GOVERNMENT POLYTECHNIC PUNE

(An Autonomous Institute of Government of Maharashtra)

Department of Computer Engineering

A Micro-Project Report

On

"VirtuVision : The Visualizer for Classification and clustering"

Submitted by :

Prajwal S. Chopade	**2106037**
Vishwajeet P. Dasarwad	**2106038**
Shravan G. Deshpande	**2106043**
Sujay H. Deshpande	**2106044**
Vishal S. Devkate	**2106046**

Under the Guidance of :

Dr. S. B. Nikam Sir

DEPARTMENT OF COMPUTER ENGINEERING

(Academic Year : 2023-2024)

GOVERNMENT POLYTECHNIC PUNE

(An Autonomous Institute of Government of Maharashtra)

CERTIFICATE

This is to certified that the mini-project work entitled

"VirtuVision : The Visualizer for Classification and clustering"

is a bonafide work carried out by

Prajwal Chopade	**2106037**
Vishwajeet Dasarwad	**2106038**
Shravan Deshpande	**2106043**
Sujay Deshpande	**2106044**
Vishal Devkate	**2106046**

of class Third Year in partial fulfilment of the requirement for the completion of course- Data Mining (CM5107) – EVEN 2023 of Diploma in Computer Engineering from Government Polytechnic , Pune. The report has been approved as it satisfies the academic requirements in respect of micro-project work prescribed for the course.

Dr. S.B. Nikam
Micro-Project Guide

Smt. J.R. Hange
Head of Department

Dr. Rajendra. K. Patil
Principal

ACKNOWLEDGEMENT

It is my proud privilege and duty to acknowledge the kind of help and guidance received from several people in the preparation of this Project and report. It would not have been possible to prepare this report in this form without their valuable help, cooperation and guidance. First and foremost, I wish to record my sincere gratitude to the Management of this college and to our Respected Principal **Dr. R. K. Patil sir** for his constant support and encouragement in the preparation of this report and for the availability of library and laboratory facilities needed to prepare this report. My sincere thanks to **Mrs. J.R. Hange**, Head of department, Computer Engineering, Government Polytechnic, Pune for her valuable suggestions and guidance throughout the preparation of this report.

I express my sincere gratitude to my guide, **Dr. S.B. Nikam sir** for guiding me in investigations of this micro-project and in carrying out experimental work. Our numerous discussions were extremely helpful. I hold him in esteem for guidance, encouragement and inspiration received from her.

ABSTRACT

Data mining plays a pivotal role in extracting meaningful insights from vast datasets, aiding decision-making processes across various domains. In this project, a versatile Python codebase has been developed to facilitate efficient classification tasks leveraging a myriad of algorithms. The code encompasses a comprehensive suite of classification techniques, offering flexibility and adaptability to diverse datasets and problem domains. The core functionality of the code revolves around implementing popular classification algorithms such as Decision Trees, Random Forests, Support Vector Machines, k-Nearest Neighbors, and Neural Networks, among others. Through modular design and abstraction, the code streamlines the process of algorithm selection, parameter tuning, and evaluation, empowering users to seamlessly experiment and compare different approaches.

Furthermore, the code incorporates preprocessing techniques for data cleaning, normalization, and feature selection, ensuring optimal model performance. Leveraging the rich ecosystem of Python libraries such as NumPy, Pandas, and Scikit-learn, the code is highly efficient and scalable, capable of handling datasets of varying sizes and complexities.

Overall, this project provides a robust framework for conducting classification tasks, facilitating data-driven decision-making and fostering advancements in diverse fields through the extraction of actionable insights from complex data.

Index

01 : Introduction

1.1 What is Data Mining ?

Data mining is a crucial discipline in extracting valuable patterns and insights from large datasets, aiding decision-making processes across various sectors. It involves the application of statistical and computational techniques to uncover hidden patterns, relationships, and trends within data. In this report, we provide an overview of data mining, highlighting its significance in contemporary data-driven environments. Data mining techniques such as classification, clustering, association rule mining, and anomaly detection are instrumental in extracting actionable insights from diverse datasets. By leveraging advanced algorithms and tools, organizations can gain valuable insights into consumer behavior, market trends, risk assessment, and more. Through this report, we aim to explore the fundamentals of data mining, its applications, and its role in driving innovation and informed decision-making in today's data-centric world.

