Automated Event Prioritization for Security Operation Center using Graph-based Features and Deep Learning

by

Nitika Jindal

B.Tech. Punjab Technical University, 2015

A Thesis Submitted in Partial
Fulfillment of the Requirements for the
Degree of

Master of Applied Science

in the Department of Electrical and Computer Engineering

SUPERVISORY COMMITTEE

Automated Event Prioritization for Security Operation Center using Graph-based Features and Deep Learning

by

Nitika Jindal

B.Tech. Punjab Technical University, 2015

Supervisory Committee

Dr. Issa Traore, Department of Electrical and Computer Engineering
Supervisor

Dr. Imen Bourguiba, Department of Electrical and Computer Engineering
Departmental Member

ABSTRACT

A security operation center (SOC) is a cybersecurity clearinghouse responsible for monitoring, collecting and analyzing security events from organizations' IT infrastructure and security controls. Despite their popularity, SOCs are facing increasing challenges and pressure due to the growing volume, velocity and variety of the IT infrastructure and security data observed on a daily basis. Due to the mixed performance of current technological solutions, e.g. intrusion detection system (IDS) and security information and event management (SIEM), there is an over-reliance on manual analysis of the events by human security analysts. This creates huge backlogs and slows down considerably the resolution of critical security events. Obvious solutions include increasing the accuracy and efficiency of crucial aspects of the SOC automation workflow, such as the event classification and prioritization. In the current thesis, we present a new approach for SOC event classification and prioritization by identifying a set of new machine learning features using graph visualization and graph metrics. Using a real-world SOC dataset and by applying different machine learning classification techniques, we demonstrate empirically the benefit of using the graph-based features in terms of improved classification accuracy. Three different classification techniques are explored, namely, logistic regression, XGBoost and deep neural network (DNN). The experimental evaluation shows for the DNN, the best performing classifier, area under curve (AUC) values of 91% for the baseline feature set and 99% for the augmented feature set that includes the graph-based features, which is a net improvement of 8% in classification performance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENT

I would like to first express my sincere gratitude to my supervisor, Dr. Issa Traore for his continuous support and motivation for me to pursue my studies and research at the University of Victoria. I am greatly appreciative to Dr. Issa Traore, who provided me an opportunity as his research student. It would not have been possible to conduct this research without his continuous encouragement and excellent mentorship.

I would also like to thank Dr. Imen Bourguiba, for serving on my supervisory committee.

Finally, I owe a deep sense of gratitude to my parents and to my friends for always being there and providing me continuous encouragement throughout my years of study.

Chapter 1: Introduction

1.1 Context

Over the last few years, there has been significant growth in the number of cyberattacks. In 2016, there were 758 million malicious attacks occurred according to KasperskyLab [2]. In 2014, a hack on Yahoo impacted more than 3 billion user accounts which is the biggest hacking of individuals against a single organization [11]. In 2017, NotPetya and WannaCry ransomware [2] disrupted many organizations' networks. In 2017, hackers targeted Equifax and compromised the personal information (name, address, social security number and credit card number) of over 145 million individuals [2].

Large organizations secure their computers and networks by using firewalls, Intrusion Detection System (IDS) and Intrusion Prevention Systems (IPS). Firewalls are used to protect the network by blocking and filtering unauthorized access and allowing only the authorized communications. Because firewalls can be compromised or bypassed, it is important to have additional protection mechanisms, such as IDS, which is widely used in organizations. Intrusion detection is the process of monitoring the events occurring in the network and devices and analyzing these events for signs of possible incidents or policy violations. Network-based Intrusion Detection System (NIDS) monitors network connections for suspicious activities for all the hosts in that network. Host-based Intrusion Detection System (HIDS) resides in the host and can monitor suspicious activities that are starting and coming to that host only [15].

In a broader way, intrusion detection can be categorized into two types: anomaly-based intrusion detection and signature-based intrusion detection. In anomaly-based intrusion detection we basically check for 'abnormal activities'. In this case, a baseline model is defined that characterizes normal system behavior, and if the traffic/activity is found to deviates from the baseline an alert is triggered. A signature-based detection model consists of predefined attack patterns encoded, for instance, using a rule base. Whenever a new event is generated the intrusion detection system checks it against the predefined patterns or rules in the database and if there is a match, an alert is generated.

Despite the benefits of IDS, current systems have several weaknesses, one of the major ones being the fact that it generates a lot of false alerts [8]. For instance, a rule for detecting Simple Network Management Protocol (SNMP) probes from external networks may also trigger intrusion detection alerts on SNMP GET, although these are merely a routine monitoring activity.

A large number of false alerts makes it very difficult for the security analysts to investigate the alerts and there is a high probability that the alerts which are important will be missed. Some organizations investigate alerts all by themselves but there are many organizations that want extra security on their data, and so rely on a Security Operation Center (SOC).

A security operation center (SOC) is a cybersecurity clearinghouse responsible for monitoring, collecting and analyzing security events from organizations' IT infrastructure and security controls. It is accountable for the continuous surveillance and analysis of the organization's data and provides security measures accordingly [12]. The typical SOC leverages cybersecurity tools for log aggregation, correlation and analysis, such as the security information and event management system (SIEM). But they also heavily depend on a dedicated team of security analysts who pore over the outputs of the tools to prioritize the events and conduct incident management and response activities, which are essential tasks in any threat hunting endeavor. In Figure 1.1, we can see that SOC is getting the alert logs generated by many different organizations IDS. The SOC team will investigate these alerts and notify the clients about suspicious alerts; alerts found not to be suspicious will be not be notified to the clients.

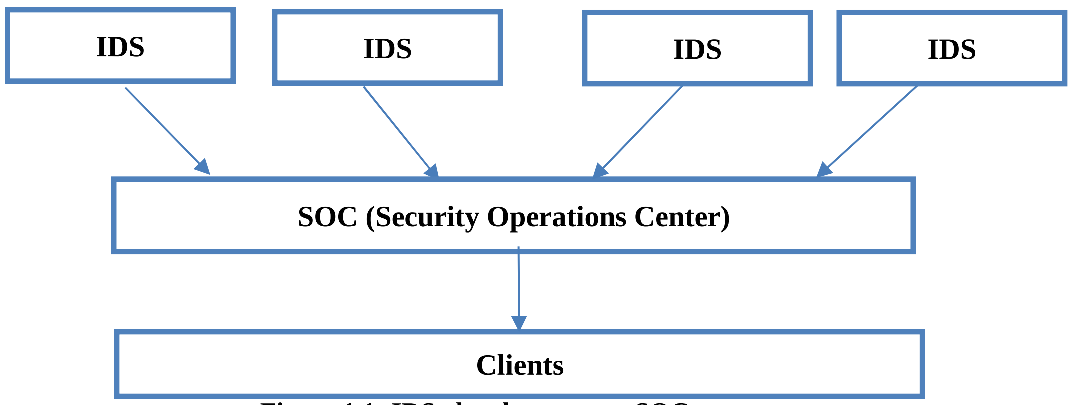


Figure 1.1: IDS alert logs sent to SOC.

1.2 Research Problem

Despite the growing popularity of SOCs and their essential role in improving organizations' security posture, according to a recent survey of 554 IT security practitioners involved in SOCs by the Ponemon Institute [17], there is a frustration among SOC customers about the ineffectiveness of current platforms in detecting attacks. According to the study, the mean time to resolution (MTTR) of security events can span several months. While 22% of the survey participants indicated that the resolution can take hours or days, 42% of the participants indicated that the MTTR can be months or years, which represents extremely long windows of vulnerability. Some of the causes for the low effectiveness of SOC are rooted in the low confidence in the ability to identify threats, the high-stress work environment for analysts and the lack of visibility in the network traffic and the underlying IT security infrastructure. Another key reason for the low effectiveness of SOC is the large amounts of indicator of compromise (IOCs) and false positives to handle, and the huge amount of internal traffic to consider by the analysts. The typical SOC receives on a regular basis massive amounts of diverse log data generated from IT environments that arrive at high velocity. For instance, Ślęzak et al. studied in [20] the case of a SOC where the growth in the observed monitored traffic was over 300 billion new events per month. The data is characterized by great amounts of false positives and false negatives. This represents a significant challenge to the current SOC business model which is heavily centered around human operators. While technological solutions such as log aggregators and SIEMs are heavily used in the initial layers of event processing, event prioritization and resolution are generally performed manually by human analysts. Such a model is becoming unsustainable with the fast-growing volume, velocity, and variety of IT infrastructure and security systems data.

According to the aforementioned survey by the Ponemon Institute, automation of the workflow is seen as the top solution (by 67% of respondents) to improve the quality of the SOC services. Also, the need for help in prioritizing incidents and tasks is seen as another top solution (by 49% of respondents) and this can also be addressed through automation. Our focus in the current thesis is on the latter concern by developing a new approach to assist in the automation of incidents prioritization.

The purpose of the research conducted in this thesis is to develop an effective method for analyzing automatically SOC data. We make a step forward such vision by identifying from the initial features describing the event log data, a set of new features using graph visualization and graph metrics. Specifically, we introduce a set of new features by using the graph metrics as the features themselves. We explore different classification techniques and show that the introduction of the new features improve noticeably the accuracy of the classifiers.

1.3 Approach Outline and Thesis Contributions

Our proposed approach consists of studying sample SOC data by deriving an underlying graph model and extracting a set of features based on graph metrics. The extracted features are analyzed using machine learning techniques. We make two main contributions as follows.

1. The first contribution of the thesis is the introduction of a new feature model based on graph metrics for SOC data classification. The graph metrics which we used are eccentricity,

diameter, radius, degree, center, periphery and sum of degrees. We show experimentally that the incorporation of these metrics in the feature model helps boost the accuracy of the machine learning models and helps them classify the suspicious and non-suspicious events in a better way compared to using only the original features and their derivatives.

2. The second contribution of the thesis is validating the potential of deep learning in improving the classification performance of SOC events compared with shallow learners. We investigated different linear and non-linear machine learning models. We started our experimentation with linear models such as logistic regression, but we were not able to achieve better results, so we switched to non-linear models such as XG Boost and Deep Neural Network (DNN) which helped us in achieving the best results.

The combination of graph-based features and deep learning provides an effective way to automate the process of identifying suspicious and non-suspicious events and reducing the manual work of the SOC analysts. Experimental evaluation based on a dataset collected by an existing SOC, yields for Deep Neural Network (DNN), the best performing of all algorithms, an area under curve (AUC) of 99%.

1.4 Thesis outline

The outline of the thesis is as follows:

Chapter 1 gives an outline of the context of the research, formulates the research problem and summarizes the thesis contributions.

Chapter 2 discusses the background knowledge and related works.

Chapter 3 presents the SOC dataset and related feature set.

Chapter 4 presents the proposed graph-based feature identification using graph visualization and graph metrics.

Chapter 5 presents the feature analysis and data preprocessing techniques used in the proposed framework.

Chapter 6 presents the experiments conducted to assess the impact of the derived features on performance and evaluate the F1 and area under curve of the proposed model.

Chapter 7 makes concluding remarks and discusses future work.

Chapter 2: Background and Related Work

The amount of published research work on SOC and graph-based feature engineering is limited. In this chapter, we summarize and discuss the few related works on SOC operation and graph-based feature engineering that we have come across in our review of the literature.

2.1 On SOC

Although there is a rich literature on SOC enabling techniques such as intrusion detection, and alert aggregation and correlation, the research on crucial aspects of the SOC workflow are still at a nascent stage. As a result, there is a limited number of papers published on improving SOC effectiveness and efficiency.

Ślęzak et al [20] discussed the complementarity between machine learning aimed at events classification and interactive analytics tools used to resolve uncertain cases by human analysts by considering the case of an SOC. The authors presented a solution to deal with the huge amount of data received and processed by analysts at SOCs using partially approximate queries based on information granulation and summary-based processing. The underlying rationale being to trade-off adequately the speed, accuracy and cost underlying the decision making involved in threat analysis.

Chandran et al. [6] conducted an anthropological study of three SOCs, two of which belongs to corporations and the third belong to a public university. A key outcome of the study is the emphasis placed on the human resource over technological resources (e.g. SIEM). While the technology still plays an important role in those SOCs, they tend to rely predominantly on human analysts in operations. The human analysts are considered the most critical drivers in the operational tasks. For some of the SOCs, there is a preference for technological solutions that support a workflow centered around the human intervention. There is a lack of confidence in the ability of existing SIEMs to perform some critical tasks such as event prioritization and alert verification. It is expected that the correlated events produced by the SIEM will have to be classified manually as true or false positive by a human analyst.

Jacobs, Arnab, & Irwin [9] developed a classification and rating scheme for an SOC based on its capability and maturity level. The goal of the proposal was to provide SOC's stakeholders a way to assess objectively the effectiveness and maturity of SOC services. The proposed model enables measuring SOC capability and maturity from three different perspectives: SOC services, SOC aspects and SOC processes per aspect.

Marty [14] presented a cloud logging framework and guidelines to provide a proactive approach to logging. The proposed approach leverages proactive information and system logging to make sure that the data needed for forensic investigation is available. Because forensic analysis is considered as the main component of a SOC, it is limited to post-event and reverse engineering analysis.

Alruwaili and Gulliver [4] introduced a SOC as a service framework for cloud computing with a focus on security and regulatory compliance. The framework relies on the aggregation of events and logs collected from security devices and systems deployed on cloud infrastructure.

Miloslavskaya [13] highlighted the highly heterogeneous and huge amounts of data handled by SOC and the fact that this was unsustainable by the typical headcount in the current SOCs, especially considering the over-reliance on manual processing of security events by human analysts. The author also highlighted the limitations of traditional SOCs which depend on rule-based SIEMs. While such systems perform relatively well in detecting conventional attacks, they are ineffective when confronted with emerging attacks such as advanced persistent threats (APT), which use targeted and stealthier techniques to evade detection. This leads to a great number of false positives and false negatives, which are exacerbated by a huge volume of data arriving in the pipeline.

Aijaz et al. [1] highlighted the importance of the threats faced by academic institutions and emphasized the need for these organizations for establishing SOC as a way to improve their security posture.

Van Niekerk and Jacobs [23] discussed the suitability of security-as-a-service for critical cloud information infrastructure services. They suggested a model to integrate traditional security solutions into a cloud infrastructure. The proposed model provides the high-level description and details for implementing a SOC in the cloud infrastructure.

2.2 On Graph-based Feature Engineering

To our knowledge, only a limited number of works have been published on graph-based feature engineering. We summarize those works in the following.

Heim et al. [7] proposed to extend traditional requirements analysis and management by a graph-based visualization that allows representing multidimensional relations in a flexible way. In particular, the authors proposed a special representation form that enables the exploration of requirements along with their relationships and facilitates the understanding of dependencies between requirements by making use of the graph-based visualization. In our work we leverage graph-based visualization to extract machine learning features from the dataset.

Van Ham and Wattenberg [24] described the solution to the problem of the small graphs with low diameter; the existing techniques break down visually even when the graph has only a few hundred nodes. The authors used a global edge metric technique to determine a subset of edges that captures the graph's intrinsic clustering structure. This structure is then used to create an embedding of the graph, after which the remaining edges are added back in.

Polzlbauer et al [18] presented a novel technique consisting of using the Self-Organizing Map for data analysis by taking the density of data into account. The authors defined the graphs resulting from the nearest neighbor and radius-based distance calculations in data space and showed projections of these graph structures on the map. In our work, we set one node as the source node and considered all the other nodes as the destination nodes and then calculated the graph metrics based on this technique.

