5-2020

Machine Learning Algorithms in Network Security

Sasidhar Jakka
St. Cloud State University, sasi9009@gmail.com

Machine Learning Algorithms in Network Security

by

Sasidhar Jakka

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in Information Assurance

May, 2020

Abstract

This starred paper aimed to analyze different machine learning algorithms using security log data and to identify the best algorithm, which is both accurate and fastest in detecting the attacks by analyzing security data. In this paper, we reviewed different security risk assessments and machine learning algorithms and code. We brought together the security risk and machine learning algorithms to analyze security data by creating a test environment. For any organization detecting the attacks accurately and quickly is an essential factor in reducing the risk of a security breach. No amount of systems, standards, compliance guidelines can assure a complete hundred percent guarantee of avoiding the security breach. The assumption is security breaches will happen, and the best way to reduce the risk is to detect the attack early and implement the mitigation procedures. The early detection of the attack will provide security professionals the time to reduce the impact and safeguard the organization. We have discussed in risk assessment how different security guidelines are implemented within the organization, which slow and provide more time and increase the effort of hackers in getting access to core organization systems. This will be achieved by making sure the attack is detected early and once creating multiple layers of security so that it becomes difficult for attackers as risk procedures prevent the Kill chain of attackers by slowing and stopping the attack at different levels.

Acknowledgment

I would like to express my sincere gratitude to my supervisors as well as committee members Dr. Susantha Herath, Dr. Abu Hussein, Abdullah, Dr. Manamperi, Nimantha P and Dr. Collen, Lynn A, for providing their invaluable guidance, comments, and suggestions throughout the course of the paper.

Table of Contents

List of Tables

List of Figures

Chapter I: Introduction

Introduction

Analyzing security log data is critical for any information security division of the company. Still, with the growing number of devices and connections, it is becoming a difficult task for security administrators to analyze and review the logs. It takes a lot of innovation and improvement time from the security professionals and just as the company grows more significant the number of logs that need to be analyzed is snowballing and with this demand of security log analytics multiple new software companies have created security log analysis software's which assist the security analysts in focusing on the right alerts. This paper will review briefly about security logs and multiple security log analytics tools.

This paper will consider the existing challenges of log analysis and evaluate the effectiveness of machine learning in detecting anomalies that are generated by various distributed systems of the organization from hardware and software tools. To create a standalone intrusion detection system and apply different machine learning algorithms to identify the most suitable algorithms which can be used to log analysis. An average web server receives visitors in thousands per day, generating a log a volume size of more than a gigabyte, and to analyze this data by a system administrator is an impossible task unless a considerable number of man-hours are dedicated to that single task. The majority of the commercial of the shelf security tools use log-based intrusion detection systems, which are based on the pattern matching which is only effective in detecting preprogrammed or known attack patterns and is not effective in detecting zero-day attacks and takes time from security professionals to update the system with new patterns. Advanced intrusion detecting systems need to detect both known and unknown patterns automatically and identify key areas in a security network where the system administrator can focus on which our project is developed to solve.

Problem Statement

Machine learning algorithms can analyze big data ad provide data insights to the user. When a security developer wants to build a security-solution based on machine learning algorithms, there is no data that helps the user understand which machine learning algorithm can be applied to make the security solution effective. The lack of information on machine learning algorithms in security is a problem for independent security tool developers and academia. This paper makes a humble effect to address this problem on how machine learning algorithms can be applied to security data to analyze and identify which algorithm performs better in comparison to other algorithms. These are the questions this paper is trying to address.

Nature and Significance of the Problem

The dynamic nature of security is constantly changing and has seen an explosive growth of security tools that wanted to grab the opportunity created by the information technology boom. Security professionals cannot just leave it at evaluating, but they must also decide whether to process it or just gather information from it. The number of tools increases both due to the perceived threats and real issues faced by the security professionals.

This need to have security logs for organizational security was first documented in the "*Department of Defense Trusted Computer System Evaluation Criteria*, (DOD, 1983)". Previously the need to save the log data was not considered important most of the logs were overwritten by midnight on systems and thus any log over specified period unless if saved in another location was erased within a short span of time. The value of these logs increased only when there was a need to evaluate, reconstruct, and analyze.

Today organizations have a lot of systems connected and are increasing with the rapid adoption of technology in business growth. All these systems generate numerous logs, and the first step is consolidating these logs and is analyzed based on security policies.

Objective of the Research

The objective of the project is to review machine learning algorithms and to present the results of how different ML algorithms work in network log analytics intrusion detection systems based on machine learning, which detects anomalies in real-time. The major focus is on identifying machine learning algorithms that are used for autonomous learning and detection of anomalies in logs.

Definition of Terms

IDS (Intrusion detection system) – Collects the packets of ingress data and detects the anomalies by comparing with known malicious attack patterns. Pattern Matching is comparing the network traffic packets of data or logs in real-time. Supervised learning is training a system with labeled data. Un-supervised learning is training a system with Un-labelled data. Machine learning algorithm is mathematical formula to analyze data.

Summary

A review of the need for security tools and its limitation analyzed various security tools with the implementation of automated log analytics and intrusion detection system.

Chapter II: Background and Review of Literature

Introduction

The need for information security came into existence due to multiple risks that started to be discovered during the rapid progression of information technology into organizations and businesses. The main objective of information security is to support the goal of the organization. Every organization is exposed to some major uncertainties and, most of the effect, the organization in an undesired way. In simple terms, any undesired event for the organization is called risk. Information security comes into to picture to defend the organization in guiding them and steering them away from the pitfalls of information technology growth and, at the same time, defending the organization from cyber threats.

Risk management is a complex task and can be limited by the resources that the organization has in the order of priority. The risk that an organization faces is dynamic and change ever too often to have a stable solution for the complex risk factors. The vulnerabilities and threats make it impossible to provide a complete security state for an organization. In order to counter these predicaments, Information technology professionals develop a toolset to assist them in providing an overview in distributing the risks strategy with the assistance of IT and business managers with regards to potential impact and risk appetite of the organization based on the threats of the organization. The security toolset needs to dependable, resilient, and cost-effective and should be reasonable in reducing the risk of the organization.

Risk management is not a new phenomenon. There are many techniques and solutions available for effectively navigating the common risk that an organization faces. This paper tries to explore various risk assessment and risk management issues in relation to information systems. I am primarily focusing on risk in terms of information systems, priority in understanding risk, risk assessment and risk management, and common risk management techniques, methods, and tools.

Risk in Information Systems

Risk is any unwanted event that may occur from the current event or future events that impact negatively. Risk in Information technology perspective is the process of understanding and analyzing the factors that may lead to negative impact or total failure of CIA triad (Confidentiality, Integrity, and Availability) of the Information system. The information security risk is the negative impact due to information resulted due to intentional or accidental attacks on a process or the related information.

"Risk is a function of the likelihood of given threat sources exercising a particular potential vulnerability, and the resulting impact of that adverse event on the organization." (National Institute of Standards and Technology Special Publication 800-30, 2002).

Threats

The most standard definition of threat and threats sources are from NIST (National Institute of Standards and Technology Special Publication 800-30, Risk Management Guide for Information Technology Systems (July 2002)).

"Threat: The potential for a threat source to exercise (accidentally trigger or intentionally exploit) a specific vulnerability" (National Institute of Standards and Technology Special Publication 800-30, 2002).

"Threat-Source: Either (1) intent and method targeted at the intentional exploitation of a vulnerability or (2) a situation and method that may accidentally trigger a vulnerability." (National Institute of Standards and Technology Special Publication 800-30, 2002).

The threat potential of a vulnerability is the impact it has on the process. Threats are not actions on their own. Threats must be unified with threat sources to be high risk. This is an important factor in assessing and managing risk, as each threat source may have a different likelihood, which is directly proportional and affects the risk assessment and risk management. It is often better to include threat sources with a threat for the root cause analysis.

Table 1

List of Threats with Threat Sources Taken into Consideration

Threat (Including Threat Source)	Description
Accidental Disclosure	The unauthorized or accidental release of classified, personal, or sensitive information
Acts of Nature	All types of natural occurrences (e.g., earthquakes, hurricanes, tornadoes) that may damage or affect the system/application. Any of these potential threats could lead to a partial or total outage, thus affecting availability.
Alteration of Software	An intentional modification, insertion, deletion of operating system or application system programs, whether by an authorized user or not, which compromises the confidentiality, availability, or integrity of data, programs system, or resources controlled by the system. This includes malicious code, such as logic bombs, Trojan horses, trapdoors, and viruses
Bandwidth Usage	The accidental or intentional use of communications bandwidth for other than intended purposes
Electrical Interference/ Disruption	An interference or fluctuation may occur as the result of a commercial power failure. This may cause a denial of service to an authorized user (failure) or a modification of data (fluctuation)
Intentional Alteration of Data	An intentional modification, insertion, or deletion of data, whether by an authorized user or not, which compromises confidentiality, availability, or integrity of the data produced, processed, controlled, or stored by data processing systems.
System Configuration Error (Accidental)	An accidental configuration error during the initial installation or upgrade of hardware, software, communication equipment, or operational environment.
Telecommunication Malfunction/ Interruption	Any communications link, unit or component failure sufficient to cause interruptions in the data transfer via

	telecommunications between computer terminals, remote or distributed processors, and host computing facility

(Institute, 2006)

Vulnerabilities

Vulnerability is flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system's security policy. (National Institute of Standards and Technology Special Publication 800-30, 2002) .

The vulnerability can be a flaw in the system, which can be exploited by unwanted entities. Vulnerabilities are often loop-holes in standard operating systems or operating procedures that systems perform. The process of storing passwords, misconfigured audit logs are vulnerabilities. Another contributor to vulnerabilities is identified at the policy level. This shows that if a lack of a properly defined security policy is directly proportional to a lack of vulnerability analysis.

Importance of Managing the Risk

The main reason for managing risk in an organization is to reduce the guard the objective and assets of the organization. This shows risk management should be more of a management function rather than a technical function as the managers have a holistic view of the organization and can integrate the risks for each business function and their potential business impact. It is critical to manage risk to information systems and to understand the specific risks to a system allows the systems administrator to protect the system from vulnerabilities with mitigation procedures according to the mission of the organization. Organizations have limited resources to tackle risk; having a clear understanding of the risk will assist us in determining the priorities based on the magnitude of the risk, and as the risk cannot be completely reduced, organizations use the 80/20 rule to prioritize the risk. Eighty percent of the risk impact coming from twenty percent of the vulnerabilities. The top risk impact vulnerabilities will have priority in determining which application with high vulnerabilities will be selected for the mitigation procedures and budget allocation.

Measuring Risk

Risk is identified by measuring the threats and vulnerabilities and calculating the likelihood of the impact of each risk. Risk assessment is a complex procedure included with partial information. Multiple methodologies were devised to tackle these pitfalls and provide consistent data regarding risk assessment.

Major Risk Assessment Methodologies

Quantitative Risk assessment: Quantitative risk uses methodologies developed for financial and insurance and trading & hedge fund companies in determining the risk of potential assets and applies those to information systems for risk impact business impact cost of risk and overall risk. This can be measured as the direct cost of the risk and indirect cost of the risk.

This quantitative risk is described mathematically as Annualized Loss of expectancy (ALE). The expected risk in terms of cost and loss in monetary terms can be expressed as the loss occurred due to risk over a year.

ALE=SLE* ARO

SLE (Single Loss Expectancy) is the value of a single loss of the asset. This may
or may not be the entire asset. This is the impact of the loss.
ARO (Annualized Rate of Occurrence) is how often the loss occurs. This is the
likelihood.(Institute, 2006).

