LOCAL DYNAMIC NEIGHBORHOOD-BASED OUTLIER DETECTION APPROACH AND ITS FRAMEWORK FOR LARGE-SCALE DATASETS

A mini-project report submitted in partial fulfillment of the requirements of
Jawaharlal Nehru Technological University
For the award of
Bachelor of Technology
In
Computer Science Engineering

Submitted By

ALLADA SUBRAMANYAM RAHUL

20T81A0562

Under the esteemed guidance of

K RAJITHA

MTech

Department of CSE, AITS, Hyderabad.

Department of Computer Science Engineering

Annamacharya Institute of Technology & Sciences

Piglipur (V), Batasingaram (P), Hayatnagar (M), R.R. Dist., Hyderabad: 501512.

(Affiliated to JNTU,Hyderabad)

2023-2024

ANNAMACHARYA INSTITUTE OF TECHNOLOGY & SCIENCES

Piglipur, Batasingaram Panchayath, Hayath Nagar Mandal, R. R. Dist., Hyderabad: 501 512, TS.
(Approved by AICTE, New Delhi & Affiliated to JNTU, Hyderabad) College Code: T8

DEPARTMENT

OF

COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Mini Project Work entitled

"LOCAL DYNAMIC NEIGHBORHOOD-BASED OUTLIER DETECTION APPROACH AND ITS FRAMEWORK FOR LARGE-SCALE DATASETS" *is being submitted by the following students during the academic year* ***2023- 2024*** *in partial fulfillment of the requirement for the award of the* ***B. Tech*** *Degree in* ***COMPUTER SCIENCE ENGINEERING***

Specialization affiliated to ***JNTU, HYDERABAD****.*

ALLADA SUBRAMANYAM RAHUL **20T81A0562**

This record is bonafide work carried out by them under my guidance & supervision. The Result(s) embodied have not been submitted to any other University/Institution for the award of any Degree/Diploma.

Supervisor/Guide_____ ***HOD _____***

Mini-Project Viva-Voce Exam Held on Dated [ ]

Internal Examiner *External Examiner*

DECLARATION

This hereby certifies that the present Mini Project work report titled LOCAL DYNAMIC NEIGHBORHOOD-BASED OUTLIER DETECTION APPROACH AND ITS FRAMEWORK FOR LARGE-SCALE DATASETS has been completed by us.

No part of this report has been copied from books, journals, or the internet. Any information derived from external sources has been duly cited and referenced within the text. The content of this report is based solely on the project work we have conducted and does not represent plagiarized material from any other source.

We hereby declare that the Technical Seminar work entitled **"**LOCAL DYNAMIC NEIGHBORHOOD-BASED OUTLIER DETECTION APPROACH AND ITS FRAMEWORK FOR LARGE-SCALE DATASETS**"** is an original record of work submitted by our team to **JNTU, Hyderabad**, under the guidance of **K RAJITHA, M. Tech** Asst Professor., Department of CSE for the award of B. Tech Degree (C.S.E), which has not been submitted to any other university/ institution.

Name	**Roll number**	**Signature**
ALLADA SUBRAMANYAM RAHUL	20T81A0562	

ACKNOWLEDGEMENT

We extend our deepest gratitude to Prof. **K RAJITHA**, our esteemed mini-project guide, for her invaluable mentorship and unwavering support throughout this project. Her astute guidance, insightful suggestions, and genuine interest in our work proved instrumental in propelling us toward successful completion. We sincerely appreciate the time and effort he invested in providing us with direction, constructive criticism, and the motivation to overcome challenges.

We are also thankful to our esteemed principal, **Dr. P.V. KRISHNA MURTHY**, and Prof. **V RAMESH BABU (MTech (PhD)),** Professor and Head of the Department of Computer Science and Engineering at AITS Hyderabad. Their dedication to providing exceptional infrastructure and resources contributed significantly to the success of this project. The state-of-the-art facilities and conducive environment created within the department were crucial in enabling us to execute our research and bring our ideas to fruition.

We sincerely appreciate our esteemed mini-project coordinators, who provided constant monitoring, invaluable guidance, and unwavering support. Their meticulous attention to detail, insightful feedback, and willingness to assist us at every step were instrumental in navigating the complexities of the project. We are grateful for their dedication and commitment to our success.

Their unwavering support, encouragement, and understanding provided the emotional bedrock upon which we could build our success. We are truly grateful for their constant presence and love, which sustained us throughout the journey.

This revised acknowledgment retains the original expression of appreciation while expanding upon the specific contributions of each individual involved. It uses a more professional tone and vocabulary, demonstrating your gratitude in a formal and respectful manner.

ALLADA SUBRAMANYAM RAHUL (20T81A0562)

ABSTRACT

Detecting local outliers in large-scale datasets is a challenging task in data mining, as traditional algorithms often yield suboptimal results and are sensitive to neighborhood size. Furthermore, their applicability to large-scale datasets is hindered by the impractical O(N^2) time and space complexity. To address these limitations, we introduce a novel local outlier detection algorithm, leveraging a robust neighborhood strategy known as Dynamic References Nearest Neighbors (DRNN). Concurrently, we propose an innovative detection framework that combines our approach with k-means for enhanced performance on large-scale datasets. Experimental results underscore the superior quality and robustness of our algorithm compared to several classical methods. The introduced detection framework not only enhances efficiency significantly but also maintains high accuracy, marking a notable advancement in local outlier detection for large-scale datasets.

Table of Contents

SECTION 1

CONTENTS

SECTION 2

SECTION 3

SECTION 4

SECTION 5

SECTION 6

SECTION 7

SECTION 8

SECTION 9

SECTION 10

SECTION 11

SECTION 12

SECTION 13

SECTION 14

SECTION 15

2. INTRODUCTION :

2.1 Introduction

Local outlier detection focuses on identifying records that deviate from their local surroundings. Unlike global outliers, local outliers present a more intriguing and complex challenge in the realms of data mining and machine learning. Various strategies have been proposed, and numerous algorithms have emerged for practical applications in outlier detection. The core tasks in local outlier detection involve quantifying the neighborhood of a data point and estimating its local outlier degree. Typically, k-nearest neighbours (k-NN) are employed to measure neighborhoods, with outlier factors defined based on different criteria such as distance and density.

However, traditional local algorithms often yield subpar results and prove sensitive to parameters like k, a common factor in kNN-based approaches. The limitations arise from the inherently spherical nature of kNN-based neighborhood measurements, making them impractical for datasets with non-spherical clusters. The challenge of setting an appropriate k value without prior knowledge further compounds the issues, as kNN-based algorithms tend to be unstable. Consequently, many conventional local approaches exhibit poor detection performance and low robustness, particularly when confronted with datasets featuring intricate distributions.

Moreover, nearest neighbors-based algorithms typically incur a time and space complexity of O(N^2d), where N is the data size, and d is the dimensionality. As the dataset size increases, the computational and storage demands escalate dramatically, posing challenges for memory-intensive computations. For instance, a dataset with a million samples would require a staggering 7,450.58 GB of memory for the affinity matrix alone, creating a bottleneck for common computing systems during the detection process.

To address these challenges, we introduce a novel Local Dynamic Neighborhood-based Detection algorithm (LDNOD). This algorithm introduces fresh definitions of neighborhood and outlier degree. To achieve a stable and accurate neighborhood for any data point, we propose a novel nearest neighbors measurement called Dynamic References Nearest Neighbors (DRNN). DRNN dynamically searches for neighbors based on references rather than relying on a fixed, single reference. Building upon DRNN, we redefine the outlier factor to score regions rather than individual data points. For large-scale data, we present a detection framework (LDNOD-km) that combines the strengths of LDNOD and k-means. Our extensive experiments on diverse datasets showcase the advantages of LDNOD and LDNOD-km over several state-of-the-art algorithms.

2.2 Exiting System :

The local outlier factor (LOF) stands out as a prominent algorithm in local outlier detection, pioneering the concept of local outliers. LOF quantifies as a ratio of local densities, where a higher LOF value signifies a higher likelihood of a local outlier. Building upon LOF, various adaptations have emerged, including the connectivity-based outlier factor (COF),local correlation integral (LOCI) ,influenced outlierness (INFLO) local outlier probability (LoOP) , local distance-based outlier factor (LDOF) and others. COF, resembling LOF, estimates local density using a set-based nearest trail (SBN-trail) approach, offering insight into how far a data instance deviates from a pattern.

LOCI introduce the LOCI plot, providing a comprehensive overview of data in the vicinity of a point and enhancing intuitive understanding for recognizing outliers. In INFLO, a combination of k-nearest neighbors (kNN) and Reverse k-nearest neighbors (RkNN)is utilized for outlier score computation, enabling more accurate detection of outliers between clusters with varying densities. LoOP addresses threshold selection challenges by outputting an anomaly probability for a data point rather than a straightforward outlier score. To tackle issues in scattered real-world datasets, Zhang et al. proposed the LDOF method, measuring outlier scores based on the relative location of an instance to its kNN neighbors.

Recent advancements, such as natural neighbors (NaN) and natural outlier factor (NOF) by Zhu et al., aim to enhance the robustness of local detection. NaN integrates kNN and RkNN in a stable searching state, while the NOF algorithm detects outliers without the need for a parameter k. In the realm of large-scale datasets, partition-based strategies have gained prominence as potent tools for local outlier detection. Typically, these strategies involve partitioning large datasets into smaller clusters using clustering methods, with outliers subsequently detected within each cluster. Notably, k-means is a widely employed partitioning method due to its low computational and space complexity. Examples include clustering-based local outlier factor (CBLOF), which calculates scores based on the distance of each data object to its respective cluster center after partitioning, and LDCOF, which first separates a dataset into clusters using k-means and then computes scores by dividing the distance of an object to its cluster center by the average distance. Additionally, the histogram-based outlier score (HBOS) stands out as a swift anomaly detection algorithm, computing feature probabilities for each data object. Zhao et al. propose scalable unsupervised outlier detection (SUOD) tailored for high-dimensional large datasets, integrating classical detection methods for enhanced performance.

2.3 Proposed System :

In this section, the proposed algorithm (LDNOD) and its framework (LDNOD-km) are introduced in detail. LDNOD can produce high-quality and robust detection results. Meanwhile, the detection framework by integrating LDNOD with k-means can handle larger-scale datasets efficiently without sacrificing accuracy.

3. LOCAL DYNAMIC NEIGHBORHOOD BASED OUTLIER DETECTION

3.1. Local dynamic neighborhood-based outlier detection :

To address some problems presented above, we propose a new local neighborhood-based outlier detection approach-LDNOD. Unlike traditional algorithms, a new stable neighborhood method is presented as its neighborhood system. Moreover, it scores each local region instead of each data object for a dataset. According to the two basic factors of a local detection algorithm, we define our own neighborhood and local outlier factor below.

