

# Prediction of weather using DBSCAN and Spectral Clustering

A THESIS

submitted by

DANISH AHMAD ANSARI  
(19CS4142)

for the award of the degree

of

MASTER OF TECHNOLOGY



DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING  
NATIONAL INSTITUTE OF TECHNOLOGY

DURGAPUR - 713209

MAY 2021

## Declaration of Authorship

I hereby certify that the work which is being presented in the dissertation entitled as, Smart Contract and Consensus algorithm in Blockchain Prediction of weather model using DBSCAN and Spectral Clustering in the partial fulfillment of the requirement for the award of degree Master of Technology in Computer Science and Engineering department of the National Institute of Technology, Durgapur is an authentic record of my own work carried out during the period from July 2020 to June 2021 under the supervision of Dr. Anirban Sarkar, Associate Professor, Department of Computer Science and Engineering, National Institute of Technology, Durgapur.

I declare that the work presented here has not been submitted, either in part or whole, for the award of any degree at this or any other university.

Date: .....

Danish Ahmad Ansari

19CS4142

This is to certify that the above statements made by the candidate are correct to the best of my knowledge.

Date: .....

Dr. Anirban Sarkar,  
Associate Professor,  
Computer Science and Engineering,  
National Institute of Technology,  
Durgapur - 713209, India.

The M.Tech viva-voce examination of Danish Ahmad Ansari, research scholar has been held on .....17th May, 2021

(Signature of Supervisor)

(Signature of Invigilator)

## ABSTRACT

Abstract— Data clustering is the process of identifying natural groupings or clusters within multidimensional data based on some similarity measure. Clustering is a fundamental process in many different disciplines. Hence, researchers from different fields are actively working on the clustering problem. This paper provides an overview of the different representative clustering methods. In addition, several clustering validations indices are shown. Furthermore, approaches to automatically determine the number of clusters are presented. Finally, application of different heuristic approaches to the clustering problem is also investigated.

Clustering can be considered the most important unsupervised learning problem; so, as every other problem of this kind, it deals with finding a structure in a collection of unlabeled data. A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”. A cluster is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.

In this thesis we have worked on comparing the two algorithms and measuring their performance and using them accordingly for prediction using a decision tree model.

**Keywords:** Clustering; Clustering Validation; Hard Clustering; Fuzzy Clustering; Unsupervised Learning ; Decision Tree; prediction

## ACKNOWLEDGEMENTS

I would like to thank the Almighty for giving me the strength, knowledge, ability, and opportunity to undertake this research study and to persevere and complete it satisfactorily.

First and foremost, I would like to express my deep sense of profound gratitude to my research supervisor Dr. Anirban Sarkar, Associate , Professor/CSE, NIT Durgapur for his constant guidance, support, and motivation during my research work. His systematic and planned approach had consistently motivated me towards all activities including the completion of the research work in time. His valuable suggestions and comments at appropriate times during this investigation have contributed a lot in smooth conduct of the research work. Motivation, appreciation, moral support, and flexibility permitted during the course have helped me to complete my research successfully.

I would like to thank the Ministry of Human Resource Development (MHRD), Government of India, for granting financial support during the M.Tech program. I would also like to thank the Director: Dr. Anupam Basu.

I must also thank the HoD: Dr. Tandra Pal, faculty members, and staff of the Department of Computer Science and Engineering, NIT Durgapur for their kind help and support.

I must acknowledge the extended support of my fellow M.Tech scholars and friends : Rahul Mukharjee, Kuldeep Kumar, Vijay Jagannath Zaware, Ajit Kumar.

I owe my utmost gratitude to my parents for their love and support throughout my life. I thank both for giving me strength, guidance, and freedom to chase my dreams.

There are few more uncredited good hearts behind this success. I would like to thank every one of them.

Again, I am very grateful to my supervisor and my family, who believed and encouraged me for making this mission possible.

**Danish Ahmad Ansari**

## TABLE OF CONTENTS

Title	Page No.
ABSTRACT	i
ACKNOWLEDGEMENTS	ii
TABLE OF CONTENTS	iv
LIST OF TABLES	vii
LIST OF FIGURES	viii
ABBREVIATION	ix
CHAPTER 1 INTRODUCTION	1
1.1 PRELIMINARY FOUNDATION .....	2
1.1.1 Data Mining Steps .....	2
1.1.2 Data Mining Models .....	2
1.1.3 Applications of Data Mining .....	2
...	
1.1.4 Special Remarks. ....	3
1.2 MOTIVATION FOR THEWORK .....	4
1.3 PROBLEM STATEMENT AND OBJECTIVE .....	4
1.4 THESIS OUTLINE.....	...
1.4.1 Chapter 2: Supervised and Unsupervised Machine Learning Algorithm .....	4
1.4.2 Chapter 3: DBSCAN and Spectral Clustering .....	4
1.4.3 Chapter 4: Literature Review .....	4
1.4.4 Chapter 5: Proposed Scheme and Framework .....	4
1.4.5 Chapter 6: Conclusion and Future work .....	4
CHAPTER 2 Supervised and Unsupervised Machine Learning Algorithm	5
2.1 Supervised learning .....	5
2.2 Unsupervised learning .....	6
2.3 Clustering Types .....	7
2.4 Types of Clustering Methods .....	7

## CHAPTER 3 DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE AND SPECTRAL CLUSTERING 22

iv

3.1 Density-Based Clustering . . . . .	22
3.1.1 Algorithmic steps for DBSCAN clustering. . . . .	22
3.1.2 Parameter Estimation . . . . .	23
3.1.3 The complexity of DBSCAN Clustering Algorithm . . . . .	25
. . .	
3.2 Spectral Clustering . . . . .	26
3.2.1 Eigenvectors and Eigenvalues . . . . .	26
3.3 Graphs . . . . .	27
3.4 Framework . . . . .	28
3.4.1 Graphs . . . . .	28
3.4.2 Adjacency Matrix . . . . .	29
3.4.3 Laplacian . . . . .	30
3.4.4 Nearest Neighbours . . . . .	31
3.4.5 Conclusion . . . . .	32
3.5 Conclusion . . . . .	32
CHAPTER 4 Literature Review	33
4.1 Prediction using Convex hull . . . . .	33
4.2 Using ANN. . . . .	33
4.2.1 Time Series forecasting. . . . .	34
4.2.2 Protocol Overview . . . . .	34
4.2.3 Multimodal Weather forecast. . . . .	34
4.2.4 Conclusion. . . . .	34
.	
CHAPTER 5 Proposed Scheme	35
5.1 Introduction . . . . .	35
5.2 Data Cleaning . . . . .	35
5.2.1 Label Binarizer . . . . .	36
5.2.2 DBSCAN code . . . . .	37
5.2.3 Spectral code . . . . .	38
5.2.4 Recall score . . . . .	39
5.2.5 Decision tree . . . . .	40
5.2.6 ROC AUC curve . . . . .	41
.	

5.4 Conclusion .....	41
----------------------	----

CHAPTER 6 Future Works	43
References	44

#### LIST OF TABLES

Table No.	Title	Page No.
2.1	Block Header .....	14
2.2	Types of Transactions .....	15
2.3	Hashes stored in a block .....	15
4.1	Comparison of consensus .....	31

## LIST OF FIGURES

Figure No.	Title	Page No.
1	Algorithm comparison. . . .	3
2	Data mining steps . . . . .	6
3	Example of cluster . . . . .	7
4	Cluster Dendogram . . . . .	8
5	Agglomerative . . . . .	14
6	Mean Shift . . . . .	17

6

## ABBREVIATION

DBSCAN Density-based spatial clustering of applications with noise (DBSCAN)  
 SI Spectral clustering  
 FPR false positive rate  
 FN false negative  
 FP False positive  
 TP true positive  
 TN true negative  
 AUC Area under curve



# CHAPTER 1

## INTRODUCTION

Computer Science and its engineering has a huge impact on human life, it has provided a huge platform so that every problem can be dealt with optimized and easy solutions. It has given the world a new style of thinking and innovation for the benefits of human life. It has given solutions to financial problems, helped in development and research, education and you name it any field, its contribution can not be denied. Computer Engineering has evolved as the main weapon in dealing with many peculiar problems which were not just so easy to solve for a human mind. Either we talk about its contribution to space research and calculating complex problems or providing the ample infrastructures in those laboratories to monitor all the happenings in the outside world. In the field of data analytics it has helped a lot. Data mining is a very popular topic nowadays. Unlike a few years ago, everything is bind with data now and we are capable of handling these kinds of large data well.

By collecting and inspecting these data, people were able to discover some patterns. Even the whole data set is a junk, there are some hidden patterns that can be extracted by combining multiple data sources to provide valuable insights. This is called as **Data Mining**.

### 1.1 PRELIMINARY FOUNDATION

#### 1.1.1 Data Mining Steps

The basic steps of data mining are follows

1. **Data Collection**
  2. **Data Cleaning**
  3. **Data Analysis**
  4. **Interpretation**
1. **Data collection** — The first step is to collect some data. As much as information we have is good to make the analysis easier later. We have to make sure that the source of data is reliable.
  2. **Data cleaning** — Since we are getting a large amount of data, we need to make sure that we only have the necessary data and remove the unwanted. Otherwise, they may lead us to false conclusions.
  3. **Data Analysis** — As the name says the analysis and finding patterns is done here
  4. **Interpretation** — Finally the analyzed data is interpreted to take important conclusions like prediction

### 1.1.2 Data mining Models

There are different kinds of models associated with data mining

1. Descriptive modeling
2. Predictive modeling
3. Prescriptive modeling

In **Descriptive Modeling**, it detects the similarities between the collected data and the reasons behind them. This is very important in constructing the final conclusion from the data set.

**Predictive Modeling** is used to analyze the past data and predict the future behavior. Past data give some kind of hint about the future.

With the significant development of web, text mining has added as a related discipline to data mining. It is required to process, filter and analyze data properly to create such predictive models.

### 1.1.3 Applications of Data Mining

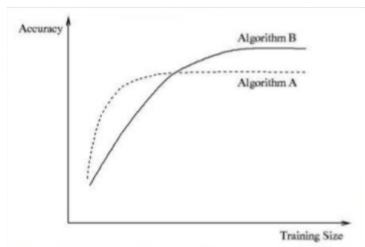
Data mining is useful in many ways. For marketing, it can be applied effectively. Using data mining we can analyze the behavior of customers and we can do advertising by getting more close to them. It will help to identify trends of customers for goods in the market and it allows the retailer to understand the purchase behavior of a buyer. In education domain we can identify the learning behaviors of students and the learning institutions can upgrade their modules and courses accordingly.

We can use data mining to solve natural disasters as well. If we can collect some information, we can use them to predict things like land sliding, rainfall, tsunami etc.

There are much more applications in data mining nowadays. They can vary from very simple things like marketing to very complex domains like making environmental disaster predictions etc

### 1.1.4 Special Remarks

- Data mining should not be used when complete, accurate solution for a particular problem is possible. When such solution is not possible we can use data mining techniques with lots of data to characterize the problem as input-output relationship.
- Need to analyze the problem property to determine whether it is a **Classification** (discrete output **Ex: True or False**) or **Estimation** (continuous output **Ex: real numbers between 0,1**) problem.
- The inputs should have sufficient information to make an accurate output. Otherwise it will lead to an inevitable decrease.
- There should be enough data make a accurate result. Need to select a proper algorithm according to the input data. Some algorithms need large amount of data to reach a good accuracy while others reach quickly.



## 1.2 MOTIVATION FOR THE WORK

Data mining is one of many processes organisations can use to analyse their collected data. The process of data mining allows businesses to gather useful information. The data can be analysed from a number of different perspectives in order to give businesses valuable information that can boost revenue or cut costs.

Data mining software analyses the relationships between and patterns within data. There are a number of different types of analytical data mining software available for use, including statistical, machine learning, and neural networks.

Clustering is a process that organisations can use within the data mining process, but what is clustering and how can it benefit businesses?

### What is clustering?

In everyday terms, clustering refers to the grouping together of objects with similar characteristics. When it comes to data and data mining the process of clustering involves portioning data into different groups.

There are six main methods of data clustering – the partitioning method, hierarchical method, density based method, grid based method, the model based method, and the constraint-based method. Each method groups the data in a different way. In the density based method, for instance, the data is clustered together according to its density, as the name suggests. In the grid based method, the objects are organised to create a grid structure.