Mathematical representation of the risk in statistical terms gets very complex and is beyond the present scope of the paper. When using quantitative risk assessment looks logical, there some drawbacks of this approach in using for information systems. It’s easy to define the cost of the system, but the indirect costs are high, and the recovery cost is high and sometimes unknown.

The quantitative risk assessment for information systems comes with a huge error margin as this method was not designed primarily for information systems in focus. Nowadays, there is a lot of improvement with new data processing and the high availability of data to use complex statistical processes to extrapolate and uncover hidden trends and risks. The statistical process also ensures that the quantitative risk assessment is repeatable and reproducible with the same consistency.

Quantitative risk assessment is not a cost-effective solution for an Information technology system review as it is difficult to find accurate information as variables and incomplete information. But if the information is reliable its better select, the qualitative risk assessment to communicate risk occurs the management.

Quantitative risk assessment is the general risk measurement method used in many different fields like finance, insurance, actuarial science, but mainly used for risk assessment in information technology systems. The main reason for this is complexity in the identification and assigning of value to assets and limitations of statistical data, which helps in determining the frequency. Due to this majority of risk assessment measures in information technology systems are based on qualitative risk methodology.

Qualitative Risk Assessment: The main assumption in qualitative risk assessment is that there exists a high degree of uncertainty of impact values and likelihood that’s defined. Risk is subjective in terms of issues where the greater the difficulty in qualitative risk is in describing the likelihood and impact values. These values should be standard scaled so that it can be used with consistency across different risk assessment methods.

Some drawbacks of qualitative risk assessment are harder to communicate the results to management without precise information. Just saying the risk is high or low will not give a better understanding. To improve and overcome this hurdle, a table with impact and likelihood tables and the potential impact will assist the management in better handling the risk.

Threat Identification

Threat identification includes identifying bit the threats and its sources to have a better understanding and accurate assessment. Common threats sources are Natural threats based on environmental factors like tsunami, earthquake, hurricanes, floods. Human threats are resulting in intentional damage or unintentional damage. Like hacking and DDoS attacks. And other habitat threats like power failure, chemicals, and water damage.

Risk management analysts who understand the organization analyze all the attack vectors and surfaces to identify the threats and their sources. The list of threats that are identified is collected and discusses with all major stakeholders who have a good understanding of different domains collate and define the risks which are applicable based on the high priority.

This threat list is highly valuable in decision making, and the organizations use it to baseline risk management actions and processes based on them. As the main criteria for risk management are to have consistency and repeatability, reproducibility due to which the organizational threat list is invaluable. (Coates, 2011)

Vulnerability Identification

Vulnerabilities identification uses multiple risk management methods to document all the vulnerabilities. The first part starts with identifying the commonly found vulnerability lists. Then move to system owners and business owners to identify vulnerabilities applicable to their systems and organization. There are public lists and archives of vulnerabilities available for assessment.

Some of them are Common vulnerabilities and exposures (CVE), National Vulnerability Database (NVD), and if there is already a previous risk assessment report, it becomes a good starting point.

Tools and techniques for vulnerability assessment. Vulnerability scanners are the software tools that analyze information systems and networks and application and program code for any known legacy vulnerabilities that match with vulnerability database. Penetration testing is a systemic testing of network and information systems by information security personnel who evaluate the system against threats. Internal audit and controls of information systems is auditing of information systems and management control's the effectiveness of policies by evaluating documentation and benchmarking with best practices of NIST and ISO guidelines.

Correlation of Threats to Vulnerabilities

The most complex task in the risk management process is correlating threats to vulnerabilities. Establishing the correlation between these is compulsory since the risk is termed as the effect of threat against vulnerability. This is called threat vulnerability pairing. There are multiple methods to complete this process. In threat vulnerability pairing assigning the threats and vulnerabilities accurately and identification of threats that don't relate to vulnerability is an important factor.

Likelihood of risk

The defining likelihood is based on the probability of the threat produced by the threat source and its occurrence against the vulnerability. To benchmark risk assessments, standard definitions are used for risk assessment likely to be consistent.

Examples of likelihood:

1. Low: 0 to 25% chance of occurrence of threat during a one-year timeline
2. Moderate: 26 to 75% chance of occurrence of threat during a one-year timeline.

3. High: 76 to 100% chance of occurrence of threat during a one-year timeline.

Defining impact:

The impact is determined based on the impact on availability, integrity, and impact on confidentiality. CIA triad-based approach helps us to focus our attention or evaluating the impact of confidentially integrity and availability in an information security perspective, so in order to be standardized and repeatable and easily understood by the organization, stakeholders should understand.

Sample impact definitions:

Low:

- Confidentiality: loss of confidentiality leads to a limited effect on the organization.
- Integrity: loss of integrity leads to a limited effect on the organization.
- Availability: loss of availability leads to a limited effect on the organization.

Moderate:

- Confidentiality: loss of confidentiality leads to a serious effect on an organization.
- Integrity: loss of integrity leads to a serious effect on the organization.
- Availability: loss of availability leads to a serious effect on the organization.

High:

- Confidentiality: loss of confidentiality leads to severe effect on origination.
- Integrity: loss of integrity leads to severe effects on the organization.
- Availability: loss of availability leads to severe effects on the organization.

Assessing risk

Risk assessment is the process of determining the threat occurrence against the vulnerability and which results in compromise and impacts the business. In risk assessment, likelihood and impact threat environment and controls should be taken into consideration.

Senior management in an organization look for more quantitative risk assessment and want accurate measures of a risk assessment report with numbers.

Risk Management process

Risk management is divided into four different processes: mitigation, transfer, acceptance, avoidance. A risk management strategy must include the processes to reduce the risk to a manageable level within the cost assigned.

Mitigation

The common risk management strategy is mitigation. It involves steps that either eliminate the risk or reduce the risk by adding controls or applying solutions.

Transference

Like the word transfer, it means to transfer the risk to another entity for a cost who can handle the risk on your behalf. For example, car insurance or health insurance is where you consider the individual risk and share it with the combined collective risk, which does not reduce the likelihood but decreases the impact of the risk.

Acceptance

This is the process of allowing the system to operate with known risks. Most of the low risks are simply put into accepted category similar risks, which have high mitigation costs. Information security personnel should always document this in writing and make sure that it should be accepted so that the IT security team will not be blamed for the compromise.

Avoidance

This is the process of completely removing the vulnerable part of the system or the entire system, which poses a risk.

Risk Communication

Once when the risk is analyzed and measured, the risk management strategy and risk should be clearly communicated to the management of the organization in a lucid method. Presenting the risk report to managers who will better manage the risk. The Quantitative risk assessment used decision making on comparing the cost of the risk management strategy.

Risk management implementation

The risk management implementation plan of action milestones is an important part. The POAM assists management in communication and tracking of implementation of risk management strategies.

Common Risk Assessment tools and methods

There are multiple Risk assessment and management methods and tools—some of the methods for managing risks in information systems.

National Institute of standards & technology:

NIST is based on a design that is primarily qualitative analysis and depends upon qualified security analysts working with technical experts to thoroughly analyze and identify, evaluate, and manage risk in information systems.

NIST has nine steps:

- Step 1: System Characterization
- Step 2: Threat Identification
- Step 3: Vulnerability Identification

- Step 4: Control Analysis
- Step 5: Likelihood Determination
- Step 6: Impact Analysis
- Step 7: Risk Determination
- Step 8: Control Recommendations
- Step 9: Results Documentation

OCTAVE: Octave was developed by a software engineering institute (SEI) at Carnegie Mellon University. It stands for Operationally Critical, Threat, Asset, and Vulnerability Evaluation (OCTAVE) process. It helps organizations to better handle the risk and manage and to protect them. OCTAVE is workshop based and is not a tool.

Three phases of the workshop:

- **Phase 1**
  - Process 1: Identify Senior Management Knowledge
  - Process 2: (multiple) Identify Operational Area Management Knowledge
  - Process 3: (multiple) Identify Staff Knowledge
  - Process 4: Create Threat Profiles
- **Phase 2**
  - Process 5: Identify Key Components
  - Process 6: Evaluate Selected Components
- **Phase 3**
  - Process 7: Conduct Risk Analysis
  - Process 8: Develop Protection Strategy (workshop A: strategy development) (workshop

- B: strategy review, revision, approval). These OCTAVE workshops produce output, and those processes are Protection Strategy, Mitigation Plan, Action List.

FRAP is created by Thomas Peltier, and it stands for the Facilitated Risk Assessment Process. The basic principles of FRAP utilize risk management techniques in a cost-effective way. FRAP utilizes a qualitative risk analysis methodology using Vulnerability Analysis, Hazard Impact Analysis, Threat Analysis, and Questionnaires. FRAP correlates Risk with Business impact analysis for determining impact.

COBRA stands for Consultative, Objective, and Bi-functional Risk Analysis, which focuses on the business issue rather than a technical issue. It uses tools that can be purchased to do a self- assessment of risk based on the primary knowledge bases.

- IT Security (or default)
- Operational Risk
- 'Quick Risk' or 'high-level risk.'
- e-Security

The two Primary products are RISK consultant and ISO compliance.

Introduction to ML in security risk.

This chapter will provide detailed information about various aspects of security logs and machine learning concepts and security tools that are in use.

Background

Developing and implementing automated log analytics and intrusion detection system requires a broad knowledge of security, networking, machine learning, and programming know-how. Learning algorithms and detection approaches.

Intrusion is unauthorized access to an organization network through system vulnerabilities, acquiring access ids of staff of a company through social engineering are some of the examples of intrusion. Hackers try to access an organization network through running malicious programs like port scanning or through zero-day vulnerabilities and try to create a superuser access account. Once hackers create this account, they can act as an authorized user and can carry on with modification encryption and copying data, which affects the confidentiality, integrity, and availability.

Syslogs are logs generated by an operating system. These logs record system admin specified activities that occur in a system. Logs comprise of access logs, changelogs, authentication logs, executable programs installation logs. Whenever you enter a username or password wrong, the system captures that information and records it as a failed log attempt.
These are the main inputs for log analysis and intrusion detection, where this data from the logs is analyzed to identify anomalies. (Alspaugh, 2014)

The logs have different sources. Major contributors for the logs are hardware, software, and security tools as follows:

- Hardware logs – logs generated from physical components of a system like hard disk, processor, network router.
- Software logs – logs generated by the operating systems and other software that is installed in the host. Software like databases that store critical information will have additional stringent log requirements.
- Security tool logs- host-based intrusion systems and firewalls, antivirus agents installed on endpoint devices will constantly track changes and record the logs.

SIEM is a security information and event manager, which is a prominent security tool system that collects logs from multiple distributed systems and analyzes the logs for anomalies. The endpoint devices like mobile devices, individual client systems have limited resources for holding and maintain the logs. These logs from endpoint systems are sent to a centralized server, which consolidates all the logs and stores based on security segregation like for example, grouping by source and destination.

Anomaly detection process is a key part of the project. This anomaly detection can be made effective using various detection methods and using the results from different sources and creating correlation rules.

Pattern matching is a basic pattern matching techniques were the first form of detecting anomalies. In pattern matching, the network traffic is captured, and the packets are analyzed on a real-time basis. The network packets during attack follow a sequential approach where we match the sequence of incoming data packets to the preprogrammed patterns defined and approved as attack patterns. This process is a simple logical step of detecting known patterns, but its biggest drawback is ineffective in identifying new attack patterns. (Mbugua, 2016).

State-full pattern matching *a*s the name indicates, this type of pattern matching works by looking at patterns in a continuous stream of data and comparing it with its own database pattern, which is recorded by the egress data.

State-less pattern matching is a type of pattern matching, where there is no saving of data stream patterns. Instead, the matching is defined rules which are calculated by the volume of data coming from a certain source and noticing the fluctuations or sudden surge. (Muller, (2001))

Heuristic algorithm-based Analysis is a detection method which uses heuristic algorithms that are derived from statistical analysis and provide statistically significant results. (Mbugua, 2016).