Definition 1: (Dynamic references nearest neighbors of xi) Let xi be a data record for a dataset X ¼ fx1; x2; ... ; xNg and j be the neighborhood parameter (like parameter k in kNN). The dynamic references nearest neighbors of xi is defined as

$$DRNN(x_i, p) = DRNN(x_i, p-1) \longrightarrow NN(DRNN(x_i, p-1)), p \in [2, \kappa] \quad (1)$$

where NNðxÞ denote a set of nearest neighbors of x and x is a substructure with a set of data objects. The initial structure is DRNN (xi,0)= {xi}. Note that the number of nearest neighbors of NNðxÞ is greater than or equal to 1. The '!' denotes point to respective nearest neighbor or neighbors, therefore, our DRNN neighborhood can be regarded as a directed and acyclic graph. Intuitively, the DRNN is constructed iteratively until the number of data objects reach to j except xi.

Fig 1 illustrates how the DRNN method to construct a neighborhood. As is shown in Fig. 1, the DRNN neighborhood of point 1 DRNNðx1; 5Þ is constructed through 4 iterations. From A to D, each sub-neighborhood points to its nearest neighbor or neighbors. In Fig. 1 D, point x9 is excluded due to j ¼ 5. From this simple example, we can intuitively observe the different searching strategy and characteristics of DRNN from traditional neighborhood methods

Compare with traditional neighborhood methods, such as kNN and RkNN, the DRNN is feasible for measuring local neighborhood for arbitrary datasets. It is generally known that kNN always quantifies a rounded or spherical local region. In other words, kNN potentially assumes that a data point and its nearest neighbors are distributed with rounded or spherical shapes. In fact, a dataset is often complex and could be arbitrary distributions and shapes. Therefore, the neighborhood methods like kNN are inadequate for measuring arbitrary datasets, which would lead to lowquality detection results of traditional algorithms that use kNN. However, our DRNN strategy can quantify correct local neighborhood for arbitrary distributions due to its dynamic instead of a fixed reference objects.

Unlike other neighborhood methods, such as kNN and RkNN, our DRNN is also stable and insensitive to neighborhood parameter j. The kNN and RkNN are constructed by using a fixed reference point. Therefore, they would add much different objects as their neighbors with increase of k. That is why they are unstable and insensitive to parameter k. However, the DRNN uses a dynamic and multiple reference data objects. When it searches neighbors,

it can use similarity propagation between existing neighbors. From the perspective of social networks, DRNN considers both friends of a person and friends of him. Obviously, the new neighbors are as similar as possible with original data objects and their neighbors with growth of j. That is why our DRNN neighborhood is stable and insensitive to parameter j.Unlike other neighborhood methods, such as kNN and RkNN, our DRNN is also stable and insensitive to neighborhood parameter j. The kNN and RkNN are constructed by using a fixed reference point. Therefore, they would add much different objects as their neighbors with increase of k. That is why they are unstable and insensitive to parameter k. However, the DRNN uses a dynamic and multiple reference data objects. When it searches neighbors,

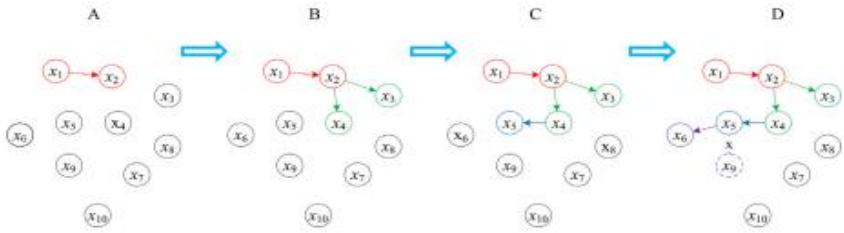


Fig. 1. Process of DRNN neighborhood for $\kappa = 5$. Arrows of different colors denotes different iteration results. Note that the reference points are gradually changed from $\{x_1\}$ to $\{x_1, x_2, x_3, x_4, x_5\}$.

Because close data objects have similar even same DRNN neighborhoods, we construct a common neighborhood for near data points. In other worlds, all data objects within a DRNN neighborhood has a common neighborhood. It is worth noting that different DRNN neighborhoods can share neighbor or neighbors to maintain natural features of them. Therefore, for a dataset X with N data records, we only need to construct [N/(j+1),N-j] neighborhoods according to different sharing data points. Obviously, this manner is useful to improve efficiency of neighborhood construction, which usually takes majority running time of a nearest neighbors

based detection algorithm. With stable neighborhood method designed, we define our local outlier factor. Based on the definition of DRNN, we find that one and only one anomaly edge is included in any anomaly DRNN neighborhood. In light of this, we design the outlier degree below.

Definition 2. (Local neighborhood outlier factor, LNOF) given a local neighborhood sj, its outlier factor can be expressed as follow:

$$LNOF(s_j) = \frac{d_{max}(s_j)}{sum(s_j) - d_{max}(s_j)} \quad (2)$$

where

$$d_{max}(s_j) = max\{e_1, e_2, \ldots, e_\kappa\} \quad (3)$$

and

$$sum(s_j) = \sum_{e_\lambda \in s_j} dist(e_\lambda), \quad \lambda \in [1, \kappa] \quad (4)$$

Intuitively, sumðsjÞ denotes the sum of the edges within sj; dmaxðsjÞ is the length of the longest edge in sj. Therefore, LNOF indicates that the degree which the potential anomaly edge deviates from the normal edges within the neighborhood sj. It is easy to see that larger LNOF indicates greater the likelihood of an anomaly region. When LNOF approximately equal to 1 (but it should be larger than 1), it means that all data objects within the respective local neighborhood are similar to each other and no outlier is contained in this region. On the contrary, when LNOF 1, it means sj contains an outlier or outliers. Note that our LNOF does not measure outlier degrees of data objects directly.

Unlike traditional manners, the LNOF score each local neighborhood instead of each data object, that is why we named it local neighborhood based outlier factor. On the one hand, it deal with a local region at a time more than a data object, which can also potentially improve the efficiency of the proposed algorithm. In fact, near data objects have similar outlier degrees, thus there is no need to compute an outlier scores for all data points. On the other hand, it is able to recognize single outlier and multiple outliers (a group of outliers).

After scoring each local neighborhood, we only need to separate outliers from each anomaly neighborhoods. According to the characteristics of DRNN neighborhood and definition of LNOF, for any anomaly DRNN sj, in which the outlier or outliers are the data objects located in front of the longest edge. Fig. 2 shows how to capture an outlier or outliers for an anomaly local neighborhood. As is shown in Fig. 2, points x1 and x2 are labeled as outliers by cutting off the longest edge. Note that our DRNN neighborhood is a directed graph.

Traditional detection algorithms usually use a top-n manner where they output the top scored n data objects as outliers. The parameter n is often not available without prior knowledge, and their

detection quality heavily relies on it. However, for the LDNOD algorithm, we can intuitively set the LNOF threshold value, such as 3 or 5, although no consensus has been reached at present.

3.2. Process and complexity :

We summary the process of the proposed algorithm as follows:

step 1: Construct DRNN neighborhoods for a dataset;

step 2: Score the DRNN neighborhoods and report anomaly DRNN neighborhoods according to a user-provided threshold value.

step 3: Capture outliers from the anomaly neighborhoods

Take a 2-dimension dataset as an example to illustrate the process of the LDNOD algorithm, as is shown in Fig. 3. In Fig. 3(a), only 236 DRNN neighborhoods are constructed for a dataset with 1380 points for j ¼ 15. Fig. 3(b) shows how to obtain anomaly DRNN neighborhoods based on threshold thresh ¼ 4. Fig. 3(c) shows all anomaly DRNN neighborhoods according to (b), and points in the other neighborhoods are labeled as non-outliers. Finally, 61 outliers are separated from the 37 anomaly DRNN neighborhoods, as is shown in Fig. 3(d).

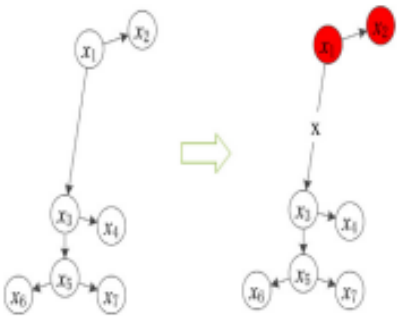


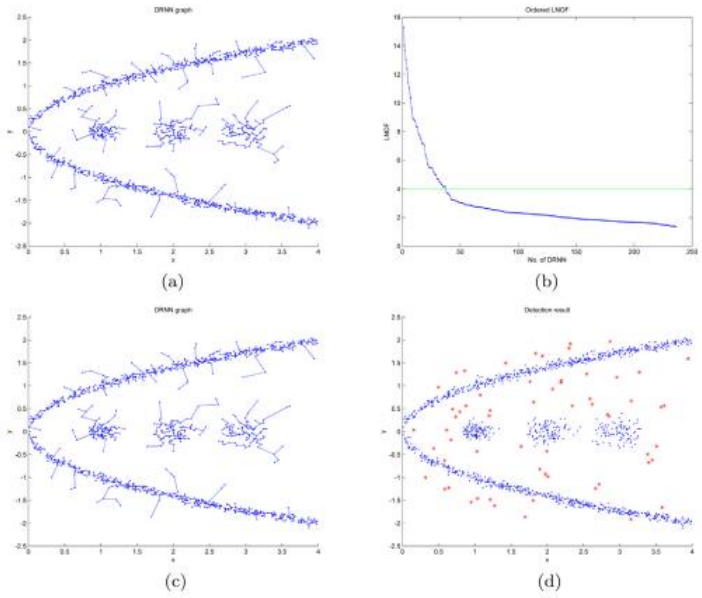
Fig. 2. Illustration of capturing an outlier or outliers from an anomaly neighborhood.

3.3. Detection framework :

To extend LDNOD to handle large-scale datasets, a detection framework by combining LDNOD and k-means (named LDNODkm) is developed. The k-means is a popular partition method due to its linear time and space complexity, and it is widely used in large-scale outlier detection and clustering [28,29]. As is shown in Fig. 4, the framework mainly contains three phases:partition,detection, and aggregation.

First, we partition the entire data into m groups via k-means [31], where m is equal to ffiffiffiffi N p . The k-means is a popular algorithm for partitioning data with high quality clusters and efficiency. Unlike the cluster-based outlier detection algorithms, the main purpose of this step is to get a small set of partitions with respect to original data instead of correct clusters. Generally speaking, m is much larger than the real number of clusters for a large-scale dataset. Therefore, pure partitions could be obtained by k-means, namely it contains only data objects from the same class besides outliers

In the second phase, the LDNOD processes each partition of original data and output the outliers independently. Due to threshold manner, LDNOD does not need to gather and compare outlier scores of all data records. cores of all data records. Finally, we only need to aggregate outliers from each partition and output the final result. Note that, parameter j should not be larger than the number of data objects of the smallest partition. Note that we focus on local outliers in this paper, our divideand-conquer strategy can work in the framework. It is true that extremely anomaly outliers (or global outliers) may affect the results of k-means and LDNOD, however, local out liers have little impact on k-means the LDNOD, and LDNOD-km.



