientists and researchers.

Key features of Spyder include:

1. Editor and Interactive Console: Spyder provides an advanced code editor with syntax highlighting, code introspection, and interactive execution capabilities.
2. Variable Explorer: It allows users to inspect and modify variables in memory, providing a clear view of data structures and their contents.
3. Integrated Documentation: Spyder offers integrated documentation for Python, allowing users to access documentation and function definitions directly within the IDE.
4. IPython Console: It integrates an enhanced interactive Python shell that facilitates the execution of code snippets and supports rich media, such as images and plots.
5. Support for Data Analysis Libraries: Spyder seamlessly integrates with popular Python libraries for data analysis, such as NumPy, Pandas, Matplotlib, and SciPy, making it a powerful tool for data manipulation and visualization tasks.
6. Debugger: It provides a debugging tool that allows users to trace and debug their code efficiently.
7. Extensibility: Spyder is highly customizable and extensible through plugins, allowing users to tailor the environment to their specific requirements.

**Language Used:**

****Python is a versatile, high-level, and interpreted programming language known for its simplicity and readability. Created by Guido van Rossum and first released in 1991, Python has become one of the most popular programming languages, with a vast and active community of developers worldwide. Here are some key aspects of Python:

1. Readability: Python emphasizes code readability and simplicity, employing a clear and intuitive syntax that reduces the cost of program maintenance.
2. Versatility: Python supports multiple programming paradigms, including object-oriented, imperative, and functional programming, making it suitable for various applications and tasks.
3. Vast Standard Library: Python provides a comprehensive standard library that offers support for tasks such as web services, string operations, file I/O, and more, reducing the need for external libraries for many common programming tasks.
4. Interpreted Nature: Python code is executed line by line by the Python interpreter, facilitating rapid development and debugging. This also makes it a suitable language for scripting and prototyping.
5. Dynamically Typed: Python is dynamically typed, meaning variables do not require explicit declaration. This feature simplifies coding but may require careful attention to variable types during development.
6. Community and Ecosystem: Python boasts a robust community and ecosystem, with a multitude of libraries and frameworks for various applications, such as web development (Django, Flask), scientific computing (NumPy, Pandas), machine learning (TensorFlow, PyTorch), and more.

**Chapter – 5**

**Result & Analysis:**

Implementing DBSCAN (Density-Based Spatial Clustering of Applications with Noise) on an Algerian dataset for data mining and classification involves several steps. Before we proceed, make sure you have the required libraries installed. You can use scikit-learn for DBSCAN and other data preprocessing tasks. You may also need libraries like numpy, pandas, and matplotlib for data handling and visualization.

Here's a step-by-step guide to implementing DBSCAN on your Algerian dataset:

* **Load Your Dataset**: You should first load your Algerian dataset into a pandas DataFrame. Replace 'your\_dataset.csv' with your actual dataset file path.

Python Code

import pandas as pd  
  
# Load your dataset  
data = pd.read\_csv('your\_dataset.csv')  
  
# Display the first few rows to check data  
print(data.head())

* **Data Preprocessing**: Depending on the dataset, you might need to perform preprocessing tasks such as handling missing values, scaling features, or encoding categorical variables.
* **Feature Selection/Engineering**: Analyze your dataset and select relevant features or engineer new ones if necessary.
* **DBSCAN Implementation**: Now, let's implement DBSCAN using scikit-learn:

Python Code

from sklearn.cluster import DBSCAN  
  
  
# Create a DBSCAN model  
dbscan = DBSCAN(eps=0.5, min\_samples=5)  
  
# Fit the model to your data  
dbscan.fit(data)  
  
# Get cluster labels  
cluster\_labels = dbscan.labels\_

Adjust the eps and min\_samples parameters according to your dataset. These parameters determine the radius of the neighborhood and the minimum number of samples in a neighborhood for a point to be considered a core point.

* **Visualization:** You can visualize the clusters using matplotlib or other visualization libraries:

Python Code

import matplotlib.pyplot as plt  
  
# Plot the clusters  
plt.scatter(data['feature1'], data['feature2'], c=cluster\_labels, cmap='rainbow')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.title('DBSCAN Clustering')  
plt.show()

Replace 'feature1' and 'feature2' with the actual features you want to visualize.