Network Protocol Analysis is an analysis identifies how most of the intrusion attacks occur by using vulnerabilities identified in network protocols like TCP /UDP. In this, if there is any suspicious change in protocol events generated from router or firewall, this helps in identifying attacks utilizing network protocols.

Machine Learning Analysis

Machine learning has three major types of learning.

Supervised learning

Supervised learning is based on an algorithm that focuses on target – outcome variables, and this result is produced based on input variables that are independent. A generated function matches the inputs to the intended outputs. This process continues until the desired accuracy is achieved by the model. *Examples of Supervised Learning: Decision Tree, Random Forest, Logistic Regression, KNN, Regression,* etc. (Ray, 2017).

Unsupervised learning

Unsupervised learning is based on an algorithm that does not have any target outcome or goal to achieve. It is mostly used to clustering inputs into different groups and is widely used in segmenting the data into separate groups for analysis. *Examples of Unsupervised Learning: Apriori algorithm, K-means*. (Ray, 2017).

Reinforcement learning

Reinforcement learning is based on the algorithm by making the machine trained to make a pre-determined decision. It is achieved by exposing the machine to an environment where the system learns by making trial and error. The system is programmed so that it observes previous results and builds upon them and picks the best possible scenario to achieve the desired outputs. (Manevitz, (2001). 2(Dec),)

Literature Review

Review of Machine Learning Algorithms (Ray, 2017)

There are Multiple Machine learning algorithms before undertaking a machine learning-based project one should study and understand various machine learning algorithms upon which we could accurately identify the best algorithms for our experiment.

Based on the Website article (Ray, 2017). I will describe the common algorithms used in machine learning for data analysis problems with python and R code for the application.

- Linear Regression
- Logistic Regression
- Decision Tree
- SVM
- Naive Bayes
- KNN
- K-Means
- Random Forest
- Dimensionality Reduction Algorithms
- Gradient Boosting algorithms
    i. GBM
    ii. XGBoost

iii. LightGBM

iv. CatBoost

Linear Regression

Linear regression is used to calculate accurate values. This algorithm is based on continuous variables where we establish a relationship between dependent and independent variables by fitting the best line.

The ideal fit line is known as the regression line and is represented by the linear equation

$$Y = a * X + b.$$

"*The best way to understand linear regression is to relive this experience of childhood. Let us say, you ask a child in fifth grade to arrange people in his class by increasing order of weight, without asking them their weights! What do you think the child will do? He/she would likely look (visually analyze) at the height and build of people and arrange them using a combination of these visible parameters. This is linear regression in real life! The child has actually figured out that height and build would be correlated to the weight by a relationship, which looks like the equation above."* (Ray, 2017).

In this equation:

Y – Dependent Variable

a – Slope

X – Independent variable

b – Intercept

"The coefficients a and b are derived based on minimizing the sum of squared difference of distance between data points and regression line." (Ray, 2017).

Linear regression is further divided into two types of regression. Simple Linear regression based on one independent variable. Multiple Linear regression based on multiple independent variables. In the process of identifying the best fit line, you can further choose a specific type of regressions like curvilinear regression and polynomial regression.

Logistic Regression

Logistic regression is a classification and not essentially an algorithm. This is mainly used to estimate discrete values like true or false and yes or no depending on given independent variables. Logistic regression works by predicting the probability of occurrence of an event by suiting data to a logistic function. Hence it is called logistic regression, and as it predicts the probability, the result can be binary like yes or no. (Li W. , (2013).)

Let’s say your friend gives you a puzzle to solve. There are only two outcome scenarios – either you solve it, or you don’t. Now imagine that you are being given a wide range of puzzles/quizzes in an attempt to understand which subjects you are good at. The outcome of this study would be something like this – if you are given a trigonometry based tenth-grade problem, you are 70% likely to solve it. On the other hand, if it is a grade fifth history question, the probability of getting an answer is only 30%. This is what Logistic Regression provides you. (Ray, 2017).

This model can be further improved by including interaction terms, removing features, regularization techniques and using a non-linear model. (Ma, (2003). . 1-77.).This is used for likelihood prediction

Equation:

$$\ln(p(X) / 1 - p(X)) = b0 + b1 * X$$

$\ln(odds) = b0 + b1 * X$

$odds = e^{(b0 + b1 * X)}$

Decision Tree

The decision tree algorithm is mostly used for classification problems, but it also works for continuous dependent variables and categorical dependent variables. In this algorithm, we split the data into multiple homogenous sets based on the most significant attributes and independent variables to make segmented groups.

Equation:

$g(E(y)) = \alpha + \beta x1 + \gamma x2$

SVM (Support Vector Machine)

SVM is an algorithm based on the classification method. It works based on plotting the data points in an n-dimensional space where n is the number of attributes. The value of a coordinate is the value of the attribute. (Li, (2003, November))

It is a classification method. In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of each coordinate.

Equation:

$D=\{(x_i,y_i)|x_i\in R_p,y_i\in\{-1,1\}\}_{ni=1}$

Naive Bayes

Naive Bayes algorithm is a classification method based on Bayes' theorem.

Which is an assumption of independence between predictors. "*In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature*." (Ray, 2017).

Bayes' Theorem is stated as:

$$P(h|d) = (P(d|h) * P(h)) / P(d)$$

$$MAP(h) = max(P(h|d))$$

or

$$MAP(h) = max((P(d|h) * P(h)) / P(d))$$

or

$$MAP(h) = max(P(d|h) * P(h))$$

$$MAP(h) = max(P(d|h))$$

KNN (K- Nearest Neighbors)

KNN is a classification used for regression analysis and for classification.
The K nearest neighbor algorithm stores all the data groups and classifies new groups based on the majority vote of its neighbors, which is why it is more popular when in the industry when the classification of groups is done. These groupings are done based on the distance function.

These distance functions can be Euclidean, Manhattan, Minkowski, and Hamming distance. The first three functions are used for continuous function and the fourth one (Hamming) for categorical variables. If K = 1, then the case is simply assigned to the class of its nearest neighbor. At times, choosing K turns out to be a challenge while performing KNN modeling. (Ray, 2017)

KNN is commonly used to identify relations for a new group like in social media. It is used to identify your friends by relation and group based on interests and associations. The drawbacks of this algorithm are due to its high computational requirements, and the variables need to be normalized to prevent bias and should only be done after removing outliers from data.

Equation: 1k∑xiϵNk(x)Yi

K-Means

K-Means algorithm is used to solve the clustering problem. This algorithm follows an easy approach to classify the data by utilizing a certain number of clusters like K-clusters.
The data inside a cluster is homogenous with its neighbor groups being heterogenous.

Random Forest

Random forest is the most popular algorithm for a group of decision trees. The algorithm works by using a collection of decision trees knowns as forest. A new object is based on the classification provided by each tree, which votes for the class. The random forest chooses the classification, which had the majority number of votes.

Equation: $y=1 \times I(x<5)+2 \times I(5 \leq x \leq 10)+3 \times I(x>10)$

Dimensionality Reduction Algorithms

Dimensionality reduction algorithm is a process of minimizing the random variables which are considered by retrieving a group or primary variables which are segmented based on feature selection and feature extraction.

Gradient Boosting Algorithms

GBM is an algorithm that is used to make to increase the prediction power. It is a conglomerate of learning algorithms which envelopes multiple base estimators to increase the power of single estimator(Ray, 2017).

XGBoost algorithm increases the predictive power, which is widely used for increasing the accuracy of events. It is based on a linear model and tree learning algorithm, which makes it better ten times better than other boosting algorithms. (Ray, 2017)

LightGBM algorithm uses tree-based learning algorithms that use a gradient boosting framework, which is designed to be efficient and distributed.
Advantages of LightGBM are due to rapid training speed and efficiency, Low memory usage, supported by parallel and GPU learning, ability to handle large scale data. (Ray, 2017).

Catboost is an open-sourced machine learning algorithm from Yandex.it can easily integrate with the machine learning platform. Its unique property is that it does not need extensive data training like other machine learning models. (Ray, 2017) .

Limitations of the Study

Due to the detection system being developed as an experimental system, there are some limitations to the functionality and data.

Software Limitations:

System development has many complexities included due to the involvement of machine learning algorithms and the systems that process the network data, which require software and computing resources. Due to this, free software available for machine learning was used.

Data used in the experiment:

Due to the unavailability of real-time data, the data being used in the experiment is test data created for the usage of academic users to be used for academic studies.

Summary

Reviewing all the major machine learning algorithms provided us with a brief understanding of different machine learning algorithms. Sorting through the algorithms, we can find favorable algorithms that help us in our experiment to obtain the desired result.

Chapter III: Methodology

Introduction

This chapter discusses the study methodology and design of the study in detail, including how the data is collected for the study.

Design of the Study:

The design of the study is a quantitative approach. Which is based on leveraging software application system, which includes the capabilities to use Machine learning algorithms. This will be a Quantitative focused.

Data Collection

The data collection approach is using secondary data. The source of Network data used in this experiment is from academic resources. The "Canadian Institute for Cybersecurity datasets is used around the world by universities, private industry, and independent researchers." (Brunswick). The data sets are created and provided freely to students to be used in academic research and experiments. The data is collected by using AWS (Amazon web service).

Data Analysis

The experiment uses Anaconda, which is a free and open-source software of the Python and R programming languages used mostly for scientific computing. This software simplifies the management of machine learning software. (scikit-learn: machine learning in Python — scikit-learn 0.18.1 documentation, (2016).) Software's uses listed below.

- Python
- Scipy
- Numpy

- Matplotlab
- Pandas
- Scikit-learn

Summary

The quantitative study design was chosen for this paper. The data used was secondary data provided by the Canadian Institute for Cybersecurity. Using the data and processing them with ML Software, we can understand how different ML algorithms are effective in analyzing security logs.

Chapter IV: Data Presentation and Analysis

Introduction

The secondary data was collected from Canadian. The source of Network data used in this experiment is from academic resources. The “Canadian Institute for Cybersecurity datasets is used around the world by universities, private industry, and independent researchers.” (Brunswick). The actual network data collected was huge and at 34 Gigabytes in size. Processing Packet capture data .pcap files require time and high amounts of computing resources. In the next section, we will investigate the Data and understand it more.

Data Presentation

The data used for the study is.CSV files, which are captured during different days with different attack threats. The network information was captured at packet level with attacks simulated on the network ranging from DDoS attack, Port Scan, Infiltration attack, Web Attacks. The file size ranges are from 50,000 KB to 220,000 KB.

The data capture different attributes of network data like packet length, source port, destination port, packet length, network packet flags, header length. There are seventy-nine attributes that are captured.

File Names:

1. Friday-WorkingHours-Afternoon-DDos.pcap_ISCXX
2. Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX
3. Friday-WorkingHours-Morning.pcap_ISCX
4. Monday-WorkingHours.pcap_ISCX
5. Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX
6. Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX

7. Tuesday-WorkingHours.pcap_ISCX
8. Wednesday-workingHours.pcap_ISCX

Data Analysis

Data analysis was done by Machine learning Algorithms on different attack data attributes, out of seventy-nine variables present in data I have selected a few depending on the focused analysis. The data analysis is focused on a few attributes, like below.

Algorithm Accuracy equals the number of attacks detected divided by the total number of attacks in data.

Focused attributes are:

- Packet length
- Packet size
- Flow duration
- Header flags

Equations:

$$\textbf{Algorithm Accuracy} = \frac{\textbf{Number of Attacks Detected}}{\textbf{Total Number of Attacks in Data}}$$

Algorithm Analysis Time = Total Time Taken for Algorithm to Analyze Data

Algorithm Efficiency is determined by comparing **Algorithm Accuracy Rank** and **Algorithm Analysis Time** of all algorithms and select the algorithm that ranks higher.

