2015-03-10

Can Passive Mobile Application Traffic be Identified using Machine Learning Techniques

Peter Holland
Technological University Dublin

Can passive mobile application traffic be identified using machine learning techniques?

Peter Holland

D99991048

A dissertation submitted in partial fulfilment of the requirements of
Dublin Institute of Technology for the degree of
M.Sc. in Computing (Data Analytics)

March 2015

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Knowledge Management), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: _____

Date: ***6th March 2015***

ABSTRACT

Mobile phone applications (apps) can generate background traffic when the end-user is not actively using the app. If this background traffic could be accurately identified, network operators could de-prioritise this traffic and free up network bandwidth for priority network traffic. The background app traffic should have IP packet features that could be utilised by a machine learning algorithm to identify app-generated (passive) traffic as opposed to user-generated (active) traffic.

Previous research in the area of IP traffic classification focused on classifying high level network traffic types originating on a PC device. This research was concerned with classifying low level app traffic originating on mobile phone device.

An innovative experiment setup was designed in order to answer the research question. A mobile phone running Android OS was configured to capture app network data. Three specific data trace procedures where then designed to comprehensively capture sample active and passive app traffic data. Feature generation in previous research recommend computing new features based on IP packet data. This research proposes a different approach. Feature generation was enabled by exposing inherent IP packet attributes as opposed to computing new features. Specific evaluation metrics were also designed in order to quantify the accuracy of the machine learning models at classifying active and passive app traffic.

Three decision tree models were implemented; C5.0, C&R tree and CHAID tree. Each model was built using a standard implementation and with boosting. The findings indicate that passive app network traffic can be classified with an accuracy up to 84.8% using a CHAID decision tree algorithm with model boosting enabled. The finding also suggested that features derived from the inherent IP packet attributes, such as time frame delta and bytes in flight, had significant predictive value.

Key words: *Internet Traffic classification, IP Traffic Classification, Internet Traffic Categorisation, Internet protocol, Machine Learning.*

ACKNOWLEDGEMENTS

I would like to thank my dissertation supervisor, Luca Longo, for his expert advice throughout the dissertation process. I would also like to thank the members of the IBM Now Factory team that helped me to define the research proposal and patiently answered numerous questions throughout the last 4 months.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

1.1 Overview of Project Area

Mobile phone applications (apps) can generate background traffic when the end-user is not using the application. Even if an app has not been opened by the end-user, the app could still generate traffic on the network. This background network traffic should have Internet Protocol (IP) packet statistical features that will make it identifiable as app-generated as opposed to user-generated traffic.

The main reasons mobile apps create passive network traffic is to have content ready for the end-user when the app is opened, such as syncing emails or loading profile feeds from social network apps. "Pre-caching" is a well-established technique where mobile apps and web browser software uses machine learning techniques to guess what content an end-user will click on next and pre-cache the content (Klein and Chung, 2006). Pre-caching is implemented to improve the user experience.

By examine IP packets captured from a mobile device, it should be possible to derive distinct statistical packet features that can be used as input to a machine learning model. The machine learning model could then be used to correctly identify active versus passive traffic.

It is important for network operators to know what type of traffic is flowing through their network. IP traffic categorisation underpins a number of important network management tasks, such as: 1) Understanding the traffic load on the network 2) Automated intrusion detection such as Denial of Service (Dos) attack 3) Reallocation of network resources such as traffic shaping 4) Quality of Service (QoS) management. Prioritising traffic for high value customers or for particular services 6) Identify customer use of the network resource that in some way contravenes the operators terms of service 7) Legal obligation for lawful interception of IP traffic for persons of interest to law enforcement agencies 8) Evolution of the type of traffic on a network is important for long term capacity planning on the network

1.2 Background

Early research in IP traffic classification focused on port number based approaches. In port number based packet inspection, port numbers are captured from the IP packet headers. A packet is classified based on a lookup of the port number against IANA reserved ports. The classification accuracy for port-based approaches is reported to be between 50% and 70%.

Researchers then proposed analysis of IP packet payloads. This technique involves inspecting the payload of each IP packet for features that can be used to classify the traffic.

Some research suggests using features derived from the full TCP flows. TCP is a connection-orientated protocol. The TCP protocol sets up a connection between source and destination points. All packets with the same source address/port and destination address/port within a time period, or until the connection is terminated, are considered as one flow. The approach must wait until a flow completes or times-out before generating features about the statistical characteristics of the flow and packets in the flow. Examples of flow features include; average flow duration and average packet size per flow.

A number of papers have investigated IP traffic classification based on packet header statistical features. The majority of packet based analysis take the position that any practical IP traffic classification system must be capable of running in real-time on a live network. To achieve real-time traffic classification a Machine Learning (ML) system must be lean, for example, the system must meet the following requirements; use a small feature set, have a fast model training time, have a fast classification time, low memory requirements and low processing requirements.

The idea of using ML techniques for IP traffic classification was first introduced in the context of intrusion detection. A machine learning algorithm automatically builds a classifier by learning the inherent structure of a dataset based on the characteristic features. ML techniques for real-time and offline analysis have demonstrated high classification accuracy of up to 99% for a various types of Internet applications traffic.

1.2.1 Limitations of Current Approaches

There are a number of limitations with the extent of the current body of knowledge in network traffic categorisation.

The research has primarily focused on personal computer based traffic. There is only a small body of knowledge, starting in the last 2 years, concerning mobile generating traffic.

Previous network traffic categorisation research is concerned with identifying types of network traffic at a very high level. For example, TCP, FTP, Telnet, Web. These traffic types are too broad. There is a lack of research on specific traffic from individual services such as Google maps or Apple iTunes. Specific service information is more valuable to network operators.

Previous research has also focussed on payload based inspection. However, payload inspection is not practical due to large processing overheads of inspecting all the data in each IP packet. Also, the increasing amount of encrypted traffic make this task impossible. Data protection laws may also be a barrier to this type of analysis due to the potentially sensitive personal information contained in payload data.

Finally, existing research does not appear to utilise the massive amount of IP packet features available. IP packets have over 100,000 inherent features available that can be exposed using IP packet analysis software. Some of these features may not be distinctive or may be unpopulated. However, IP packet features require no processing to generate and may be very valuable to a machine learning algorithm.

1.2.2 Research Gap in Current Knowledge Base

Based on a review of the body of knowledge on network traffic categorisation, a number of research gaps have been identified:

- To the best of this author's knowledge, there is no research into classifying user generated (active) versus app generated (passive) traffic.
  This project will investigate this previously unexamined research area.

- The vast majority of research is based on a fixed network IP traffic generated by personal computers.
  This research is based on mobile device originating network traffic which is an important area of research
- Previous research does not take advantage of the large amount of inherent IP packet features. Instead, previous research had added extra steps, complexity and processing calculating new features.
  This project will leverage the large amount of inherent IP packets features to allow a machine learning algorithm to successful identify user generated or app generated traffic

1.3 Research Project

This project will conduct empirical research on network traffic categorisation based on mobile phone app data. Specifically, this project will build a supervised machine learning model to distinguish app traffic that is generated by an end-user actively using a mobile phone application, as opposed to traffic that is generated by the application in the background without any end-user initiation.

By examine IP packets captured from an Android device, this project will derive important statistical packet features and then build a classification model to identify user-generated versus app-generated traffic. Because of the very high number of features that can be generated, feature selection and reduction will be important parts of this research.

Being able to distinguish user-generated, also known as active traffic, versus the app-generated traffic, also known as passive traffic, would have two valuable uses for mobile phone network operators:

1. Network operators could optimise their networks and improve the customer experience by prioritising known user-generated traffic. Conversely, the app-generate or background traffic could be deprioritised because the end-user is unaware of this traffic and has no experience of it.
2. Network operators need to be able to accurately count the number of active users of an app. Many apps come pre-loaded on mobile devices and can

generate background traffic. Without knowing how to identify this traffic a network operator may significantly over count the number of active app users.

Based on the gaps identified in the current body of knowledge on network traffic categorisation, the research question is stated as follows:

"Can passive mobile app traffic be identified using machine learning techniques?"

1.4 Research Objectives

Guided by the research question, the project objectives are:

1. Gain knowledge in the research domain of network traffic classification
2. Design research question experiment solution
3. Implement the experiment solution and capture results
4. Evaluate outcomes from experiment implementation

1.5 Research Methodology

This research will use quantitative research methods based on the numerical analysis of network traffic data collected from a mobile phone. The data will be based on network data traces. The first data trace will collect non-user generated traffic. The second data trace will collect user generated traffic. The quantitative research designs will be descriptive and will aim to establish the associations between variables.

This research will be based on empirical research methods. Empirical data will be produced by experiment and conclusions will be based on evaluation of the experimental data.

The project research methodologies to achieve the project objections are:

1. Complete a literature review of knowledge base
2. Methodical experiment design that is practical and implementable
3. Quantitative research methodology
    a. Capture IP data from mobile phone
    b. Generate features
    c. Select and reduce features
    d. Build machine learning classification models

4. Critical analysis of results
    a. Evaluate accuracy of built models
    b. Critically analyse results and report conclusion and future work

1.6 Scope and Limitations

This project will have the following scope limitations:

An Android OS device is used, specifically a HTC One S running Android 4.1.1. An Android OS device was chosen because special software needs to be installed on the mobile device to capture network traffic. It is much easier to install this software on an Android device compart to and Apple device running IOS.

A limited number of mobile apps are considered. 11 specific apps were chosen. This limitation was introduced for two reasons. Firstly, the 11 apps chosen are the most commonly downloaded and used apps in the world. Hench using these apps will cover most traffic scene on a network. The second reason for limitation the number of apps considered is to control the scope of the research.

Each data trace will cover a 30 minute period. This is to manage the size of the test data. Packet capture (pcap) files can be very large, for example, the 30 minute data traces are expected to create approximately 200,000 records.

This project will not cover any IP packet payload inspection. This project will look to find distinctive packet features rather than directly inspect payload inspection. Also, payload inspection is complex and in the case of encrypted content, it is not possible to inspect packet payloads.

This project will not reconstruct of end-to-end TCP flows. This project will look at low level packet detail not higher level flow detail.

Real time processing is not in scope because testing a real time deployment would be technically very difficult. However, real-time classification will be considerations will

be important evaluation metrics. Such as performance in terms of the trade-off between the model accuracy and processing overhead.

1.7 Document Outline

The remainder of this document is organised as follows:

Chapter 2 reviews the literature relating to network traffic categorisation. The evolution of network traffic categorisation research is outlined. The limitations in the research area are identified and discussed. Other important considerations are also discussed such as the important network issues that relate to this paper.

Chapter 3 outlines the experiment design. This chapter outlines the step by step experiment process from running data traces on a mobile device, exposing packet features, the ML techniques to be used and finally, the evaluation criteria for the ML model.

Chapter 4 documents how the experiment design was implemented. Details of the data trace process is clearly presented. Analysis of the data packets and feature generation are also presented. Details of the ML modelling training set up are also discussed.

Chapter 5 reports on the findings from the empirical study, as implemented in Chapter 4. The machine learning models are evaluated against the evaluation criteria. Weaknesses and limitations are also discussed.

Chapter 6 will providing a clear summary of thesis and contribution to the body of knowledge. Future work and recommendations are presented to highlight how the project can be clearly extended and enhanced

2 LITERATURE REVIEW

2.1 Introduction

This chapter surveys and reviews the literature relating to the network traffic classification. A history of different approaches to the problem is presented. Different classification techniques are discussed and assessed. Finally, performance measures and evaluation criteria used in network traffic classification research are grouped and explored.

2.1.1 Evolution of Approaches for IP Traffic Classification

This section will highlight the important evolution of the approaches to IP traffic classification, including the recommended best-in-class techniques based on current knowledge.

2.1.2 Port Number Based Analysis

Early research in IP traffic classification focused on port number based approaches. In port number based packet inspection, port numbers are captured from the IP packet headers. IANA (Internet Assigned Numbers Authority) recommend reserve specific port numbers for specific application e.g. port number 80 is reserved for web based applications. A packet is classified based on a lookup of the port number against IANA reserved ports. The classification accuracy for port-based approaches is reported to be between 50% and 70% (Moore and Papagiannaki, 2005). Port number based classification has a number of limitations:

- Port based method are deceive by a simple change of ports used by an application.
- A server port can serve multiple services. For example, a VoIP application, a chat messaging system and a web page browsing request could use the same port (Li and Moore, 2007).
- Emerging applications often avoid the use of standard ports (Moore and Papagiannaki, 2005)
- Web applications such as passive FTP or video/voice communication can use dynamic ports unknowable in advance (Zander et al., 2005).
- The proportion of network traffic that is encrypted is increasing. The port numbers may not be visible.

2.1.3 IP Packet Payload Based Analysis

Some research then proposed analysis of IP packet payloads (Moore and Papagiannaki, 2005). This technique involves inspecting the payload of each IP packet for features that can be used to classify the traffic. However, there are also major limitations with this approach;

- Payload analysis tools cannot classify encrypted packets (Bernaille et al., 2006)
- Payload-based schemes have large processing overheads and are very time-consuming as the process involves inspecting all the data in each IP packet.
- Due to the time-consuming nature of full packet payload inspection it cannot realistically be considered for real-time in high-speed links
- There are legal and privacy concerns when inspecting packets

2.1.4 Full TCP Flow Based Analysis

Some research suggests using features derived from the full TCP flows (Erman et al., 2006; Williams et al., 2006; Zander et al., 2005; Zuev and Moore, 2005). TCP is a connection-orientated protocol. The TCP protocol sets up a connection between source and destination points. TCP provides reliable, ordered and error-checked delivery of a stream of packets. All packets with the same source address/port and destination address/port within a time period, or until the connection is terminated, are considered as one flow. The approach must wait until a flow completes or times-out before generating features about the statistical characteristics of the flow and packets in the flow. Examples of flow features include; average flow duration and average packet size per flow.

The flow based analysis approach is useful for offline analysis but could never be utilised in a real network due to the below limitations;

- TCP flows can have variable time duration. IP Traffic is generally made of a large majority of flows with a short time period and a small number of flows with a very long time period (Bernaille et al., 2006). A flow must complete before it can be analysed. This can take a number of minutes.
- Flow based analysis has large processing and memory requirements in order to reconstruct flows
- Approaches relaying on summarise flow information are sensitive to simple alterations of packet size and inter-arrival times using evasion techniques

2.1.5 IP Packet Based Analysis

A number of papers have investigated IP traffic classification based on packet header statistical features (Auld et al., 2007; Bernaille et al., 2006; Kim et al., 2008; Li and Moore, 2007; Singh et al., 2013).

The majority of packet based analysis take the position that any practical IP traffic classification system must be capable of running in real-time on a live network. To achieve real-time traffic classification the ML system must be lean, for example, the system must meet the following requirements; use a small feature set, have a fast model training time, have a fast classification time, low memory requirements and low processing requirements. Singh et al. (2013) investigated near-real time classification techniques. Bernaille et al. (2006) developed a classifier that only considers the first five packets of each flow. Karagiannis et al. (2005) proposed a novel method for IP traffic classification. The authors developed a model that operates "in the dark", by this they mean the classification model has no access to packet payload, no knowledge of port numbers and no additional information other than the packets captured.
Using the statistical features of packets to generate candidate features for a ML model is the current best practice approach for building ML IP traffic classifiers. The ML algorithm should be able to classify the IP network traffic using the minimum number of features possible. This is due to the constraints of practical real-time IP traffic classification.

2.1.6 History of Machine Learning in IP Traffic Classification

The idea of using ML techniques for IP traffic classification was first introduced in the context of intrusion detection (Frank, 1994). A machine learning algorithm automatically builds a classifier by learning the inherent structure of a dataset based on the characteristic features. ML techniques for real-time and offline analysis have demonstrated high classification accuracy of up to 99% for a various types of Internet applications traffic (Nguyen and Armitage, 2008). Refer to Appendix 1 for a full summary of the classification accuracy of the various ML algorithms used in the papers reviewed in the Project Summary section above.

Numerous different ML techniques has been extensively applied to the problem of IP traffic classification. Below is a summary of the ML techniques used in the papers reviewed in the Project Summary section above.

Supervised ML techniques

- Decision Tree
  - Decision Tree (4 papers)
  - Naïve Bayes Tree
- Neural Network
  - Neural Net (2 papers)
  - Multilayer Perceptron (MLP)
  - Radial Basis Function Neural Network (RBF)
  - Bayesian trained neural network
- K-Nearest Neighbour
  - k-NN (3 papers)
- Support Vector Machine
  - SVM (2 papers)
- Various Naïve Bayes techniques
  - Naïve Bayes Algorithm (2 papers)
  - Naïve Bayes Estimator (2 papers)
  - Bayes Net Algorithm
  - Bayesian Network
  - Naïve Bayesian classifier
  - Naïve Bayes Discretisation
  - Naïve Bayes Kernel density estimation

Unsupervised ML techniques

- AutoClass (2 papers)
- K-Means
- DBSCAN

2.1.7 Feature Selection

Feature selection is highlighted as a critical step in IP traffic categorisation process, especially in real-time systems (Fahad et al., 2013; Singh et al., 2013; Williams et al., 2006; Yuan et al., 2010; Zander et al., 2005). This section will summarise the key feature selection techniques proposed in the relevant literature.