Cui and Qu [25] focused on various visualization techniques for massive graphs. As the data has become very large nowadays, traditional graph visualization techniques fail to reveal the pattern hidden in the data and massive data pose many challenges for graph visualization, such as visual clutter, layout and evaluation criteria. The authors described the clutter reduction algorithms and user navigation techniques to use the graphs in many applications, such as social networks and Internet communications. We also used graph visualization technique to find the patterns in our dataset, e.g. on which day the maximum attacks occurred, or which network the notified and non-notified attacks occurred.

Wang and Chang [26] proposed a neural graph-based dependency parsing model which utilizes hierarchical Long short-term memory (LSTM) networks on character level and word level to learn word representations, allowing the model to avoid the problem of the limited-vocabulary and capture both distributional and compositional semantic information. The model shows effectiveness in recovering dependencies involving out-of-vocabulary words.

Atzmueller and Sternberg [5] proposed a mixed-initiative feature engineering approach using the explicit knowledge captured in a knowledge graph complemented by a novel interactive visualization method. Using the explicitly captured relations and dependencies between concepts and their properties, feature engineering is enabled in a semi-automatic way. The results obtained throughout the process can be utilized for refining the features and the knowledge graph.

Nguyen et al. [16] presented a novel neural network model that learns the part of speech (POS) tagging and graph-based dependency parsing jointly. The model uses bidirectional LSTM to learn feature representations shared for POS tagging and dependency parsing tasks, and with this, it handles the feature-engineering problem. In their extensive experiments, they have worked on 19 languages from the Universal Dependencies project, showing that the model outperforms the state-of-the-art neural network-based stack propagation model for joint POS tagging and the transition-based dependency parsing, resulting in a new state of the art.

2.3 Summary

In this chapter, we summarized related work on SOC and on graph-based feature engineering. It is clear from the reviewed research that only a limited amount of work has been published on SOC event classification. The graph-based techniques have been used on the LSTM neural network to learn word representations; some authors have explored pattern recognition in the graph and captured the relations and dependencies between concepts and properties of the graph. In our research we have explored for the first time the usage of the graph metrics as machine learning features.

We introduced a deep learning-based approach for automating the process of identifying the suspiciousness of the alerts generated by the IDS using two new sets of features: graph metrics as features and features based on graph-based visualization. The benefit of using the aforementioned technique is improved accuracy of the machine learning classifier.

Chapter 3: The SOD Dataset and Feature Set

In this chapter, we present the SOC dataset used in our experimental evaluation. We present the initial features used to describe the security events in the dataset and derive a group of features by converting categorical features and merging some of the initial features. We use the aforementioned features as baseline against which we can assess the effectiveness of the graph-based features introduced in the next chapter.

3.1 Dataset Overview and Initial Feature Set

The dataset used in our evaluation consists of event log data from Security On-Demand (SOD) [20], which is a company that runs an SOC on behalf of various customers. The data has been sanitized by obfuscating the security and privacy sensitive information. The dataset was released in two phases as part of the IEEE BigData 2019 Suspicious Network Event Recognition track. The dataset is a labeled dataset consisting of security events and their classification by security analysts. The classification is either 'notified' or 'non-notified', which corresponds to whether or not the customer was notified about the alert.

Feature	Description	Feature Domain
weekday	A day of the week of the first log event that is assumed to be the first event that corresponds to the alert.	Mon, Tue, Wed, Thurs, Fri, Sat, Sun
overallseverity	Alert severity generated by the system rules.	1, 2, 3, 4, 5
reportingdevice_cd	Device that reported the event.	0, 1, 2, 3, 4, 5, …, 30
devicetype_cd	Reporting device type.	0, 1, 2, 3, 4, 5
devicevendor_cd	Vendor that produced the reporting device.	0, 1, 2, 3, 4, 7
categoryname	A category name of the alert that corresponds to its severity.	Attack, Exploit, Malicious Activity, Suspicious Network Activity, Suspicious Attack Activity

Table 3.1: Sample Original Features from the SOD dataset.

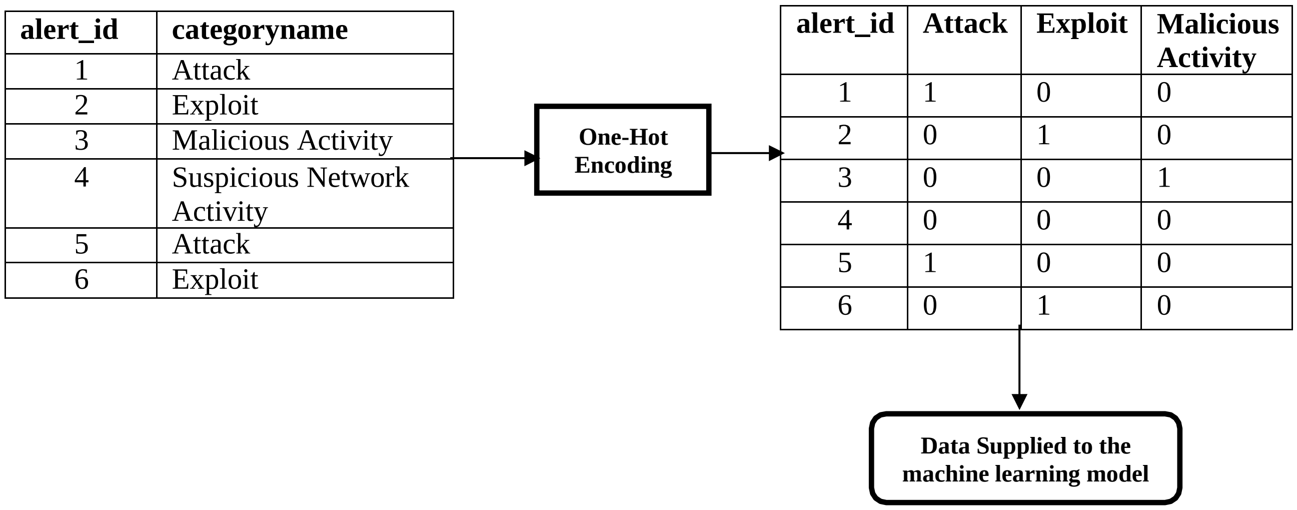
The initial SOD dataset consists of a total number of 39,427 records out of which 37,151 records were not being notified to the client as the SOC team did not find them suspicious (class 0) and 2, 276 records were notified to the client as the SOC team found those records suspicious (class 1). This dataset is a skewed dataset as the number of observations that belong to one class is higher than the other class.

The second SOD dataset which consists of about 430 GB of event log data is similar to the previous one, split into training and testing data. The test data was missing the notified values (i.e. the label). Each record in the dataset is described by 62 original features. Table 3.1 presents sample features from the original dataset. We have used the initial dataset for the experimentation in the current thesis.

3.2 Features Conversion

3.2.1 Converting Categorical Features

As mentioned above, the original feature set in the SOD consists of 62 original features; 50 among these features are categorical features. We started the data analysis by converting the categorical features into numerical features by using the one-hot encoding technique. For example, Figure 3.1 illustrates one-hot encoding using *categoryname*, which is a categorical feature from the original feature set.



alert_id	categoryname
1	Attack
2	Exploit
3	Malicious Activity
4	Suspicious Network Activity
5	Attack
6	Exploit

alert_id	Attack	Exploit	Malicious Activity
1	1	0	0
2	0	1	0
3	0	0	1
4	0	0	0
5	1	0	0
6	0	1	0

Figure 3.1: Applying one-hot encoding to the categoryname feature

This feature describes a category name for the alert that corresponds to its severity. It takes the following values: *Attack, Exploit, Malicious Activity, Suspicious Network Activity* and *Suspicious Attack Activity*. When one-hot encoding is applied to this feature it will create the features with the values of the categoryname column and will assign value ‘1’ to its corresponding column and ‘0’ to others.

After applying one-hot encoding, we handle the missing values in the dataset by taking the median of the numerical values and the mode of the categorical values, and then replace the missing values with the median and mode values, respectively.

The next step in the analysis is to normalize the data and handle outliers. In our work, this is done using Z-score normalization, which is also known as mean normalization and consists of transforming the data in such a way that mean equates to 0 and standard deviation varies according to the data points.

3.2.2 Feature Conversion through merging

We identified several new features by merging some existing features with common characteristics. Some of the initial features in the SOD dataset carry related or common knowledge although such relation cannot necessarily be exposed through statistical correlation. By reviewing the original features, we identified two groups of features that meet the aforementioned criteria: time-based features and IP related features, named Time and IP_Type, respectively, and obtained by merging the features in each of the separate groups.

IP_Type: IP information is an important component of the security event data as this provides knowledge about the participants and locations involved in the underlying activity. In the SOD dataset there are two columns named ipcategory_name and ipcategory_scope. These two columns are dependent on the IP column and their information can be gathered from the IP column itself. So, we decided to merge these two columns and come up with a new column 'IP_Type' which describes what the IP address type is.

IP	IP_TYPE	DESCRIPTION
AB.AB.01.01	II	Internet Internet
192.AB.AB.02	PP	Private Network
169.KU.AZ.103	LS	Link-local Subnet
255.PQ.CZ.255	BS	Broadcast Subnet
127.JI.PF.1/255	LH	Loopback Host
ZC.WB.188.1	MI	Multimedia Internet

Table 3.2: Representing IP_TYPE feature value.

Table 3.2 represents the new IP_Type feature values; sample IP values, the categorical values and their descriptions are provided in the table. For instance, IP_Type feature value II stands for *Internet Internet*. This new feature merges the two columns ipcategory_name and ipcategory_scope from the original SOD dataset.

Time: In the SOD dataset there are three columns named start_hour, start_minute and start_second. Start_hour represents an hour of the first log event, start_minute represents a minute of the first log event and start_second represents a second of the first log event. Basically, these represent the time, so we decided to remove all these three features and come up with a new feature *Time (sec)*. The new feature *Time (sec)* converts the time in seconds. For instance, if the start_hour value is 8, start_minute value is 14 and start_second value is 32, the absolute time is 8:14:32; when we will store the values in our new column, we will convert the given time in seconds.

3.3 Summary

In this chapter, we presented the SOC dataset used in our research and the original features used to describe the corresponding security events. One Hot encoding has been applied to all the categorical features so that those features make sense to machine learning models. Specifically, we have derived two more features from the original features, i.e. IP_Type and Time, respectively. Through the above transformation, we end up in total with 41 features including 39 original features and 2 derivatives. We will use these original features and their derivatives as our baseline feature space to assess the effectiveness of the new graph-based features that will be introduced in the next chapter.

Chapter 4: Graph Based Feature Identification

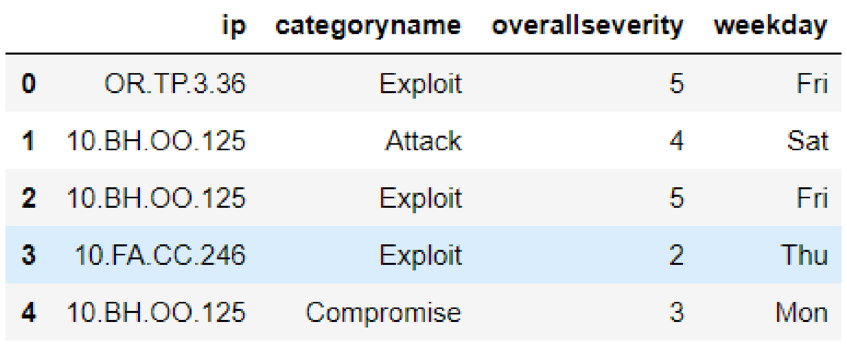
In this chapter, we present a new approach for extracting and deriving machine learning features using a graphical representation of the data. The approach consists of generating a graphical representation of the training data and identifying salient characteristics and relationships and converting those into new features. Two main types of features are extracted: features derived from the graph structure through visualization and features based on standard graph metrics. We illustrate in more detail the proposed approach by using the SOD dataset in the following.

4.1 Proposed Graph Model

Our feature derivation approach assumes that a dataset is available, such as a training set. Let's assume that the dataset consists of *m* records, each described by *n* different feature types. Let assume that one of the feature types plays the most pivotal role in the problem domain captured by the dataset. Our proposed graph model consists of a knowledge graph describing the unique feature values from the dataset and their relationships. In other words, each unique feature value will be represented as a node in the graph. We refer to the nodes corresponding to the aforementioned most pivotal feature type as pivotal nodes. Obviously, each record will have a unique pivotal node.

Given a record $R_i = [x_{i1},..,x_{ik},...,x_{im}]$ from the dataset, let us assume that x_{ik} is the pivotal node. The edges in the knowledge graph will be drawn by connecting the pivotal node as source to the other nodes involved in the record, giving a sequence of edges $e^i_{kj} = (x_{ik}, x_{ij})$, where $k \neq j$. Note that there is no edge from a pivotal node to itself. So, each record R_i will induce *n-1* edges.

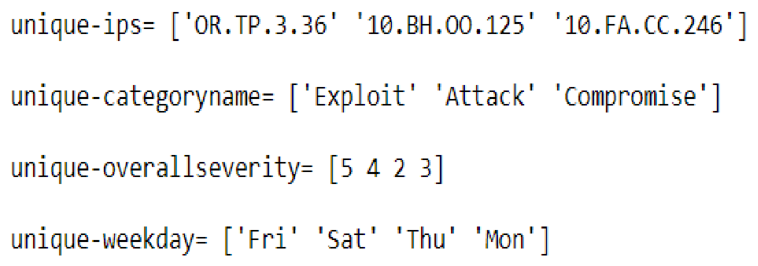
To illustrate the graph construction process, we consider a small subset of the SOD dataset containing a few records and a subset of the features as shown in Table 4.1.



	ip	categoryname	overallseverity	weekday
0	OR.TP.3.36	Exploit	5	Fri
1	10.BH.OO.125	Attack	4	Sat
2	10.BH.OO.125	Exploit	5	Fri
3	10.FA.CC.246	Exploit	2	Thu
4	10.BH.OO.125	Compromise	3	Mon

Table 4.1: Sample records and feature subsets

As it can be observed, we have duplicate values for all the features. So, the first thing to do is taking the unique values of all the features. Figure 4.1 depicts the unique values of all the features from the sample dataset. In total, we have 14 unique values, so the graph will have 14 vertices/nodes.



```
unique-ips= ['OR.TP.3.36' '10.BH.OO.125' '10.FA.CC.246']

unique-categoryname= ['Exploit' 'Attack' 'Compromise']

unique-overallseverity= [5 4 2 3]

unique-weekday= ['Fri' 'Sat' 'Thu' 'Mon']
```

Figure 4.1: Unique feature values from the sample records (in Table 4.1)

Figure 4.2 shows the knowledge graph derived from the sample records based on the model described above. The most critical feature used to describe security events is the IP address. So, in the SOD dataset and by extension in the sample records, the pivotal nodes correspond to IP addresses.

The sample graph is built by considering the unique IP addresses as the source nodes and all the other nodes as the destination nodes. In the above example, we have 3 unique IPs, so we have 3 source nodes, i.e. OR.TP.3.36, 10.BH.OO.125 and 10.FA.CC.246.

