ABSTRACT

Anomaly detection in network traffic is a critical aspect of cybersecurity to identify suspicious activities indicative of potential attacks or breaches. Leveraging machine learning techniques, this study aims to develop a robust anomaly detection system using the KDD Cup dataset. The research methodology involves a comprehensive analysis of network traffic patterns, including normal behaviors and various attack scenarios represented in the dataset. Features extracted from network traffic data, such as duration, protocol type, service, and flag, serve as inputs for machine learning algorithms. To train and validate the anomaly detection model, the KDD Cup dataset is utilized, comprising labeled instances of normal and anomalous network connections. Unsupervised learning approaches are explored to effectively identify outliers and anomalies in network traffic. Furthermore, feature engineering techniques and dimensionality reduction methods, such as principal component analysis (PCA), are employed to enhance model performance and scalability. The developed anomaly detection system is evaluated using metrics like precision, recall, and F1-score to assess its effectiveness in accurately detecting network-based threats while minimizing false positives. Through this research, we aim to contribute to the advancement of cybersecurity practices by providing a reliable and efficient solution for detecting anomalous network behaviors, thus fortifying network defenses and safeguarding against potential cyber threats.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER-1
INTRODUCTION

INTRODUCTION

In the contemporary digital realm, where connectivity is ubiquitous, cybersecurity stands as a pivotal domain demanding unwavering attention from both entities and individuals. As the landscape continually evolves, propelled by technological advancements, the threats lurking in the digital ether have grown in sophistication and frequency. Among these perils, network-based threats loom large, presenting formidable challenges to the integrity and confidentiality of data.

Malware, phishing schemes, and denial-of-service (DoS) attacks represent just a fraction of the arsenal wielded by cyber adversaries, each posing unique risks to the sanctity of digital infrastructure. To confront these threats head-on, organizations and cybersecurity professionals must deploy proactive defense mechanisms capable of swiftly identifying and neutralizing anomalous activities within network traffic.

Enter machine learning (ML), a field at the forefront of innovation in cybersecurity. ML techniques, with their ability to discern patterns and anomalies within vast datasets, offer a beacon of hope in the fight against cyber threats. By leveraging ML algorithms, cybersecurity practitioners can develop sophisticated anomaly detection systems capable of sifting through the deluge of network traffic data to pinpoint suspicious behaviors that may signify an ongoing or imminent attack.

At the heart of this research endeavor lies the KDD Cup dataset, a cornerstone resource in the field of cybersecurity. This dataset, replete with labeled examples of benign and malicious network traffic, serves as a fertile testing ground for the development and validation of anomaly detection algorithms. Armed with this dataset, researchers embark on a journey to devise a comprehensive solution that not only identifies known threats but also adapts to emerging risks in real-time.

The envisioned anomaly detection solution is not merely a reactive tool but a proactive guardian of digital assets. By harnessing the power of ML, it can learn from past incidents, anticipate future threats, and fortify defenses accordingly. Through continuous refinement and iteration, this solution evolves alongside the ever-shifting cybersecurity landscape, offering a robust bulwark against the nefarious forces that seek to exploit vulnerabilities in network infrastructure.

In essence, this research represents a concerted effort to marry the realms of machine learning and cybersecurity, forging a symbiotic relationship that empowers defenders to stay one step ahead of adversaries. By embracing innovation and leveraging cutting-edge technologies, we strive to usher in a new era of resilience and security in an increasingly interconnected world.

1.1 PROBLEM DEFINITION

The primary objective of this research is to design and implement an effective anomaly detection system for network traffic analysis. The proliferation of cyber threats poses significant risks to organizational infrastructure, data integrity, and user privacy. Traditional security mechanisms often fall short in detecting sophisticated and stealthy attacks, necessitating the development of advanced anomaly detection techniques. The key challenge lies in accurately distinguishing between normal network behaviors and malicious activities, given the complex and dynamic nature of modern cyber threats.

1.2 MOTIVATION

In an era dominated by digital connectivity and reliance on technology, the specter of cyber threats looms larger than ever. From nation-state actors to sophisticated criminal enterprises, the landscape of cyberattacks is constantly evolving, presenting formidable challenges to the security of organizations, governments, and individuals worldwide. Despite substantial investments in cybersecurity measures, traditional defense mechanisms often struggle to keep pace with the ingenuity and agility of cyber adversaries. Amidst this backdrop, the potential of machine learning (ML) to revolutionize cybersecurity is increasingly recognized. ML algorithms, with their ability to analyze vast datasets, detect subtle patterns, and adapt to dynamic environments, offer a new frontier in the quest for proactive threat detection and mitigation. By leveraging the power of ML-driven anomaly detection, organizations can augment their defensive capabilities, gaining insight into emerging threats and preemptively neutralizing risks before they materialize into breaches. The impetus behind this research lies in the imperative to bridge the gap between evolving cyber threats and the defenses required to counter them. By harnessing the potential of ML algorithms and advanced analytics, this study aims to develop innovative anomaly detection systems capable of identifying anomalous behaviors indicative of potential security breaches. Through rigorous experimentation and analysis, we seek to advance the state-of-the-art in cybersecurity, equipping organizations with the tools and insights needed to stay ahead of emerging threats. Furthermore, this research endeavor is motivated by the broader goal of fostering collaboration and knowledge-sharing within the cybersecurity community. By disseminating best practices, lessons learned, and cutting-edge research findings, we aim to empower cybersecurity professionals and organizations to collectively confront the challenges posed by cyber threats. Ultimately, our motivation is rooted in the belief that through collaborative efforts and technological innovation, we can build a more secure and resilient digital ecosystem for all stakeholders.