The goal of feature selection is to reduce the amount of information required to make good predictions, and to improve the error rate of classifiers. The ability to eliminate redundant features is an important ML task because it helps to identify the best

features in order to improve the classification accuracy as well as to reduce the computational complexity related to the construction of the classifier (Fahad et al., 2013). Zhang et al. (2013a) demonstrated that a Naïve Bayes classifier with feature discretization demonstrates not only significantly higher accuracy but also much faster classification speed.

(Fahad et al., 2013) analysed six well-known feature selection techniques to identify the best features for network traffic based on the following evaluation criteria: information, dependence, consistency, distance, and transformation. The six feature selection techniques are

- Information Gain (for information-based criteria),
- Gain Ratio (for information-based criteria),
- Principal Component Analysis (PCA) (for transformation- based criteria),
- Correlation-based Feature Selection (CBF) (for dependence-based criteria),
- Chisquare (for statistical criteria)
- Consistency-based Search (CBC) (for consistency-based criteria).

The Authors propose a LOA (Local Optimisation Approach) feature selection technique that combines the five well-known feature selection techniques. This combined technique can compensate for some of the limitations of the individual techniques. The experimental results also showed that LOA performs significantly better than any individual technique.

2.1.8 Feature Subset Search Techniques

Williams et al. (2006) created a feature subsets using two subset search techniques. The Best First and Greedy search methods were used in the forward and backward directions.

- Greedy search examines changes to the current feature subset through the addition or removal of features. For a given ‘parent’ feature set, all possible ‘child’ subsets are tested through either the addition or removal of features. The child subset that shows the highest improvement (goodness measure) replaces the parent subset. The process is repeated until no more improvement can be made.
- Best First search is similar to greedy search. The process creates new subsets based on the addition or removal of features. However, this technique has the ability to backtrack if the current path no longer shows improvement. A limit is placed on the number of non-improving subsets that are considered to prevent the search from backtracking through all possibilities in the feature space

2.1.9 Feature Reduction Algorithms

Williams et al. (2006) then passed the feature subset generated from the subset search process to two different algorithms, to create reduced feature sets. These algorithms evaluate different combinations of features to identify an optimal subset:

- Consistency-based feature subset search searches for the optimal feature subset, which is the smallest subset of features that can identify instances of a class as consistently as the complete feature set.
- Correlation-based feature subset search uses an evaluation heuristic. The heuristic is used to examine the usefulness of individual features along with the level of inter-correlation among the features. The goal is to find feature subsets containing attributes that are highly correlated with the class and have low inter-correlation with each other.

2.1.10 Evaluation Metrics

There are a four main of evaluation metrics proposed in the IP traffic classification literature for supervised ML algorithms:

- Accuracy**:** Overall accuracy is the percentage of the sum of all correctly classified packets/flows over the sum of all testing packets/flows. This metric is used to measure the accuracy of a classifier on all testing data
- Recall**:** recall is the ratio of correctly classified packets/flows over all ground truth data in a class
- Precision**:** precision is the ratio of correctly classified packets/flows over all predicted packets/flows in a class
- F-measure: F-Measure is used to evaluate the per-class performance. F-measure is calculated by

$$F - measure = \frac{2 \times precision \times recall}{precision + recall},$$

There are additional evaluating criteria proposed in the literature for real-time IP traffic classifiers.

- Model build time (Erman et al., 2006)
- Classification time (Singh et al., 2013)
- System Throughput: (Li and Moore, 2007) defined custom evaluation metrics for real time classifiers. System throughput is a measure of the computational complexity in calculating features.
- System Latency: (Li and Moore, 2007) also define latency as the ability to identify a flow as quickly as possible.

System Throughput and System Latency evaluation metrics are out of scope for this research. Model build time and Classification Time are in scope.

2.2 Traffic Classification Papers

This section will summarise the existing research in the area of IP traffic classification that most closely relate to this project.

Frank (1994) introduced the idea of using Machine Learning (ML) techniques for TCP flow classification in the context of intrusion detection. This research reviewed supervised ML techniques; neural network, decision tree, and unsupervised clustering techniques; k-nearest neighbour (k-NN). The research investigated using feature selection to improve the classification of network connections. The k-NN model was found to have a classification accuracy of 95%.

Karagiannis et al. (2005) defined a fundamentally different approach to classifying traffic flows by identifying patterns of host behaviour at the transport layer. This research focused on identifying the unique fingerprint of the connection between the application and the server. The research found that each internet applications/services has a unique connection fingerprint. The results showed that the research was able to classify 80%- 90% of the traffic with more than 95% accuracy by identifying connection patterns.

Moore and Papagiannaki (2005) demonstrated that using port numbers to classify internet traffic is no longer reliable. This research investigate the inaccuracies in port-based classification and identified the types of errors that may result. The research also quantifies the errors encountered. The research devises a Naïve Bayes estimator classification methodology that relies on the full packet payload inspection. The classifier has an accuracy approach 100% but proves to be a labour-intensive process due to the full packet payload inspection.

Moore and Zuev (2005) applied a supervised Naïve Bayes estimator to categorize traffic by service type. The authors used a hand classified dataset. The results indicated a 65% accuracy on per-flow classification using the simplest of Naive Bayes estimator. The research presents two refinement of Naive Bayes method that improves the overall accuracy to better than 95%. Firstly using kernel density estimation theory. Secondly using a method of feature selection and redundancy reduction, Fast Correlation-Based Filter (FCBF).

Zander et al. (2005) used the AutoClass unsupervised Bayesian classifier to learn the natural classes or clusters within network traffic. Each class represents a network traffic type. Network flows are classified based on statistical characteristics generated from packet header data. The authors used feature selection to find an optimal feature set and determine the influence of different features. The authors defined an accuracy metric termed intra-class homogeneity. The accuracy of the Auto Class classifier was found to be 86.5%.

Zuev and Moore (2005) created a hand-classified network dataset that was used as input to a supervised Bayes estimator. The classifier developed requires only the network protocol headers of unknown traffic for a successful classification. Most research looks at per flow or per packet accuracy. This research looked at per byte accuracy. The research demonstrated an accuracy of better than 66% of flows and better than 83% for packets and bytes.

Bernaille et al. (2006) used a Simple K-Means clustering algorithm to perform classification using only the first five packets of the flow. This research focused on classifying packets before the end of a TCP flow. The authors reported a 84.2% to 96.92% classification accuracy by service type. This research specifically considered real time classification of traffic in terms of memory and processing requirements

Erman et al. (2006) evaluated three unsupervised ML techniques for traffic classification. The authors compared K-Means and DBSCAN algorithms with previously used AutoClass technique. Although the authors found that the AutoClass algorithm produces the best overall accuracy at 97.6% there were positive findings from the other two clustering techniques. The DBSCAN algorithm placed the majority of the connections in a small subset of the clusters which can lead to a high predictive power of a single category of traffic. The K-Means algorithm had an overall accuracy that was only marginally lower than that of the AutoClass algorithm but may be more suitable for traffic classification due to its much faster model building time.

Williams et al. (2006) conducted a comparison of five supervised ML algorithms for practical traffic Classification, namely Naïve Bayes Discretisation, Naïve Bayes Kernel density estimation, Decision Tree C4.5, Bayesian Network and Naïve Bayes

Tree. When evaluating each algorithm, the authors specifically considered computational performance metrics such as build time and classification speed rather than classification accuracy alone. The authors concluded that classification accuracy between the algorithms is similar but computational performance differs significantly. When comparing the classification speed, the authors found that C4.5 is able to identify network flows faster than the remaining algorithms. The C4.5 algorithm had the best overall classification accuracy percentage at 94.13%, just ahead of the Bayes Net algorithm. This research also has extensive investigation into the use of feature reduction techniques to reduce the feature space.

Auld et al. (2007) designed a network traffic classifier that could achieve a high accuracy across a range of internet application types based on IP packet header-derived statistics. The ML technique used was a Bayesian trained neural network that produced a classification accuracy of up to 99%.

Li and Moore (2007) presented a ML approach to classify live network traffic. The authors created 12 features based on the packets at the start of each flow, without inspecting the packet payload, and used a C4.5 decision tree to classify the traffic. The method could identify different types of applications on live network traffic with 99.8% total accuracy. The research was not exclusively focused on classification accuracy, the latency and throughput of the classification system were investigated as highly important considerations.

Kim et al. (2008) conducted an evaluation of three ML traffic classification techniques, namely Support Vector Machine (SVM), neural network and k-NN. The feature space was based on transport layer ports, host behaviour, and flow statistical features. The results showed that SVM consistently achieved the highest classification accuracy at 99.42%.

Yuan et al. (2010) proposed a ML internet traffic classification method based on SVM. The research pays particular attention to real-time traffic classification considerations such as computation and storage requirements. This research actively tries to reduce the feature space to a small number of features that can be generated in real time from the packet headers. The SVM model achieves a classification accuracy of 99.42%.

Singh et al. (2013) focused on real time considerations of traffic classification using machine learning techniques. Five ML techniques were investigated, namely Multilayer Perceptron (MLP), Radial Basis Function Neural Network (RBF), C 4.5 Decision Tree Algorithm, Bayes Net Algorithm and Naïve Bayes Algorithm. The results showed that the Bayes Net classifier had the highest classification accuracy at 88.12%. However this technique has a long training time which does not meet the criteria for real-time traffic classification. The number of features was then reduced using Correlation based Feature Selection (FS) Algorithms, and Consistency based FS Algorithm was also tested. Using the new dataset the Bayes Net classifier gave the highest classification accuracy at 91.87% with the real time processing constraints.

Zhang et al. (2013a) investigated improving traffic classification using a limited amount of training data is available. Traffic flows were described using the discretized statistical features and flow correlation information modelled by bag-of-flow (BoF). The authors demonstrate that feature discretization can improve the Naïve Bayes model classification accuracy by approximately 5 percent when only 10 training samples are available for each traffic class. The overall classification accuracy of the Naïve Bayes classifier was 89.00%.

Zhang et al. (2013b) propose a framework based on Traffic Classification using Correlation (TCC) information. The approach is designed to address the problem of very few training samples. The research demonstrates that the TCC approach can be used on a small number of training samples to effectively improve the classification accuracy. The nearest neighbour (NN)-based method was found to have the highest classification accuracy of over 90%.

2.3 Survey Papers

Nguyen and Armitage (2008) reviewed 18 significant works that cover the period from 2004 to early 2007. The survey paper looks at emerging research into the application of ML techniques in IP traffic classification. This survey paper charts the move away from port based traffic classification techniques to the emerging techniques that use of statistical traffic characteristics. The paper also covers the more recent work on ML-based real-time IP traffic classification in operational networks.

The paper compiles a list of all the ML algorithms used in each of the papers review. The paper also contains a full list of statistical packet features generated in each paper reviewed and information about the level of classification. The paper concludes that a number of different ML algorithms such as AutoClass, Expectation Maximisation, Decision Tree, Naïve Bayes have demonstrated high classification accuracy of up to 99% for a range of Internet applications traffic.

Callado et al. (2009) explain the main techniques and problems known in the field of IP traffic classification. Packet-based and flow-based classification approaches are investigated. The advantages and problems for each approach is summarised. The paper summarises the many ML techniques used is key papers along with the evaluation of the accuracy of each technique. The paper also covers open research topics in the area of traffic classification. The paper concludes that there is no definitive best technique for IP traffic classification.

2.4 Feature Selection Paper

Feature selection, particularly reducing the feature space is an essential step in the traffic categorisation process. Fahad et al. (2013) focuses on feature selection for internet traffic classification. The paper introduces three new metrics, namely goodness, similarity and stability. These metrics can be used to compare feature selection techniques as well as to compare the quality of their outputs. The experimental results show that no existing feature selection technique performs well on all the three metrics. The paper conclude that identifying the best and most robust features, in terms of similarity, from the large feature space is critical importance for IP traffic classification. The results derived from real network traffic data shows that the Local Optimisation Approach (LOA) has the ability to identify the best features for traffic classification.

Appendix 1 contains a table that summarises the key points of the above reviewed research.

2.5 Discussion

Network traffic classification is a key element in a number of important network management tasks. Research is the area of Network traffic classification has firmly focused on network traffic generated from personal computers. Research is the area has also focused on identifying high level classes or types of data such as web service or email.

This research will focus on network traffic generated on a mobile device. The continuing proliferation of mobile devices, the increase in mobile phone network traffic and mobile devices becoming the primary internet access device means that research into mobile device traffic classification is very relevant.

Network traffic generated on mobile devices is fundamentally different to network traffic generated by a personal computer. Users on mobile devices primarily use mobile apps to access a service as opposed to a web browser on a personal computer. This research will investigate mobile app traffic generate on a mobile device.

Mobile apps present a further unique problem regarding network traffic. Mobile apps can be designed to generated background network traffic that was not initiated by a user. This type of traffic can include checking for new messages, downloading new content, sending phone location updates etc. There is no existing research into classifying user generated versus app generated network traffic.

Based on the gaps identified in the current body of knowledge on network traffic categorisation, the research question is stated as follows:

"Can passive mobile app traffic be identified using machine learning techniques?"

2.6 Conclusion

This chapter has presented a literature review of the existing knowledge base. Research gaps have been identified and the research question had been clearly stated. The research question and the literature review will serve as the foundations for the experiment design and implementation.

3 DESIGN METHODOLOGY

3.1 Introduction

This chapter outlines the experiment design, including the experimental methodology and introduces the key considerations of this experiment. This project will propose and innovative experiment design in order to capture and analyse active and passive mobile phone app network traffic. Important experiment setup decisions, including the data capture process and machine learning techniques consider are discussed and justified.

This chapter starts by describing the overall solution approach. Then the hardware and software requirements will be outlined. Next the data collection process is presented. Followed by feature creation process, feature reduction and machine learning techniques for modelling. Finally the model evaluation set up is explained. For clarity and completeness, Chapter 4 (Experiment Implementation) will follow the same heading structure as this chapter.

3.2 Solution Approach

This research project will follow a five step solution approach outlined in Figure 3-1 below.

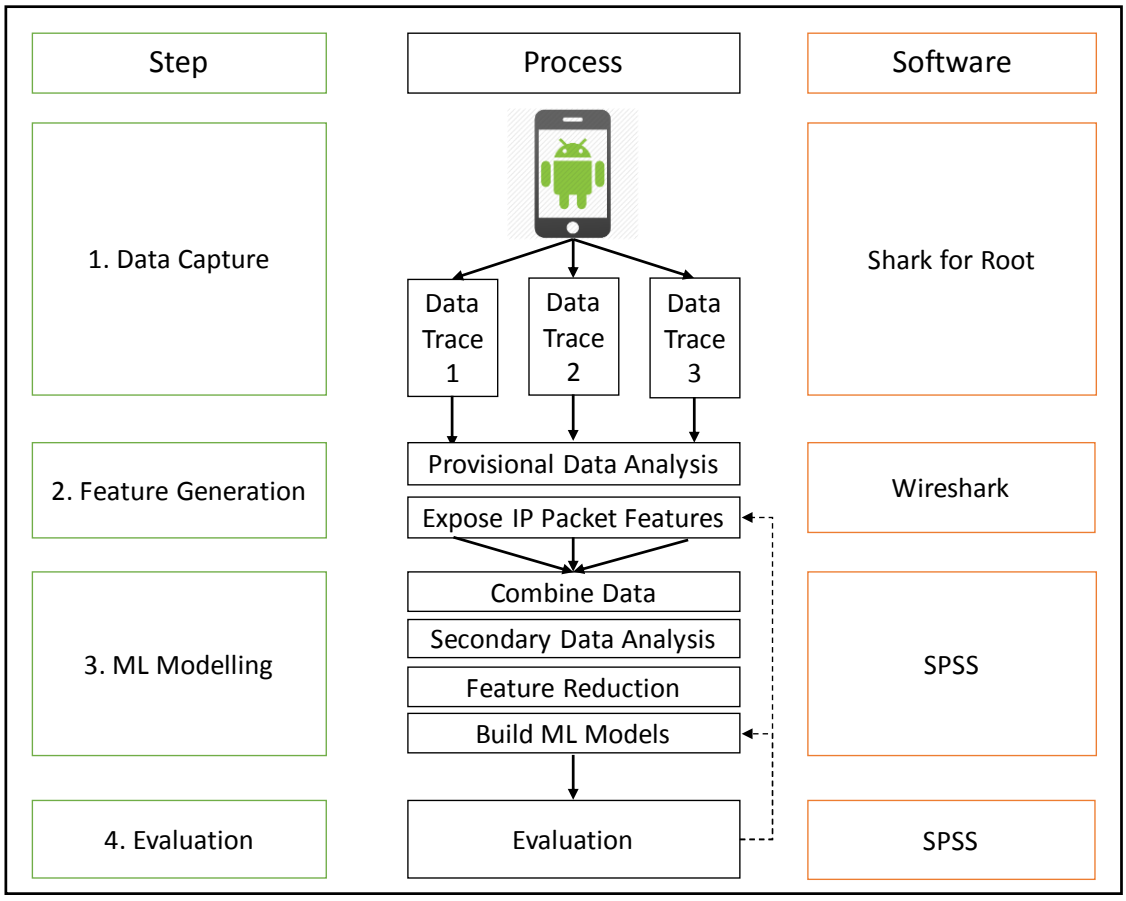


Figure 3-1 : Experiment Process Diagram

The experiment processes starts by collecting network traffic data generated by apps on an Android OS mobile phone device. The captured network traffic data will then be examined and features will be created based on the properties of the IP packets. The output files containing all the relevant IP packet features will then be passed to statistical modelling software.