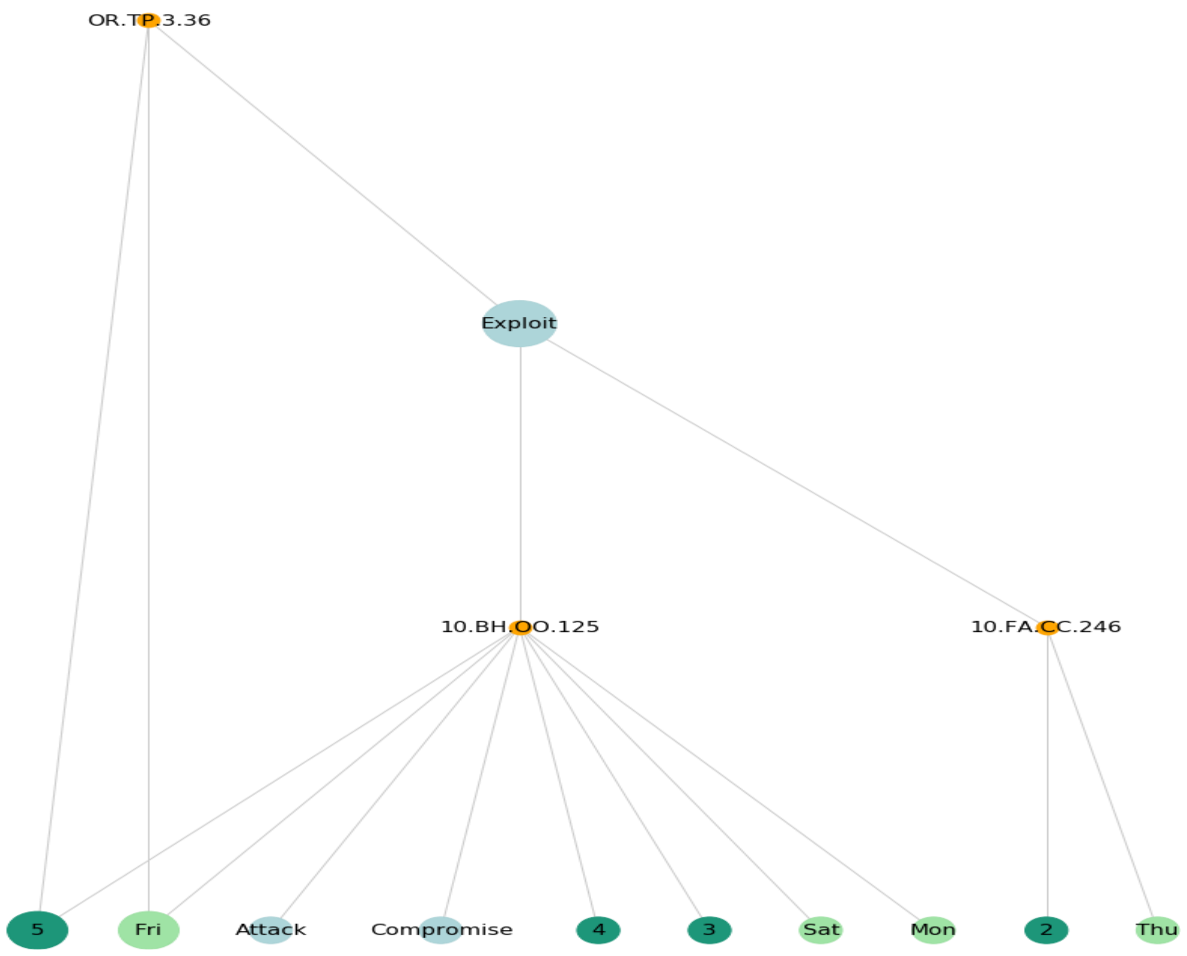


Figure 4.2: Knowledge graph derived from the sample records (in Table 4.1)

In the example dataset, we can see in record number 0 that IP OR.TP.3.36 is associated with Exploit, 5 and Fri. This is translated in the graph into 3 edges connecting OP.TP.3.36 to Exploit, 5 and Fri, respectively.

Records numbers 1 and 2 share the same IP address, i.e. 10.BH.OO.125. In the graph, this results in 9 edges coming from 10.BH.OO.125 to Attack, 4, Sat, Exploit, 5, Fri, Compromise, 3 and Mon.

In the graph we have orange dots on the IP. We use this notation to express the fact that a particular IP has a degree greater than 1. The degree represents the number of edges incident to the vertex and as we can see all the 3 IPs have more than one edge coming out of it; so, we have orange dots on all the 3 IPs.

4.2 Features based on Graph Metrics

Here, the main idea is to use graph metrics as features. To illustrate and validate this concept, we consider 6 graph metrics as described in the following.

4.2.1 Graph Metrics

Given a training set, our approach consists of building separate knowledge graphs for negative and positive samples and deriving the metrics values for each of the separate graphs. While there is a wide variety of graph metrics that could represent good candidates for defining our suggested features, we will illustrate our approach using a subset consisting of six graph metrics described in the following.

Eccentricity: It is the maximum of the distances between one vertex to all other vertices.

For instance, Table 4.2 shows the eccentricity values for the nodes of the sample graph depicted in Figure 4.2.

Node-name	**Eccentricity**
OR.TP.3.36	3
Exploit	2
10.BH.OO.125	3
Attack	4
10.FA.CC.246	3
Compromise	4
5	4
4	4
2	4
3	4
Fri	4
Sat	4
Thu	4
Mon	4

Table 4.2: Eccentricity measures obtained from the graph shown in Figure 4.2

Diameterofrow: The diameter is the maximum value of the eccentricity for all the vertices. We define a new feature based on the diameter that we call *diameterofrow*. This feature is obtained by taking the maximum value of the eccentricity among the vertices generated from a given record $R_i = [x_{i1},..,x_{ik},..., x_{im}]$.

Radiusofrow: The radius is the minimum value of the eccentricity for all the vertices. Based on the radius we define a new feature called *radiusofrow*. This feature is obtained by taking the minimum value of the eccentricity among the vertices generated from a given record $R_i = [x_{i1},..,x_{ik},..., x_{im}]$.

Centerinrow: The center of a graph is the set of vertices for which the eccentricity is equal to the radius of the graph. These vertices are called central points. We define a new feature called *centerinrow*, which is obtained by identifying the central points among the set of vertices generated from a given record $R_i = [x_{i1},..,x_{ik},..., x_{im}]$, and then from those vertices we select the vertex whose feature holds the highest importance based on some preset feature importance metrics. In other words, the value of *centerinrow* will be equal to the feature value associated with the selected vertex.

Peripheryinrow: The periphery of a graph is the set of vertices for which the eccentricity is equal to the graph diameter. We define a new feature called *peripheryinrow*, which is obtained by first identifying the vertices that have graph eccentricities equal to the graph diameter among the set of vertices generated from a given record $R_i = [x_{i1},..,x_{ik},..., x_{im}]$. Then from the identified vertices, we select the vertex whose feature holds the highest importance based on some preset feature importance metrics and assigns its value to *peripheryinrow*.

Degreeinrow: The degree is the number of edges that are incident to a vertex. Based on the degree we define a new feature called *degreeinrow*. This feature is obtained by calculating the degree row wise, i.e. for all the vertices induced by a given record $R_i = [x_{i1},..,x_{ik},...,x_{im}]$ and then taking the maximum value in the row as the feature value. For example, Table 4.3 shows the degrees for the sample graph shown in Figure 4.2.

Node-name	**Degree**
OR.TP.3.36	3
Exploit	3
10.BH.OO.125	9
Attack	1
10.FA.CC.246	3
Compromise	1
5	2
4	1
2	1
3	1
Fri	2
Sat	1
Thu	1
Mon	1

Table 4.3: Degree measures obtained from the graph shown in Figure 4.2

Sumofdegreesinrow: Based on the sum of degrees of vertices, we define a new feature called *sumofdegreesinrow*. The feature is obtained by calculating the sum of degrees of vertices row wise, i.e. for all the vertices associated with a given record $R_i = [x_{i1},..,x_{ik},...,x_{im}]$.

4.2.2 Graph Structures

In the SOC dataset, the positive samples correspond to the notified class while the negative samples correspond to the non-notified class. As the number of data points for class 1 (i.e. positive samples) are less as compared to class 0 (i.e. negative samples), we decided to build a single connected graph for class 1 and multiple connected graphs for class 0.

For class 0, the data points were divided based on the time scale. In our SOC dataset we introduce a new column *time(sec)*, which describes the time in seconds. Based on this column we define a time window of 18,000 seconds (i.e. 5 hours) and all the data points which are falling in the first 18,000 seconds time window are used to build the first graph and the data points which are falling in the next 18000 seconds are used to build the next graph and so on, as shown in figure 4.3. In this way we calculate the graph metrics for class 0.

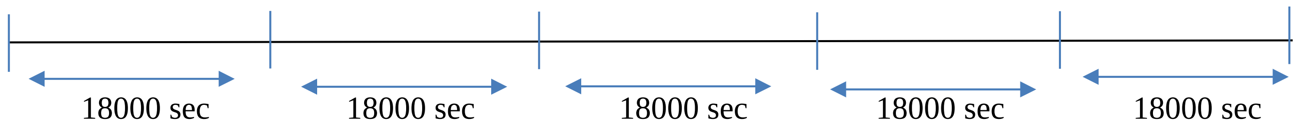


Figure 4.3: Sliding time windows

Figure 4.4 depicts the graph generated for a subset of the records of class 1 from the SOD dataset. The graph is built by considering a subset of the initial features from the dataset to give a clearer picture. All the green nodes represent IP_Type, blue nodes represent weekdays and the other nodes represent categoryname and overallseverity.

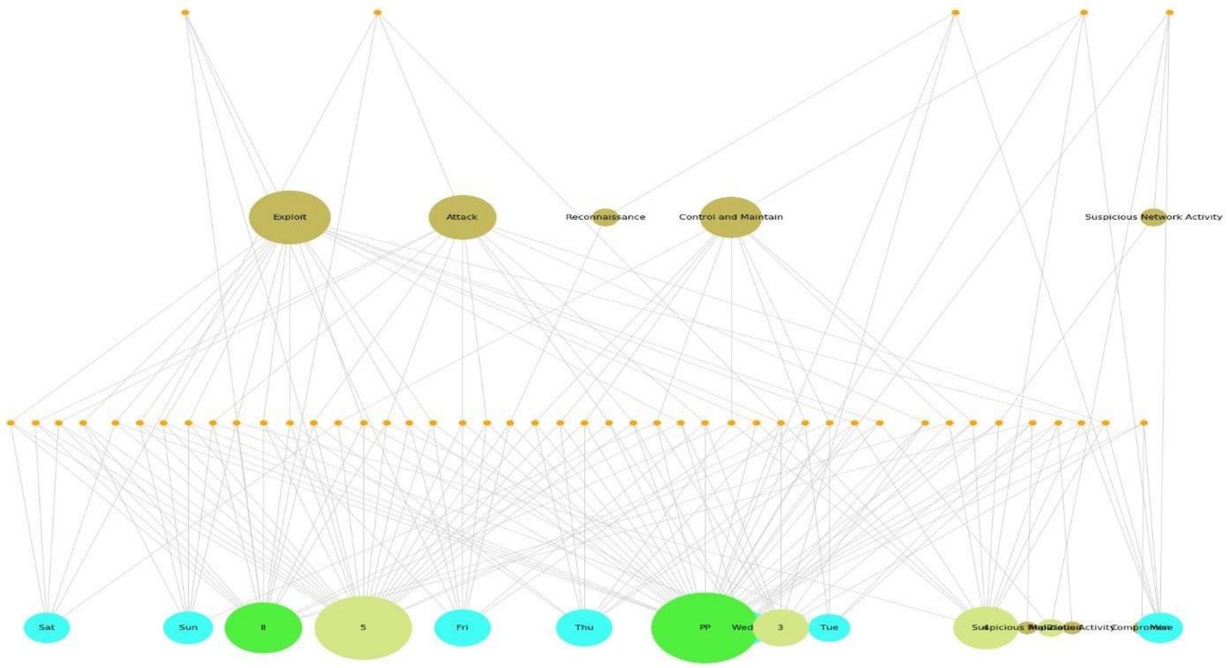
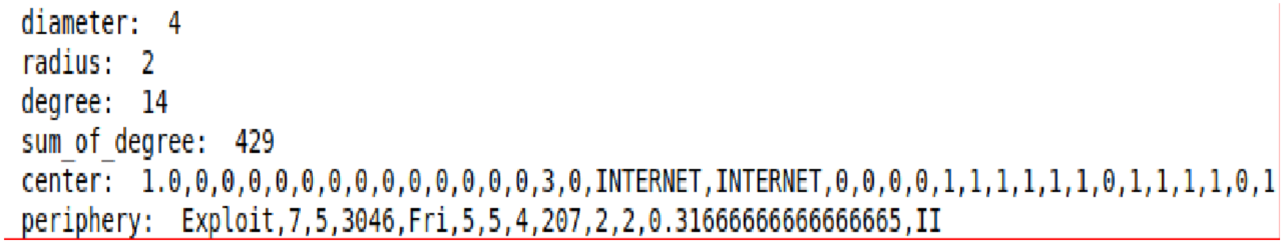


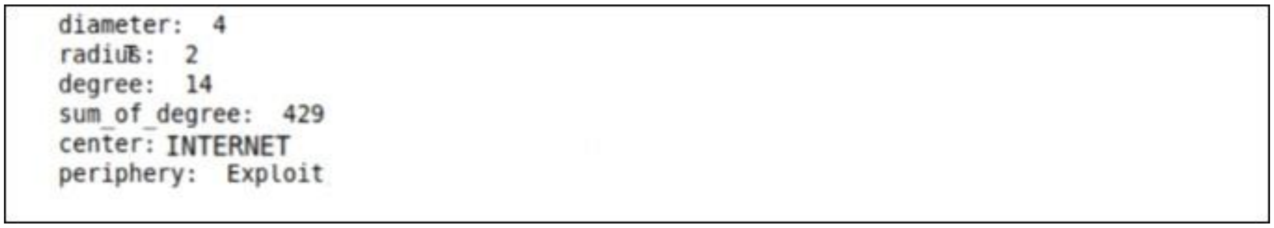
Figure 4.4: Graph representing the incidents for class 1 with 19 vertices for a subset of the SOD dataset

Figure 4.5 shows the graph metrics calculated for one of the records representing class 1. For centrality and periphery, we have more than 1 value; from all these values we select the value which holds higher importance in the feature importance graph. As shown in Figure 4.6, for *center* we select INTERNET value which belongs to the feature srcipcategory_dominate which has higher feature importance compared to the other features, and for *periphery* we select EXPLOIT value, which belongs to the feature categoryname, which holds more importance than other features.



```
diameter:  4
radius:  2
degree:  14
sum_of_degree:  429
center:  1.0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,INTERNET,INTERNET,0,0,0,0,1,1,1,1,1,1,0,1,1,1,1,0,1
periphery:  Exploit,7,5,3046,Fri,5,5,4,207,2,2,0.31666666666666665,II
```

Figure 4.5: Sample graph metrics



```
diameter:  4
radius:  2
degree:  14
sum_of_degree:  429
center: INTERNET
periphery:  Exploit
```

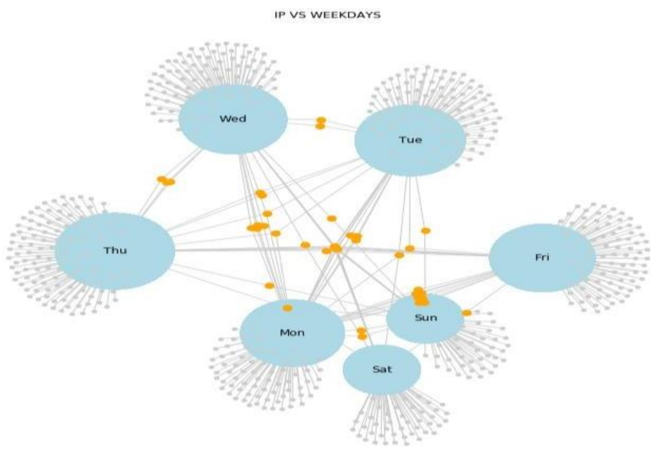
Figure 4.6: Graph metrics calculated for class 1 for a single row.