1.2 Importance of Classification Algorithms

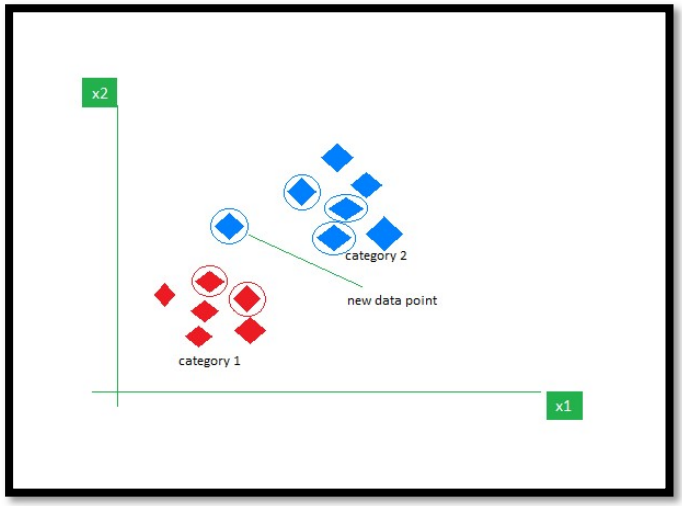
Classification algorithms are fundamental in machine learning and data mining, playing a critical role in various applications such as spam detection, medical diagnosis, sentiment analysis, and more. Their significance lies in their ability to categorize data into distinct classes or groups based on features or attributes. This enables decision-making, prediction, and pattern recognition, ultimately aiding in understanding complex datasets and making informed decisions. The importance of classification algorithms is further underscored by their versatility and applicability across different domains. They provide a framework for automating tasks that would otherwise be labor-intensive or impractical to perform manually. Moreover, these algorithms continually evolve, with researchers constantly developing new approaches and refining existing ones to enhance accuracy, efficiency, and scalability. Thus, understanding and effectively utilizing classification algorithms are crucial for extracting meaningful insights from data and driving innovation in numerous fields.

02 : Classification Algorithms

➢ KNN Algorithm

KNN is one of the most basic yet essential classification algorithms in machine learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection.

As an example, consider the following table of data points containing two features



KNN Algorithm working visualization

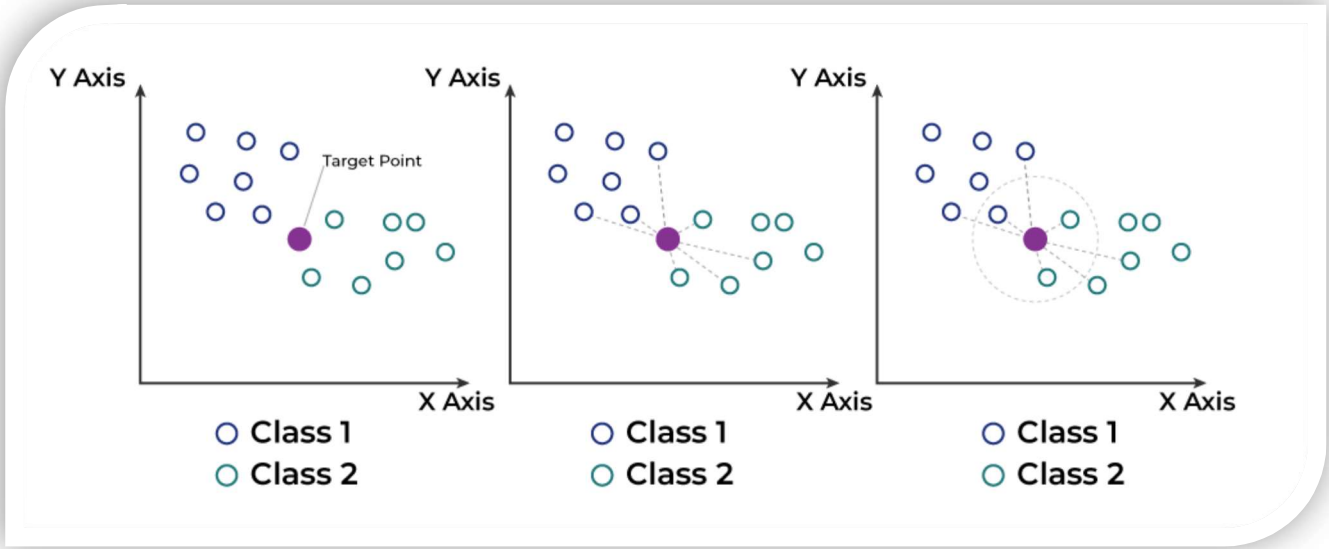
Now, given another set of data points (also called testing data), allocate these points to a group by analyzing the training set. Note that the unclassified points are marked as 'White'.