Multiple machine learning techniques are applied to the data. The performance of each machine learning model will be evaluated based on the evaluation criteria. If the models are evaluated to have a poor performance, the model evaluation phase may lead back to the feature selection/generation phase or to another round of model building in order to improve model accuracy. New IP packet features can be generated and the models can be re-evaluated.

The sections below outlines the high level process for each of the steps in the experiment:

3.2.1 Step 1 - Data Capture

The data collection phase requires mobile phone hardware and software as well as personal computer based software. The full data capture process will be performed once which will result in a total of 3 data traces.

3.2.1.1 Hardware: Android device

The first step in the data collection process was to set up the mobile phone device. The collection task, also known as a data trace, was performed using on an Android device. The specific device used to capture the data traces was a HTC One S running Android OS version 4.1.1.

The mobile phone device requires IP packet capture software to be install in order to capture app network traffic data. The IP packet capture software is non-standard Android OS software and therefore cannot be installed on a standard Android OS device. In order to install the necessary packet capture software the Android device needs to be "rooted".

Rooting is the process that allows users of devices running the Android mobile operating system to attain privileged control, also known as "root access" to the Android OS sub-system. Once the Android OS device has been rooted, users can run specialized apps that require administrator-level permissions[1].

3.2.1.2 Software: Shark for Root

Once the device is rooted the IP packet capture software is installed on the mobile device. This project uses Shark for Root software available from the Google Play Android Store[2]. The Shark for Root software will be used to capture all IP packets created by the mobile apps during the data collection events outlined below.

[1] http://en.wikipedia.org/wiki/Rooting_%28Android_OS%29 Date Accessed 20/11/2014

[2] https://play.google.com/store/apps/details?id=lv.n3o.shark Date Accessed 01/10/2014

3.2.1.3 Apps: Applications considered

A specific set of eleven apps were chosen to be used during the data trace process. These apps have been chosen for two reasons, they are either the most downloaded apps from the Google Play Store or they are the most used apps on mobile networks[3][4]. By limiting the experiment to these apps, the experiment will cover the apps that take up the most network bandwidth. This increases the relevance and value of the project.

Limiting the number of apps also allows the scope of the project to be clearly defined. The limited number of apps will also help during the analysis phase when captured packets can be attributed to a specific app if required. The list of eleven apps in scope for this project are:

1. Facebook
2. Facebook Messenger
3. YouTube
4. Gmail
5. Instagram
6. Snapchat
7. Twitter
8. WhatsApp
9. Viber
10. Skype
11. Angry Birds

3.2.1.4 Process: Data Traces

The data traces will capture network IP packets generated by apps running on the mobile device. In order to get samples of active and passive app network traffic data, three separate data traces have been designed. Each trace will capture a specific type of active or passive app network data.

[3] http://www.comscore.com/Insights/Presentations-and-Whitepapers/2014/The-US-Mobile-App-Report Date Accessed 01/02/2015

[4] http://en.wikipedia.org/wiki/List_of_most_downloaded_Android_applications Date Accessed 01/02/2015

Data trace 1 will capture active app network traffic data. This data trace will capture active, user-generated traffic over a period of 30 minutes directly from the mobile device. During the data trace capture period, each of the eleven apps will be opened and specific actions will be performed on each app. The app actions will include opening the app, opening content, streaming videos, VoIP calls, sending message and receiving message.

In order to fully capture passive app network data, two separate data traces have been deigned.

Data trace 2 is designed to capture passive app network traffic. Specifically, data trace 2 is designed to capture passive app network data when the specified apps are known to be open on the phone but the apps are not actively being used by the end-user.

Data trace 2 will be undertaken following on from data trace 1 with a 5 minute gap after the end of data trace 1. The mobile phone device will then be left idle for a period of 30 minutes. No actions at all will be performed on the mobile phone device during the idle time to allow passive app network traffic to be isolated using this data trace set up.

Data trace 3 is designed to capture passive app network traffic when no apps have specifically been opened by the end-user. For this data traces, the mobile phone will be restarted. There will be a five minute wait so that phone and the apps are in steady state post start-up. The device will then be left idle for a period of 30 minutes. Similarly to data trace 2, no actions at all will be performed on the mobile phone device during the idle time to allow passive app network traffic to be isolated using this data trace set up.

It is important to distinguish between the two different situations captured in data trace 2 and data trace 3. The two different data traces will allow this project to capture the network traffic generated by an app that has been opened but is no longer in use, and an app that has never been opened but is sending background data.

3.2.1.5 Data Preparation

After the data is collected, the data in each data trace needs to be tagged in preparation for the modelling phase of the project. Each record in data trace 1 will be tagged as "active". Each record in data trace 2 and 3 will be tagged as "passive". These datasets will form the basis of the training and testing data for the machine learning algorithms. The data traces are then exported from the mobile device to a PC for analysis and prepared for the modelling phase.

3.2.2 Step 2 - Feature Generation

3.2.2.1 Software: Wireshark

Personal computer based packet analysis software is required as part of this project. Wireshark software[5] was chosen for this task. Wireshark software is a fully featured, open source network packet analysis software[6]. The system specification of the personal computer used in this research is as follows:

- OS – Windows 7 Professional Service Pack 1
- Processor – Intel Core i5 – 3320m CPU @ 2.60GHz
- RAM – 16.0GB

The Wireshark software enables two key tasks in this project:

1) Inspect and analyse the network packets. For example

- Review the data and check the data properties such as data volumes for each protocol or packet size distributions.
- Build visualisation such as input/output (IO) graphs of network data over time.
- Reconstruct TCP flows from network packet

2) Build data record features for machine learning stage

- Using Wireshark software, IP packet attributes can be exposed. By default Wireshark only shows approx. 20 general IP packet attributes such as IP address and port numbers. Other packet attributes need to be specifically added to the Wireshark view using specific commands.

[5] www.wireshark.org Date Accessed 12/02/2015

[6] http://en.wikipedia.org/wiki/Wireshark Date Accessed 12/02/2015

- For a full list of available packet name attributes please see the Wireshark Display Filter Reference Index web page[7].

3.2.2.2 Provisional Data Analysis

The provisional data analysis task is focused on examining the network properties of the data using the in-built tools in the Wireshark software. The in-build analysis tools are specially designed to allow network data to be reconstructed and viewed from a network level. The in-built analysis tools allow the data to be examined in a way that would not be possible within statistical analysis software.

It is anticipated that the three sub-tasks will be undertaken during this task. 1) Data record exploration to understand the high level characteristics of the dataset 2) TCP/UDP flow reconstruction to show how may flows were created within each data trace 3) Network input/output (IO) graphs to help visualise the traffic flow on the network and potentially highlight any differences between the active and the passive traces.

3.2.2.3 Expose IP Packet Features

Network IP packets have hundreds of potential features available. In Wireshark the packet attributes are called display filters. For example, the TCP protocol part of the IP packet has 207 display filters[8]. Wireshark's most powerful feature is its vast array of display filters (over 174000 fields in 1000 protocols as of version 1.12.3)[9]. Some display filters may not be populated for an IP packet because not all the attribute are populated within and IP packet. Also encrypted network packets will have a reduced set of display filters available because attributes can't be identified due to the encryption.

There are a large amount of features recommended in the literature. Nguyen and Armitage (2008) presented a list of features previously used in 18 key papers in the

[7] https://www.wireshark.org/docs/dfref/ Date Accessed 12/02/2015

[8] https://www.wireshark.org/docs/dfref/t/tcp.html Date Accessed 12/02/2015

[9] https://www.wireshark.org/docs/dfref/ Date Accessed 12/02/2015

area of IP traffic classification. A summary of the key features are presented below. The number of papers that the feature appears in is shown in square brackets.

- Packet features
    - Packet length [5]
    - Packet length statistics (min, max, mean, std dev.) [4]
    - Inter-Packet lengths statistics (min, max, mean, std dev.) [1]
    - Average inter packet gap [1]
    - Packet Inter-arrival time (minimum, mean, maximum and standard deviation) [5]
    - Packet arrival order [1]
- Protocol features
    - Size of TCP/IP control fields [1]
    - Protocol [2]
    - Numerous TCP-specific values derived from TCP trace (e.g. total payload bytes transmitted, total number of PUSHED packets, total number of ACK packets carrying SACK information etc.) [1]
- Bytes features
    - Payload size [2]
    - mean payload length excluding headers [2]
    - Number of bytes transferred (in each direction and combined) [1]
    - Number of bytes transferred [1]
    - Message size (the length of the message encapsulated into the transport layer protocol segment) [1]
- Flow features
    - Flow volume in bytes and packets [1]
    - Flow metrics (duration, packet-count, total bytes) [1]
    - Flow duration [3]
    - Total packets in each direction and total for bi-directional flow [1]
    - Total number of packets [3]

Similar to previous research this project will be based on packet level analysis, as opposed to flow based analysis. This project will take a different approach to previous literature when it comes to feature generation.

Previous research has tried to compute packet or flow features such as "Mean payload length excluding headers". Computing packet or flow features can have significant processing overheads. Also, waiting for a flow to complete could mean storing all contents of the flow, possible in memory for a considerable period of time. There will also be a very large number of flows active on a network at any point in time. Storing all flows until completion is not possible in real world networks.

This project will look at inherent packet features only. Inherent IP packet features are available to a classification algorithm as soon as the packet is presented. No feature computation will be considered. This approach is taken for two reasons 1) to allow for real time deployment where fast classification of packets is required as each packet arrives 2) to test if feature computation is necessary or inherent packet features alone can be used by a machine learning algorithm to classify active versus passive network traffic with an equivalent level of accuracy as previous research.

This project will also look to identify new features, not previously considered in research that could be beneficial during the machine learning modelling phase. Identifying new features is especially important for this project of the approached taken not to compute new features. A propriety set of features will be required. The new feature generation process will be enabled by the large number of inherent IP packet attributes available. It is anticipated that the feature generation will be an iterative process.

3.2.3 Step 3 – Machine Learning Modelling

Supervised learning techniques are the predominant ML classification used in research in IP traffic classification. This research will also investigate supervised learning techniques to try to identify user-generated versus app-generated traffic. Please refer to Section 2.1.6 for an overview of ML techniques previously used in the area of IP

traffic classification. These ML techniques will form the starting point for this phase of the project. Other techniques may be used if deemed appropriate.

3.2.3.1 Software: SPSS

This project requires machine learning software to build the end to end machine learning models. The software is also used to evaluate the performance of the machine learning techniques. IBM SPSS Software was chosen as the software to build and test the machine learning algorithms (IBM SPSS Modeler 15.0). IBM SPSS software was chosen for the below reasons:

- Industry leading statistical analysis software.
- Fully featured statistical analysis tool
- Full suite of ML algorithms available, including ML techniques considered in this research
- Access – software is available for this research for free
- Experience and expertise in using this software

3.2.3.2 Combine Data

Each of the individual data trace files will be passed to the SPSS software as csv files. The files will then be combined to create a single dataset that can be used in the ML model building process.

3.2.3.3 Secondary Data analysis

Using SPSS the final dataset is analysed. This is an important data understanding phase. The analysis is broken out by the active and passive indicator to allow for analysis of features that may be distinctive between the active and passive datasets and therefore have high importance factors for machine learning algorithms. During this step the final data preparation tasks will take place such as filtering non distinctive or null value features.

3.2.3.4 Feature Reduction

This project will attempt to find the best features to use to detect user-generated versus app-generated IP traffic. The best features to use will vary based on the data mining technique and the data being analysed. The task at this step is to identify the optimal

set of features that minimizes the processing cost, while maximizing the classification accuracy.

3.2.3.5 Build Machine Learning Models

The literature review section outlined the numerous different ML techniques that have been extensively applied to the problem of IP traffic classification. Supervised ML techniques are much more prevalent in the literature. Below is a summary of the ML techniques used in the papers section above.

Supervised ML techniques

- Decision Tree
    - Decision Tree [4 papers]
    - Naïve Bayes Tree
- Neural Network
    - Neural Net [2 papers]
    - Multilayer Perceptron (MLP)
    - Radial Basis Function Neural Network (RBF)
    - Bayesian trained neural network
- K-Nearest Neighbour
    - k-NN [3 papers]
- Support Vector Machine
    - SVM [2 papers]
- Various Naïve Bayes techniques
    - Naïve Bayes Algorithm [2 papers]
    - Naïve Bayes Estimator [2 papers]
    - Bayes Net Algorithm
    - Bayesian Network
    - Naïve Bayesian classifier
    - Naïve Bayes Discretisation
    - Naïve Bayes Kernel density estimation

Unsupervised ML techniques

- AutoClass [2 papers]

- K-Means
- DBSCAN

This project will look at the five supervised ML learning techniques. These techniques have been chose because they are the most used techniques in the relevant literature and are the best performing ML learning techniques based on the experimental outputs.

ML Technique	No. Papers in Literature Review with Technique	No of time Best Performing Technique	Average Accuracy
Decision Tree	4	2	96.9%
SVM	2	2	98.7%
Naïve Bayes Estimator	2	2	89.1%
Neural Network	2	1	95-97%
KNN	3	1	>90%

Table 3-1 : Machine Learning Techniques in Scope

3.2.4 Step 4 - Model Evaluation

In the evaluation task, the most commonly used evaluation criteria found in the literature are applied to the machine learning models built as part of this experiment.

3.2.4.1 Evaluation Criteria

The evaluation criteria will be based on the evaluation metrics outlined in the literature review. There are a four main of evaluation metrics proposed in the IP traffic classification literature for supervised ML algorithms: accuracy, precision, recall and F-measure: There are two additional evaluating criteria proposed in the literature for real-time IP traffic classifiers that will be considered in this research; model build time and classification time. Refer to Section 2.1.10 for full details of the evaluation criteria.

3.3 Expected Results

The expected results from this research would be to develop a supervised machine learning model that performs well at classifying active and passive app network data based on the chosen evaluation metrics.

3.4 Conclusion

This chapter has outlined the design methodology for the project experiment. The experiment, as designed, should allow this research project to develop a machine learning algorithm that can accurately identify active and passive app network data. The next chapter will detail how the experiment design methodology was implemented.

4 EXPERIMENT IMPLEMENTATION

4.1 Introduction

This chapter outlines the experiment implementation. The implementation of each step in the experiment designed is outlined and discussed. This chapter will follow the same heading structure as Chapter 3 Design Methodology.

4.2 Solution Approach

The experimental implementation followed a five step solution approach outlined in experimental design chapter.

4.2.1 Step 1 - Data Capture

The data collection phase requires mobile phone hardware and software as well as personal computer based software.

4.2.1.1 Hardware: Android device

The first step in the data collection process is to set up the mobile phone device. The HTC One S running Android OS version 4.1.1 is rooted to prepare the device for the installation of the required IP packet capture software.

4.2.1.2 Software: Shark for Root

Once the device is rooted the IP packet capture software is installed on the mobile device. Shark for Root software was installed from the Google Play Android Store.

4.2.1.3 Apps: Applications considered

A specific set of eleven apps were chosen to be used during the data trace. These apps were downloaded from the Google Play Android Store and installed on the device. If necessary, accounts were set up within the app to allow access to the app and to facilitate app traffic generation.

4.2.1.4 Process: Data Traces

In order to capture samples of active and passive app network traffic data, three separate data traces were implemented. Each trace captured a specific type of active or passive app network data. The full data capture process was performed once which resulted in a total of 3 data traces.

4.2.1.4.1 Data Trace Process

Data trace 1 captured active app network traffic data.

Trace 1 – Active trace

a. Restart phone – wait 1 minute
b. Open Shark For Root software on mobile device
c. Start data capture with parameters *–I wlan0 –s 0*
d. Open and Use all 11 apps for 30 minutes.
e. Dates and time:
    i. (Nov 27th at 5.53pm to 6.25)
f. Actions
    i. YouTube
        1. Watch 1 min video - stop
        2. Watch 2nd 1 minute video.
    ii. Gmail
        1. Opened app twice
        2. Sent 2 emails
    iii. Facebook
        1. Opened App twice
        2. Open Facebook time line
        3. Browse who to follow section
        4. Open Facebook wall profile page
    iv. Twitter
        1. Opened app twice
        2. Send 2 tweets
    v. Angry Birds
        1. Opened app once
        2. Play game for 1 minute

vi. Skype
    1. Opened app twice
    2. 3 messages sent – 3 received
    3. 1 minute video call
vii. WhatsApp
    1. Opened app 3 times
    2. 4 messages sent – 4 received
viii. Facebook Messenger
    1. Opened app 3 times
    2. 4 messages sent – 4 received
ix. Viber
    1. Opened app 3 times
    2. 4 messages sent – 4 received
x. Snapchat
    1. Opened app twice
    2. 2 pictures sent
xi. Instagram
    1. Opened app twice
    2. 1 pic sent –
    3. 2 users followed
    4. multiple pics browsed
    5. 1 person started following me.

In order to fully capture passive app network data, two separate passive data traces were implemented. Data trace 2 was implemented to capture passive app network traffic when an app may have been left running on the mobile device.