4.3 Features Derivation from Graph Visualization

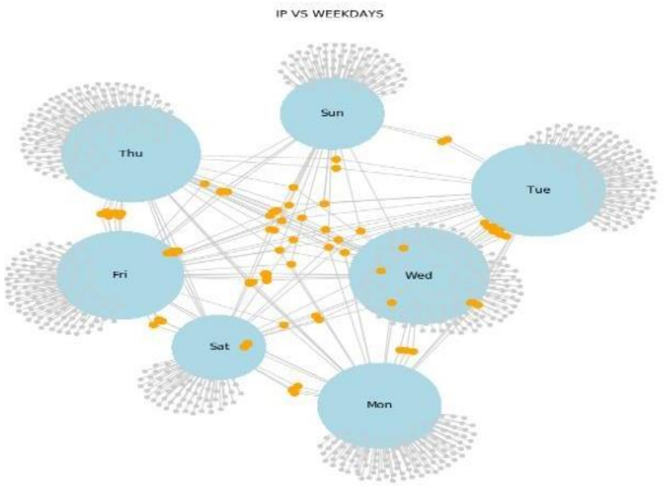
The proposed approach consists of visualizing the security events from the training dataset by highlighting their spread based on different features, and then analyzing the underlying characteristics and relationships.

Using the SOD dataset, by constructing various graphs based on different features combination, the major observation from the analysis is that the notified attacks commonly occurred on private network whereas all the non-notified attacks occurred on the public network with overall severity value as 5 and on weekdays. Based on this observation, we introduce 10 new features, which are described in the following.

Criticalityofweekday: Figure 4.7 shows a graphical depiction of IP vs. weekday for notified and non-notified events in the SOD dataset.



(a) Weekday graph for class 0

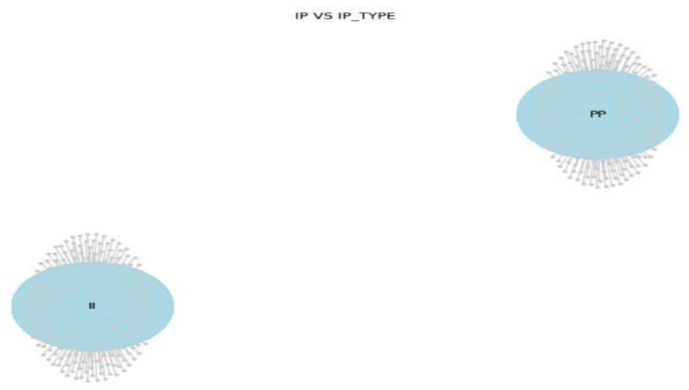


(b) Weekday graph for class 1

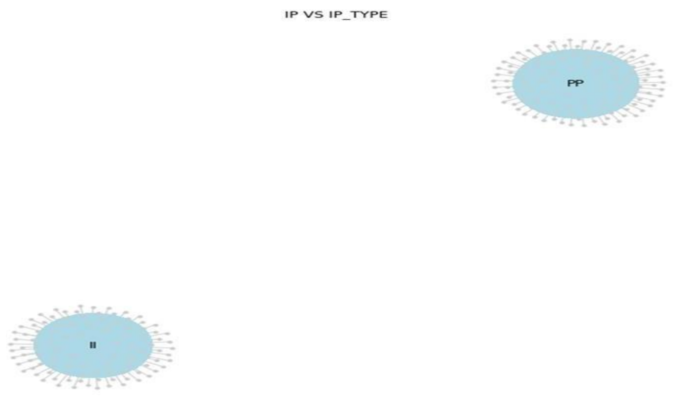
Figure 4.7: Graphical representation of IP vs. weekday in the SOD dataset.

The orange points in the graph represent the IPs which occurred more than once, specifying that alerts from the same IP were captured on two or three different days and this holds a degree greater than 1; these IPs are the popular IPs. The analysis shows that most of the events in both classes (i.e. notified and non-notified) occurred on weekdays and not on weekends. For the notified class, the maximum number of alerts occurred on Wednesday and for non-notified class, the maximum number of alerts occurred on Thursday. So, based on this observation, we define the Criticalityofweekday feature as a binary feature, which takes value 1 where the original SOD weekday corresponds to Mon, Tue, Wed, Thurs or Fri and value 0 where the weekday corresponds to Sat or Sun.

DomainofIP_typeforclass1: As mentioned earlier, we have introduced a new feature called IP_Type, which represents the name and the scope of the IP. Figure 4.8 shows a graphical depiction of the IP vs. IP_Type for non-notified and notified events in the SOD dataset. As we can see in the graph there are no orange points; this is because one particular IP belonged to just one network either private, internet or subnet. The analysis of Figure 4.8 (b) shows that the maximum number of alerts which were notified (class 1) by the SOC analysts occurred on the private network which is denoted as PP. It can be noted also that most of the attacks occurred either on the Internet (II) or on the private network (PP). So, based on this observation, we introduce a new feature called domainofIP_typeforclass1 which takes value 1 where the value in the IP_Type column corresponds to PP and value 0 for all the other values in IP_Type column.



(a) IP_Type graph for class 0

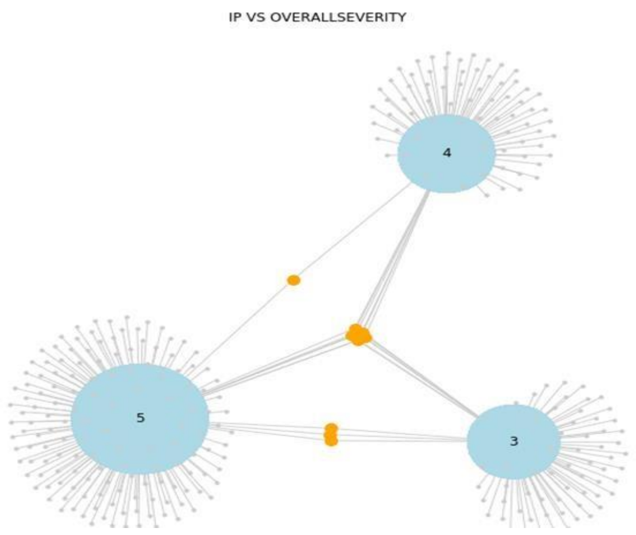


(b) IP_Type graph for class 1

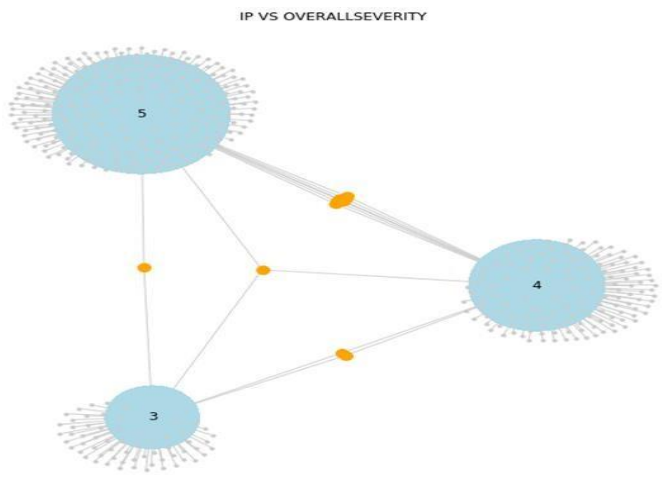
Figure 4.8: Graphical representation of IP vs. IP_Type in the SOD dataset.

DomainofIP_typeforclass0: The analysis of Figure 4.8 (a) shows that the maximum number of alerts that were not notified (class 0) by the SOC analysts occurred on the public network i.e. Internet which is represented as II. So, based on this observation, we introduce a new feature called domainofIP_typeforclass0 which takes value 1 where the value in the IP_Type column corresponds to II and value 0 for all the other values in IP_Type column.

Severityofalert: One of the original features in the SOD dataset is a feature named overallseverity, which represents an estimation of the alert severity.



(a) Overallseverity graph for class 0



(b) Overallseverity graph for class 1

Figure 4.9: Graphical representation of IP vs. overallseverity in the SOD dataset

Figure 4.9 shows a graphical depiction of the IP vs. overallseverity in the SOD dataset. The orange points in the graph represent the IPs that occurred more than once, specifying that alerts from the same IP were captured with different values of overall severity. The analysis shows that for both the notified (class 1) and non-notified (class 0) alerts, the maximum number of alerts generated was with overall severity 5. So, based on this observation, we introduce a new feature called severityofalert which takes value 1 where the value in the overallseverity column corresponds to 5 and value 0 for all the other values.

Using a similar approach, several new features can be identified. Three additional features that were considered in our model include the CriticalityofIPAddress, CriticalityofSrcIP and CriticalityofDestIP defined as follows.

CriticalityofIPAddress: enables distinguishing between the IPs which occurred more often from the ones which occurred just once or twice. We set a threshold of 100: the IPs whose occurrence is above the threshold were given the value 1 and the IPs occurring below the threshold were given the value 0.

CriticalityofSrcIP: enables distinguishing source IP addresses that occur more frequently compared to others. Source IP addresses that occur with high frequency could be, for instance, spoofed IP addresses used in volumetric attacks such as flooding denial of service (DOS) attacks. We set a threshold of 100: if one destination IP is occurring more than 100 times it is given a value of 1 and if it is occurring less than the threshold value then it is given a value of 0.

CriticalityofDestIP: enables distinguishing destination IP addresses that occur more frequently compared to others. If one destination IP is occurring very often this means that the IP belongs to some crucial site and hackers are hitting on it to hack it. We set a threshold of 100: if one destination IP is occurring more than 100 times it is given a value of 1, otherwise it is given a value of 0.

4.3 Summary

This chapter introduces our proposed graph-based feature identification approach and corresponding feature model. Specifically, we showed how features can be derived on the one hand from graph visualization and on the other hand based on graph metrics. In total, 13 new graph-based features have been identified. The proposed features will help the machine learning models achieve the improved classification of SOC events. In the next chapter we discuss about the feature selection and data preprocessing techniques used in our framework.

Chapter 5: Feature Analysis

Exploratory data analysis (EDA) consists of analyzing the datasets to summarize their important characteristics. EDA helps in exploring the data to gain important insights about the data, which further helps in building a better machine learning model. In this chapter, we present the different techniques used to analyze the features identified in our approach and process the data involved.

5.1 Feature Selection

Our global feature space consists of two groups of features:

1. Original features and derivatives
2. Graph-based features

Number	Columns	Type
1	categoryname	categorical
2	parent_category	categorical
3	overallseverity	categorical
4	timestamp_dist	numerical
5	weekday	categorical
6	correlatedcount	numerical
7	score	categorical
8	srcip_cd	numerical
9	dstip_cd	numerical
10	srcport_cd	numerical
11	dstport_cd	numerical
12	alerttype_cd	categorical
13	eventname_cd	categorical
14	severity_cd	categorical
15	reportingdevice_cd	categorical
16	protocol_cd	categorical
17	username_cd	categorical
18	srcipcategory_cd	categorical
19	dstipcategory_cd	categorical
20	isiptrusted	categorical
21	untrustscore	categorical
22	flowscore	categorical

23	trustscore	categorical
24	enforcementscore	categorical
25	dstipcategory_dominate	categorical
26	srcipcategory_dominate	categorical
27	dstportcategory_dominate	categorical
28	srcportcategory_dominate	categorical
29	thrcnt_month	numerical
30	thrcnt_week	numerical
31	thrcnt_day	numerical
32	p6	categorical
33	p9	categorical
34	p5m	categorical
35	p5w	categorical
36	p5d	categorical
37	p8m	categorical
38	p8w	categorical
39	p8d	categorical
40	IP_Type	categorical
41	Time (sec)	numerical
42	alert_ids	categorical
43	client_code	categorical
44	grandparent_category	categorical
45	domain_cd	categorical
46	direction_cd	categorical
47	n1	categorical
48	n2	categorical
49	n3	categorical
50	n4	categorical
51	n5	categorical
52	n6	categorical
53	n7	categorical
54	n8	categorical
55	n9	categorical
56	n10	categorical
57	devicetype_cd	categorical
58	devicevendor_cd	categorical
59	start_hour	numerical
60	start_minute	numerical
61	start_second	numerical
62	IP	categorical
63	ipcategory_name	categorical
64	ipcategory_scope	categorical

Table 5.1: Original Features and their derivatives.

Number	Columns	Type
1	Criticalityofweekday	categorical
2	DomainofIP_typeforclass1	categorical
3	DomainofIP_typeforclass0	categorical
4	Severityofalert	categorical
5	CriticalityofIPAddress	categorical
6	CriticalityofSrcIP	categorical
7	CriticalityofDestIP	categorical
8	Diameterofrow	numerical
9	Radiusofrow	numerical
10	Centralinrow	categorical
11	Peripheryinrow	categorical
12	Degreeinrow	numerical
13	Sumofdegreesinrow	numerical

Table 5.2: Graph-based Features.

Tables 5.1 and 5.2 lists the original features and derivatives and the graph-based features, respectively. To evaluate the effectiveness of the graph-based features, we will apply in the next chapter different classification techniques to the first group of features and to the global features set including all the features, and then compare the corresponding classification performances. Before applying the classifiers to the different models, we apply feature selection techniques which consist of automatically or manually identifying the most important features and removing the ones that do not contribute much to the model's effectiveness.

The features that we use to train a machine learning model have a huge impact on its performance, so it is necessary to discard the irrelevant features and keep the best features. Discarding insignificant data helps improve accuracy, avoid unnecessary resource allocation and reduces the time taken to train the model. Filter, wrapper and embedded techniques are three different methods available for feature selection.

Filter methods which are also known as univariant selection methods, apply statistical measures to all available features, and assign scores to them based on the target class. Important features are assigned high score values and unimportant features are assigned low score values.

Examples of filter selection methods include the Chi-square test and Linear Discriminant Analysis (LDA). The wrapper methods are computationally expensive compared to the filter methods. In a wrapper method a model is trained using a subset of features and then based on the conclusions of the previous model we add or remove the features from that subset. Examples of wrapper methods include forward selection, backward elimination and recursive feature elimination. In the forward selection method, we start with having no feature in our model and then we keep on adding the features until it is improving the performance of the model. In the backward elimination method, we start with having all the features and keep removing the least significant feature until this is impacting the performance of the model. In the recursive feature elimination method, unimportant features are discarded one by one recursively. Embedded methods learn the best features of the model while the model is being created. Examples of embedded methods include regularization methods and the use of drop out layers [3].

In the current thesis, we use a filter method while assigning scores to the features in a feed forward neural network and later while training the model we use an embedded method to avoid overfitting. As the filter method we use a Chi-square test.

In statistics, the Chi-square test is applied to test the independence of two variables. Chi square test is applicable only for categorical or nominal data. We calculate the Chi-Square statistics between each feature variable and a target variable and assess the relationship between them.

The formula of Chi-square statistic is given as:

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

Where O represents the observed value and E represents the expected value. As per the equation, the Chi-square statistic is based on the difference between what is observed in the data and what would be expected if there is no relationship between the feature and the target. A very small chi square value means observed data fits expected data very well. In this case, there is a relationship. A very large Chi-square value means that the data do not fit well, so there is no relationship. All features are assigned a score after calculation of the Chi-square statistics between every feature variable and the target variable. We discard features with low scores and features with high scores are marked as important and selected to train a machine learning model.

Figure 5.1 shows the feature importance graph for the original features and their derivatives, which has been built using tree-based classifier. Feature importance assigns a score to each feature in the dataset: the higher the score the more important or relevant that feature is. Based on the feature importance computation, we discarded all the features with importance equal zero and kept the remaining features. Specifically, we kept in this first group, 41 features out of 64 features as listed in Table 5.1.