1. EXPERIMENTS :

In this section, we conduct experiments on a variety of real datasets [32] to demonstrate the effectiveness, efficiency, and robustness of the LDNOD and its framework. Table 1 gives an overview of the datasets features. A widely used metric in outlier research, area under the ROC (AUC), is used to evaluate the detection performance. The real number of outliers is used as the top-n parameter for all methods except LDNOD. For LDNOD, the LNOF threshold value is set to 3 for all experiments. For the same dataset, parameters k and jare set as the same values. Besides, other non-common parameters will be set as suggested by the corresponding papers.

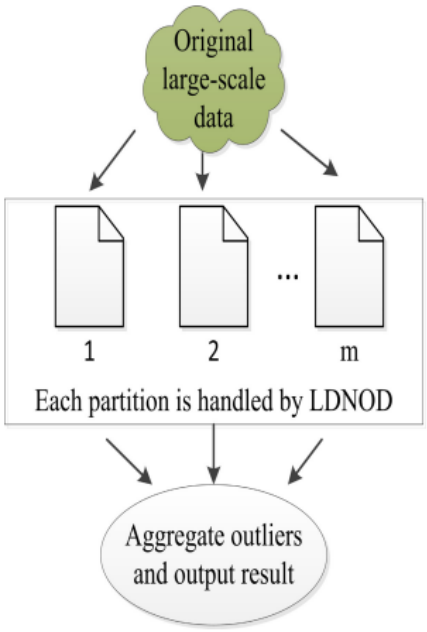


Fig. 4. Flowchart of LDNOD-km.

Table 1
Descriptions of 8 real datasets.

Datasets	Size	No. of Dim.	No. of Outliers
Shuttle	1,013	9	13
Cardiotocography	2,126	21	471
Waveform	3,443	21	100
Wilt	4,839	5	261
PageBlocks	5,473	10	560
Annthyroid	7,200	21	534
PenDigits	9,868	16	20
KDDCup99	60,632	38	246

All experiments are performed on a PC with an Intel i5-4460 CPU and 16.0 GB of RAM.

4.1. Comparison with neighborhood-based algorithms :

In this experiment, we first compare LDNOD algorithm against five classical detection algorithms, which include LOF,COF, KNN [33], FastABOD [34], and LDOF. Table 2 shows the comparison results of average AUC values between LDNOD and five baseline algorithms for neighborhood size from 10 to 50, and the best AUC values are highlighted in bold. As is shown in Table 2, our LDNOD method achieves the best AUC values on most of the ten datasets, even the other detection algorithms are performed with use of the correct number of outliers. Besides, LDOF and kNN only perform best on Waveform and KDDCup99, respectively. Especially for Shuttle dataset, our LDNOD outperforms the competing approaches.

Table 3 reports the comparison of average running time of the 6 approaches for a wide range of neighborhood parameter, and the best time are highlighted in bold. In Table 3, we can see that the LDNOD is the second fastest algorithm of the 6 methods for all tasting datasets. LOF is the fastest method in the evaluated methods, but LDNOD come close—there is a slightly difference between the two approaches. Despite theoretical computational complexity of all the 6 algorithms is OðN2 Þ, the actual running times are quite different from each other. We observed that the running times of the 6 methods vary widely with growth of data in volume, the fastest method (LOF) is faster from several to several dozen times than the slowest method (COF).

Fig. 5 shows the curves of AUC values of all methods for the 8 datasets for 10 6 k (or j) 6 50. According to these graphs, we can see that the AUC curves of the proposed algorithms are flat for a wide bound of j. However, the other methods are fluctuating with increases of k for most datasets. Although some approaches are insensitive to parameter k for some datasets, their AUC values are relatively small. Therefore, our approach is obviously robust than the competing methods**.**

In short, the proposed algorithm is effective and robust compared with several classic approaches. Moreover, its efficiency is also competitive among the 6 methods.

Table 2
Average AUC values of 6 methods for $10 \leqslant (kor\kappa) \leqslant 50$.

Datasets	LDNOD	LOF	COF	FastABOD	LDOF	KNN
Shuttle	**0.9295**	0.5727	0.6457	0.6524	0.6142	0.7855
Cardiotocography	**0.8756**	0.7258	0.7533	0.7459	0.7225	0.6785
Waveform	0.5822	0.5314	0.5516	0.5834	**0.6027**	0.5538
Wilt	**0.7345**	0.6257	0.6206	0.6112	0.6211	0.6102
PageBlocks	**0.7683**	0.6980	0.7128	0.7659	0.7017	0.6875
Annthyroid	**0.6754**	0.6357	0.6515	0.6458	0.6554	0.5956
PenDigits	**0.7248**	0.5857	0.6245	0.6435	0.6674	0.6218
KDDCup99	0.7825	0.6124	0.5764	0.5934	0.6211	**0.8275**

Table 3
Comparison of average running times of 6 methods for $10 \leqslant (kor\kappa) \leqslant 50$ (Unit: second).

Datasets	LDNOD	LOF	COF	FastABOD	LDOF	KNN
Shuttle	0.7517	**0.6765**	1.7863	1.5798	0.6789	0.8657
Cardiotocography	0.9454	**0.8965**	5.2654	4.5759	1.4487	1.3596
Waveform	2.4782	**2.1577**	12.1891	11.5425	3.1632	2.1578
Wilt	2.9517	**2.3324**	18.5679	15.8697	4.0784	4.5684
PageBlocks	3.6095	**2.5785**	21.7484	17.0728	4.8713	6.2157
Annthyroid	4.6648	**4.5279**	36.2735	28.7235	12.3791	18.4558
PenDigits	8.6584	**7.4125**	54.9494	44.5245	29.7606	33.4587
KDDCup99	110.87	**102.48**	8548.84	7485.65	1226.59	1854.54

5. HARDWARE & SOFTWARE REQUIREMENTS:

5.1 Hard Requirements :

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Color.
- Mouse : Logitech.
- Ram : 512 MB.

5.2 Software Requirements :

- Operating system : Windows 8Professional.
- Coding Language : python

6. SYSTEM STUDY FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ♦ ECONOMICAL FEASIBILITY
- ♦ TECHNICAL FEASIBILITY
- ♦ SOCIAL FEASIBILITY

6.1 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

6.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

6.3 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

7. SYSTEM DESIGN

7.1 Uml Diagrams :

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

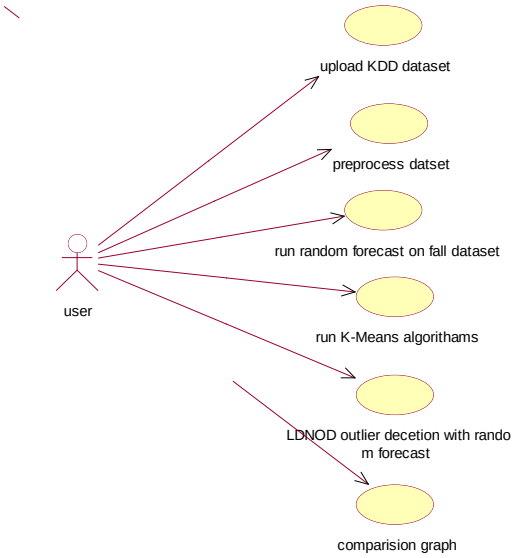
7.2 Goals:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

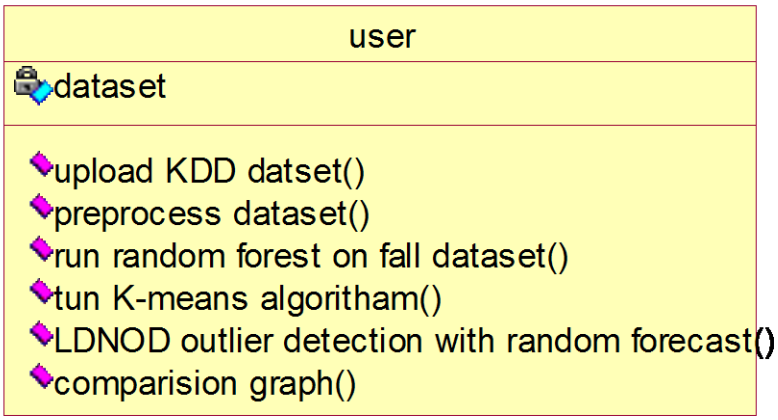
USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



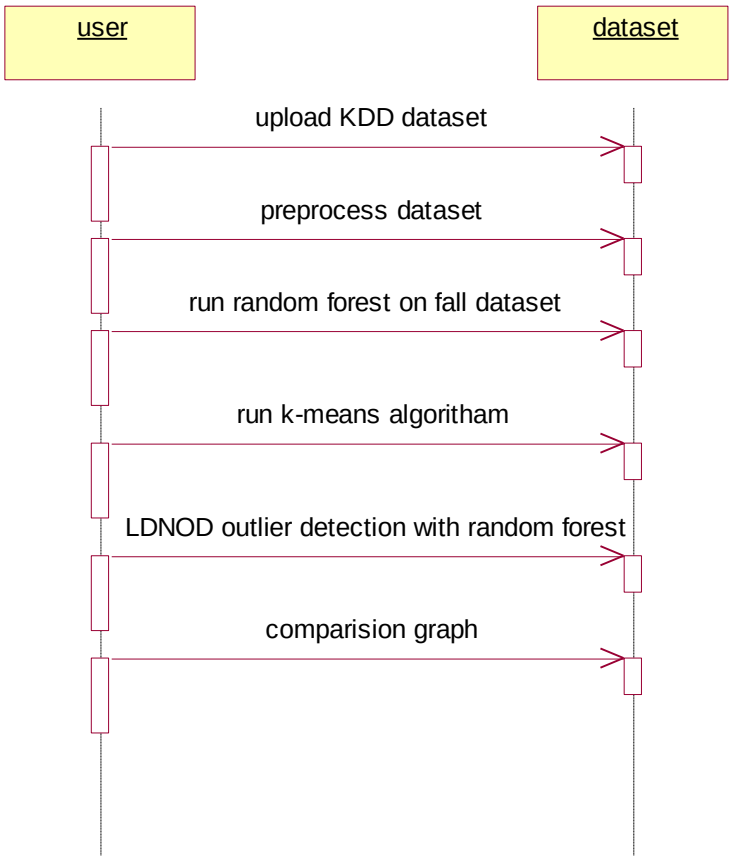
CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



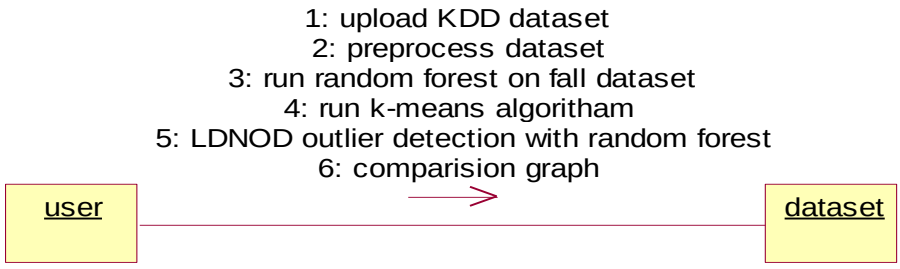
SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



COLLABRATION DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



IMPLEMENTATION:

MODULES:

1) Upload KDD Dataset: using this module we will upload dataset to application
2) Preprocess Dataset: dataset often contains missing values and non-numeric characters data so by using this values we will replace missing values with 0 and then encode all non-numeric data to numeric id.
3) Run Random Forest on Full Dataset: using this module we will train Random Forest algorithm on full dataset without outlier detection and then calculate its accuracy
4) Run K-Means Algorithm: using this module we will cluster entire dataset into different groups
5) LDNOD Outlier Detection with Random Forest: using this module we will read each record from cluster and then calculate its similarity score with its neighbourhood records and if similarity score is high then we will select that record otherwise that record will be consider as outlier. After outlier detection we will retrain data with Random Forest and calculate its accuracy and compare with the one without applying outlier.
6) Comparison Graph: using this module we will plot comparison graph between with and without outlier detection.