Weightage and Ranking of Algorithms

In network security, both accuracy and time have equal importance when considering the detection component of the systems. If a system has high accuracy in detecting the anomaly but requires a high amount of time, such a system cannot be leveraged in security. During a security incident, the amount of time it takes to detect a breach has a high impact on the risk of the breach. If we focus more on time and give less weightage to accuracy, it results in too many false positives degrading the trust in the system and reliability of security measures to effectively detect a security incident. Hence when it comes to network security, both the time is taken for detection and accuracy of the result have high weightage.

The final scores are measured from a scale of 0.00 to 1.00, where scores closer to 0.00 received low weightage and score closer 1.00 will receive higher weightage. The algorithm with a higher score will be awarded rank one and will continue in descending order, and when two algorithms receive the same score, the rank will be awarded the same to both. The final ranking is determined by combining both accuracy rank and time rank, and the algorithm with the lowest numeric rank can be concluded as the best performing algorithm from the list algorithms.

This ranking provides us better visibility and options to select an algorithm based on our needs. If a security developer is creating a security tool, the user can assign his own weightage, whether if the user prefers an algorithm with higher accuracy or an algorithm with a rapid analysis time.

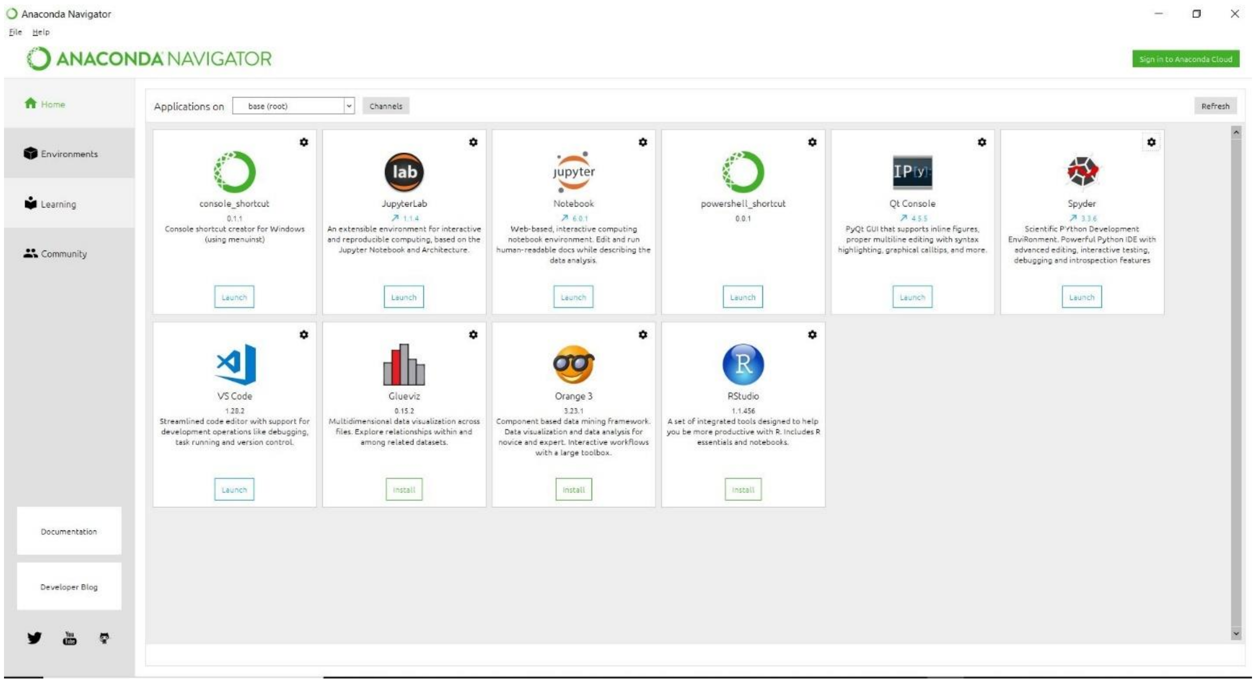
Software analysis Steps

Setting up the Test Lab

We Install the Anaconda Machine learning analysis software on our System – Windows 10 Operation System. Figure 1 shows the Anaconda User interface which is a collection of multiple data and machine learning software's

Figure 1

Setting up the Test Lab



console_shortcut
0.1.1
Console shortcut creator for Windows (using menuinst)

JupyterLab
1.1.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

powershell_shortcut
0.0.1

Qt Console
4.5.5
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
3.3.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code
1.28.2
Streamlined code editor with support for development operations like debugging, task running and version control.

Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.

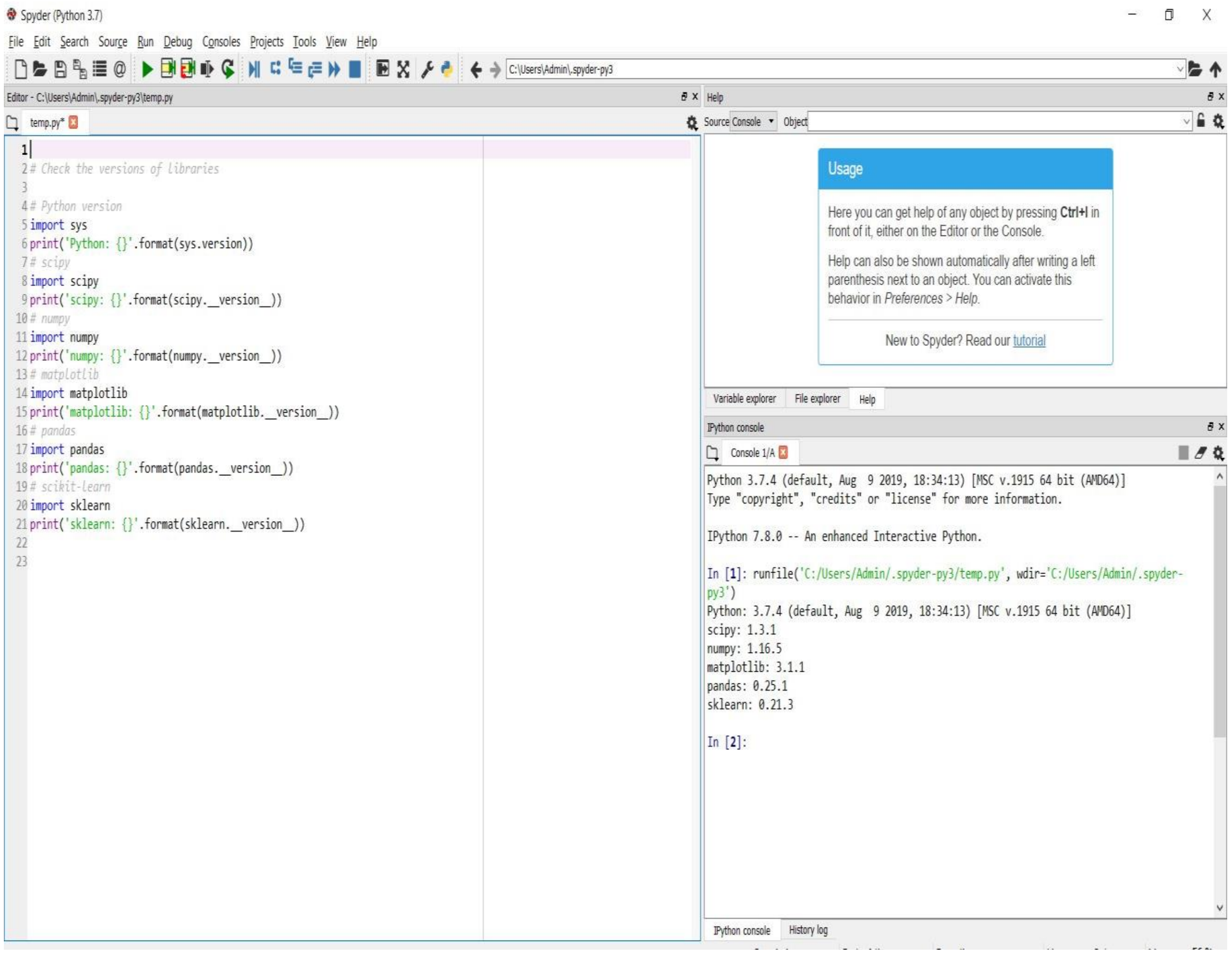
Orange 3
3.23.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Importing Software: After installing the anaconda, the below image shows you importing different machine learning analysis software(s). Anaconda has a huge set of libraries of this software, out of which we select the software(s) which are required for our study of network attack data. Figure 2 provides an understanding of how the system imports data into the system.

Figure 2

Importing Software


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Admin\.spyder-py3
Editor - C:\Users\Admin\.spyder-py3\temp.py
temp.py*
Help
Source Console Object
Variable explorer File explorer Help
IPython console
Console 1/A
IPython console History log

```python

# Check the versions of libraries

# Python version
import sys
print('Python: {}'.format(sys.version))
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
# matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
# pandas
import pandas
print('pandas: {}'.format(pandas.__version__))
# scikit-learn
import sklearn
print('sklearn: {}'.format(sklearn.__version__))
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help.*

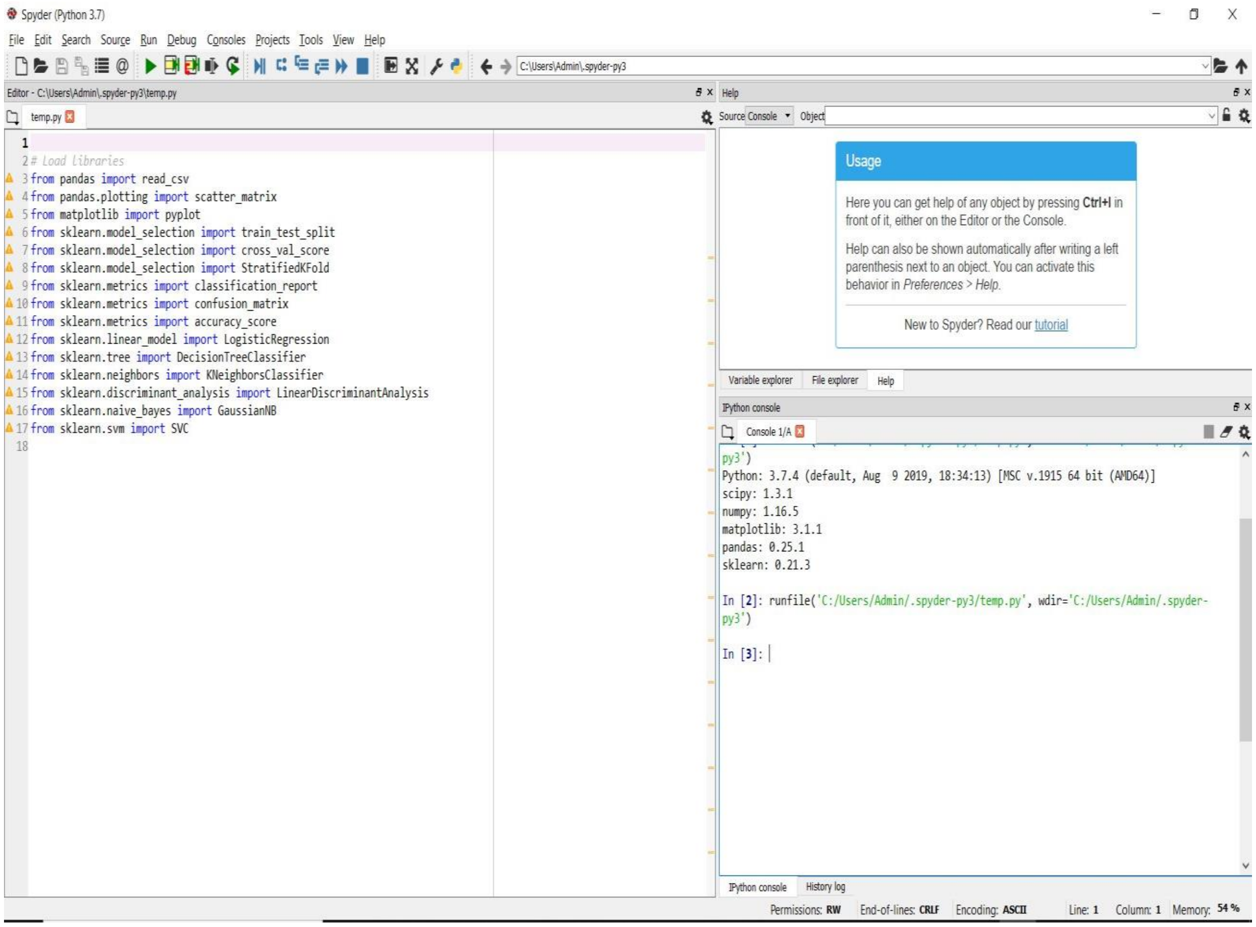
New to Spyder? Read our tutorial