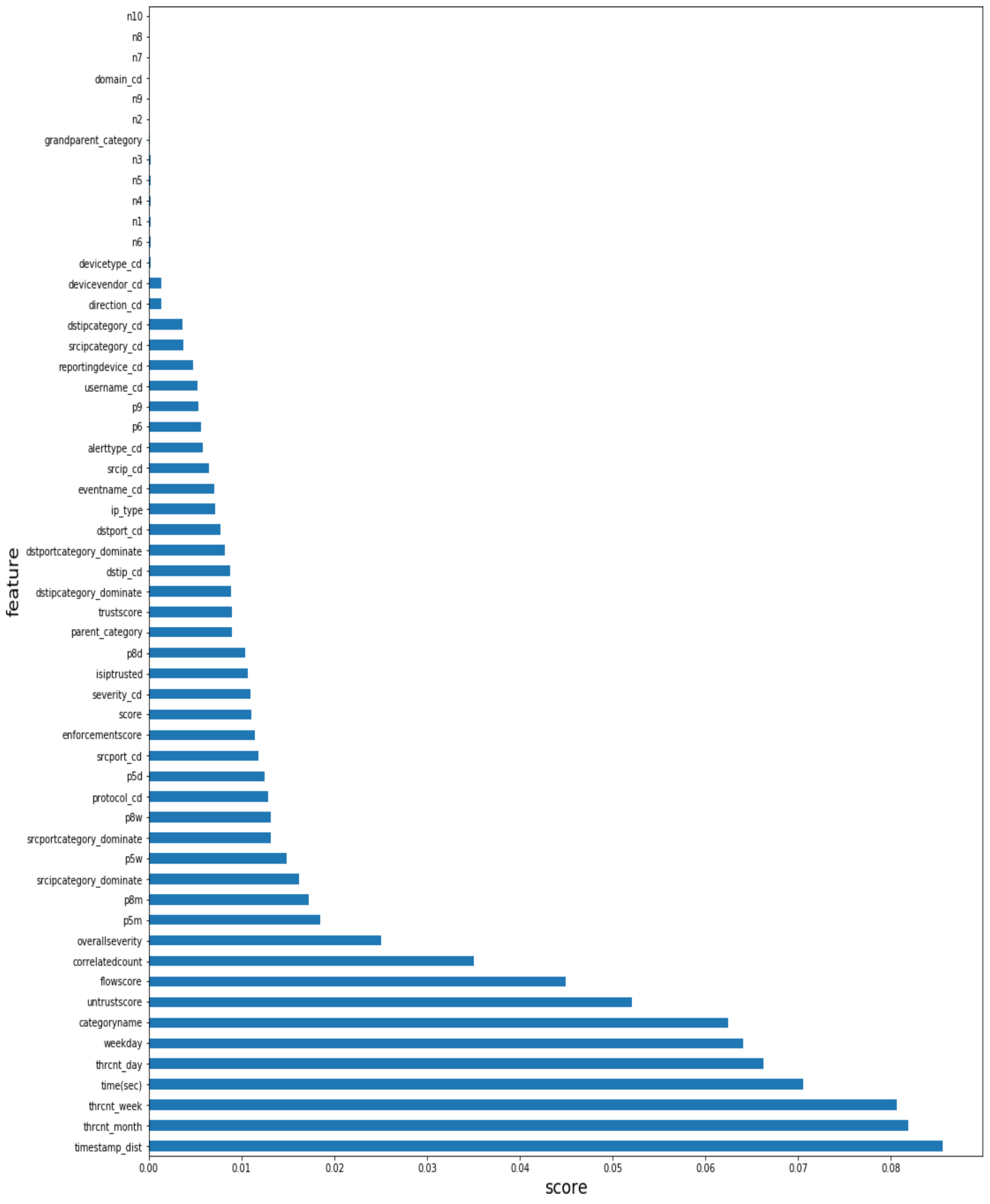


Figure 5.1: Feature Importance graph based on Original Features and their derivatives.

Figure 5.2 shows the feature importance graph based on the entire feature space, i.e. the original features and their derivatives, and the graph-based features.

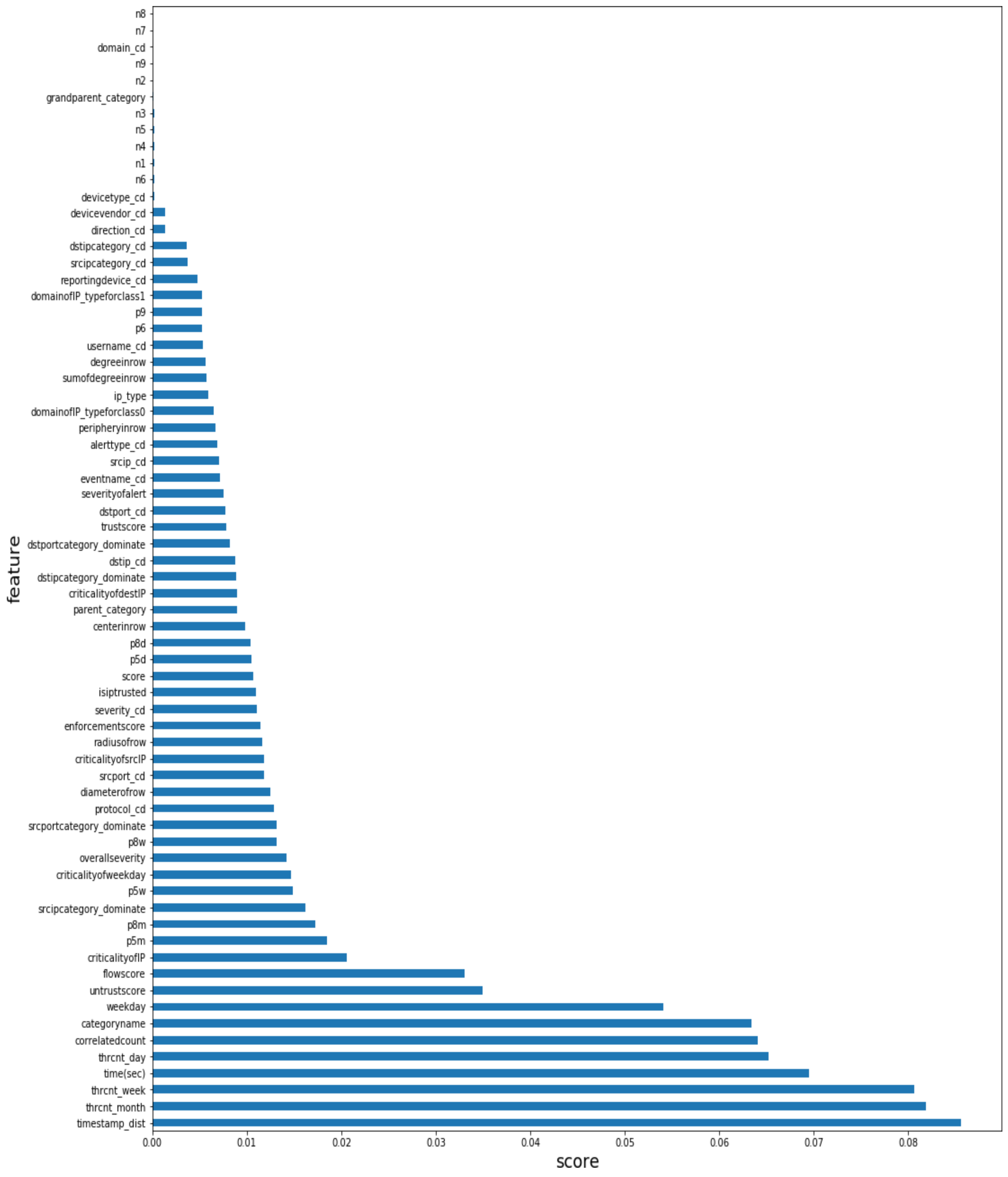


Figure 5.2: Feature Importance graph based on the entire feature space (original and graph-based).

Figure 5.2 shows the feature importance considering a total number of 77 features, out of which 64 features belong to the original features and their derivatives and 13 features belong to the introduced graph-based features.

Based on feature importance, we discarded a few features as follows. Firstly, we calculated the importance scores for all the features and normalized those values so that they range between 0 and 1. Then, we kept all those features whose feature importance value was greater than or equal to 0.001. We discarded 23 features which are alert_ids, client_code, grandparent_category, domain_cd, direction_cd, n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, devicetype_cd, devicevendor_cd, start_hour, start_minute, start_second, IP, ipcategory_name and ipcategory_scope. Specifically, we kept in the second group a total number of 54 features out of 77 features.

5.2 Data Preprocessing

Data Preprocessing which is known as feature processing is a technique in which the raw data is transformed into an understandable format. It is the initial and the most essential step in machine learning as the raw data cannot be fed directly to the machine learning models. The raw data contain null values and some missing values, so some operations must be applied to this type of data to get valuable information from it. Data preprocessing includes cleaning the data, handling the outliers, normalizing the data, standardizing the data and handling the missing values.

5.2.1 Handling Missing Values

In the SOD dataset there were many missing values. We handled the missing values in the dataset by taking the median of the numerical values and taking the mode (most frequent class) of the categorical values and then replacing the missing values with the median and the mode values, respectively.

5.2.2 Data Standardization

Data Standardization which is also known as feature scaling is the process of bringing the numerical data into a common format. It is used to normalize the range of feature values. This process ensures that all the data points in the dataset follow the same distribution, without losing information or distorting differences in the range of values.

Feature scaling becomes an essential requirement when features are of varying scales. There are different techniques for feature scaling [10], such as Z-Score normalization, Min-Max normalization, etc. To achieve feature scaling, we have used Z-score normalization which is also known as mean normalization to transform the data in such a way that mean equates to 0 and standard deviation varies according to the datapoints.

The Z-score normalization calculates the Z-score of each value and then replaces the given value with the calculated Z-score. The Z-score is calculated using the following formula:

$$x = \frac{x_i - \mu}{\sigma}$$

where x_i represents one single value and μ (mean) is the mean value of the feature and σ is the standard deviation of the feature. The reason for choosing Z-score normalization over other techniques is that, when Z-score normalization is applied on the datapoints it will handle outliers as well, unlike in the case with min-max normalization. The Outlier is a point that deviates significantly from other points. It is very important to handle the outliers as they can alter the training process of the machine learning algorithms, ultimately resulting in less accuracy and poorer results. Min-Max normalization does not handle outliers but brings all the data points in one single range.

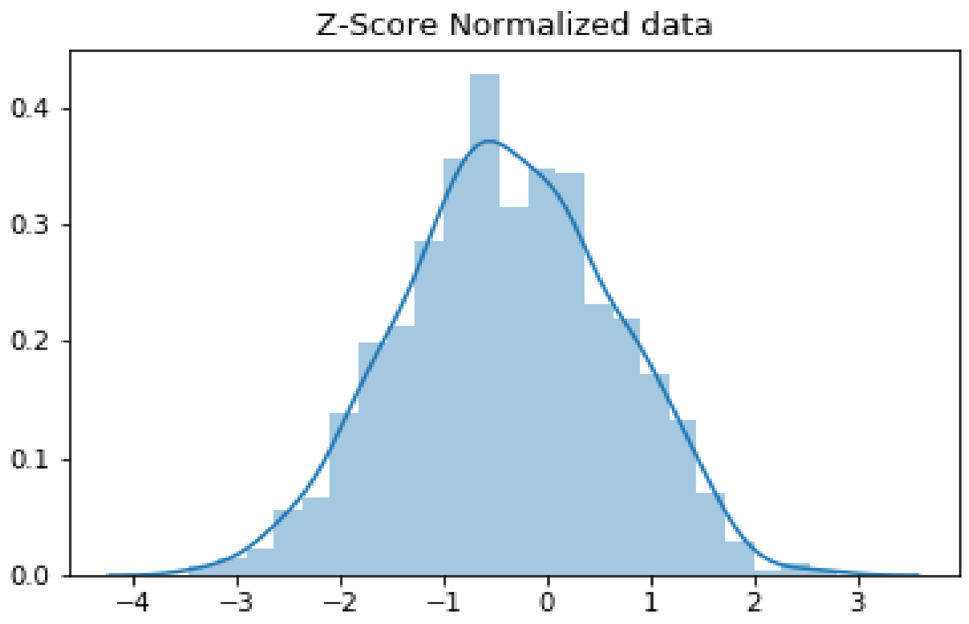


Figure 5.3 Representing Z-score normalized data

Figure 5.3 represents the distribution of all the feature values after Z-score normalization has been applied to the dataset. The distribution is a bell-shaped curve which shows that not all the data points are in a range of just 0, instead there is a deviation. If the value is equal to the mean which is 0 then it is normalized to 0, if it is below mean it is normalized to a negative number and if it is above the mean it is normalized to a positive number. The range of those positive and negative numbers is determined by the standard deviation of the feature.

5.3 Addressing Data Imbalance

After data preprocessing, the final step is to train the model on the given dataset. In the SOD dataset, one class dominates the other class. In machine learning and data science, this scenario is called imbalanced class distribution. As the data points of one class are higher than the other class, there are chances that the predictive model developed using this dataset could be biased and inaccurate. So, it becomes very important to balance the dataset. Various data sampling techniques are available in data science to overcome these challenges such as undersampling, oversampling and SMOTE.

Undersampling is a technique in which the majority class is under-sampled randomly and uniformly so that the samples of the majority class are brought closer to the minority class. This technique solves the class imbalance issue but there is a risk that the technique removes some of the majority class instances which are more representative, thus discarding the useful information. Oversampling is a technique that duplicates the samples of the minority class such that its proportion becomes almost equal to that of the majority class samples. This way the data become balanced. The downside of oversampling is that because of duplicate data the model may overfit the samples and performing a cross validation can be more difficult. As undersampling and oversampling techniques have more disadvantages, so we implemented synthetic minority oversampling technique (SMOTE) to balance our dataset.

SMOTE uses the k-nearest neighbors' algorithm to generate new data for the underrepresented class in the dataset. It is an over-sampling approach in which the minority class is over-sampled by creating synthetic examples along with the line segments joining any/all of the *k* minority class nearest neighbors rather than by over-sampling with replacement. Depending upon the amount of over-sampling required, neighbors from the *k* nearest neighbors are randomly chosen. In our dataset, we have class 1 as a minority class. Before applying the SMOTE technique, we split our data in an 80:20 ratio for training and validation purposes. Post data split, our dataset consisted of 39,427 samples in total, consisting of 37,151 samples of class 0 and 2,276 samples of 'class 1'. SMOTE is applied only to 80% of the data which we will use for training purposes. SMOTE oversampled our minority class and balanced the dataset in equal distribution of each class.

5.4 Summary

This chapter presented the techniques for feature analysis and data preprocessing. For feature selection, the Chi-square test is applied on the one hand to the original features and derivatives, and on the other hand to the global feature space covering all the features including both the original and new features. On the one hand, we selected 41 features out of the 64 original features and derivatives to process the machine learning models and on the other hand, we selected 54 features from the entire feature space including the graph-based features out of 77 features to process the machine learning models. In the next chapter, we will conduct different experiments to assess the performance benefit of the proposed graph-based features using different machine learning classifiers.

Chapter 6: Classification Techniques and Experimental Evaluation

In this chapter, we give an overview of the classification techniques used in our work, and then conduct the experimental evaluation of the proposed approach based on the SOD dataset.

We will apply the selected classification models to the first group of features (i.e. original features and derivatives) and evaluate their performance, and then repeat the process by applying the same classifiers to the global feature space including both feature groups.

6.1 Classification Techniques

In this section, we give an overview of the classification techniques explored in our study. In the current thesis, we studied and compared different linear and nonlinear machine learning models. As a linear model we explored Logistic Regression while as nonlinear models, we explored deep neural network (DNN) and XG Boost. We used python Scikit-learn libraries to implement these algorithms.