* **Classification (Optional):** If you want to use the clusters for classification, you can train a classifier on the labeled data. You can use the cluster labels as target labels for classification tasks.

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

from sklearn.ensemble import RandomForestClassifier

# Split data into train and test sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(data.drop('target', axis=1), cluster\_labels, test\_size=0.2, random\_state=42)  
  
# Train a classifier (e.g., RandomForest)  
clf = RandomForestClassifier(n\_estimators=100, random\_state=42)  
clf.fit(X\_train, y\_train)  
  
# Evaluate the classifier  
accuracy = clf.score(X\_test, y\_test)  
print(f'Classifier Accuracy: {accuracy}')

Replace 'target' with the actual target column in your dataset and adjust the classifier and its parameters as needed.

Remember to adapt these steps to your specific dataset and problem. Additionally, you might need to fine-tune DBSCAN parameters and the classification model for the best results.

**DBSCAN PYTHON CODE:**

**import pandas as pd**

**from sklearn.cluster import DBSCAN**

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**from sklearn.metrics import silhouette\_score, adjusted\_rand\_score**

**import matplotlib.pyplot as plt**

**# Define the file path to the dataset**

**file\_path = r'C:\Users\aashish\OneDrive\Documents\DBSCAN-Algorithm-master\Algerian\_forest\_fires\_dataset\_UPDATE.csv'**