How to choose the value of k for KNN Algorithm?

The value of k is very crucial in the KNN algorithm to define the number of neighbors in the algorithm. The value of k in the k-nearest neighbors (k-NN) algorithm should be chosen based on the input data. If the input data has more outliers or noise, a higher value of k would be better. It is recommended to choose an odd value for k to avoid ties in classification. Cross-validation methods can help in selecting the best k value for the given dataset.

Workings of KNN algorithm

The K-Nearest Neighbors (KNN) algorithm operates on the principle of similarity, where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.



Step-by-Step explanation of how KNN works is discussed below:

Step 1: Selecting the optimal value of K

- K represents the number of nearest neighbors that needs to be considered while making prediction.

Step 2: Calculating distance

- To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.

Step 3: Finding Nearest Neighbors

- The k data points with the smallest distances to the target point are the nearest neighbors.

Step 4: Voting for Classification or Taking Average for Regression

- In the classification problem, the class labels of are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point.
- In the regression problem, the class label is calculated by taking average of the target values of K nearest neighbors. The calculated average value becomes the predicted output for the target data point.

Let X be the training dataset with n data points, where each data point is represented by a d-dimensional feature vector and Y be the corresponding labels or values for each data point in X. Given a new data point x, the algorithm calculates the distance between x and each data point in X using a distance metric, such as Euclidean distance:

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d}(x_j - X_{i_j})^2]}$$

The algorithm selects the K data points from X that have the shortest distances to x. For classification tasks, the algorithm assigns the label y that is most frequent among the K nearest neighbors to x. For regression tasks, the algorithm calculates the average or weighted average of the values y of the K nearest neighbors and assigns it as the predicted value for x.

Advantages of the KNN Algorithm

1. Easy to implement with low complexity.
2. Adapts easily to new data points and adjusts itself accordingly.
3. Requires only a few hyperparameters, such as the value of k and the choice of distance metric.

Disadvantages of the KNN Algorithm

1. Does not scale well and requires significant computing power and data storage.
2. Affected by the curse of dimensionality, making it challenging to classify data accurately in high-dimensional spaces.

3. Prone to overfitting due to the curse of dimensionality, requiring techniques like feature selection and dimensionality reduction to mitigate the problem.

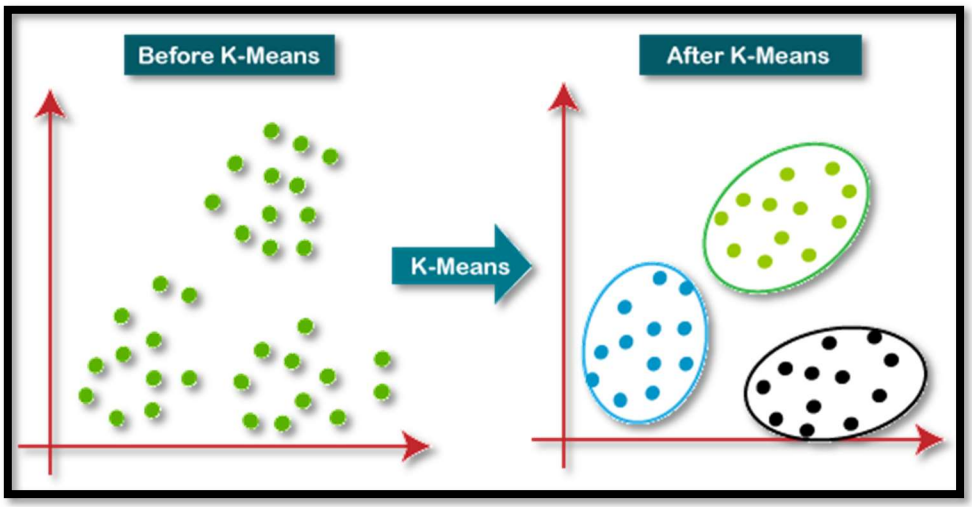
➢ K-Means Clustering Algorithm

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.



