CRANFIELD UNIVERSITY

HANLIN CHEN

WEB ROBOT DETECTION USING SUPERVISED LEARNING ALGORITHMS

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
MSc by Research

MSc
Academic Year: 2019 - 2020

Supervisor: Hongmei He
Associate Supervisor: Andrew Starr
June 2020

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
MSc by Research

MSc

Academic Year 2019 – 2020

HANLIN CHEN

Web Robot Detection using Supervised Learning Algorithms

Supervisor: Hongmei He
Associate Supervisor: Andrew Starr
June 2020

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc
(NB. This section can be removed if the award of the degree is based solely on examination of the thesis)

ABSTRACT

Web robots or Web crawlers have become the main source of Web traffic. Although some bots perform well, such as search engines, other bots can perform DDoS attacks, posing a huge threat to websites. The project aims to develop an offline system that can effectively detect malicious web robots, which is not only conducive to network traffic cleaning, but also conducive to improving the network security of IoT systems and services. A comprehensive literature review for the years 2010-2019 was conducted to identify the research gap. The key contributions of the research are: 1) it provided a systematic methodology to address the web robot detection problem based on the log file from industrial company; 2) it provided an approach of feature engineering, thus overcoming the challenge of curse of dimensionality; 3) It made a big progress in the accuracy of off-line web robot detection through a holistic study on the three types of machine learning techniques based on real data from industry.

Three algorithms based on Keras sequential model, random forest, and SVM, were developed with python to detect web robots from human visitors on the TensorFlow 2.0 platform. Experimental results suggested that random forest obtained the best performance in accuracy and training time.

The parameters of each model were adjusted using trial and error approach. For the Keras sequential model, the impact of parameters such as activation function, training epoch and dropout on the detection performance were examined. The linear activation function obtained higher accuracy than sigmoid activation function. Training accuracy increased with the increase of training epoch and tended to stabilize when the training epoch reached 40. Experiments suggested that dropout could prevent overfitting. For random forest, the impact of the number of trees on the detection performance was examined. The training accuracy decreased slightly when the number of estimators exceeded one. For the SVM model, the impact of the kernel function was examined. The linear kernel obtained higher accuracy than the nonlinear kernel.

Feature importance was investigated through random forest and information gain. The calculation results of these two methods are consistent. Then an incremental approach was applied to observe the impact of features in the order of decreased feature importance. The accuracy of all models showed a trend of rising volatility. A feature could have different effects on different models, for example, when feature *os (operating system of user machine)* and *app version (the version of the application which is running on user machine)* was added to training, the accuracy of Keras sequential model increased, while the accuracy of random forest and SVM decreased slightly. When more features were added to training, the accuracy of random forest and SVM increased, and the accuracy of neural network slowly decreased. This indicates that different features could have different effect on different decision processes, although we have sorted the order of attribute importance through random forest and information gain. When top 6 important features were used to the training of neural network, it achieved the highest accuracy of 100%; when the selected 22 features were added to the training process of random forest, it obtained the highest accuracy of 100%. This project uses the confusion matrix as an evaluation method, combined with training time, to measure the performance of a model. Although both neural networks and random forests can achieve 100% accuracy under certain conditions, when the same accuracy is achieved, the training time of random forests is 50% less than that of neural networks. Thus, the conclusion of this project is that random forest is the best model for offline detection of web robots.

Keywords:

Web robot, Web crawler, Sequential model, Random forest, SVM, Feature importance, TensorFlow 2.0.

ACKNOWLEDGEMENTS

Firstly, I would like to thank Dr. Hongmei He and Dr. Andrew Starr. Without their supervision, I may not be able to complete my research on time. They pointed out the direction of my research and sought opportunities for my academic development. When I needed any support, they always offered a helping hand without hesitation.

Secondly, I want to thank Cranfield University, including Registry, Education Support, SASSATM-Research, and other departments who may have helped me even I have not remembered their name. Especially, I would like to thank the manufacturing department who offered financial support and a lot of other help to my research.

Thirdly, I would like to give my great thanks to my parent and my family. I could not have studied at a wonderful university without the love and support of them. At the same time, in this difficult period, it is hard for me to study and live with peace of mind if they had not sent me their care and greetings.

Finally, I want to thank my girlfriend and my amazing friends. They always showed great warmth and support to me. In this difficult time, we always encourage each other and be there when needed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

IT	Information Technology
ML	Machine Learning
NN	Neural Network
SVM	Support Vector Machine
RF	Random Forest
WBA	Weak Bayesian Approach
SBA	Strong Bayesian Approach
BDT	Boosted Decision Tree
NNGE	Non-Nested Generalized Exemplar
CSS	Cascading Style Sheets
CV	Computer Vision

1 INTRODUCTION

1.1 Web Robot

Web robots, also known as spiders, crawlers, walkers, wanderers and harvesters, are computer programs that travel across the Internet and collect data autonomously (Schmidt, 2015; Udapure, Kale and Dharmik, 2014).

The most popular way for people to grasp information from the web is by searching with a search engine (Udapure, Kale and Dharmik, 2014). Search engines like Google use web robot to index web pages to be used in their page ranking process (Algiriyage, 2017). Except for search engines, web robots can be used to collect data for analysis, monitoring public opinions or even booking tickets.

Recent academic report suggests web robots accounted for 60% http requests on the internet (Zabihimayvan et al., 2017). According to the 2019 Bad Bot Report shown in Figure 1-1, which was published by Distil Networks. Web robots accounted for 37.9 percent of all online traffic during 2018 among which 20.4 percent are bad (Distil Networks, 2019).

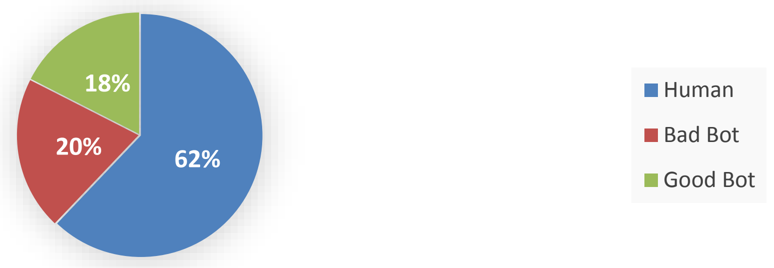


Figure 1-1 Bad bot vs. good bots vs. human traffic 2018 *(Distil Networks, 2019)*

The illegitimate robots can perform some activities that violate the robot.txt file, collect sensitive information and consume large amounts of bandwidth by staging Distributed Deny of Service (DDoS) attacks. This could cause serious loss to companies that rely on web traffic (Catalin and Cristian, 2017). The behaviour of web robots might be different on different websites. Menshchikov et al. (2017) analysed the behaviour of web robots on different websites.

According to a report from robotattaack.org, between 2012 and 2018, more than 20 companies received the effects of robot attacks. These include well-known companies such as Cisco and IBM. Specifically, the Cisco products affected include WebEx Business Suite (including Meeting Center, Training Center, Event Center, Support Center), and services including Cisco ACE 4710 Application Control Engine Appliance. These attacks were promptly repaired without causing serious damage. However, this still suggests that any traffic-centric site is likely to be threatened by a robot attack. For IBM, more than 16 software was attacked, and the main action of robots' attack is exposing sensitive information to unauthorized actors.

With the development of Internet of Things technology, network traffic security has become an even more challenging issue, it is also a critical challenge for IoT enabled systems and services. Effectively detecting malicious web robots will benefit not only for the security of network traffic but also for the IoT enabled systems and services.

1.2 Machine-Learning and Web Robot Detection

ML is the algorithms and statistical models that computer systems use to complete certain tasks without explicit instructions. Instead, ML algorithms rely on patterns and inference. It is a subset of AI (Artificial Intelligence). ML algorithms can build a mathematical model based on given data, or so called “training data”, then the models can make predictions when similar data is fed into them (Koza et al., 1996; Bishop, 2006). ML are widely used in various scenarios, such as spam detection and auto pilot, in which it is extremely difficult to develop a conventional algorithm for the task.

ML algorithms could be categorized into several broad themes, such as unsupervised learning, supervised learning, reinforcement learning and so on. The supervised learning algorithms will build a mathematical model from a dataset which contains both features and labels (Russell et al., 2010). While in unsupervised learning, the algorithm builds a mathematical model from a dataset that contains only features.

Doran and Gokhale (2011) classified web robot detection techniques to four themes: (1) Syntactical log analysis; 2) Traffic pattern analysis; (3) Analytical learning techniques; (4) Turing test systems. In the recent decade, three major themes, such as offline web robot detection based on web log analysis with machine-learning techniques (Menshchikov et al., 2017; Doran et al., 2011; Ma et al., 2017; Rovetta et al., 2020; Iliou et al., 2019; Doran et al., 2016), honeypots (McKenna, 2016; Priyanka et al., 2016; Abdullahi et al., 2019; Selvaraj et al., 2015), and online web robot detection (Guo et al., 2019; Balla et al., 2011), were popularly investigated in the literature.

1.3 Python and TensorFlow

Python is an interpreted, high-level, general-purpose programming language. Python’s design philosophy emphasizes code readability with its notable use of significant whitespace. It supports multiple programming paradigms and it is often described as a “batteries included” language due to its comprehensive standard library (Kuhlman et al., 2004; Python Software Foundation, 2012). Python has a wealth of libraries for data science and machine learning. For example, numpy, which is an efficient library to deal with numeric data; pandas, which is suitable for string process; sklearn, which contains various classical machine-learning algorithms (e.g. SVM, Decision Tree), Boosting algorithms (e.g. Adaboost), and Bagging algorithms (e.g. random forest).

TensorFlow is an open source software library that uses data flow graphs for numerical calculations. The nodes in the graph represent mathematical operations, and the edges in the graph represent the multidimensional arrays (tensors) passed between these nodes. With this flexible architecture, you can deploy computing work to one or more CPUs or GPUs in desktop devices, servers, or mobile devices through an API. TensorFlow was originally developed by researchers and engineers in the Google Brain team (belonging to Google's machine intelligence research department) and was designed to be used for machine learning and deep neural network research. However, the system has good versatility and can also be applied to many other fields (2018).

1.4 Research Motivation, Aim, and Objectives

1.4.1 Motivation and aim

The motivation for this project can be summed up as follows: Today's Internet is growing rapidly, with more and more companies with traffic at its core, yet malicious bots can attack such corporate websites, affecting their business and even suffering serious financial losses. At the same time, websites attacked by malicious reptiles often block normal human access, which seriously affects their browsing activities. Besides, they waste bandwidth resources, mislead people, and they can trick search engine to gain unfair search results (T. H. Sardar and Z. Ansari, 2014). If such a network robot is detected and blocked in a timely manner, the normal user's online experience will be greatly improved. So, this project was started. The project aims to develop an offline system that can effectively detect malicious web robots, which is not only conducive to network traffic cleaning, but also conducive to improving the network security of IoT systems and services. The key contributions of the research are: 1) it provided a systematic methodology to address the web robot detection problem based on the log file from industrial company; 2) it provided a methodology for feature selection, overcoming the challenge of curse of dimensionality; 3) it investigated three types of machine learning techniques based real data from industry, making a big progress in the accuracy of off-line web robot detection.

1.4.2 Objectives

To achieve the goal, this research will implement the following objectives:

1) To identify research gaps through literature review.
2) To find out important features in web log files determine the identity of web robots.
3) To develop cutting-edge machine learning techniques, such as neural network, random forest, and SVM algorithms for web robot detection.
4) To assess and compare the developed machine learning models for web robot detection.

1.5 Thesis Structure

The rest of the thesis is structured as follows:

Chapter 2: Research methodology

This chapter illustrates what methods were employed in each phase of the research.

Chapter 3: Literature review

This chapter gives a comprehensive review of existing works that related to web robot detection and identifies research gap.

Chapter 4: Feature engineering

This chapter shows how the dataset was pre-processed and how the feature importance was calculated.

Chapter 5: Investigated model for web robot detection

This chapter explains the mathematical principles of the investigated models and their implementation.

Chapter 6: Experiments and evaluation

This chapter demonstrates the experimental process using the investigated model, after which the experimental results are evaluated and discussed.

Chapter 7: Conclusion and future work

This chapter describes the key findings and contributions to knowledge. There are also ideas for future work.

2 RESEARCH METHODOLOGY

This section introduces the methods and tools used in this research program. Table 2-1 illustrates the research methodology.

Table 2-1 Overview of the research methodology

Literature Review (Hanlin et al., 2020) Chapter 3	• Reviewing the literatures related to web robot detection; machine-learning • Taking a close look to the existing web robot detection strategies. • Identifying research gap
Feature Engineering Chapter 4	• Data cleaning: Inspecting the industrial data; deleting the obviously irrelevant fields, Dealing with missing values • Data editing: Normalize data that cannot be recognized by models. • Feature selecting: Two methods were used to calculate the influence of each features on the classification results. • Splitting the dataset into two subsets: training data and test data
Investigated Model for Web Robot Detection Chapter 6	• Inspecting 5 types of models/ algorithms that will be used in this project, including: decision tree, SVM, Boosting, Bagging and TensorFlow Keras sequential model. • Part of training data was used to train the mentioned models. • Selecting models that has the best and the worst results for further experiments
Experiments and Evaluation Chapter 7	• Experiment are conducted using all features and using selected features • Training data was fed to each model that were trained with different parameters. • Test each model with test dataset and evaluate the test results.
Conclusion and Future Work Chapter 8	• Random forest is the most accurate and the fastest algorithm for distinguishing the robots from humans by web log-based-learning. • Further research could be undertaken to carry out this study to focus on online detection, web robot detection devices, web robot datasets, and cloud-based web robot detection.

This research project includes five stages. They are:

Stage 1: *Literature Review*

To identify the research gap review in web robot detection, a comprehensive literature was conducted. The review covers main works in the last decade.

Database used include Google Scholar, Springer, IEEE Xplore Library, Wikipedia and so on.

The documents that have been viewed are sorted according to their views. The views of these documents were refined and summarized, and an attempt was made to find their shortcomings. Thus, identified the research gap. As the final step, summary of the literature is to identify the research gap of web robot detection.

Stage 2: *Feature engineering*

The dataset is a comma separated value file which has columns and row when opened by Microsoft Excel. Each column is a feature in machine learning progress. At this stage, Python machine-learning libraries (numpy, pandas, TensorFlow, and sklearn) were imported and used to pre-processing the data.

The first step of data pre-processing includes fill in the missing values, deleting the duplicated values, normalize numerical values and using hash to transform string values into vectors.

In the second step, random forest and Information Gain were used to evaluate the importance of each feature. Feature importance is a key factor for some algorithms.

Stage 3: *Models and implementation*

This stage focuses on the mathematical theory of investigated models and how they were implemented in this project.

Stage 4: *Training, Test and Evaluation*

At this stage, the dataset will be divided into three parts: training set, validation set and test set. Then some parameters of investigated models will be adjusted for the training process. After training, decision region, confusion matrix, and time-efficiency test were implied to evaluate models' performance.

Stage 5: *Conclusion and Future Work*

At this stage, the key findings and conclusions were listed according to the research objectives. random forest algorithm obtained the highest accuracy and the fastest training speed.

3 LITERATURE REVIEW

This chapter introduces a literature review covering the past 10 years. These documents are all related to web robot detection. This part contains three topics, web log analysis, honeypot technology and online web robot detection. For each topic, this review compares the performance of the systems proposed in the work, listing their respective advantages and their shortcomings. Finally, summarize their performance, and on this basis, put forward the main challenges and research gaps.

3.1 Web Log Analysis

This section focuses on the use of web log analysis methods to detect web crawlers. In this section, a total of five types of web robot detection methods are introduced, all of which are based on offline web log analysis.

Rajabnia and Jahan (2016) proposed a hybrid fuzzy inference system based on NNGE (non-nested generalized exemplar) algorithm. In the proposed NNGE algorithm, only the main features are used to train the model. In addition, a hybrid inference system was developed in this work to infer the possibility that the web log came from a robot.

Bayesian network is a popular method in web robot detection. Suchacka and Sobków (2015) used a Bayesian approach to robot detection based on pattern of user sessions. In their work the performance of Weak Bayesian Approach (WBA) and Strong Bayesian Approach (SBA) were compared. Under normal circumstances, the accuracy of SBA is better than that of WBA. However, how to choose these two methods depends on the tolerance for errors. If a small amount of errors is allowed in the usage scenario, one can use SBA, otherwise one can only use WBA.

Sisodia et al. (2015) believed the result of web server log analysers are not very reliable due to the highly inflated input log files. They proposed an agglomerative approach combining web logs with actual visitors' knowledge extraction, and evaluated the performance of these ensemble learners with recall, precision, and

F1 measure. The precision for the web robot sessions with ensemble classifiers is more than 80% in the first experiment and 98% in the second experiment.

Haidar (2017) developed a two-class Boosted Decision Tree (BDT) for web robot detection based on website navigation behavior analysis. In addition to DBT, his works also involve SVM, neural network and random forest. These were used in controlled trials. Two websites (named wheelers and wherever) provide the data needed for this experiment. The biggest advantage of this system is that it can be retrained according to the evolution of web robot types. The experiment results are shown in Table 3-1. The fields in Table 3-1 are defined as follows:

- TP: true positive, is the percentage of positive cases correctly classified as belonging to the positive class.
- FP: false positive, is the percentage of negative cases misclassified as belonging to the positive class.
- Precision: (also called positive predictive value) is the fraction of relevant instances among the retrieved instances: Precision = TP / (TP + FP)
- Recall: (also known as sensitivity) is the fraction of the total amount of relevant instances that were retrieved: Recall = TP / (TP + FN)
- F1-score is a measure of a test's accuracy. It considers both the precision and the recall of the test to compute the score, and it is the harmonic mean of the precision and recall: F1 = 2 × Precision × Recall / (Precision + Recall)

Table 3-1 Comparison between different web robot classification methods

Classification Methods	Source of Data	No. of Selected Features	Precision	Recall	F1-score	Accuracy
NNGE-fuzzy Inference system	Pars Web Server	4	0.9931	0.9931	0.9931	0.993
SBA	Real E-commerce Site	20	N/A	N/A	N/A	0.931
Agglomerative Approach	Unclear	23	0.9800	0.9800	0.9800	N/A
Random Forest	Wheelers / Whereleb	Unknown	0.828	0.739	0.756	0.797
BDT	Wheelers	496	0.920	0.765	0.815	0.831
	Whereleb	472	0.916	0.502	0.601	0.731

Lagopoulos et al. (2018) proposed a semantic approach for web robot detection. They designed this system based on a basic assumption, that is, web robots will randomly grab the content they encounter, but humans will follow a topic to access related web content.

Typical features extracted from sessions includes:

- Total Requests
- Session Duration
- Average Time: average time between two consecutive requests.
- Standard Deviation Time: the standard deviation of the time between two consecutive requests.
- Repeated Requests: a request for an already visited page using the same HTTP method as the previous one.
- HTTP requests: four features, each containing the percentage of requests associated with one of the following HTTP response codes: Successful (2xx), Redirection (3xx), Client Errors (4xx) and Server Errors (5xx).

- Specific Type Requests: The percentage of requests of a type over the number of all requests. This feature is application dependent.

While the semantic features extracted from a session are:

- Total Topics (TT): The number of topics with non-zero probability.
- Unique Topics (UT). The number of unique topics with non-zero probability.
- Page Similarity (PS). The ratio of unique topics with non-zero probability over all the topics with non-zero probability.
- Page Variance (PV). The semantic variance of the pages of a session.
- Boolean Page Variance: It is a Boolean version of PV.

The experiments were carried out with four different models: an SVM with an RBF kernel (RBF), a gradient boosting (GB) model, a multi-layer perceptron (MLP), and an eXtreme Gradient Boosting (XGB) model. From Table 3-2, it can be seen that RBF achieved the best performance when using semantic features; MLP performed relatively poorly; GB and XGB achieved the best performance when both semantic features and simple features were used at the same time; (Lagopoulos, Tsoumakas and Papadopoulos, 2018).