```
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Admin/.spyder-py3/temp.py', wdir='C:/Users/Admin/.spyder-
py3')
Python: 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
scipy: 1.3.1
numpy: 1.16.5
matplotlib: 3.1.1
pandas: 0.25.1
sklearn: 0.21.3

In [2]:
```

Importing machine learning algorithms: Once we have imported the software, now we need to import the algorithms that we are going to use for analysis. Figure 3 shows the algorithms being imported into our test setup.

Figure 3

Importing machine learning algorithms


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Admin\.spyder-py3
Editor - C:\Users\Admin\.spyder-py3\temp.py
temp.py
Help
Source Console Object
Variable explorer File explorer Help
IPython console
Console 1/A
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 1 Column: 1 Memory: 54 %

```python

# Load libraries
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help.*

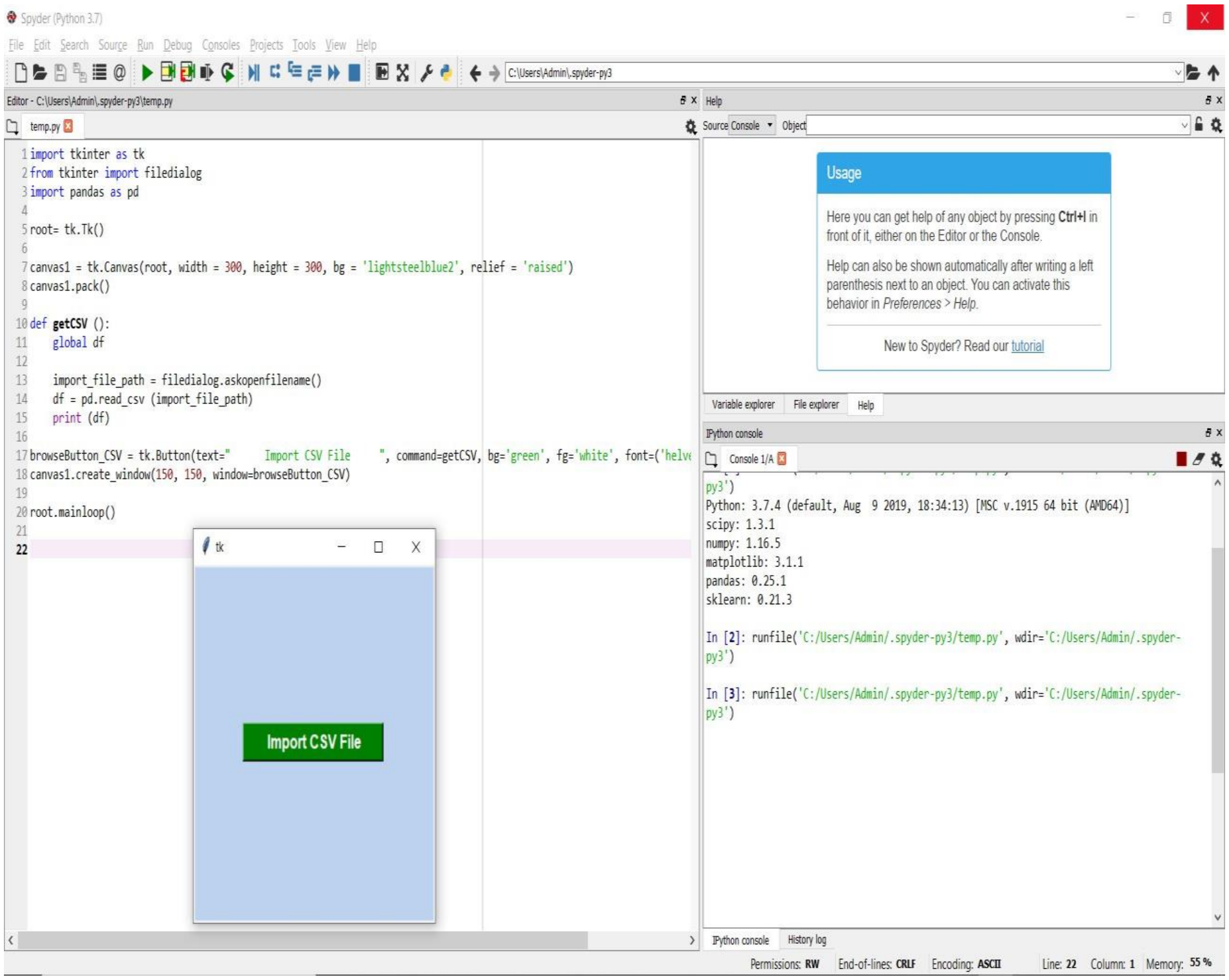
New to Spyder? Read our tutorial

```
py3')
Python: 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
scipy: 1.3.1
numpy: 1.16.5
matplotlib: 3.1.1
pandas: 0.25.1
sklearn: 0.21.3

In [2]: runfile('C:/Users/Admin/.spyder-py3/temp.py', wdir='C:/Users/Admin/.spyder-
py3')

In [3]:
```

Importing Network data into the test environment: The test environment has all the required software and algorithms installed now we need to import the test data into our testing environment. Figure 4 shows the importing of test data into the testing environment.

Figure 4

Importing Network data into the test environment



```
import tkinter as tk
from tkinter import filedialog
import pandas as pd

root= tk.Tk()

canvas1 = tk.Canvas(root, width = 300, height = 300, bg = 'lightsteelblue2', relief = 'raised')
canvas1.pack()

def getCSV ():
    global df

    import_file_path = filedialog.askopenfilename()
    df = pd.read_csv (import_file_path)
    print (df)

browseButton_CSV = tk.Button(text="      Import CSV File     ", command=getCSV, bg='green', fg='white', font=('helve
canvas1.create_window(150, 150, window=browseButton_CSV)

root.mainloop()
```

```
py3')
Python: 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
scipy: 1.3.1
numpy: 1.16.5
matplotlib: 3.1.1
pandas: 0.25.1
sklearn: 0.21.3

In [2]: runfile('C:/Users/Admin/.spyder-py3/temp.py', wdir='C:/Users/Admin/.spyder-
py3')

In [3]: runfile('C:/Users/Admin/.spyder-py3/temp.py', wdir='C:/Users/Admin/.spyder-
py3')
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

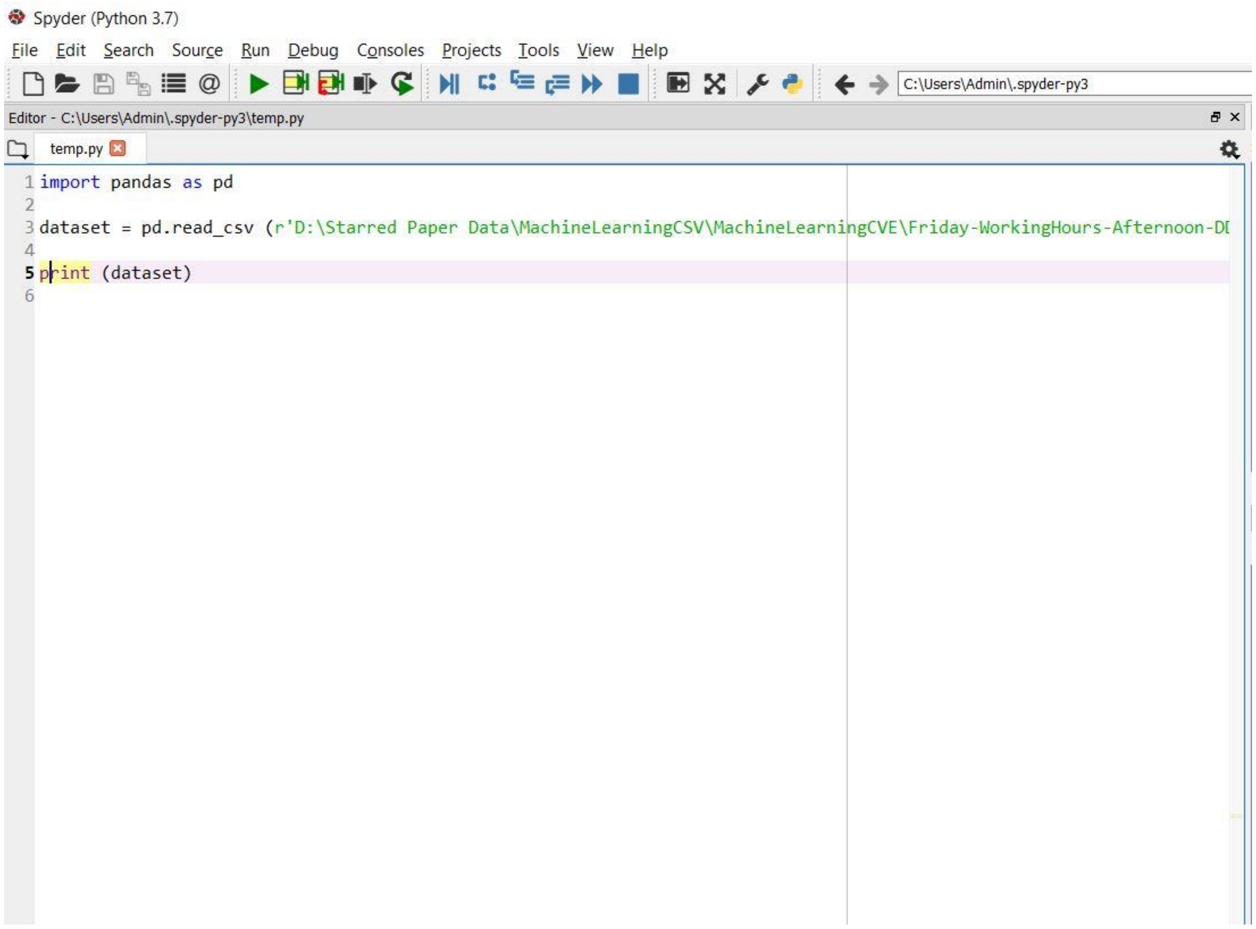
New to Spyder? Read our tutorial

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Admin\.spyder-py3
Editor - C:\Users\Admin\.spyder-py3\temp.py
temp.py
Help
Source Console Object
Variable explorer File explorer Help
IPython console
Console 1/A
tk
Import CSV File
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 22 Column: 1 Memory: 55 %

Selecting different test data files: Figure 5 shows us the code that can be modified to import different test files into our test environment. Using this code, we can analyze different sets of data with the same data attributes.

Figure 5

Selecting different test data files

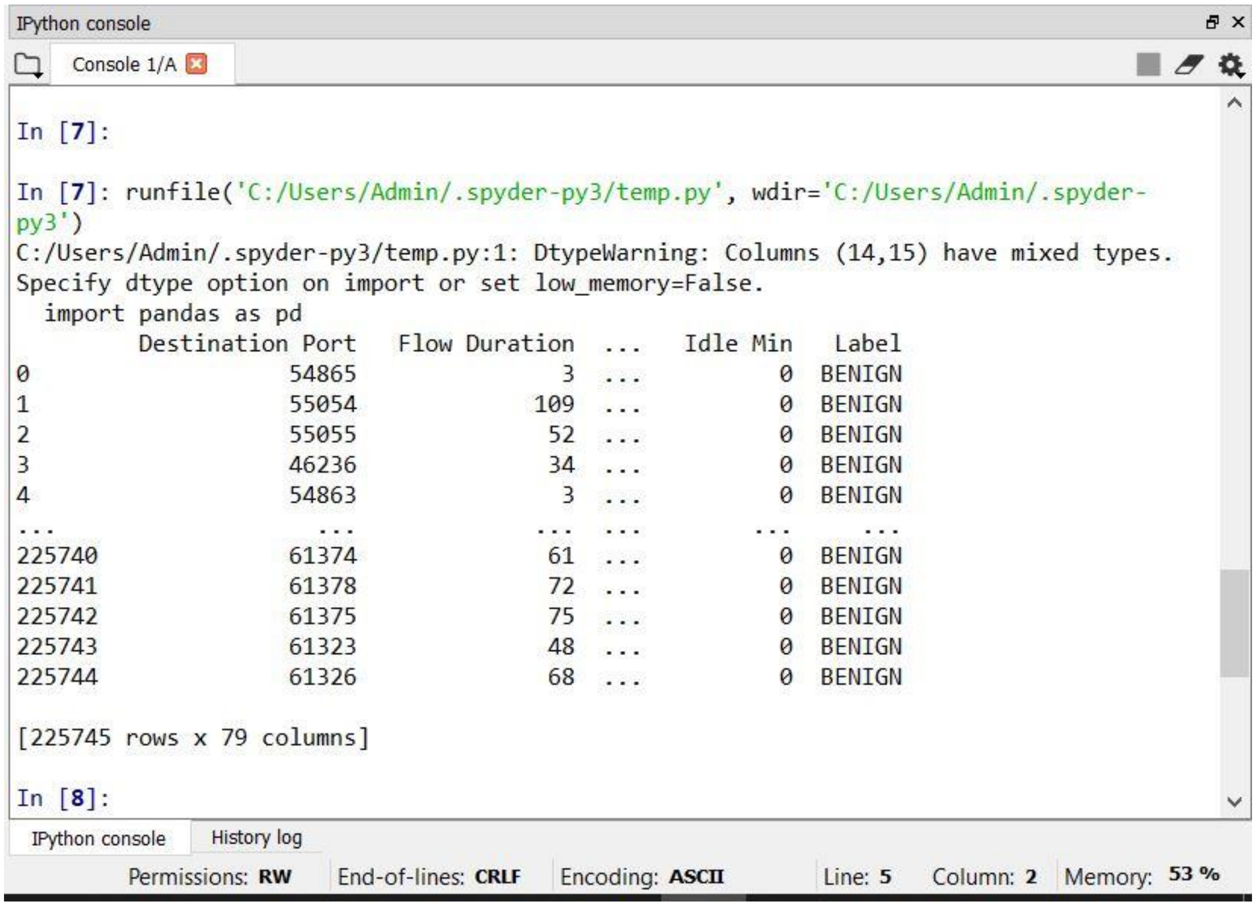

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Admin\.spyder-py3
Editor - C:\Users\Admin\.spyder-py3\temp.py
temp.py