1.3 **OBJECTIVES**

1) Implement an anomaly detection project using unsupervised machine learning techniques.
2) Perform preprocessing on the KDD dataset, including data loading, cleaning, and preprocessing to prepare the data for analysis.
3) Conduct feature selection or extraction to identify relevant features and reduce dimensionality, focusing on the most informative aspects of the data.
4) Utilize clustering algorithms such as K-means and DBSCAN to identify clusters within the dataset, effectively segmenting the data based on similarities.
5) Evaluate the performance of different clustering algorithms and configurations to determine the most effective approach for anomaly detection.
6) Identify clusters that deviate significantly from the norm as potential anomalies, indicating compromised systems or unusual behavior within the network.
7) Compare the effectiveness of different anomaly detection methods and configurations to identify the optimal approach for detecting and mitigating cyber threats.
8) Develop insights and recommendations for enhancing anomaly detection capabilities in real-world cybersecurity environments, based on the findings of the project

CHAPTER-2
LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 Related Work

1. COMPROMISING SYSTEMS: IMPLEMENTING HACKING PHASES

In the cyber world, more and more cyber-attacks are being perpetrated. Hackers have now become the warriors of the internet. They attack and do harmful things to compromised system. This paper will show the methodology used by hackers to gain access to systems, the different tools used by them, and how they are grouped based on their skills. It will identify exploits that can be used to attack a system and find mitigation for those exploits. In addition, the paper discusses the actual implementation of the hacking phases and the virtual machines used in the process. The virtual machines specification is also listed. It will also provide means and insights on how to protect one system from being compromised.

2. A COMPREHENSIVE STUDY ON ETHICAL HACKING

This study on ethical hacking provides insights into the methodologies and practices used by ethical hackers to identify vulnerabilities and enhance system security. While the focus of the paper is on ethical hacking techniques, it indirectly addresses the importance of anomaly detection in cybersecurity. Ethical hackers often employ anomaly detection techniques as part of their arsenal to proactively safeguard systems against cyber threats.

“Role of Ethical Hacking in Systems,” no. May, 2018.

This paper highlights the role of ethical hacking in bolstering system security and resilience against cyber threats. It emphasizes the proactive approach of ethical hackers to identifying anomalies and potential vulnerabi2018, within systems. By employing various techniques, such as vulnerability scanning and penetration testing, ethical hackers contribute to the development of robust anomaly detection systems.

“What is a white hat? (definition from WhatIs.com)

The definition provided by WhatIs.com elucidates the concept of white-hat hackers, also known as ethical hackers, who engage in security testing and vulnerability assessments to identify and mitigate potential security risks. Ethical hackers often leverage anomaly detection methodologies to identify and address potential security loopholes, thereby enhancing the overall security posture of organizations and individuals.

“Types of Hackers and What They Do: White, Black, and Grey | EC-Council Official Blog.”

The EC-Council Official Blog categorizes hackers into different types and discusses their motivations and behaviors. While the focus of the blog post is on various hacker types, it underscores the proactive nature of white-hat hackers in employing anomaly detection techniques to safeguard systems against cyber threats.

3. Types of Attacks

Types of Attacks and Examples: Anomaly detection in cybersecurity often involves the classification of attacks into distinct categories, each with its own characteristics and methodologies. The KDD Cup dataset, a widely used benchmark dataset in the field of intrusion detection, contains examples of various types of attacks, including:

Denial of Service Attack (DoS):
Examples: back, land, neptune, pod, smurf, teardrop
Description: A DoS attack is characterized by the attacker making computing or memory resources too busy or too full to handle legitimate requests, thereby denying access to legitimate users.
User-to-Root Attack (U2R):
Examples: buffer_overflow, ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
Description: A U2R attack occurs when the attacker gains access to a normal user account on the system and exploits vulnerabilities to escalate privileges to root access. Remote to Local Attack (R2L):
Examples: rootkit, perl, loadmodule
Description: An R2L attack happens when an attacker with network access exploits vulnerabilities to gain local access as a user of the target machine.
Probing Attack:
Examples: ipsweep, nmap, portsweep, satan

Description: A probing attack involves attempting to gather information about a network of computers for the purpose of circumventing security controls. By classifying attacks into these categories and analyzing their characteristics, researchers and practitioners can develop more effective anomaly detection systems capable of identifying and mitigating various types of cyber threats

4. Machine Learning for anomaly Detection : A Systematic Review

Anomaly detection has been used for decades to identify and extract anomalous components from data. Many techniques have been used to detect anomalies. One of the increasingly significant techniques is Machine Learning (ML), which plays an important role in this area. In this research paper, we conduct a Systematic Literature Review (SLR) which analyzes ML models that detect anomalies in their application. Our review analyzes the models from four perspectives; the applications of anomaly detection, ML techniques, performance metrics for ML models, and the classification of anomaly detection. In our review, we have identified 290 research articles, written from 2000-2020, that discuss ML techniques for anomaly detection. After analyzing the selected research articles, we present 43 different applications of anomaly detection found in the selected research articles. Moreover, we identify 29 distinct ML models used in the identification of anomalies. Finally, we present 22 different datasets that are applied in experiments on anomaly detection, as well as many other general datasets. In addition, we observe that unsupervised anomaly detection has been adopted by researchers more than other classification anomaly detection systems. Detection of anomalies using ML models is a promising area of research, and there are a lot of ML models that have been implemented by researchers. Therefore, we provide researchers with recommendations and guidelines based on this review.

CHAPTER-3
METHODOLOGY

METHODOLOGY

For the implementation of our project, we performed the following steps:

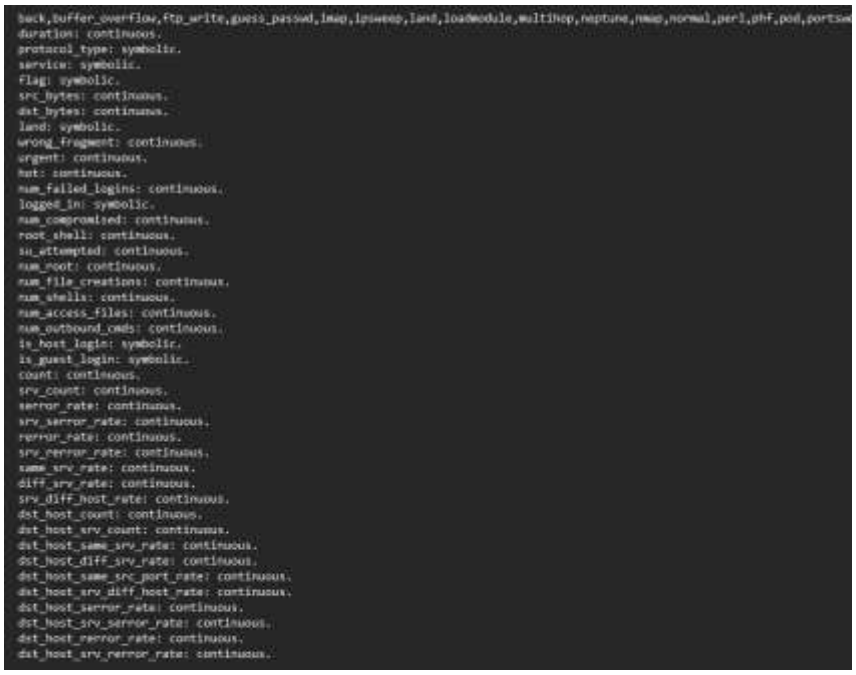
3.1 Preparation of Dataset

To prepare the dataset for anomaly detection using the KDD Cup dataset, we need to follow several steps:

1. Obtaining the KDD Cup Datase: The KDD Cup dataset is a well-known benchmark dataset in cybersecurity. It contains a vast amount of network traffic data, including both normal and anomalous instances, making it suitable for training and evaluating anomaly detection models.