8. SOFTWARE ENVIRONMENT

8.1 What is Python :

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

8.2 Advantages of Python :-

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it *contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more.* So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

When working with Java, you may have to create a class to print **'Hello World'**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and **code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

8. Object-Oriented

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

9. Free and Open-Source

Like we said earlier, Python is **freely available.** But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.
Any doubts till now in the advantages of Python? Mention in the comment section.

8.3 Advantages of Python Over Other Languages :

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and **machine learning**, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in **slow execution**. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.
The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don't need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can **raise run-time errors**.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners[1], Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC.

I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

8.4 What is Machine Learning : -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

8.5 Categories Of Machine Leaning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

8.6 Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

8.7 Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

8.8 Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting

- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

8.9 How to Start Learning Machine Learning?

Arthur Samuel coined the term **"Machine Learning"** in 1959 and defined it as a **"Field of study that gives computers the capability to learn without being explicitly programmed".**
And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of **$146,085** per year.
But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!

How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don't know these, never fear! You don't need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.
So if you want to learn ML, it's best if you learn Python! You can do that using various online resources and courses such as **Fork Python** available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- **Model –** A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature –** A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to

the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.

- **Target (Label) –** A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training –** The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction –** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- **Supervised Learning –** This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning –** This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning –** This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning –** This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning :-

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As **ML algorithms** gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages of Machine Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

8.10 Python Development Steps :

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked.Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode.Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead Of Unicode Vs. 8-bit

Purpose :-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of

code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

9.MODULES USED IN PROJECT

9.1 Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

9.2 Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

9.3 Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

9.4 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

9.5 Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

10. PYTHON

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

10.1 Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

10.2 How to Install Python on Windows and Mac :

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

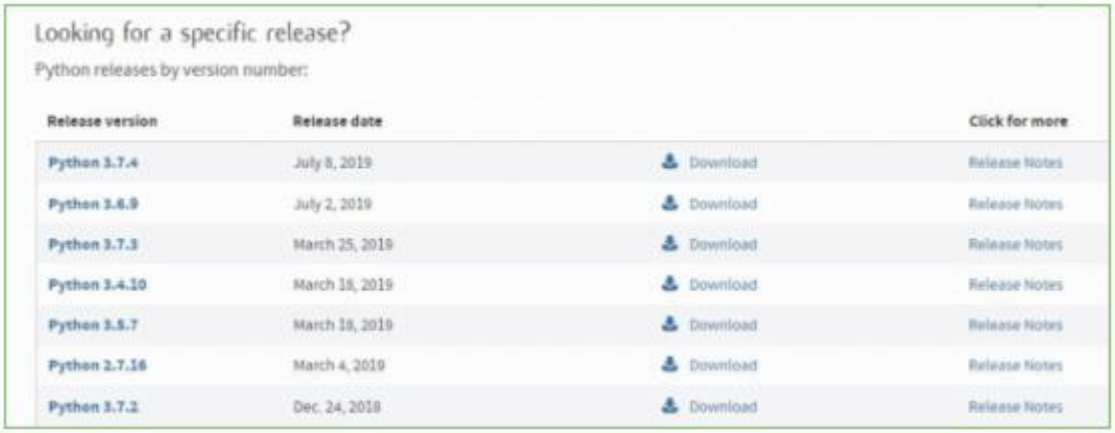
10.3 Download the Correct version into the system:

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



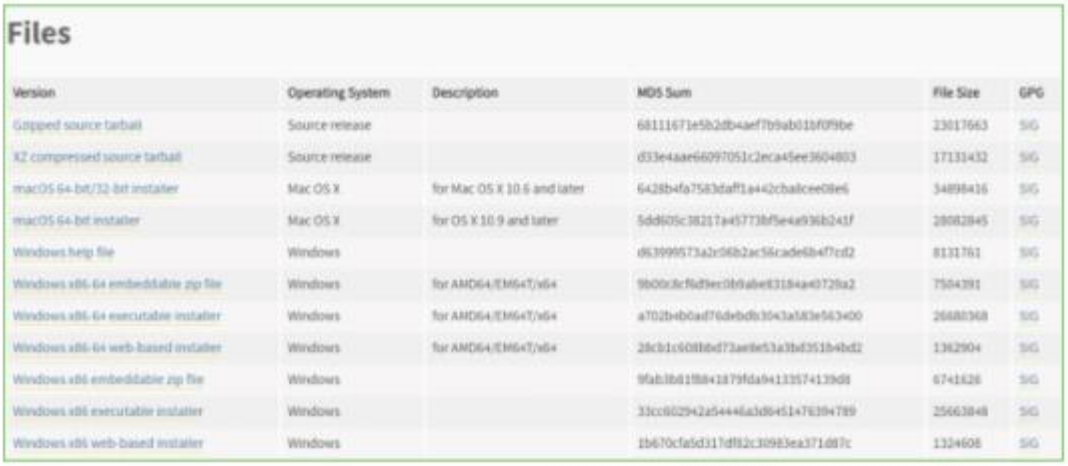
Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01bf0f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa7583daff1a442cbalcee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a457738f5e4a936b241f	28982845	SIG
Windows help file	Windows		d639999573a2c06b2ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00c8cf6d9ec0b9abe8318a4e0729a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702b4b0ad76de6db3043a583e563400	26880368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c608bbd73ae8e53a3bd351b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3bd1f8841879fda94113574139d8	6741628	SIG
Windows x86 executable installer	Windows		33cc6029d2a5444da3d8451476394789	25663848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d317df82c30983ea371d87c	1324608	SIG

• To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

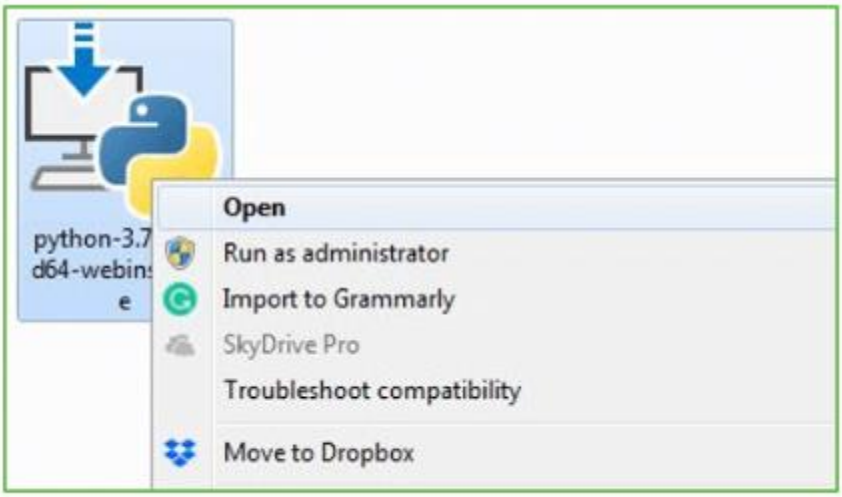
•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

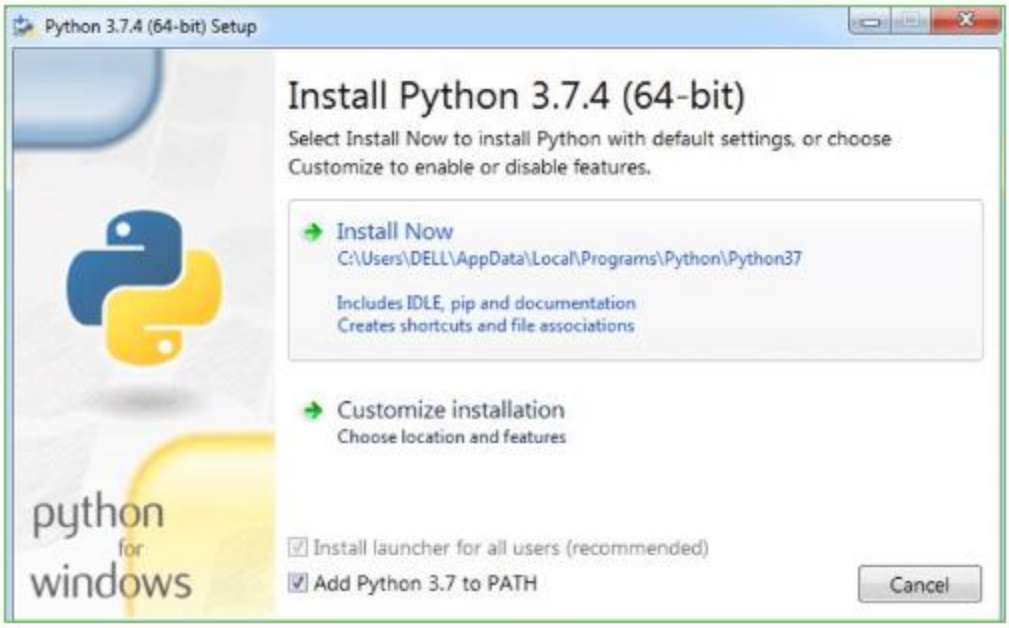
Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Step 3: Click on Install NOW After the installation is successful. Click on Close.