```python
import pandas as pd

dataset = pd.read_csv (r'D:\Starred Paper Data\MachineLearningCSV\MachineLearningCVE\Friday-WorkingHours-Afternoon-DI

print (dataset)
```

Validating the test files and data elements: Validation of test files after importing data is an integral part of making sure we are ingesting accurate test files and data elements into the test environment. Figure 6 shows that the test file contains seventy-nine columns and two hundred and twenty-five thousand seven forty-five rows in the test file.

Figure 6

Validating the test files and data elements



IPython console

Console 1/A

```
In [7]:

In [7]: runfile('C:/Users/Admin/.spyder-py3/temp.py', wdir='C:/Users/Admin/.spyder-
py3')
C:/Users/Admin/.spyder-py3/temp.py:1: DtypeWarning: Columns (14,15) have mixed types.
Specify dtype option on import or set low_memory=False.
  import pandas as pd
        Destination Port  Flow Duration  ...  Idle Min   Label
0                  54865              3  ...         0  BENIGN
1                  55054            109  ...         0  BENIGN
2                  55055             52  ...         0  BENIGN
3                  46236             34  ...         0  BENIGN
4                  54863              3  ...         0  BENIGN
...                  ...            ...  ...       ...     ...
225740             61374             61  ...         0  BENIGN
225741             61378             72  ...         0  BENIGN
225742             61375             75  ...         0  BENIGN
225743             61323             48  ...         0  BENIGN
225744             61326             68  ...         0  BENIGN

[225745 rows x 79 columns]

In [8]:
```

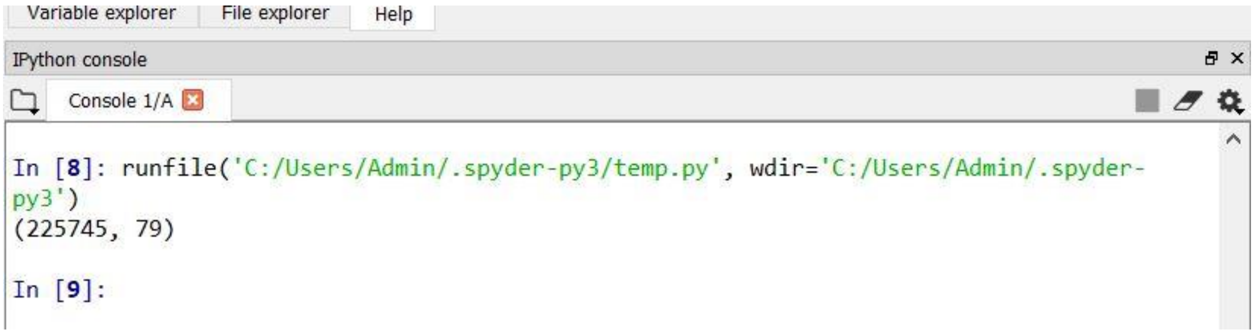
IPython console | History log

Permissions: **RW** End-of-lines: **CRLF** Encoding: **ASCII** Line: **5** Column: **2** Memory: **53 %**

Figure 7 can provide a quick information snapshot of the shape of the data. The data analysis takes a long time with the size of the data. To prevent operating system crashes due to running out of memory and computing space, sometimes data is truncated to manageable limits by reducing the number of rows included in the analysis of a file. This should be consistent with all the test files.

Figure 7

Shape of Data



```
In [8]: runfile('C:/Users/Admin/.spyder-py3/temp.py', wdir='C:/Users/Admin/.spyder-
py3')
(225745, 79)