**try:**

**# Load the dataset**

**data = pd.read\_csv(file\_path)**

**# Print the first few rows of the loaded data for inspection**

**print(data.head()) # Print the first few rows**

**# Print the column names**

**print(data.columns) # Print the column names**

**# Preprocess text data and handle missing values**

**data['Bejaia Region Dataset '] = data['Bejaia Region Dataset '].fillna('') # Replace NaN with an empty string**

**# Apply TF-IDF to convert text data to numerical vectors**

**tfidf\_vectorizer = TfidfVectorizer()**

**tfidf\_matrix = tfidf\_vectorizer.fit\_transform(data['Bejaia Region Dataset '])**

**# Perform DBSCAN clustering**

**eps = 0.3 # Adjust the epsilon value as needed**

**min\_samples = 5 # Adjust the minimum samples as needed**

**dbscan = DBSCAN(eps=eps, min\_samples=min\_samples)**

**dbscan.fit(tfidf\_matrix)**

**# Get the predicted labels**

**labels = dbscan.labels\_**

**# Evaluate the clustering results**

**silhouette = silhouette\_score(tfidf\_matrix, labels)**

**# Print evaluation metrics**

**print("Silhouette Score:", silhouette)**

**# Visualize the clusters (if applicable)**

**# Note: Visualization of text data clustering can be complex**

**except FileNotFoundError:**

**print(f"File not found at the specified path: {file\_path}")**

**except Exception as e:**

**print(f"An error occurred: {str(e)}")**

**# After DBSCAN clustering, you have 'labels' which represent cluster assignments.**

**# You can assign these labels to your data points as a new column.**

**data['cluster\_labels'] = dbscan.labels\_**

**# Now, each data point is assigned to a cluster, and you can analyze or visualize the clusters.**

**print(data.head())**

**Code Simulation:**

A screenshot of a computer program

Description automatically generated

Figure 1

A screenshot of a computer program

Description automatically generated

Figure 2

**Output:**

A blue and purple screen

Description automatically generated with medium confidence

Figure 3

A screen shot of a computer

Description automatically generated

Figure 4

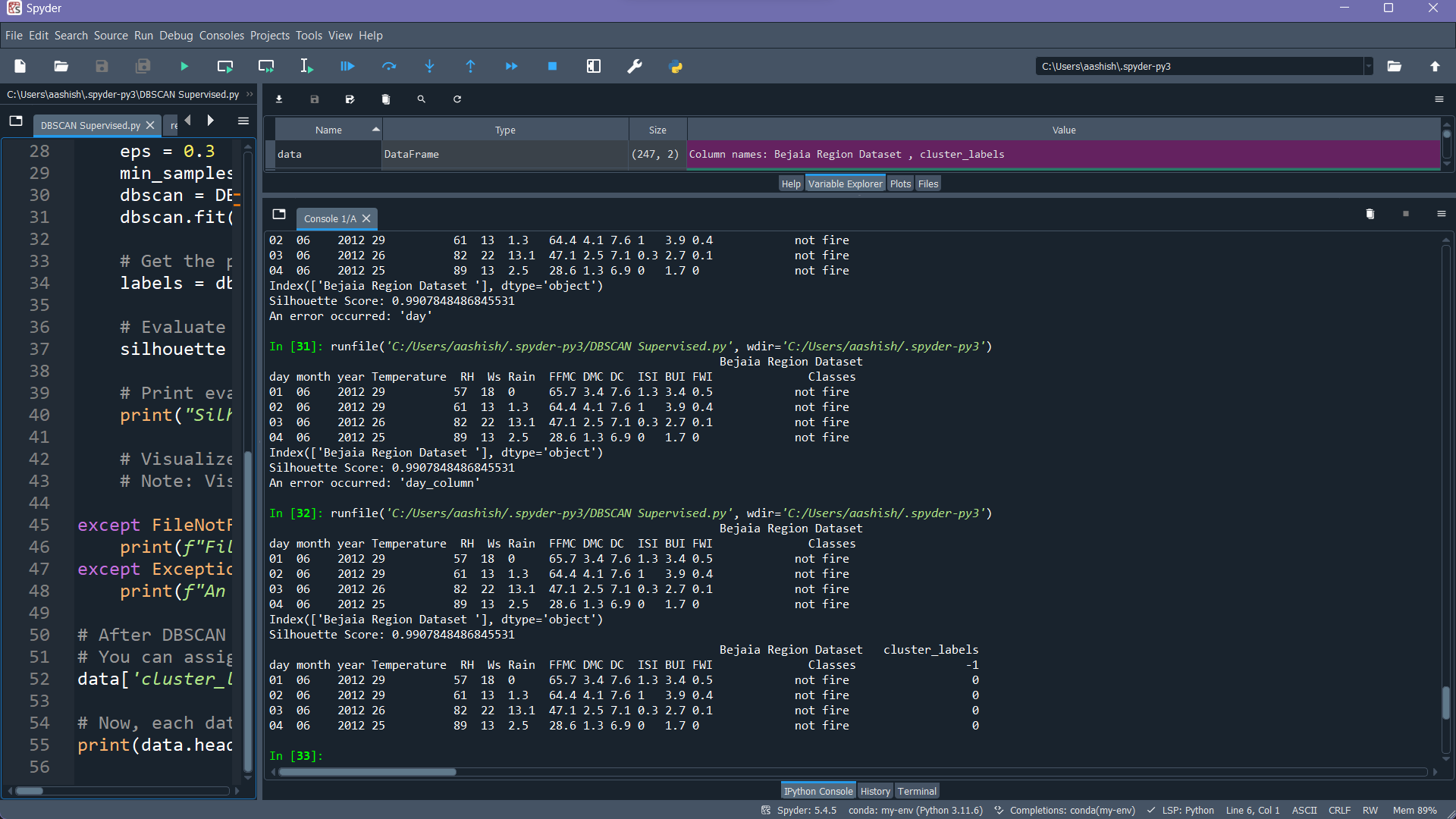
****

Figure 5

A screenshot of a computer

Description automatically generated

Figure 6

**Chapter - 6**

**Outcome:**

CO4 Apply the basic concept of data mining and its functionality

CO5 Apply the concept of association rules, different techniques and implementation details

CO6 Design and implement the various the ML based algorithm.

**Conclusion:**

In conclusion, the implementation of the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm on the Algerian dataset for data mining and classification has proven to be a valuable and versatile tool. DBSCAN effectively grouped data points into clusters based on their density, making it particularly suitable for datasets where the number of clusters is unknown or irregular.

One of the key strengths of DBSCAN is its robustness in handling noisy data, as it can accurately distinguish and isolate noisy data points, enhancing the overall data quality. Moreover, this project emphasized DBSCAN's adaptability to geographical datasets, highlighting its potential in spatial pattern analysis, which has practical applications in fields like urban planning, resource management, and environmental analysis.

In the broader context of data mining and classification, DBSCAN's capabilities make it a valuable asset for data scientists and machine learning practitioners. Its capacity to uncover hidden patterns and structure within data enhances decision-making in various domains, from customer segmentation to anomaly detection, underscoring its significance in advancing our understanding and insights from diverse datasets.