2.Understanding the Dataset Fields: The dataset contains various fields that describe different aspects of network traffic. Some of the fields mentioned in your list include:

- `back`, `buffer_overflow`, `ftp_write`, `guess_p`: These are categorical fields representing different types of attacks or anomalies. Each field indicates whether a particular type of attack is present or not.
- `srv_count`, `serror_rate`, `same_srv_rate`: These are continuous numerical fields providing information about the number of services, error rates, and rates of connections to the same service, respectively.
- `flags`: This is a categorical field indicating flags associated with network connections.
- `src_bytes`, `dst_bytes`: These are continuous numerical fields representing the number of source and destination bytes transferred in the connection.
- `service`: This is a categorical field specifying the network service associated with the connection.



```
back,buffer_overflow,ftp_write,guess_passwd,imap,ipsweep,land,loadmodule,multihop,neptune,nmap,normal,perl,phf,pod,portsw
duration: continuous.
protocol_type: symbolic.
service: symbolic.
flag: symbolic.
src_bytes: continuous.
dst_bytes: continuous.
land: symbolic.
wrong_fragment: continuous.
urgent: continuous.
hot: continuous.
num_failed_logins: continuous.
logged_in: symbolic.
num_compromised: continuous.
root_shell: continuous.
su_attempted: continuous.
num_root: continuous.
num_file_creations: continuous.
num_shells: continuous.
num_access_files: continuous.
num_outbound_cmds: continuous.
is_host_login: symbolic.
is_guest_login: symbolic.
count: continuous.
srv_count: continuous.
serror_rate: continuous.
srv_serror_rate: continuous.
rerror_rate: continuous.
srv_rerror_rate: continuous.
same_srv_rate: continuous.
diff_srv_rate: continuous.
srv_diff_host_rate: continuous.
dst_host_count: continuous.
dst_host_srv_count: continuous.
dst_host_same_srv_rate: continuous.
dst_host_diff_srv_rate: continuous.
dst_host_same_src_port_rate: continuous.
dst_host_srv_diff_host_rate: continuous.
dst_host_serror_rate: continuous.
dst_host_srv_serror_rate: continuous.
dst_host_rerror_rate: continuous.
dst_host_srv_rerror_rate: continuous.
```

fig1. Fields of dataset

3.2 Pre-processing

Loading data from pre-saved files and storing it into variables like train, train10, and test.

One hot encode

One-hot encoding is a common technique used to represent categorical variables as binary vectors. In the context of your project, where you've separated the class protocol into smaller categories like protocol_type_tcp and protocol_type_udp, as well as handling service and flag categories, one-hot encoding can be applied to each of these categorical features.

Here's how you can perform one-hot encoding for each categorical feature:

1. Protocol Type:
   - Create binary variables for each protocol type, such as protocol_type_tcp and protocol_type_udp.
   - For each instance, set the value to 1 for the corresponding protocol type and 0 for others.

2. Service:
   - Create binary variables for each service category.
   - Assign a value of 1 to the respective service category and 0 to others for each instance.

3. Flag:
   - Similar to the above, create binary variables for each flag category.
   - Set the value to 1 for the relevant flag and 0 for others for each instance.

After performing one-hot encoding for these categorical features, you'll end up with a dataset where each categorical feature has been transformed into multiple binary features, each representing a specific category within that feature.

For example, if you initially had a protocol_type feature with values like "tcp" and "udp", after one-hot encoding, you'll have two separate binary features: protocol_type_tcp and protocol_type_udp. If an instance originally had a protocol_type value of "tcp", the protocol_type_tcp feature would be set to 1, and protocol_type_udp would be set to 0.

This process allows machine learning algorithms to effectively utilize categorical data in predictive modeling tasks, ensuring that the model can interpret and learn from these features accurately.

PCA

Principal Component Analysis (PCA) is a powerful technique for dimensionality reduction, particularly useful when dealing with high-dimensional datasets. In your project, you aim to apply PCA to the training data `x_train` to reduce its dimensionality while preserving most of the variance in the data. Let's break down the process:

1. Import PCA from Scikit-Learn: First, you need to import the PCA class from the appropriate library. In Python, you can do this using Scikit-Learn:

```
#from sklearn.decomposition import PCA
```

2. Instantiate PCA: Next, you instantiate the PCA object. You specify the number of components you want to retain after dimensionality reduction. In this case, you want to retain enough principal components to explain at least 99.99% of the variance in the data.

#pca = PCA(n_components=0.9999, random_state=42)

3. Fit PCA to Training Data: Then, you fit the PCA object to your training data to learn the transformation matrix.

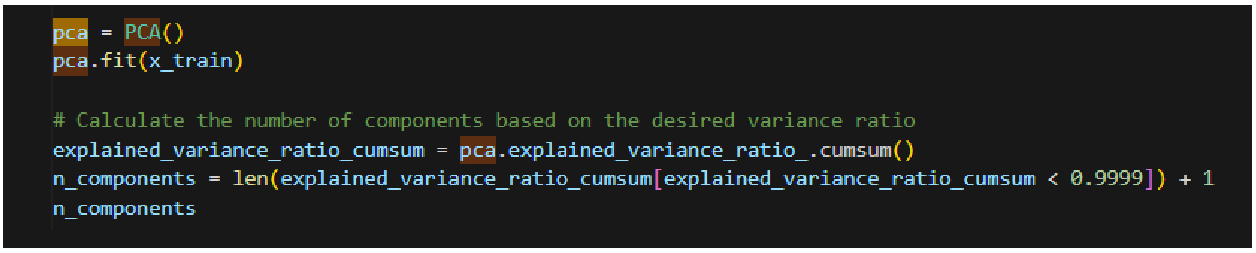
#pca.fit(x_train)

4. Transform Data: Finally, you transform both the training and testing data using the learned transformation matrix.

#x_train_pca = pca.transform(x_train)
#x_test_pca = pca.transform(x_test)

After applying PCA, your `x_train_pca` and `x_test_pca` datasets will have reduced dimensionality, containing only the principal components that capture the most important information in the original data while discarding the less informative dimensions.