### What are the benefits of clustering in data mining?

When it comes to business, data mining is most commonly used by companies with a strong focus on customers – so retail, finance, and marketing are some of the key organisations that benefit from data mining. Data mining is so important to these kinds of businesses because it allows them to ‘drill down’ into the data, and using clustering methods to analyse the data

From this they can examine the relationships between both internal factors – pricing, product positioning, staff skills – and external factors – such as competition and the demographics of customers. For instance, utilising one of the clustering methods during data mining can help business to identify distinct groups within their customer base. They can cluster different customer types into one group based on different factors, such as purchasing patterns. The factors analysed through clustering can have a big impact on sales and customer satisfaction, making it an invaluable tool to boost revenue, cut costs, or sometimes even both.

### 1.3 PROBLEM STATEMENT AND OBJECTIVE

Clustering algorithms aim to group the fingerprints in classes of similar elements. The clustering requires the concept of a metric. These algorithms implement the straightforward assumption that similar data belongs to the same class.

In our thesis we are using two algorithms for clustering. a) DBSCAN and  
b) Spectral Clustering.

We will employ cluster evaluation metrics to check which one performs good for our predictions model.

In cluster analysis, no assumption is made about the number of classes and their structure (statistical distribution); rather, the number of classes can be defined from the result of the analysis.

### 1.4 THESIS OUTLINE

#### 1.4.1 Chapter 2: Supervised and Unsupervised Machine Learning Algorithm

#### 1.4.2 Chapter 3: DBSCAN and Spectral Clustering

#### 1.4.3 Chapter 4: Literature Review

#### 1.4.4 Chapter 5: Proposed Scheme and Framework

This chapter discusses methods used for the research. It also provides the algorithms and tools used for the dissertation work. Additionally, this chapter details the solution of the problem using algorithms of DBSCAN and Spectral Clustering

#### Chapter 6: Conclusion and Future work

Finally, a summary of the overall work done in this research work is given in this chapter. The future scope for the research has also been discussed. Further, what areas of this research work can be improvised which may bring interesting results.

## CHAPTER 2

### Supervised and Unsupervised Learning

The previous chapter gives a brief introduction to data mining. In this chapter a detailed study of supervised and unsupervised machine learning.

#### 2.1 Introduction

##### 2.1 Supervised learning

Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Basically supervised learning is when we teach or train the machine using data that is well labeled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labeled data.

**For instance**, suppose you are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all different fruits one by one like this:

- If the shape of the object is rounded and has a depression at the top, is red in color, then it will be labeled as –**Apple**.
- If the shape of the object is a long curving cylinder having Green-Yellow color, then it will be labeled as –**Banana**
- Now suppose after training the data, you have given a new separate fruit, say Banana from the basket, and asked to identify it.
- Since the machine has already learned the things from previous data and this time have to use it wisely. It will first classify the fruit with its shape and color and would confirm the fruit name as BANANA and put it in the Banana category. Thus the machine learns the things from training data(basket containing fruits) and then applies the knowledge to test data(new fruit).

Supervised learning classified into two categories of algorithms:

- **Classification:** A classification problem is when the output variable is a category, such as “Red” or “blue” or “disease” and “no disease”
- **Regression:** A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Supervised learning deals with or learns with “labeled” data. This implies that some data is already tagged with the correct answer.

**Types:-**

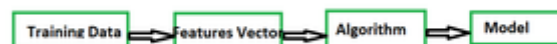
- Regression
- Logistic Regression
- Classification
- Naive Bayes Classifiers
- K-NN (k nearest neighbors)
- Decision Trees
- Support Vector Machine

**Advantages:-**

- Supervised learning allows collecting data and produces data output from previous experiences.
- Helps to optimize performance criteria with the help of experience.
- Supervised machine learning helps to solve various types of real-world computation problems.

**Disadvantages:-**

- Classifying big data can be challenging.
- Training for supervised learning needs a lot of computation time. So, it requires a lot of time.



2

## **2.2 Unsupervised learning**

Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to find the hidden structure in unlabeled data by itself.

**For instance**, suppose it is given an image having both dogs and cats which it has never seen.



Thus the machine has no idea about the features of dogs and cats so we can't categorize it as 'dogs and cats '. But it can categorize them according to their similarities, patterns, and differences, i.e., we can easily categorize the above picture into two parts. The first may contain all pics having **dogs** in it and the second part may contain all pics having **cats** in it. Here you didn't learn anything before, which means no training data or examples.

It allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with unlabelled data.

Unsupervised learning is classified into two categories of algorithms:

- **Clustering:** A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

## Types of Unsupervised Learning:-

### Clustering

1. Exclusive (partitioning)
2. Agglomerative
3. Overlapping
4. Probabilistic

### 2.3 Clustering Types:-

1. Hierarchical clustering
2. K-means clustering
3. Principal Component Analysis
4. Singular Value Decomposition
5. Independent Component Analysis

### 2.4 Types of Clustering Methods

As we made a point earlier that for a successful grouping, we need to attain two major goals: one, a similarity between one data point with another and two, a distinction of those similar data points with others which most certainly, heuristically differ from those points. The basis of such divisions begins

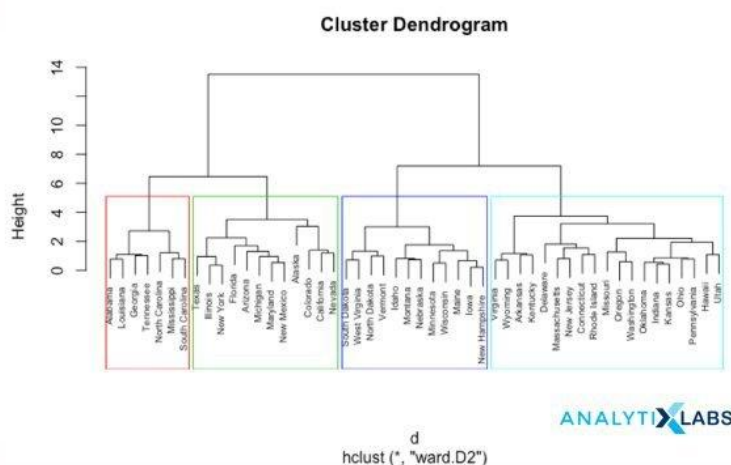
with our ability to scale large datasets and that's a major beginning point for us. Once we are through it, we are presented with a challenge that our data contains different kinds of attributes – categorical, continuous data, etc., and we should be able to deal with them. Now, we know that our data these days is not limited in terms of dimensions, we have data that is multi-dimensional in nature. The clustering algorithm that we intend to use should successfully cross this hurdle as well.

**The various types of clustering are:**

1. **Connectivity-based Clustering (Hierarchical clustering)**
2. **Centroids-based Clustering (Partitioning methods)**
3. **Distribution-based Clustering**
4. **Density-based Clustering (Model-based methods)**
5. **Fuzzy Clustering**
6. **Constraint-based (Supervised Clustering)**

### 2.4.2.1. Connectivity-Based Clustering (Hierarchical Clustering)

Hierarchical Clustering is a method of unsupervised machine learning clustering where it begins with a pre-defined top to bottom hierarchy of clusters. It then proceeds to perform a decomposition of the data objects based on this hierarchy, hence obtaining the clusters. This method follows two approaches based on the direction of progress, i.e., whether it is the top-down or bottom-up flow of creating clusters. These are Divisive Approach and the Agglomerative Approach respectively.

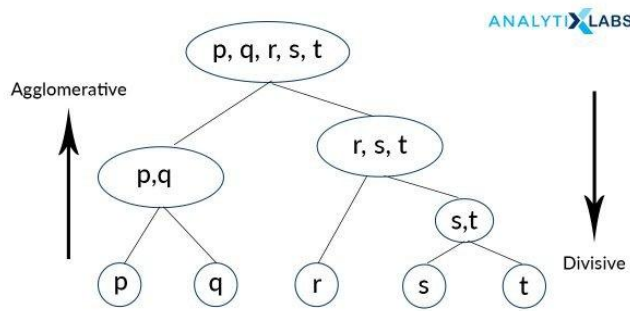


**1.1 Divisive Approach :** This approach of hierarchical clustering follows a top-down approach where we consider that all the data points belong to one large cluster and try to divide the data into smaller groups based on a termination logic or, a point beyond which there will be no further division of data points. This termination logic can be based on the minimum sum of squares of error inside a cluster or for categorical data, the metric can be the GINI coefficient inside a cluster.

Hence, iteratively, we are splitting the data which was once grouped as a single large cluster, to “n” number of smaller clusters in which the data points now belong to.

It must be taken into account that this algorithm is highly “rigid” when splitting the clusters – meaning, once a clustering is done inside a loop, there is no way that the task can be undone.





## 1.2 Agglomerative Approach

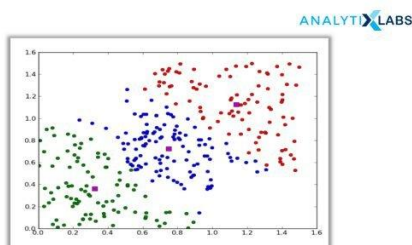
Agglomerative is quite the contrary to Divisive, where all the “N” data points are considered to be a single member of “N” clusters that the data is comprised into. We iteratively combine these numerous “N” clusters to fewer number of clusters, let’s say “k” clusters and hence assign the data points to each of these clusters accordingly. This approach is a bottom-up one, and also uses a termination logic in combining the clusters. This logic can be a number based criterion (no more clusters beyond this point) or a distance criterion (clusters should not be too far apart to be merged) or variance criterion (increase in the variance of the cluster being merged should not exceed a threshold, Ward Method)

## 2.4.3 Centroid Based Clustering

Centroid based clustering is considered as one of the most simplest clustering algorithms, yet the most effective way of creating clusters and assigning data points to it. The intuition behind centroid based clustering is that a cluster is characterized and represented by a central vector and data points that are in close proximity to these vectors are assigned to the respective clusters.

These groups of clustering methods iteratively measure the distance between the clusters and the characteristic centroids using various distance metrics. These are either of Euclidian distance, Manhattan Distance or Minkowski Distance.

The major setback here is that we should either intuitively or scientifically (Elbow Method) define the number of clusters, “k”, to begin the iteration of any clustering machine learning algorithm to start assigning the data points.



Despite the flaws, Centroid based clustering has proven it’s worth over Hierarchical clustering when working with large datasets. Also, owing to its simplicity in implementation and also interpretation, these algorithms have wide application areas viz., market segmentation, customer segmentation, text

topic retrieval, image segmentation etc.

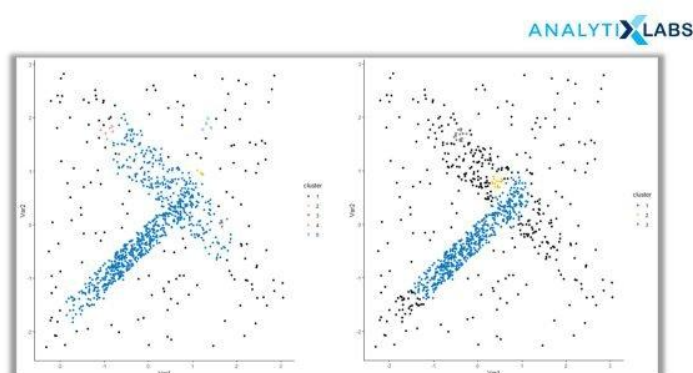
## 2.4.4. Density-based Clustering (Model-based Methods)

If one looks into the previous two methods that we discussed, one would observe that both hierarchical and centroid based algorithms are dependent on a distance (similarity/proximity) metric.

The very definition of a cluster is based on this metric. Density-based clustering methods take density into consideration instead of distances. Clusters are considered as the densest region in a data space, which is separated by regions of lower object density and it is defined as a maximal-set of connected points.

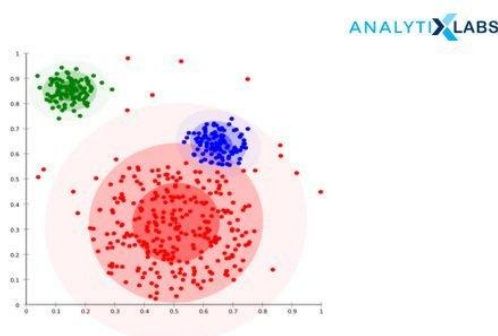
When performing most of the clustering, we take two major assumptions, one, the data is devoid of any noise and two, the shape of the cluster so formed is purely geometrical (circular or elliptical). The fact is, data always has some extent of inconsistency (noise) which cannot be ignored. Added to that, we must not limit ourselves to a fixed attribute shape, it is desirable to have arbitrary shapes so as to not to ignore any data points. These are the areas where density based algorithms have proven their worth!