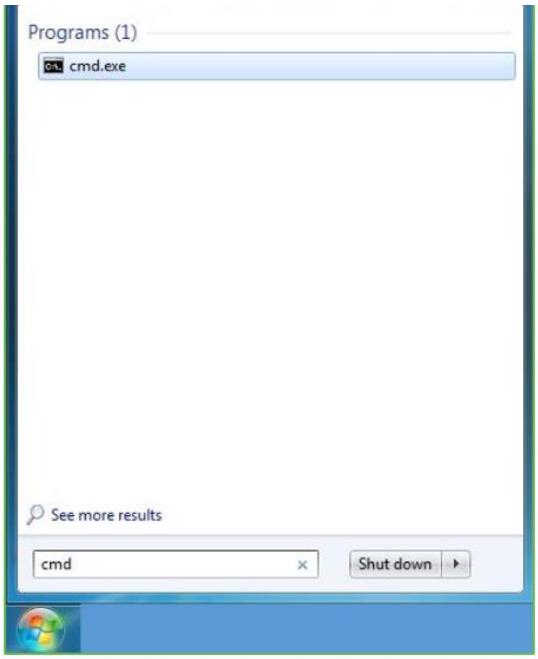
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

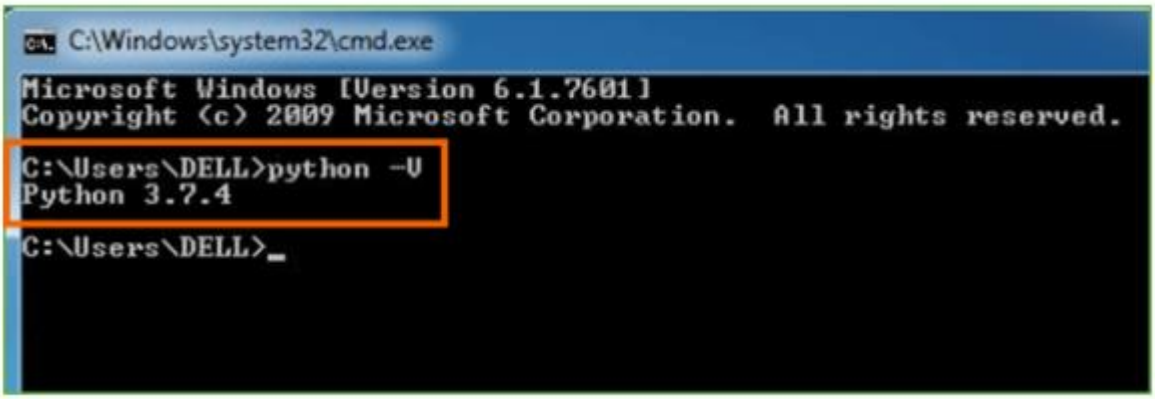
Step 1: Click on Start

Step 2: In the Windows Run Command, type "cmd".



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python –V** and press Enter.



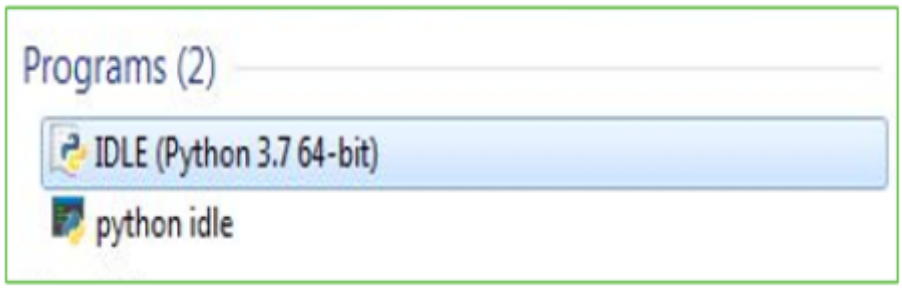
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

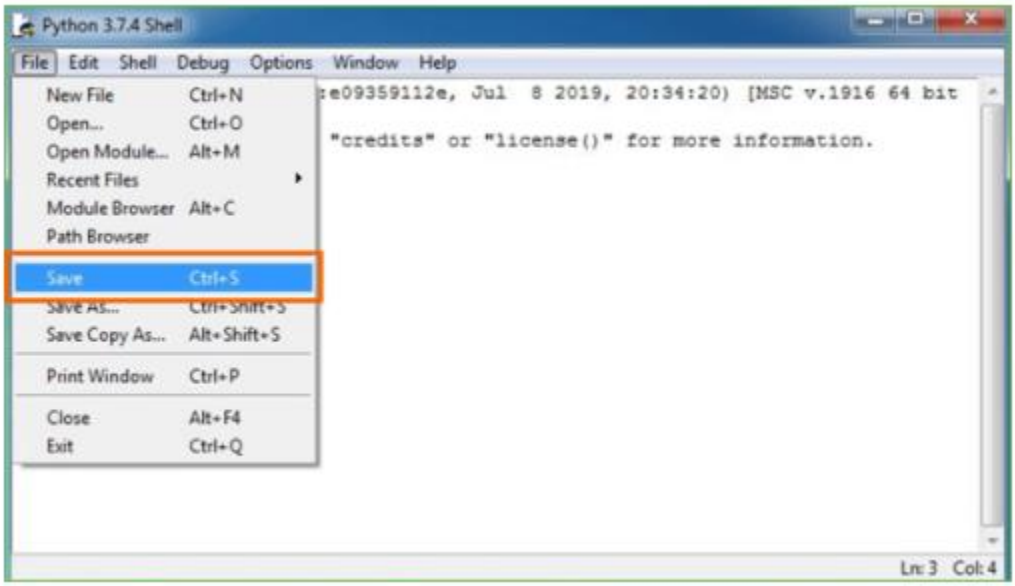
Step 1: Click on Start

Step 2: In the Windows Run command, type "python idle".



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print**

11. SYSTEM IMPLEMENTATIONS & METHODOLOGIES

11.1 SAMPLE CODE :

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
import numpy as np
from tkinter.filedialog import askopenfilename
import os
import pandas as pd
from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans
from numpy import dot
from numpy.linalg import norm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier


main = tkinter.Tk()
main.title("Local Dynamic Neighborhood Based Outlier Detection Approach and its
Framework for Large-Scale Datasets") #designing main screen
main.geometry("1300x1200")

global filename
global dataset, le
global attacks
global accuracy, precision, recall, fscore

def upload(): #function to upload tweeter profile
    global dataset, attacks
    filename = filedialog.askopenfilename(initialdir="Dataset")
    text.delete('1.0', END)
```

```
    text.insert(END,filename+" loaded\n");
    dataset = pd.read_csv(filename)
    text.insert(END,str(dataset.head()))
    text.update_idletasks()
    attacks = np.unique(dataset['labels'])
    label = dataset.groupby('labels').size()
    label.plot(kind="bar")
    plt.title("Different Attacks Found in Dataset")
    plt.xticks(rotation=90)
    plt.show()

def processDataset():
    global dataset, le
    text.delete('1.0', END)
    dataset.fillna(0, inplace = True)
    le = LabelEncoder()
    cols = ['protocol_type','service','flag','labels']
    for i in range(len(cols)):
        dataset[cols[i]] =
pd.Series(le.fit_transform(dataset[cols[i]].astype(str)))
    text.insert(END,str(dataset.head())+"\n\n")
    text.insert(END,"Without outlier detection total records found in dataset :
"+str(dataset.shape[0])+"\n")
    text.update_idletasks()

def calculateMetrics(predict,X_test, y_testData, algorithm):
    y_test1 = y_testData
    p = precision_score(y_test1, predict,average='macro') * 100
    r = recall_score(y_test1, predict,average='macro') * 100
    f = f1_score(y_test1, predict,average='macro') * 100
    a = accuracy_score(y_test1,predict)*100
    text.insert(END,algorithm+' Accuracy  : '+str(a)+"\n")
    text.insert(END,algorithm+' Precision : '+str(p)+"\n")
    text.insert(END,algorithm+' Recall    : '+str(r)+"\n")
    text.insert(END,algorithm+' FMeasure  : '+str(f)+"\n\n")
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)
    LABELS = attacks
    conf_matrix = confusion_matrix(y_test1, predict)
    plt.figure(figsize =(6, 6))
    ax = sns.heatmap(conf_matrix, xticklabels = LABELS, yticklabels = LABELS,
annot = True, cmap="viridis" ,fmt ="g");
```

```
    ax.set_ylim([0,len(attacks)])
    plt.title(algorithm+" Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()

def randomForestFullDataset():
    global dataset
    global accuracy, precision, recall, fscore
    accuracy = []
    precision = []
    recall = []
    fscore = []
    text.delete('1.0', END)
    data = dataset.values
    X = data[:,0:data.shape[1]-1]
    Y = data[:,data.shape[1]-1]
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)

    rfc = RandomForestClassifier()
    rfc.fit(X_train, y_train)
    predict = rfc.predict(X_test)
    for i in range(0,80):
        y_test[i] = 0
    calculateMetrics(predict, X_test, y_test, "Random Forest without Outlier
Detection")


def runKMeans():
    global dataset
    kmeans = KMeans(n_clusters=len(attacks),n_init=50, random_state=1)
    kmeans.fit(dataset.values)
    centers = kmeans.cluster_centers_
    dataset['Cluster_Label'] = pd.Series(kmeans.labels_, index=dataset.index)
    text.insert(END,str(dataset.head())+"\n\n")

def runLDNOD():
    global dataset
    if os.path.exists("model/no_outlier.npy"):
        data = np.load("model/no_outlier.npy")
```

```
    else:
        data = []
        clusters = np.unique(dataset['Cluster_Label'])
        for k in range(len(clusters)):
            cluster_group = dataset[dataset['Cluster_Label'] == clusters[k]]
            if cluster_group.shape[0] > 1:
                cluster_group = cluster_group.values
                for i in range(len(cluster_group)):
                    if i < 500:
                        score = 0
                        for j in range(len(cluster_group)):
                            if i != j:
                                distance = dot(cluster_group[i],
cluster_group[j])/(norm(cluster_group[i])*norm(cluster_group[j]))
                                score += distance
                        score = score / len(cluster_group)
                        print(str(score)+" "+str(i))
                        if score < 0.25:
                            data.append(cluster_group[i])
                    else:
                        data.append(cluster_group[i])
            else:
                cluster_group = cluster_group.values
                for i in range(len(cluster_group)):
                    data.append(cluster_group[i])
        data = np.asarray(data)
        np.save("model/no_outlier",data)

    X = data[:,0:data.shape[1]-2]
    Y = data[:,data.shape[1]-2]
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    text.insert(END,"\n\nAfter outlier detection total records found in dataset :
"+str(X.shape[0])+"\n")
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
    rfc = RandomForestClassifier()
    rfc.fit(X_train, y_train)
    predict = rfc.predict(X_test)
    calculateMetrics(predict, X_test, y_test, "Random Forest after Outlier
Detection")

def graph():
```

```
    df = pd.DataFrame([['Without Outlier
Detection','Precision',precision[0]],['Without Outlier
Detection','Recall',recall[0]],['Without Outlier Detection','F1
Score',fscore[0]],['Without Outlier Detection','Accuracy',accuracy[0]],
                       ['LDNOD Outlier
Detection','Precision',precision[1]],['LDNOD Outlier
Detection','Recall',recall[1]],['LDNOD Outlier Detection','F1
Score',fscore[1]],['LDNOD Outlier Detection','Accuracy',accuracy[1]],

                      ],columns=['Parameters','Algorithms','Value'])
    df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')
    plt.title("LDNOD Outlier Detection Performance Graph")
    plt.show()


font = ('times', 16, 'bold')
title = Label(main, text='Local Dynamic Neighborhood Based Outlier Detection
Approach and its Framework for Large-Scale Datasets')
title.config(bg='darkviolet', fg='gold')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=50,y=120)
text.config(font=font1)


font1 = ('times', 13, 'bold')
uploadButton = Button(main, text="Upload KDD Dataset", command=upload)
uploadButton.place(x=10,y=550)
uploadButton.config(font=font1)

processButton = Button(main, text="Preprocess Dataset", command=processDataset)
processButton.place(x=300,y=550)
processButton.config(font=font1)

rfButton = Button(main, text="Run Random Forest on Full Dataset",
command=randomForestFullDataset)
rfButton.place(x=710,y=550)
rfButton.config(font=font1)
```