Table 3-2 Performance of the models using the simple, the semantic and both simple and semantic features *(Lagopoulos et al., 2018)*

Performance Indicator	Feature sets	RBF	MLP	GB	XGB
F1-score	Simple	0.655	0.784	**0.907**	0.905
	Semantic	**0.848**	0.749	0.848	0.846
	Both	0.648	0.816	**<u>0.918</u>**	0.917
Accuracy	Simple	0.651	0.768	**0.900**	0.898
	Semantic	**0.848**	0.771	0.845	0.841
	Both	0.651	0.801	**<u>0.913</u>**	0.912
G-mean	Simple	0.583	0.743	**0.898**	0.896
	Semantic	**0.847**	0.767	0.843	0.839
	Both	0.565	0.781	**<u>0.912</u>**	0.911

3.2 Honeypot

Honeypot is represented as an invisible link or vulnerable web page that intentionally designed by developers to mislead web robots. A human cannot see the links or other resources that have been hidden. However, a web crawler, looking at the source code, does not check the visibility before requesting them. Based on this assumption, McKenna (2016) proposed a strategy to detect and classify web robots with honeypots. He used a CSS rule named *display: none* to construct hidden contents. Besides, they built a sand trap, which implements a server-side PHP script to catch crawlers. This system will also detect whether a certain web robot complies with the constraints of robot.txt. If it complies, the system will classify it as a good robot, otherwise it will mark it as a bad robot. But his experiment did not distinguish between good robots and bad robots as expected In his final conclusion, he indicated that pure honeypot technology is not suitable for direct web robot detection and classification, because good robots may also be misclassified as bad ones. (McKenna, 2016).

In 2015, Gržinić and his colleagues developed a data collection system called Lino. This system will simulate an unsafe web page to attract web robots. For the collected data, select the characteristics. And use the data set to train the decision tree C4.5 and SVM model. Both models were used in two sets of experiments. In Experiment 1, only the selected features were used in the training process. In Experiment 2, in addition to using the selected features, two additional features were added, the country and the customer's ASN. Experiment results are shown in Table 3-3

With Lino they selected top 5 features that dominate the dataset including:

- Post data, which shows us whether the client has filled/not filled the fake form in the Lino system.
- Session change, which shows us if user, during the session, has changed the session identifier or not.
- Session duration, duration of the session in seconds.
- Robots, which shows us whether the user accessed /not accessed to the robots.txt file, which defines the rules of robot conduct.

Table 3-3 Performance of C4.5 and SVM *(Gržinić, Mršić and Šaban, 2015)*

	Class	TP	FP	F1	AUC
C 4.5 Experiment #1	Human Robot	0.177 1	0 0.823	0.301 0.972	0.773 0.773
C 4.5 Experiment #2	Human Robot	0.793 0.998	0.002 0.207	0.872 0.992	0.985 0.985
SVM Experiment #1	Human Robot	0.625 1	0 0.735	0.419 0.979	0.801 0.801
SVM Experiment #2	Human Robot	0.962 0.998	0.006 0.042	0.942 0.997	0.976 0.978

The flaw of this algorithm is that the false positive rate is too high. In other words, many human users may be marked as web robots. (Gržinić, Mršić and Šaban, 2015).

Priyanka et al. (2016) employed similar strategy to detect malicious web robot. However, honeypot was utilized in a different way. The flow chart of the system is shown in Figure 3-1. This system was designed to be deployed on the server side. This system uses a combination of honeypot technology and an intrusion detection system.

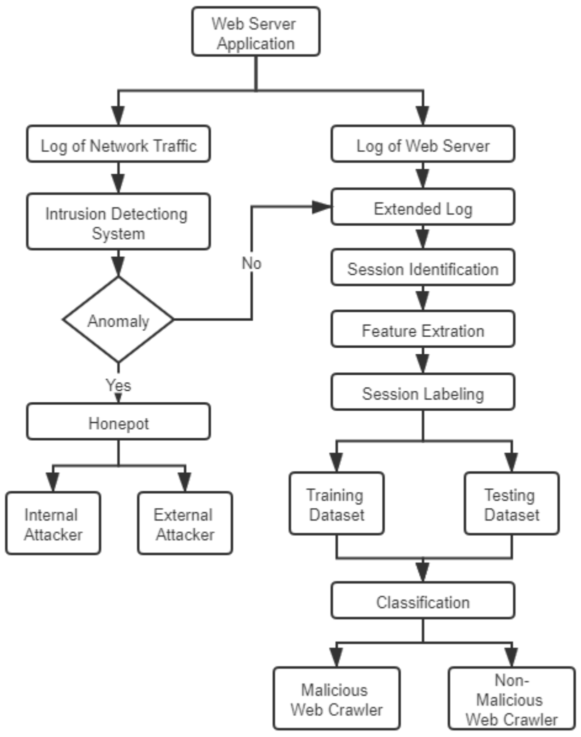


Figure 3-1 Data flow diagram of the crawler detection system *(Priyanka et al., 2016)*

Figure 3-1 shows that the web-side application will do two things at the same time. Transaction A is to send the webpage traffic log to the intrusion detection system for analysis. If the analysis result is abnormal, the producer of a certain log is induced into the honeypot and analyzed in the honeypot whether he is an internal attacker or an external attacker. Transaction B is: when the detection result of the intrusion detection system in transaction A is no abnormal, the information of the web log and the information of the server log are combined to create an extended log, and then the extended log is subjected to session extraction, feature extraction, and session labeling. And divide the log file into a training set and a test set, and then perform classification operations to distinguish malicious crawlers from non-malicious crawlers.

3.3 Online Web Robot Detection

Most of the existing web robot detection systems are offline web log analysis. Few systems can detect web robots online.(Cabri et al., 2018). In 2019, Wan et al. (2019) proposed an online web robot detection system called PathMarker. It can trace the page that leads to the access to an URL by adding a marker to the URL and identify the user, who accesses to this URL. SVM was utilized to distinguish malicious web crawler from normal users. Experimental results showed the proposed system could successfully identify 96.74% crawlers’ long sessions and 96.43% normal users’ long sessions. The architecture of PathMarker is shown in Figure 3-2.

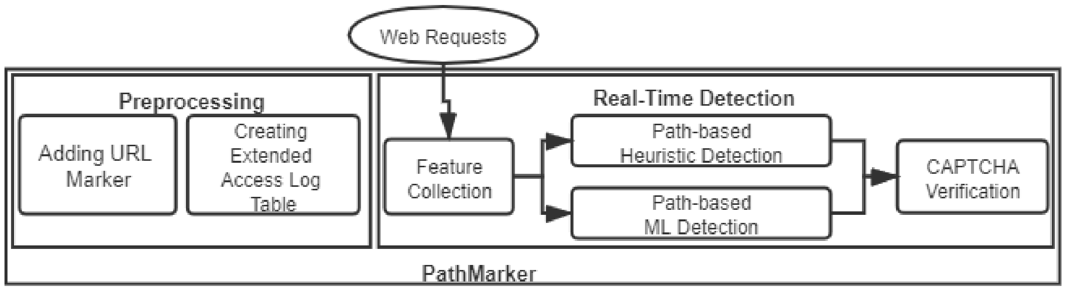


Figure 3-2 PathMarker architecture *(Wan et al., 2019)*

PathMarker's workflow consists of two parts: Preprocessing and Real-time detection. In preprocessing period this system will add a tag to some URLs in the page. when. Then the system will create an extended access log table containing IP address, visited URL, timestamp, and extended information. In real-time detection period, First, extract the features of the web page request and send these features to its unique A and B subsystems, and finally send the output of these two subsystems to CAPTCHA verification to finally determine the identity of the requester.

Experimental results are shown in the Table 3-4. Type 0 stands for human users. Type 1, 2, and 3 stands for breadth-first crawlers, depth-first crawlers, and random-like crawlers. The system could successfully identify 96.74% crawlers' long sessions and 96.43% normal users' long sessions.

Table 3-4 Confusion matrix of PathMarker *(Wan et al., 2019)*

Original type	**Classify as 0**	**Classify as 1**	**Classify as 2**	**Classify as 3**
0	96.43%	0%	3.57%	0%
1	0%	100%	0%	0%
2	0%	6.25%	93.75%	0%
3	1.51%	1.77%	0%	96.72%

Except for web robot identification, Wan also conducted experiments to inspect the impact PathMarker has on distributed crawlers. They take the crawling efficiency of a distributed web robot with only one worker as a reference standard when the PathMarker system is not used. Figure 3-3 illustrates the relationship between the crawling efficiency of a single worker and the total number of workers when the PathMarker system is applied.

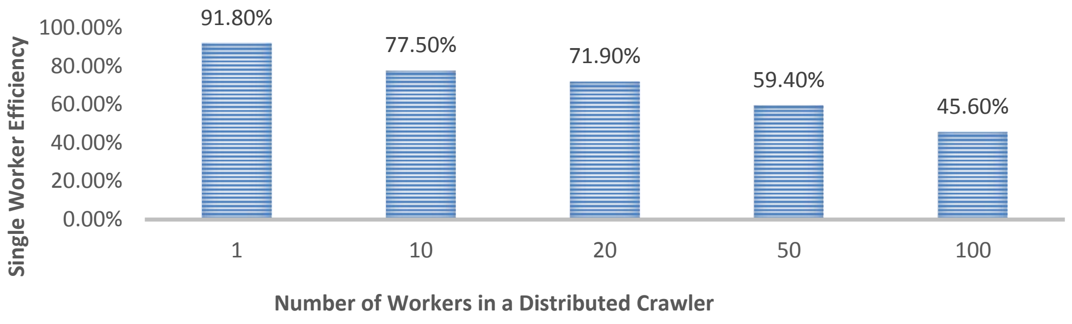


Figure 3-3 PathMarker suppressing distributed crawler *(Wan et al., 2019)*

This figure shows that when the PathMarker system is used, if there is only one worker in the distributed web pages and robots, then its crawling efficiency will not be greatly affected. However, as the number of workers increases, the efficiency of each worker will be increasingly affected. When the number of workers reaches 100, the work efficiency of each worker is less than half of their expected efficiency.

In the same year, Cabri et al. (2018) proposed a novel approach for binary classification of a multivariate data stream incoming on a web server. Deep neural networks and Wald's sequential Probability Ratio Test were utilized to represent the relation between subsequent HTTP requests. Results showed that the proposed approach could detect robots with a high accuracy and had slight impact on human visitors. Figure 3-4 shows that the developed system could achieve stable Precision (0.979), Accuracy (0.963), F1 (0.959) and Recall (0.940) after the number of requests reached 8.

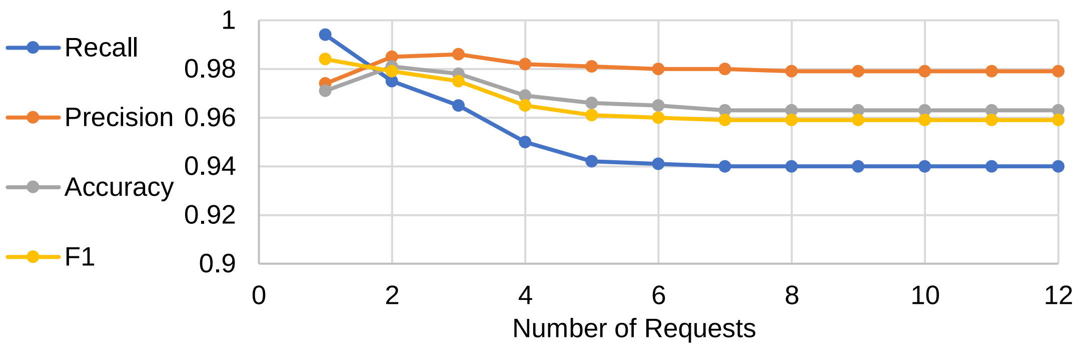


Figure 3-4 Evaluation scores of the sequential classification approach *(Cabri et al., 2018)*

Figure 3-4 shows that as the number of requests increases, the recall rate of the system is low, and the accuracy is high. Using the formula of Recall and Precision (mentioned in 3.1) to analyze the experimental results, it shows that the FN rate of this system is greater than the FP rate, which in turn indicates that this system is more inclined to classify web robots as normal users.

3.4 The Identified Research Gap and Challenges

A review of the relevant literature shows that the existing methods have their advantages and disadvantages. This section summarizes the advantages and disadvantages mentioned above. This leads to the challenges and research gaps in the field of web robot detection, which is listed in Table 3-5.

Table 3-5 Challenges in web robot detection

Detection methods	Challenges
Web log analysis	• Curse of dimensionality (D. Stevanovic et al., 2012). • Positive case labelling is a critical challenge (D. Doran and S. S. Gokhale, 2011). • Web log analysis systems cannot detect and block web robots in real-time (A. Mason et al., 2019). • The detection accuracy is not high enough.
Honeypot	• It could drop the performance of a web browser (Gržinić et al., 2015). • Normal users could be fooled too (Gržinić et al., 2015). • Honeypot defence techniques can be thwarted if web robots are employing certain countermeasures (McKenna, 2016).
Online web robot detection	• Uncertainty of web robot situations (Cabri et al.,2018). • Poor session labelling (A. Mason et al., 2019). • Unknown web robots. • The real-time performance of online web robot detection system. • The diversity of web robots.

According to the results of the literature review, the current research status of web robots can be summarized as follows: First, most of the existing systems detect web robots based on offline web log analysis. Although these methods have achieved considerable accuracy, they cannot detect web crawlers in real time. Second, honeypot technology is not recommended for direct detection of web robots. Instead, it can be used as a good data collector. Third, the accuracy of some online detection systems is generally lower than that of offline detection systems, and most online monitoring systems are only effective for certain types of crawlers.

Through this review, we can draw conclusions: First, some specific algorithms or models are only suitable for specific samples, so the choice of model will have a great impact on the detection results. Second, the choice of features will also be crucial to the test results. Third, to get a good understanding of the performance of an algorithm or system, a variety of evaluation methods should be adopted.

When researching related topics, you should first consider the browsing experience of web users. In other words, the system used to detect web crawlers online should not have a significant impact on the loading speed of web pages.

In the latest research, there is a method for online detection of web robots. It is to study the relationship between each request of a certain user in a session, and then determine the identity of the user. For future study, it is demanded to design and develop high performance and highly efficient online web techniques or methods for detecting web robots and/or distributed web crawlers. This is required by "Security by Design" for Industry 4.0.

4 FEATURE ENGINEERING

The dataset for the project comes from an Internet company whose business area is local living services. The original datasets are two comma-separated-value (csv) files, one of which contains 1000 human traffic records, and the other contains 1000 web crawler traffic records. The dataset has 43 columns among which 42 are features and 1 column is label. The outlook and the field definition of the dataset is shown in the Table 4-1 below. Note that the definition of some features may be somewhat vague, as this involves the interests of the company.

Table 4-1 Field definition of the dataset

Field name	Definition
_mt_datetime	log generation time
_mt_servername	log generation server
_mt_appkey	log generated service appkey
_mt_leve	log level
_mt_thread	the thread which generated the log
_mt_action	logger name generated by the log
_mt_message	log content
request_id	request id
request_time	request time
union_id	user device tag
user_id	user id
Os	operating system of user machine
app_version	application version number
Mac	no data due to permission problems
page _city_id	page city id
locate_city_id	target city id
Lat	Latitude
Lng	Longitude
page_index	page order, counting from 0
Offset	page offset
page_size	length of each page
Stock	number of ads requested
page_id	the page ID of the requested ad slot
lx_page_id	the actual page ID of the requested ad
Channel	ad request channel
channel_source	distinguish between search requests on the homepage or channel page
area_id	area id
cate_id	front desk category id
Keyword	search keyword
query_type	query type
slot_id	ad slot id
intent_type	intent type of request
Extensions	extended field
cate_ids	request the front desk category ids, multiple id stitching
query_analysis	keyword understanding result JSON
search_request	original request
search_info	search Information
ab_trace_tag	experiment id
slot_ids	request ad slot ids
dt	date
hour	hour
ctime	time

In this project, a Python library named *pandas* was used to pre-process the data for further experiment.

4.1 Data Cleaning

The original dataset has 43 columns among which 42 are features and 1 column is label. However, many columns have the same meaning, or the other case is many columns have too much null values, which may bring too much extra calculation or causing overfitting when training the model. If users think the data is messy, they are less likely to believe that the mining results based on this data, that is, the output results are unreliable. The main idea of data cleaning is to "clean up" the data by reducing missing values, smoothing noisy data, deleting outliers, and resolving data inconsistencies.

4.1.1 Handling the missing values

The methods for dealing with these missing values are mainly based on the distribution characteristics of the variables and the importance of the variables, the amount of information and the ability to predict. The following operations are performed in this project to handle missing values:

- Deleting a column: if a column has a high rate of null value (greater than 80%), and it has a minor importance, the column will be deleted.
- Statistics filling: If the missing rate is lower (less than 95%) and the importance is low, the filling is performed according to the data distribution. For data that conforms to a uniform distribution, use the mean of the variable to fill in the missing. For data that have a skewed distribution, use the median to fill.

After deleting the columns that are not needed. The dataset has twenty-one columns left plus one label column.

4.1.2 Data transformation

Data transformation includes normalizing, discretizing, and thinning the data to make them suitable for machine learning. Before transforming the data, a built-in function in pandas called *dtype* was applied to inspect each column in the dataset file to find out what type they are. The number of columns with the following data types is shown in the Table 4-2.

Table 4-2 Number of columns with each data types

Data type of columns	**Number of columns**
String	10
Categorical column	2
Numeric column	9

There are three types of features in the given dataset. However, machine learning models can only recognize numbers and matrices. All these types of features were transformed into the format of number or matrices.

This section illustrates the methods applied to transform each column to the format which can be fed into machine learning models. For random forest and SVM all the values of each feature in the dataset were digitized by a built-in function called *factorize* () from a python library called "pandas". After that, all the same values in each column of the table will be replaced by the same number. This number will increase one by one starting from zero and is only used to indicate how many different states there are in this column. However, for the TensorFlow models, they can be digitalized by a TensorFlow built-in function called *categorical_column_with_hash_bucket ()*.

At this stage, the final step is to split the dataset into training set, validation set and test set using a function called *train_test_split ()* imported from *sklearn* library.

4.2 Feature Selection

Feature selection could have a huge impact on the performance of models. In this project, feature selection will be used as a variable to observe its influence on each model. In this section, two method were employed to evaluate the importance of each feature: information gain and random forest.

4.2.1 Information gain

In machine learning, information gain has the same meaning as Kurbach-Leeblier's divergence. The amount of information about one random variable obtained by observing another random variable. In decision trees, information gain is sometimes synonymous with mutual information, which is the univariate probability distribution of one variable and the Kullback-Leibler’s divergence given the conditional distribution of another variable.

The information gain of random variable X obtained from the observation value of random variable A with value A = a is defined as below

$$IG_{X,A}(X,a) = D_{KL}(P_X(x|a)||P_X(x|I)) \tag{4-1}$$

the Kullback–Leibler divergence of the prior distribution $P_X(x|I)$ for x from the posterior distribution $P_{X|A}(x|a)$ for x given a.

In summary, the expected information gain change information entropy H from the previous state requires some information:

$$IG(T,a) = H(T) - H(T|a), \tag{4-2}$$

Where $H(T|a)$ is the conditional entropy of T given the value of attribute a.

Then the information gain of $\boldsymbol{T}$ for attribute $\boldsymbol{a}$ is the difference between the a priori Shannon entropy $\boldsymbol{H}(\boldsymbol{T})$ of the training set and the conditional entropy $\boldsymbol{H}(\boldsymbol{T}|\boldsymbol{a})$.

$$H(T|a) = \sum_{v\in vals(a)} \frac{|S_a(v)|}{|T|} \cdot H(S_a(v)). \tag{4-3}$$

4.2.2 Feature importance evaluation with random forest

The idea of evaluating the importance of features in a random forest is to determine the contribution of each feature to each tree in the random forest, then take the average, and finally compare the contributions between the features.