It's crucial to note that the choice of the number of principal components or the explained variance threshold (`n_components`) may require experimentation and validation to ensure that the reduced-dimensional representation maintains the necessary information for your specific task while minimizing information loss.



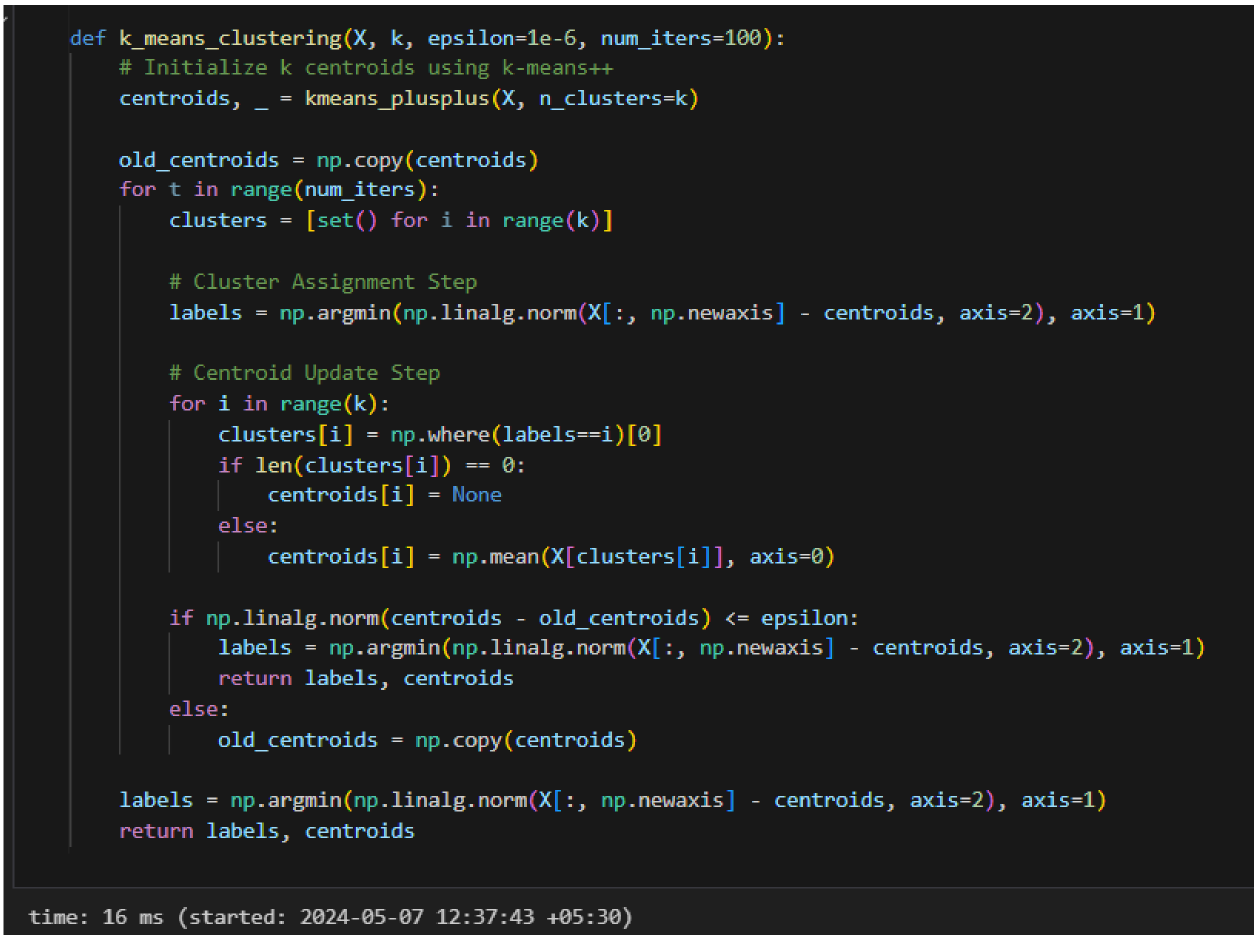
```python
pca = PCA()
pca.fit(x_train)

# Calculate the number of components based on the desired variance ratio
explained_variance_ratio_cumsum = pca.explained_variance_ratio_.cumsum()
n_components = len(explained_variance_ratio_cumsum[explained_variance_ratio_cumsum < 0.9999]) + 1
n_components
```

Fig2. PCA

3.3 Algorithims

3.3.1 K -means Clustering

K-means Clustering is a popular unsupervised machine learning algorithm that groups unlabeled data into clusters. The goal is to minimize the sum of the squared distances between the objects and their assigned cluster mean.
For our project, we used a range of clusters from 7 to 45 and calculated each cluster's accuracy!!!