Density-based algorithms can get us clusters with arbitrary shapes, clusters without any limitation in cluster sizes, clusters that contain the maximum level of homogeneity by ensuring the same levels of density within it, and also these clusters are inclusive of outliers or the noisy data.



## 2.4.5 Distribution-Based Clustering

Until now, the clustering techniques as we know are based around either proximity (similarity/distance) or composition (density). There is a family of clustering algorithms that take a totally different metric into consideration – probability. Distribution-based clustering creates and groups data points based on their likely hood of belonging to the same probability distribution (Gaussian, Binomial etc.) in the data.



A major drawback of density and boundary-based approaches is in specifying the clusters apriori to some of the algorithms and mostly the definition of the shape of the clusters for most of the algorithms. There is at least one tuning or hyper-parameter which needs to be selected and not only that is trivial but also any inconsistency in that would lead to unwanted results.

Distribution based clustering has a vivid advantage over the proximity and centroid based clustering methods in terms of flexibility, correctness and shape of the clusters formed. The major problem however is that these clustering methods work well only with synthetic or simulated data or with data where most of the data points most certainly belong to a predefined distribution, if not, the results will overfit.

### 2.4.6 Fuzzy Clustering

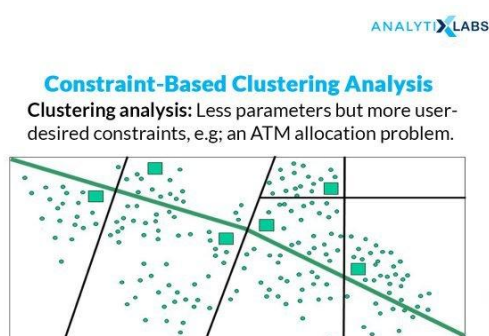
The general idea about clustering revolves around assigning data points to mutually exclusive clusters, meaning, a data point always resides uniquely inside a cluster and it cannot belong to more than one cluster. Fuzzy clustering methods change this paradigm by assigning a data-point to multiple clusters with a quantified degree of belongingness metric. The data-points that are in proximity to the center of a cluster, may also belong in the cluster that is at a higher degree than points in the edge of a cluster. The possibility of which an element belongs to a given cluster is measured by membership coefficient that vary from 0 to 1.

Fuzzy clustering can be used with datasets where the variables have a high level of overlap. It is a strongly preferred algorithm for Image Segmentation, especially in bioinformatics where identifying overlapping gene codes makes it difficult for generic clustering algorithms to differentiate between the image's pixels and they fail to perform a proper clustering.

### 2.4.7 Constraint-based (Supervised Clustering)

The clustering process, in general, is based on the approach that the data can be divided into an optimal number of “unknown” groups. The underlying stages of all the clustering algorithms to find those hidden patterns and similarities, without any intervention or predefined conditions. However, in certain business scenarios, we might be required to partition the data based on certain constraints. Here is where a supervised version of clustering machine learning techniques come into play.

A constraint is defined as the desired properties of the clustering results, or a user's expectation on the clusters so formed – this can be in terms of a fixed number of clusters, or, the cluster size, or, important dimensions (variables) that are required for the clustering process.



Usually, tree-based, [Classification machine learning algorithms](#) like Decision Trees, Random Forest, and Gradient Boosting, etc. are made use of to attain constraint-based clustering. A tree is constructed by splitting without the interference of the constraints or clustering labels. Then, the leaf nodes of the

tree are combined together to form the clusters while incorporating the constraints and using suitable algorithms.

## 2.5 Types of Clustering Algorithms with Detailed Description

### 1. k-Means Clustering

k-Means is one of the most widely used and perhaps the simplest unsupervised algorithms to solve the clustering problems. Using this algorithm, we classify a given data set through a certain number of predetermined clusters or “k” clusters. Each cluster is assigned a designated cluster center and they are placed as much as possible far away from each other. Subsequently, each point belonging gets associated with it to the nearest centroid till no point is left unassigned. Once it is done, the centers are re-calculated and the above steps are repeated. The algorithm converges at a point where the centroids cannot move any further. This algorithm targets to minimize an objective function called the squared error function  $F(V)$  :

$$F(V) = \sum_{i=1}^C \sum_{j=1}^{C_i} (\|X_i - V_j\|)^2$$

where,

$\|x_i - v_j\|$  is the distance between  $X_i$  and  $V_j$ .

$C_i$  is the count of data in cluster.  $C$  is the number of cluster centroids.

#### 2.5.1 Implementation:

In R, there is a built-in function `kmeans()` and in Python, we make use of `scikit-learn` cluster module which has the `KMeans` function. (`sklearn.cluster.KMeans`)

#### Advantages:

1. Can be applied to any form of data – as long as the data has numerical (continuous) entities.
2. Much faster than other algorithms.
3. Easy to understand and interpret.

#### Drawbacks:

1. Fails for non-linear data.
2. It requires us to decide on the number of clusters before we start the algorithm – where the user needs to use additional mathematical methods and also heuristic knowledge to verify the correct number of centers.
3. This cannot work for Categorical data.
4. Cannot handle outliers.

## Application Areas:

- a. Document clustering – high application area in Segmenting text-matrix related like data like DTM, TF-IDF etc.
- b. Banking and Insurance fraud detection where majority of the columns represent a financial figure – continuous data.
- c. Image segmentation.
- d. Customer Segmentation.

## 2. 6 Hierarchical Clustering Algorithm

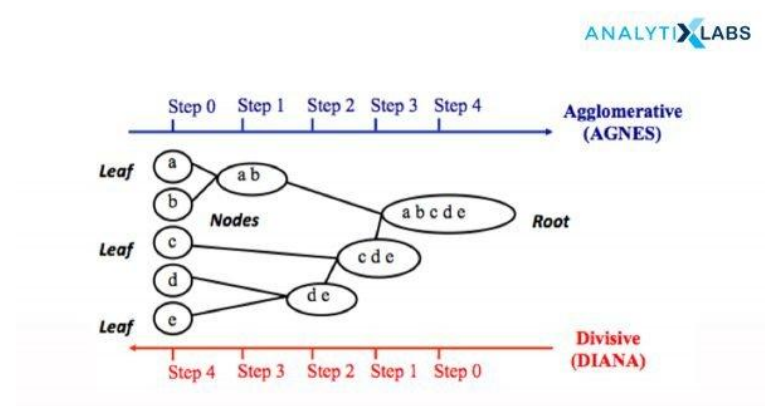
As discussed in the earlier section, Hierarchical clustering methods follow two approaches – Divisive and Agglomerative types. Their implementation family contains two algorithms respectively, the divisive DIANA (Divisive Analysis) and AGNES (Agglomerative Nesting) for each of the approaches.

### 2.7 DIANA or Divisive Analysis

As discussed earlier, the divisive approach begins with one single cluster where all the data points belong to. Then it is split into multiple clusters and the data points get reassigned to each of the clusters on the basis of the nearest distance measure of the pairwise distance between the data points. These distance measures can be Ward's Distance, Centroid Distance, average linkage, complete linkage or single linkage. Ideally, the algorithm continues until each data has its own cluster.

## Implementation:

In R, we make use of the `diana()` function from cluster package (`cluster::diana`)



### 2.8 Agglomerative Nesting or AGNES

AGNES starts by considering the fact that each data point has its own cluster, i.e., if there are  $n$  data rows, then the algorithm begins with  $n$  clusters initially. Then, iteratively, clusters that are most similar – again based on the distances as measured in DIANA – are now combined to form a larger cluster. The iterations are performed until we are left with one huge cluster that contains all the data-points.

### Implementation:

In R, we make use of the `agnes()` function from `cluster` package (`cluster::agnes()`) or the built-in `hclust()` function from the native `stats` package. In python, the implementation can be found in `scikit-learn` package via the `AgglomerativeClustering` function inside the `cluster` module (`sklearn.cluster.AgglomerativeClustering`)

### Advantages:

1. No prior knowledge about the number of clusters is needed, although the user needs to define a threshold for divisions.
2. Easy to implement across various forms of data and known to provide robust results for data generated via various sources. Hence it has a wide application area.

### Disadvantages:

1. The cluster division (DIANA) or combination (AGNES) is really strict and once performed, it cannot be undone and re-assigned in subsequent iterations or re-runs.
2. It has a high time complexity, in the order of  $O(n^2 \log n)$  for all the  $n$  data-points, hence cannot be used for larger datasets.
3. Cannot handle outliers and noise

## 2.9 Fuzzy C Means Algorithm – FANNY (Fuzzy Analysis Clustering)

This algorithm follows the fuzzy cluster assignment methodology of clustering. The working of FCM Algorithm is almost similar to the k-means – distance-based cluster assignment – however, the major difference is, as mentioned earlier, that according to this algorithm, a data point can be put into more than one cluster. This degree of belongingness can be clearly seen in the cost function of this algorithm as shown below:

$$\sum_{j=1}^k \sum_{x_i \in C_j} u_{ij}^m (x_i - u_j)^2$$

$u_{ij}$  is the degree of belongingness of data  $x_i$  to a cluster  $c_j$

$u_j$  is the cluster center of the cluster  $j$

$m$  is the fuzzifier.

So, just like the k-means algorithm, we first specify the number of clusters  $k$  and then assign the degree of belongingness to the cluster. We need to then repeat the algorithm till the `max_iterations` are reached, again which can be tuned according to the requirements.

**Implementation:**

In R, FCM can be implemented using `fanny()` from the `cluster` package (`cluster::fanny`) and in Python, fuzzy clustering can be performed using the `cmeans()` function from `skfuzzy` module. (`skfuzzy.cmeans`) and further, it can be adapted to be applied on new data using the predictor function (`skfuzzy.cmeans_predict`)

**Advantages:**

1. FCM works best for highly correlated and overlapped data, where k-means cannot give any conclusive results.
2. It is an unsupervised algorithm and it has a higher rate of convergence than other partitioning based algorithms.

**Disadvantages:**

1. We need to specify the number of clusters “k” prior to the start of the algorithm
2. Although convergence is always guaranteed, the process is very slow and this cannot be used for larger data.
3. Prone to errors if the data has noise and outliers.

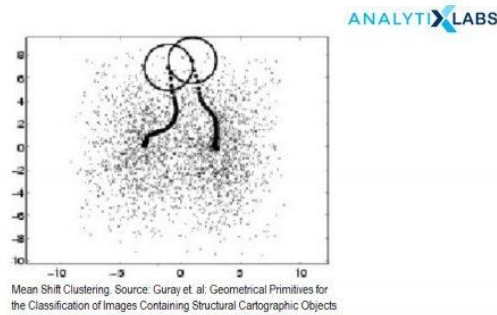
**Application Areas**

1. Used widely in Image Segmentation of medical imagery, especially the images generated by an MRI.
2. Market definition and segmentation.

**2.10 Mean Shift Clustering**

Mean shift clustering is a form of nonparametric clustering approach which not only eliminates the need for apriori specification of the number of clusters but also it removes the spatial and shape constraints of the clusters – two of the major problems from the most widely preferred k-means algorithm.

It is a density-based clustering algorithm where it firstly, seeks for stationary points in the density function. Then next, the clusters are eventually shifted to a region with higher density by shifting the center of the cluster to the mean of the points present in the current window. The shift if the window is repeated until no more points can be accommodated inside of that window.



### Implementation:

In R, `bmsClustering()` function from MeanShift package performs the clustering (`MeanShift::bmsClustering()`) and `MeanShift()` function in scikit learn package does the job in Python. (`sklearn.cluster.MeanShift`)

### Advantages:

1. Non-parametric and number of clusters need not be specified apriori.
2. Owing to it's density dependency, the shape of the cluster is not limited to circular or spherical.
3. More robust and more practical as it works for any form of data and the results are easily interpretable.

### Disadvantages:

1. The selection of the window radius is highly arbitrary and cannot be related to any business logic and selecting incorrect window size is never desirable.

### Applications:

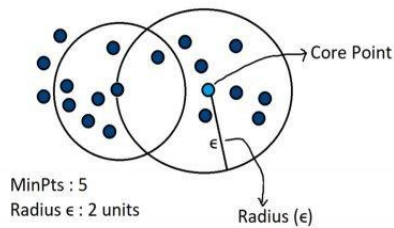
1. Image segmentation and computer vision – mostly used for handwritten text identification.
2. Image tracking in video analysis.