The first step to calculate the variable importance in a data set $D_n = \{(X_i, Y_i)\}_{i=1}^{n}$ is to fit a random forest to the data. During the fitting process an out-of-bag error

(OBE) is recorded and averaged over the forest (errors on an independent test set can be substituted if bagging is not used during training).

To measure the importance of the j-th feature after training, the values of the j-th feature are permuted among the training data and the OBE is again computed on this perturbed data set. The importance score for the j-th feature is computed by averaging the difference in OBE before and after the permutation over all trees. The score is normalized by the standard deviation of these differences. Features importance is shown in Figure 4-1.

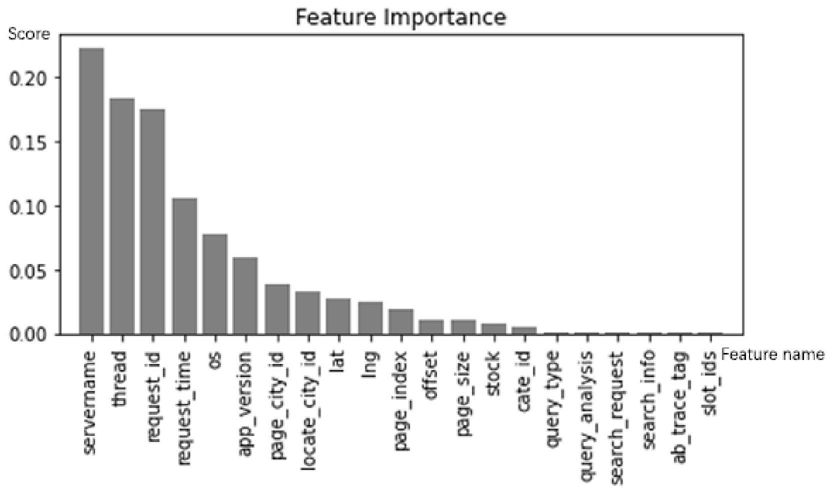


Figure 4-1 Feature importance

Figure 4-1 shows the importance of each feature in the dataset which was pre-processed. The abscissa is the name of each feature, and the ordinate is the importance of each feature. The larger the ordinate value, the more important a certain feature is, and vice versa. This figure shows that the first six most important features are server name, thread, request id, request time, os, and app version. Subsequent experiments will study the impact of these first six important features on the accuracy of the model.

4.3 Getting Features Ready for Model Training

For TensorFlow Keras sequential model, all features of type string should be transformed into their hash matrix using a TensorFlow built-in function named

feature_column.categorical_column_with_hash_bucket()

And all the categorical columns should transform to matrices with another TensorFlow built-in function called:

feature_column.categorical_column_with_vocabularty_list ()

And for the numeric columns, they should be fed in to another TensorFlow built-in function called:

feature_column.numeric_column ()

After defining each feature, they were ready to be added into Keras feature layers.

5 INVESTIGATED MODELS FOR WEB ROBOT DETECTION

The goal of this project is to develop machine learning models that can accurately identify web robots from the given log dataset. In this project, TensorFlow Keras model, random forest, and SVM were investigated. The reason for choosing TensorFlow Keras Model is that we have not found any existing research using it for web robot detection. There are two reasons for choosing random forest. First, few works mainly study random forest; second, in the existing works, random forest did not perform well. For example, in Haidar and Elbassuoni 's work that aimed to develop a classifier which can identify various classes of web robots. They used random forest as comparison. And the model only achieved an accuracy of 80% and a recall of 73.9%. The reason for choosing SVM is that it is a very classic machine learning model, and it is often used to study binary classification problems.

5.1 Keras Sequential Model

Keras sequential model is a linear stack of layers. It is a multi-layer feedforward neural network. The structure diagram of this model is shown in Figure 5-1

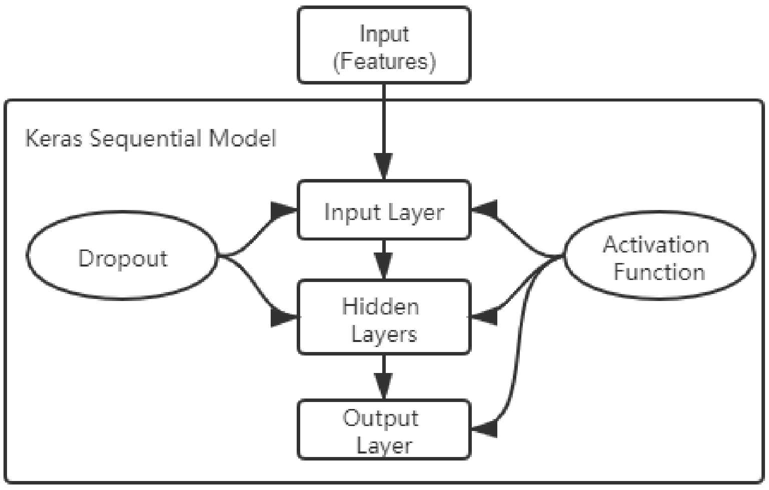


Figure 5-1 Structure diagram of Keras sequential model

The Keras sequential model shown in the diagram has three layers. Each layer could be a dense neural network that contains multiple neurons. The number of neurons can range from 1 to a very big number. But it will have a huge impact on performance of the model. Too many neurons may cause overfitting, but too few may cause underfitting. Within each layer, every neural has its weighs and bias, model will adjust those weighs and bias in the training process as the training epoch moving forward.

In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs. In Keras sequential model, each layer could have its own activation function. There are at least five types of activation function in machine learning. They are identity function, binary step function, bipolar step function, sigmoidal function, and ramp function. Activation function has significantly impact on the model's performance too.

Dropout is a regularization technique patented by Google for reducing overfitting in neural networks by preventing complex co-adaptations on train data.

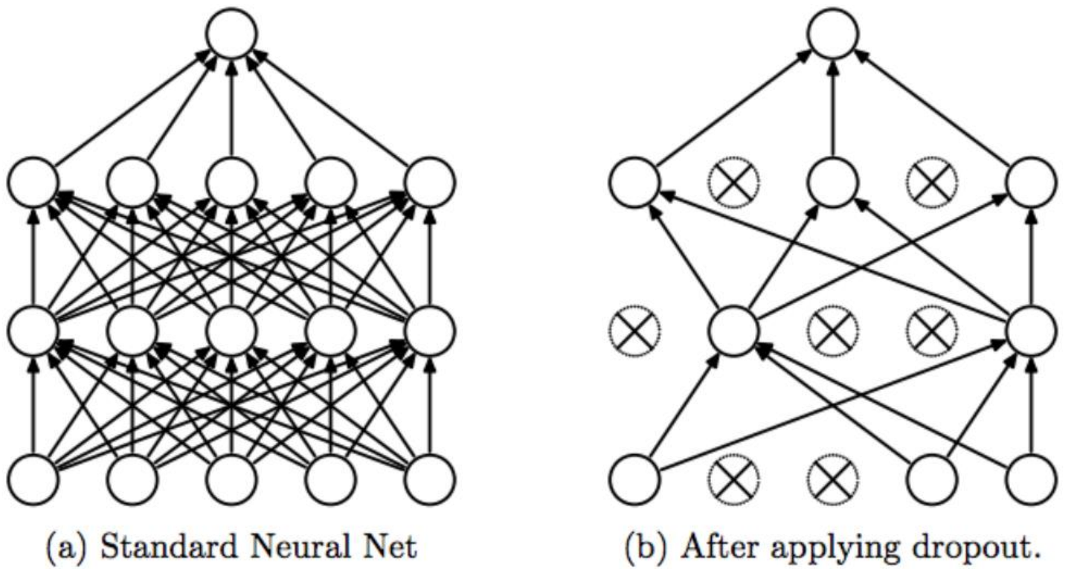


Figure 5-2 Schematic diagram of dropout *(Srivastava, et al., 2014)*

In the training process of the neural network, for a part of the layered neural network trained at a time, some of the neurons in the random selection are hidden first, and then this training and optimization is performed. In the next iteration, continue to hide some neurons randomly, and so on until the end of training.

5.1.1 Implementation

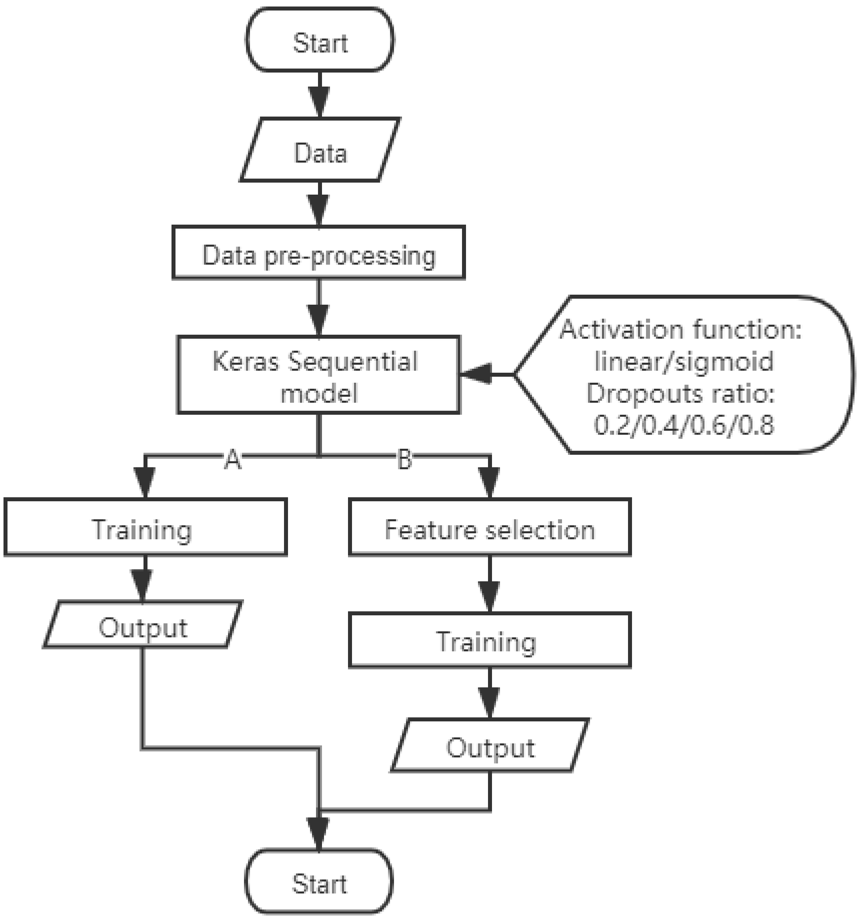


Figure 5-3 Flowchart of the algorithm using Keras sequential model

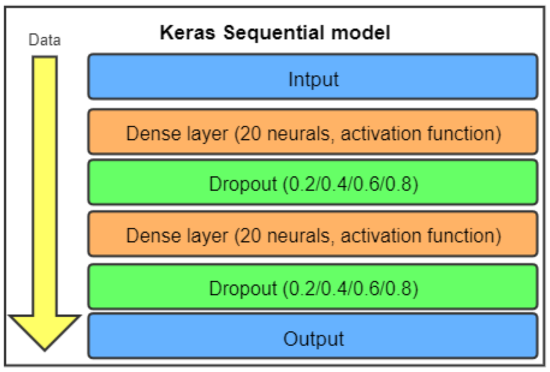

Data
Keras Sequential model
Intput
Dense layer (20 neurals, activation function)
Dropout (0.2/0.4/0.6/0.8)
Dense layer (20 neurals, activation function)
Dropout (0.2/0.4/0.6/0.8)
Output

Figure 5-4 The internal structure of Keras sequential model

Figure5-3 demonstrates the algorithm using the Keras sequential model. The model has two hidden layers, each with 20 neurons. After each hidden layer, a dropout layer that can control the active neurons is added. Stage 1: Data pre-processing, including data cleaning and data normalization; Stage 2, model parameter tuning, including the use of different kernel functions and different proportions of dropout. The next stage is two parallel sub-experiments. Experiment A did not use feature selection, training and test was conducted right after parameter adjustment, followed by training and test results, while Experiment B conducted model training and output training and test results after feature selection. Figure 5-4 is the internal structure of Keras sequential model. Data goes into the model through input layer and then two dense neural layers where activation function and dropouts will be applied. Then the result comes out from output layer. In the following experiments, parameters activation function, dropout will be examined.

5.2 Random Forest

In machine learning, a random forest is a classifier containing multiple decision trees, and the output category is determined by the mode of the category output by the individual trees. Leo Breiman and Adele Cutler developed an algorithm to infer random forest. And "random forest" is their trademark. This term is derived from random decision forests proposed by Tin Kam Ho of Bell Labs in 1995. This method combines Breimans' "Bootstrap aggregating" idea and Ho's "random subspace method" to build a set of decision trees. (Ho,1995).

Random forests correct for decision trees' habit of overfitting to their training set (Hastie, 2008). To tree learners. Given a training set $X = x_1,\ldots,x_n$ with responses $Y = y_1,\ldots,y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $b = 1,\ldots, B$:

1. Sample, with replacement, n training examples from X, Y, call these X_b, Y_b.

2. Train a classification or regression tree $f_b$on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x') \tag{5-1}$$

or by taking the majority vote in the case of classification trees.

Additionally, an estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}} \tag{5-2}$$

The above procedure describes the original tree bagging algorithm. There is only one difference between random forest and this generic scheme: they use an improved tree learning algorithm to select a random subset of features at each candidate segmentation point during the learning process. (Ho et al., 2002).

5.2.1 Implementation

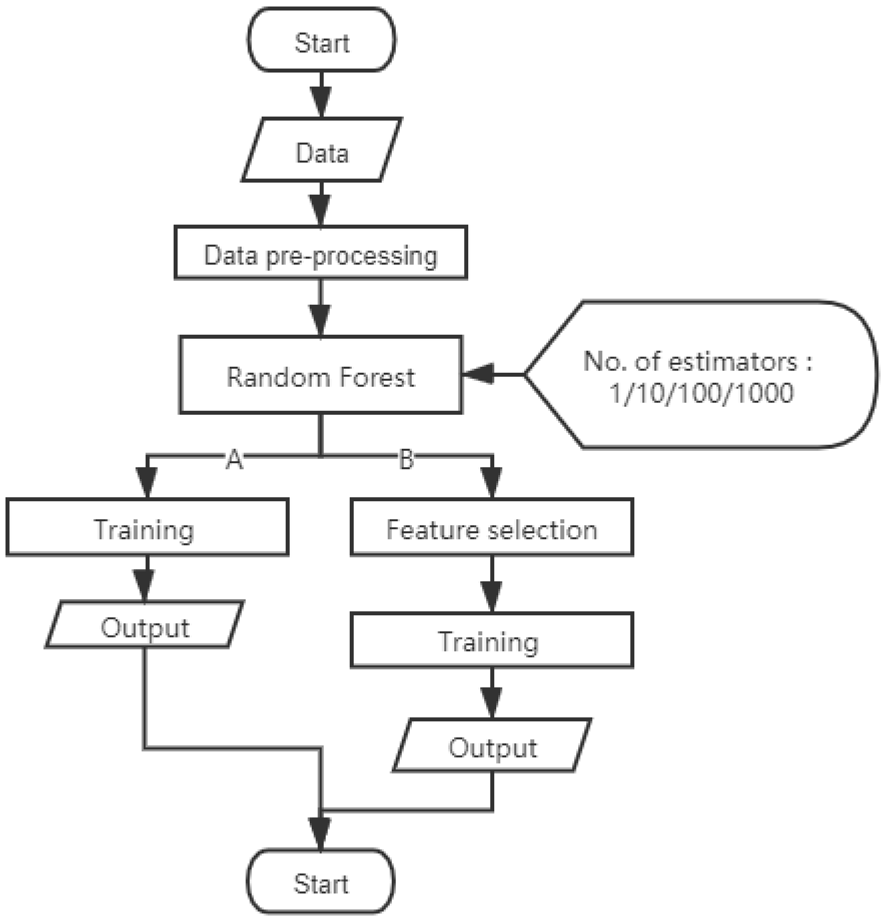


Figure 5-5 Flowchart of the algorithm using random forest

Figure 5-5 demonstrates the algorithm flow using random forest. Stage 1: Data pre-processing, including data cleaning and data normalization; Stage 2, model parameter tuning, including the number of estimators (trees). The next stage is two parallel sub-experiments. Experiment A did not use feature selection, but directly conducted model training and output training and test results, while Experiment B conducted model training and output training and test results after feature selection.

5.3 Support Vector Machine

A Support-vector machine (SVM) is a supervised learning model with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belong to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making is a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

5.3.1 Linear SVM

As shown in Figure 5-6, given a training dataset of n points of the form $(\overrightarrow{x_1}, y_1), \ldots, (\overrightarrow{x_n}, y_n)$, where the y_i are either 1 or -1, each indicating the class to which the point $\overrightarrow{x_i}$ belongs. Each $\overrightarrow{x_i}$ is a $p-$ dimensional real vector. The objective is to find the 'maximum-margin hyperplane" that divides the group of points $\overrightarrow{x_i}$ for which $y_i = -1$, which is the defined so that the distance between the hyperplane and the nearest point $\overrightarrow{x_i}$ from either group is maximized. Any hyperplane can be written as the set of points $\overrightarrow{x_i}$ satisfying $\vec{\omega} \cdot \vec{x} - b = 0$, where $\vec{\omega}$ is the normal vector to the hyperplane. Samples on the margin are called the support vectors.

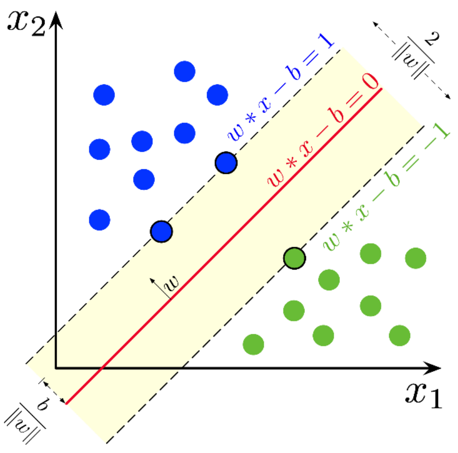


Figure 5-6 Maximum-margin hyperplane and margins for an SVM

If the training data is linearly separable, two parallel hyperplanes can be selected to separate the two types of data, making the distance between them as large as possible. The area defined by these two hyperplanes is called the "boundary", and the maximum boundary hyperplane is the hyperplane between the two. For normalized and standardized data sets, $\vec{\omega} \cdot \vec{x} - b = 1$ described by these hyperplane equations (any of the above boundaries is a class, and label 1) and $\vec{\omega} \cdot \vec{x} - b = -1$ (or below this boundary) Any other class of, with label −1).

The distance between the two hyperplanes is $\frac{2}{\|\vec{\omega}\|}$ so to maximize the distance between the planes, $\|\vec{\omega}\|$ needs to be minimized. The distance is calculated by the distance from a point to the plane equation. To prevent data points from falling into the margin, following constraint is added: for each i either $\vec{\omega} \cdot \vec{x_i} - b \geq 1$, if $y_i = 1$, or $\vec{\omega} \cdot \vec{x_i} - b \leq -1$ if $y_i = -1$. These constraints state that each data point must lie on the correct side of the margin. This can be rewritten as

$$y_i(\vec{\omega} \cdot \vec{x_i} - b) \geq 1, \text{ for all } 1 \leq i \leq n. \qquad \textbf{(5-3)}$$

Now the optimization problem becomes:

“Minimize $\|\vec{\omega}\|$ subject to $y_i(\vec{\omega} \cdot \vec{x_i} - b) \geq 1$ for $i = 1, \dots, n$.”

The$\|\vec{\omega}\|$ and b that solve this problem determine the classifier, $\vec{x} \rightarrow sgn(\vec{\omega} \cdot \vec{x} - b)$.