Data Trace 2 – Passive trace 1

a. Following above test – wait 5 minutes
b. Apps are open and may be running on the device but there is no user interaction with the apps
c. Open Shark For Root software on mobile device
d. Start data capture with parameters *–I wlan0 –s 0*
e. Do not use any apps for 30 minutes. Phone is completely idle
    i. Dates and time (Nov 27th at 6:30pm to 7:00)

Data trace 3 was implemented to capture passive app network traffic when no apps have specifically been opened by the end-user.

Data Trace 3 – Passive trace 2

a. Restart phone
b. Wait 5 minutes so that phone is in a steady state
c. No apps were opened
d. Open Shark For Root software on mobile device
e. Start data capture with parameters *–I wlan0 –s 0*
f. Do not use any apps for 30 minutes. Phone is completely idle
    i. Dates and times (Nov 27th at 7:15 pm to 7:45)

4.2.1.5 Data Preparation

Each record in data trace 1 were tagged as "active". Each record in data trace 2 and 3 were tagged as "passive". These datasets form the basis of the training and testing data for the machine learning algorithms. The data traces are then exported from the mobile device to a PC for analysis and prepared for the modelling phase.

4.2.2 Step 2 - Feature Generation

This section starts by examining the data captured on the mobile device. Then the process of exposing the inherent IP packet features is documented. Finally the output record format is presented.

4.2.2.1 Software: Wireshark

Wireshark software was installed and configured on the experiment PC hardware.

4.2.2.2 Provisional Data Analysis

Wireshark has a number on in-built tools that allows for easy analysis of the network characteristics of captured network traffic data. This section outlines the provisional data analysis using some of the Wireshark tools.

4.2.2.2.1 Data Exploration

This section focuses on exploring and understanding the characteristic of the data collected. Visualisations were created to help add meaning to the data. A clear understanding of the data characteristics will lead to an optimised machine learning process and improved interpretation of the results.

The data exploration task was undertaken using the Wireshark software. The packet capture (.pcap) files are collected from the mobile device and opened using the Wireshark software. Table 4-1 shows the number of records in each data traces.

Data Trace Name	**Number of Records**
Data Trace 1.1 (Active)	128,856
Data Trace 1.2 (Passive)	10,234
Data Trace 1.3 (Passive)	17,976
Total	157,066

Table 4-1 : Data Traces Record Summary

4.2.2.2.2 Flow Reconstruction

Network flows are point to point connections between source and destination ports. The network flows in the captured data traces were reconstructed using Wireshark. This project is based on packet level analysis rather than flow level analysis but it is important to understand the number and types of flows present in the data in order to have a full understanding of the data.

Table 4-2 shows the count of each type of network flow found in each of the data traces. The active data trace has a far greater number of flows compared to the two passive data traces. UDP network flows are significantly higher in the active data trace.

Trace	**TCP Flows**	**UDP Flows**	**IPv4 Conversations**
Trace 1.1 (Active)	502	467	223
Trace 1.2 (Passive 1)	152	26	24
Trace 1.3 (Passive 2)	101	15	28

Table 4-2 : Network flow reconstruction analysis

4.2.2.2.3 End Point Analysis

End point analysis allows the number of end point IP addresses to be analyses. End point analysis was performed using the Wireshark software. Table 4-3 shows the number of end point IP address identified in each of the data traces. The active data traces has network traffic going to significantly more IP addresses than both of the passive data traces.

Trace	Total End Point IP Addresses
Trace 1.1 (Active)	223
Trace 1.2 (Passive 1)	24
Trace 1.3 (Passive 2)	31

Table 4-3 : End point IP address analysis

4.2.2.2.4 Network IO Graphs

Using Wireshark, network IO graphs can be generated. The IO graphs are graphic representation of the number of packets travelling on the network over a period of time. The IO graph visualisations can also be very useful to understand the differences in packet volumes between the active and passive traces.

The IO graphs show the three main protocols detected by volume, TCP in blue, UDP in red and ICMP in green. The units in the graph are packets. The time interval is 1 second. The x-axis shows time of day and the y-axis shows packet count.

4.2.2.2.4.1 Active Trace: Network IO Graph

Figure 4-1 shows the IO graph created for the active data trace. The graphs shows a visual representation of the packets associated with each of the 3 main protocols. The y-axis scale is 0-1000 packets.

In general, there is a constantly high stream of packets across the whole data trace. The increase in UPD (red line) traffic at 18:01 to 18:02 is the 1 minute Skype call established as part of the data trace actions outlined in Section 4.2.1.4.1.

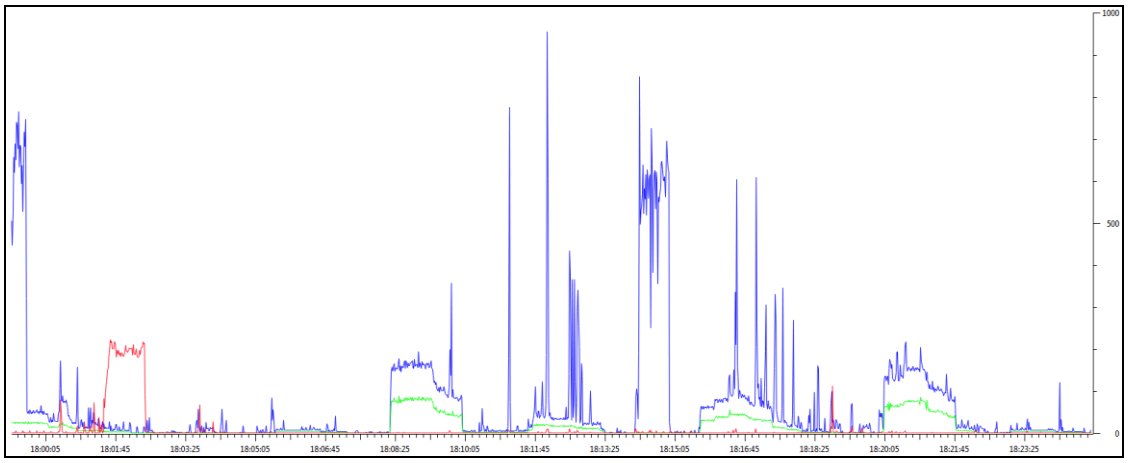


Figure 4-1: IO Graph 1 - Active data trace (Scale 0-1000)

4.2.2.2.4.2 Passive Trace 1: Network IO Graph

Figure 4-2 shows the IO graph created for the first passive data trace 1. The graphs shows a visual representation of the packets associated with each of the 3 main protocols. The y-axis scale is 0-1000 packets. In general, there is a relatively low stream of packets across the first passive data trace.

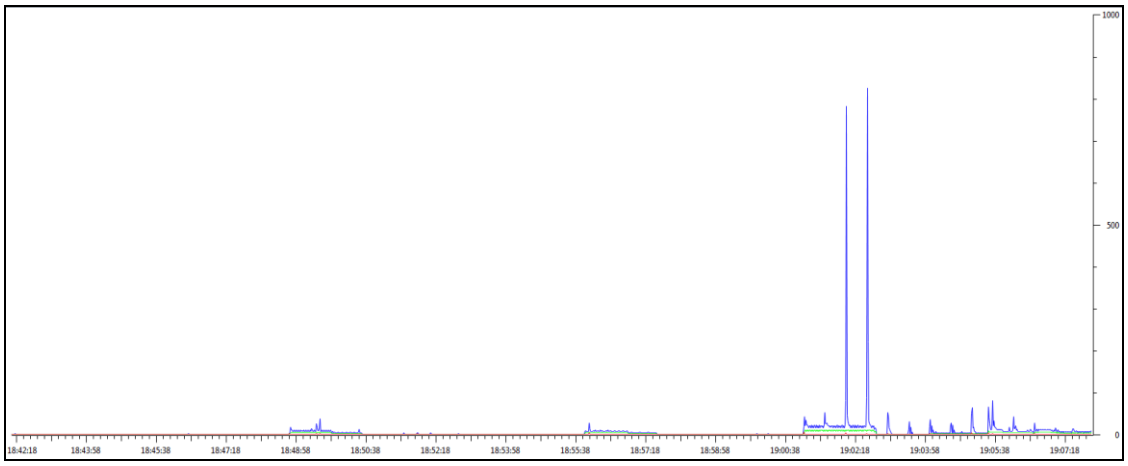


Figure 4-2 : IO Graph 2 - Passive data trace 1 (Scale 0-1000)

In order to demonstrate that even though the steam of packets is low, there is a constant flow of packets across the data trace, the same graph as Figure 4-2 was created with a y-axis rescaled to 0-100 packets. Figure 4-3 shows the re-scaled IO Graph.

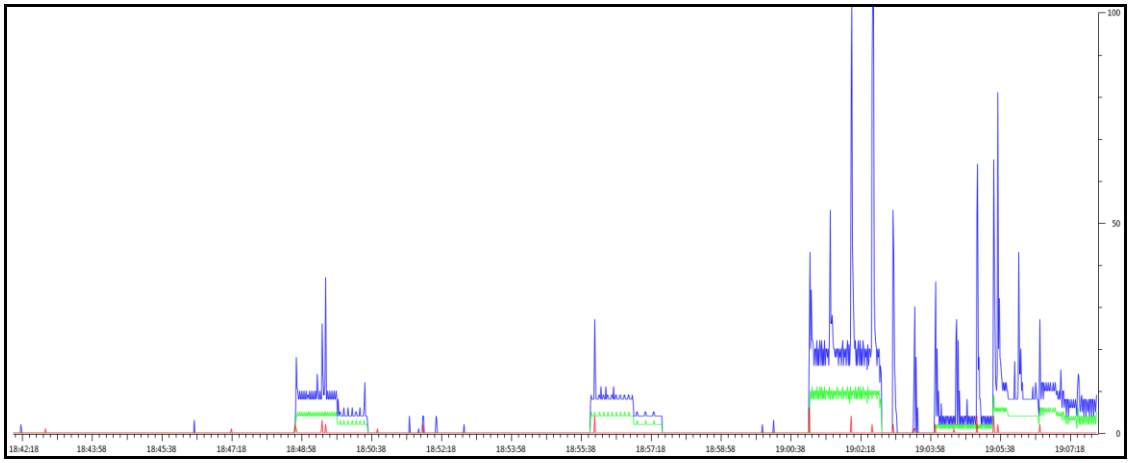


Figure 4-3 : IO Graph 3 - Passive data trace 1 (Scale 0-100)

4.2.2.2.4.3 Passive trace 2

Figure 4-4 shows the IO graph created for the second passive data trace. The graphs shows a visual representation of the packets associated with each of the 3 main protocols. The y-axis scale is 0-1000 packets. In general, there is a relatively low stream of packets across the second passive data trace

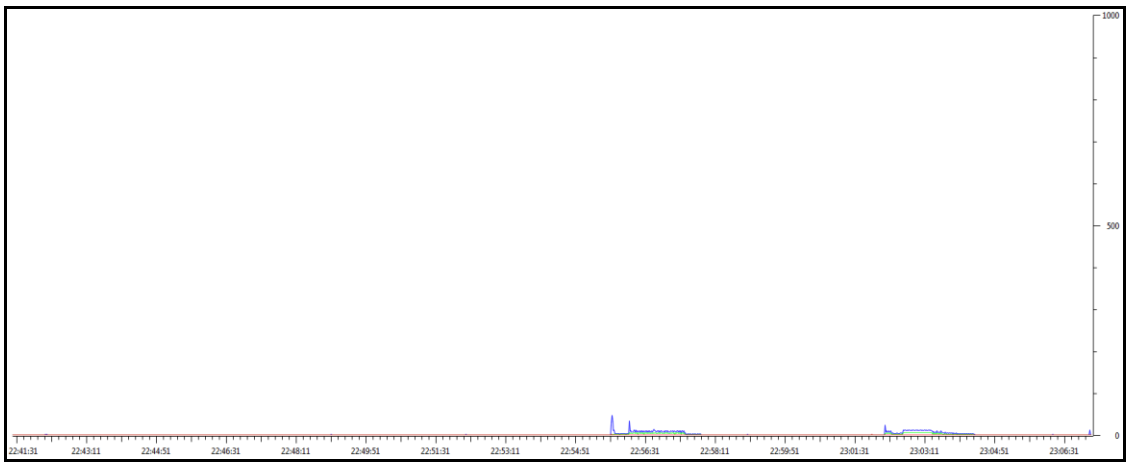


Figure 4-4: IO Graph 4 - Passive data trace 2 (Scale 0-1000)

In order to demonstrate that even though the steam of packets is low, there is a constant flow of packets across the data trace, the same graph as Figure 4-4 was created with a y-axis rescaled to 0-100 packets. Figure 4-5 shows the re-scaled IO Graph.

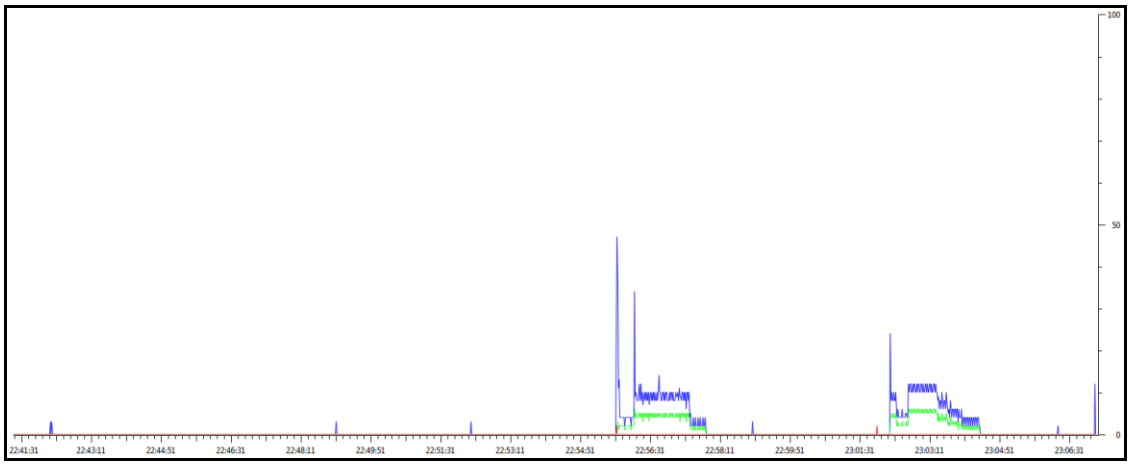


Figure 4-5: IO Graph 5 - Passive data trace 2 (Scale 0-100)

The IO graphs from the three different data traces were compared and clearly highlight the very different characteristics of the active and passive traces.

4.2.2.3 Expose IP Packet Features

Each record in the data trace datasets represents a single network IP Packet. Network packets have hundreds of potential features available. The section will outline the network packet features examined and which of these features were used in the machine learning process.

Network packets have hundreds of potential features available. In Wireshark the network packets attributes are called display filters. Wireshark has over 174,000 display filters available, as of software version 1.12.3.

The primary display filters examined as part of this project were

- Frame
- Eth
- IP
- TCP
- UDP
- HTTP
- x509sat
- SSL

The IP packets were exposed using the preference screen in Wireshark.

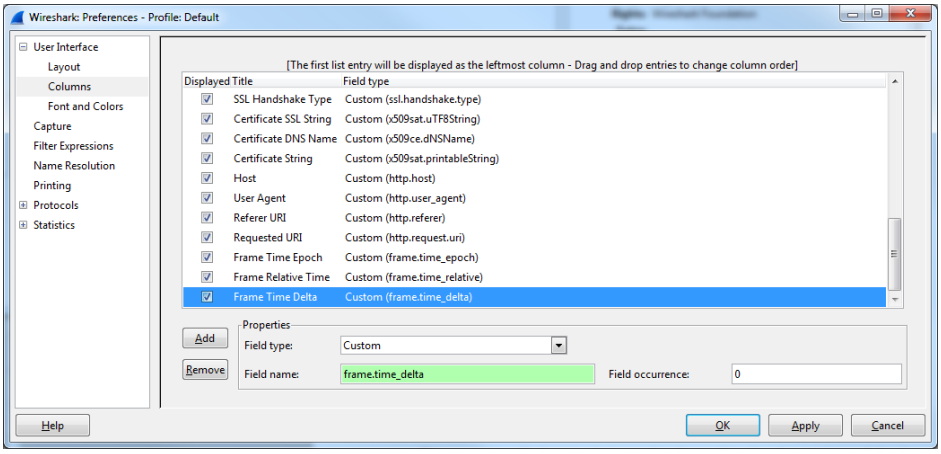


Figure 4-6 : Wireshark preference menu

4.2.2.3.1 Output Data record format – Data dictionary

After completing the examination of the main packet attributes, 49 field were identified to form the output record. This section outlines the data dictionary for the output data set. The output dataset from this step will form the input to the next step, ML modelling.

Note that the data is at packet level. Also the data is bi-directional, for example the IP address of the mobile device is the source IP address in outbound data records, for inbound data records the IP address of the mobile device is the destination IP address.

Some fields will not be useful for ML but were left in to help with data exploration and troubleshooting, such as IP Frame Number, TCP Flow No. and TCP Sequence Number. A full table of the data dictionary is available in Appendix B.