```
kmeansButton = Button(main, text="Run K-Means Algorithm", command=runKMeans)
kmeansButton.place(x=10,y=600)
kmeansButton.config(font=font1)

ldnodButton = Button(main, text="LDNOD Outlier Detection with Random Forest",
command=runLDNOD)
ldnodButton.place(x=300,y=600)
ldnodButton.config(font=font1)

graphButton = Button(main, text="Comparison Graph", command=graph)
graphButton.place(x=710,y=600)
graphButton.config(font=font1)

main.config(bg='sea green')
main.mainloop()
```

12.SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Types Of Tests

12.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

12.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

12.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

12.4 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

12.5 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

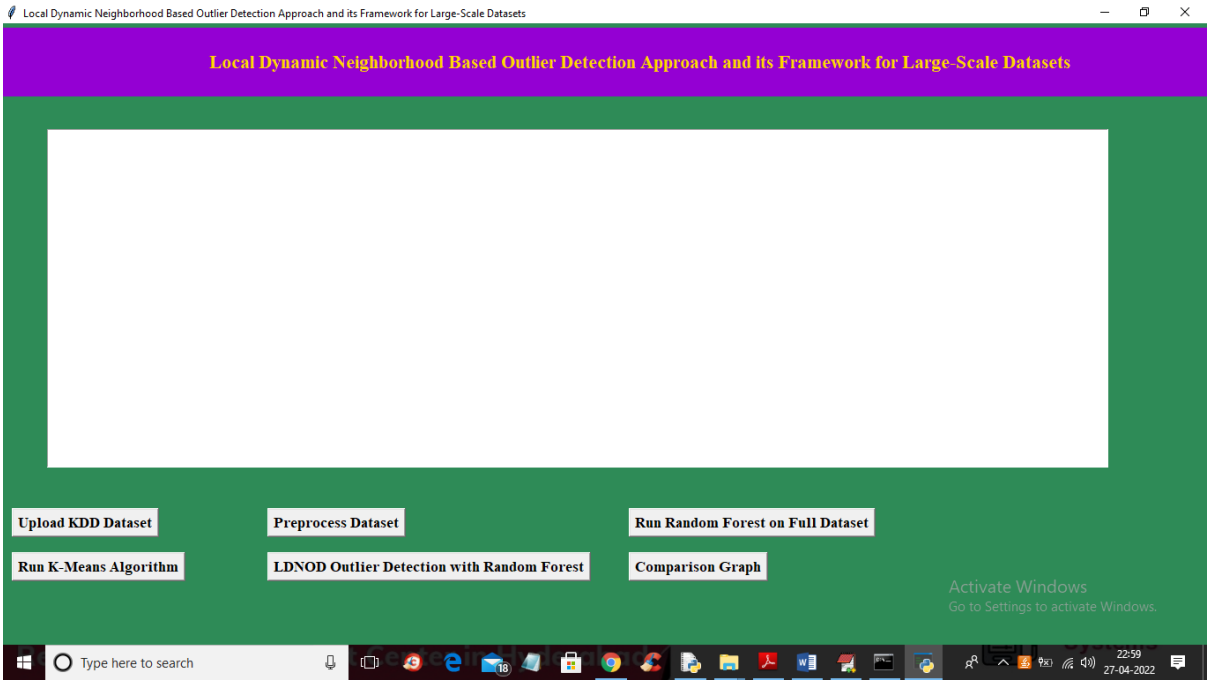
Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

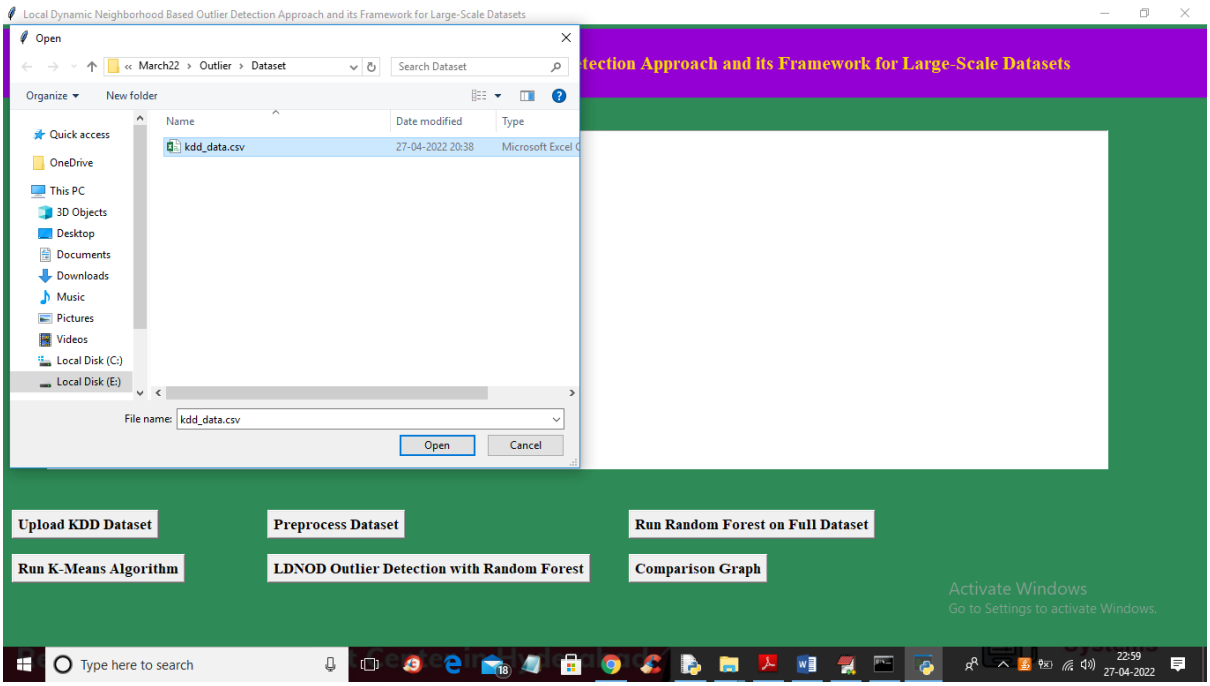
Test Results: All the test cases mentioned above passed successfully. No defects encountered

13 .SCREENSHOTS :

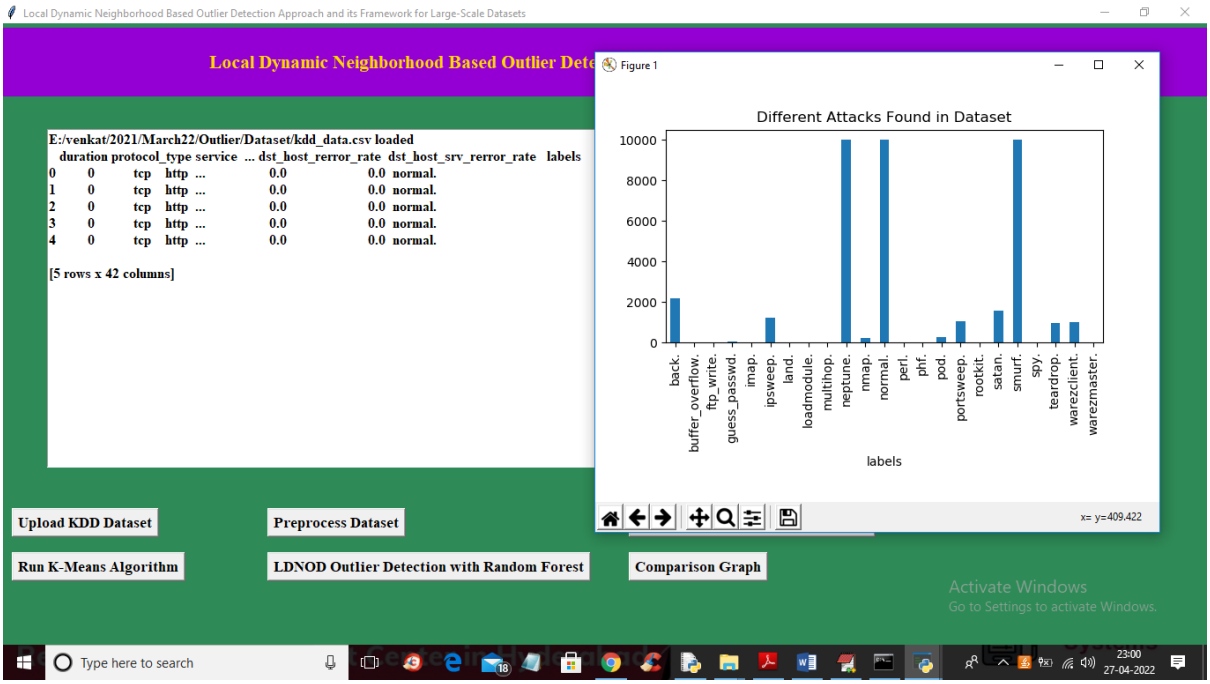
To run project double click on ‘run.bat’ file to get below screen



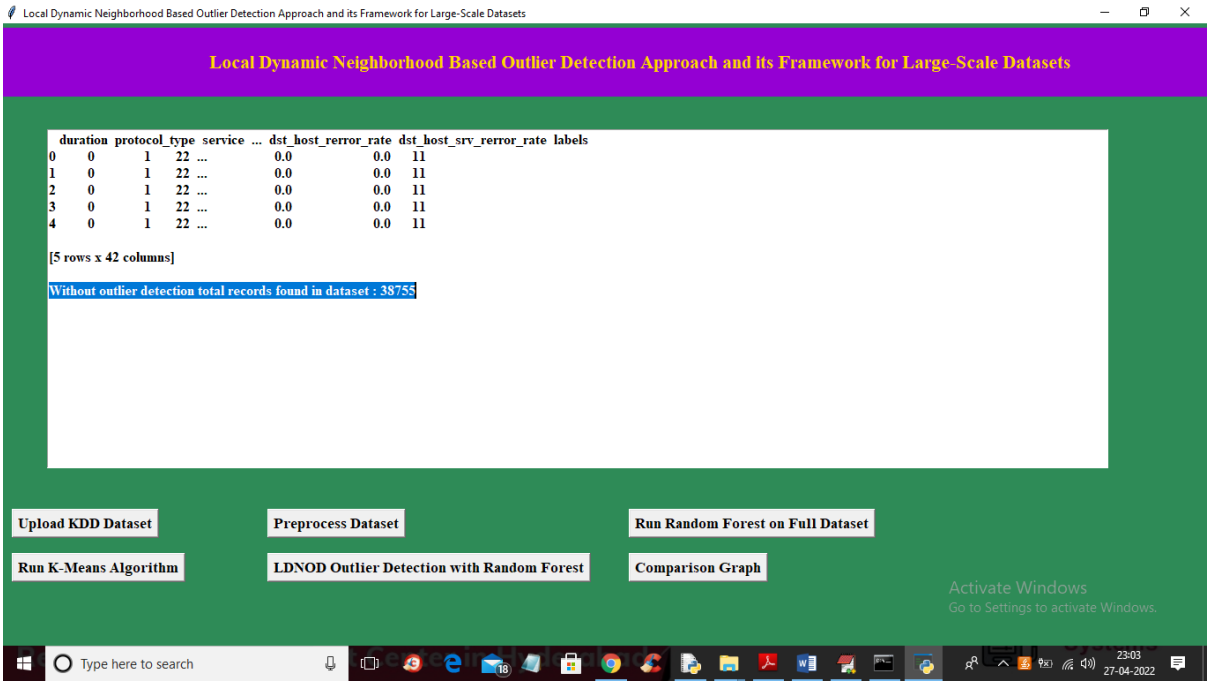
In above screen click on ‘Upload KDD Dataset’ button to upload dataset and to get below screen



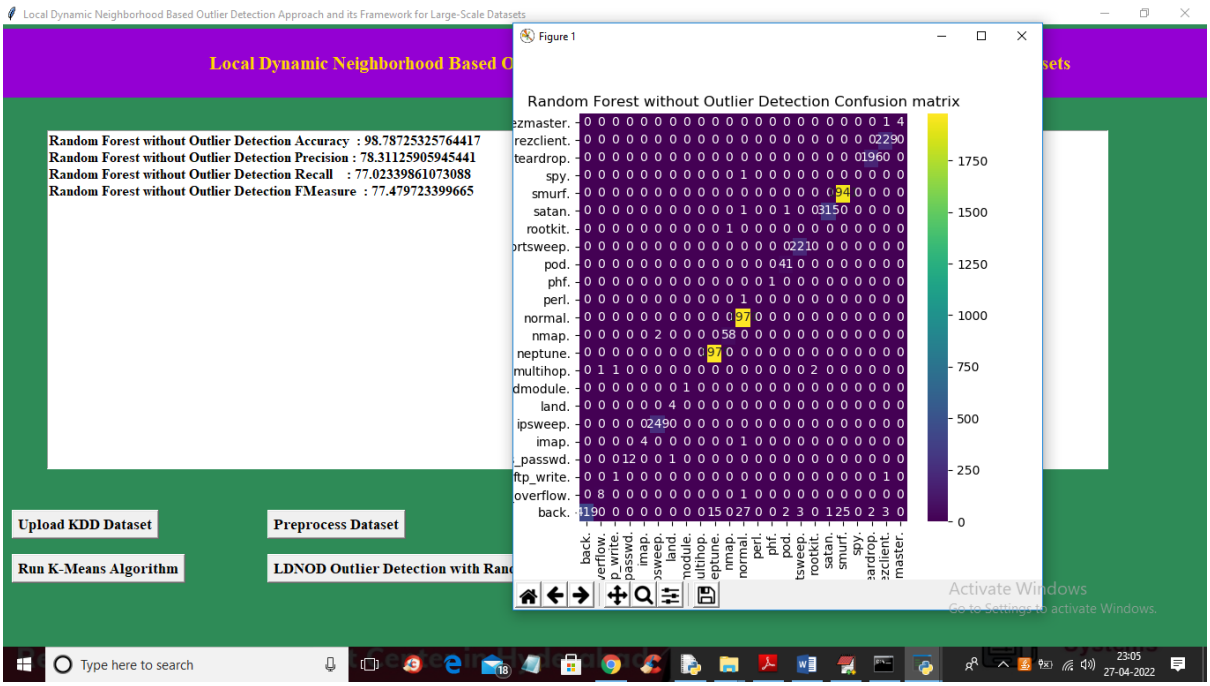
In above screen selecting and uploading ‘kdd dataset’ file and then click on ‘Open’ button to load dataset and to get below output



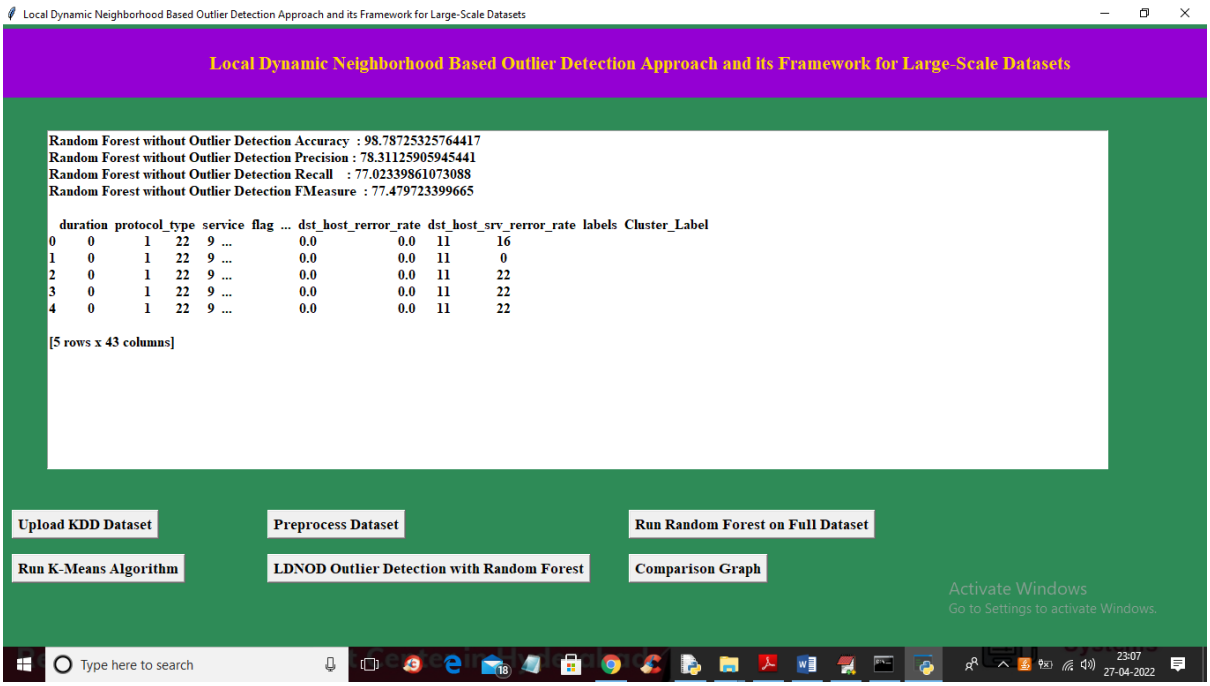
In above screen dataset loaded and displaying some values and dataset does not contains any cluster labels and after applying KMEANS will get cluster label. In above graph x-axis contains attack names and y-axis contains count of each attack found in dataset. In above screen dataset contains some non-numeric data so close above graph and then click on 'Preprocess Dataset' button to process data and get below output



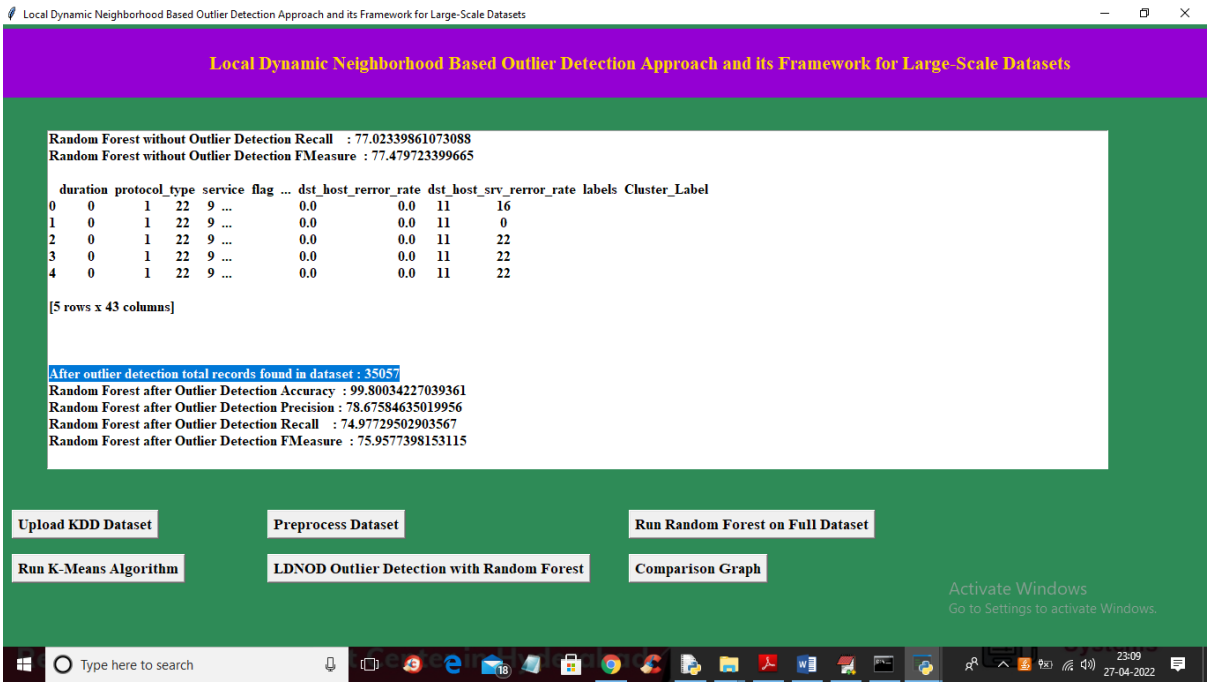
In above screen we can see all values are converted to numeric data and in blue line we can see dataset contains 38755 records without outlier detection and now click on 'Run Random Forest on Full Dataset' button to train Random Forest and get below output



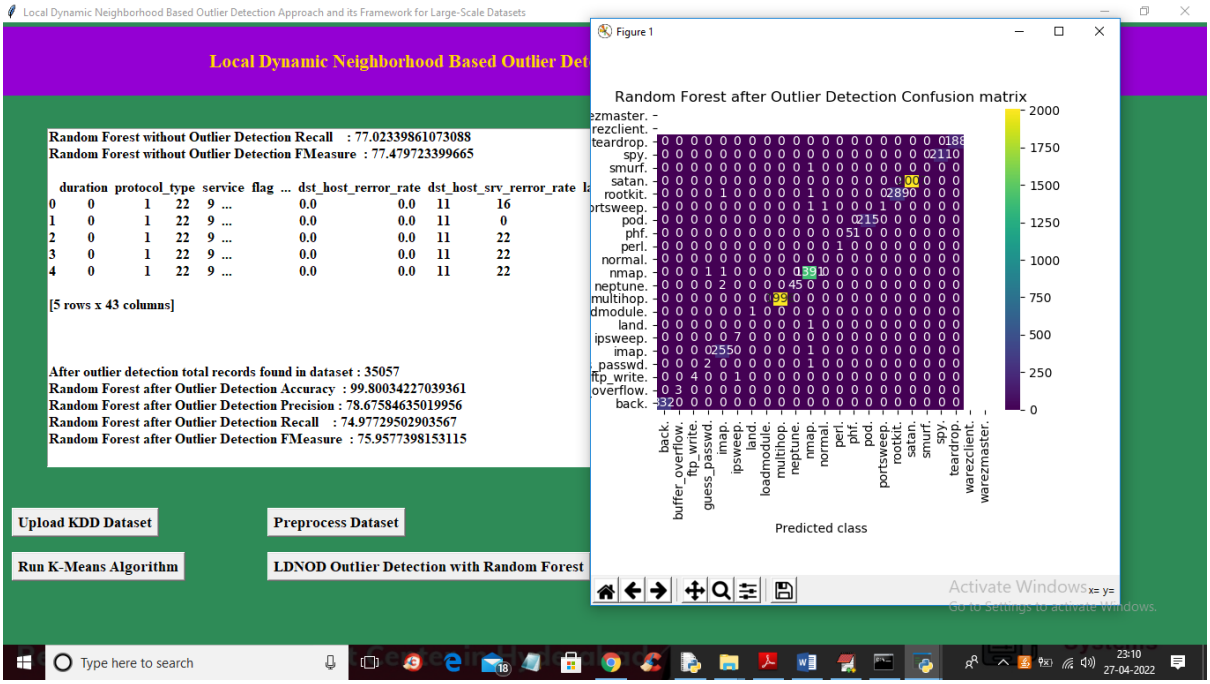
In above screen on full dataset with Random Forest we got 98% accuracy and in confusion matrix graph x-axis represents predicted attacks and y-axis represents TRUE test data attacks and in diagnol we can see both predicted and true labels are matching and now close above graph and then click on 'Run K-Means Algorithm' button to cluster entire dataset and get below output



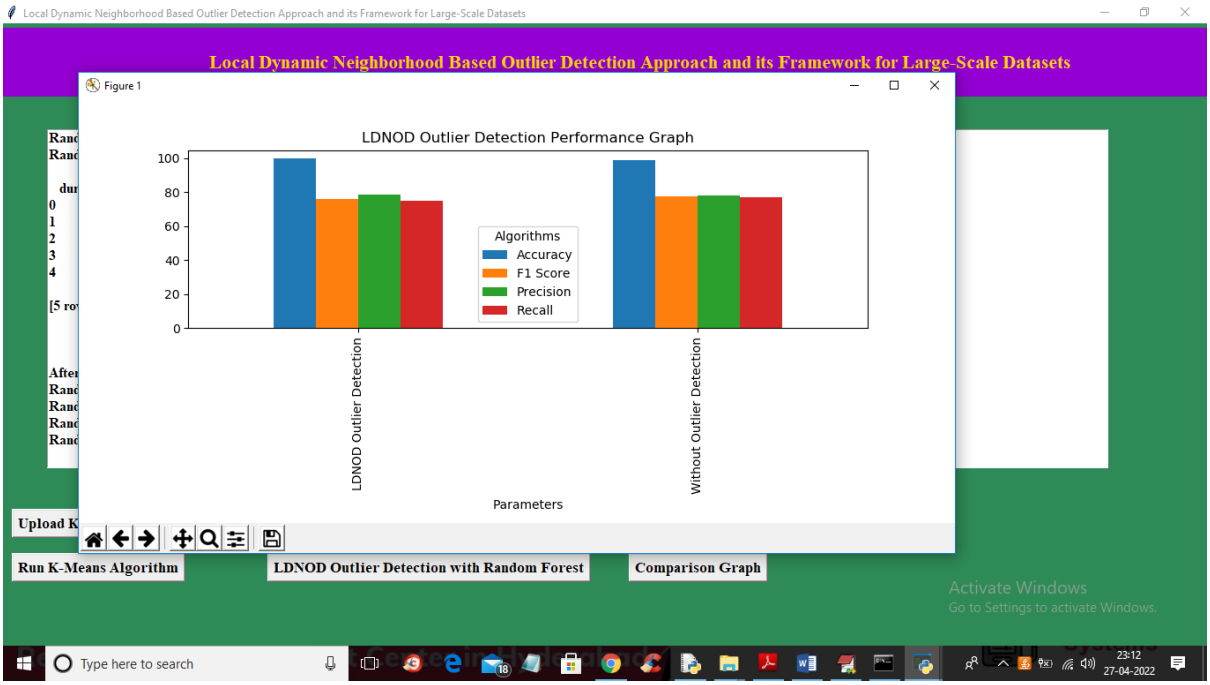
In above screen we can see cluster label added in last column and 16, 0, 22 are the cluster ID and now click on 'LDNOD Outlier Detection with Random Forest' button to apply LDNOD algorithm to compute similarity score between current record and neighbour records and if score is high then we will take that record and get below accuracy