In [9]:
```

Summary

The data presentation and data analysis are completed as a quantitative study approach. The secondary data has a total of seventy-nine attributes, out of which we focused on four key attributes. The data analysis approach was discussed, and the formula to determine the best algorithm and approach was briefly discussed in this chapter.The study test environment was provided with figures to provide a better understanding of how the test environment was setup.

Chapter V: Results, Conclusion, and Recommendations

Introduction

On reviewing the results of the study. We have tested multiple files of network data, and we will discuss in detail the results of the study. We will look into the output and rank algorithms based on their accuracy and time took for analysis and determine the most suitable algorithm that provides both the accuracy and also takes less time for analysis. The results will provide data that makes us select the algorithm based on our requirements based on different use cases. Results will be captured in three tables to understand the different outputs.

Based on the outputs, we can conclude determining which algorithm can be used for different security log data analysis. We will also briefly discuss different future works, and that can leverage this study to improve and also apply to varied data analysis approaches.

Results:

Based on the analysis, I have collated the final results based on the quantitative study methodology. Below you can find the legend of what different algorithm abbreviations mean in the following results tables.

Legend:

1. Logistic Regression - LR
2. Linear Discriminant Analysis -LDA
3. Support Vector Machine - SVM
4. Naive Bayes - NB
5. K- Nearest Neighbors – KNN
6. Random Forest -RF

1. What is the accuracy of different machine learning algorithms in detecting the attacks from network log data?

The results in Table 2 show that the KNN algorithm is most efficient in detecting the attacks from the test data. KNN (K- Nearest Neighbors) has the highest accuracy in detection with 0.98 accuracies. RF (Random Forest) comes in second with an accuracy of 0.97 accuracies.

This shows that the two algorithms were able to detect most of the know attacks with an accuracy of 0.98 and 0.97. This is a statistically significant result as these results are above 0.95 statistical significance.

Table 2

Algorithm Accuracy ranking

Algorithm	**Accuracy**	**Algorithm ranking**
KNN	**0.98**	**1**
RF	**0.97**	**2**
NB	**0.88**	**3**
LR	**0.77**	**4**
LDA	**0.72**	**5**
SVM	**0.65**	**6**

The other algorithms were not able to detect the attacks as accurately as KNN and RF.

2. What is the Time taken (Seconds) for different machine learning algorithms to analyze the attacks from network log data?

The results in Table 3 show that RF (Random Forest) algorithm is the quickest with a time of **78.95 seconds** in analyzing the attacks from the test data. KNN (K- Nearest Neighbors), which has the highest accuracy in detection with 0.98 accuracy falls behind significantly with time of **2564.25 seconds**. SVM comes in second with a time of 115.85 **seconds** of 0.97. This shows that the two algorithms were able to analyze the data taking the least amount of time.

Table 3

Algorithm Time ranking

Algorithm	**Time in Seconds**	**Algorithm ranking T**
KNN	**2564.25**	**6**
RF	**78.95**	**1**
NB	**1654.45**	**5**
LR	**356.87**	**3**
LDA	**1155.29**	**4**
SVM	**115.85**	**2**

3. How does each machine learning algorithm stand based on the accuracy and time taken to analyze the test data?

To identify the best Machine learning algorithm for analysis of security log data, we need to add the accuracy rank and time rank of algorithms to get an overall rank of the algorithm. This total ranking was represented in Table 4.

Table 4

Algorithm Total Ranking

Algorithm	**Accuracy Rank**	**Time Rank**	**Total Rank**
KNN	**1**	**6**	**7**
RF	**2**	**1**	**3**
NB	**3**	**5**	**8**
LR	**4**	**3**	**7**
LDA	**5**	**4**	**9**
SVM	**6**	**2**	**8**

4. What is the best Machine learning algorithm to use to analyze the security log data?

Based on the results of the ranking, we can determine that the Random forest algorithm is the winner with high rank both in accuracy and time required to analyze the data. This shows while deciding the algorithm, security analysts can prefer the **Random forest** Algorithm to get the best of both accuracy and speed in detecting the attacks. The final ranking was represented in Table 5**.**

Table 5

Algorithm Final Ranking

Algorithm	**Total Rank**	**Final Rank**
RF	**3**	**1**
KNN	**7**	**2**
LR	**7**	**2**
NB	**8**	**3**
SVM	**8**	**3**
LDA	**9**	**4**

Conclusion

The aim of this starred paper was to analyze different machine learning algorithms using security log data and to identify the best algorithm, which is both accurate and fastest in detecting the attacks by analyzing security data.

In this paper, we reviewed different security risk assessments and machine learning algorithms and code. We brought together the security risk and machine learning algorithms to analyze security data by creating a test environment.For any organization detecting the attacks accurately and quickly is the most important factor in reducing the risk of a security breach. No amount of systems, standards, compliance guidelines can assure a complete hundred percent guarantee of avoiding the security breach. The assumption is security breaches will happen, and the best way to reduce the risk is to detect the attack early and implement the mitigation procedures.

The early detection of the attack will provide security professionals the time to reduce the impact and safeguard the organization. We have discussed in risk assessment how different security guidelines are implemented within the organization, which slow and provide more time and increase the effort of hackers in getting access to core organization systems. This will be achieved by making sure the attack is detected early and once creating multiple layers of security so that it becomes difficult for attackers as risk procedures prevent the Kill chain of attackers by slowing and stopping the attack at different levels.

The paper provides the reader with preliminary information in understanding and implementing a basic security process and on how to reduce the number of man-hours required to analyze security data. This is achieved by leveraging machine learning algorithms like Random Forest.

Future Work

This system can be improved and customized to different requirements; some of the future work thoughts are as below:

Information security:

Creating an automated procedure to review the network data to find anomalies that can alert security professionals to deep dive further. Leveraging and combining different Machine learning algorithms to create a system that can detect multiple threats.

Epidemiology:

Like network data, viruses, and diseases spread fast. So, to better counter the spread, it is important to detect the spread of viruses by analyzing data. If we can collate the data from hospitals across the globe just by taking the attributes new admission and type of illness whenever we see a spike of particular illness spreading significantly in association to geographic data, this can indicate a possible epidemic and can alert health professionals and administrators to review the situation fast.

Government Administration:

Similarly, government institutions can collect data from citizens, and masking personally identifiable data can look to travel patterns of citizens, and also matching the hospital data can identify which locations the chances of diseases can spread and can prepare before spread increases. Data can also assist in organizing rules and personal to high-risk areas for slowing down the spread of new diseases or riots. As we are using algorithms, we can identify patterns that are not visible for genera individuals. This provides the users actionable intelligence for reacting to any threat faster and efficiently.

References

Alspaugh, S., Chen, B., Lin, J., Ganapathi, A., Hearst, M., & Katz, R. (2014). Analyzing log an analysis: An empirical study of user log mining. In 28th Large Installation System Administration Conference (LISA14), (pp. 62-77).

Coates, A., Lee, H., & Ng, A. Y. (2011). An analysis of single-layer networks in unsupervised feature learning. Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, PMLR 15:215-223, 1-9. http://proceedings.mlr.press/v15/coates11a.html

Department of Defense. (1983, August). Department of Defense Trusted Computer System Evaluation Criteria, (Publication No. DoD 5200.28-STD) https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/dod85.pdf

Li, K. L., Huang, H. K., Tian, S. F., & Xu, W. (2003, November). Improving one-class SVM for anomaly detection. Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693), 5, 3077-3081 Vol.5.

Li, W. (2013). Automatic Log Analysis using Machine Learning: Awesome Automatic Log Analysis version 2.0. [Dissertation, Uppsala University, Disciplinary Domain of Science and Technology, Mathematics and Computer Science, Department of Information Technology.]. Uppsala University Publications. http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-211626

NIST -National Institute of Standards and Technology Special Publication 800-30, Risk Management Guide for Information Technology Systems (July 2002) https://csrc.nist.gov/publications/detail/sp/800-30/archive/2002-07-01

Ma, P. (2003). Log Analysis-Based Intrusion Detection via Unsupervised Learning. Master of Science School of Informatics,University of Edinburgh. 1-77. http://swsoft.nsu.ru/~brylev/loganalyzer/documentation/log.pdf

Manevitz, L. M., & Yousef, M. (2001). One-class SVMs for document classification. Journal of Machine Learning Research, 2(Dec), 139-154.

Mika, S., Ratsch, G., Tsuda, K., & Scholkopf, B. (2001). An introduction to kernel-based learning algorithms. IEEE transactions on neural networks, 12(2), 181-201.

Ray Sunil. (2017, September). Essentials of Machine Learning Algorithms (with Python and R Codes), https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

scikit-learn: machine learning in Python — scikit-learn 0.18.1 documentation. (2016). Scikit-learn.org. 2 December 2016, http://scikit-learn.org/stable/

Joseph, Gitau Mbugua (2016). Automated log analysis using AI: Intelligent Intrusion Detection System. https://www.jooust.ac.ke/projects/siis/2016/JGM-10-2016.pdf

Appendix

Table 6

Data Attributes in CSV File

Attribute Names	Attribute Data Sample 1	Attribute Data Sample 2	Attribute Data Sample 3	Attribute Data Sample 4
Destination Port	22	22	22	22
Flow Duration	1266342	1319353	1319353	1319353
Total Fwd Packets	41	41	41	41
Total Backward Packets	44	44	44	44
Total Length of Fwd Packets	2664	2664	2664	2664
Total Length of Bwd Packets	6954	6954	6954	6954
Fwd Packet Length Max	456	456	456	456
Fwd Packet Length Min	0	0	0	0
Fwd Packet Length Mean	64.97560976	64.97560976	64.97560976	64.97560976
Fwd Packet Length Std	109.864573	109.864573	109.864573	109.864573
Bwd Packet Length Max	976	976	976	976
Bwd Packet Length Min	0	0	0	0
Bwd Packet Length Mean	158.0454545	158.0454545	158.0454545	158.0454545
Bwd Packet Length Std	312.6752498	312.6752498	312.6752498	312.6752498

Flow Bytes/s	7595.10464	7289.93681	7289.93681	7289.93681
Flow Packets/s	67.12246771	64.42551766	64.42551766	64.42551766
Flow IAT Mean	15075.5	15706.58333	15706.58333	15706.58333
Flow IAT Std	104051.3997	104861.8701	104861.8701	104861.8701
Flow IAT Max	948537	955790	955790	955790
Flow IAT Min	0	1	1	1
Fwd IAT Total	1266342	1319353	1319353	1319353
Fwd IAT Mean	31658.55	32983.825	32983.825	32983.825
Fwd IAT Std	159355.2595	159247.9008	159247.9008	159247.9008
Fwd IAT Max	996324	996423	996423	996423
Fwd IAT Min	2	1	1	1
Bwd IAT Total	317671	363429	363429	363429
Bwd IAT Mean	7387.697674	8451.837209	8451.837209	8451.837209
Bwd IAT Std	19636.44809	21337.26261	21337.26261	21337.26261
Bwd IAT Max	104616	104815	104815	104815
Bwd IAT Min	1	1	1	1
Fwd PSH Flags	0	0	0	0
Bwd PSH Flags	0	0	0	0
Fwd URG Flags	0	0	0	0
Bwd URG Flags	0	0	0	0
Fwd Header Length	1328	1328	1328	1328
Bwd Header Length	1424	1424	1424	1424

Fwd Packets/s	32.37671972	31.07583793	31.07583793	31.07583793
Bwd Packets/s	34.74574799	33.34967973	33.34967973	33.34967973
Min Packet Length	0	0	0	0
Max Packet Length	976	976	976	976
Packet Length Mean	111.8372093	111.8372093	111.8372093	111.8372093
Packet Length Std	239.6868477	239.6868477	239.6868477	239.6868477
Packet Length Variance	57449.78495	57449.78495	57449.78495	57449.78495
FIN Flag Count	0	0	0	0
SYN Flag Count	0	0	0	0
RST Flag Count	0	0	0	0
PSH Flag Count	1	1	1	1
ACK Flag Count	0	0	0	0
URG Flag Count	0	0	0	0
CWE Flag Count	0	0	0	0
ECE Flag Count	0	0	0	0
Down/Up Ratio	1	1	1	1
Average Packet Size	113.1529412	113.1529412	113.1529412	113.1529412
Avg Fwd Segment Size	64.97560976	64.97560976	64.97560976	64.97560976
Avg Bwd Segment Size	158.0454545	158.0454545	158.0454545	158.0454545
Fwd Header Length	1328	1328	1328	1328
Fwd Avg Bytes/Bulk	0	0	0	0
Fwd Avg Packets/Bulk	0	0	0	0

Fwd Avg Bulk Rate	0	0	0	0
Bwd Avg Bytes/Bulk	0	0	0	0
Bwd Avg Packets/Bulk	0	0	0	0
Bwd Avg Bulk Rate	0	0	0	0
Subflow Fwd Packets	41	41	41	41
Subflow Fwd Bytes	2664	2664	2664	2664
Subflow Bwd Packets	44	44	44	44
Subflow Bwd Bytes	6954	6954	6954	6954
Init_Win_bytes_forward	29200	29200	29200	29200
Init_Win_bytes_backward	243	243	243	243
act_data_pkt_fwd	24	24	24	24
min_seg_size_forward	32	32	32	32
Active Mean	0	0	0	0
Active Std	0	0	0	0
Active Max	0	0	0	0
Active Min	0	0	0	0
Idle Mean	0	0	0	0
Idle Std	0	0	0	0
Idle Max	0	0	0	0
Idle Min	0	0	0	0
Label	BENIGN	BENIGN	BENIGN	BENIGN

Program application Code: (Python 3.27)

```
# Check the versions of libraries

# Python version

import sys

print('Python: {}'.format(sys.version))

# scipy

import scipy

print('scipy: {}'.format(scipy.__version__))

# numpy

import numpy

print('numpy: {}'.format(numpy.__version__))

# matplotlib

import matplotlib

print('matplotlib: {}'.format(matplotlib.__version__))

# pandas

import pandas

print('pandas: {}'.format(pandas.__version__))

# scikit-learn

import sklearn

print('sklearn: {}'.format(sklearn.__version__))

# Load libraries

import pandas as pd

from pandas import read_csv
from pandas.plotting import scatter_matrix

from matplotlib import pyplot

from sklearn.model_selection import train_test_split

from sklearn.model_selection import cross_val_score

from sklearn.model_selection import StratifiedKFold

from sklearn.metrics import classification_report

from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn.naive_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.model_selection import train_test_split

from sklearn import preprocessing

from sklearn.ensemble import RandomForestRegressor

from sklearn.pipeline import make_pipeline

from sklearn.model_selection import GridSearchCV

from sklearn.metrics import mean_squared_error, r2_score

from sklearn.externals import joblib

import pandas as pd
```

```
dataset = pd.read_csv (r'C:\Users\Admin\Desktop\MachineLearningCVE\Friday-WorkingHours-Afternoon-DDos.pcap_ISCXX.csv')
print (dataset)
# shape
print (dataset.shape)
# head
print(dataset.head(20))
# descriptions
print(dataset.describe())
print (dataset.dtypes)
# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(4,8), sharex=False, sharey=False)
pyplot.show()
# histograms
dataset.hist()
pyplot.show()
# Split-out validation dataset
array = dataset.values
X = array[:,0:4]
y = array[:,4]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y, test_size=0.20, random_state=1)
```

```
# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
# evaluate each model in turn
results = []
names = []
for name, model in models:
	kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
	cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
	results.append(cv_results)
	names.append(name)
	print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
pyplot.boxplot(results, labels=names)
pyplot.title('Algorithm Comparison')
pyplot.show()
```

```
# Make predictions on validation dataset

model = SVC(gamma='auto')

model.fit(X_train, Y_train)

predictions = model.predict(X_validation)


# Evaluate predictions

print(accuracy_score(Y_validation, predictions))

print(confusion_matrix(Y_validation, predictions))

print(classification_report(Y_validation, predictions))
```

Machine Learning Code

Linear Regression:

Python Code:

```
“#Import Library

#Import other necessary libraries like pandas, numpy...