4.2.3 Step 3 - Machine Learning Modelling

Supervised learning techniques are the predominant ML techniques used in research in IP traffic classification. This research also investigated supervised learning techniques to attempt to identify user-generated versus app-generated traffic. Please refer to Section 2.1.6 for an overview of ML techniques previously used in the area of IP traffic classification. These ML techniques formed the starting point for this phase of the project.

4.2.3.1 Software: SPSS

This project requires machine learning software to build the end to end machine learning models. IBM SPSS Modeler 15.0 was installed on the research PC device.

4.2.3.2 Combine Data

Each of the individual data trace were loaded into the SPSS software via intermediate csv files. The files were then combined to create a single dataset that can be used in the machine learning modelling process.

4.2.3.3 Secondary Data Analysis

Once the data is available in SPSS, secondary data analysis was undertaken. The secondary data analysis was concerned with examining the statistical characteristics of the data that may be important considerations during the ML model build step.

A number of key graphs were created that highlight the important characteristics of the data. The first graph created was used to examine the protocols present in the full data set. Figure 4-7 shows a bar chart of the number of packets (data rows) by protocol. The graph shows that there are 4 main protocols in the data trace TCP, ICMP, TLSv1 and UDP. Table 4-4 shows the percentage of the total packets by protocol.

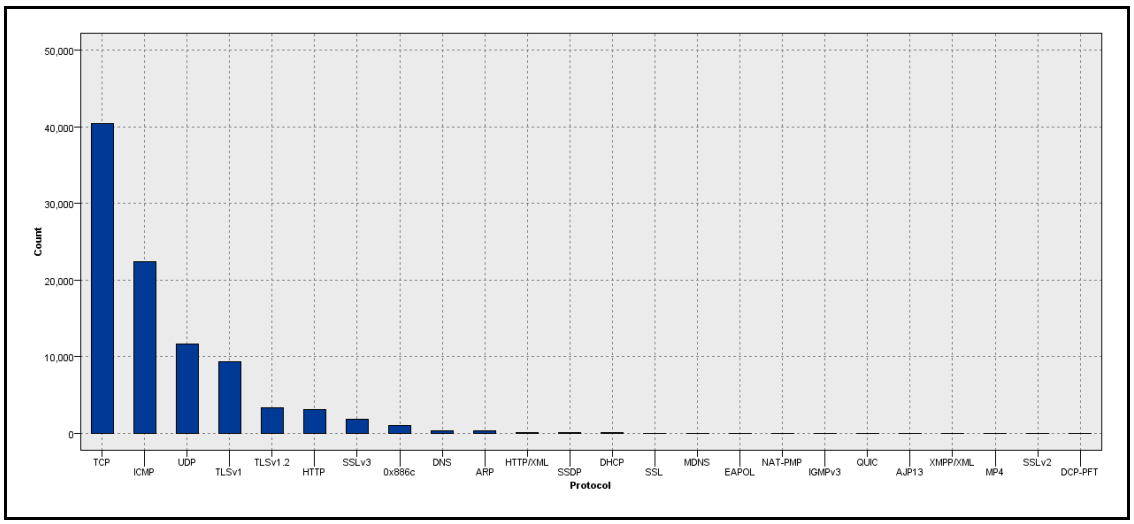


Figure 4-7 : Bar Chart of Total Packets by Protocol

Protocol	Packet Count	% of Total Packets
TCP	82,753	52.7%
ICMP	34,593	22.0%
TLSv1	13,823	8.8%
UDP	11,819	7.5%
SSLv3	4,088	2.6%
HTTP	3,784	2.4%
TLSv1.2	3,619	2.3%
0x886c	1,287	0.8%
DNS	569	0.4%
ARP	337	0.2%

Table 4-4 : Breakdown of Top 10 Protocols

The next step in analysing the protocol characteristics of the data was to break out the protocol bar chart by active versus passive traces. This process allows for a side by side comparison of the protocol characteristics of the data, which highlighted key differences the active and passive datasets. Figure 4-8 shows very different counts of packets in each protocol between the active and passive traces. Also, there is almost no UDP in the passive trace. There is almost no secure protocol traffic either (SSLv3 or TLSv1.2).

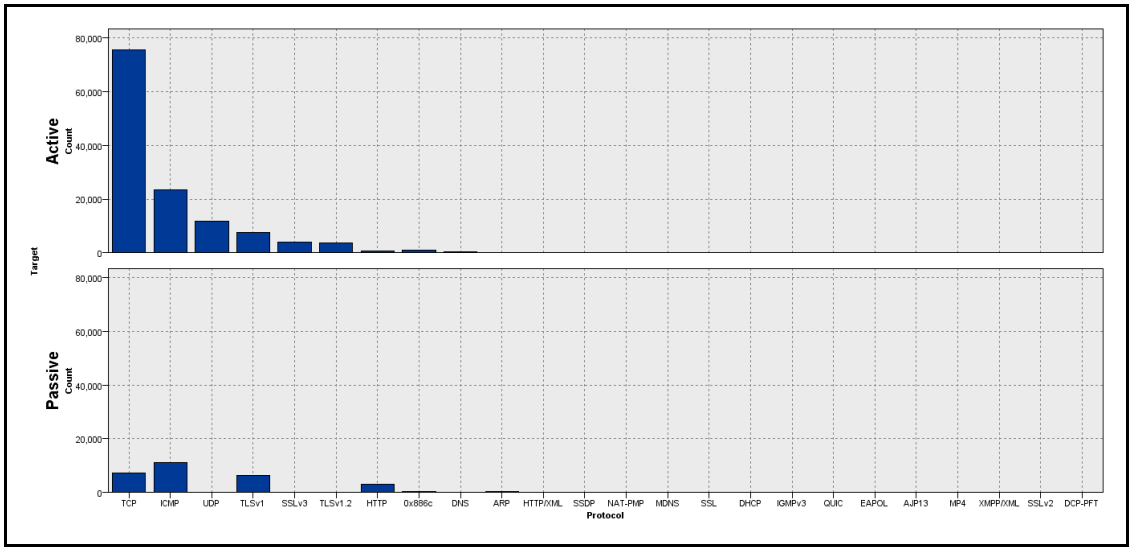


Figure 4-8 : Active versus Passive - Bar Chart of Total Packets by Protocol

The destination IP addresses were then analysed. Figure 4-9 shows a bar chart of the number of packets by destination IP address broken out by active versus passive datasets. The graph is limited to the top 30 destination IP addresses. Also, the IP address of the mobile device has been excluded from the graph. Figure 4-9 highlights the wider range of destination IP addresses found in the active data trace.

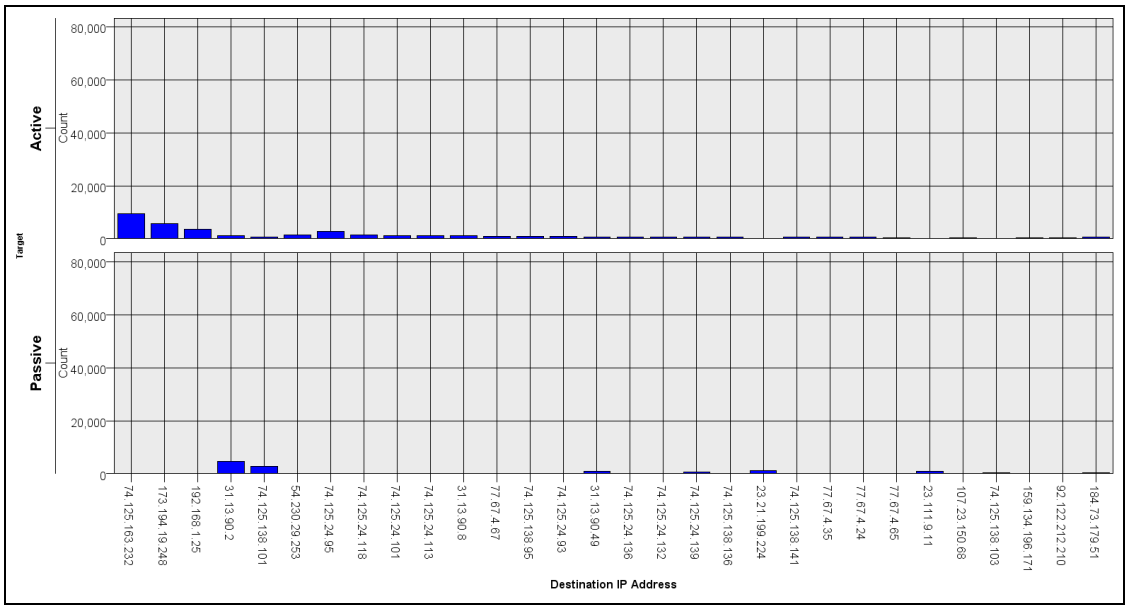


Figure 4-9 : Bar Chart of Total Packets by Destination IP Address

The Time Frame Delta feature was then analysed. Figure 4-10 shows a distribution of the Time Frame Delta values broken out for the active and passive data traces. Time Frame Delta is a measure in seconds (to 6 decimal places) of the time between successive network frames or packets arriving at the data capture point.

Figure 4-10 highlights that for the active trace, almost all Time Frame Delta values are close to zero. Whereas for the passive trace there is a far greater spread of Time Frame Delta values.

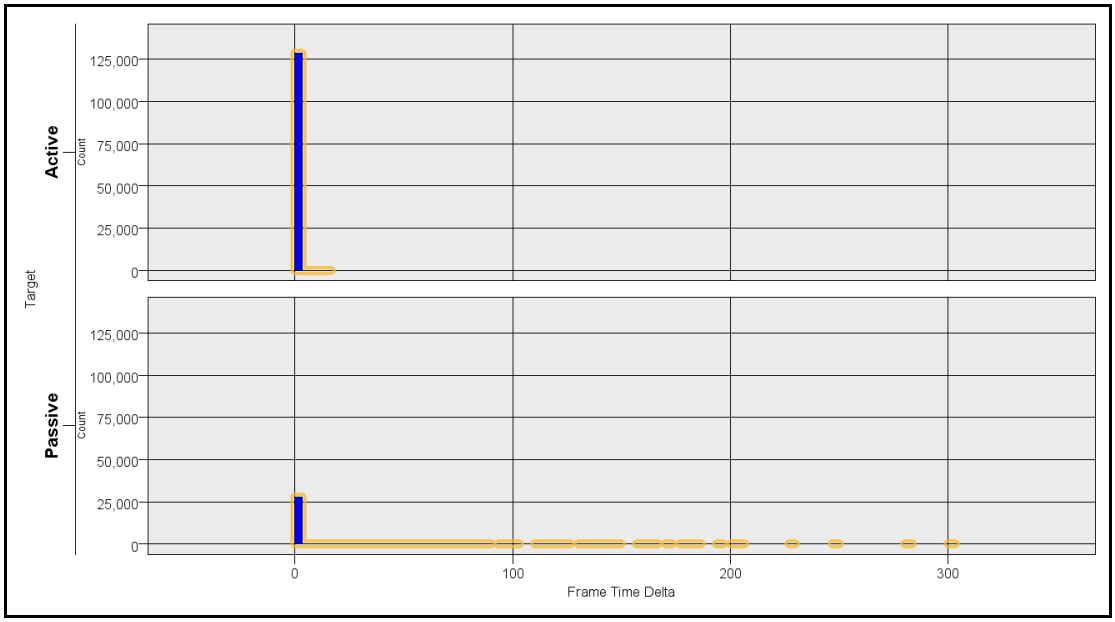


Figure 4-10 : Distribution of the Time Frame Delta feature values

The next feature analysed was Bytes Total. Bytes total is a count of the number of bytes contained in each network packet. Figure 4-11 shows a distribution of the Bytes Total broken out for the active and passive data traces. The difference in the two distribution graphs can clearly be seen in Figure 4-11. The active trace has a large number of packets with approx. 1500 bytes, whereas the passive data trace has very few packets with a large number of bytes.

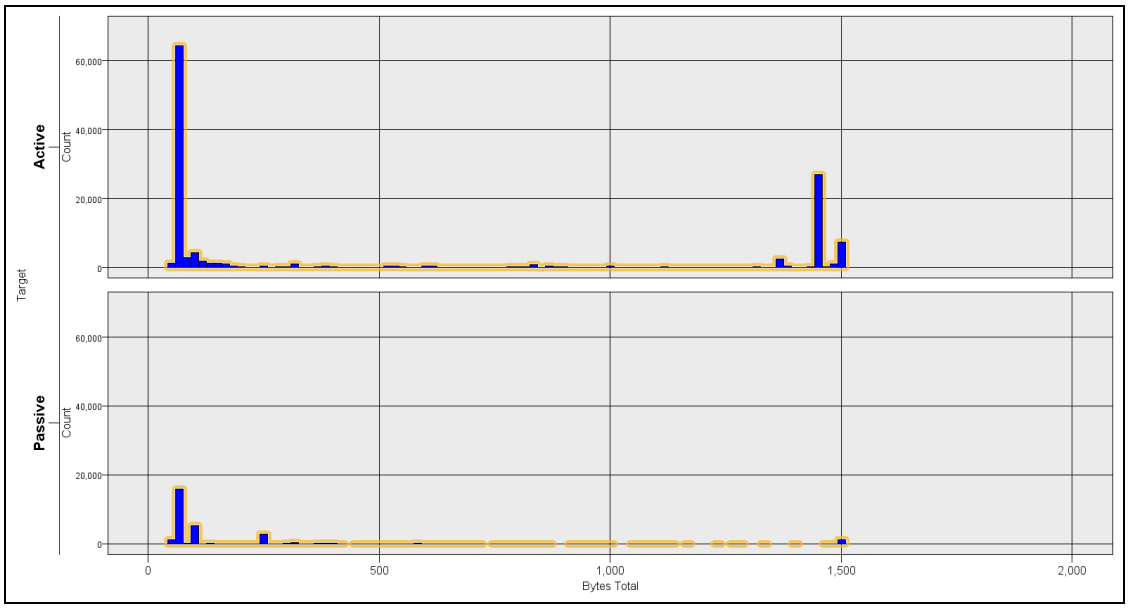


Figure 4-11 : Distribution of the Bytes Total feature values

In summary, the secondary data analysis phase highlighted some key differences in the characteristics of the active and passive data sets. These differences will be used by the machine learning algorithms to try to accurately classify active and passive data records.

4.2.3.4 Feature Reduction

This section outlines the feature reduction tasks. The input dataset into the ML model has 49 features based on the complete data trace dataset.

Following analysis and data inspection, 12 features were filtered out of the data set.
3 Features were removed because they were identifying record values:

- IP Frame Number
- TCP Flow No
- TCP Sequence No

9 Features were removed because they contained sparse or no data values

- Info
- DNS Answer Count
- DNS Query Name
- SSL Handshake Type
- Certificate SSL String
- Certificate DNS Name
- Certificate String
- User Agent
- TCP Option Len

As part of the modelling process, the feature space was further reduced. For example, an implementation of a CHAID decision tree reduced the feature space by ignoring 27 features listed below:

- Time
- Source IP Address
- Destination IP Address
- Combined Ports
- TCP Flag Syn

- TCP Flag Ack
- TCP Flags
- DNS IP Addresses
- Host
- Referer URI
- Requested URI
- Frame Time Epoch
- Frame Relative Time
- TCP Window Size Scalefactor
- TCP Ack RTT
- TCP Window Size Value
- TCP Options
- TCP Options MSS
- TCP Options Sack Perm
- TCP Options Sack Count
- TCP Options Sack Len
- TCP Options Time tsval
- TCP Options Type
- TCP Options Type Class
- TCP Options Type Number
- TCP Options WScale Shift
- TCP Options WScale Multi

The CHAID decision tree found for example found 10 features with a predictor importance.

- Frame Time Delta
- TCP Bytes in Flight
- Protocol
- TCP Window Size
- Source Port
- Bytes Total
- TCP Options Time tsec

- TCP Options Kind
- TCP Flag Fin
- Destination Port

4.2.3.5 Build Machine Learning Models

The experiment implemented five supervised ML techniques; Decision trees, SVM, Naïve Bayes Estimator, Neural Networks and KNN. These techniques have been chose because they are the most used techniques in the relevant literature and are the best performing ML learning techniques based on the experimental outputs.

SVM, Naïve Bayes Estimator, Neural Networks and KNN were found to have low overall model accuracy after the initial model build phase and were excluded from further investigation.

During the model building phase three decision tree algorithms emerged as having a high overall model accuracy, C5.0, CHAID Tree and C&R Tree. These models formed the focus of the full model development and evaluation.

4.2.4 Step 4 - Model Evaluation

This project attempted to find the best features to use to detect user-generated versus app-generated IP traffic. The best features to use will vary based on the data mining technique and the data being analysed. The task at this step is to identify the optimal set of features that minimizes the processing cost, while maximizing the classification accuracy. Please refer to the Feature Selection section above for a list of previously use feature selection techniques in the area of IP traffic classification. These techniques will form the starting point of the feature reduction process. Other techniques may be used if deemed appropriate.

Although the three decision tree models were found to have a high overall model accuracy, two of the models (CHAID tree and C&R tree) had low precision values when classifying passive traffic. Following an initial review of the evaluation metrics model boosting was implemented on the three decision tree models to specifically try to improve the passive traffic classification precision metric.

4.3 Strength and Limitations of Experiment Approach

This section will outline the strengths and limitation of the experimental approach.