An important result of this geometric description is that the maximum margin hyperplane is determined entirely by the $\vec{x_i}$ data that is closest to it. These $\vec{x_i}$ are called support vectors.

5.3.2 Non-linear SVM

In 1992, Boser et.al proposed a method to create a nonlinear classifier by applying the nuclear technique to the maximum margin hyperplane (1992). The proposed algorithm is similar in form, except that each dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum margin hyperplane into the transformed feature space. See Figure 5-7, The transformation can be nonlinear, the space after transformation can be high dimensional. Although the classifier is a hyperplane in the transformed eigenspace, it may be nonlinear in the original input space.

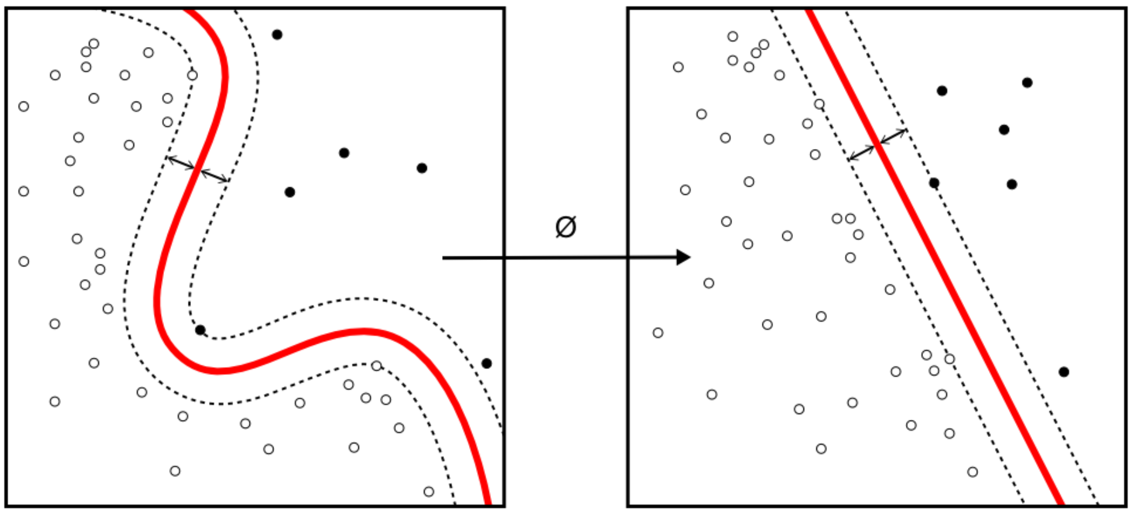


Figure 5-7 Kernel machine

5.3.3 Implementation

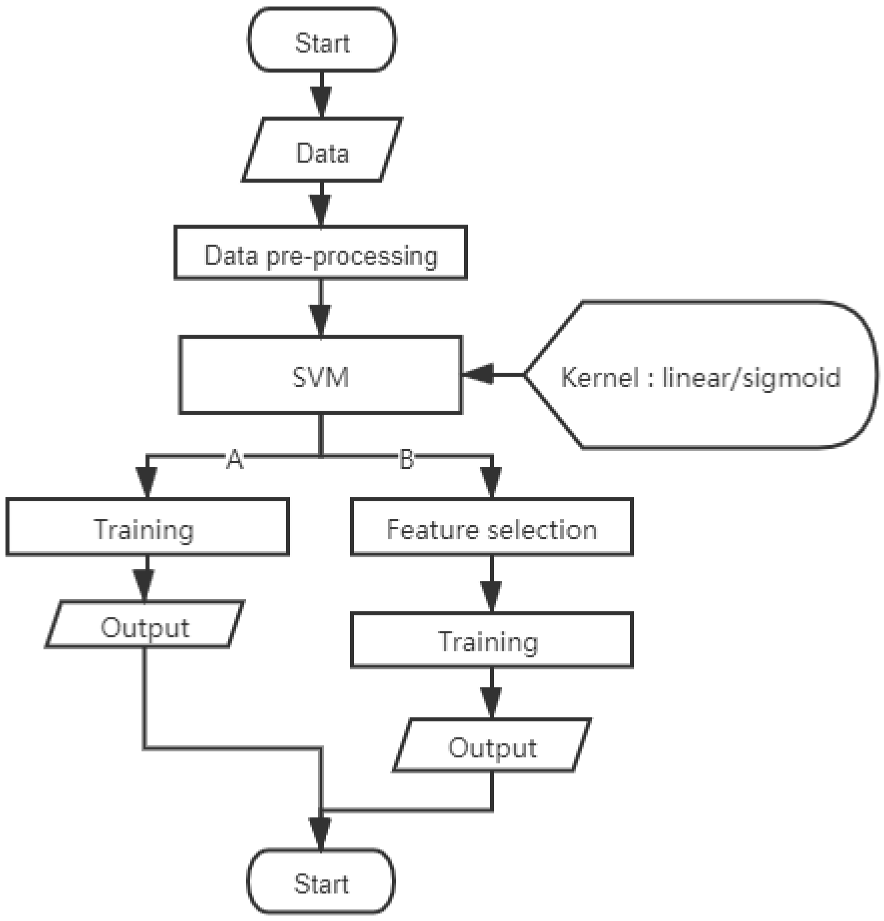


Figure 5-8 Flowchart of the algorithm using SVM

Figure 5-8 demonstrates the algorithm flow using SVM. Stage 1: Data pre-processing, including data cleaning and data normalization; Stage 2, model parameter tuning, including the kernel selection. The next stage is two parallel sub-experiments. Experiment A did not use feature selection, but directly conducted model training and output training and test results, while Experiment B conducted model training and output training and test results after feature selection. In the following experiments, parameter kernel will be inspected and adjusted.

6 EXPERIMENTS AND EVALUATION

Google's Colab was chosen as the experimental environment, Python3.7 as the programming language, TensorFlow as the deep learning framework, and python sklearn library as the provider of random forest and SVM. Accuracy and confusion matrices were chosen to evaluate experimental results. The tools and methods mentioned above was considered appropriate for 4 reasons:

1) Google Colab is a cloud-based Python development environment that can debug and run python programs in a browser without any download, and it can automatically save code on a google cloud drive.

2) TensorFlow is a free, open source machine learning platform that integrates popular machine learning libraries such as Keras;

3) Python is a rapidly growing programming language that can run on any server, making Python programs very easy to port.

4) Accuracy and confusion matrix are commonly used performance measures in machine learning research, especially in the dichotomy problem, that is, the field to which this project belongs.

6.1 Experiment Design

6.1.1 Experiment design and evaluation methods

To improve the performance of each model for web robot detection, the experiments in Table 6-1 are conducted. For the Keras sequential Model, the impact of kernal, dropout, features on the accuracy and time efficiency are examined. For the random forest Model, the impact of estimator number and features and time efficiency are examined; For the SVM model, the impact of kernels and features as well as time efficiency are assessed.

Table 6-1 Experiment design

EXPERIMENT PHASE	KERAS SEQUENTIAL MODEL (NN)	RF	SVM
1	Activation function	Number of estimators	Kernel
2	Dropout		
3	Feature selection	Feature selection	Feature selection
4	Time efficiency	Time efficiency	Time efficiency

Table 6-1 demonstrates the experiment design of the project. At phase 1, the kernel of Keras sequential model and SVM were adjusted. And the number of estimators in random forest was adjusted. At phase 2, the dropout of Keras sequential model was adjusted. At phase 3, feature selection was applied to all the models. At phase 4, time efficiency tests were conducted on every model.

Two performance measurements used in this project :1) confusion matrix, which was used to evaluate all models; 2) accuracy, which was used in the training process of neural network model. The following table illustrates a basic confusion matrix:

Table 6-2 Confusion matrix

	Positive Sample	Negative Sample
Classified as Positive	True Positive (TP)	False Positive (FP)
Classified as Negative	False Negative (FN)	True Negative (TN)

- TP: cases are classified as positive samples and are actually positive samples.
- FP: cases are classified as positive samples, but actually negative samples.
- FN: cases are not classified as positive samples, but they are actually positive samples.
- TN: cases are not classified as positive samples and are actually negative samples.
- Accuracy: ratio of the number of samples correctly classified to the total number of samples, (TP + TN) / (TP + TN + FP + FN).
- Precision: ratio of the number of samples correctly classified to the total number of samples classified, TP / (TP + FP).
- Recall: ratio of the number of samples correctly detected to the number of samples to be detected, (TP / TP + FN).

In this project, positive samples are log records generated by web robots, and negative samples are log records generated by humans.

6.1.2 Experiment environment

This section introduces the experimental environment of this project, including hardware equipment, software and configurations, and model parameters.

Table 6-3 Hardware and Software configurations

Hardware	Software
CPU: Intel Core i7-4720HQ (2.6GHz/L3 6M) Memory: 8GB DDR3 1600 Hard drive: 1TB HDD	Platform: Google Colab, TensorFlow 2.0 Programming language: Python3.7

Table 6-3 illustrates the environment of the project. The experiments of this project were performed on a laptop. Google Colab is a free online Python programming service from Google that integrates with TensorFlow, a popular deep learning framework. It can be run in a browser, as long as there is Internet access.

Table 6-4 Parameters of each model

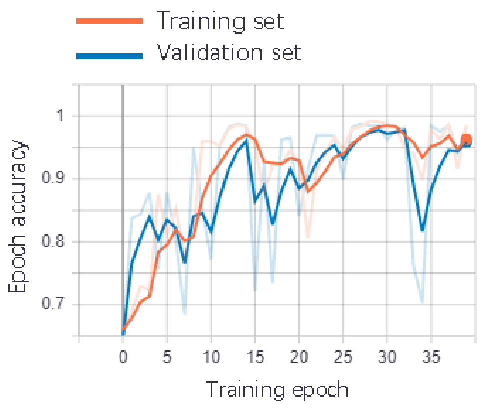
Keras Sequential Model	Random Forest	SVM
layer 1: feature layer layer 2: dense layer, 20 neural, activation function linear, dropout 0.6 layer 3 dense layer, 20 neural, activation function linear, dropout 0.6 layer 4 dense layer, 1 neural, activation function linear learning rate: 0.001 batch size: 32 epochs: 40 loss function: binary cross entropy	criterion: Gini max features: None max depth: None	C: 1.0 gamma: auto

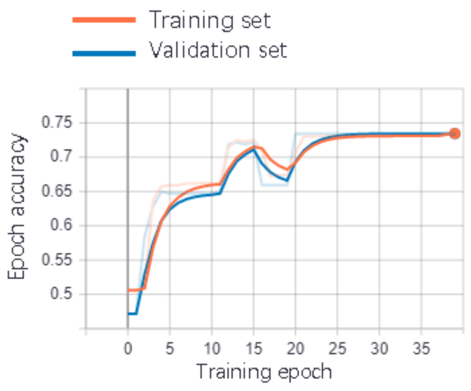
Table 6-4 shows the key parameters of each model. Keras sequential model is a linear stack of multiple neural network layers. The default learning rate of the model is 0.001 (adjustable), and the batch size is 32 (adjustable). Binary cross entropy is a loss function recommended in the Keras Sequential model document for binary classification problems. The parameters of the random forests and support vector machines shown in the table are all default, and the following experiments focused on the number of estimators in random forests and the kernels of SVM.

6.2 Experiments with TensorFlow Keras Sequential Model

6.2.1 Experiments with two different activation functions

In this period, all features were used to train the model, no dropout were employed, the variable between two experiments was kernel. The training epoch was set to 40 to observe the accuracy and overfitting during the training process. Overfitting means the model is over trained. The main phenomenon of overfitting is accuracy on training set surpass that on test set.





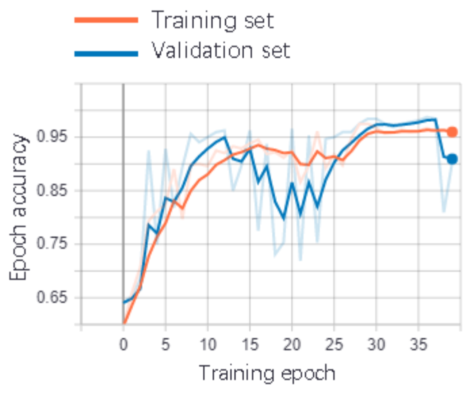
(a) Linear **(b) Sigmoid**

Figure 6-1 Epoch accuracy with different activation function

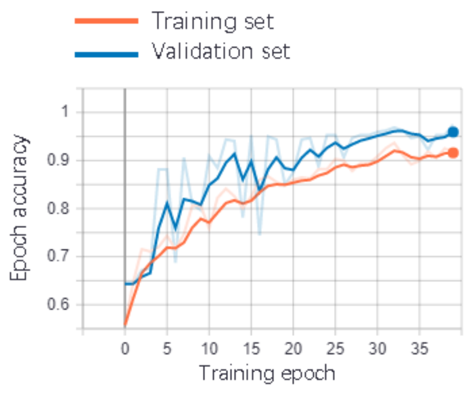
Figure 6-1 shows that sequential model with linear activation function has a better accuracy (0.9085) than the model with sigmoid activation function (0.7314). For the model with linear activation function, overfitting happened between epoch 8 and 20. And for the model with sigmoid activation function, there was no obvious sign of overfitting, but the peak epoch accuracy was much lower. So sequential model with linear kernel was selected for further experiment.

6.2.2 Experiments with dropouts

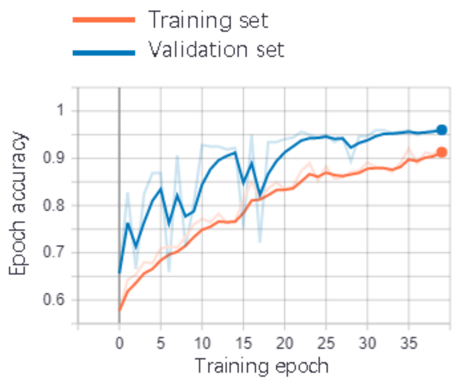
In this period, four different proportions of dropouts were applied to the model with linear activation function.



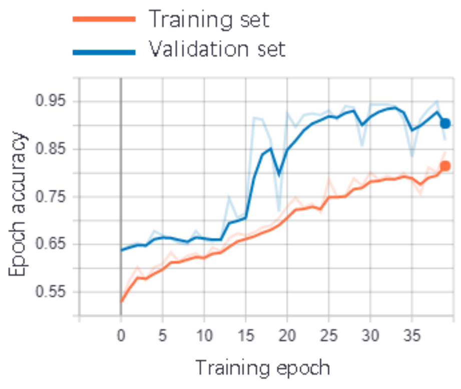
(a) 20% dropout



(b) 40% dropout



(c) 60% dropout



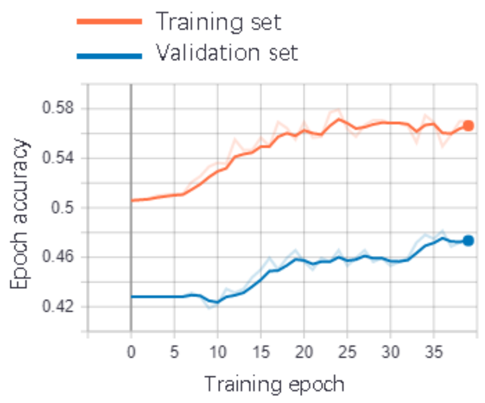
(d) 80% dropout

Figure 6-2 Epoch accuracy with different percentage of dropout applied

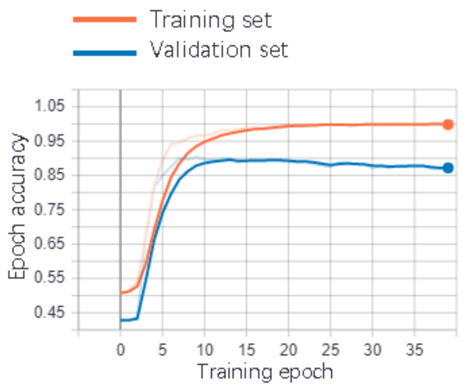
As shown in the graphs, over fitting gradually disappeared as dropout percentage went up. This suggested reducing model complexity could reduce overfitting. It can be inferred by comparing Figure 6-2 (c) and (d) that too much dropout may cause decrease in peak epoch accuracy. The best performance was achieved by 60% dropout experiments. Its test accuracy was 0.9725.

6.2.3 Experiments with selected features

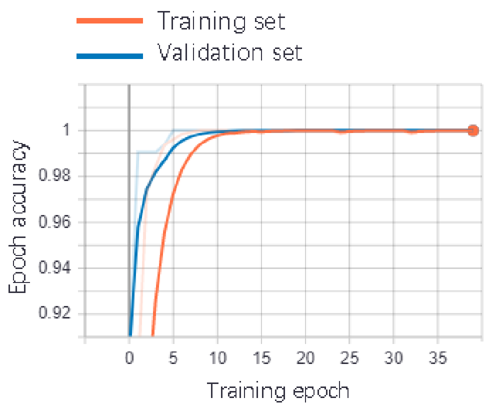
In this period, the model was trained with different number of features.



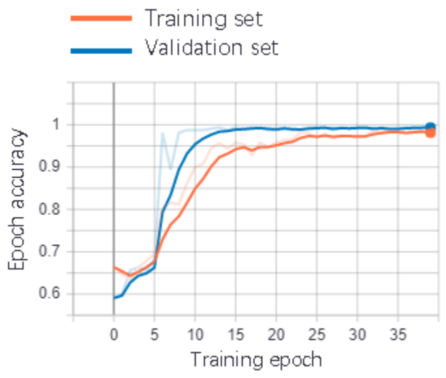
(a) top 2 features



(b) top 4 features



(c) top 6 features



(d) top 8 features

Figure 6-3 Epoch accuracy with different amount of features

Horizontal axis represents the number of epochs in training. The four subfigures in Figure 6-3 suggests that epoch accuracy can be improved by adding more features. But adding features may cause overfitting in different epoch of training. The best accuracy on validation set without overfitting is 100% using top 6 features. This may suggest that training with the selected features allow the model to obtain higher accuracy on the validation set while too much unimportant features may cause the accuracy of the model to decrease. The reason for this is phenomenon is that unimportant features will cause model parameters to tend to value that are not conducive to the performance of the model.

6.3 Experiments with Random Forest

6.3.1 Experiments with different amount of estimators

At this stage, all features were used to train the model, the number of estimators (trees) were set to 1, 10, 100 and 1000.

Table 6-5 Confusion matrixes of random forest with different number of trees

No. of Estimators	Original Type / Classified as	Human	Robot	Precision	Recall
1	Human	186	0	1.000	1.000
	Robot	0	214		
10	Human	186	1	1.000	0.9953
	Robot	0	213		
100	Human	186	1	1.000	0.9953
	Robot	0	213		
1000	Human	186	1	1.000	0.9953
	Robot	0	213		

Table 6-5 shows that as the number of trees increases, the accuracy did not change, but some robots are classified as human, in other words, false negative went up. Therefore, experiments were carried out with one-tree-Random Forest at next stage.

6.3.2 Experiments with selected features

At this stage, different amounts (2, 4, 6 and 8) of features were employed to train the random forest with one estimator.