## 2.11 DBSCAN – Density-based Spatial Clustering

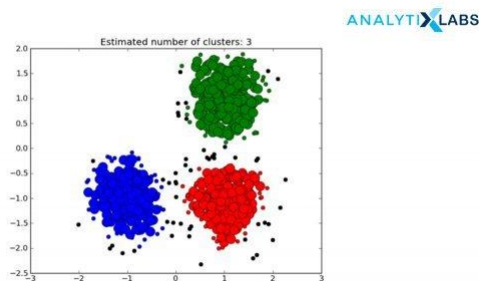
Density-based algorithms, in general, are pivotal in the application areas where we require non-linear cluster structures, purely based out of density. One of the ways how this principle can be made into reality is by using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm. There are two major underlying concepts in DBSCAN – one, Density Reachability and second, Density Connectivity. This helps the algorithm to differentiate and separate regions with varying degrees of density – hence creating clusters.

For implementing DBSCAN, we first begin with defining two important parameters – a radius parameter  $\epsilon$  ( $\epsilon$ ) and a minimum number of points within the radius ( $m$ ).





- a. The algorithm starts with a random data point that has not been accessed before and its neighborhood is marked according to  $\epsilon$ .
- b. If this contains all the  $m$  minimum points, then cluster formation begins – hence marking it as “visited” – if not, then it is labeled as “noise” for that iteration, which can get changed later.



- c. If a next data point belongs to this cluster, then subsequently the  $\epsilon$  neighborhood now around this point becomes a part of the cluster formed in the previous step. This step is repeated until there are no more data points that can follow Density Reachability and Density Connectivity.
- d. Once this loop is exited, it moves to the next “unvisited” data point and creates further clusters or noise.
- e. The algorithm converges when there are no more unvisited data points remain.

### Implementation:

In Python its implemented via `DBSCAN()` function from `scikit-learn` cluster module (`sklearn.cluster.DBSCAN`) and in R its implemented through `dbscan()` from `dbscan` package (`dbscan::dbscan(x, eps, minpts)`)

### Advantages:

1. Doesn't require prior specification of clusters.
2. Can easily deal with noise, not affected by outliers.
3. It has no strict shapes, it can correctly accommodate many data points.

### Disadvantages:

1. Cannot work with datasets of varying densities.
2. Sensitive to the clustering hyper-parameters – the  $\epsilon$  and the  $\text{min\_points}$ .
3. Fails if the data is too sparse.
4. The density measures (Reachability and Connectivity) can be affected by sampling.

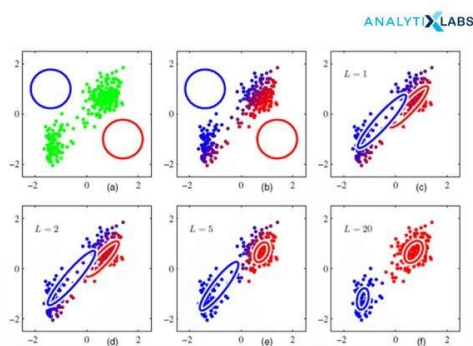
### Applications:

1. Used in document Network Analysis of text data for identifying plagiarism and copyrights in various scientific documents and scholarly articles.
2. Widely used in recommendation systems for various web applications and eCommerce websites.
3. Used in x-ray Crystallography to categorize the protein structure of a certain protein and to determine its interactions with other proteins in the strands.
4. Clustering in Social Network Analysis is implemented by DBSCAN where objects (points) are clustered based on the object's linkage rather than similarity.

## 2.12 Gaussian Mixed Models (GMM) with Expectation-Maximization Clustering

In Gaussian Mixed Models, we assume that the data points follow a Gaussian distribution, which is never a constraint at all as compared to the restrictions in the previous algorithms. Added to that, this assumption can lead to important selecting criteria for the shape of the clusters – that is, cluster shapes can be now quantified. This quantification happens by use to the two most common and simple metrics – mean and variance.

To find the mean and variance, Expectation-Maximization is used, which is a form of optimization function. This function starts with random Gaussian parameters, say  $\theta$ , and check if the Hypothesis confirms that a sample actually belongs to a cluster  $c$ . Once it does, we perform the maximization step where the Gaussian parameters are updated to fit the points assigned to the said cluster. Maximization step aims at increasing the likelihood of the sample belonging to the cluster distribution.

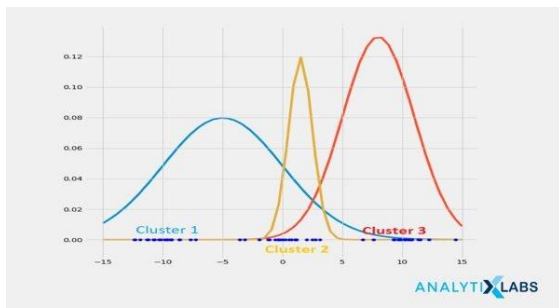


### Implementation:

In Python, it is implemented via the `GaussianMixture()` function from `scikit-learn`. (`sklearn.mixture.GaussianMixture`) and in R, it is implemented using `GMM()` from the `cluster` package. (`clusterR.GMM()`)

### Advantages:

1. The associativity of a data point to a cluster is quantified using probability metrics – which can be easily interpreted.
2. Proven to be accurate for real-time data sets.
3. Some versions of GMM allows for mixed membership of data points, hence it can be a good alternative to Fuzzy C Means to achieve fuzzy clustering.



### Disadvantages:

1. Complex algorithm and cannot be applicable to larger data
2. It is hard to find clusters if the data is not Gaussian, hence a lot of data preparation is required.

### Applications:

1. GMM has been more practically used in Topic Mining where we can associate multiple topics to a particular document (an atomic part of a text – a news article, online review, Twitter tweet etc.)
2. Spectral clustering, combined with Gaussian Mixed Models-EM is used in image processing.

### Applications of Clustering

We have seen numerous methodologies and approaches for clustering in machine learning and some of the important algorithms that implement those techniques. Let's have a quick overview of business applications of clustering and understand its role in Data Mining.

1. It is the backbone of search engine algorithms – where objects that are similar to each other must be presented together and dissimilar objects should be ignored. Also, it is required to fetch objects that are closely related to a search term, if not completely related.
2. A similar application of text clustering like search engine can be seen in academics where clustering can help in the associative analysis of various documents – which can be in-turn used in – plagiarism, copyright infringement, patent analysis etc.

- Used in image segmentation in bioinformatics where clustering algorithms have proven their worth in detecting cancerous cells from various medical imagery – eliminating the prevalent human errors and other bias.
- Netflix has used clustering in implementing movie recommendations for its users.
- News summarization can be performed using Cluster analysis where articles can be divided into a group of related topics.

### 2.13 Different Clustering Methods

Clustering Method	Description	Advantages	Disadvantages	Algorithms
<b>Hierarchical Clustering</b>	Based on top-to-bottom hierarchy of the data points to create clusters.	Easy to implement, the number of clusters need not be specified apriori, dendrograms are easy to interpret.	Cluster assignment is strict and cannot be undone, high time complexity, cannot work for a larger dataset	DIANA, AGNES, hclust etc.
<b>Partitioning methods</b>	Based on centroids and data points are assigned into a cluster based on its proximity to the cluster centroid	Easy to implement, faster processing, can work on larger data, easy to interpret the outputs	We need to specify the number of centroids apriori, clusters that get created are of inconsistent sizes and densities, Effected by noise and outliers	k-means, k-medians, k-modes
<b>Distribution-based Clustering</b>	Based on the probability distribution of the data, clusters are derived from various metrics like mean, variance etc.	Number of clusters need not be specified apriori, works on real-time data, metrics are easy to understand and tune	Complex algorithm and slow, cannot be scaled to larger data	Gaussian Mixed Models, DBCLASD
<b>Density-based Clustering (Model-based)</b>	Based on density of the data points, also known as model based clustering	Can handle noise and outliers, need not specify number of clusters in the start, clusters that are created are highly homogenous, no	Complex algorithm and slow, cannot be scaled to larger data	DENCAST, DBSCAN

ased methods)		restrictions on cluster shapes.		
<b>Fuzzy Clustering</b>	Based on Partitioning Approach but data points can belong to more than one cluster	Can work on highly overlapped data, a higher rate of convergence	We need to specify the number of centroids apriori, Effected by noise and outliers, Slow algorithm and cannot be scaled	Fuzzy C Means, Rough k means
<b>Constraint Based (Supervised Clustering)</b>	Clustering is directed and controlled by user constraints	Creates a perfect decision boundary, can automatically determine the outcome classes based on constraints, future data can be classified based on the training boundaries	Overfitting, high level of misclassification errors, cannot be trained on larger datasets	Decision Trees, Random Forest, Gradient Boosting

## 2.14 Conclusion

This chapter discussed the concept, its architecture and working of data mining, supervised and unsupervised learning. . We came to understand various types of clustering algorithms.. In the next chapter of the thesis we will discuss basically two algorithms (DBSCAN and Spectral Clustering) that we are going to use for our problem statement

## DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE AND SPECTRAL CLUSTERING

In the previous chapter we saw different types of algorithms in clustering. In this chapter we will discuss the above mentioned algorithms and their performance.

**3.1 Density-Based Clustering** refers to unsupervised learning methods that identify distinctive groups/clusters in the data, based on the idea that a cluster in data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a base algorithm for density-based clustering. It can discover clusters of different shapes and sizes from a large amount of data, which is containing noise and outliers.

The DBSCAN algorithm uses two parameters:

**minPts:** The minimum number of points (a threshold) clustered together for a region to be considered dense.

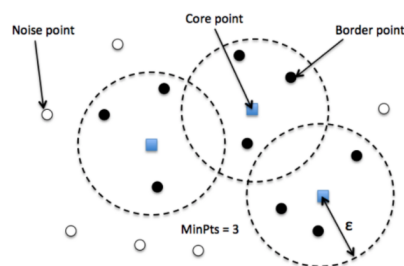
**eps ( $\epsilon$ ):** A distance measure that will be used to locate the points in the neighborhood of any point.

These parameters can be understood if we explore two concepts called Density Reachability and Density Connectivity.

**Reachability** in terms of density establishes a point to be reachable from another if it lies within a particular distance (eps) from it.

**Connectivity**, on the other hand, involves a transitivity based chaining-approach to determine whether points are located in a particular cluster. For example, p and q points could be connected if  $p \rightarrow r \rightarrow s \rightarrow t \rightarrow q$ , where  $a \rightarrow b$  means b is in the neighborhood of a.

There are three types of points after the DBSCAN clustering is complete:



**Core** — This is a point that has at least  $m$  points within distance  $n$  from itself.

**Border** — This is a point that has at least one Core point at a distance  $n$ .

**Noise** — This is a point that is neither a Core nor a Border. And it has less than  $m$  points within distance  $n$  from itself.

### 3.1.1 Algorithmic steps for DBSCAN clustering

The algorithm proceeds by arbitrarily picking up a point in the dataset (until all points have been visited).

If there are at least 'minPoint' points within a radius of ' $\epsilon$ ' to the point then we consider all these points to be part of the same cluster.

The clusters are then expanded by recursively repeating the neighborhood calculation for each neighboring point



DBSCAN in action

### 3.1.2 Parameter Estimation

Every data mining task has the problem of parameters. Every parameter influences the algorithm in specific ways. For DBSCAN, the parameters  $\epsilon$  and **minPts** are needed.

**minPts**: As a rule of thumb, a minimum *minPts* can be derived from the number of dimensions  $D$  in the data set, as  $\text{minPts} \geq D + 1$ . The low value  $\text{minPts} = 1$  does not make sense, as then every point on its own will already be a cluster. With  $\text{minPts} \leq 2$ , the result will be the same as of [hierarchical clustering](#) with the single link metric, with the dendrogram cut at height  $\epsilon$ . Therefore, *minPts* must be chosen at least 3. However, larger values are usually better for data sets with noise and will yield more significant clusters. As a rule of thumb,  $\text{minPts} = 2 \cdot \text{dim}$  can be used, but it may be necessary to choose larger values for very large data, for noisy data or for data that contains many duplicates.

$\epsilon$ : The value for  $\epsilon$  can then be chosen by using a [k-distance graph](#), plotting the distance to the  $k = \text{minPts} - 1$  nearest neighbor ordered from the largest to the smallest value. Good values of  $\epsilon$  are where this plot shows an “elbow”: if  $\epsilon$  is chosen much too small, a large part of the data will not be clustered; whereas for a too high value of  $\epsilon$ , clusters will merge and the majority of objects will be in the same cluster. In general, small values of  $\epsilon$  are preferable, and as a rule of thumb, only a small fraction of points should be within this distance of each other.