The strengths of the experimental approach are as follows:

- The research sets out an innovative experimental approach to capture, analyse and run predictive analytics on mobile network data traffic
- The experimental approach sets out a clear, robust and repeatable end-to-end process for mobile device network traffic predictive analytics.
- The research uses the inherent IP Packet features rather than generating features. Generating features can be time consuming and processor intensive.
- The experimental approach sets out to find new features not considered in previous research.

There are some limitations with the experimental approach that need to be acknowledged:

- Only one device was used to capture the data. Using only one device make it easier to attribute traffic to a device. Using multiple devices would address real world deployment considerations.
- A limited number of apps were considered in order to set a manageable scope for the research. Apps can have very different architectures so including more apps would increase the validity of the research.
- The data traces were captured for a limited time period of 30 minutes in order to keep the output data at a manageable level. Longer data traces could show characteristics of the data not captured in this experimental set up.
- The research only focused on a device running Android OS. Running data traces on an Apple device is far more complex process then running data traces on and Android device. Apple IOS has an equivalent mobile market share to Android[10] so the research would benefit from network data captured on an Apple IOS device.

[10] https://gigaom.com/2015/02/04/android-and-ios-are-nearly-tied-for-u-s-smartphone-market-share/ Date Accessed 10/02/2015

4.4 Expected Results

The expected results from this research would be to develop a supervised machine learning model that performs well at identifying active and passive app network traffic based on the chosen evaluation metrics.

4.5 Conclusion

This chapter has outlined the design methodology for the project experiment. The experiment, as designed, should allow this project to develop a machine learning algorithm that can accurately identify active and passive app network data. The next chapter will detail how the experiment design methodology was implemented.

5 EVALUATION AND DISCUSSION

5.1 Introduction

This chapter presents the findings from the empirical study, as specified in Chapter 4. The results were analysed and compared to the findings from the literature review. The weaknesses and limitations of the research are presented and critically discussed.

5.1.1 Evaluation Criteria

The evaluation criteria were based on the prominent evaluation metrics identified in the literature review of relevant research in the area of network traffic classification, as outlined in the section 2.1.10.

There are a four main of evaluation metrics proposed in the IP traffic classification literature for supervised ML algorithms:

- Accuracy
- Recall (or Sensitivity or True Positive Rate (TPR))
- Precision (or Positive Predictive Value (PPV))
- F-Measure

The first step in evaluating the ML models created was to create a confusion matrix for each model. Predicting passive traffic correctly is the main objective of the research. Passive traffic makes up only 18% of the total dataset so it is important to specifically understand how accurately a model can predict passive traffic. For example, a model could have an overall accuracy of 82% but still predict all passive traffic incorrectly. In a real work network correctly identifying passive traffic is vital because passive traffic can be deprioritised on the network due to the fact that the end-user is not actively seeking the data.

		Predicted Class	
		Passive	Active
Actual Class	Passive	True Positive (TP)	False Positive (FP)
	Active	False Negative (FN)	True Negative (TN)

Table 5-1 : Confusion Matrix Example

Based on the confusion matrix, the evaluation criteria were then calculated as follows:

Evaluation Criteria	Calculation
Accuracy	(TP + TN) / (TP + TN + FP + FN)
Recall	TP / (TP+FP)
Precision	TP / (TP+FN)
F-Measure	(2 x Precision x Recall) / (Precision + Recall)

Table 5-2 : Evaluation Criteria Calculations

After a preliminary review of the results, it became clear that an additional evaluation criteria would need to be included. Negative Predictive Value (NPV) was added to the list of evaluation criteria. For clarity NPV will be referred to as Active Predictive Precision (APP). The existing precision metrics is a measure of passive traffic precision. For clarity precision, in this context, will be referred to as Passive Predictive Precision (PPP). Table 5-3 shows the PPP and APP metric calculations

Evaluation Criteria	Calculation
Passive Predictive Precision (PPP)	Precision = TP / (TP+FN)
Active Predictive Precision (APP)	NPV = TN / (TN + FN)

Table 5-3 : Passive and Active Predictive Precision Metrics

By comparing the PPP and APP metrics side by side the true accuracy of each model at predicting each class can be easily interpreted. A good model should have a high percentage value for both evaluation criteria.

There are two secondary evaluation criteria metrics that were considered. These metrics are posed in the literature relating to evaluating real time IP traffic classifiers. Table 5-4 shows the real-time evaluation metric calculations.

Evaluation Criteria	Calculation
Model Build Time	Total time to build model
Classification Time	Records classified in 1 sec

Table 5-4 : Real time evaluation criteria

5.1.2 Machine Learning Models Considered

The experiment implemented five supervised ML techniques; Decision trees, SVM, Naïve Bayes Estimator, Neural Networks and KNN. SVM, Naïve Bayes Estimator, Neural Networks and KNN were found to have low overall model accuracy after the initial model build phase and were excluded from further investigation.

During the model building phase three decision tree algorithms emerged as having a high overall model accuracy, C5.0, CHAID Tree and C&R Tree. These models formed the focus of the full model development and evaluation.

5.2 Evaluation of Results

In this section the overall experiment results are presented. The evaluation metrics for each of the three ML models built are presented, compared and critically evaluated. The key findings are compared to the findings from the literature review.

The experimental setup and parameters of each model are detailed in Appendix D, E and F.

5.2.1 Evaluation Criteria Results and Comparison

Table 5-6 shows the evaluation metric results for the three ML models built during the experiment implementation.

	CHAID		C&R		C5.0	
Evaluation Criteria	**Std.**	**Boost**	**Std.**	**Boost**	**Std.**	**Boost**
Accuracy	88.3%	95.5%	90.4%	92.4%	95.6%	95.6%
Precision	50.2%	84.8%	60.7%	79.7%	75.8%	75.6%
Recall	76.6%	89.7%	81.2%	78.4%	99.4%	99.9%
F-Measure	60.7%	87.2%	69.4%	79.1%	86.0%	86.1%
Passive Predictive Precision	50.2%	84.8%	60.7%	79.7%	**75.6%**	75.8%
Active Predictive Precision	96.6%	97.9%	96.9%	95.2%	**99.9%**	99.9%
Model Build Time (mm : ss)	00:14	10:37	00:15	10:28	00:52	12:46
Classification Time (Records per second)	1026	400	1000	470	800	500

Table 5-5 : Results - Evaluation Metrics

Round 1 in the model building phase covered developing a standard implementation of the ML techniques. The Std. column in Table 5-6 (highlighted in grey) contains the round 1 results. Round 2 in the model building phase covered developing an implementation of the ML techniques with model boosting enabled. The Boost column in Table 5-8 contains the round 2 results. The model with the highest combination of PPP and APP in round 1 was the C5.0 tree (highlighted in yellow). After model boosting was enabled, the model with the highest combination of PPP and APP in round 2 was the CHAID tree (highlighted in green).

5.2.2 Round 1 – Standard Model Implementation

This section discusses the results from the round 1 model development. APP is very high across all models in round 1 ranging from 96% to 99%. However PPP is lower and had a much wider range from 50.2% to 75.8%. The models are evaluated based on a combination of PPP and APP but due to the consistently high APP values, PPP was the main evaluation metric.

5.2.2.1 CHAID Tree

The standard implementation of the CHAID tree model had the lowest PPP and the lowest combination of PPP and APP. Although APP was very high at 96.6%, PPP is only 50.2%. The model failed to accurately predict passive traffic.

Table 5-1 shows the predictor importance within the model. Two features stand out as important predictors, Frame Time Delta and TCP Bytes in Flight. The preliminary data analysis detailed in Figure 4-10 highlighted the different characteristics of the Frame Time Delta feature in the active and passive data sets.

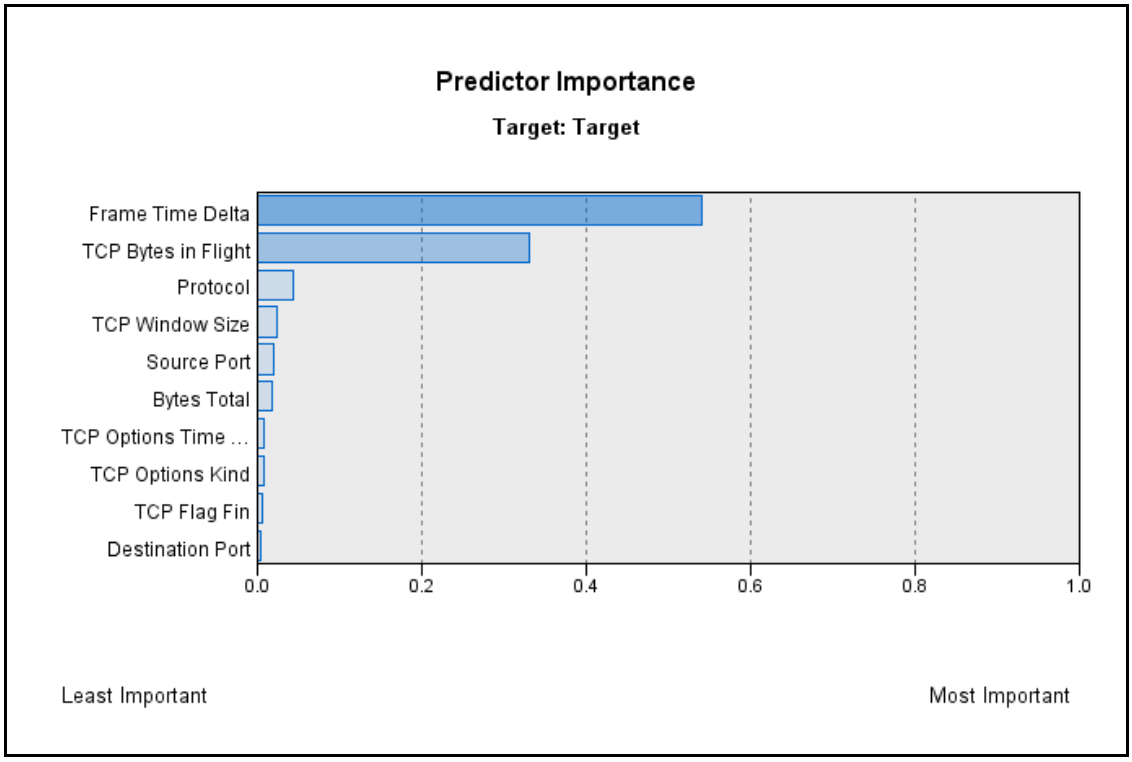


Figure 5-1 : Feature Predictor Importance

5.2.2.2 C&R Tree

The standard implementation of the C&R tree had a slightly higher PPP value at 60.7% comparted to the CHAID tree. The C&R model had a similarly high APP value of 96.9%. The PPP value is still consider too low.

5.2.2.3 C5.0 Tree

The C5.0 tree was the best performing model in round 1. Similar to the CHAID and C&R trees the C5.0 tree had a high APP at 99.9%. The PPP was 75.8. The PPP was 15% higher than the second place C&R model.

5.2.2.4 Results comparison with literature review

This section will compare the findings from model building round 1 with equivalent findings from the literature review. The evaluation metric used by pervious research is overall model accuracy. Because of the importance of correctly predicting passive traffic this research use a specific PPP evaluation metric, but the overall accuracy metric for each model is available. Table 5-7 shows the overall accuracy metric for the models built in round 1.

Model	Overall Accuracy
C5.0 Decision Tree	95.6%
C&R Tree	90.4%
CHAID	88.3%

Table 5-6 : Round 1 – Overall Model Accuracy

Table 5-8 shows the overall accuracy metric for decision trees presented in the literature review.

Paper	Algorithm	Best Performing	Overall Accuracy
Li and Moore, 2007	C4.5 Decision Tree	Y	99.8%
Williams et al., 2006	C4.5 Decision Tree	Y	94.13%
Singh et al., 2013	C4.5 Decision Tree	N	83.1%

Table 5-7: Literature Review - Decision Tree Overall Accuracy

The decision trees built in this research have a very similar overall accuracy range to previous decision tress built in previous research. There is a difference in the datasets between this research and previous research. Also, there is a difference in the decision tree algorithms used in this research and previous research. The C5.0 decision tree

used in this research is the most similar to previous research that used a previous implementation of the same algorithm, C4.5 decision tree. The C&R tree and the CHAID tree do not specifically appear in previous research.

Model boosting is not covered in previous literature. Previous research only investigated the C4.5 decision tree algorithm which does not support boosting. Support for boosting was added in the C5.0 algorithm implementation, which allowed this research to extend the experiment into model boosting development and evaluation.

5.2.2.5 Round 1 Evaluation

APP is very high across all tree decision tree algorithm implementations. However PPP varies widely. The PPP value could still be improved. Other ML algorithms have already been discounted so the next step was to continue evaluation decision tree algorithms by extending their capabilities using model boosting.

5.2.3 Round 2 – Boosting Model Implementation

This section discusses the results from the round 2 model development. This round of development used the same decision tree algorithms as round 1 but added model boosting capabilities with the specific purpose of improving PPP.

5.2.3.1 C5.0 Tree

After round 1 the C5.0 was the best performing algorithm based on having the highest PPP. The C5.0 model was implemented again with model boosting. However there is no improvement in the model percussion metrics. After reviewing the output from the model ensemble each successive model shows no improvement in overall accuracy. In this case, with the specific research data model boosting does not increase the overall accuracy of the C5.0 Model. The C5.0 model drops to 3rd place in round 2 evaluation. It is worth noting that the C5.0 model with boosting classified almost 100% of active traffic correctly.

5.2.3.2 C&R Tree

The C&R tree model was the second place model after round 1. Following implementing the model with boosting functionality the C&R model was evaluated to be in second place in round 2 also. The model with boosting enabled does show an improvement in PPP from 60.7% to 79.7%.

Figure 5.2 shows the overall accuracy improvement in the C&R tree with and without boosting enabled.

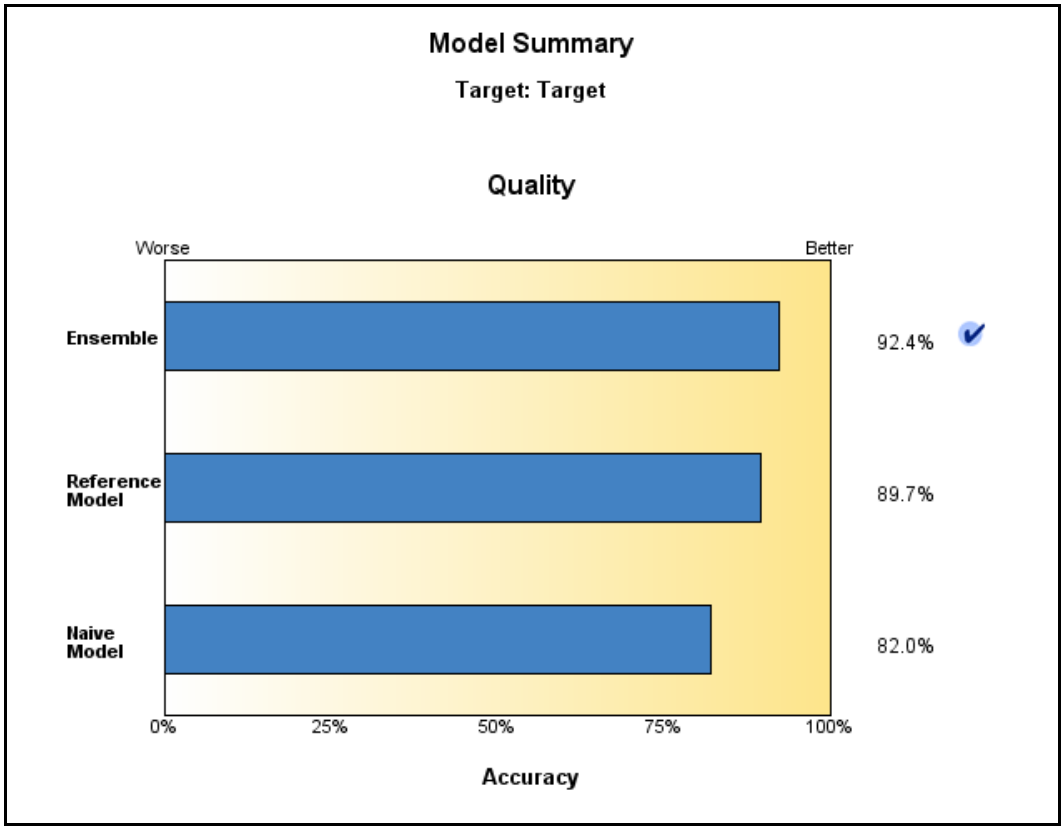


Figure 5-2 C&R Tree Model with Boosting – Accuracy comparison

Figure 5-3 shows the list of features used by the C&R tree model ensemble and how frequently the feature was used by a component model. A total of 22 features considered by the model ensemble. The large number of features used by the model, and the high overall accuracy results, supports the assumption that inherent IP packet attributes can be used to classify active and passive network traffic or network traffic in general.