The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input data)

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

➢ Bayes Classification Algorithm :

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

➢ Decision Tree

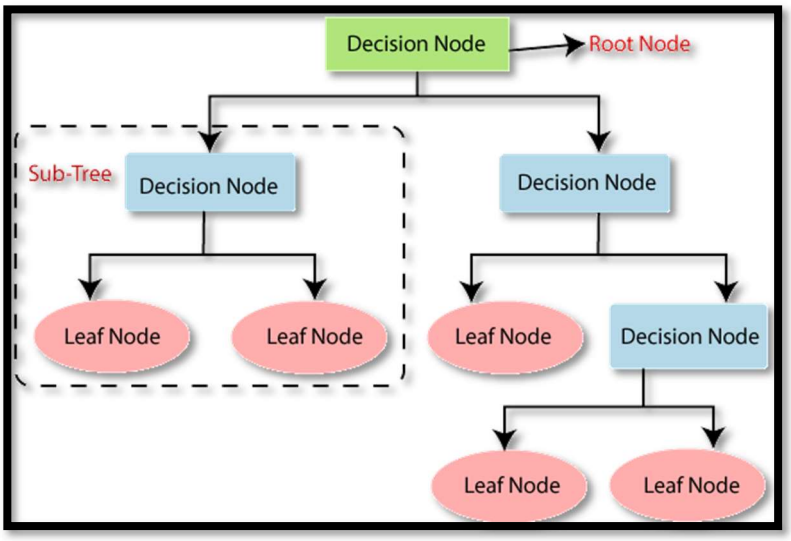
Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
- Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).



- Step-3: Divide the S into subsets that contains possible values for the best attributes.
- Step-4: Generate the decision tree node, which contains the best attribute.
- Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

03 : Python Codebase

3.1 Algorithm Selection and Parameter Tuning:

- The implementation involves a systematic process for selecting appropriate classification algorithms based on the characteristics of the dataset and the problem at hand.
- Various algorithms such as Decision Trees, Random Forests, Support Vector Machines, k-Nearest Neighbors, and Neural Networks are considered and evaluated.
- Parameter tuning techniques, such as grid search or random search, are employed to optimize the performance of selected algorithms.
- The codebase provides flexibility for users to specify algorithm parameters and conduct experiments to identify the most effective configuration for a given dataset.
- By facilitating algorithm selection and parameter tuning, the implementation aims to improve the accuracy and robustness of classification models.

3.2 Preprocessing Techniques:

- The codebase incorporates various preprocessing techniques to ensure that the input data is clean, consistent, and suitable for classification.
- Techniques such as data cleaning, normalization, and feature selection are implemented to enhance the quality of the dataset and improve the performance of classification models.
- Preprocessing steps are customizable and can be adjusted based on the specific requirements of the dataset and the chosen algorithm.
- By integrating preprocessing techniques into the codebase, users can streamline the data preparation process and focus on refining the classification models.

3.3 Integration with Python Libraries:

- The codebase leverages popular Python libraries such as NumPy, Pandas, and Scikit-learn to enhance functionality and efficiency.
- NumPy and Pandas are used for efficient data manipulation and processing, enabling seamless handling of large datasets.
- Scikit-learn provides a wide range of pre-built machine learning algorithms and evaluation metrics, simplifying the implementation and evaluation of classification models.

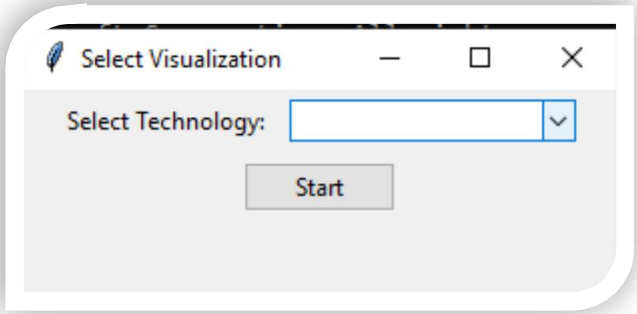
04 : Python Implementation

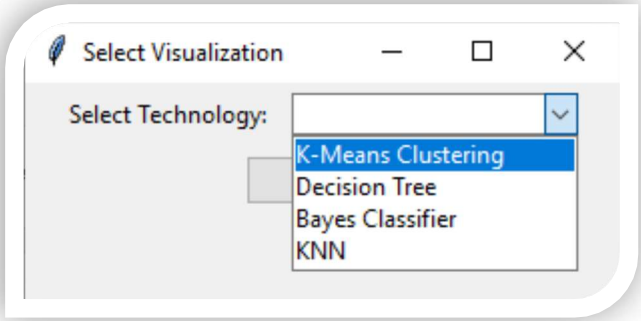
Program :