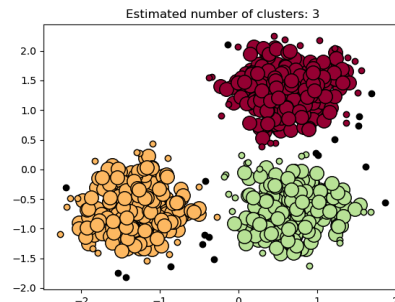
**Distance function**: The choice of distance function is tightly linked to the choice of  $\epsilon$ , and has a major impact on the outcomes. In general, it will be necessary to first identify a reasonable measure of similarity for the data set, before the parameter  $\epsilon$  can be chosen. There is no estimation for this parameter, but the distance functions need to be chosen appropriately for the data set.

### 3.1.3 DBSCAN python implementation using sklearn

Let us first apply DBSCAN to cluster spherical data.

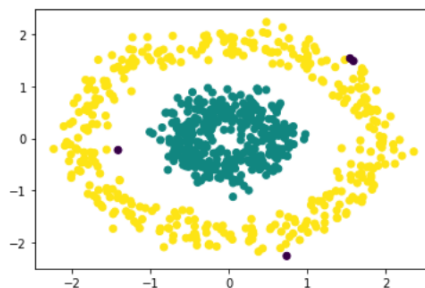
We first generate 750 spherical training data points with corresponding labels. After that standardize the features of your training data and at last, apply DBSCAN from the sklearn library.

Estimated number of clusters: 3  
 Estimated number of noise points: 18  
 Homogeneity: 0.953  
 Completeness: 0.883  
 V-measure: 0.917  
 Adjusted Rand Index: 0.952  
 Adjusted Mutual Information: 0.883  
 Silhouette Coefficient: 0.626



DBSCAN to cluster spherical data

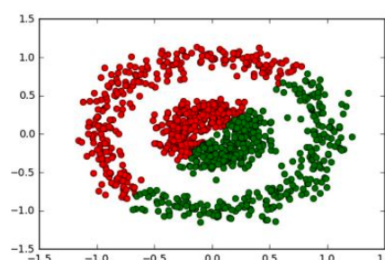
The black data points represent outliers in the above result. Next, apply DBSCAN to cluster



non-spherical data.

DBSCAN to cluster non-spherical data

Which is absolutely perfect. If we compare with K-means it would give a completely incorrect output like:



K-means clustering result

### 3.1.4 The complexity of DBSCAN Clustering Algorithm

**Best Case:** If an indexing system is used to store the dataset such that neighborhood queries are executed in logarithmic time, we get  $O(n \log n)$  average runtime complexity.

**Worst Case:** Without the use of index structure or on degenerated data (e.g. all points within a distance less than  $\epsilon$ ), the worst-case run time complexity remains  $O(n^2)$ .

**Average Case:** Same as best/worst case depending on data and implementation of the algorithm.



### 3.1.5 Conclusion

Density-based clustering algorithms can learn clusters of arbitrary shape, and with the Level Set Tree algorithm, one can learn clusters in datasets that exhibit wide differences in density.

However, I should point out that these algorithms are somewhat more arduous to tune contrasted to parametric clustering algorithms like K-Means. Parameters like the epsilon for DBSCAN or for the Level Set Tree are less intuitive to reason about compared to the number of clusters parameter for K-Means, so it's more difficult to choose good initial parameter values for these algorithms.

That is all for this article. I hope you guys have enjoyed reading it, please share your suggestions/views/questions in the comment section.

## 3.2 Spectral Clustering

Spectral clustering is a technique with roots in graph theory, where the approach is used to identify communities of nodes in a graph based on the edges connecting them. The method is flexible and allows us to cluster non graph data as well. Spectral clustering uses information from the eigenvalues (spectrum) of special matrices built from the graph or the data set. We'll learn how to construct these matrices, interpret their spectrum, and use the eigenvectors to assign our data to clusters.

### 3.2.1 Eigenvectors and Eigenvalues

Critical to this discussion is the concept of eigenvalues and eigenvectors. For a matrix  $A$ , if there exists a vector  $x$  which isn't all 0's and a scalar  $\lambda$  such that  $Ax = \lambda x$ , then  $x$  is said to be an eigenvector of  $A$  with corresponding eigenvalue  $\lambda$ .

We can think of the matrix  $A$  as a function which maps vectors to new vectors. Most vectors will end up somewhere completely different when  $A$  is applied to them, but eigenvectors only change in magnitude. If you drew a line through the origin and the eigenvector, then after the mapping, the eigenvector would still land on the line. The amount which the vector is scaled along the line depends on  $\lambda$ .

26

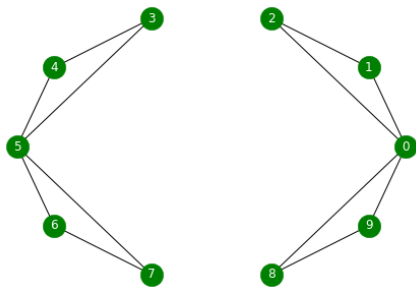
Eigenvectors are an important part of linear algebra, because they help describe the dynamics of systems represented by matrices. There are numerous applications which utilize eigenvectors, and we'll use them directly here to perform spectral clustering.

### 3.2.2 Graphs

Graphs are a natural way to represent many types of data. A graph is a set of nodes with a corresponding set of edges which connect the nodes. The edges may be directed or undirected and can even have weights associated with them.

A network of routers on the internet can easily be represented as a graph. The routers are the nodes, and the edges are the connections between pairs of routers. Some routers might only allow traffic in one direction, so the edges could be directed to represent which direction traffic can flow. The weights on the edges could represent the bandwidth available along that edge. With this setup, we could then query the graph to find efficient paths for transmitting data from one router to another across the network.

Let's use the following undirected graph as a running example:



This graph has 10 nodes and 12 edges. It also has two connected components  $\{0,1,2,8,9\}$  and  $\{3,4,5,6,7\}$ . A connected component is a maximal subgraph of nodes which all have paths to the rest of the nodes in the subgraph.

Connected components seem important, if our task is to assign these nodes to communities or clusters. A simple idea would be to make each connected component its own cluster. This seems reasonable for our example graph, but it's possible that the entire graph might be connected, or that the connected components are very large. There could also be smaller structures within a connected component which are good candidates for communities. We'll shortly see the importance of this connected component idea for spectral clustering.

### 3.2.3 Adjacency Matrix

We can represent our example graph as an adjacency matrix, where the row and column indices represent the nodes, and the entries represent the absence or presence of an edge between the nodes. The adjacency matrix for our example graph looks like this:

In the matrix, we see that row 0, column 1 has a value of 1. That means that there is an edge connecting node 0 with node 1. If the edges were weighted, the weights of the edges would go in this matrix instead of just 1s and 0s. Since our graph is undirected, the entries for row  $i$ , col  $j$  will be equal to the entry at row  $j$ , col  $i$ . The last thing to notice is that the diagonal of this matrix is all 0, since none of our nodes have edges to themselves.

### 3.2.4 Degree Matrix

The degree of a node is how many edges connect to it. In a directed graph we could talk about in-degree and out-degree, but in this example we just have degree since the edges go both ways. Looking at our graph, we see that node 0 has degree 4, since it has 4 edges. We could also get the degree by taking the sum of the node's row in the adjacency matrix.

The degree matrix is a diagonal matrix where the value at entry  $(i, i)$  is the degree of node  $i$ . Let's find the degree matrix for our example:

First, we took the sum across axis 1 (the rows) of our adjacency matrix, and then we put those values into a diagonal matrix. From the degree matrix, we can easily see that nodes 0 and 5 have 4 edges, while the rest of the nodes have only 2.

### 3.2.5 Graph Laplacian

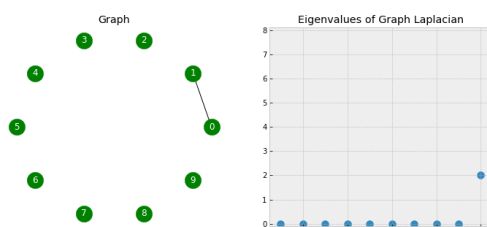
Now we're going to calculate the Graph Laplacian. The Laplacian is just another matrix representation of a graph. It has several beautiful properties, which we will take advantage of for spectral clustering. To calculate the normal Laplacian (there are several variants), we just subtract the adjacency matrix from our degree matrix:

The Laplacian's diagonal is the degree of our nodes, and the off diagonal is the negative edge weights. This is the representation we are after for performing spectral clustering.

### 3.2.6 Eigenvalues of Graph Laplacian

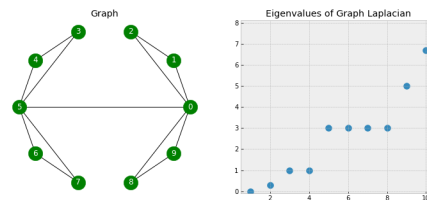
28

As mentioned, the Laplacian has some beautiful properties. To get a sense for this, let's examine the eigenvalues associated with the Laplacian as I add edges to our graph:



We see that when the graph is completely disconnected, all ten of our eigenvalues are 0. As we add edges, some of our eigenvalues increase. In fact, the number of 0 eigenvalues corresponds to the number of connected components in our graph!

Look closely as that final edge is added, connecting the two components into one. When this happens,



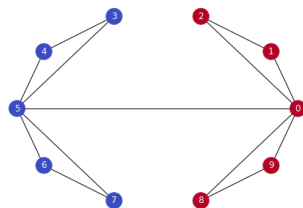
all of the eigenvalues but one have been lifted:  
of 0-eigenvalues is number of connected components.

Number

The first eigenvalue is 0 because we only have one connected component (the whole graph is connected). The corresponding eigenvector will always have constant values (in this example all the values are close to 0.32).

The first nonzero eigenvalue is called the spectral gap. The spectral gap gives us some notion of the density of the graph. If this graph was densely connected (all pairs of the 10 nodes had an edge), then the spectral gap would be 10.

The second eigenvalue is called the Fiedler value, and the corresponding vector is the Fiedler vector. The Fiedler value approximates the minimum graph cut needed to separate the graph into two connected components. Recall, that if our graph was already two connected components, then the Fiedler value would be 0. Each value in the Fiedler vector gives us information about which side of the cut that node belongs. Let's color the nodes based on whether their entry in the Fiedler vector is



positive or not:

Nodes colored based on whether their entry in the Fiedler Vector is  $>0$ .

29

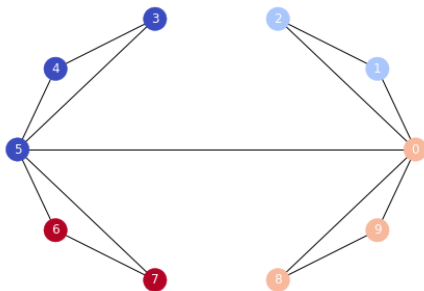
This simple trick has segmented our graph into two clusters! Why does this work? Remember that zero eigenvalues represent connected components. Eigenvalues near zero are telling us that there is almost a separation of two components. Here we have a single edge, that if it didn't exist, we'd have two separate components. So the second eigenvalue is small.

To summarize what we know so far: the first eigenvalue is 0 because we have one connected component. The second eigenvalue is near 0 because we're one edge away from having two connected components. We also saw that the vector associated with that value tells us how to separate the nodes into those approximately connected components.

You may have noticed that the next two eigenvalues are also pretty small. That tells us that we are "close" to having four separate connected components. In general, we often look for the first large gap

between eigenvalues in order to find the number of clusters expressed in our data. See the gap between eigenvalues four and five?

Having four eigenvalues before the gap indicates that there is likely four clusters. The vectors associated with the first three positive eigenvalues should give us information about which three cuts need to be made in the graph to assign each node to one of the four approximated components. Let's build a matrix from these three vectors and perform K-Means clustering to determine the assignments:

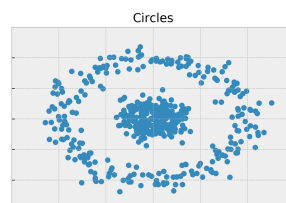


Spectral Clustering for 4 clusters.

The graph has been segmented into the four quadrants, with nodes 0 and 5 arbitrarily assigned to one of their connected quadrants. That is really cool, and that is spectral clustering!

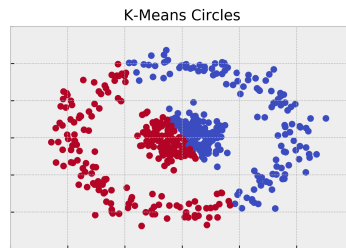
To summarize, we first took our graph and built an adjacency matrix. We then created the Graph Laplacian by subtracting the adjacency matrix from the degree matrix. The eigenvalues of the Laplacian indicated that there were four clusters. The vectors associated with those eigenvalues contain information on how to segment the nodes. Finally, we performed K-Means on those vectors in order to get the labels for the nodes. Next, we'll see how to do this for arbitrary data.