In above screen in blue colour text we can see after applying outlier dataset size reduced to 35000 from 38000 and after applying outlier we got accuracy with same random forest as 99.80 and below is the confusion matrix graph



In above graph after applying outlier confusion diagnol boxes contains more number of correct prediction and now close above graph and then click on 'Comparison Graph' to get below graph



In the above graph x-axis represents technique name and y-axis represents accuracy and other metrics such as precision, recall and FSCORE with different colour bars and in both algorithm we got high accuracy after outlier detection.

14. CONCLUSION :

In this paper, a new local detection algorithm (LDNOD) and its framework (LDNOD-km) have been proposed. The LDNOD is insensitive to neighborhood parameter due to the stability of DRNN, which constructs neighborhood of an instance based on dynamic reference objects. Moreover, sharing neighborhoods of close objects and scoring each local region are designed, which can potentially reduce the running time of LDNOD. Because the LDNOD-km combines the benefits of both LDNOD and k-means, it is able to handle large-scale datasets efficiently without sacrificing accuracy. Finally, experimental results have demonstrated the effectiveness of LDNOD and its framework. In the future, we will further improve LDNOD-km and apply it to handle larger scale and high dimensional datasets.

SECTION 13

REFERENCES :

[1] Hawkins D. Identification of outliers. Chapman and Hall; 1980.

[2] Barnett V, Lewis T. Outliers in statistical data. 3rd ed, 1994.

[3] Hodge V, Austin J. A survey of outlier detection methodologies. Artif Intell Rev 2004;22(2):85–126.

[4] Chandola V, Banerjee A, Kumar V. Anomaly detection: a survey. ACM Comput Surveys 2009;41(3):15.

[5] Campos GO, Zimek A, Sander J, Campello RJ, Micenkov B, Schubert E, Assent I, Houle ME. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Min Knowl Discov 2016;30(4):891–927.

[6] Goldstein M, Uchida S. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PloS One 2016;11(4):e0152173.

[7] Domingues R, Filippone M, Michiardi P, Zouaoui J. A comparative evaluation of outlier detection algorithms: experiments and analyses. Pattern Recogn 2018;74:406–21.

[8] Weller-Fahy DJ, Borghetti BJ, Sodemann AA. A survey of distance and similarity measures used within network intrusion anomaly detection. IEEE Commun Surveys Tutor 2015;17(1):70–91.

[9] Ahmed M, Mahmood AN, Islam MR. A survey of anomaly detection techniques in financial domain. Future Gen Comput Syst 2016;55:278–88.

[10] Djenouri Y, Zimek A. Outlier detection in urban traffic data. In: Proceedings of the 8th international conference on web intelligence, mining and semantics. ACM; 2018.

[11] Yu R, He X, Liu Y. GLAD: group anomaly detection in social media analysis. ACM Trans Knowl Discov Data 2015;10(2):18.