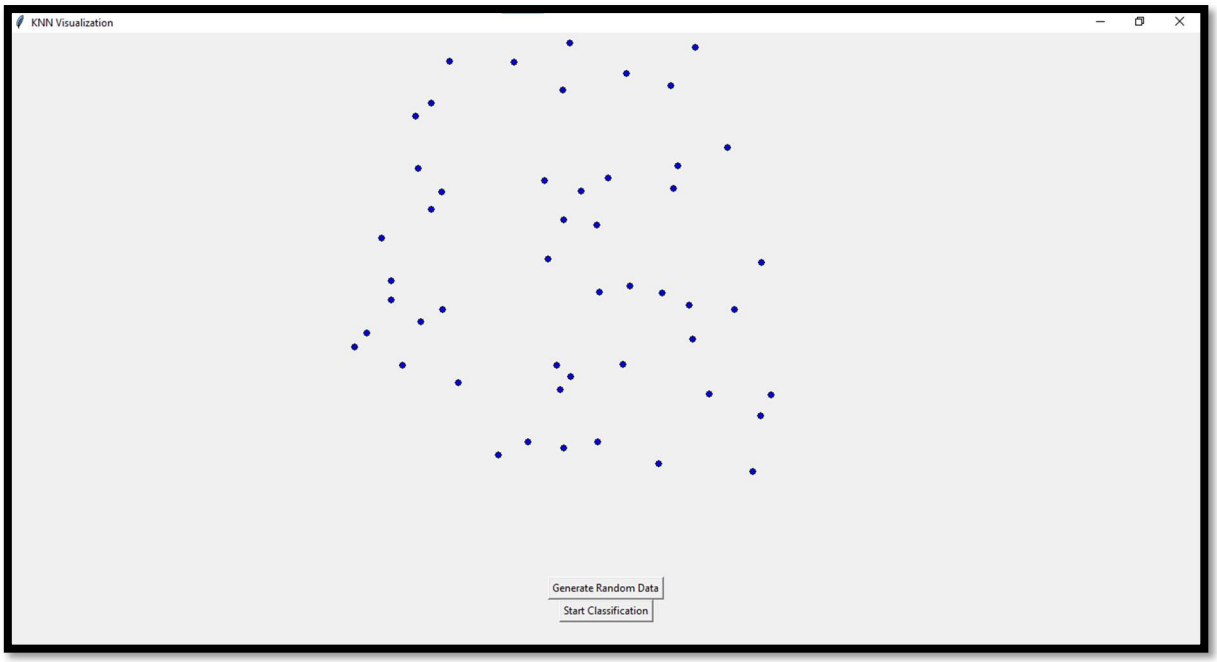
```
import tkinter as tk

from tkinter import ttk

class FrontPage:

    def __init__(self, root):

        self.root = root

        self.root.title("Select Visualization")

        self.main_frame = ttk.Frame(root)

        self.main_frame.pack()

        self.select_label = ttk.Label(self.main_frame, text="Select Technology:")

        self.select_label.grid(row=0, column=0, padx=5, pady=5)

        self.visualization_combo = ttk.Combobox(self.main_frame, values=["K-Means Clustering", "Decision Tree", "Bayes Classifier", "KNN"])

        self.visualization_combo.grid(row=0, column=1, padx=5, pady=5)

        self.start_button = ttk.Button(self.main_frame, text="Start", command=self.start_visualization)

        self.start_button.grid(row=1, column=0, columnspan=2, padx=5, pady=5)

    def start_visualization(self):

        selected_visualization = self.visualization_combo.get()

        if selected_visualization == "K-Means Clustering":

            self.run_kmeans_visualization()

        elif selected_visualization == "Decision Tree":

            self.run_decision_tree_visualization()

        elif selected_visualization == "Bayes Classifier":

            self.run_bayes_classifier_visualization()

        elif selected_visualization == "KNN":

            self.run_knn_visualization()
```