**Spectral Clustering Arbitrary Data** Look at the data below. The points are drawn from two concentric circles with some noise added. We'd like an algorithm to be able to cluster these points into the two



circles that generated them.

This data isn't in the form of a graph. So first, let's just try a cookie cutter algorithm like K-Means. K-Means will find two centroids and label the points based on which centroid they are closest too.



Here are the results:

Obviously, K-Means wasn't going to work. It operates on euclidean distance, and it assumes that the clusters are roughly spherical. This data (and often real world data) breaks these assumptions. Let's try to tackle this with spectral clustering.

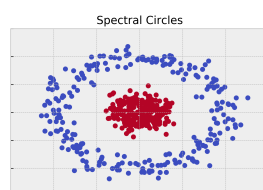
### 3.2.7 Nearest Neighbors Graph

There are a couple of ways to treat our data as a graph. The easiest way is to construct a k-nearest neighbors graph. A k-nearest neighbors graph treats every data point as a node in a graph. An edge is then drawn from each node to its k nearest neighbors in the original space. Generally, the algorithm isn't too sensitive of the choice of k. Smaller numbers like 5 or 10 usually work pretty well.

Look at the picture of the data again, and imagine that each point is connected to its 5 closest neighbors. Any point in the outer ring should be able to follow a path along the ring, but there won't be any paths into the inner circle. It's pretty easy to see that this graph will have two connected components: the outer ring and the inner circle.

Since we're only separating this data into two components, we should be able to use our Fiedler vector trick from before. Here's the code I used to perform spectral clustering on this data:

31



And here are the results:

### 3.2.8 Conclusion

We've covered the theory and application of spectral clustering for both graphs and arbitrary data. Spectral clustering is a flexible approach for finding clusters when your data doesn't meet the requirements of other common algorithms.

First, we formed a graph between our data points. The edges of the graph capture the similarities between the points. The eigenvalues of the Graph Laplacian can then be used to find the best number of clusters, and the eigenvectors can be used to find the actual cluster labels.

## CHAPTER 4

### Literature Review

In the previous chapter we read about DBSCAN and spectral clustering and its application together with some extended functionalities of extra programming. In this chapter we will discuss about the progress and research work done in the field of clustering and prediction.

**4.1 Convex-Hull & DBSCAN Clustering to Predict Future Weather** by Schakraborty and Dr Ratul, their analysis is based on the observation of the air pollution data has been collected from the “West Bengal Air Pollution Control Board”

They used Convex-Hull to get structural format of air molecules data, so we found a structural database from a large dataset in one step. Now we can create cluster using DBSCAN which give a particular radius, min value, Threshold Distance, in case of K-Mean algorithm we found only min value and threshold distance, so the cluster centre chosen by manually which is completely unethical. Here we use incremental DBSCAN, if some new data want to store in the previous database then check fast the nearest outer cluster data that can create a new cluster with criteria of min threshold value and min point, if not satisfy DBSCAN condition then choose the nearest cluster header and again check criteria of DBSCAN [5,6]. If satisfy then include on that particular cluster or treated as outer cluster. For weather prediction we use priority based protocol, which give some statistical value of previous year data to predict upcoming weather condition. Now this protocol can predict per day weather forecast and also predict long term nature of weather.

**4.2 An Efficient Weather Forecasting System using Artificial Neural Network** by Pooja Malik, Prof. Saranjeet Singh, Binni Arora

This paper propose a new technique of weather forecasting by using Feed-forward ANN. The data is taken from Rice Research center (Kaul) Haryana. In this paper data is trained by LM algorithm. This is the fastest method among other weather forecasting methods. As there are many BP algorithm but among them Levenberg BP has better learning rate.

A layered feed forward neural network has layers, or subgroups of processing elements. The first step in training is to create network object. Feed-forward networks with more layers might learn complex relationships more quickly. Feedforward network has no feedback. FFN allows signal to travel one way only. Feed-forward ANN are straight forward network that uses inputs with outputs. To create a network, you provide typical input and output values that initialize weight and bias values and determine the size of the output layer. In the training process the weights are so adjusted that mean squared error obtained between experimental and obtained result can be minimized. This network consists of multiple layers. This architecture consists of three layers, first is input layer, second is hidden layer and third is output layer.

Weather forecasting is done by collecting past and current data of the atmosphere then by using this data train the neural network. There are generally four steps in the training process: 1. Assemble the training data. 2. Create the network object. 3. Train the network. 4. Simulate the network response to new inputs.



### **4.3 Time Series forecasting through clustering by Vipul Kedia, Vamshidhar Thummala, Kamalar Karlapalem**

Time series forecasting or time series prediction takes existing data  $x_{n-1}, x_{n-2}, x_{n-3} \dots x_i$  to  $x_{i+1}, x_{i+2}, x_{i+3} \dots x_i$  the goal is to model the existing data series to forecast the weather of the future.

### **4.5 Advances in atmospheric sciences improving multimodal weather forecast using FSU by TN Krishnamurti, AD Sagadevan, A Chakraborty, AK Mishra and A Simon**

In this paper we present the current capabilities for numerical weather prediction of precipitation over China using a suite of ten multimodels and our superensemble based forecasts. Our suite of models includes the operational suite selected by NCARs TIGGE archives for the THORPEX Program. These are: ECMWF, UKMO, JMA, NCEP, CMA, CMC, BOM, MF, KMA and the CPTEC models. The superensemble strategy includes a training and a forecasts phase, for these the periods chosen for this study include the months February through September for the years 2007 and 2008. This paper addresses precipitation forecasts for the medium range i.e. Days 1 to 3 and extending out to Day 10 of forecasts using this suite of global models. For training and forecasts validations we have made use of an advanced TRMM satellite based rainfall product. We make use of standard metrics for forecast validations that include the RMS errors, spatial correlations and the equitable threat scores. The results of skill forecasts of precipitation clearly demonstrate that it is possible to obtain higher skills for precipitation forecasts for Days 1 through 3 of forecasts from the use of the multimodel superensemble as compared to the best model of this suite. Between Days 4 to 10 it is possible to have very high skills from the multimodel superensemble for the RMS error of precipitation. Those skills are shown for a global belt and especially over China. Phenomenologically this product was also found very useful for precipitation forecasts for the Onset of the South China Sea monsoon, the life cycle of the mei-yu rains and post typhoon landfall heavy rains and flood events. The higher skills of the multimodel superensemble make it a very useful product for such real time events.

### **4.6 Conclusion**

In this chapter we discussed different literature related work related to forecasting based on the performance and feasibility.

In the next chapter we will propose an idea for that to use our algorithm and check the performance.

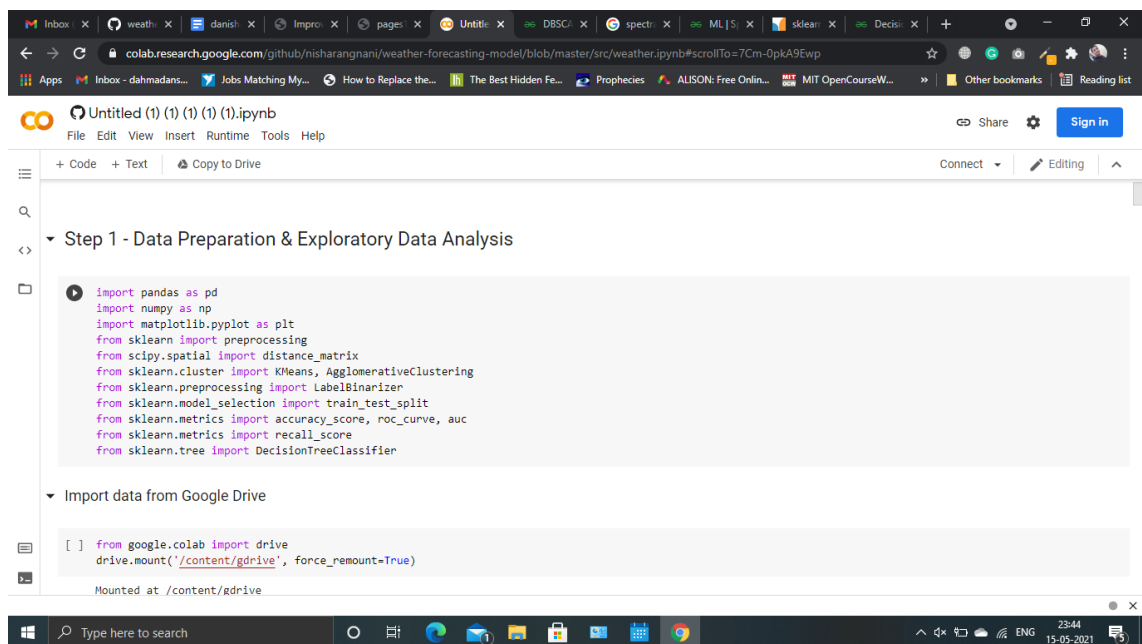
## CHAPTER 5

### Proposed Scheme

In the previous chapter, we discussed all the works which were done in the successful prediction of weather. In this chapter we have proposed an idea which can help in solving that existing problem using some other method. We have given a improved model of implementation of forecasting through clustering.

#### 5.1 Introduction

5.1.1 In our model will use cluster analysis and Decision Tree Modeling. We will first clean the data using numpy and pandas and PCA



The screenshot shows a Google Colab notebook interface. The browser tabs at the top include 'Inbox', 'weath', 'danish', 'Impro', 'pages', 'Untitled', 'DBSC', 'spectr', 'ML15', 'sklearn', 'Decis', and others. The notebook title is 'Untitled (1) (1) (1) (1).ipynb'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. The toolbar shows '+ Code', '+ Text', 'Copy to Drive', 'Connect', 'Editing', and 'Sign in'. The code editor displays the following code:

```
Step 1 - Data Preparation & Exploratory Data Analysis

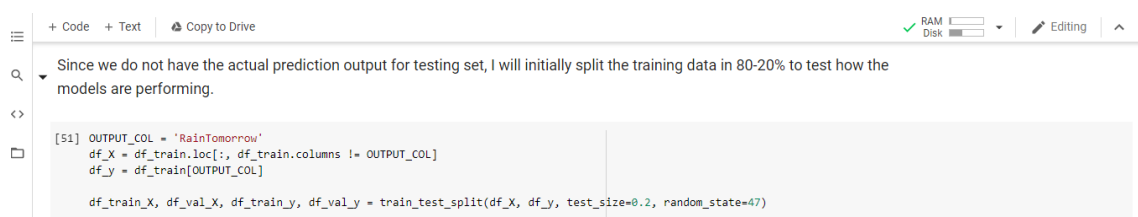
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from scipy.spatial import distance_matrix
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_curve, auc
from sklearn.metrics import recall_score
from sklearn.tree import DecisionTreeClassifier

Import data from Google Drive

[ ] from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)

Mounted at /content/gdrive
```

5.1.2 Since we do not have the actual prediction output for testing set, I will initially split the training data in 80-20% to test how the models are performing.



The screenshot shows a Google Colab notebook interface. The toolbar shows '+ Code', '+ Text', 'Copy to Drive', 'RAM', 'Disk', 'Editing', and a refresh icon. The code editor displays the following code:

```
Since we do not have the actual prediction output for testing set, I will initially split the training data in 80-20% to test how the models are performing.

[51] OUTPUT_COL = 'RainTomorrow'
df_X = df_train.loc[:, df_train.columns != OUTPUT_COL]
df_y = df_train[OUTPUT_COL]

df_train_X, df_val_X, df_train_y, df_val_y = train_test_split(df_X, df_y, test_size=0.2, random_state=47)
```

In order to use unsupervised learning algorithms like clustering for classifying data, while training - I am creating 2 clusters and finding out majority label from both clusters and that

would the final label for that cluster. While testing, I used the above model to add testing data points to above created cluster - if that row falls in the cluster that is labeled as 'Yes' - then it'll be predicted to rain for that row.

5.1.3 Creating reusable functions below to get majority label and classify data from the 2 clusters (for both DBSCAN and Spectral Clustering).

5.1.4 Since we're dealing with Weather forecasting problem, I'll go ahead with measuring Recall for model evaluation, as to predict if it'll rain, we can afford to have False positives. False negatives (predicting that it will not rain, but it ends up raining on that day) is more harmful here.