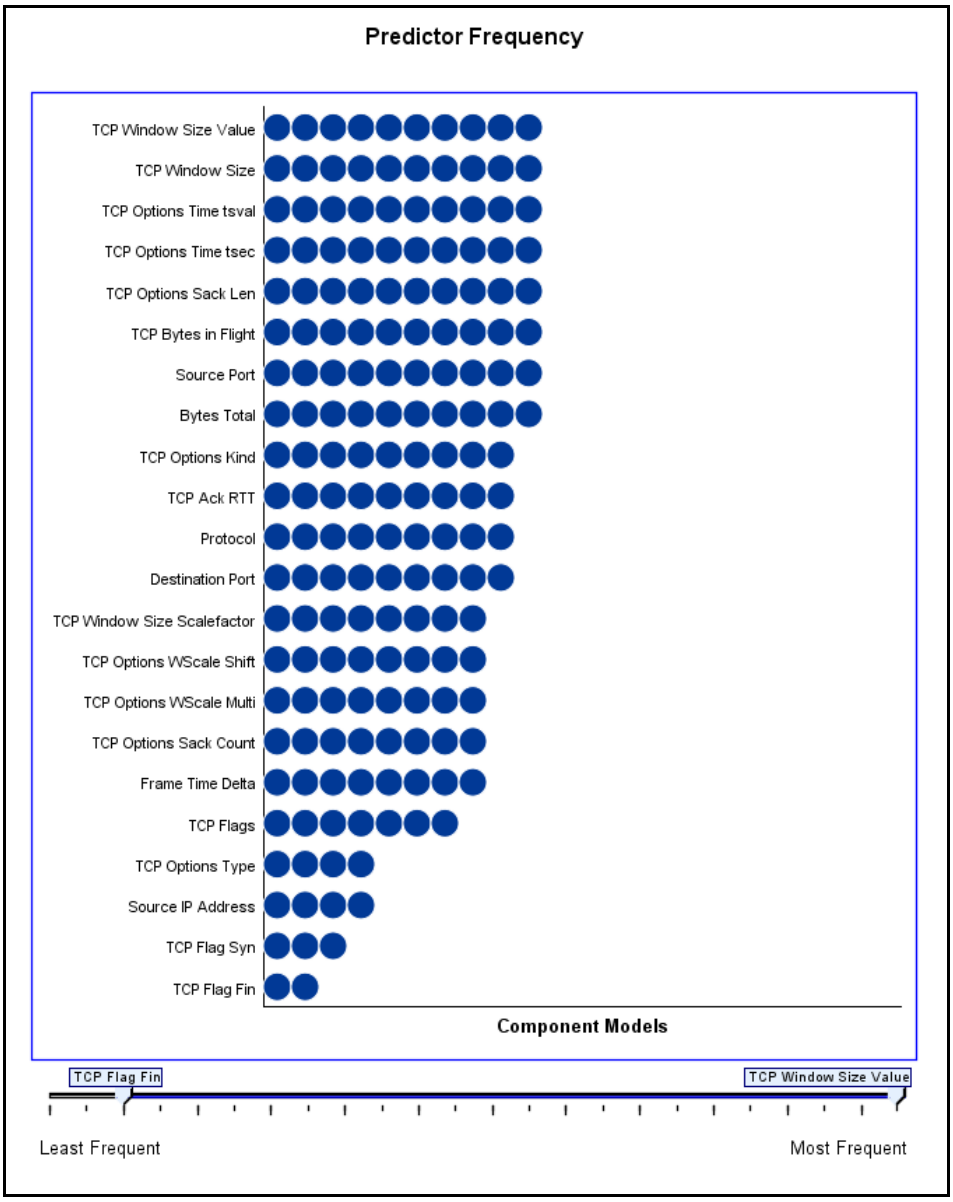


Figure 5-3 : C&R Tree Features and Frequency

5.2.3.3 CHAID

The CHAID model was evaluated in 3rd position in round 1. However, there was a significant improvement in PPP from 50.2% to 84.8% in round 2 with model boosting enabled. APP increased slightly from 94.6% to 97.9%. Overall the combination of PPP and APP were highest for the CHAID implementation with boosting.

Figure 5.4 shows the overall accuracy improvement in the CHAID model with and without boosting enabled.

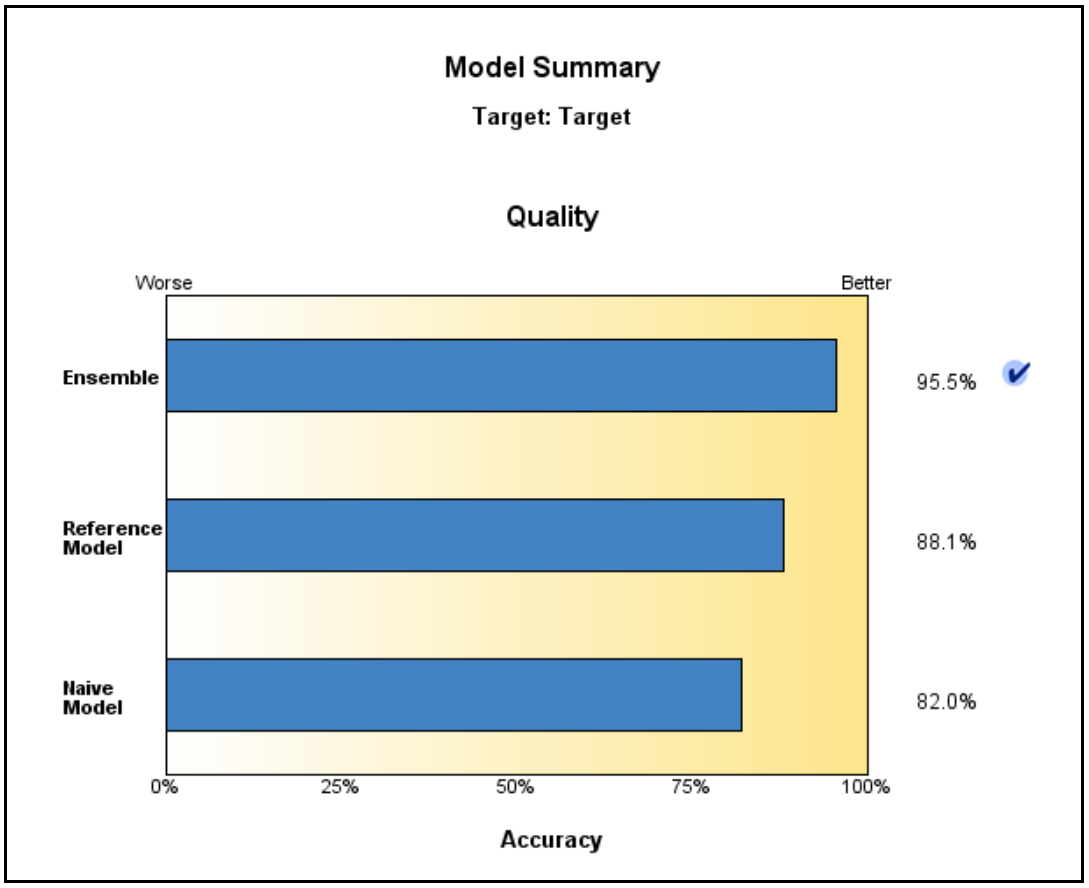


Figure 5-4 : CHAID Model with Boosting – Accuracy comparison

Figure 5-5 shows the list of features used by the CHAID model ensemble and how frequently the feature was used by a component model. A total of 16 features were utilised by the model ensemble. Similar to the C&R model, the large number of features used by the model, and the high overall accuracy results, supports the assumption that inherent IP packet attributes can be used to classify active and passive network traffic or network traffic in general.

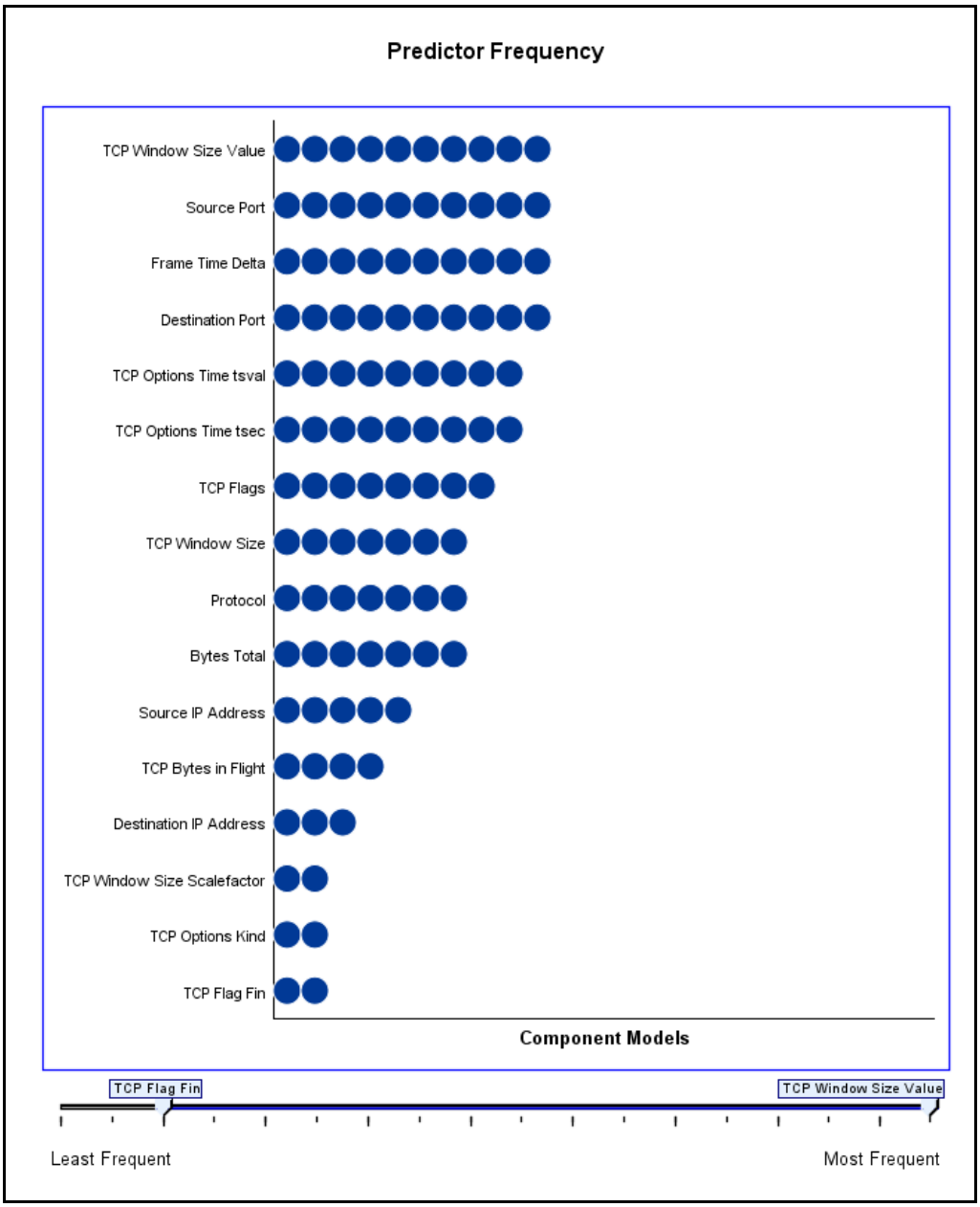


Figure 5-5 : C&R Tree Features and Frequency

5.2.3.4 Round 2 Evaluation

Implementing model boosting led to a significant improvement in PPP in the two of the three decision tree models. There was no change in PPP in the C5.0 model with boosting which was unexpected. The CHAID tree model showed the largest improvement in PPP. APP was already very high in standard model implementation and remained high or increased slightly in the models with boosting. The models with boosting utilised up to 22 features which supports the assumption that inherent IP packet features can be used to classify network traffic.

5.2.4 Real Time Evaluation Criteria

Two real time evaluation metrics were considered; Build Time and Classification Time. Classification time is particularly important because in real world networks the traffic volume will be very high and fast classification is critical.

The best performing model based on PPP and APP metrics was the CHAID model with boosting. This model can classify 400 records per second[11]. However the C5 Tree model without boosting can classify 800 records per second. The CHAID model with boosting uses a model ensemble which is slower to classify than a single tree. Also, the CHAID model ensemble has a larger feature space which can increase classification time.

The CHAID model with boosting has a PPP of 84.8%. The C5 Tree model without boosting has a PPP of 75.8%. In a real work network situation, the absolute difference in accuracy of 9% would probably be sacrificed for an algorithm that can run twice as fast.

5.3 Discussion

This section will discuss the key findings from the research.

Overall, the best performing model based on a combination of PPP and APP was the CHAID tree with boosting. The model had a PPP vale 84.8% and APP value of 97.9%. CHAID trees can generate non-binary trees, meaning that some splits have more than two branches Unlike the C&R trees for example. CHAID tends to create a wider tree than the binary growing methods. CHAID trees can work for all types of inputs.

The research found that decision trees algorithms have the highest accuracy for network traffic classification problems. This finding is backed up by previous research outlined in the literature review.

[11] Based on running the algorithm on PC with system specification as outlined in section 3.2.2.1

The research implemented and evaluated three decision tree algorithms that were not observed in the literature review. C5.0, CHAID Tree and C&R tree were each. Each model generally performed well after a full evaluation of round 1 and round 2 model building implementations. APP was consistently high in round 1 and round 2. PPP was low with standard implementation but improved with boosting.

Model boosting produce significant PPP improvement in the CHAID tree and C&R tree models. For the CHAID tree with boosting, PPP changed significantly from 50.2% to 84.8%. There was no chance in C5.0 with boosting which was unexpected

Using the inherent IP packet attributes worked well. The high PPP figures produced by the models backed up assumption that the IP packet attributes can be used to classify network traffic. Some of the IP packet attributes that worked well were, Time frame delta, Bytes total, Bytes in flight and TCP window size

5.4 Strengths and Limitation of Results

This section will outline the strengths and limitation of the results and key findings.

Results and key finding strengths:

- Results suggest that decision tress were the most accurate ML model to use for the network classification problems. This finding was backed up by research covered in the literature review.
- The research presents the results for three decision tree algorithms not previous covered in the literature.
- The results suggest the classification of low level traffic (active or passive) with a very high degree of accuracy. Previous research has focused on much higher level traffic such as traffic class types like email or FTP.
- The results highlight the need for specific evaluation metrics for low level traffic classification, namely PPP and APP
- The large number of features used by the model, and the high overall accuracy results, supports the assumption that inherent IP packet attributes can be used to classify active and passive network traffic or network traffic in general.

Results and key finding limitations:

- Overall limitation is that the data the results are based on relates to a single device and a single data trace.
- The results outline new features using the inherent IP packet attributes. Most of the features used in the models have not been covered by previous research so there is no knowledge base to compare to.
- The results outline three implementation of decision trees, C5.0, CHAID and C&R. Although C4.5 models have been covered by previous research, there is no knowledge base to compare to for the CHAID and C&R tree models.
- Previous literature documented SVM and Naïve Bayes Estimators models that had a high degree of accuracy in network traffic classification problems. These models were found to have low accuracy in this research.
- The implementation of the C5.0 model with boosting showed no accuracy improvement against the standard implementation of the model. This was an unexpected finding.

5.5 Conclusion

Using data traces captured on a mobile device, this study designed and implemented an experiment to answer a specific research question

"Can passive mobile app traffic be identified using machine learning techniques?"

The findings indicate that passive traffic can be classified using decision tree algorithms with an accuracy up to 84.8%. Using the inherent attributes of IP packets to create features worked well. The IP packets attributes produced predictive features that could be utilised by the ML models to produce a high model accuracy.

The findings demonstrate that the standard implementation of the three decision tree models had a high overall accuracy but a low PPP. Boosting implementations of each model led to significant increase in PPP apart from C5.0 model.

Although the findings show that boosting models produce the most accurate classification models, it is unlikely the boosting models would be deployed on a real world network due to the slow classification time.

6 CONCLUSION

6.1 Introduction

This chapter revisits the overall aim and objectives of this research study. A summary of the key findings is presented. The contribution of this research to the existing body of knowledge are discussed. The limitation of the research are then explained. Finally, areas for further research specifically related to this research are also outlined.

6.2 Problem Definition & Research Overview

The research process followed a logical set of steps with each step informing the subsequent steps. Figure 6-1 shows the steps involved in the research process and how the output of each step cascaded down as input to the next step.