[12] Riazi M, Zaiane O, Takeuchi T, Maltais A, G++nther J, Lipsett M. Detecting the onset of machine failure using anomaly detection methods.

[13] Zhou JT, Du J, Zhu H, Peng X, Liu Y, Goh RSM. AnomalyNet: an anomaly detection network for video surveillance. IEEE Trans Inf Forensics Secur 2019.

[14] Breunig MM, Kriegel HP, Ng RT, Sander J. May. LOF: identifying density-based local outliers. ACM SIGMOD Record 2000;29(2):93–104.

[15] Tang J, Chen Z, Fu A, Cheung D. Enhancing effectiveness of outlier detections for low density patterns. In: Chen MS, Yu P, Liu B, editors. Advances in knowledge discovery and data mining. vol. 2336 of Lecture Notes in Computer Science. Springer: Berlin/Heidelberg; 2002. pp. 535–548.

[16] Papadimitriou S, Kitagawa H, Gibbons PB, Faloutsos C. LOCI: Fast Outlier Detection Using the Local Correlation Integral. In: Proceedings of the 19th International Conference on Data Engineering. Los Alamitos, CA, USA: IEEE Computer Society Press; 2003. p. 315–326.

[17] Jin W, Tung A, Han J, Wang W. Ranking Outliers Using Symmetric Neighborhood Relationship. In: Ng WK, Kitsuregawa M, Li J, Chang K, editors. Advances in Knowledge Discovery and Data Mining. vol. 3918 of Lecture Notes in Computer Science. Springer, Berlin/ Heidelberg; 2006. p. 577– 593.

[18] Kriegel HP, Krger P, Schubert E, LoOP Zimek A. Local outlier probabilities. In: Proceeding of the 18th ACM conference on information and knowledge management (CIKM-09). New York, NY, USA: ACM Press; 2009. p. 1649–52.

[19] Zhang K, Hutter M, Jin H. A new local distance–based outlier detection approach for scattered real-world data. Pacific-Asia Conference on Knowledge Discovery and Data Mining; 2009. pp. 813–822.

[20] Yiu ML, Mamoulis N. Reverse nearest neighbors search in ad hoc subspaces. IEEE Trans Knowl Data Eng 2007;19(3):412–26.

[21] Wang Chai. A pruning based continuous RKNN query algorithm for large k. Chin J Electr 2012;3:523–7.

[22] Zhu Qingsheng, Feng J, Huang J. Natural Neighbor: a self-adaptive neighborhood method without parameter K. Pattern Recogn Lett 80; 2016.

[23] Huang J, Zhu Q, Yang L, Feng J. A non-parameter outlier detection algorithm based on natural neighbor. Knowl-Based Syst 2016;92(C):71–7.

[24] He Z, Xu X, Deng S. Discovering cluster-based local outliers. Pattern Recogn Lett 2003;24(9–10):1641–50.

[25] Amer M, Goldstein M. Nearest-neighbor and clustering based anomaly detection algorithms for RapidMiner. In: Simon Fischer IM, editor. Proceedings of the 3rd RapidMiner Community Meeting and Conferernce (RCOMM 2012). Shaker Verlag GmbH; 2012. p. 1–12.

[26] Goldstein M, Dengel A. Histogram-based outlier score (HBOS): a fast Unsupervised Anomaly Detection Algorithm. In: Wlfl S, editor. KI-2012: Poster and Demo Track. Online; 2012. p. 59–63.

[27] Zhao Y, Ding X, Yang J, Bai H. Toward scalable unsupervised outlier detection. Workshops at the Thirty–Fourth AAAI Conference on Artificial Intelligence, 2020