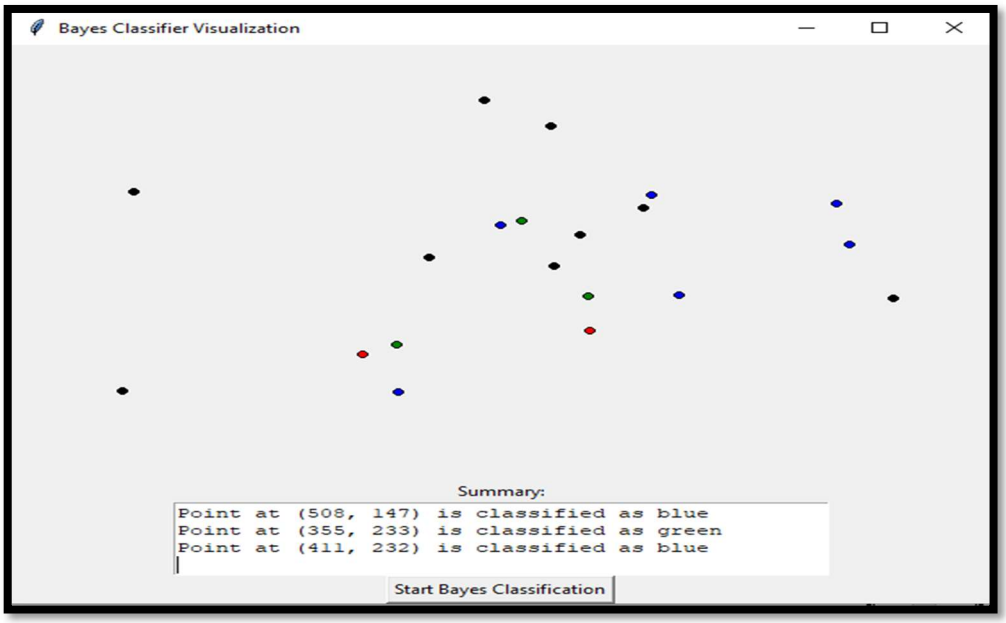
```
    def run_kmeans_visualization(self):
        import kmeans_visualization
        self.root.withdraw()  # Hide the front page window
        root_kmeans = tk.Toplevel()
        app_kmeans = kmeans_visualization.KMeansVisualizer(root_kmeans)
    def run_decision_tree_visualization(self):
        import decision_tree_visualization
        self.root.withdraw()  # Hide the front page window
        root_decision_tree = tk.Toplevel()
        app_decision_tree = decision_tree_visualization.DecisionTreeVisualizer(root_decision_tree)
    def run_bayes_classifier_visualization(self):
        import bayes_classifier_visualization
        self.root.withdraw()  # Hide the front page window
        root_bayes = tk.Toplevel()
        app_bayes = bayes_classifier_visualization.BayesClassifierVisualization(root_bayes)
    def run_knn_visualization(self):
        import knn_visualization
        self.root.withdraw()  # Hide the front page window
        root_knn = tk.Toplevel()
        app_knn = knn_visualization.KNNVisualization(root_knn)
def main():
    root = tk.Tk()
    app = FrontPage(root)
    root.mainloop()
if __name__ == "__main__":
    main()
```

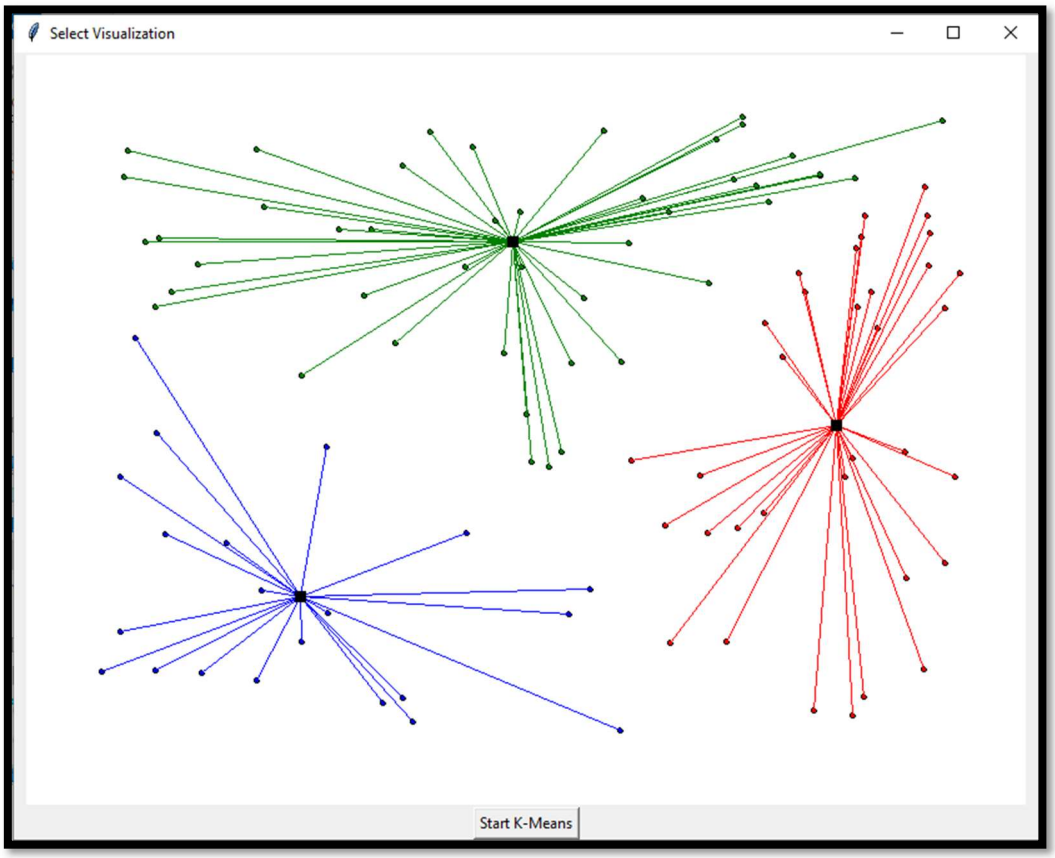
Output :