6.1.1 Logistic Regression

Logistic regression is a popular and simple algorithm used to solve any classification task. The underlying model uses linear regression and the logistic function for classification. Logistic regression is a shallow neural network with no hidden layers, and this makes logistic regression a linear classifier.

6.1.2 XGBoost

XGBoost stands for eXtreme Gradient Boosting. It is a prototype designed to achieve high execution speed and model performance of gradient boosted decision trees. As compared to the implementations of other gradient boosting, XGBoost is fast. It is mainly used in tasks involving supervised learning, such as regression, classification, and ranking. The feature that makes XGBoost unique is that it helps in reducing overfitting.

6.1.3 Deep Neural Network

Deep neural network (DNN) is a nonlinear machine learning technique that is also part of deep learning. Deep learning methods are better than traditional machine learning methods as when we supply more data to the deep learning methods such as deep neural network, the performance of the algorithm eventually improves unlike in the case of traditional shallow classifiers. The performance of shallow classifiers such as SVM becomes constant at some point even when we feed the model with more data and the model stops learning from the supplied data. Traditional shallow classifiers work well when we have less data, while deep learning methods outperform when they are supplied with more data.

As a deep learner, we use a deep neural network (DNN) model depicted in Figure 6.1.
We use a feed forward neural network which is also known as multilayer neural network. In our architecture we have 3 hidden layers and 1 output layer.

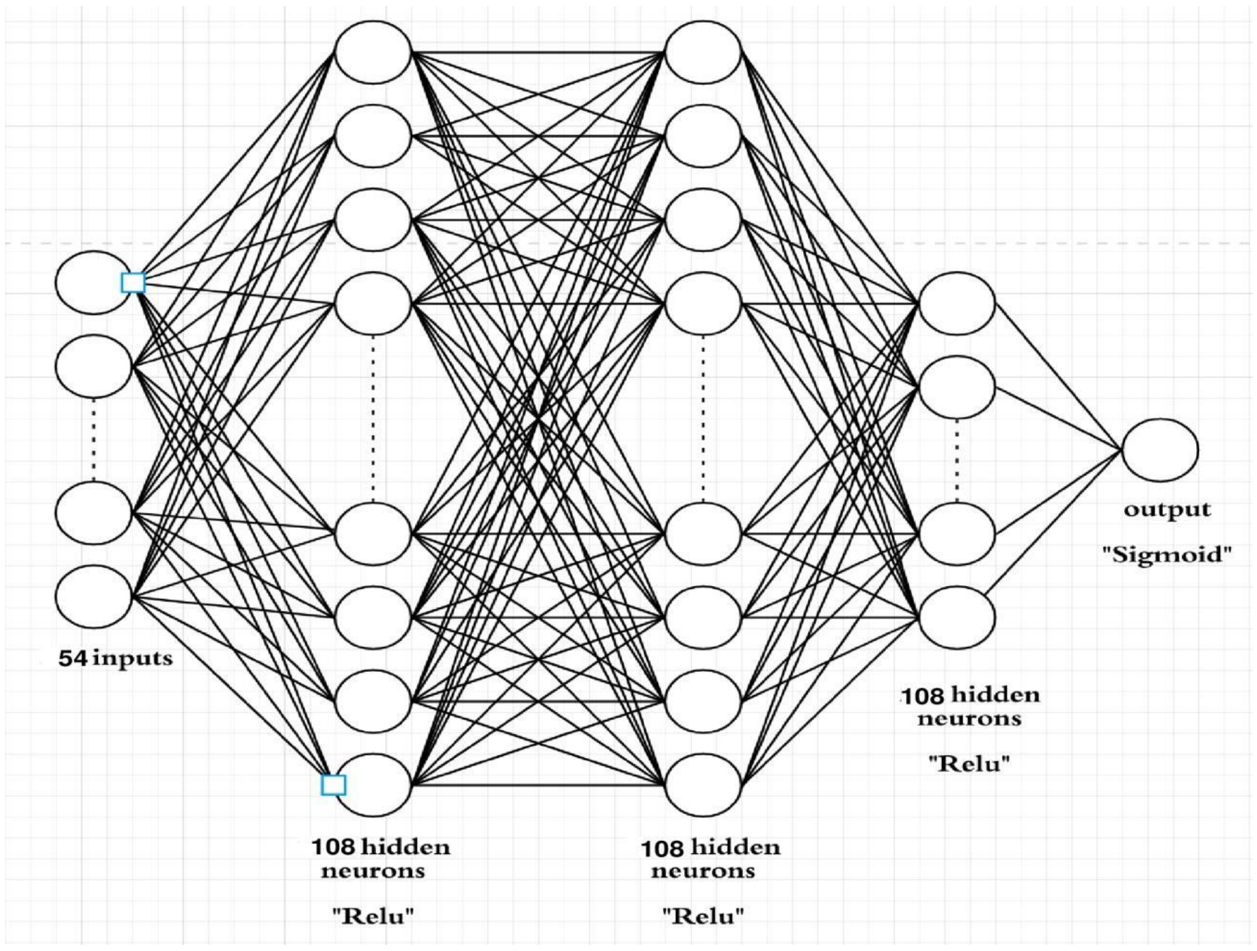


Figure 6.1. Neural Network Architecture with the hidden layers and the activation functions used.

Our deep neural network consists of an input layer which comprises of 54 inputs, a first, a second and a third hidden layer consisting of each 108 neurons, and an output layer consisting of 1 neuron. We use the activation function relu, which is also known as a rectified linear unit. Relu is fast compared to sigmoid and tanh activation functions and it is not affected by the problem of vanishing gradient descent. This function is nowadays used widely because it works well with the multiple layers of a neural network. Its range lies between 0 to infinity and it is also used in the hidden layers of neural networks. When this function is applied on a variable whose value is less than 0 it is given the value of 0 and if it is greater than 0, the function will return the same value.

Due to the binary nature of the SOD dataset, we use the binary cross entropy, which is the default loss function for binary classification problems. To optimize the algorithm and to update the weights, we use a variant of gradient descent called *adam* optimizer; *adam* is an abbreviation for Adaptive Moment Estimation.

To avoid overfitting, we apply dropout on the second and third hidden layers while keeping the keep_prob hyper parameter value as 0.8. The keep_prob hyper parameter value states that on each iteration every neuron has 80% probability of being included in forward and backward propagation and 20% probability of being dropped out. In this way, our learning algorithm has no idea which neurons will be dropped, and which neurons will be included, and thus it will not focus on specific neurons.

6.2 Evaluation based on the Original Features and Derivatives

6.2.1 Using Logistic Regression

Using logistic regression, we evaluated the performance by calculating the F1_score and area under curve (AUC).

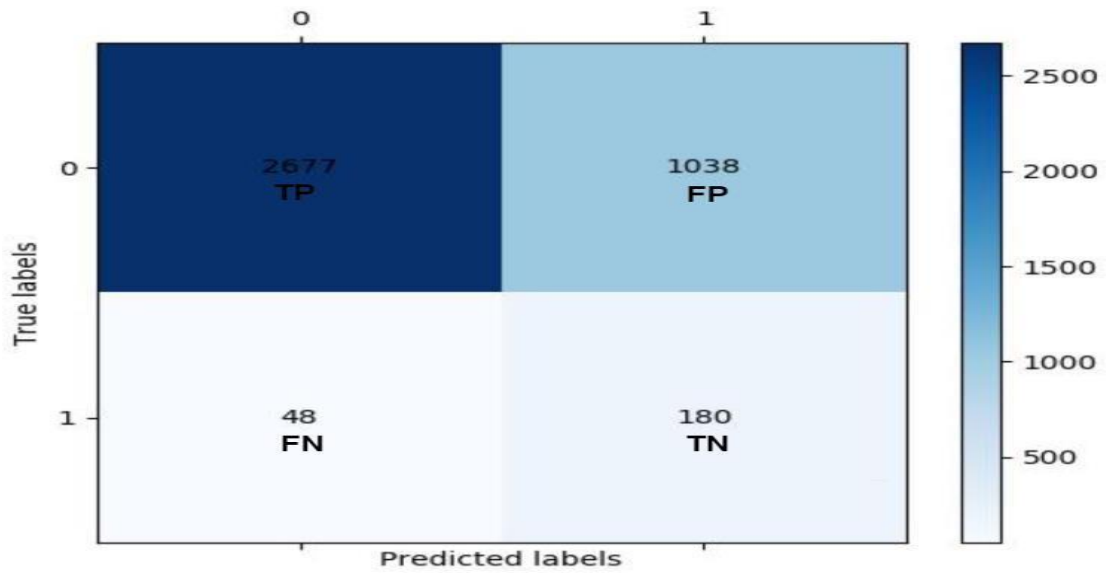


Figure 6.2: Confusion Matrix for Logistic Regression using the original features group

We can observe from figure 6.2 that out of a total of 3,715 samples of class 0, the model has truly predicted 2677 (TP) samples and falsely predicted 1038 (FP) samples. For class 1, out of a total of 228 samples, the model has truly predicted 180 (TN) samples and falsely predicted 48 (FN) samples.

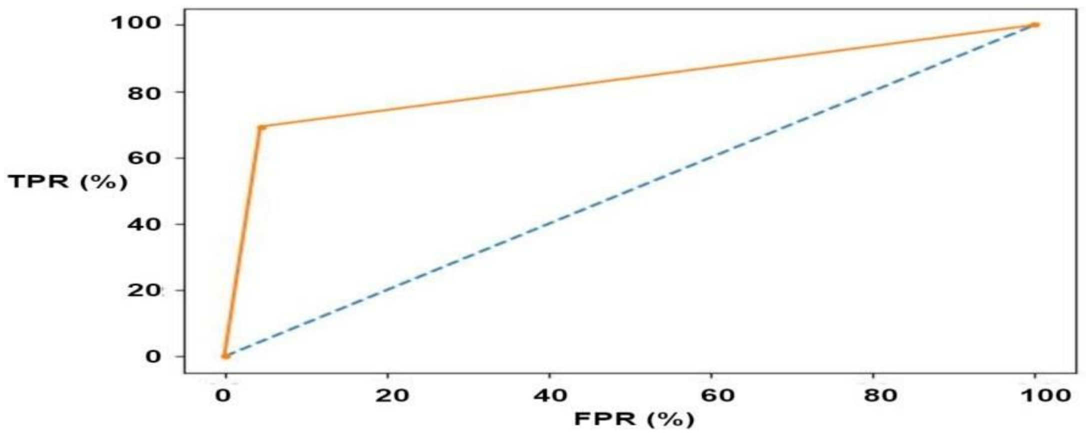


Figure 6.3: ROC for Logistic Regression using the original features group.

Figure 6.3 shows the Receiver Operating Curve (ROC) for the logistic regression. The x-axis represents the false positive rate (FPR) and the y-axis represents the true positive rate (TPR). The area under curve (AUC) derived from the ROC is a performance measure that tells how much a model is capable of distinguishing between classes. The higher the AUC, the better the model is at predicting 1's as 1's and 0's as 0's. The AUC value for this model evaluated to 75.50%.

	precision (%)	**recall (%)**	**f1_score (%)**	**support**
class 0	98	72	83	3715
class 1	15	79	25	228
micro avg	72	72	72	3943
macro avg	57	76	54	3943
weighted avg	93	72	80	3943

Table 6.1: Classification results for logistic regression classifier using the original features group.

From table 6.1, it can be observed that the results obtained from logistic regression are relatively poor and the model is not able to classify the results in a better way. The F1_score for class 1 is very low at just 25%.

6.2.2 Using XGBoost

Using XGBoost, we evaluated the performance by calculating the F1_score and AUC.

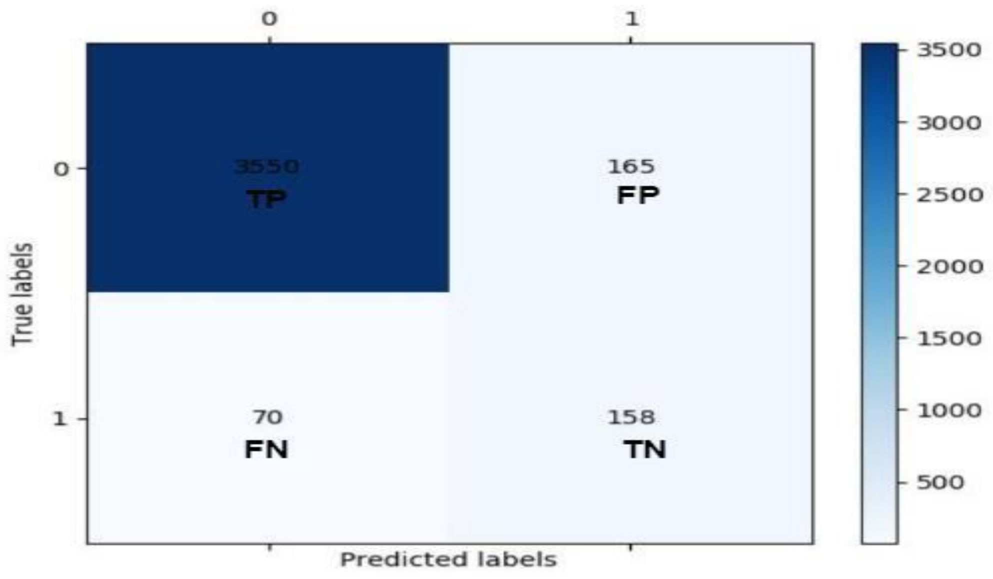


Figure 6.4: Confusion Matrix for XGBoost using the original features group

Figure 6.4 depicts the confusion matrix obtained when using XGBoost. We can observe from figure 6.4 that out of a total of 3,715 samples of class 0, the model has truly predicted 3,550 (TP) samples and falsely predicted 165 (FP) samples. For class 1, out of a total of 228 samples, the model has truly predicted 158 (TN) samples and falsely predicted 70 (FN) samples.

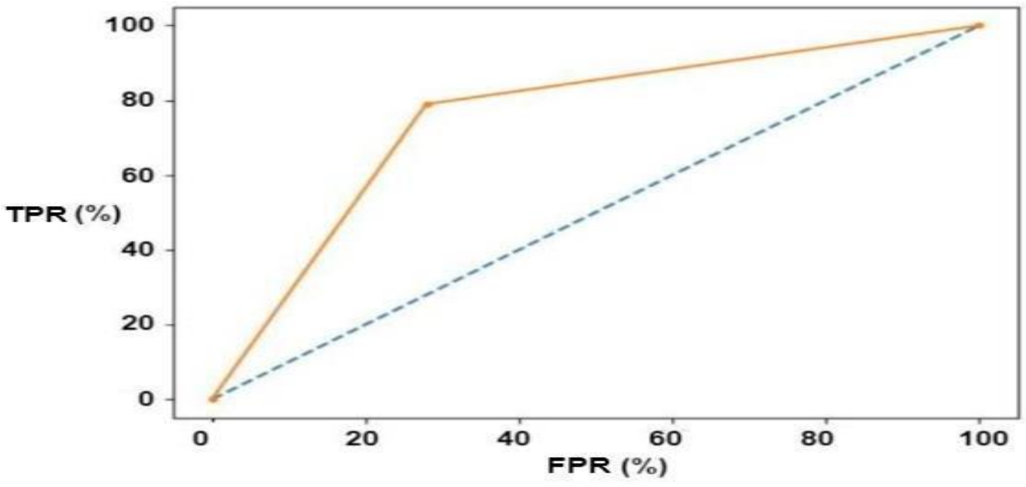


Figure 6.5: ROC for XGBoost using original features group.

Figure 6.5 shows the ROC for the XGBoost model. The AUC value for this model evaluated to 82.42% which is better than the value obtained for the logistic regression model.

	precision (%)	**recall (%)**	**f1_score (%)**	**support**
class 0	98	96	97	3715
class 1	49	69	57	228
micro avg	94	94	94	3943
macro avg	73	82	77	3943
weighted avg	95	94	95	3943

Table 6.2 Classification results for XGBoost using the original features group.