5.1.5 Also, it also correct to measure accuracy score as we half almost equal number of Yes and No labels in the training set.

+ Code
+ Text
Copy to Drive
RAM
Disk
Editing

Also, since location is no more useful, we can drop location attribute.

```
[140] lb = LabelBinarizer()
df_train['RainToday'] = lb.fit_transform(df_train['RainToday'])
df_train['RainTomorrow'] = lb.fit_transform(df_train['RainTomorrow'])
df_test['RainToday'] = lb.fit_transform(df_test['RainToday'])

df_train.head(10)
```

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed	Humidity	Pressure	Cloud	Temp	RainToday	RainTomorrow
0	NorahHead	18.9	23.7	0.0	5.177570	6.53058	41.0	28.0	55.0	1026.0	5.175647	22.4	0	1
1	Nuriotpa	11.1	20.8	0.0	4.800000	8.30000	39.0	26.0	48.0	1014.4	7.000000	19.5	0	0
2	GoldCoast	15.9	19.5	17.6	5.177570	6.53058	44.0	9.0	99.0	1028.5	5.175647	17.8	1	1
3	Bendigo	0.0	14.9	0.0	3.697044	6.53058	35.0	19.0	55.0	1023.0	4.000000	14.2	0	1
4	Walpole	9.1	22.7	0.0	5.177570	6.53058	41.0	7.0	40.0	1027.1	5.175647	22.5	0	0
5	MelbourneAirport	6.6	16.3	8.4	3.400000	6.90000	50.0	30.0	55.0	1021.4	7.000000	14.6	1	0
6	Mildura	9.7	20.6	0.0	1.000000	4.20000	61.0	20.0	35.0	1015.8	7.000000	19.5	0	0

Decision Tree

```
[ ] dtc = DecisionTreeClassifier(random_state=47, criterion="entropy", min_samples_split=9, max_depth=8, min_samples_leaf=6, max_leaf_nodes= 56).fit(df_train_X, df_t
val_pred_dtc = dtc.predict(df_val_X)
print(accuracy_score(df_val_y, val_pred_dtc))

0.7680838784147749

[ ] print(recall_score(df_val_y, val_pred_dtc))

0.7394908229721728
```

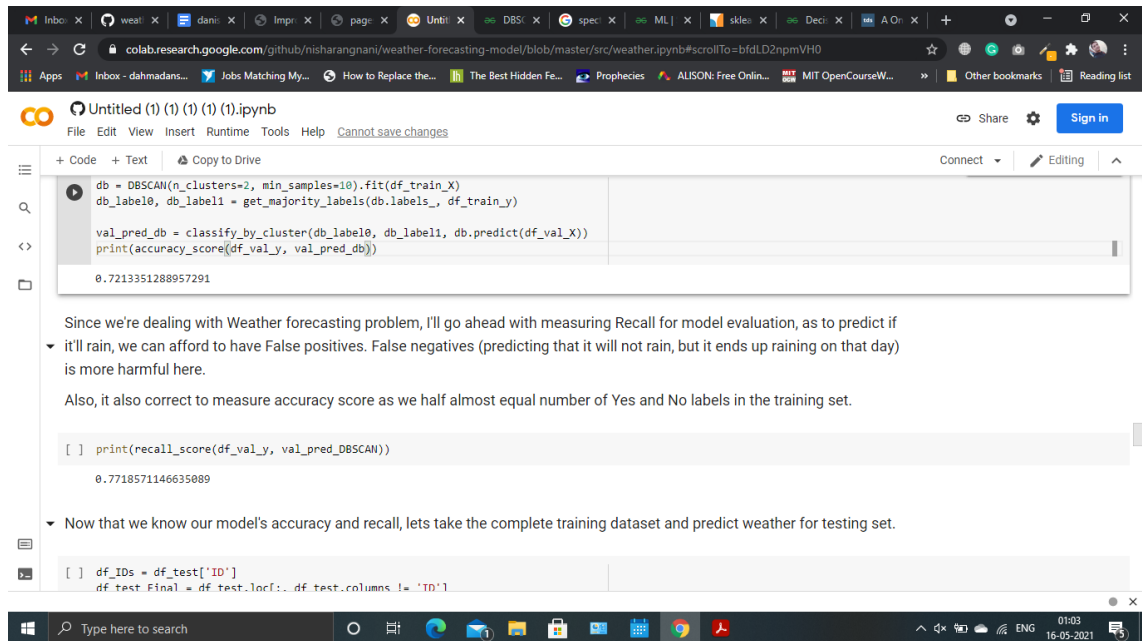
5.1.6 In order to use unsupervised learning algorithms like clustering for classifying data, while training - I am creating 2 clusters and finding out majority label from both clusters and that would the final label for that cluster. While testing, I used the above model to add testing data points to above created cluster - if that row falls in the cluster that is labeled as 'Yes' - then it'll be predicted to rain for that row.

```
[56] def get_majority_labels(cluster_labels, df_true_labels):
    cluster_0_rain, cluster_0_no_rain = 0, 0
    cluster_1_rain, cluster_1_no_rain = 0, 0

    for i in range(len(cluster_labels)):
        if df_true_labels.iloc[i] == 1:
            if cluster_labels[i] == 0:
                cluster_0_rain += 1
            else:
                cluster_1_rain += 1
        else:
            if cluster_labels[i] == 0:
                cluster_0_no_rain += 1
            else:
                cluster_1_no_rain += 1
    cluster_0_majority = 1 if cluster_0_rain > cluster_0_no_rain else 0
    cluster_1_majority = 1 if cluster_1_rain > cluster_1_no_rain else 0
    return cluster_0_majority, cluster_1_majority
```

### 5.1.7 Creating reusable functions below to get majority label and classify data from the 2 clusters (for both DBSCAN and Spectral).

Now we use DBSCAN and check the recall score



The screenshot shows a Jupyter Notebook with the following code and output:

```
db = DBSCAN(n_clusters=2, min_samples=10).fit(df_train_X)
db_label0, db_label1 = get_majority_labels(db.labels_, df_train_y)

val_pred_db = classify_by_cluster(db_label0, db_label1, db.predict(df_val_X))
print(accuracy_score(df_val_y, val_pred_db))
```

0.7213351288957291

Since we're dealing with Weather forecasting problem, I'll go ahead with measuring Recall for model evaluation, as to predict if it'll rain, we can afford to have False positives. False negatives (predicting that it will not rain, but it ends up raining on that day) is more harmful here.

Also, it's also correct to measure accuracy score as we have almost equal number of Yes and No labels in the training set.

```
[ ] print(recall_score(df_val_y, val_pred_DBSCAN))
```

0.7718571146635089

Now that we know our model's accuracy and recall, let's take the complete training dataset and predict weather for testing set.

```
[ ] df_ids = df_test['ID']
df_test_Final = df_test.loc[:, df_test.columns != 'ID']
```

DBSCAN recall score is 0.77

## 5.2 Compute the recall

The recall is the ratio  $tp / (tp + fn)$  where  $tp$  is the number of true positives and  $fn$  the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The best value is 1 and the worst value is 0.

Read more in the User Guide .

### Parameters

**y\_true** : 1d array-like, or label indicator array / sparse matrix

Ground truth (correct) target values.

**y\_pred** : 1d array-like, or label indicator array / sparse matrix

Estimated targets as returned by a classifier.

**labels** : list, optional

The set of labels to include when `average != 'binary'`, and their order if `average` is `None`. Labels present in the data can be excluded, for example to calculate a multiclass average ignoring a majority negative class, while labels not present in the data will result in 0 components in a macro average. For multilabel targets, labels are column indices. By default, all labels in `y_true` and `y_pred` are used in sorted order.

**pos\_label** : str or int, 1 by default

The class to report if `average='binary'`. Until version 0.18 it is necessary to set `pos_label=None` if seeking to use another averaging method over binary targets.

**average** : string, [None, 'binary' (default), 'micro', 'macro', 'samples', 'weighted']

This parameter is required for multiclass/multilabel targets. If None, the scores for each class are returned. Otherwise, this determines the type of averaging performed on the data:

'binary':

Only report results for the class specified by `pos_label`. This is applicable only if targets (`y_{true,pred}`) are binary.

'micro':

Calculate metrics globally by counting the total true positives, false negatives and false positives.

'macro':

Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

'weighted':

Calculate metrics for each label, and find their average, weighted by support (the number of true instances for each label). This alters 'macro' to account for label imbalance; it can result in an F-score that is not between precision and recall.

'samples':

Calculate metrics for each instance, and find their average (only meaningful for multilabel classification where this differs from [accuracy\\_score](#)).

Note that if `pos_label` is given in binary classification with `average != 'binary'`, only that positive class is reported. This behavior is deprecated and will change in version 0.18.

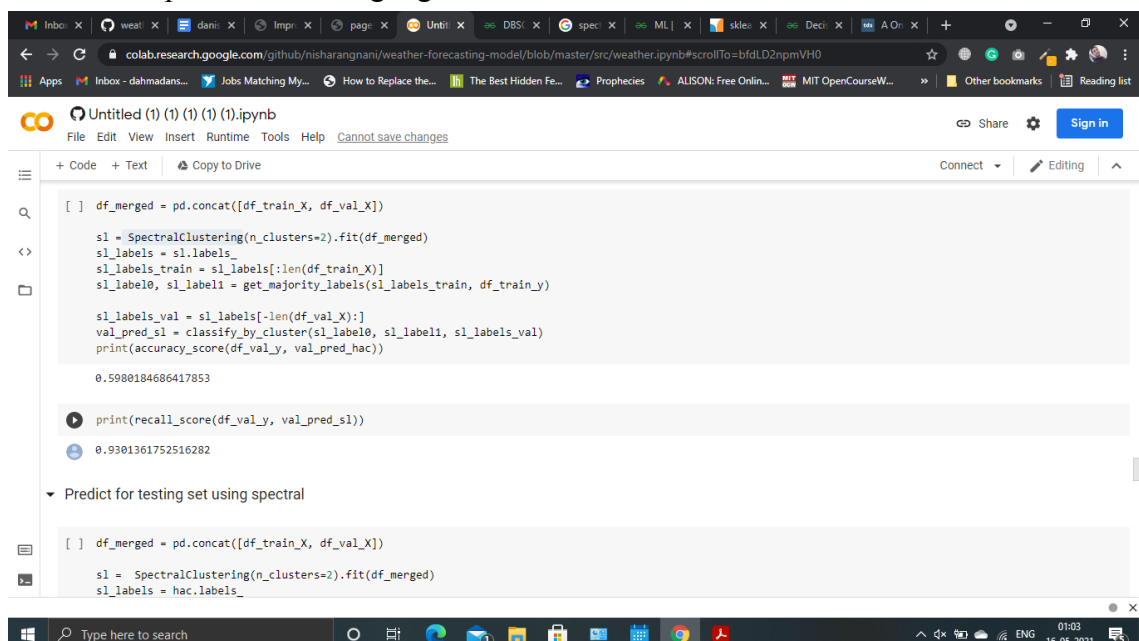
`sample_weight` : array-like of shape = `[n_samples]`, optional  
Sample weights.

Returns

`recall` : float (if average is not None) or array of float, shape = `[n_unique_labels]`

Recall of the positive class in binary classification or weighted average of the recall of each class for the multiclass task.