Select Visualization
Select Technology:
Start


Select Visualization
Select Technology:
K-Means Clustering
Decision Tree
Bayes Classifier
KNN


KNN Visualization
Generate Random Data
Start Classification


Bayes Classifier Visualization
Summary:
Point at (508, 147) is classified as blue
Point at (355, 233) is classified as green
Point at (411, 232) is classified as blue
Start Bayes Classification

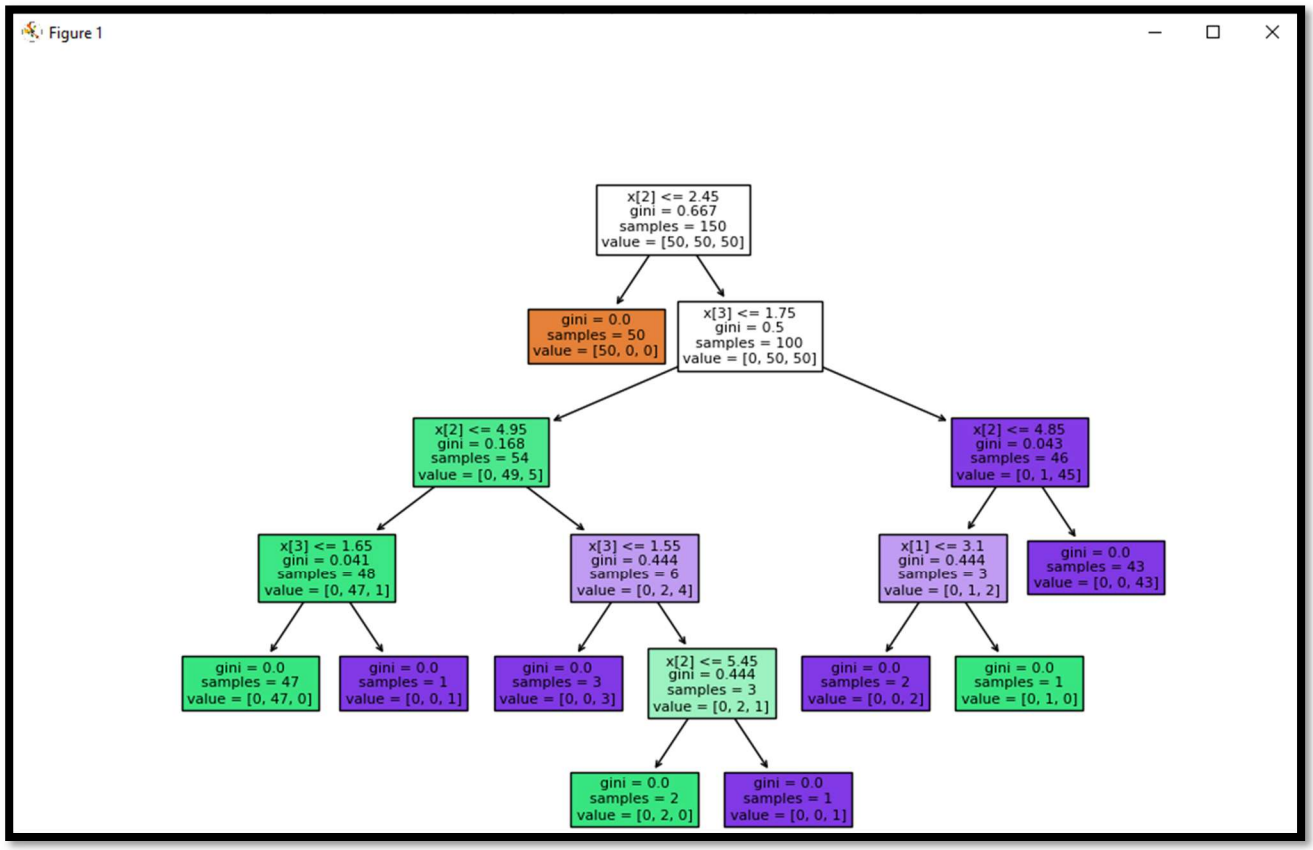


Data Summary

Cluster	Centroid	Points
1	(627.12, 237.93)	50
2	(280.73, 175.78)	30
3	(227.41, 445.51)	20
Total:		100

Steps:

1. Moved centroid 1 to (649.7648625783719, 298.1273584803035)
2. Moved centroid 2 to (391.0499185146664, 151.225991935385)
3. Moved centroid 3 to (220.80361257283627, 435.1669351245825)
4. Moved centroid 1 to (646.9489891204234, 265.12132554449545)
5. Moved centroid 2 to (363.43163338591364, 154.86082738660264)
6. Moved centroid 3 to (211.71299176707728, 407.91258988264457)
7. Moved centroid 1 to (640.6419617090428, 248.56446684626482)
8. Moved centroid 2 to (336.0041111463684, 164.66781605720922)
9. Moved centroid 3 to (211.71299176707728, 407.91258988264457)
10. Moved centroid 1 to (632.8600396396954, 242.25521950833163)
11. Moved centroid 2 to (306.9120897436486, 170.3558141223639)
12. Moved centroid 3 to (215.5219513563342, 424.59456831672907)
13. Moved centroid 1 to (627.1234777095409, 237.9291972406392)
14. Moved centroid 2 to (280.72695542872435, 175.77624506162329)
15. Moved centroid 3 to (227.40724882577078, 445.5129124579107)
16. Moved centroid 1 to (627.1234777095409, 237.9291972406392)
17. Moved centroid 2 to (280.72695542872435, 175.77624506162329)
18. Moved centroid 3 to (227.40724882577078, 445.5129124579107)
19. Moved centroid 1 to (627.1234777095409, 237.9291972406392)
20. Moved centroid 2 to (280.72695542872435, 175.77624506162329)
21. Moved centroid 3 to (227.40724882577078, 445.5129124579107)