from sklearn import linear_model

#Load Train and Test datasets

#Identify feature and response variable(s) and values must be numeric and numpy arrays

x_train=input_variables_values_training_datasets

y_train=target_variables_values_training_datasets

x_test=input_variables_values_test_datasets

# Create linear regression object

linear = linear_model.LinearRegression()
```

```
# Train the model using the training sets and check score
linear.fit(x_train, y_train)
linear.score(x_train, y_train)
#Equation coefficient and Intercept
print('Coefficient: \n', linear.coef_)
print('Intercept: \n', linear.intercept_)
#Predict Output
predicted= linear.predict(x_test)” (Ray, 2017)
```

R Code

```
“#Load Train and Test datasets
#Identify feature and response variable(s) and values must be numeric and numpy arrays
x_train <- input_variables_values_training_datasets
y_train <- target_variables_values_training_datasets
x_test <- input_variables_values_test_datasets
x <- cbind(x_train,y_train)
# Train the model using the training sets and check score
linear <- lm(y_train ~ ., data = x)
summary(linear)
#Predict Output
predicted= predict(linear,x_test)”. (Ray, 2017)
```

Logistic Regression

Python Code

```
“#Import Library
from sklearn.linear_model import LogisticRegression
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset
# Create logistic regression object
model = LogisticRegression()
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Equation coefficient and Intercept
print('Coefficient: \n', model.coef_)
print('Intercept: \n', model.intercept_)
#Predict Output
predicted= model.predict(x_test)” (Ray, 2017)
```

R Code

```
“x <- cbind(x_train,y_train)
# Train the model using the training sets and check score
logistic <- glm(y_train ~ ., data = x,family='binomial')
summary(logistic)
#Predict Output
predicted= predict(logistic,x_test)” (Ray, 2017)
```

3. Decision Tree

Python Code

```
“#Import Library
#Import other necessary libraries like pandas, numpy...
from sklearn import tree
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset
# Create tree object
model = tree.DecisionTreeClassifier(criterion='gini') # for classification, here you can change the algorithm as gini or entropy (information gain) by default it is gini
# model = tree.DecisionTreeRegressor() for regression
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)” (Ray, 2017)
```

R Code

```
“library(rpart)
x <- cbind(x_train,y_train)
# grow tree
fit <- rpart(y_train ~ ., data = x,method="class")
summary(fit)
#Predict Output
```

```
predicted= predict(fit,x_test)” (Ray, 2017)
```

SVM (Support Vector Machine)

Python Code

```
“#Import Library
from sklearn import svm
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset
# Create SVM classification object
model = svm.svc() # there is various option associated with it, this is simple for classification. You can refer link, for mo# re detail.
# Train the model using the training sets and check score
model.fit(X, y)
model.score(X, y)
#Predict Output
predicted= model.predict(x_test)” (Ray, 2017)
```

R Code

```
“library(e1071)
x <- cbind(x_train,y_train)
# Fitting model
fit <-svm(y_train ~ ., data = x)
summary(fit)
#Predict Output
```

```
predicted= predict(fit,x_test)” (Ray, 2017)
```

Naive Bayes

Python Code

```
“#Import Library
from sklearn.naive_bayes import GaussianNB
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset
# Create SVM classification object model = GaussianNB() # there is another distribution for multinomial classes like Bernoulli Naive Bayes, Refer link
# Train the model using the training sets and check score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)” (Ray, 2017)
```

R Code

```
“library(e1071)
x <- cbind(x_train,y_train)
# Fitting model
fit <-naiveBayes(y_train ~ ., data = x)
summary(fit)
#Predict Output
predicted= predict(fit,x_test)” (Ray, 2017)
```

KNN (K- Nearest Neighbors)

Python Code

```
“#Import Library
from sklearn.neighbors import KNeighborsClassifier
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset
# Create KNeighbors classifier object model
KNeighborsClassifier(n_neighbors=6) # default value for n_neighbors is 5
# Train the model using the training sets and check score
model.fit(X, y)
#Predict Output
predicted= model.predict(x_test)” (Ray, 2017)
```

R Code

```
“library(knn)
x <- cbind(x_train,y_train)
# Fitting model
fit <-knn(y_train ~ ., data = x,k=5)
summary(fit)
#Predict Output
predicted= predict(fit,x_test)” (Ray, 2017)
```

7. K-Means

Python Code

“#Import Library

from sklearn.cluster import KMeans

#Assumed you have, X (attributes) for training data set and x_test(attributes) of test_dataset

# Create KNeighbors classifier object model

k_means = KMeans(n_clusters=3, random_state=0)

# Train the model using the training sets and check score

model.fit(X)

#Predict Output

predicted= model.predict(x_test)” (Ray, 2017)

R Code

“library(cluster)

fit <- kmeans(X, 3) # 5 cluster solution” (Ray, 2017)

Random Forest

Python

“#Import Library

from sklearn.ensemble import RandomForestClassifier

#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset

# Create Random Forest object

model= RandomForestClassifier()

# Train the model using the training sets and check score

model.fit(X, y)

#Predict Output

predicted= model.predict(x_test)” (Ray, 2017)

R Code

“library(randomForest)

x <- cbind(x_train,y_train)

# Fitting model

fit <- randomForest(Species ~ ., x,ntree=500)

summary(fit)

#Predict Output

predicted= predict(fit,x_test)” (Ray, 2017)

9. Dimensionality Reduction Algorithms

Python Code

“#Import Library

from sklearn import decomposition

#Assumed you have training and test data set as train and test

# Create PCA obeject pca= decomposition.PCA(n_components=k) #default value of k =min(n_sample, n_features)

# For Factor analysis

#fa= decomposition.FactorAnalysis()

# Reduced the dimension of training dataset using PCA

train_reduced = pca.fit_transform(train)

#Reduced the dimension of test dataset

test_reduced = pca.transform(test)” (Ray, 2017)

R Code

“library(stats)

pca <- princomp(train, cor = TRUE)

train_reduced <- predict(pca,train)

test_reduced <- predict(pca,test)” (Ray, 2017)

Gradient Boosting Algorithms

GBM

Python Code

“#Import Library

from sklearn.ensemble import GradientBoostingClassifier

#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset

# Create Gradient Boosting Classifier object

model= GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1, random_state=0)

# Train the model using the training sets and check score

model.fit(X, y)

#Predict Output

predicted= model.predict(x_test)” (Ray, 2017)

R Code

```
“library(caret)
x <- cbind(x_train,y_train)
# Fitting model
fitControl <- trainControl( method = "repeatedcv", number = 4, repeats = 4)
fit <- train(y ~ ., data = x, method = "gbm", trControl = fitControl,verbose = FALSE)
predicted= predict(fit,x_test,type= "prob") “ (Ray, 2017)
```

XGBoost

Python Code:

```
“from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X = dataset[:,0:10]
Y = dataset[:,10:]
seed = 1
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=seed)
model = XGBClassifier()
model.fit(X_train, y_train)
#Make predictions for test data
y_pred = model.predict(X_test)” (Ray, 2017)
```

R Code:

```
“require(caret) from R
```

x <- cbind(x_train,y_train)

# Fitting model

TrainControl <- trainControl(method = "repeatedcv", number = 10, repeats = 4)

model<- train(y ~ ., data = x, method = "xgbLinear", trControl = TrainControl,verbose = FALSE)

OR

model<- train(y ~ ., data = x, method = "xgbTree", trControl = TrainControl,verbose = FALSE)

predicted <- predict(model, x_test) " (Ray, 2017)

LightGBM

Python Code:

"data = np.random.rand(500, 10) # 500 entities, each contains 10 features

label = np.random.randint(2, size=500) # binary target

train_data = lgb.Dataset(data, label=label)

test_data = train_data.create_valid('test.svm')

param = {'num_leaves':31, 'num_trees':100, 'objective':'binary'}

param['metric'] = 'auc'

num_round = 10

bst = lgb.train(param, train_data, num_round, valid_sets=[test_data])

bst.save_model('model.txt')

# 7 entities, each contains 10 features

data = np.random.rand(7, 10)

ypred = bst.predict(data)" (Ray, 2017)

R Code:

```
“library(RLightGBM)
data(example.binary)
#Parameters
num_iterations <- 100
config <- list(objective = "binary",  metric="binary_logloss,auc", learning_rate = 0.1, num_leaves
= 63, tree_learner = "serial", feature_fraction = 0.8, bagging_freq = 5, bagging_fraction = 0.8,
min_data_in_leaf = 50, min_sum_hessian_in_leaf = 5.0)
#Create data handle and booster
handle.data <- lgbm.data.create(x)
lgbm.data.setField(handle.data, "label", y)
handle.booster <- lgbm.booster.create(handle.data, lapply(config, as.character))
#Train for num_iterations iterations and eval every 5 steps
lgbm.booster.train(handle.booster, num_iterations, 5)
#Predict
pred <- lgbm.booster.predict(handle.booster, x.test)
#Test accuracy
sum(y.test == (y.pred > 0.5)) / length(y.test)
#Save model (can be loaded again via lgbm.booster.load(filename))
lgbm.booster.save(handle.booster, filename = "/tmp/model.txt")
require(caret)
require(RLightGBM)
data(iris)
```

```
model <-caretModel.LGBM()
fit <- train(Species ~ ., data = iris, method=model, verbosity = 0)
print(fit)
y.pred <- predict(fit, iris[,1:4])
library(Matrix)
model.sparse <- caretModel.LGBM.sparse()
#Generate a sparse matrix
mat <- Matrix(as.matrix(iris[,1:4]), sparse = T)
fit <- train(data.frame(idx = 1:nrow(iris)), iris$Species, method = model.sparse, matrix = mat,
verbosity = 0)
print(fit)” (Ray, 2017)
```

Catboost

Python Code:

```
“import pandas as pd
import numpy as np
from catboost import CatBoostRegressor
#Read training and testing files
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
#Imputing missing values for both train and test
train.fillna(-999, inplace=True)
test.fillna(-999,inplace=True)
```

```
#Creating a training set for modeling and validation set to check model performance
X = train.drop(['Item_Outlet_Sales'], axis=1)
y = train.Item_Outlet_Sales
from sklearn.model_selection import train_test_split
X_train, X_validation, y_train, y_validation = train_test_split(X, y, train_size=0.7,
random_state=1234)
categorical_features_indices = np.where(X.dtypes != np.float)[0]
#importing library and building model
from catboost import CatBoostRegressormodel=CatBoostRegressor(iterations=50, depth=3,
learning_rate=0.1, loss_function='RMSE')
model.fit(X_train, y_train,cat_features=categorical_features_indices,eval_set=(X_validation,
y_validation),plot=True)
submission = pd.DataFrame()
submission['Item_Identifier'] = test['Item_Identifier']
submission['Outlet_Identifier'] = test['Outlet_Identifier']
submission['Item_Outlet_Sales'] = model.predict(test)” (Ray, 2017)
```

R Code:

```
“set.seed(1)
require(titanic)
require(caret)
require(catboost)
tt <- titanic::titanic_train[complete.cases(titanic::titanic_train),]
data <- as.data.frame(as.matrix(tt), stringsAsFactors = TRUE)
```

```
drop_columns = c("PassengerId", "Survived", "Name", "Ticket", "Cabin")

x <- data[,!(names(data) %in% drop_columns)]y <- data[,c("Survived")]

fit_control <- trainControl(method = "cv", number = 4,classProbs = TRUE)

grid <- expand.grid(depth = c(4, 6, 8),learning_rate = 0.1,iterations = 100, l2_leaf_reg = 1e-3,
rsm = 0.95, border_count = 64)

report <- train(x, as.factor(make.names(y)),method = catboost.caret,verbose = TRUE, preProc =
NULL,tuneGrid = grid, trControl = fit_control)

print(report)

importance <- varImp(report, scale = FALSE)

print(importance)” (Ray, 2017)
```