### 5.3 We use spectral clustering algorithm now



The screenshot shows a Jupyter Notebook with the following code and output:

```
[ ] df_merged = pd.concat([df_train_X, df_val_X])

s1 = SpectralClustering(n_clusters=2).fit(df_merged)
s1_labels = s1.labels_
s1_labels_train = s1_labels[:len(df_train_X)]
s1_label0, s1_label1 = get_majority_labels(s1_labels_train, df_train_y)

s1_labels_val = s1_labels[-len(df_val_X):]
val_pred_s1 = classify_by_cluster(s1_label0, s1_label1, s1_labels_val)
print(accuracy_score(df_val_y, val_pred_hac))

0.5980184686417853

print(recall_score(df_val_y, val_pred_s1))

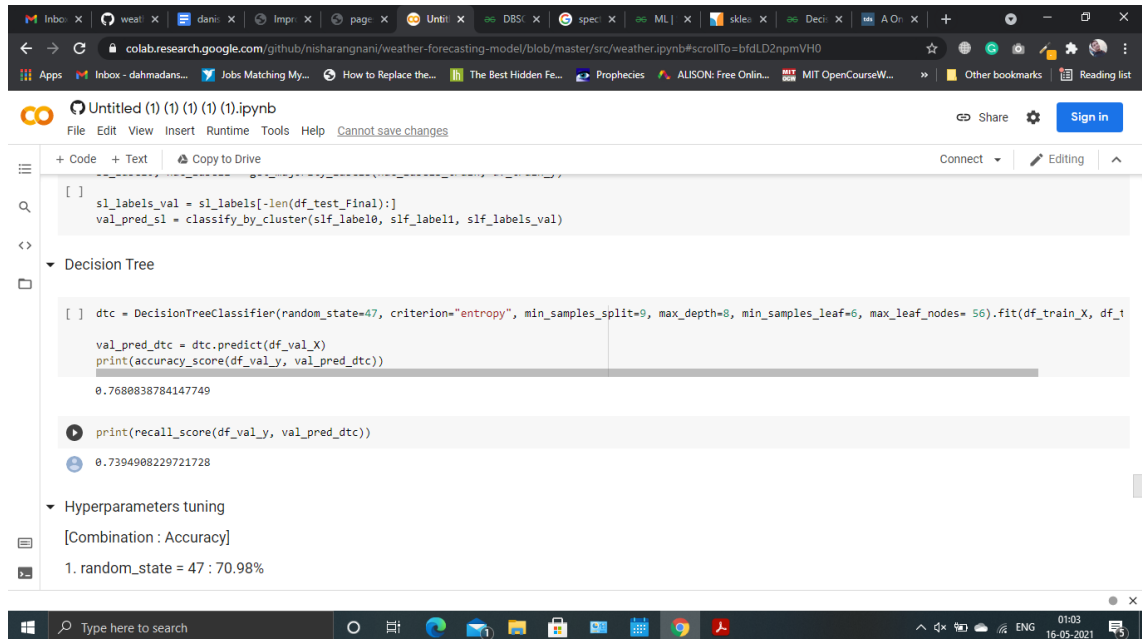
0.9301361752516282
```

Below the code, there is a section titled "Predict for testing set using spectral" with the following code:

```
[ ] df_merged = pd.concat([df_train_X, df_val_X])

s1 = SpectralClustering(n_clusters=2).fit(df_merged)
s1_labels = hac.labels_
```

### 5.3 Spectral clustering recall score is 0.93



```
[ ]
s1_labels_val = s1_labels[-len(df_test_Final):]
val_pred_s1 = classify_by_cluster(s1f_label0, s1f_label1, s1f_labels_val)

Decision Tree

[ ] dtc = DecisionTreeClassifier(random_state=47, criterion="entropy", min_samples_split=9, max_depth=8, min_samples_leaf=6, max_leaf_nodes= 56).fit(df_train_X, df_t
val_pred_dtc = dtc.predict(df_val_X)
print(accuracy_score(df_val_y, val_pred_dtc))

0.7688838784147749

print(recall_score(df_val_y, val_pred_dtc))

0.7394908229721728

Hyperparameters tuning
[Combination : Accuracy]
1. random_state = 47 : 70.98%
```

Now we apply decision tree model

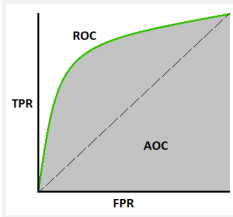
We got better results from Decision tree classifier than using Spectral , which did better than DBSCAN . We can use clustering to perform classification, but classification algorithms perform better than clustering algorithms when it comes to classifying data.

### 5.4 ROC-AUC for decision tree

In Machine Learning, performance measurement is an essential task. So when it comes to a classification problem, we can count on an AUC - ROC Curve. When we need to check or visualize the performance of the multi-class classification problem, we use the AUC (**Area Under The Curve**) ROC (**Receiver Operating Characteristics**) curve. It is one of the most important evaluation metrics for checking any classification model's performance. It is also written as AUROC (**Area Under the Receiver Operating Characteristics**)

AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.

The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.



AUC - ROC Curve [Image 2] (Image courtesy: My Photoshopped Collection)

Defining terms used in AUC and ROC Curve.

TPR (True Positive Rate) / Recall /Sensitivity

$$\text{TPR / Recall / Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Specificity

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Image 4

FPR

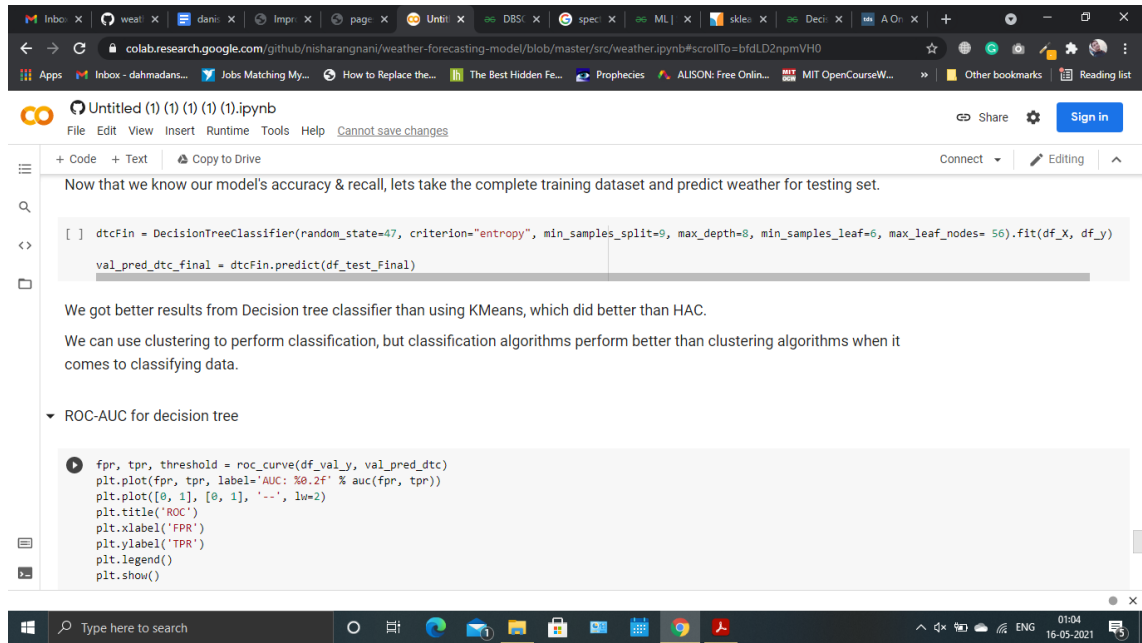
$$\begin{aligned} \text{FPR} &= 1 - \text{Specificity} \\ &= \frac{\text{FP}}{\text{TN} + \text{FP}} \end{aligned}$$

Image 5

How to speculate about the performance of the model?

An excellent model has AUC near to the 1 which means it has a good measure of separability. A poor model has AUC near to the 0 which means it has the worst measure of separability. In fact, it means it is reciprocating the result. It is predicting 0s as 1s and 1s as 0s. And when AUC is 0.5, it means the model has no class separation capacity whatsoever.

When AUC is approximately 0, the model is actually reciprocating the classes. It means the model is predicting a negative class as a positive class and vice versa.



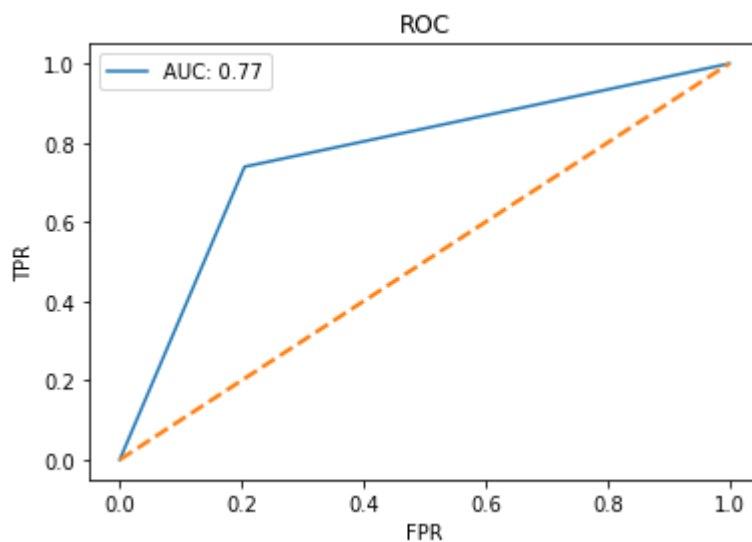
```
[ ] dtcFin = DecisionTreeClassifier(random_state=47, criterion="entropy", min_samples_split=9, max_depth=8, min_samples_leaf=6, max_leaf_nodes= 56).fit(df_X, df_y)
val_pred_dtc_final = dtcFin.predict(df_test_Final)
```

We got better results from Decision tree classifier than using KMeans, which did better than HAC.

We can use clustering to perform classification, but classification algorithms perform better than clustering algorithms when it comes to classifying data.

▼ ROC-AUC for decision tree

```
fpr, tpr, threshold = roc_curve(df_val_y, val_pred_dtc)
plt.plot(fpr, tpr, label='AUC: %0.2f' % auc(fpr, tpr))
plt.plot([0, 1], [0, 1], '--', lw=2)
plt.title('ROC')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.legend()
plt.show()
```



Area under curve is 77%  
Our task was 77% effective.

### 5.5.1 System used

- O.S: Windows 10
- O.S type: 64-bit



- RAM: 6GB
- Processor: 2.2 GHz Dual-Core Intel Core i7

### 5.5.2 Development tools

The development of the program includes different tools in the process of development and deployment. These tools are as follows:

- Google colab
- Anaconda IDE

Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

**Colab** is basically a free Jupyter notebook environment running wholly in the cloud. Most importantly, **Colab** does not require a setup, plus the notebooks that you will create can be simultaneously edited by your team members – in a similar manner you edit documents in Google Docs.

### 5.3 Conclusion

This chapter provided detailed idea for the implementation which may help in finding the performance of two algorithms. This chapter also includes the step involved in developing and deploying.

## CHAPTER 6

### Future Works

#### 6.1

Clustering is an **unsupervised learning** technique where we try to group the data points based on specific characteristics. There are various clustering algorithms with **K-Means** and **Hierarchical** being the most used ones. Some of the use cases of clustering algorithms include:

- Document Clustering
- Recommendation Engine
- Image Segmentation
- Market Segmentation
- Search Result Grouping
- and Anomaly Detection.

All these problems use the concept of clustering to reach their end goal. Therefore, it is crucial to understand the concept of clustering. But here's the issue with these two clustering algorithms.

K-Means and Hierarchical Clustering both fail in creating clusters of arbitrary shapes. They are not able to form clusters based on varying densities. That's why we need DBSCAN clustering. Thus for future purpose in large databases DBSCAN can be used for its unique feature

Spectral clustering methods have various real-world applications, such as face recognition, community detection, protein sequences clustering etc. Although spectral clustering methods can detect arbitrary shaped clusters, resulting thus in high clustering accuracy, the heavy computational cost limits their scalability.

## References

- 1.[1] Improving Multimodel Weather Forecast of Monsoon Rain Over China Using FSU Superensemble T. N. KRISHNAMURTI\*, A. D. SAGADEVAN, A. CHAKRABORTY†, A. K. MISHRA, and A. SIMON Department of Meteorology, Florida State University Tallahassee, FL 32306
- 2.[2] Convex-hull & DBSCAN clustering to predict future weather Ratul Dey Sanjay Chakraborty Computer Science & Engineering Computer Science & Engineering Institute of Engineering & Management Institute of Engineering & Management ratul170292@gmail.com schakraborty770@gmail.com
- 3.[3] Kedia, V. Thummala, K. Karlapalem, “Time Series Forecasting through Clustering A Case Study,” COMAD, 2005. pp.183-191.
- 4.[4] Chakraborty and N. K. Nagwani, “Analysis and Study of Incremental K-Means Clustering Algorithm,” in proc. of International conference on high performance architecture and grid computing (HPAGC) sponsored by Springer Germany, Punjab (India), 2011.
- 5.[5]<https://towardsdatascience.com/unsupervised-machine-learning-spectral-clustering-algorithm-implemented-from-scratch-in-python-205c87271045>
- 6.[6]<https://www.geeksforgeeks.org/ml-spectral-clustering/>
- 7.[7] <https://core.ac.uk/download/pdf/234677189.pdf>
- 8.[8]<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html>
- 9.[9]<https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80>
- 10.[10]<https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/>
- 11.[11]<https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf>
12. [12] <https://ai.stanford.edu/~ang/papers/nips01-spectral.pdf> Spectral Clustering Andrew Ng, Michael Jordan, Weiss