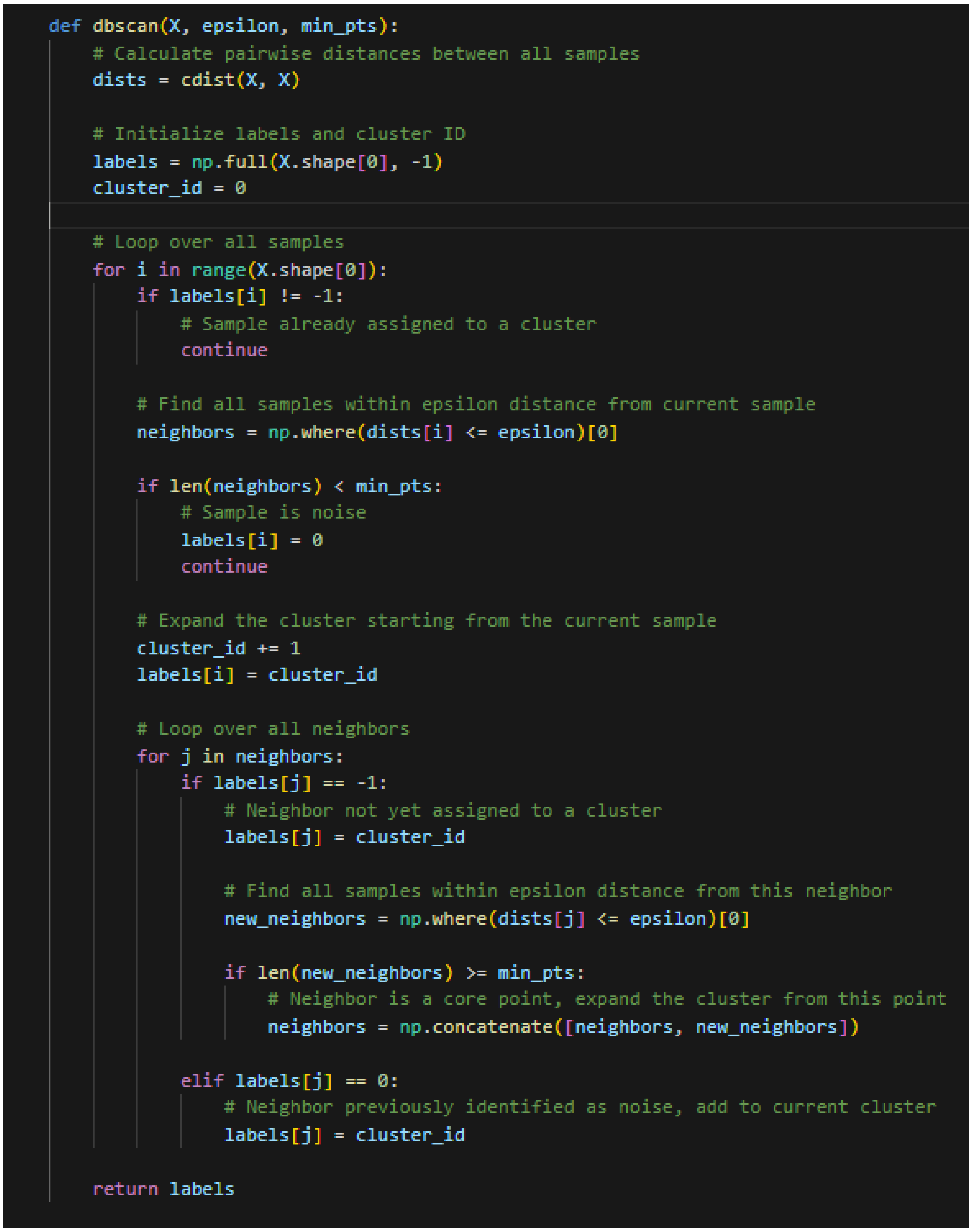
```python
def k_means_clustering(X, k, epsilon=1e-6, num_iters=100):
    # Initialize k centroids using k-means++
    centroids, _ = kmeans_plusplus(X, n_clusters=k)

    old_centroids = np.copy(centroids)
    for t in range(num_iters):
        clusters = [set() for i in range(k)]

        # Cluster Assignment Step
        labels = np.argmin(np.linalg.norm(X[:, np.newaxis] - centroids, axis=2), axis=1)

        # Centroid Update Step
        for i in range(k):
            clusters[i] = np.where(labels==i)[0]
            if len(clusters[i]) == 0:
                centroids[i] = None
            else:
                centroids[i] = np.mean(X[clusters[i]], axis=0)

        if np.linalg.norm(centroids - old_centroids) <= epsilon:
            labels = np.argmin(np.linalg.norm(X[:, np.newaxis] - centroids, axis=2), axis=1)
            return labels, centroids
        else:
            old_centroids = np.copy(centroids)

    labels = np.argmin(np.linalg.norm(X[:, np.newaxis] - centroids, axis=2), axis=1)
    return labels, centroids
```

```
time: 16 ms (started: 2024-05-07 12:37:43 +05:30)
```

Fig3. K-means Algorithm

3.3.3 DBSCAN

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is a popular clustering algorithm used for grouping together closely packed points in a dataset. Here's how it works:



```python
def dbscan(X, epsilon, min_pts):
    # Calculate pairwise distances between all samples
    dists = cdist(X, X)

    # Initialize labels and cluster ID
    labels = np.full(X.shape[0], -1)
    cluster_id = 0

    # Loop over all samples
    for i in range(X.shape[0]):
        if labels[i] != -1:
            # Sample already assigned to a cluster
            continue

        # Find all samples within epsilon distance from current sample
        neighbors = np.where(dists[i] <= epsilon)[0]

        if len(neighbors) < min_pts:
            # Sample is noise
            labels[i] = 0
            continue

        # Expand the cluster starting from the current sample
        cluster_id += 1
        labels[i] = cluster_id

        # Loop over all neighbors
        for j in neighbors:
            if labels[j] == -1:
                # Neighbor not yet assigned to a cluster
                labels[j] = cluster_id

                # Find all samples within epsilon distance from this neighbor
                new_neighbors = np.where(dists[j] <= epsilon)[0]

                if len(new_neighbors) >= min_pts:
                    # Neighbor is a core point, expand the cluster from this point
                    neighbors = np.concatenate([neighbors, new_neighbors])

            elif labels[j] == 0:
                # Neighbor previously identified as noise, add to current cluster
                labels[j] = cluster_id

    return labels
```

Fig 4. DBSCAN

3.4 TECHNOLOGIES USED

1. NumPy: NumPy is a fundamental package for scientific computing with Python. It provides support for arrays, matrices, and mathematical functions, making it essential for numerical computations in Python.

2. Matplotlib: Matplotlib is a plotting library for Python that provides a MATLAB-like interface for creating static, interactive, and animated visualizations. It's widely used for creating charts, graphs, histograms, and other types of plots.

3. Scikit-Learn: Scikit-Learn is a versatile machine learning library for Python. It provides simple and efficient tools for data mining and data analysis, including various supervised and unsupervised learning algorithms, as well as tools for model selection and evaluation.

4. SciPy: SciPy is a scientific computing library for Python that builds on top of NumPy. It provides additional functionality for optimization, integration, interpolation, linear algebra, and other scientific computing tasks.

5. Pandas: Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrame and Series, along with functions for data cleaning, transformation, and exploration. Pandas is particularly useful for working with structured data.

6. OS: The OS module in Python provides a way to interact with the operating system. It allows you to perform various operating system-related tasks, such as navigating file systems, executing commands, and managing files and directories.