Table 6-6 Confusion matrixes of one-estimator random forest with different number of features

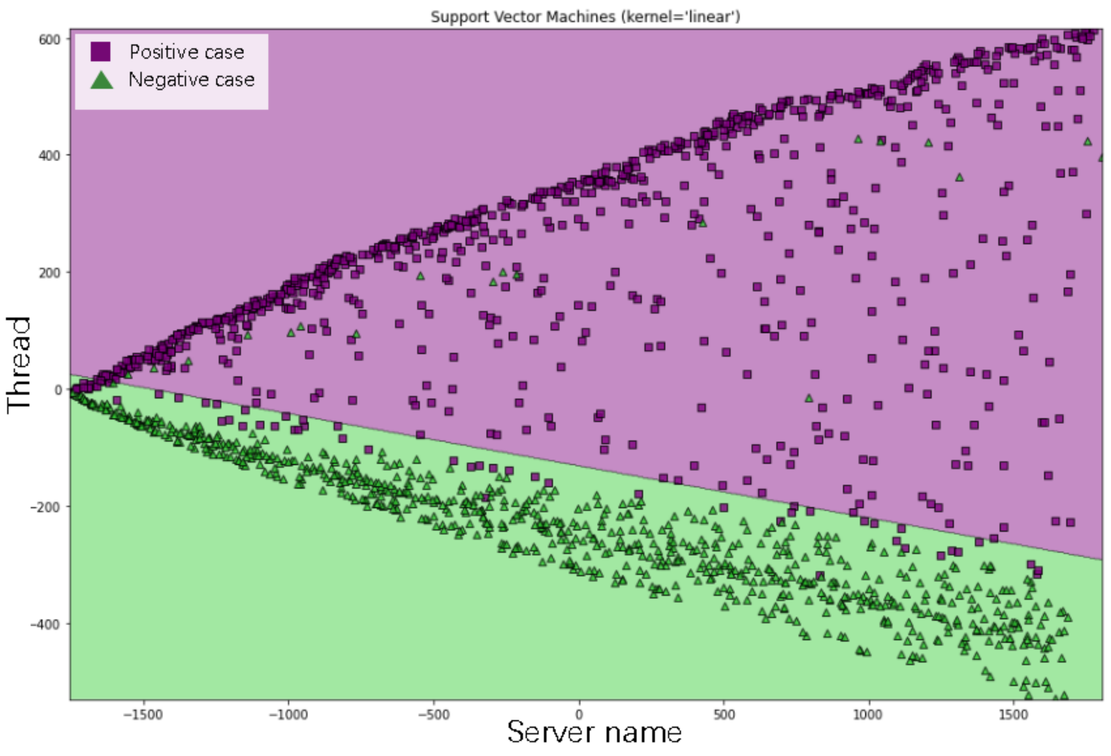
No. of Features	Original Type / Classified as	Human	Robot	Precision	Recall
2	Human	94	117	0.5123	0.4532
	Robot	92	97		
4	Human	182	6	0.9811	0.9720
	Robot	4	208		
6	Human	179	2	0.9680	0.9907
	Robot	7	212		
8	Human	184	2	0.9907	0.9907
	Robot	2	212		

As is shown in Table 6-6, Precision and Recall goes up as the number of training features increase. A quite high accuracy and recall can be achieved when using top 8 important features to train the model. Meanwhile the training time dropped significantly compared to using all features.

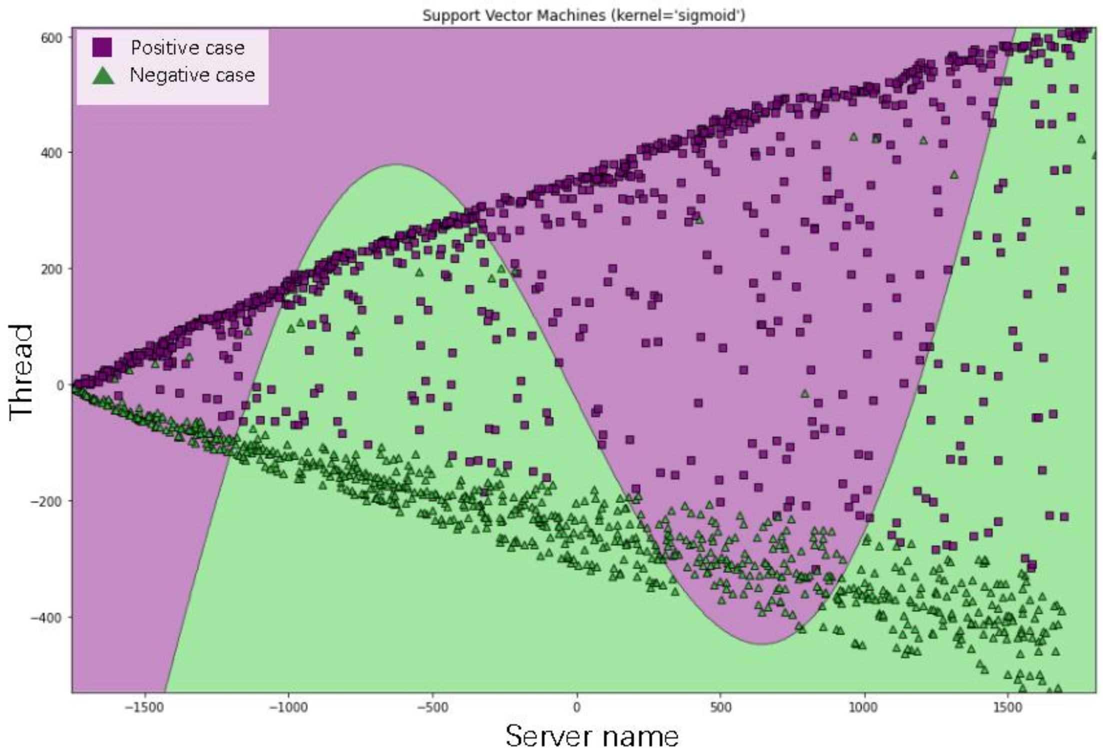
6.4 Experiments with SVM

6.4.1 Experiments with two different kernels

In this period, all the features were used to train the model with different kernels. Top 2 important features (server name and thread) were employed to plot the decision boundary. Decision boundary shows the distribution of samples in the problem space and how the kernel function separates two different samples., the outputs of the classifier are scattered in different regions with triangle and square symbols. Obviously, there are misclassified outputs.



(a) Linear kernel



(b) Sigmoid kernel

Figure 6-4 Decision region of SVM using different kernels

In Figure 6-4 (a) and (b), the abscissa represents the distribution of the sample in the server name parameter, and the ordinate represents the distribution of the sample in the thread parameter. Specifically, for a certain sample, there is a server name feature, a thread feature, and a label, indicating that it is a web robot (positive case) or a human (negative case). First, the positive cases are represented by purple squares, and the negative cases are represented by lime green triangles. Second, convert the pre-processed values of the two parameters again to make them conform to the range of the coordinate system. Then, according to the parameter value after conversion of each sample, the sample is put into the coordinate system to observe the distribution of the sample on these two parameters. By comparing Figure 6-4 (a) and (b), it can be concluded that the sample is more linear separable than sigmoid separable.

Table 6-7 Confusion matrixes of SVM using different kernels

Kernel	Original Type / Classified as	Human	Robot	Precision	Recall
Linear	Human	184	2	0.9906	0.9906
	Robot	2	212		
Sigmoid	Human	109	106	0.5838	0.5046
	Robot	77	108		

For further validation. Samples with all features were fed into two models with different kernel. After training and test, the result is shown in table 6-7. SVM with linear kernel obtained a precision of 0.9906 and a recall of 0.9906. SVM with sigmoid kernel reached a precision of only 0.5838 and recall of 0.5046. Therefore, the SVM with linear Kernel was taken to further experiments.

6.4.2 Experiments with selected features

At this stage, different amounts (2, 4, 6 and 8) of features were employed to train the model. The performance of the model on test set are showed in the confusion matrixes below.

Table 6-8 Confusion matrixes of SVM with linear kernel with different number of features

No. of Features	Original Type / Classified as	Human	Robot	Precision	Recall
2	Human	186	214	0	0
	Robot	0	0		
4	Human	158	26	0.8704	0.8785
	Robot	28	188		
6	Human	152	6	0.8595	0.9720
	Robot	34	208		
8	human	183	2	0.9860	0.9907
	Robot	3	212		

From the confusion matrixes shown in table 6-8, it can be indicated that SVM cannot tell robots from human visitors by the top 2 important features. It classified all the robot samples as human. The precision and recall go up as more features were employed to train the model.

6.5 Evaluation and Discussion

To identify which algorithm/model is the best offline web robot detection algorithm/model, both accuracy and running time are assessed. In order to examine the running time of each algorithm, the average training time and accuracy of each algorithm on different numbers of features at a training process were calculated.

6.5.1 The importance of feature selection

This section will make a summary of the feature selection experiment mentioned above.

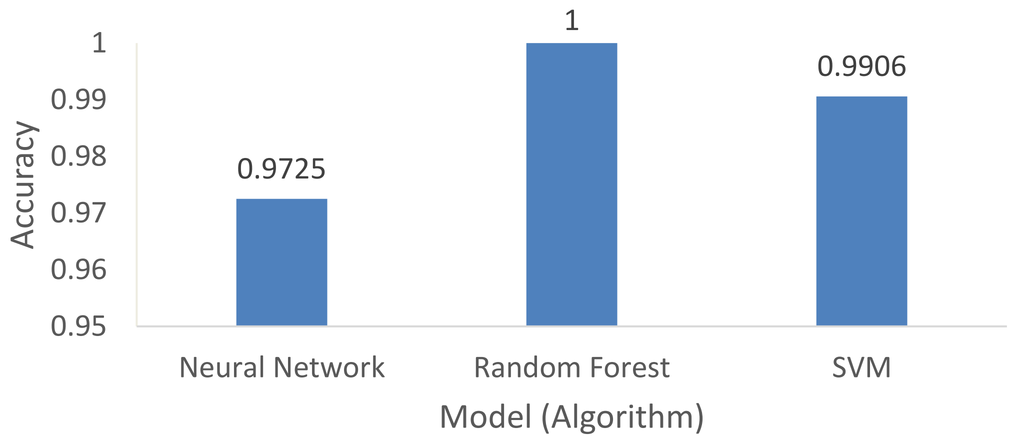


Figure 6-5 Accuracy of models without feature selection

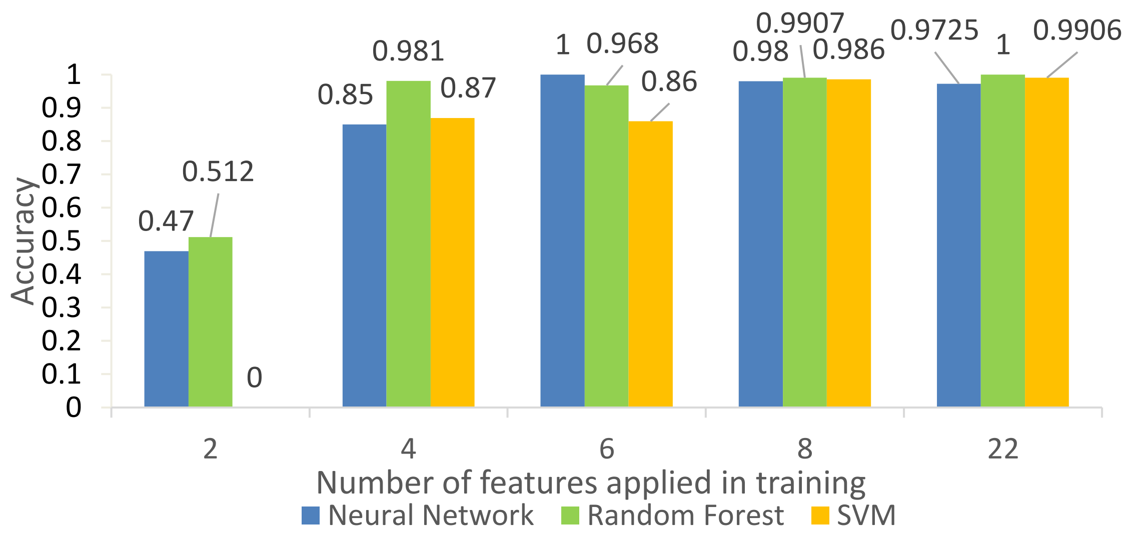


Figure 6-6 Accuracy of models with feature selection

Figure 6-5 shows that when no feature selection applied, random forest has the highest accuracy of 100%, SVM 99.06%, and NN 97.25%.

Figure 6-6 shows that when features were added to training in terms of their importance from high to low, the variation of model accuracy presents different trends. When top 2 important features applied, the accuracy of SVM model is 0, it classified every sample as human. When 2 more features applied, the accuracy of all models experienced a sharp growth: NN 38%, RF 46.9%, SVM 87%. When the top 6 features were applied to the training, the NN reached its peak accuracy of 100%, while the accuracy of RF and SVM declined slightly. As more features were added to the training, the accuracy of the NN declined slowly, while SVM and SVM rose slowly. Here, the NN and the RF are probably the best algorithms.

6.5.2 Time efficiency

This section will focus on the time efficiency of each model trained with different amounts of features.

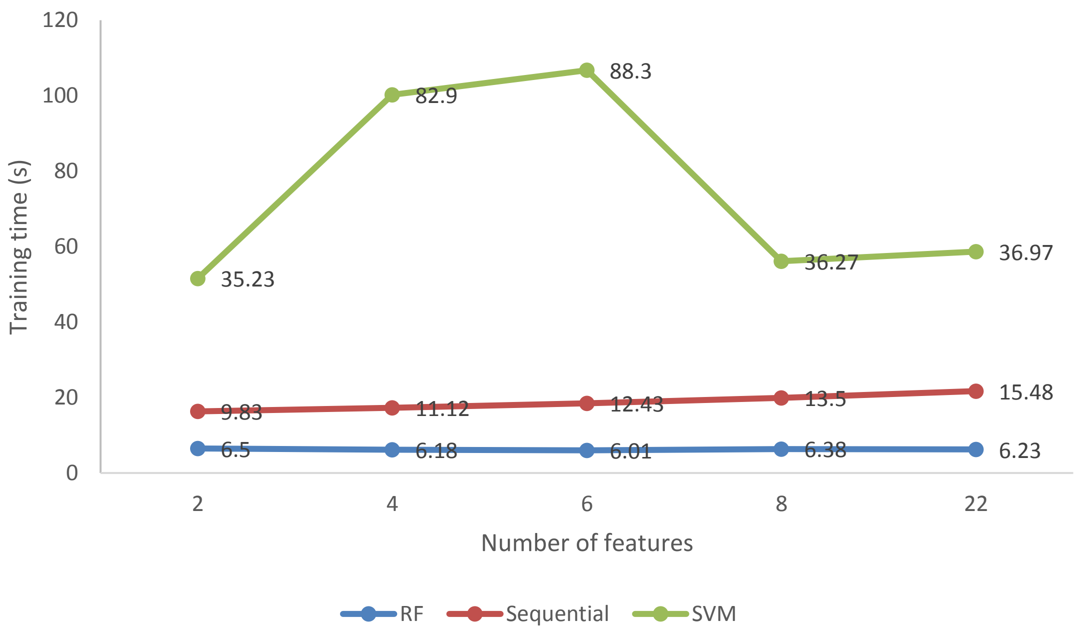


Figure 6-7 Average training time using different number of features

As shown in Figure 6-7 from the perspective of the training time of the model, the random forest model has an absolute advantage, and the SVM model takes the most time. As for the Keras sequential model, the time it takes is acceptable. Perhaps it could be achieved to reduce the training time by reducing the number of neurons, but this may cause the accuracy of the model to decrease.

From the perspective of the influence of the number of features on the training time, with the increase of the number of training features, the training time of the random forest model has not increased significantly, the training time of the Keras sequential model increases linearly, and the training time of the SVM model also shows Increasing trend.

Although both neural networks and random forests can achieve 100% accuracy under certain conditions, when the same accuracy is achieved, the training time of random forests is 50% less than that of neural networks. Here, it can be concluded that random forest is the best web crawler detection algorithm regarding both accuracy and training time.

6.5.3 Discussion

In this project, two challenges mentioned in Table 3-5 were addressed: 1) the curse of dimensionality was solved by applying feature selection; 2) very high accuracy is achieved when the appropriate number of features is used for model training. Specifically, in this project, when the features were added to the training process of the neural network in the order of decreasing importance, the accuracy of the neural network increases gradually, and reaches the highest accuracy 100% when the number of features was 6, and then began to decrease. This suggested that the use of feature selection could ease the dimensional curse. For the model using random forest, when all 22 features selected were applied to training and the number of estimators was set to 1, the highest accuracy of 100% was achieved, and the training time of the model was about 50% less than that of the neural network. In addition, the data set of this project came from an Internet industry. Although the data set was not able to mark and distinguish between good web crawlers and bad ones, it can be inferred that these web crawler records were not expected, or at least worthy of attention. Because no enterprise will spend time on things they don't care about.

The data set contains only positive and negative cases, it does not contain more types of web bots. This make the web robot detection become a decision-making problem or a binary classification problem. To train models that can recognize more types of web robots, a data set with detailed tags is required. One way to build such a data set is to build a website and design different types of web bots. Let the bots crawl the information on the site and produce a set of tagged weblog data. This could be future work.

TensorFlow was used to identify web robot logs for the first time. Even though the Keras sequential model produced a very high accuracy of 100% when top 6 features were used in training period, it is slower than random forest. To improve the training speed of the model, it might be possible to further reduce the number of neurons.

In random forest experiments, only the number of estimators was used as a variable, and high accuracy was achieved when the number of estimators (trees) was set to 1 and no feature-selection applied. In fact, the random forest has more parameters that can affect the training results, such as the depth of the tree, the maximum number of features and so on. This could be studied in future. For more complex samples, parameters tuning can be used to optimize the model.

Request Time and Request ID were not accounted in the influential features. However, the experimental results show that these two characteristics have significant effects on the accuracy of all models. This indicates web robots may usually come from specific IDs at specific times

7 CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

The project aims to develop an offline system that can effectively detect malicious web robots, which is not only conducive to network traffic cleaning, but also conducive to improving the network security of IoT systems and services. The key contributions of the research are: 1) it provided a systematic methodology to address the web robot detection problem based on the log file from industrial company; 2) it provided a methodology for feature selection, overcoming the challenge of curse of dimensionality; 3) it investigated three types of machine learning techniques based real data from industry, making a big progress in the accuracy of off-line web robot detection. Overall, this project solved two of the challenges mentioned in Table 3-5: overcoming the curse of dimensionality by applying feature selection and improving the accuracy of offline web robot detection by using proper features and parameters.

Feature engineering, modelling, model implementation, parameters impact and verification of the developed models were conducted with comprehensive experiments. The key findings are listed as below:

Feature Engineering: The investigated dataset was a csv file with 2000 web log records. It has 43 columns in which 42 were features and 1 was the decision labels. After data cleaning and data transformation 21features were reserved for feature selection. During feature selection information gain and random forest were applied to evaluate feature importance.

Machine learning models: Three models (sequential model, random forest, and SVM) were developed as web robot classifiers. When feature selection was not performed, the accuracy of the random forest algorithm was 3% higher than the Keras sequential Model, and 1% higher than the SVM algorithm, reaching 100%. The training time of the random forest saves 59.1% compared to the Keras sequential model and 83.1% compared to the SVM.

Feature importance to the decision is assessed. The most important features were identified. They are *server name, thread, request time, request id, os, app version*. Among them, the request id and request event had the greatest impact on the accuracy of the model. They increased the accuracy of the neural network by 38%, the random forest by 46.9%, the SVM by 87%.

The impact of features on the performance of model for tested data is examined. The accuracy of random forest and SVM fluctuated upward with the number of features. The accuracy of the Keras sequential model reached a peak of 100% when using the six most important features, and then decreases slowly. This phenomenon indicated that feature selection has an important impact on the performance of models. For different models, the influence of a certain feature is not the same. That is, the addition of a certain feature may help improve the accuracy of one model, but it will reduce the accuracy of another model. In terms of training time, random forest is still the most time-saving algorithm.

Impact of parameters on the performance of models for the tested data. Keras sequential model (NN) and random forest (RF) obtained accuracy of 0.47 and 0.512 with *server name* and *thread,* while SVM obtained none. When *request id* and *request time* were added, accuracy of three models increased significantly: NN to 0.85, RF to 0.981, and SVM to 0.87. when *os* and *app version* were added, NN reached its peak accuracy of 1, RF and SVM experienced a slight drop to 0.968 and 0.86. when *page city id* and *locate city id* were added. The accuracy of NN dropped a bit to 0.98 while RF and SVM increased to 0.9907 and 0.986.