Table 6.2 shows the classification results for XGBoost which are better than the values obtained for the logistic regression classifier. But the value of the precision which is calculated as the percentage of the results which are relevant is very low for class 1 at just 49%. Recall which is calculated as the percentage of the total relevant results which are correctly classified by the algorithm is also very low for class 1 at just 69%.

6.2.3 Using Deep Neural Network

Using DNN, we evaluated the performance by calculating the F1 score and AUC.

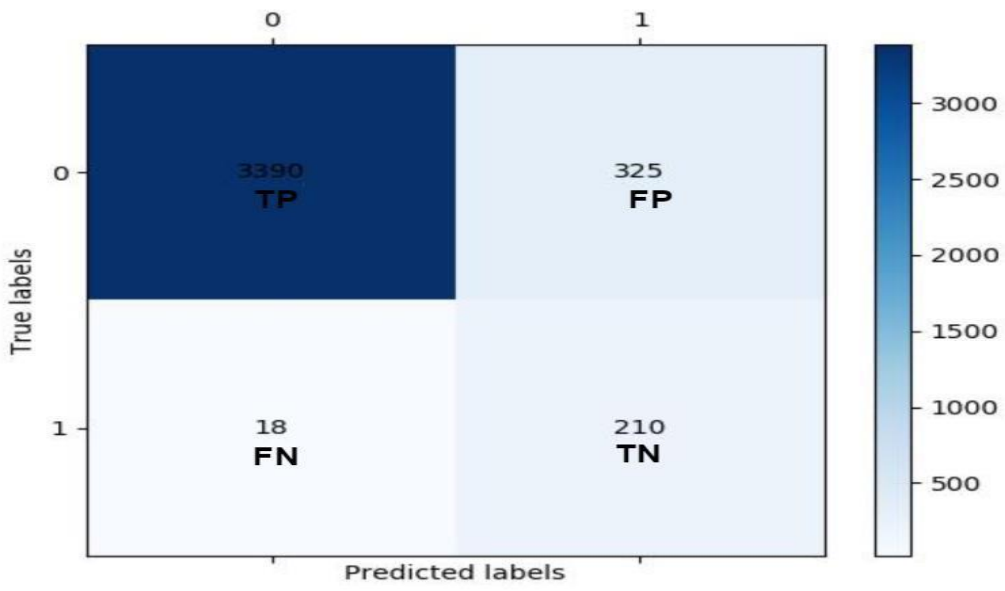


Figure 6.6: Confusion Matrix for Feed Forward Neural Network using the original features group

We can observe from figure 6.6 that out of a total of 3,715 samples of class 0, the model has truly predicted 3,390 (TP) samples and falsely predicted 325 (FP) samples. For class 1, out of a total of 228 samples the model has truly predicted 210 (TN) samples and falsely predicted 18 (FN) samples.

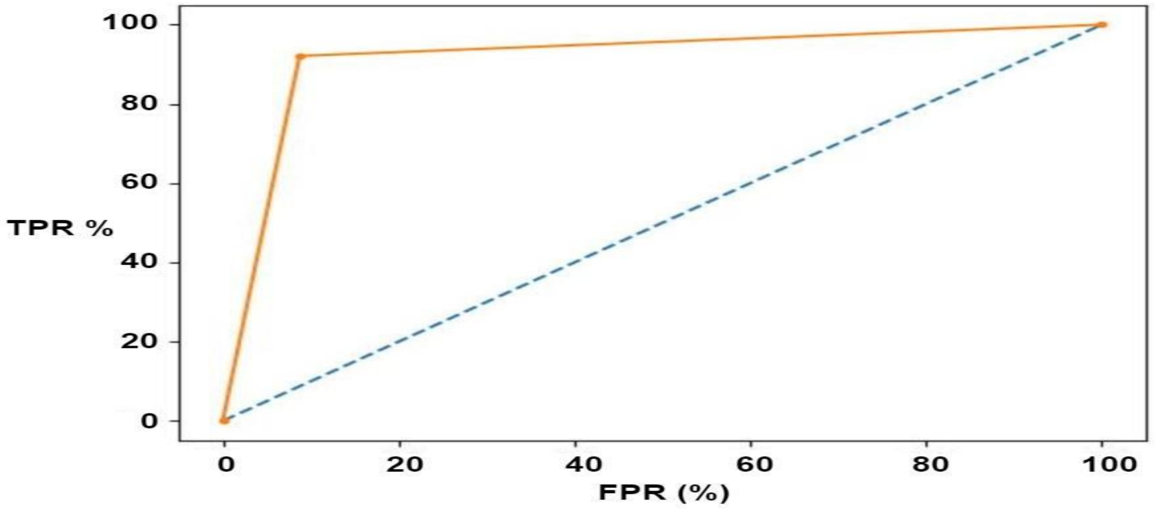


Figure 6.7: ROC for Feed Forward Neural Network using the original features group.

Figure 6.7 shows the ROC for the feed forward neural network. The AUC value for this model is 91.67%.

	precision (%)	**recall (%)**	**f1_score (%)**	**support**
class 0	99	91	95	3715
class 1	39	92	57	228
micro avg	91	91	91	3943
macro avg	69	92	75	3943
weighted avg	96	91	93	3943

Table 6.3: Classification results for Feed Forward Neural Network using original features group.

Table 6.3 shows the classification results for the Feed Forward Neural Network, which are better compared to XGBoost and Logistic Regression. Although the accuracy improved, still the precision value for class 1 is low, just 39%. So, to improve the results we evaluated the machine learning models based on our global feature model i.e. the original features, their derivatives and the graph-based features.

6.3 Evaluation based on the Global Features Model

6.3.1 Using Logistic Regression

In this section, we present the performance results using the global features model with logistic regression.

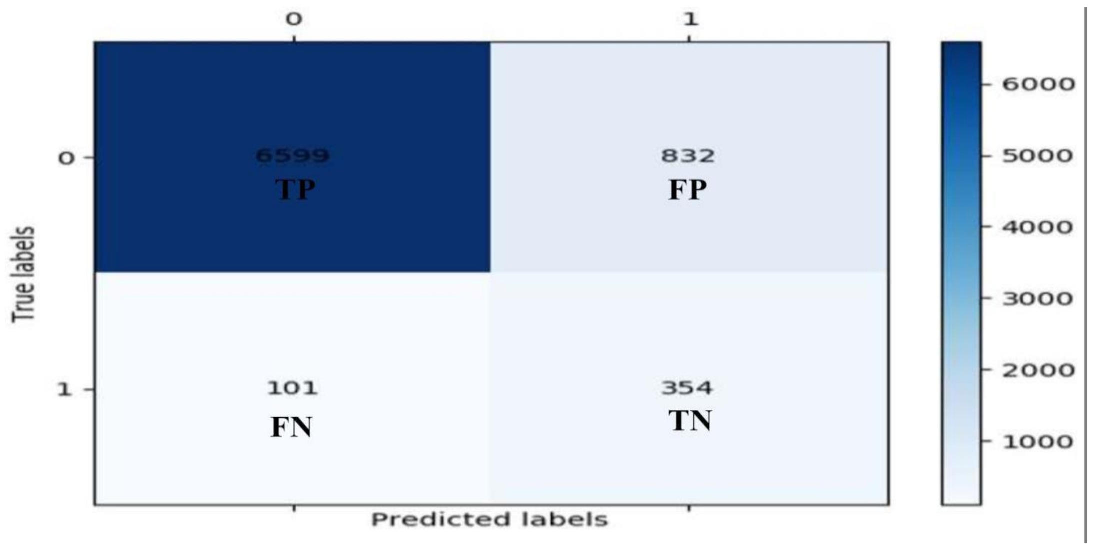


Figure 6.8: Confusion Matrix for Logistic Regression using the global features space.

We can observe from figure 6.8 that out of a total of 7,431 samples of class 0, the model has truly predicted 6599 (TP) samples and falsely predicted 832 (FP) samples. For class 1, out of a total of 455 samples, the model has truly predicted 354 (TN) samples and falsely predicted 101 (FN) samples.

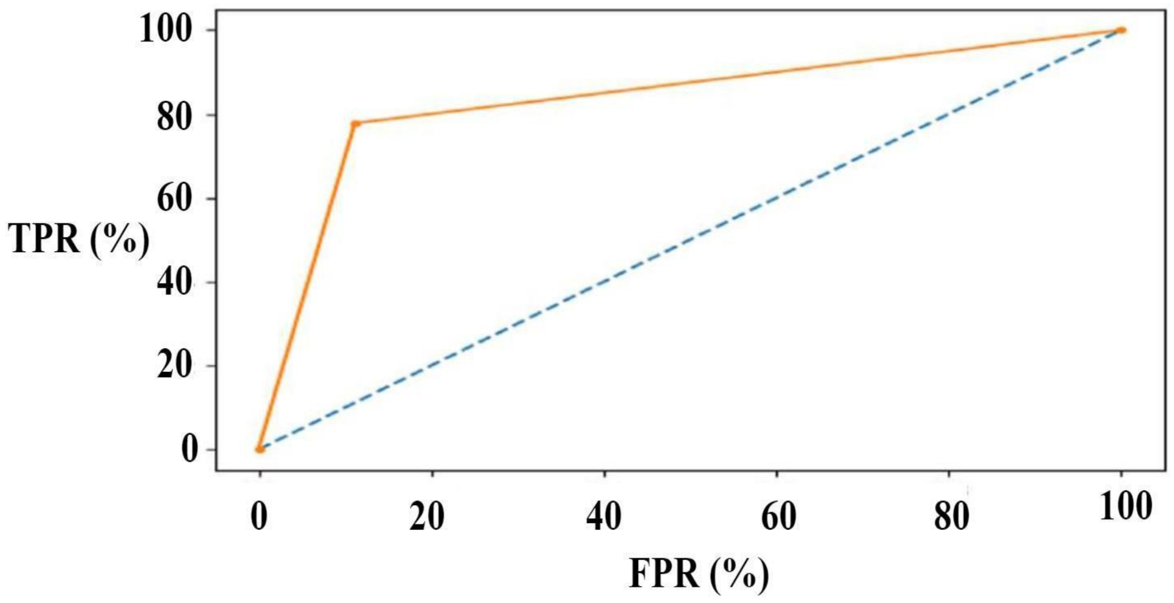


Figure 6.9: ROC for Logistic Regression using the global features set

Figure 6.9 shows the ROC for the logistic regression. The AUC value for this model evaluated to 83.30%. Table 6.4 provides a summary of the performance indicators.

	precision (%)	**recall (%)**	**f1_score (%)**	**support**
class 0	98	89	93	7431
class 1	30	78	43	455
micro avg	88	88	88	7886
macro avg	64	83	68	7886
weighted avg	95	88	90	7886

Table 6.4: Classification results for logistic regression classifier using the global features set

It can be observed that although the results obtained from logistic regression are relatively low, the obtained performance is an improvement over just using the original features and their derivatives. While the previous AUC and F1_score was 75.5% and 25%, respectively, by adding the graph metrics we now have AUC and F1_score for class 1 at 83.3% and 43%, respectively.

6.3.2 Using XGBoost

In this section, we present the performance results obtained for XGboost when using the global features model.

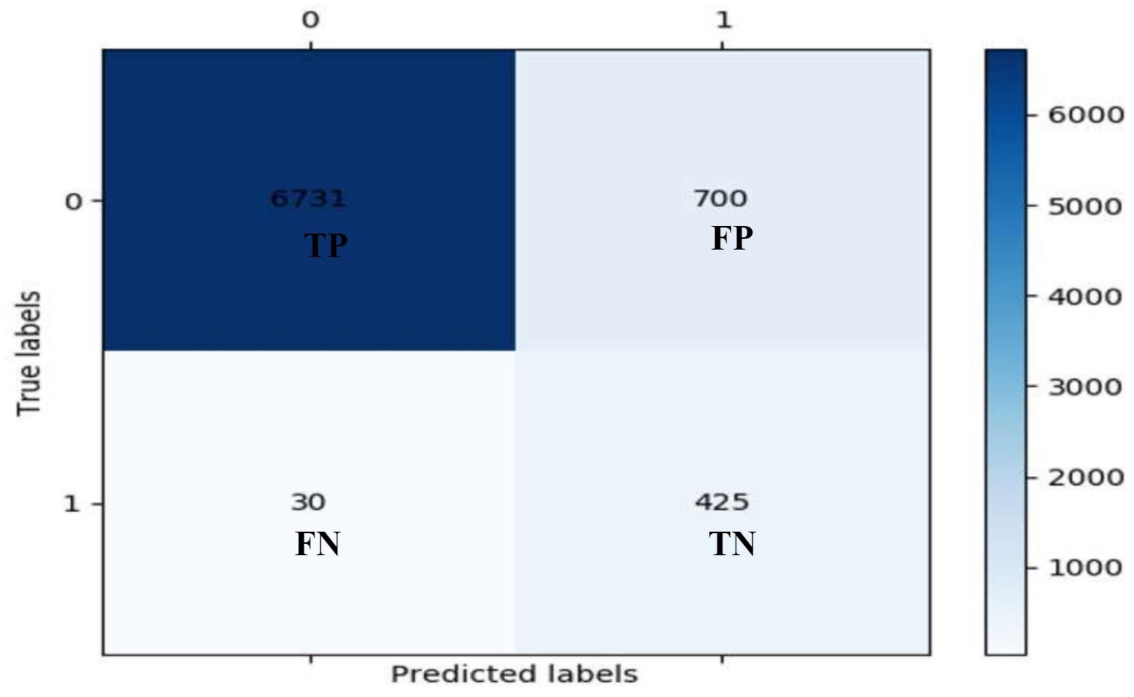


Figure 6.10: Confusion Matrix for XGBoost using the global features set

We can observe from figure 6.10 that out of a total of 7,431 samples of class 0, the model has truly predicted 6,731 (TP) samples and falsely predicted 700 (FP) samples. For class 1, out of a total of 455 samples, the model has truly predicted 425 (TN) samples and falsely predicted 30 (FN) samples.

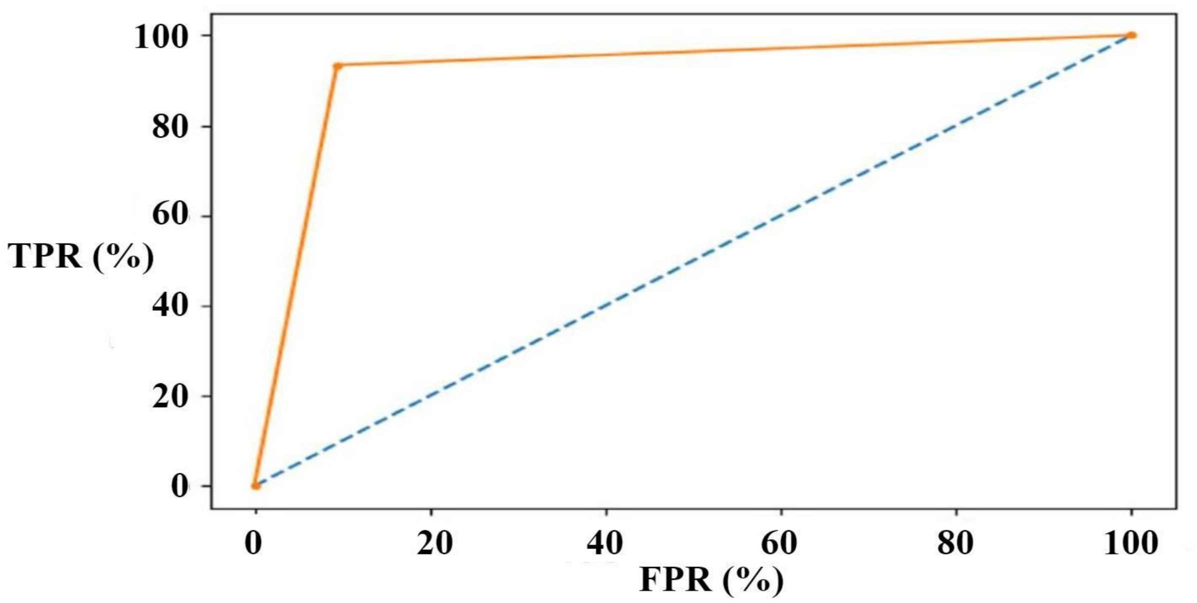


Figure 6.11: ROC for XGBoost using the global features.