CHAPTER-4
CONCLUSION

1.Result of PCA

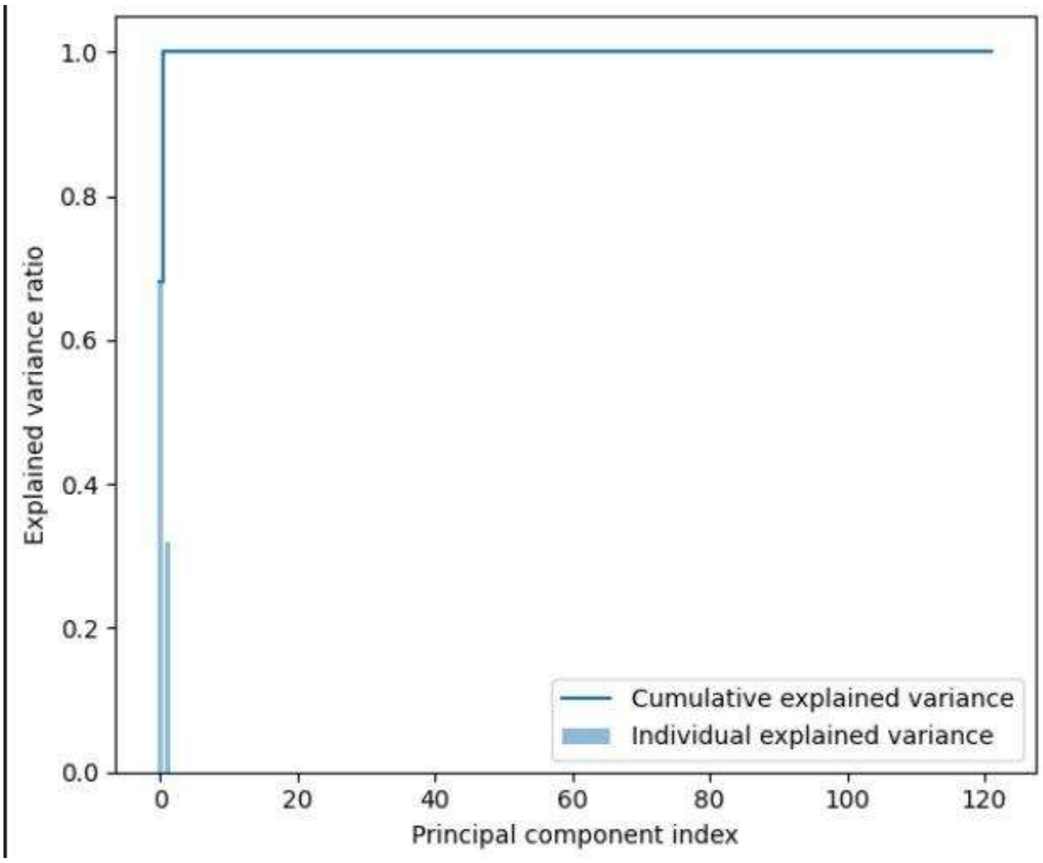
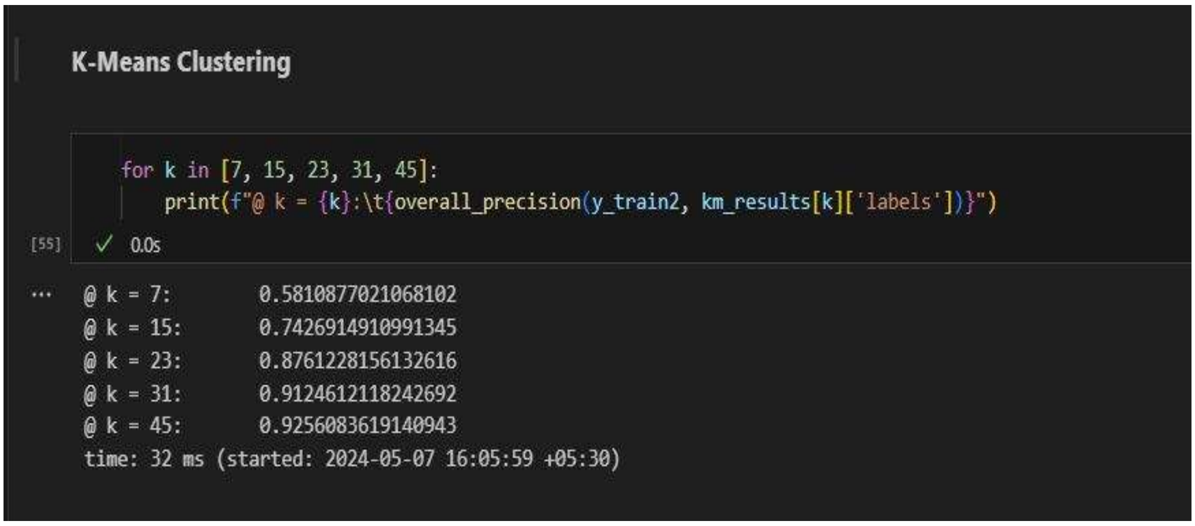


Fig 5. PCA result

2. Result of K-means



K-Means Clustering

```
for k in [7, 15, 23, 31, 45]:
    print(f"@ k = {k}:\t{overall_precision(y_train2, km_results[k]['labels'])}")
```

[55] ✓ 0.0s

```
@ k = 7:        0.5810877021068102
@ k = 15:       0.7426914910991345
@ k = 23:       0.8761228156132616
@ k = 31:       0.9124612118242692
@ k = 45:       0.9256083619140943
time: 32 ms (started: 2024-05-07 16:05:59 +05:30)
```