This project uses the confusion matrix as a performance matrix, combined with training time, to measure the performance of a model. Although both neural networks and random forests can achieve 100% accuracy under certain conditions, when the same accuracy is achieved, the training time of random forests is 50% less than that of neural networks. Thus, the conclusion of this project is that random forest is the best model for offline detection of web robots.

7.2 Future Work

The present work focused on off-line web log analysis using machine-learning techniques. In the literature, there are very few methods to investigate the distributed web robots. When distributed web robots are performing DDoS attacks, off-line methods are not able to react to handle these attacks in real-time. The further work will investigate the following areas:

- Investigate online detection of web robot based on user behaviours pattern learning.
- Develop hardware equipment for web crawler detection based on machine learning.
- Create web robot datasets for future research and share them on public repository.
- Investigate cloud-computing based web robot detection method and service.

REFERENCES

A. Cabri, G. Suchacka, S. Rovetta, and F. Masulli, Online Web Bot Detection Using a Sequential Classification Approach, Proceedings - 20th International Conference on High Performance Computing and Communications, 16th International Conference on Smart City and 4th International Conference on Data Science and Systems, HPCC/ SmartCity/ DSS 2018, 2018, pp. 1536–1540.

N. Algiriyage, Offline Analysis of Web Logs to Identify Offensive Web Crawlers, International Research Symposium on Pure and Applied Sciences (IRSPAS 2017), Kelaniya, Sri Lanka, 20 Oct. 2017.

M. Catalin, and A. Cristian, An Efficient Method In Pre-processing Phase of Mining Suspicious Web Crawlers, 2017 21st International Conference on System Theory, Control and Computing, ICSTCC 2017, 2017, pp. 272–277.

T. Gržinić, L. Mršić, and J. Šaban, Lino - An Intelligent System for Detecting Malicious Web-Robots, Intelligent Information and Database Systems. Cham: Springer International Publishing, 2015, pp. 559–568.

Distil Networks, 2019 Bad bot report available on https://www.bluecubesecurity.com/wp-content/uploads/bad-bot-report-2019LR.pdf, accessed on 11/5/2020, pp. 8.

S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach (2nd ed.). Prentice Hall, 2002, ISBN 978-0137903955.

A. Menshchikov, A. Komarova, Y. Gatchin, A. Korobeynikov and N. Tishukova, A Study of Different Web-crawler Behavior, The 20th Conference of Open Innovations Association (FRUCT), St. Petersburg, 2017, pp. 268-274.

D. Doran and S. S. Gokhale, Web Robot Detection Techniques: Overview and Limitations, Data Mining Knowledge Discovery, 2011, pp. 183–210, DOI 10.1007/s10618-010-0180-z.

K. Ma, R. Jiang, M. Dong, Y. Jia, and A. Li, Neural Network Based Web Log Analysis for Web Intrusion Detection, International Conference on Security, Privacy, and Anonymity in Computation, Communication, and Storage, 2017, pp. 194–204. ISBN: 978-3-319-72395-2

S. Rovetta, G. Suchacka, and F. Masulli, Bot Recognition in a Web Store: An Approach Based on Unsupervised Learning, Journal of Network and Computer Applications, 157, 102577, 2020.

C. Iliou, T. Kostoulas, T. Tsikrika, V. Katos, S. Vrochidis, and Y. Kompatsiaris, Towards a Framework for Detecting Advanced Web Bots, Proceedings of the 14th International Conference on Availability, Reliability and Security, 2019, pp. 1-10.

D. Doran and S. S. Gokhale, An Integrated Method for Real Time and Offline Web Robot Detection, Expert Syst., 33(6), pp. 592–606, 2016, DOI: 10.1111/exsy.12184.

P. V. Patankar, and S. Jangale, Malicious Web Crawler Detection Using Intrusion Detection System, IJMTER, 3(3), pp. 364–371, 2016.

M. Abdullahi, S. Aliyu, and S. B. Junaidu, An Enhanced Intrusion Detection System Using Honeypot and CAPTCHA Techniques, FUDMA Journal of Science, 3(3), 2019, pp. 202–209.

R. Selvaraj, V. M. Kuthadi and T. Marwala, Honey Pot: A Major Technique for Intrusion Detection, Proceedings of the 2nd International Conference on Computer and Communication Technologies, IC3T 2015, Hyderabad, India, 2015.

Y. Guo, J. Shi, Z. Cao, C. Kang, G. Xiong and Z. Li, Machine Learning Based Cloud Bot Detection Using Multi-Layer Traffic Statistics, 2019 IEEE 21st International Conference on High Performance Computing and Communications, IEEE 17th International Conference on Smart City, IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China, 2019, pp. 2428-2435.

A. Balla, A. Stassopoulou and M. D. Dikaiakos, Real-time Web Crawler Detection, 18th International Conference on Telecommunications, Ayia Napa, Cyprus, 8(11), 2011, DOI: 10.1109/CTS.2011.5898963.

J.R. Koza, F.H. Bennett, D. Andre, and M.A. Keane, Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming, In Artificial Intelligence in Design '96. Dordrecht: Springer Netherlands, 1996, pp. 151–170.

A. Lagopoulos, G. Tsoumakas, and G. Papadopoulos, Web robot detection: A semantic approach, Proceedings - International Conference on Tools with Artificial Intelligence, 2018, pp. 968–974.

S.F. McKenna, Detection and Classification of Web Robots with Honeypots. MSc thesis. Naval Postgraduate School, 2016.

D. Kuhlman, A Python Book: Beginning Python, Advanced Python, and Python Exercises. Section 1.1. Archived from the original (PDF) on 23 June 2012.

Python Software Foundation. Retrieved 24 April 2012, second section Fans of Python use the phrase "batteries included" to describe the standard library, which covers everything from asynchronous processing to zip files.

M. Schmidt, Web crawler. Master of Science. Governors State University, 2015.

All symbols in TensorFlow, Retrieved February 18, 2018.

T. V. Udapure, R.D. Kale, and R.C. Dharmik, Study of Web Crawler and its Different Types, IOSR Journal of Computer Engineering, 16(1), 2014, pp. 01–05.

M. Zabihimayvan, R. Sadeghi, H.N. Rude, and D. Doran, A Soft Computing Approach for Benign and Malicious Web Robot Detection, Expert Systems with Applications, 87 Elsevier Ltd., 2017, pp. 129–140.J. Rajabnia and M. V. Jahan, Web Robot Detection with Fuzzy Inference System Based on NNGE, available on https://www.academia.edu/8753500/Web_Robot_Detection_With_Fuzzy_Inference_System_Based_On_NNGE, accessed on 20/01/2020.

G. Suchacka and M. Sobkow, Detection of Internet Robots Using a Bayesian Approach, Proc-2015 IEEE 2nd Int. Conf. Cybern. (CYBCONF), Gdynia, Poland, 2015, pp. 365–370, DOI: 10.1109/CYBConf.2015.7175961.

D. S. Sisodia, S. Verma, and O. P. Vyas, Agglomerative Approach for Identification and Elimination of Web Robots from Web Server Logs to Extract Knowledge about Actual Visitors, J. Data Anal. Inf. Process., 3(1), 2015, pp. 1–10.

R. Haidar and S. Elbassuoni, Website Navigation Behavior Analysis for Bot Detection, Proc. - 2017 Int. Conf. Data Sci. Adv. Anal. DSAA 2017, Tokyo, Japan, 2017, pp. 60–68, DOI: 10.1109/DSAA.2017.13.

T. H. Sardar and Z. Ansari, Detection and Confirmation of Web Robot Requests for Cleaning the Voluminous Web Log Data, 2014 International Conference on the Impact of E-Technology on US (IMPETUS), Bangalore, 2014, pp. 13-19, DOI: 10.1109/IMPETUS.2014.6775871.

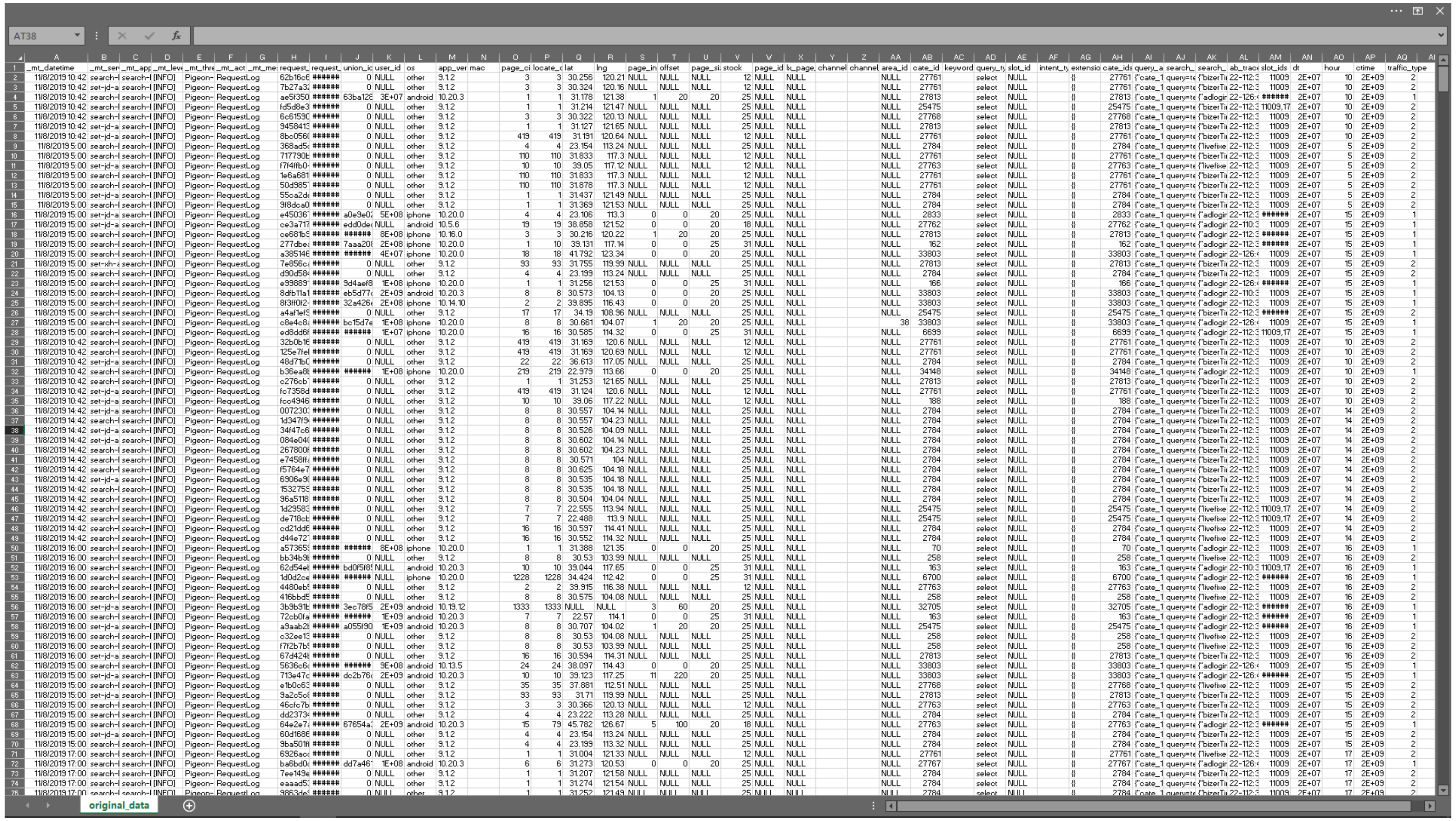
D. Stevanovic and A. A. N. Vlajic, Feature Evaluation for Web Crawler Detection with Data Mining Techniques, Expert Systems with Applications, 39(10), 2012, pp. 8707-8717.

A. Mason, Y. Zhao, and H. He, et al. Online Time-Series Anomaly Detection at Scale, Cyber Science 2019, University of Oxford, 3-4 Jun. 2019. DOI: 10.1109/CyberSA.2019.8899398

H. Chen, H. He, and A. Starr, An Overview of Web Robots Detection Techniques, Cyber Science 2020, Cranfield University, 15 Jun. 2020. DOI: 10.1109/CyberSecurity49315.2020.9138856.