Figure 6.11 and Table 6.5 show the ROC and a summary of the performance indicators for the XGBoost model.

	precision (%)	**recall (%)**	**f1_score (%)**	**support**
class 0	100	91	95	7431
class 1	38	93	54	455
micro avg	91	91	91	7886
macro avg	69	92	74	7886
weighted avg	96	91	92	7886

Table 6.5 Classification results for XGBoost using the global features set

The classification results for XGBoost are better than the results obtained for the logistic regression classifier. It can also be noted that the obtained performance for XGBoost is an improvement over just using the original features and their derivatives. While the previous AUC was 82.42%, by adding the graph metrics we now have AUC at 91.99%.

6.3.3 Using Deep Neural Network

In this section, we present the performance results obtained using deep neural network with the global features model.

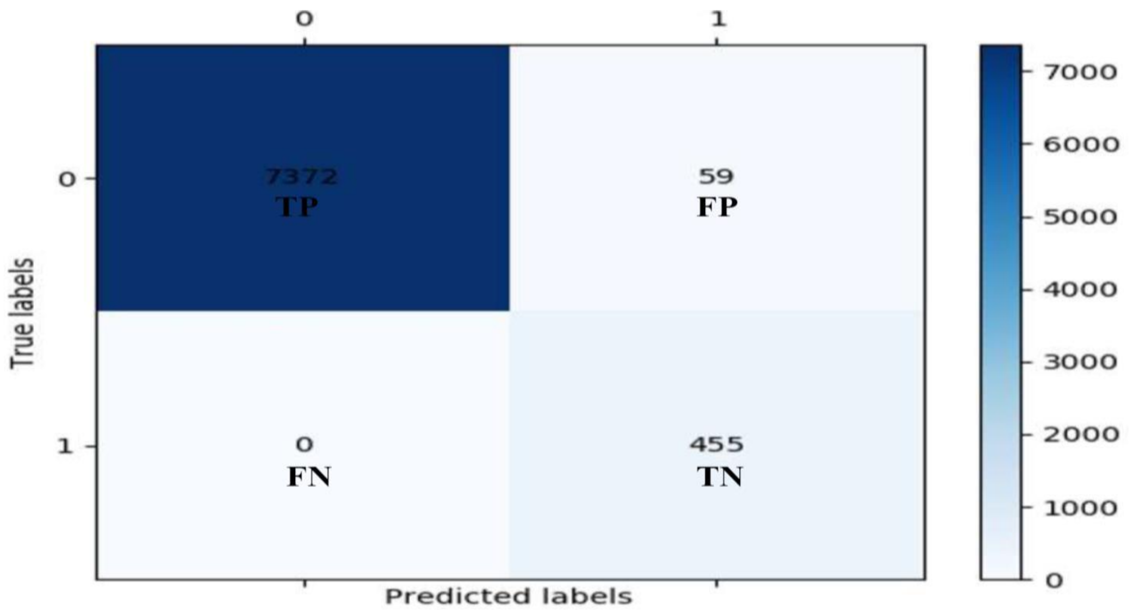


Figure 6.12: Confusion Matrix for Feed Forward Neural Network using the global features set.

We can observe from figure 6.12 that out of a total of 7,431 samples of class 0, the model has truly predicted 7,372 (TP) samples and falsely predicted 59 (FP) samples. For class 1, out of a total of 455 samples the model has truly predicted all the 455 samples as (TN) and there is not a single value that falls under the category of FN.

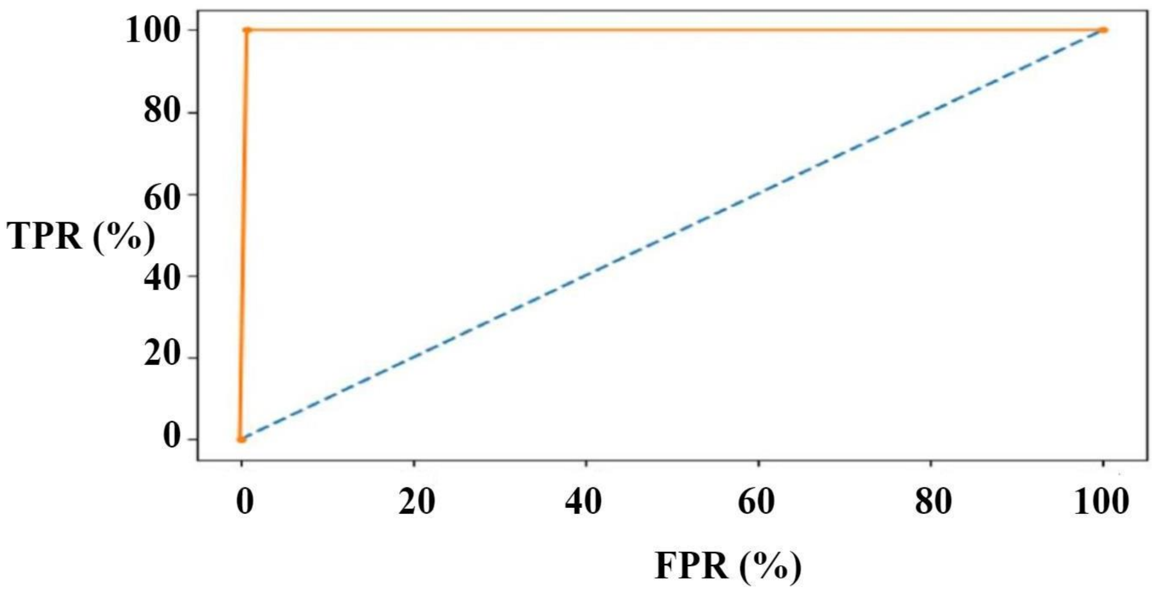


Figure 6.13: ROC for Feed Forward Neural Network using the global features set.

Figure 6.13 and Table 6.6 show the ROC and summary of performance indicators for the feed forward neural network. The AUC value for this model is 99.60%, which is a net improvement over using only the original features and their derivatives, in which case the AUC was 91.67%.

	precision (%)	**recall (%)**	**f1_score (%)**	**support**
class 0	100	99	100	7431
class 1	89	100	94	455
micro avg	99	99	99	7886
macro avg	94	100	97	7886
weighted avg	99	99	99	7886

Table 6.6 Classification results for Feed Forward Neural Network using the global features set.

6.4 Summary

In this chapter, we have demonstrated through experimentation the benefits of using graph-based metrics for SOC security events classification. Specifically, we have shown that extending the generic/default features using graph metrics helps improves the classification performance compared with simply using the generic/default features. We have shown also that using deep machine learning techniques such as DNN helps improves the classification accuracy compared with using shallow learners such as XGBoost or logistic regression.

In the next chapter, we make concluding remarks and discuss our future work.

Chapter 7: Conclusions

7.1 Contribution Summary

Due to the unreliability of the existing technologies, SOC security events classification is conducted to a large extent through manual processes by security analysts. Manual processes are time- consuming and labor-intensive, and thereby, costly and error-prone.

In the last few years, researchers have engaged in enabling the automation of SOC events by using machine learning techniques. These efforts are facing important hurdles related to sparsity, velocity and massiveness of the SOC data. This requires new feature engineering techniques to better capture the substance of the data and make more effective classification decision.

To address the aforementioned challenges, we make 2 major contributions in the current thesis:

1. Introducing a new feature engineering approach based on graph visualization and graph metrics.
2. Validating the intuition that deep learners have the potential to improve the accuracy of SOC event classification over using shallow learners.

The encouraging results obtained using deep neural network, indicates that the aforementioned feature engineering approach provides a pathway toward improving the accuracy of SOC event classification while tackling the growing volume, velocity, and variety of the monitored data.

We started our research first by providing an overview of the alerts generated by the intrusion detection systems and its impact on the organizations. Then we discussed the related work and the dataset. Then we discussed feature engineering and feature selection techniques. Then we introduced the graph metrics and used them as new features. Finally, we trained the machine learning classifiers on a real-world SOC dataset and compared the results of three classifiers, namely Logistic Regression, XGBoost and a deep learning classifier, namely Feed forward neural network.

Furthermore, we tuned hyper-parameters for each classifier to achieve the best performance measures. We noticed that feed forward neural network achieved the best results among all classifiers used for the experiment. Since the number of alerts not notified to the clients in our dataset was relatively large, the other class became a minority class. To avoid the imbalanced class problem, we also applied the SMOTE data oversampling technique to evaluate our classifier.

Through extensive experimentation, we can say that the added features play an essential role with other features in concluding and helped the model achieve 99% AUC.

7.2 Perspectives and Future Work

Our future work will consist of working on enhancing the accuracy of the proposed classification framework. We have worked on feed forward neural network with three hidden layers; the performance of the algorithm can be increased by using ensemble DNN network which is defined as many DNN’s trained separately instead of a single network.

The future work will also include exploring more graph metrics to expand the graph-based feature set. This will help toward enhancing the accuracy of the proposed framework.

SOC data is characterized by its huge size and great velocity. In the current thesis, we have focused only on the effectiveness of the event classification. In our future work, we will study the efficiency of the proposed approach by measuring the processing overhead and speed using existing performance benchmarks. If needed, we will explore techniques to address possible different performance bottleneck and ensure the scalability of the proposed framework.

Our future work will also include extending our experimental evaluation by considering the second SOD dataset which consists of about 430 GB of event log data.

References

[1] Aijaz L., Aslam B. and U. Khalid, “Security Operations Center – A Need for an Academic Environment”, World Symposium on Computer Networks and Information Security (WSCNIS), 19-21 Sept. 2015, Hammamet, Tunisia

[2] Application Security "TOP 10 of the world's largest cyberattacks | Outpost 24 blog", *Outpost24.com*, Available online: https://outpost24.com/blog/top-10-of-the-world-biggest-cyberattacks

[3] Agarwal R., "The 5 Feature Selection Algorithms every Data Scientist should know", Medium, July 2019. Available online: https://towardsdatascience.com/the-5-feature-selection-algorithms-every-data-scientist-need-to-know-3a6b566efd2

[4] Alruwaili F.F. and Gulliver T.A., “SOCaaS: Security Operations Center as a Service for Cloud Computing Environments”, International Journal of Cloud Computing and Services Science (IJ-CLOSER), Vol.3, No.2, April 2014, pp. 87-96

[5] Atzmueller M., and Sternberg E. "Mixed-initiative feature engineering using knowledge graphs." Proceedings of the Knowledge Capture Conference. ACM, 2017.

[6] Chandran S., Case J., Truong T., Zomlot L. and M. Hoffmann, “A Tale of Three Security Operation Centers”, SIW’14, November 7, 2014, Scottsdale, Arizona, USA.

[7] Heim P, Lohmann S, Lauenroth K. and Ziegler J., "Graph-based Visualization of Requirements Relationships," *2008 Requirements Engineering Visualization*, Barcelona, Catalunya, 2008, pp. 51-55. doi: 10.1109/REV.2008.2

[8] Julisch K., “Clustering Intrusion Detection Alarms to Support Root Cause Analysis,” in ACM Transactions on Information and System Security, vol. 6(4), 2003, pp. 443-471.

[9] Jacobs P., Arnab A and Irwin B, "Classification of Security Operation Centers," *2013 Information Security for South Africa*, Johannesburg, 2013, pp. 1-7.doi: 10.1109/ISSA.2013.6641054

[10] Kathuria A, “Methods and Uses of Feature Scaling", Medium, Jan 2019. Available online: https://medium.com/datadriveninvestor/methods-and-uses-of-feature-scaling-94a44b43181a

[11] Larson S., "Every single Yahoo account was hacked - 3 billion in all", *CNNMoney*, October 2017. Available online: https://money.cnn.com/2017/10/03/technology/business/yahoo-breach-3-billion-accounts/index.html

[12] Lord N., "What is a Security Operations Center (SOC)?", Digital Guardian, July 2019. Available online: https://digitalguardian.com/blog/what-security-operations-center-soc

[13] Miloslavskaya N.G., “Security Operations Centers for Information Security Incident Management”, FiCloud, August 2016

[14] Marty R. “Cloud application logging for forensics,” ACM Symposium on Applied Computing, pp. 178-184, March 2011

[15] Network Security, "Host Based IDS vs Network Based IDS | securitywing", Securitywing.com, Feb 2018. Available online: https://securitywing.com/host-based-ids-vs-network-based-ids/

[16] Nguyen, Quoc D., Dras M., and Johnson M. "A novel neural network model for joint POS tagging and graph-based dependency parsing." arXiv preprint arXiv:1705.05952 (2017).

[17] Ponemon Institute, “Improving the Effectiveness of the Security Operations Center”, Research Report, June 2019. Available online [Last accessed: 2019-10-22]: https://www.devo.com/wp-content/uploads/2019/07/2019-Devo-Ponemon-Study-Final.pdf

[18] Pölzlbauer G., Rauber A., Dittenbach M. (2005) Advanced Visualization Techniques for Self-organizing Maps with Graph-Based Methods. In: Wang J., Liao XF., Yi Z. (eds) Advances in Neural Networks – ISNN 2005. ISNN 2005. Lecture Notes in Computer Science, vol 3497. Springer, Berlin, Heidelberg

[19] Rahmani K., Arezoo B. (2006), ‘A hybrid hint-based and graph-based framework for recognition of interacting milling features’, vol 15875-4413.

[20] Ślezak D, Chadzyńska-Krasowska A, Holland J, Synak P, Glick R, Perkowski M. Scalable cyber-security analytics with a new summary-based approximate query engine. In2017 IEEE International Conference on Big Data (Big Data) 2017 Dec 11 (pp. 1840-1849). IEEE.

[21] Schinagl, Stef & Schoon, Keith & Paans, Ronald. (2015). A Framework for Designing a Security Operations Centre (SOC). 2253-2262. 10.1109/HICSS.2015.270

[22] Security Operations Center: Building, Operating, and Maintaining your SOC. Cisco Press. 2015: URL:https://supportforums.cisco.com/sites/default/files/security_operations_center_9780134052014_ch_1_final__0.pdf

[23] Van Niekerk B., & Jacobs P., “Cloud-based security mechanisms for critical information infrastructure protection,” IEEE International Conference on in Adaptive Science and Technology, pp. 1-4, November 2013

[24] Van Ham F. and Wattenberg M. (2008), Centrality Based Visualization of Small World Graphs. Computer Graphics Forum, 27: 975-982. doi:10.1111/j.1467-8659.2008.01232.x

[25] Weiwei C., Huamin Q. “A Survey on Graph Visualization”, Research Report, February 2019. Available Online: https://pdfs.semanticscholar.org/2a52/cd195942cd901d73c118d6a66c97934df3fb.pdf

[26] Wang W., Chang B. (2016) Improved Graph-Based Dependency Parsing via Hierarchical LSTM Networks. In: Sun M., Huang X., Lin H., Liu Z., Liu Y. (eds) Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data. NLP-NABD 2016, CCL 2016. Lecture Notes in Computer Science, vol 10035. Springer, Cham.