Fig 6. accuracy og kmeans

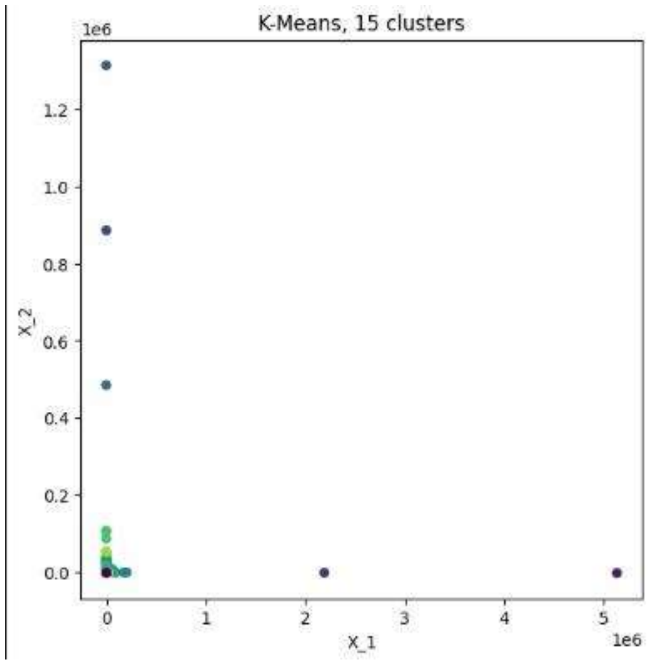


fig 7. K=15 kmeans

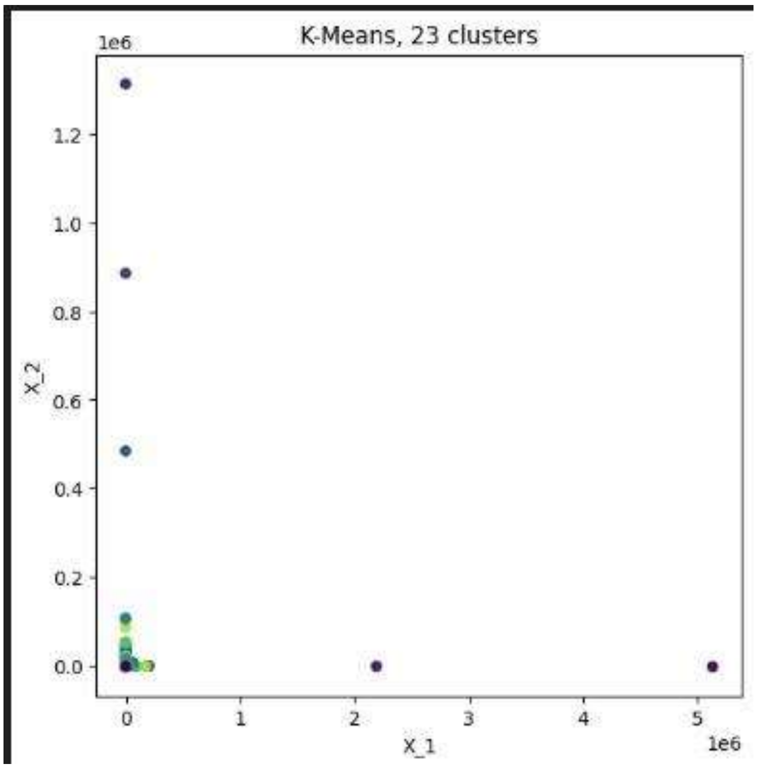


Fig 8 . k=23 kmeans

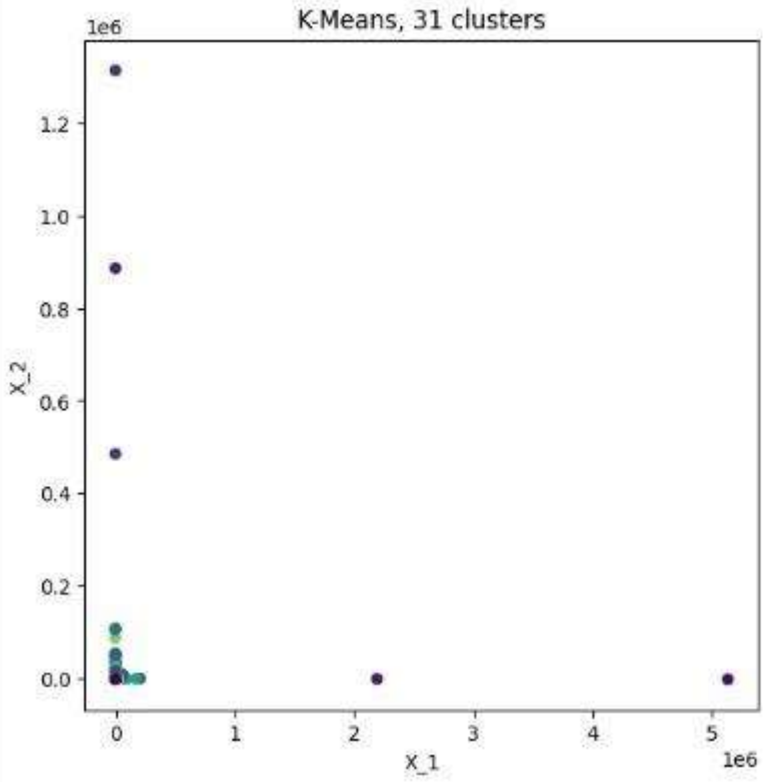


Fig 9. k=31 kmeans

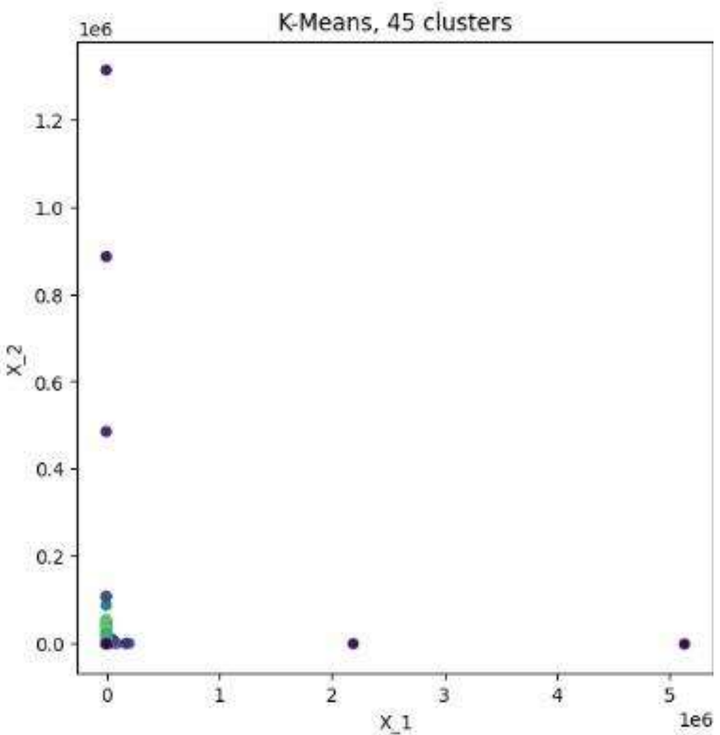


Fig 10. K=45 kmeans

3. Result of DBSCAN

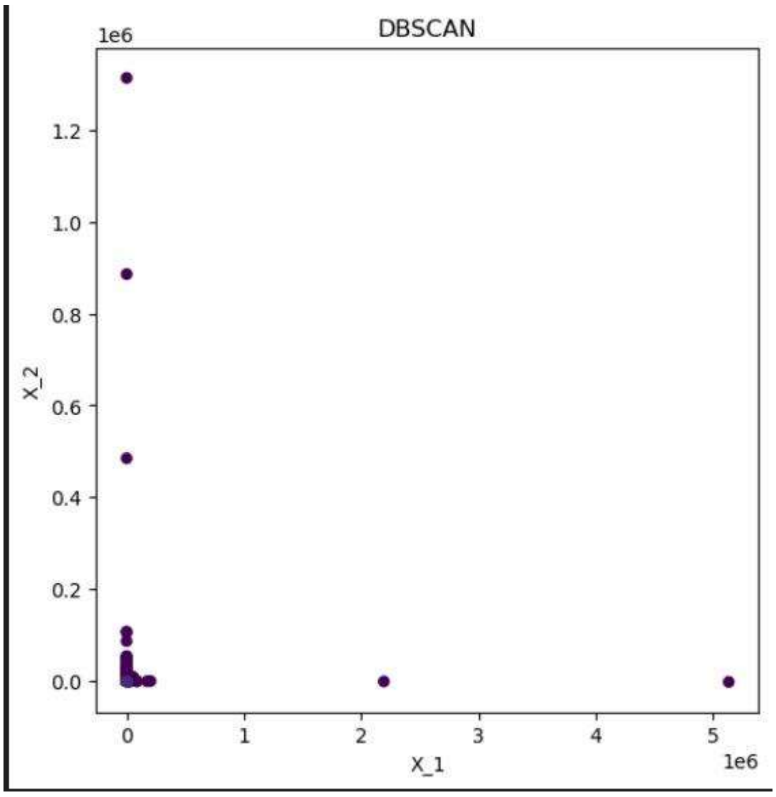


Fig. 11: Dbscan plot

4. Result of all K's

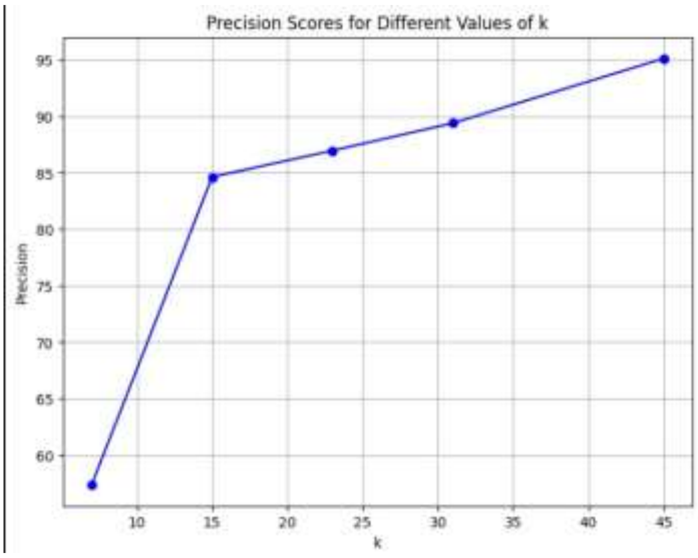


Fig. 12: K clusters plot

BIBLIOGRAPHY

[1] S. Begum and S. Kumar, "IJESRT INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY A COMPREHENSIVE STUDY ON ETHICAL HACKING," vol. 5, no. 8, pp. 214–219, 2016.

[2] M. Injadat, F. Salo, A. B. Nassif, A. Essex, and A. Shami, ‘‘Bayesian optimization with machine learning algorithms towards anomaly detection,’’ in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2018, pp. 1–6, doi: 10.1109/GLOCOM.2018.8647714.

[3] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, Unsupervised Anomaly Detection With Generative Adversarial Networks to Guide Marker Discovery, vol. 10265, no. 2. Cham, Switzerland: Springer, 2017.

[4] F. Salo, M. Injadat, A. B. Nassif, A. Shami, and A. Essex, ‘‘Data mining techniques in intrusion detection systems: A systematic literature review,’’ IEEE Access, vol. 6, pp. 56046–56058, 2018, doi: 10.1109/ACCESS.2018.2872784.