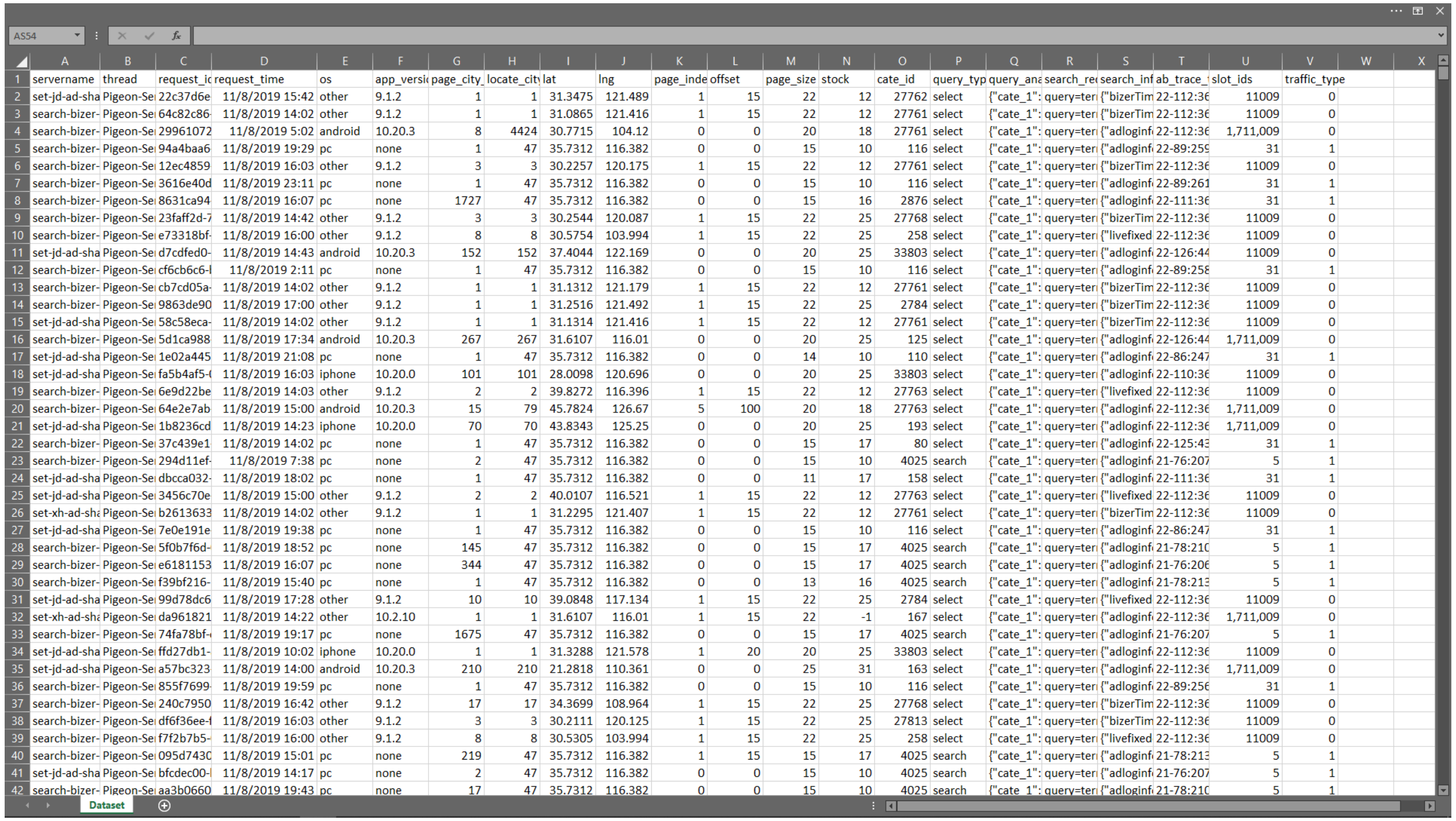
APPENDIX

Appendix A Dataset Files



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ
1	_mt_datetime	_mt_ser	_mt_app	_mt_leve	_mt_thre	_mt_act	_mt_me:	request_	request_	union_ic	user_id	os	app_ver	mac	page_ci	locate_c	lat	lng	page_in	offset	page_si:	stock	page_id	lx_page_	channel	channel	area_id	cate_id	keyword	query_ty	slot_id	intent_ty	extensio	cate_ids	query_a	search_	search_	ab_trace	slot_ids	dt	hour	ctime	traffic_type
2	11/8/2019 10:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		62b16c6	######	0	NULL	other	9.1.2		3	3	30.256	120.21	NULL	NULL	NULL	12	NULL	NULL			NULL	27761		select	NULL		{}	27761	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	10	2E+09	2
3	11/8/2019 10:42	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		7b27a3	######	0	NULL	other	9.1.2		3	3	30.324	120.16	NULL	NULL	NULL	12	NULL	NULL			NULL	27761		select	NULL		{}	27761	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	10	2E+09	2
4	11/8/2019 10:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		ae5f350	######	63ba128	3E+07	android	10.20.3		1	1	31.178	121.38	1	20	20	25	NULL	NULL			NULL	27813		select	NULL		{}	27813	{"cate_1	query=te	{"adlogir	22-126:	######	2E+07	10	2E+09	1
5	11/8/2019 10:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		fd5d8e3	######	0	NULL	other	9.1.2		1	1	31.214	121.47	NULL	NULL	NULL	25	NULL	NULL			NULL	25475		select	NULL		{}	25475	{"cate_1	query=te	{"bizerTi	22-112:3	11009,17	2E+07	10	2E+09	2
6	11/8/2019 10:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		6c6159	######	0	NULL	other	9.1.2		3	3	30.322	120.13	NULL	NULL	NULL	25	NULL	NULL			NULL	27768		select	NULL		{}	27768	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	10	2E+09	2
7	11/8/2019 10:42	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		9458413	######	0	NULL	other	9.1.2		1	1	31.127	121.65	NULL	NULL	NULL	25	NULL	NULL			NULL	27813		select	NULL		{}	27813	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	10	2E+09	2
8	11/8/2019 10:42	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		8bc056	######	0	NULL	other	9.1.2		419	419	31.191	120.64	NULL	NULL	NULL	12	NULL	NULL			NULL	27761		select	NULL		{}	27761	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	10	2E+09	2
9	11/8/2019 5:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		368ad5	######	0	NULL	other	9.1.2		4	4	23.154	113.24	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"livefixe	22-112:3	11009	2E+07	5	2E+09	2
10	11/8/2019 5:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		717790b	######	0	NULL	other	9.1.2		110	110	31.833	117.3	NULL	NULL	NULL	12	NULL	NULL			NULL	27761		select	NULL		{}	27761	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	5	2E+09	2
11	11/8/2019 5:00	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		f7f4ffb0-	######	0	NULL	other	9.1.2		10	10	39.05	117.12	NULL	NULL	NULL	12	NULL	NULL			NULL	27763		select	NULL		{}	27763	{"cate_1	query=te	{"livefixe	22-112:3	11009	2E+07	5	2E+09	2
12	11/8/2019 5:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		1e6a681	######	0	NULL	other	9.1.2		110	110	31.833	117.3	NULL	NULL	NULL	12	NULL	NULL			NULL	27761		select	NULL		{}	27761	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	5	2E+09	2
13	11/8/2019 5:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		50d985	######	0	NULL	other	9.1.2		110	110	31.878	117.3	NULL	NULL	NULL	12	NULL	NULL			NULL	27761		select	NULL		{}	27761	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	5	2E+09	2
14	11/8/2019 5:00	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		55ca2d	######	0	NULL	other	9.1.2		1	1	31.437	121.49	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	5	2E+09	2
15	11/8/2019 5:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		9f8dca0	######	0	NULL	other	9.1.2		1	1	31.369	121.53	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	5	2E+09	2
16	11/8/2019 15:00	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		e450367	######	a0e9e0	5E+08	iphone	10.20.0		4	4	23.106	113.3	0	0	20	25	NULL	NULL			NULL	2833		select	NULL		{}	2833	{"cate_1	query=te	{"adlogir	22-112:3	######	2E+07	15	2E+09	1
17	11/8/2019 15:00	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		ce3a717	######	edd0de	NULL	android	10.5.6		19	19	38.858	121.52	0	0	20	18	NULL	NULL			NULL	27762		select	NULL		{}	27762	{"cate_1	query=te	{"adlogir	22-110:3	11009	2E+07	15	2E+09	1
18	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		ce681b	######	######	8E+08	iphone	10.16.0		3	3	30.216	120.22	1	20	20	25	NULL	NULL			NULL	27813		select	NULL		{}	27813	{"cate_1	query=te	{"adlogir	22-112:3	######	2E+07	15	2E+09	1
19	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		277dbe	######	7aaa20	2E+08	iphone	10.20.0		1	10	39.131	117.14	0	0	25	31	NULL	NULL			NULL	162		select	NULL		{}	162	{"cate_1	query=te	{"adlogir	22-112:3	######	2E+07	15	2E+09	1
20	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		a38514	######	######	4E+07	iphone	10.20.0		18	18	41.792	123.34	0	0	20	25	NULL	NULL			NULL	33803		select	NULL		{}	33803	{"cate_1	query=te	{"adlogir	22-126:	11009	2E+07	15	2E+09	1
21	11/8/2019 15:00	set-xh-a	search-l	[INFO]	Pigeon-	RequestLog		7e856c	######	0	NULL	other	9.1.2		93	93	31.755	119.99	NULL	NULL	NULL	25	NULL	NULL			NULL	27813		select	NULL		{}	27813	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	15	2E+09	2
22	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		d90d58	######	0	NULL	other	9.1.2		4	4	23.199	113.24	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	15	2E+09	2
23	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		e998891	######	9d4aef8	1E+08	iphone	10.20.0		1	1	31.256	121.53	0	0	25	31	NULL	NULL			NULL	166		select	NULL		{}	166	{"cate_1	query=te	{"adlogir	22-126:	######	2E+07	15	2E+09	1
24	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		8dfb11a1	######	eb5d77	2E+09	android	10.20.3		8	8	30.573	104.13	0	0	20	25	NULL	NULL			NULL	33803		select	NULL		{}	33803	{"cate_1	query=te	{"adlogir	22-110:3	11009	2E+07	15	2E+09	1
25	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		8f3ff0f2-	######	32a426	2E+08	iphone	10.14.10		2	2	39.895	116.43	0	0	20	25	NULL	NULL			NULL	33803		select	NULL		{}	33803	{"cate_1	query=te	{"adlogir	22-112:3	11009	2E+07	15	2E+09	1
26	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		a4af1ef	######	0	NULL	other	9.1.2		17	17	34.19	108.96	NULL	NULL	NULL	25	NULL	NULL			NULL	25475		select	NULL		{}	25475	{"cate_1	query=te	{"bizerTi	22-112:3	######	2E+07	15	2E+09	2
27	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		c8e4c8	######	bc15d7e	1E+08	iphone	10.20.0		8	8	30.661	104.07	1	20	20	25	NULL	NULL			38	33803		select	NULL		{}	33803	{"cate_1	query=te	{"adlogir	22-126:	11009	2E+07	15	2E+09	1
28	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		ed8dd6f	######	######	1E+07	iphone	10.20.0		16	16	30.585	114.32	0	0	25	31	NULL	NULL			NULL	6699		select	NULL		{}	6699	{"cate_1	query=te	{"adlogir	22-112:3	11009,17	2E+07	15	2E+09	1
29	11/8/2019 10:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		32b0b1	######	0	NULL	other	9.1.2		419	419	31.169	120.6	NULL	NULL	NULL	12	NULL	NULL			NULL	27761		select	NULL		{}	27761	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	10	2E+09	2
30	11/8/2019 10:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		125e7fe	######	0	NULL	other	9.1.2		419	419	31.169	120.69	NULL	NULL	NULL	12	NULL	NULL			NULL	27761		select	NULL		{}	27761	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	10	2E+09	2
31	11/8/2019 10:42	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		48d71b0	######	0	NULL	other	9.1.2		22	22	36.613	117.05	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	10	2E+09	2
32	11/8/2019 10:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		b36ea8	######	######	1E+08	iphone	10.20.0		219	219	22.979	113.66	0	0	20	25	NULL	NULL			NULL	34148		select	NULL		{}	34148	{"cate_1	query=te	{"adlogir	22-112:3	11009	2E+07	10	2E+09	1
33	11/8/2019 10:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		c276cb	######	0	NULL	other	9.1.2		1	1	31.253	121.65	NULL	NULL	NULL	25	NULL	NULL			NULL	27813		select	NULL		{}	27813	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	10	2E+09	2
34	11/8/2019 10:42	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		fc7358d	######	0	NULL	other	9.1.2		419	419	31.124	120.6	NULL	NULL	NULL	12	NULL	NULL			NULL	27761		select	NULL		{}	27761	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	10	2E+09	2
35	11/8/2019 10:42	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		fcc4946	######	0	NULL	other	9.1.2		10	10	39.06	117.22	NULL	NULL	NULL	12	NULL	NULL			NULL	188		select	NULL		{}	188	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	10	2E+09	2
36	11/8/2019 14:42	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		007230	######	0	NULL	other	9.1.2		8	8	30.557	104.14	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	14	2E+09	2
37	11/8/2019 14:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		1d347f9	######	0	NULL	other	9.1.2		8	8	30.557	104.23	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	14	2E+09	2
38	11/8/2019 14:42	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		34f47c6	######	0	NULL	other	9.1.2		8	8	30.526	104.09	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	14	2E+09	2
39	11/8/2019 14:42	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		084e04	######	0	NULL	other	9.1.2		8	8	30.602	104.14	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	14	2E+09	2
40	11/8/2019 14:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		267800	######	0	NULL	other	9.1.2		8	8	30.602	104.23	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	14	2E+09	2
41	11/8/2019 14:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		e7458ff	######	0	NULL	other	9.1.2		8	8	30.571	104	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	14	2E+09	2
42	11/8/2019 14:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		f5764e7	######	0	NULL	other	9.1.2		8	8	30.625	104.18	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	14	2E+09	2
43	11/8/2019 14:42	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		6906e9	######	0	NULL	other	9.1.2		8	8	30.535	104.18	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	14	2E+09	2
44	11/8/2019 14:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		153275	######	0	NULL	other	9.1.2		8	8	30.535	104.18	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	14	2E+09	2
45	11/8/2019 14:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		96a5118	######	0	NULL	other	9.1.2		8	8	30.504	104.04	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	14	2E+09	2
46	11/8/2019 14:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		1d29583	######	0	NULL	other	9.1.2		7	7	22.555	113.94	NULL	NULL	NULL	25	NULL	NULL			NULL	25475		select	NULL		{}	25475	{"cate_1	query=te	{"livefixe	22-112:3	11009,17	2E+07	14	2E+09	2
47	11/8/2019 14:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		de718c	######	0	NULL	other	9.1.2		7	7	22.488	113.9	NULL	NULL	NULL	25	NULL	NULL			NULL	25475		select	NULL		{}	25475	{"cate_1	query=te	{"livefixe	22-112:3	11009,17	2E+07	14	2E+09	2
48	11/8/2019 14:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		cd21dd6	######	0	NULL	other	9.1.2		16	16	30.597	114.41	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"livefixe	22-112:3	11009	2E+07	14	2E+09	2
49	11/8/2019 14:42	search-l	search-l	[INFO]	Pigeon-	RequestLog		d44e72	######	0	NULL	other	9.1.2		16	16	30.552	114.32	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"livefixe	22-112:3	11009	2E+07	14	2E+09	2
50	11/8/2019 16:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		a57365	######	######	8E+08	iphone	10.20.0		1	1	31.388	121.35	0	0	20	25	NULL	NULL			NULL	70		select	NULL		{}	70	{"cate_1	query=te	{"adlogir	22-112:3	11009	2E+07	16	2E+09	1
51	11/8/2019 16:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		bb34b9	######	0	NULL	other	9.1.2		8	8	30.53	103.99	NULL	NULL	NULL	25	NULL	NULL			NULL	258		select	NULL		{}	258	{"cate_1	query=te	{"livefixe	22-112:3	11009	2E+07	16	2E+09	2
52	11/8/2019 16:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		62d54e	######	bd0f5f8	NULL	android	10.20.3		10	10	39.044	117.65	0	0	25	31	NULL	NULL			NULL	163		select	NULL		{}	163	{"cate_1	query=te	{"adlogir	22-110:3	11009,17	2E+07	16	2E+09	1
53	11/8/2019 16:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		1d0d2ce	######	######	NULL	iphone	10.20.0		1228	1228	34.424	112.42	0	0	25	31	NULL	NULL			NULL	6700		select	NULL		{}	6700	{"cate_1	query=te	{"adlogir	22-112:3	######	2E+07	16	2E+09	1
54	11/8/2019 16:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		4480eb	######	0	NULL	other	9.1.2		2	2	39.915	116.38	NULL	NULL	NULL	12	NULL	NULL			NULL	27763		select	NULL		{}	27763	{"cate_1	query=te	{"livefixe	22-112:3	11009	2E+07	16	2E+09	2
55	11/8/2019 16:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		416bbd5	######	0	NULL	other	9.1.2		8	8	30.575	104.08	NULL	NULL	NULL	25	NULL	NULL			NULL	258		select	NULL		{}	258	{"cate_1	query=te	{"livefixe	22-112:3	11009	2E+07	16	2E+09	2
56	11/8/2019 16:00	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		3b9b91	######	3ec78f5	2E+09	android	10.19.12		1333	1333	NULL	NULL	3	60	20	25	NULL	NULL			NULL	32705		select	NULL		{}	32705	{"cate_1	query=te	{"adlogir	22-112:3	######	2E+07	16	2E+09	1
57	11/8/2019 16:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		72cb0fa	######	######	1E+09	android	10.20.3		7	7	22.57	114.1	0	0	25	31	NULL	NULL			NULL	163		select	NULL		{}	163	{"cate_1	query=te	{"adlogir	22-112:3	######	2E+07	16	2E+09	1
58	11/8/2019 16:00	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		a9aab2	######	a055f90	1E+09	android	10.20.3		8	8	30.707	104.02	1	20	20	25	NULL	NULL			NULL	25475		select	NULL		{}	25475	{"cate_1	query=te	{"adlogir	22-112:3	######	2E+07	16	2E+09	1
59	11/8/2019 16:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		c32ee1	######	0	NULL	other	9.1.2		8	8	30.53	104.08	NULL	NULL	NULL	25	NULL	NULL			NULL	258		select	NULL		{}	258	{"cate_1	query=te	{"livefixe	22-112:3	11009	2E+07	16	2E+09	2
60	11/8/2019 16:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		f7f2b7b	######	0	NULL	other	9.1.2		8	8	30.53	103.99	NULL	NULL	NULL	25	NULL	NULL			NULL	258		select	NULL		{}	258	{"cate_1	query=te	{"livefixe	22-112:3	11009	2E+07	16	2E+09	2
61	11/8/2019 16:00	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		67d424	######	0	NULL	other	9.1.2		16	16	30.594	114.31	NULL	NULL	NULL	25	NULL	NULL			NULL	27813		select	NULL		{}	27813	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	16	2E+09	2
62	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		5636c6	######	######	9E+08	android	10.13.5		24	24	38.097	114.43	0	0	20	25	NULL	NULL			NULL	33803		select	NULL		{}	33803	{"cate_1	query=te	{"adlogir	22-126:	11009	2E+07	15	2E+09	1
63	11/8/2019 15:00	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		713e47c	######	dc2b76	2E+09	android	10.20.3		10	10	39.123	117.25	11	220	20	25	NULL	NULL			NULL	33803		select	NULL		{}	33803	{"cate_1	query=te	{"adlogir	22-126:	######	2E+07	15	2E+09	1
64	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		e1b0c63	######	0	NULL	other	9.1.2		35	35	37.881	112.51	NULL	NULL	NULL	25	NULL	NULL			NULL	27768		select	NULL		{}	27768	{"cate_1	query=te	{"livefixe	22-112:3	11009	2E+07	15	2E+09	2
65	11/8/2019 15:00	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		9a2c5c	######	0	NULL	other	9.1.2		93	93	31.71	119.99	NULL	NULL	NULL	25	NULL	NULL			NULL	27813		select	NULL		{}	27813	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	15	2E+09	2
66	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		46cfc7b	######	0	NULL	other	9.1.2		3	3	30.366	120.13	NULL	NULL	NULL	12	NULL	NULL			NULL	27763		select	NULL		{}	27763	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	15	2E+09	2
67	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		dd2373	######	0	NULL	other	9.1.2		4	4	23.222	113.28	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	15	2E+09	2
68	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		64e2e7	######	67654a	2E+09	android	10.20.3		15	79	45.782	126.67	5	100	20	18	NULL	NULL			NULL	27763		select	NULL		{}	27763	{"cate_1	query=te	{"adlogir	22-112:3	######	2E+07	15	2E+09	1
69	11/8/2019 15:00	set-jd-a	search-l	[INFO]	Pigeon-	RequestLog		60d168	######	0	NULL	other	9.1.2		4	4	23.154	113.24	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	15	2E+09	2
70	11/8/2019 15:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		9ba501f	######	0	NULL	other	9.1.2		4	4	23.199	113.32	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	15	2E+09	2
71	11/8/2019 17:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		6926ac	######	0	NULL	other	9.1.2		1	1	31.004	121.33	NULL	NULL	NULL	12	NULL	NULL			NULL	27761		select	NULL		{}	27761	{"cate_1	query=te	{"livefixe	22-112:3	11009	2E+07	17	2E+09	2
72	11/8/2019 17:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		ba6bd0	######	dd7a46	1E+08	android	10.20.3		6	6	31.273	120.53	0	0	20	25	NULL	NULL			NULL	27767		select	NULL		{}	27767	{"cate_1	query=te	{"adlogir	22-126:	11009	2E+07	17	2E+09	1
73	11/8/2019 17:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		7ee149e	######	0	NULL	other	9.1.2		1	1	31.207	121.58	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	17	2E+09	2
74	11/8/2019 17:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		eaaad5	######	0	NULL	other	9.1.2		1	1	31.274	121.54	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	17	2E+09	2
75	11/8/2019 17:00	search-l	search-l	[INFO]	Pigeon-	RequestLog		9863de	######	0	NULL	other	9.1.2		1	1	31.252	121.49	NULL	NULL	NULL	25	NULL	NULL			NULL	2784		select	NULL		{}	2784	{"cate_1	query=te	{"bizerTi	22-112:3	11009	2E+07	17	2E+09	2