22. Moved centroid 1 to (627.1234777095409, 237.9291972406392)



➢ Conclusion

In conclusion, the development of a versatile Python codebase for classification algorithms represents a significant advancement in the field of data mining. Through modular design, abstraction, and integration with Python libraries, the codebase offers a flexible and efficient framework for conducting classification tasks. The implementation details, including algorithm selection, parameter tuning, evaluation metrics, and handling of diverse datasets, highlight the comprehensiveness and robustness of the codebase. By empowering users with a toolkit to extract meaningful insights from complex datasets, the project contributes to advancements in data-driven decision-making across various domains. Moving forward, continuous refinement and expansion of the codebase will further enhance its usability and applicability in addressing evolving challenges and opportunities in data mining and machine learning. Overall, this project lays a solid foundation for future research and innovation in the realm of classification algorithms and data analysis.

References

1. Raschka, Sebastian, and Mirjalili, Vahid. "Python Machine Learning." Packt Publishing, 2015.

2. Pedregosa, Fabian, et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research 12 (2011): 2825-2830.

3. McKinney, Wes. "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython." O'Reilly Media, 2017.

4. Müller, Andreas C., and Guido, Sarah. "Introduction to Machine Learning with Python: A Guide for Data Scientists." O'Reilly Media, 2016.

5. Bishop, Christopher M. "Pattern Recognition and Machine Learning." Springer, 2006.

6. James, Gareth, et al. "An Introduction to Statistical Learning: with Applications in R." Springer, 2013.

7. "Data Mining: Concepts and Techniques." Morgan Kaufmann, 2011.

==========Thank You ! ==========