[5] F. Salo, M. N. Injadat, A. Moubayed, A. B. Nassif, and A. Essex, ‘‘Clustering enabled classification using ensemble feature selection for intrusion detection,’’ in Proc. Int. Conf. Comput., Netw. Commun. (ICNC), 2019, pp. 276–281, doi: 10.1109/ICCNC.2019.8685636.

[6] F. Salo, A. B. Nassif, and A. Essex, ‘‘Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection,’’ Comput. Netw., vol. 148, pp. 164–175, Jan. 2019, doi: 10.1016/J.COMNET.2018.11.010.

[7] P. Gogoi, D. K. Bhattacharyya, B. Borah, and J. K. Kalita, ‘‘A survey of outlier detection methods in network anomaly identification,’’ Comput. J., vol. 54, no. 4, pp. 570–588, Apr. 2011, doi: 10.1093/comjnl/bxr026.

[8] S. Agrawal and J. Agrawal, ‘‘Survey on anomaly detection using data mining techniques,’’ Procedia Comput. Sci., vol. 60, no. 1, pp. 708–713, 2015, doi: 10.1016/j.procs.2015.08.220.

[9] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. T. Hashem, E. Ahmed, and M. Imran, ‘‘Real-time big data processing for anomaly detection: A survey,’’ Int. J. Inf. Manage., vol. 45, pp. 289–307, Apr. 2019, doi: 10.1016/j.ijinfomgt.2018.08.006.

[10] V. Chandola, A. Banerjee, and V. Kumar, ‘‘Anomaly detection for discrete sequences: A survey,’’ IEEE Trans. Knowl. Data Eng., vol. 24, no. 5, pp. 823–839, Nov. 2012.