Figure A-1 Raw dataset.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	servername	thread	request_ic	request_time	os	app_versi	page_city_	locate_city	lat	lng	page_inde	offset	page_size	stock	cate_id	query_typ	query_ana	search_re	search_inf	ab_trace_	slot_ids	traffic_type
2	set-jd-ad-sha	Pigeon-Se	22c37d6e	11/8/2019 15:42	other	9.1.2	1	1	31.3475	121.489	1	15	22	12	27762	select	{"cate_1":	query=ter	{"bizerTim	22-112:36	11009	0
3	search-bizer-	Pigeon-Se	64c82c86	11/8/2019 14:02	other	9.1.2	1	1	31.0865	121.416	1	15	22	12	27761	select	{"cate_1":	query=ter	{"bizerTim	22-112:36	11009	0
4	search-bizer-	Pigeon-Se	29961072	11/8/2019 5:02	android	10.20.3	8	4424	30.7715	104.12	0	0	20	18	27761	select	{"cate_1":	query=ter	{"adloginf	22-112:36	1,711,009	0
5	search-bizer-	Pigeon-Se	94a4baa6	11/8/2019 19:29	pc	none	1	47	35.7312	116.382	0	0	15	10	116	select	{"cate_1":	query=ter	{"adloginf	22-89:259	31	1
6	search-bizer-	Pigeon-Se	12ec4859	11/8/2019 16:03	other	9.1.2	3	3	30.2257	120.175	1	15	22	12	27761	select	{"cate_1":	query=ter	{"bizerTim	22-112:36	11009	0
7	search-bizer-	Pigeon-Se	3616e40d	11/8/2019 23:11	pc	none	1	47	35.7312	116.382	0	0	15	10	116	select	{"cate_1":	query=ter	{"adloginf	22-89:261	31	1
8	search-bizer-	Pigeon-Se	8631ca94	11/8/2019 16:07	pc	none	1727	47	35.7312	116.382	0	0	15	16	2876	select	{"cate_1":	query=ter	{"adloginf	22-111:36	31	1
9	search-bizer-	Pigeon-Se	23faff2d-7	11/8/2019 14:42	other	9.1.2	3	3	30.2544	120.087	1	15	22	25	27768	select	{"cate_1":	query=ter	{"bizerTim	22-112:36	11009	0
10	search-bizer-	Pigeon-Se	e73318bf-	11/8/2019 16:00	other	9.1.2	8	8	30.5754	103.994	1	15	22	25	258	select	{"cate_1":	query=ter	{"livefixed	22-112:36	11009	0
11	set-jd-ad-sha	Pigeon-Se	d7cdfed0-	11/8/2019 14:43	android	10.20.3	152	152	37.4044	122.169	0	0	20	25	33803	select	{"cate_1":	query=ter	{"adloginf	22-126:44	11009	0
12	search-bizer-	Pigeon-Se	cf6cb6c6-l	11/8/2019 2:11	pc	none	1	47	35.7312	116.382	0	0	15	10	116	select	{"cate_1":	query=ter	{"adloginf	22-89:258	31	1
13	search-bizer-	Pigeon-Se	cb7cd05a-	11/8/2019 14:02	other	9.1.2	1	1	31.1312	121.179	1	15	22	12	27761	select	{"cate_1":	query=ter	{"bizerTim	22-112:36	11009	0
14	search-bizer-	Pigeon-Se	9863de90	11/8/2019 17:00	other	9.1.2	1	1	31.2516	121.492	1	15	22	25	2784	select	{"cate_1":	query=ter	{"bizerTim	22-112:36	11009	0
15	set-jd-ad-sha	Pigeon-Se	58c58eca-	11/8/2019 14:02	other	9.1.2	1	1	31.1314	121.416	1	15	22	12	27761	select	{"cate_1":	query=ter	{"bizerTim	22-112:36	11009	0
16	search-bizer-	Pigeon-Se	5d1ca988	11/8/2019 17:34	android	10.20.3	267	267	31.6107	116.01	0	0	20	25	125	select	{"cate_1":	query=ter	{"adloginf	22-126:44	1,711,009	0
17	set-jd-ad-sha	Pigeon-Se	1e02a445	11/8/2019 21:08	pc	none	1	47	35.7312	116.382	0	0	14	10	110	select	{"cate_1":	query=ter	{"adloginf	22-86:247	31	1
18	set-jd-ad-sha	Pigeon-Se	fa5b4af5-(	11/8/2019 16:03	iphone	10.20.0	101	101	28.0098	120.696	0	0	20	25	33803	select	{"cate_1":	query=ter	{"adloginf	22-110:36	11009	0
19	search-bizer-	Pigeon-Se	6e9d22be	11/8/2019 14:03	other	9.1.2	2	2	39.8272	116.396	1	15	22	12	27763	select	{"cate_1":	query=ter	{"livefixed	22-112:36	11009	0
20	search-bizer-	Pigeon-Se	64e2e7ab	11/8/2019 15:00	android	10.20.3	15	79	45.7824	126.67	5	100	20	18	27763	select	{"cate_1":	query=ter	{"adloginf	22-112:36	1,711,009	0
21	set-jd-ad-sha	Pigeon-Se	1b8236cd	11/8/2019 14:23	iphone	10.20.0	70	70	43.8343	125.25	0	0	20	25	193	select	{"cate_1":	query=ter	{"adloginf	22-112:36	1,711,009	0
22	search-bizer-	Pigeon-Se	37c439e1	11/8/2019 14:02	pc	none	1	47	35.7312	116.382	0	0	15	17	80	select	{"cate_1":	query=ter	{"adloginf	22-125:43	31	1
23	search-bizer-	Pigeon-Se	294d11ef-	11/8/2019 7:38	pc	none	2	47	35.7312	116.382	0	0	15	10	4025	search	{"cate_1":	query=ter	{"adloginf	21-76:207	5	1
24	set-jd-ad-sha	Pigeon-Se	dbcca032-	11/8/2019 18:02	pc	none	1	47	35.7312	116.382	0	0	11	17	158	select	{"cate_1":	query=ter	{"adloginf	22-111:36	31	1
25	set-jd-ad-sha	Pigeon-Se	3456c70e	11/8/2019 15:00	other	9.1.2	2	2	40.0107	116.521	1	15	22	12	27763	select	{"cate_1":	query=ter	{"livefixed	22-112:36	11009	0
26	set-xh-ad-sha	Pigeon-Se	b2613633	11/8/2019 14:02	other	9.1.2	1	1	31.2295	121.407	1	15	22	12	27761	select	{"cate_1":	query=ter	{"bizerTim	22-112:36	11009	0
27	set-jd-ad-sha	Pigeon-Se	7e0e191e	11/8/2019 19:38	pc	none	1	47	35.7312	116.382	0	0	15	10	116	select	{"cate_1":	query=ter	{"adloginf	22-86:247	31	1
28	search-bizer-	Pigeon-Se	5f0b7f6d-	11/8/2019 18:52	pc	none	145	47	35.7312	116.382	0	0	15	17	4025	search	{"cate_1":	query=ter	{"adloginf	21-78:210	5	1
29	search-bizer-	Pigeon-Se	e6181153	11/8/2019 16:07	pc	none	344	47	35.7312	116.382	0	0	15	17	4025	search	{"cate_1":	query=ter	{"adloginf	21-76:206	5	1
30	search-bizer-	Pigeon-Se	f39bf216-	11/8/2019 15:40	pc	none	1	47	35.7312	116.382	0	0	13	16	4025	search	{"cate_1":	query=ter	{"adloginf	21-78:213	5	1
31	set-jd-ad-sha	Pigeon-Se	99d78dc6	11/8/2019 17:28	other	9.1.2	10	10	39.0848	117.134	1	15	22	25	2784	select	{"cate_1":	query=ter	{"livefixed	22-112:36	11009	0
32	set-xh-ad-sha	Pigeon-Se	da961821	11/8/2019 14:22	other	10.2.10	1	1	31.6107	116.01	1	15	22	-1	167	select	{"cate_1":	query=ter	{"adloginf	22-112:36	1,711,009	0
33	search-bizer-	Pigeon-Se	74fa78bf-	11/8/2019 19:17	pc	none	1675	47	35.7312	116.382	0	0	15	17	4025	search	{"cate_1":	query=ter	{"adloginf	21-76:207	5	1
34	set-jd-ad-sha	Pigeon-Se	ffd27db1-	11/8/2019 10:02	iphone	10.20.0	1	1	31.3288	121.578	1	20	20	25	33803	select	{"cate_1":	query=ter	{"adloginf	22-112:36	11009	0
35	set-jd-ad-sha	Pigeon-Se	a57bc323	11/8/2019 14:00	android	10.20.3	210	210	21.2818	110.361	0	0	25	31	163	select	{"cate_1":	query=ter	{"adloginf	22-112:36	1,711,009	0
36	search-bizer-	Pigeon-Se	855f7699	11/8/2019 19:59	pc	none	1	47	35.7312	116.382	0	0	15	10	116	select	{"cate_1":	query=ter	{"adloginf	22-89:256	31	1
37	search-bizer-	Pigeon-Se	240c7950	11/8/2019 16:42	other	9.1.2	17	17	34.3699	108.964	1	15	22	25	27768	select	{"cate_1":	query=ter	{"bizerTim	22-112:36	11009	0
38	search-bizer-	Pigeon-Se	df6f36ee-f	11/8/2019 16:03	other	9.1.2	3	3	30.2111	120.125	1	15	22	25	27813	select	{"cate_1":	query=ter	{"bizerTim	22-112:36	11009	0
39	search-bizer-	Pigeon-Se	f7f2b7b5-	11/8/2019 16:00	other	9.1.2	8	8	30.5305	103.994	1	15	22	25	258	select	{"cate_1":	query=ter	{"livefixed	22-112:36	11009	0
40	search-bizer-	Pigeon-Se	095d7430	11/8/2019 15:01	pc	none	219	47	35.7312	116.382	1	15	15	17	4025	search	{"cate_1":	query=ter	{"adloginf	21-78:213	5	1
41	set-jd-ad-sha	Pigeon-Se	bfcdec00-l	11/8/2019 14:17	pc	none	2	47	35.7312	116.382	0	0	15	10	4025	search	{"cate_1":	query=ter	{"adloginf	21-76:207	5	1
42	search-bizer-	Pigeon-Se	aa3b0660	11/8/2019 19:43	pc	none	17	47	35.7312	116.382	0	0	15	10	4025	search	{"cate_1":	query=ter	{"adloginf	21-78:210	5	1

Figure A-2 Dataset after pre-processing

Appendix B Code of Experiments

B.1 Experiments with Keras Sequential model

```
# Import libraries
!pip install sklearn
from __future__ import absolute_import, division, print_function, unicode_literals


import numpy as np
import pandas as pd

try:
  %tensorflow_version 2.x
except Exception:
  pass
import tensorflow as tf

from tensorflow import feature_column
from tensorflow.keras import layers
from keras.layers import Input, Dense, Dropout
from keras.models import Model
from sklearn.model_selection import train_test_split


# Initialize Tensorboard
import datetime
%load_ext tensorboard
from tensorboard import notebook
```

```
# Create a dataframe
dataframe = pd.read_csv("/Dataset.csv", na_filter=False, header=0)
dataframe.fillna('', inplace=True)


# Split the dataframe into train, validation, and test
train, test = train_test_split(dataframe, test_size=0.2)
train, val =train_test_split(train, test_size=0.2)


# Create an input pipeline using tf.data
def df_to_dataset(dataframe, shuffle=True, batch_size=20):
  dataframe = dataframe.copy()
  labels = dataframe.pop('traffic_type')
  ds = tf.data.Dataset.from_tensor_slices((dict(dataframe), labels))
  if shuffle:
    ds = ds.shuffle(buffer_size=len(dataframe))
  ds = ds.batch(batch_size)
  return ds


batch_size = 5 # A small batch sized is used for demonstration purposes
train_ds = df_to_dataset(train, batch_size=batch_size)
val_ds = df_to_dataset(val, shuffle=False, batch_size=batch_size)
test_ds = df_to_dataset(test, shuffle=False, batch_size=batch_size)
```

```
# Read the input pipeline
for feature_batch, label_batch in train_ds.take(1):
  print('Every feature:', list(feature_batch.keys()))
  print('A batch of traffic_type',label_batch)


# Demonstrate several types of feature columns
print(dataframe.dtypes)


# We will use this batch to demonstrate several types of feature columns
example_batch = next(iter(train_ds))[0]


# A utility method to create a feature column and to transform a batch of data
def demo(feature_column):
  feature_layer = layers.DenseFeatures(feature_column)
  print(feature_layer(example_batch).numpy())


# Hashed feature columns
servername_hashed = feature_column.categorical_column_with_hash_bucket('servername',
hash_bucket_size=1000)
demo(feature_column.indicator_column(servername_hashed))

thread_hashed = feature_column.categorical_column_with_hash_bucket('thread', hash_bucket_size=1000)
demo(feature_column.indicator_column(thread_hashed))
```

```
request_id_hashed = feature_column.categorical_column_with_hash_bucket('request_id',
hash_bucket_size=1000)
demo(feature_column.indicator_column(request_id_hashed))

request_time_hashed = feature_column.categorical_column_with_hash_bucket('request_time',
hash_bucket_size=1000)
demo(feature_column.indicator_column(request_time_hashed))

query_analysis_hashed = feature_column.categorical_column_with_hash_bucket('query_analysis',
hash_bucket_size=1000)
demo(feature_column.indicator_column(query_analysis_hashed))

search_request_hashed = feature_column.categorical_column_with_hash_bucket('search_request',
hash_bucket_size=1000)
demo(feature_column.indicator_column(search_request_hashed))

search_info_hashed = feature_column.categorical_column_with_hash_bucket('search_info',
hash_bucket_size=1000)
demo(feature_column.indicator_column(search_info_hashed))

ab_trace_tag_hashed = feature_column.categorical_column_with_hash_bucket('ab_trace_tag',
hash_bucket_size=1000)
demo(feature_column.indicator_column(ab_trace_tag_hashed))

slot_ids_hashed = feature_column.categorical_column_with_hash_bucket('slot_ids', hash_bucket_size=1000)
demo(feature_column.indicator_column(slot_ids_hashed))

app_version_hashed = feature_column.categorical_column_with_hash_bucket('app_version',
hash_bucket_size=1000)
```

```
demo(feature_column.indicator_column(app_version_hashed))


# Categorical columns
os = feature_column.categorical_column_with_vocabulary_list('os', ['pc', 'android', 'iphone', 'other'])
os_one_hot = feature_column.indicator_column(os)
demo(os_one_hot)

query_type = feature_column.categorical_column_with_vocabulary_list('query_type', ['select', 'search'])
query_type_one_hot = feature_column.indicator_column(query_type)
demo(query_type_one_hot)


# Numeric cols
Num_cols = ["locate_city_id","page_city_id", "lat", "lng", "page_index", "offset", "page_size", "cate_id",
"stock"]
for i in Num_cols:
  demo(feature_column.numeric_column(i))


feature_columns = []
# Hashed feature columns
feature_columns.append(feature_column.indicator_column(servername_hashed))
feature_columns.append(feature_column.indicator_column(thread_hashed))
feature_columns.append(feature_column.indicator_column(request_id_hashed))
feature_columns.append(feature_column.indicator_column(request_time_hashed))
feature_columns.append(feature_column.indicator_column(query_analysis_hashed))
feature_columns.append(feature_column.indicator_column(search_request_hashed))
feature_columns.append(feature_column.indicator_column(search_info_hashed))
feature_columns.append(feature_column.indicator_column(ab_trace_tag_hashed))
feature_columns.append(feature_column.indicator_column(slot_ids_hashed))
feature_columns.append(feature_column.indicator_column(app_version_hashed))
# Categorical columns
feature_columns.append(os_one_hot)
feature_columns.append(query_type_one_hot)
# Numeric cols
# feature_columns.append(feature_column.numeric_column('page_city_id'))
# feature_columns.append(feature_column.numeric_column('locate_city_id'))
for i in Num_cols:
  feature_columns.append(feature_column.numeric_column(i))


# Create a feature layer
feature_layer = tf.keras.layers.DenseFeatures(feature_columns)
batch_size = 32
train_ds = df_to_dataset(train, batch_size=batch_size)
val_ds = df_to_dataset(val, shuffle=False, batch_size=batch_size)
test_ds = df_to_dataset(test, shuffle=False, batch_size=batch_size)


# clear existing log filed
!rm -rf ./logs/


# train the model
model = tf.keras.Sequential([
  feature_layer,
  layers.Dense(20, activation='linear'),
  layers.Dropout(0.6),
  layers.Dense(20,activation='linear'),
  layers.Dropout(0.6),
  layers.Dense(1, activation='linear')
])


model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])

log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

model.fit(train_ds,
          validation_data=val_ds,
          epochs=40,
          callbacks=[tensorboard_callback])
print('\n Test')
result = model.evaluate(val_ds)
dict(zip(model.metrics_names, result))


# Test the model
model.evaluate(test_ds)
%tensorboard --logdir logs
```

B.2 Experiments with Random Forest and SVM

```
import numpy as np
import pandas as pd

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,  BaggingClassifier,
ExtraTreesClassifier

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier, plot_importance

from sklearn.model_selection import train_test_split, KFold, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

df = pd.read_csv('/Dataset.csv')

for i in list(df.head(0)):
  df[i] = pd.factorize(df[i])[0].astype(np.uint64)

df.head(100)

df.columns = ['servername', 'thread', 'request_id', 'request_time', 'os',
       'app_version', 'page_city_id', 'locate_city_id', 'lat', 'lng',
       'page_index', 'offset', 'page_size', 'stock', 'cate_id', 'query_type',
       'query_analysis', 'search_request', 'search_info', 'ab_trace_tag',
       'slot_ids', 'traffic_type']

print('traffic_type: ', np.unique(df['traffic_type']))

label = np.unique(df['traffic_type'])
print(label)

X, y = df.iloc[:,:-1].values,df.iloc[:,-1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

features = df.columns[:-1]

"""# Performance of models"""

clf_rf = RandomForestClassifier()
clf_et = ExtraTreesClassifier()
clf_bc = BaggingClassifier()
clf_ada = AdaBoostClassifier()
clf_dt = DecisionTreeClassifier()
clf_xg = XGBClassifier()
clf_lr = LogisticRegression()
clf_svm = SVC()

Classifiers =
['RandomForest','ExtraTrees','Bagging','AdaBoost','DecisionTree','XGBoost','LogisticRegression','SVM']
scores = []
models = [clf_rf, clf_et, clf_bc, clf_ada, clf_dt, clf_xg, clf_lr, clf_svm]
for model in models:
  score = cross_val_score(model, X_train, y_train, scoring = 'accuracy', cv = 10, n_jobs = -1).mean()
  scores.append(score)

mode = pd.DataFrame(scores, index = Classifiers, columns = ['score']).sort_values(by = 'score', ascending
= False)

mode

"""# Training of RandomRorest, SVM (kernel = linear), and SVM (kernel = sigmoid)"""

rf_10000 = RandomForestClassifier(n_estimators=10000)

rf_10000.fit(X_train,y_train)

importances = rf_10000.feature_importances_
indices = np.argsort(importances)[::-1]

for f in range(X_train.shape[1]):
  # Evaluate feature importance based on calculation of average impure decay of 10,000 decision trees
  print ("%2d) %-*s %f" % (f+1,30, features[f], importances[indices[f]]))

# Visualize Feature Importance
plt.title('Feature Importance of RandomForest')
plt.bar(range(X_train.shape[1]), importances[indices], color='gray', align='center')
plt.xticks(range(X_train.shape[1]), features, rotation=90)
plt.xlim([-1, X_train.shape[1]])
plt.tight_layout()
plt.show()

"""# Decision Boundary Visualization of RandomForest and SVM"""
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from matplotlib.colors import ListedColormap
from mlxtend.plotting import plot_decision_regions

x = StandardScaler().fit_transform(X)
X_train_reduced = PCA(n_components = 2).fit_transform(X_train)
X_test_reduced  = PCA(n_components=  2).fit_transform(X_test)

t = np.array(y_train)
t = t.astype(np.integer)

rf_1 = RandomForestClassifier(n_estimators=1)
rf_1.fit(X_train_reduced,t)
plt.figure(figsize = [15,10])
plot_decision_regions(X_train_reduced, t, clf = rf_1, hide_spines = False, colors = 'white,gray')
plt.legend(loc='upper left')
plt.title("Randomforest (n_estimators=1)")

rf_10 = RandomForestClassifier(n_estimators=10)
rf_10.fit(X_train_reduced,t)
plt.figure(figsize = [15,10])
plot_decision_regions(X_train_reduced, t, clf = rf_10, hide_spines = False, colors = 'white,gray')
plt.legend(loc='upper left')
plt.title("Randomforest (n_estimators=10)")

rf_100 = RandomForestClassifier(n_estimators=100)
rf_100.fit(X_train_reduced,t)
```

```
plt.figure(figsize = [15,10])
plot_decision_regions(X_train_reduced, t, clf = rf_100, hide_spines = False, colors = 'white,gray')
plt.legend(loc='upper left')
plt.title("Randomforest (n_estimators=100)")

rf_1000 = RandomForestClassifier(n_estimators=1000)
rf_1000.fit(X_train_reduced,t)
plt.figure(figsize = [15,10])
plot_decision_regions(X_train_reduced, t, clf = rf_1000, hide_spines = False, colors = 'white,gray')
plt.legend(loc='upper left')
plt.title("Randomforest (n_estimators=1000)")

rf_10000.fit(X_train_reduced,t)
plt.figure(figsize = [15,10])
plot_decision_regions(X_train_reduced, t, clf = rf_10000, hide_spines = False, colors = 'white,gray')
plt.legend(loc='upper left')
plt.title("Randomforest (n_estimators=10000)")

linear_svm = SVC(kernel='linear')
linear_svm.fit(X_train_reduced,t)
plt.figure(figsize = [15,10])
plot_decision_regions(X_train_reduced, t, clf = linear_svm, hide_spines = False, colors = 'white,gray')
plt.legend(loc='upper left')
plt.title("Support Vector Machines (kernel='linear')")

sigmoid_svm = SVC(kernel='sigmoid')
sigmoid_svm.fit(X_train_reduced,t)
plt.figure(figsize = [15,10])
plot_decision_regions(X_train_reduced, t, clf = sigmoid_svm, hide_spines = False, colors = 'white,gray')
```

```
plt.legend(loc='upper left')
plt.title("Support Vector Machines (kernel='sigmoid')")

"""# Predictions and Evaluations"""

def evaluate(classifier_name, predictions, y_test):
  TP, TN, FP, FN = 0, 0, 0, 0
  for i in range(len(predictions)):
    if predictions[i] == 1 and y_test[i] == 1:
      TP += 1
    elif predictions[i] == 0 and y_test[i] == 0:
      TN += 1
    elif predictions[i] == 1 and y_test[i] == 0:
      FP += 1
    else:
      FN += 1
  print(
      """
                   Confusion Matrix of {}
        Original type  human robot
  Classified as human   {}    {}
  Classified as robot   {}    {}
      """.format(classifier_name, TN, FN, FP, TP)
  )
  print('Precision: ', TP/(TP + FP), 'recall: ', TP / (TP + FN))

rf_1 = RandomForestClassifier(n_estimators=1)
rf_1.fit(X_train,y_train)
rf_1_predictions = rf_1.predict(X_test)
```

```
evaluate('Randomforest (n_estimators=1)',rf_1_predictions,y_test)

rf_10 = RandomForestClassifier(n_estimators=10)
rf_10.fit(X_train,y_train)
rf_10_predictions = rf_10.predict(X_test)
evaluate('Randomforest (n_estimators=10)',rf_10_predictions,y_test)

rf_100 = RandomForestClassifier(n_estimators=100)
rf_100.fit(X_train,y_train)
rf_100_predictions = rf_100.predict(X_test)
evaluate('Randomforest (n_estimators=100)',rf_100_predictions,y_test)

rf_1000 = RandomForestClassifier(n_estimators=1000)
rf_1000.fit(X_train,y_train)
rf_1000_predictions = rf_1000.predict(X_test)
evaluate('Randomforest (n_estimators=1000)',rf_1000_predictions,y_test)

rf_10000 = RandomForestClassifier(n_estimators=10000)
rf_10000.fit(X_train,y_train)
rf_10000_predictions = rf_10000.predict(X_test)
evaluate('Randomforest (n_estimators=10000)',rf_10000_predictions,y_test)

linear_svm.fit(X_train, y_train)
linear_svm_predictions = linear_svm.predict(X_test)
evaluate('Linear_SVM',linear_svm_predictions,y_test)

sigmoid_svm.fit(X_train, y_train)
sigmoid_svm_predictions = sigmoid_svm.predict(X_test)
evaluate('Sigmoid_SVM',sigmoid_svm_predictions, y_test)
```