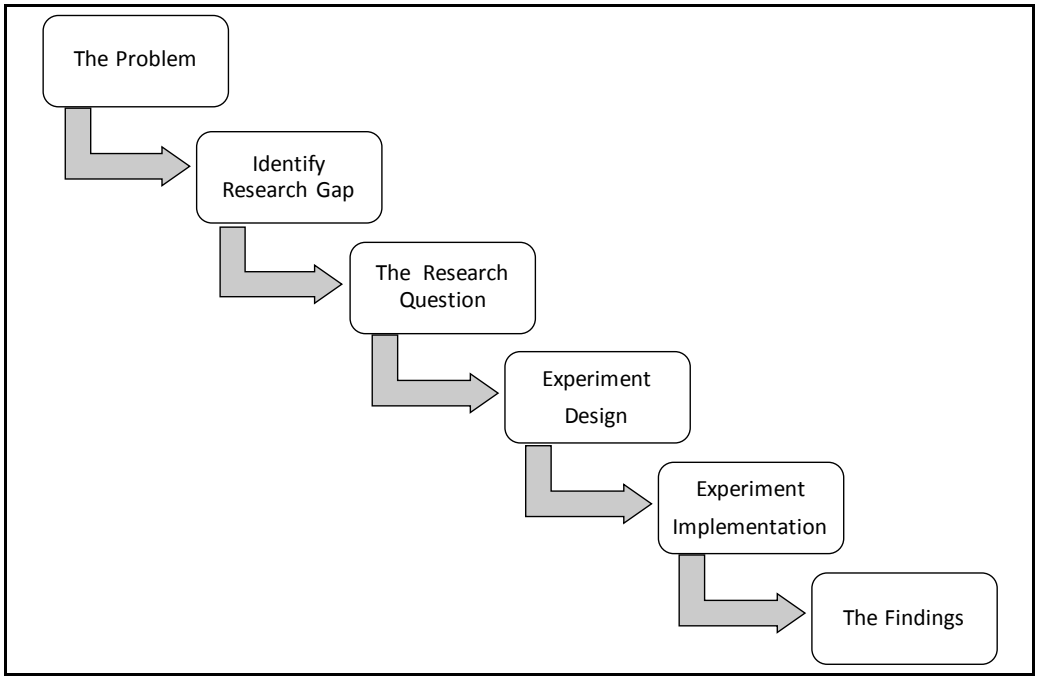


Figure 6-1 : Research Process Steps - Summary Diagram

6.2.1 The Research Problem

The research problem was concerned with the area of network traffic classification. Network traffic classification informs key actions for network operations in real-world network. There is a significant knowledge base in the research area.

This research focused a specific network traffic classification problem. Mobile phone applications (apps) can generate background traffic when the end-user is not actively using the app. If this background traffic could be accurately identified, network operators could de-prioritise the traffic and free up network bandwidth for priority network traffic. The background app traffic should have IP packet features that could be utilised by a machine learning algorithm to identify app-generated (passive) traffic as opposed to user-generated (active) traffic.

6.2.2 The Research Gap

Based on a review of the body of knowledge on network traffic categorisation, a number of research gaps were identified:

- To the best of this author's knowledge, there is no research into classifying user generated (active) versus app generated (passive) traffic.
  This project will investigate this previously unexamined research area.
- Vast majority of research is based on a fixed network IP traffic generated by personal computers.
  This research is based on mobile device originating network traffic which is an important area of research
- Previous research does not take advantage of the large amount of inherent IP packet features. Instead, previous research had added extra steps, complexity and processing calculating new features.
  This project will leverage the large amount of inherent IP packets features to allow a machine learning algorithm to successful identify user generated or app generated traffic

6.2.3 The Research Question

Informed by the gaps identified in the body of knowledge a research question was designed with a focus on classifying low level mobile device network data. Using data traces captured on a mobile device, this study designed and implemented an experiment to answer a specific research question

"Can passive mobile app traffic be identified using machine learning techniques?"

Guided by the research question, the project objectives are:

1. Gain knowledge in the research domain of network traffic classification
2. Design research question experiment solution
3. Implement the experiment solution and capture results
4. Evaluate outcomes from experiment implementation

6.2.4 Summary of the Experiment Design

An innovative experiment setup was designed in order to attempt to answer the research question. A mobile phone running Android OS was configured to capture network data based on. Three specific data trace procedures where then designed to capture active and passive app traffic.

Feature generation in previous researched recommend computing new features based on IP packet data. This research designed a different approach. Feature generation was enabled by exposing inherent IP packet attributes.

6.2.5 Summary of the Experiment Implementation

The experiment processes was initiated by collecting network traffic data generated by apps on an Android OS mobile phone device. The captured network traffic data was then examined and features were created based on the properties of the IP packets. The output files containing all the relevant IP packet features were passed to statistical modelling software.

Multiple machine learning techniques were then applied to the data. Decision tree algorithms were found to have the highest model accuracy for classifying active and passive traffic. Three decision tree models were built; C5.0, C&R Tree and CHAID.

The performance of each machine learning model was evaluated based on the evaluation criteria identified in the literature review. After the first round of ML modelling the overall accuracy of the models was found to be high, ranging from 88.3% to 95.6%. However, passive traffic prediction accuracy was found to be low at 50.3%. A second round of modelling was implemented using model boosting with the same three decision tree models. Significant improvement was observed in the accuracy in predicting passive traffic.

6.2.6 Summary of Findings and Conclusions

The findings indicate that passive app network traffic can be classified with an accuracy up to 84.8% using a CHAID decision tree algorithm with model boosting enabled.

The experiment implementation, as specified in the experiment design, proved to be a very good process to test the research question. The data capture process captured clean and useful data. The different software elements were well chosen and there were no integration issues. Using the inherent IP packet attributes as input features also worked well and findings backed up assumption that the IP packet attributes could have a high predictive value to ML models.

The models produced by the research had high accuracy and PPP values. This research found that decision trees algorithms have the highest accuracy for network traffic classification problems. This finding is backed up by previous research outlined in the literature review. New decision tree algorithms were investigated.

Summary of key findings from model build phase:

- APP was high across all modes in round 1 and round 2
- PPP was low in round 1 but improved significantly in round 2
- PPP changed significantly for CHAID with boosting from 50.2% to 84.8%

- No chance in c5.0 with boosting which was unexpected
- In the real word the standard implementation of the C5.0 tree would probably be deployed due to acceptable PPP value and fast classification time.

6.3 Contributions to the Body of Knowledge

This dissertation makes a practical contribution to the knowledge base in the following ways:

- Extends the limited knowledge base of mobile device network traffic classification.
- New research in low level network traffic classification. Active v passive traffic classification rather than high level traffic class types.
- Innovative experimental design
    - Robust, end-to-end experiment process design for mobile device data capture, analysis and modelling
    - 3 specifically designed data traces to capture active and passive network traffic from mobile device.
- New approaches proposed for feature generation by leveraging the large number of IP packet attributes. No computation needed. New features such as Frame Time Delta, Bytes in flight and TCP window size shown to be significant predictors of active and passive traffic
- Extend the knowledge of machine learning techniques used network traffic classification by developing models using C&R and CHAID decision trees.
- Extend research in using model boosting for network traffic classification.
- New evaluation metrics proposed for active and passive model accuracy; Passive Predictive Precision and Active Predictive Precision.

6.4 Experimentation, Evaluation and Limitation

There are a number of limitations with the research that should be considered:

Research Limitations:

- Device
    - Only a single device was used to capture data. This means that it is easy to relate traffic to a specific device. Using multiple devices to create network traffic would deliver more robust research.
    - Only Android OS considered. Apple IOS has a similar market share to Android and can reasonably be expected to generate large volume of network traffic.
- Data Trace
    - Only one dataset was created during an hour on a single day. Having more data sets, from different times, would allow for comparison of the two separately gathered data set. Having multiple data sets will also allow for testing and comparison of the machine learning models. This should reduce the risk of incorrect findings that may occur when only one data set was used.
- Mobile Apps
    - Only 11 specific apps were considered.
    - The data trace captured app generated traffic at one point in time. App network architectures can change over time. Popular apps can make use of Content Delivery Networks (CDNs). These CDNs that can change architecture frequently.
    - Any model would need to be checked regularly with new data traces.
- App actions:
    - The data traces captured a limited number of app actions, opening the app, downloading content, sending message. This may not be a true reflection of how apps are used on real word networks.
- Encrypted traffic:
    - Secure, encrypted traffic is consistently increasing on networks. It is unknown what impact an increase in encrypted traffic will have on this

research but it could impact the number of IP attributes available for modelling

Research Strengths:

- IP classification of mobile app generated traffic
- Low level classification of active and passive app traffic
- Innovative and original experiment design
- New decision tree models developed as part of this research
- Research demonstrates the predictive power of inherent IP packet attributes.
- The large number of features used by the model, and the high overall accuracy results, supports the assumption that inherent IP packet attributes can be used to classify active and passive network traffic or network traffic in general.

6.5 Future Work & Research

This section outlines how this research could be extended and identifies areas for future research.

- Data Traces
    - Future research could create multiple data traces could be created and used to build models to fully test the robustness of the model output
    - Data traces could be run at different time of day
    - Increase number of apps considered
    - Include more mobile phone OS platforms such as Apple IOS
- Feature generation
    - Future research could explore more of the Wireshark display filters available (over 174000 fields in 1000 protocols as of version 1.12.3)
- Extend to real word network with real-time testing
    - Future research could test if active and passive network traffic classification can be deployed on a real work network.
    - Consider computational speed issue on real world network such as System Throughput: (Li and Moore, 2007) and System Latency: (Li and Moore, 2007) metric proposed in the literature

6.6 Conclusion

This research attempted to answer a very specific research question - Can passive mobile app traffic be identified using machine learning techniques?

Guided by the research question, the project objectives were:

1. Gain knowledge in the research domain of network traffic classification
2. Design research question experiment solution
3. Implement the experiment solution and capture results
4. Evaluate outcomes from experiment implementation

By implementing an innovative experiment design the findings indicate that passive app network traffic can be classified with an accuracy up to 84.8% using a CHAID decision tree algorithm with model boosting enabled. At the same time the project objectives have been achieved.

This dissertation makes a practical contribution to the knowledge base to help inform practitioners.

Future work recommendations have been documented to help shape the direction of future research.

BIBLIOGRAPHY

Auld, T., Moore, A., Gull, S.F., 2007. Bayesian Neural Networks for Internet Traffic Classification. IEEE Trans. Neural Netw. 18, 223–239. doi:10.1109/TNN.2006.883010

Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A., Salamatian, K., 2006. Traffic Classification on the Fly. SIGCOMM Comput Commun Rev 36, 23–26. doi:10.1145/1129582.1129589

Callado, A., Kamienski, C., Szabo, G., Gero, B., Kelner, J., Fernandes, S., Sadok, D., 2009. A Survey on Internet Traffic Identification. IEEE Commun. Surv. Tutor. 11, 37–52. doi:10.1109/SURV.2009.090304

Erman, J., Arlitt, M., Mahanti, A., 2006. Traffic Classification Using Clustering Algorithms, in: Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data, MineNet '06. ACM, New York, NY, USA, pp. 281–286. doi:10.1145/1162678.1162679

Fahad, A., Tari, Z., Khalil, I., Habib, I., Alnuweiri, H., 2013. Toward an efficient and scalable feature selection approach for internet traffic classification. Comput. Netw. 57, 2040–2057. doi:10.1016/j.comnet.2013.04.005

Frank, J., 1994. Artificial intelligence and intrusion detection: Current and future directions, in: Proceedings of the 17th National Computer Security Conference. Presented at the The 17th national computer security conference.

Karagiannis, T., Papagiannaki, K., Faloutsos, M., 2005. BLINC: Multilevel Traffic Classification in the Dark, in: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '05. ACM, New York, NY, USA, pp. 229–240. doi:10.1145/1080091.1080119

Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., Lee, K., 2008. Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices, in: Proceedings of the 2008 ACM CoNEXT Conference, CoNEXT '08. ACM, New York, NY, USA, pp. 11:1–11:12. doi:10.1145/1544012.1544023

Klein, P.F., Chung, L.B., 2006. Intelligent pre-caching on a network. Google Patents.

Li, W., Moore, A., 2007. A Machine Learning Approach for Efficient Traffic Classification, in: 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS '07. Presented at the 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS '07, pp. 310–317. doi:10.1109/MASCOTS.2007.2

Moore, A.W., Papagiannaki, K., 2005. Toward the Accurate Identification of Network Applications, in: Dovrolis, C. (Ed.), Passive and Active Network Measurement, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 41–54.

Moore, A.W., Zuev, D., 2005. Internet Traffic Classification Using Bayesian Analysis Techniques, in: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '05. ACM, New York, NY, USA, pp. 50–60. doi:10.1145/1064212.1064220

Nguyen, T.T.T., Armitage, G., 2008. A survey of techniques for internet traffic classification using machine learning. IEEE Commun. Surv. Tutor. 10, 56–76. doi:10.1109/SURV.2008.080406

Singh, K., Agrawal, S., Sohi, B.S., 2013. A Near Real-time IP Traffic Classification Using Machine Learning. Int. J. Intell. Syst. Appl. 5, 83–93. doi:10.5815/ijisa.2013.03.09

Williams, N., Zander, S., Armitage, G., 2006. A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. SIGCOMM Comput Commun Rev 36, 5–16. doi:10.1145/1163593.1163596

Yuan, R., Li, Z., Guan, X., Xu, L., 2010. An SVM-based machine learning method for accurate internet traffic classification. Inf. Syst. Front. 12, 149–156. doi:10.1007/s10796-008-9131-2

Zander, S., Nguyen, T., Armitage, G., 2005. Automated traffic classification and application identification using machine learning, in: The IEEE Conference on Local Computer Networks, 2005. 30th Anniversary. Presented at the The IEEE Conference on Local Computer Networks, 2005. 30th Anniversary, pp. 250–257. doi:10.1109/LCN.2005.35

Zhang, J., Chen, C., Xiang, Y., Zhou, W., Xiang, Y., 2013a. Internet Traffic Classification by Aggregating Correlated Naive Bayes Predictions. IEEE Trans. Inf. Forensics Secur. 8, 5–15. doi:10.1109/TIFS.2012.2223675

Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y., Guan, Y., 2013b. Network Traffic Classification Using Correlation Information. IEEE Trans. Parallel Distrib. Syst. 24, 104–117. doi:10.1109/TPDS.2012.98

Zuev, D., Moore, A.W., 2005. Traffic Classification Using a Statistical Approach, in: Dovrolis, C. (Ed.), Passive and Active Network Measurement, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 321–324.

APPENDIX A – LITERATURE REVIEW SUMMARY TABLE

Summary of key points from papers in literature review:

Author	Packet or Flow Based	Payload Inspection	Real-Time Consideration	Network Layer	Supervised (Classification) or Unsupervised (Clustering)	Supervised Algorithms Used	Unsupervised Algorithms Used	Best performing Algorithm	Accuracy Metrics	Accuracy
(Frank, 1994)	Packet header statistical features	No	**Yes**	Transport Layer	Supervised	Decision Tree Neural Net k-NN		na	Classification Error	95%
(Karagiannis et al., 2005)	~	~	**Yes**	Transport Layer	~	~	~	~	Correctly labelled Traffic (Accuracy) Completeness	95%
(Moore and Papagiannaki, 2005)	Packet header statistical features	Full payload inspection	No	Transport Layer	Supervised	Authors developed a content-based classification process	~	Content based classification	Packets correctly identified	approaching 100%
(Moore and Zuev, 2005)	Packet header statistical features	No	No	Transport Layer	Supervised	Naïve Bayes Estimator	~	Naïve Bayes Estimator	Average percentage of accurately classified flows	> 95%
(Zander et al., 2005)	Flow statistical properties	No	No	Application	Unsupervised	~	AutoClass unsupervised Bayesian classifier	AutoClass	Author defined metric termed intra-class homogeneity	86.50%
(Zuev and Moore, 2005)	Flow statistical properties	No	No	Transport Layer	Supervised	Naïve Bayes Estimator	~	Naïve Bayes Estimator	Average percentage of accurately classified flows	83%
(Bernaille et al., 2006)	Packet header data (First 5 packets of a flow)	No	**Yes**	Transport Layer	Unsupervised	~	K-Means	K-Means	Average percentage of accurately classified flows	84.2% to 96.92% (Accuracy by Application)
Erman et al., 2006)	Flow statistical properties	No	No	Transport Layer	Unsupervised	~	Compares K-Means and DBSCAN with previously used AutoClass.	AutoClass	Overall Accuracy	97.60%
(Williams et al., 2006)	Flow statistical properties	No	**Yes**	Transport Layer	Supervised	Naïve Bayes Discretisation Naïve Bayes Kernel density estimation Decision Tree C4.5 Bayesian Network Naïve Bayes Tree	~	**Decision Tree C4.5 (Speed)**	Accuracy Precision Recall & Speed	94.13%
(Auld et al., 2007)	Packet header statistical features	No	No	Transport Layer	Supervised	Bayesian trained neural network Naïve Bayesian classifier.	~	Bayesian trained neural network	Average percentage of accurately classified flows	95-99%
(Li and Moore, 2007)	Packet header statistical features	No	**Yes**	Transport Layer	Supervised	Decision Tree C4.5	~	**Decision Tree C4.5**	Accuracy Precision Recall & Latency and Throughput of the ML system	99.80%
(Kim et al., 2008)	Packet header statistical features	No	No	Transport Layer	Supervised	SVM Neural Net k-NN	~	**SVM**	Accuracy Precision Recall & F-Measure	98%
(Yuan et al., 2010)	Network flow parameter obtains from the packet headers	No	**Yes**	Transport Layer	Supervised	SVM	~	**SVM**	Average percentage of accurately classified flows	99.42%
(Singh et al., 2013)	Packet header statistical features	No	**Yes**	Transport Layer	Supervised	Multilayer Perceptron (MLP) Radial Basis Function Neural Network (RBF) C 4.5 Decision Tree Algorithm Bayes Net Algorithm Naïve Bayes Algorithm	~	Bayes Net (*Near real time classification)	Classification Accuracy Recall Precision Training time Number of features used Packet capture duration	91.80%
(Zhang et al., 2013a)	flow statistical features	No	No	Transport Layer	Supervised	Naïve Bayes Algorithm	~	Naïve Bayes Algorithm	Classification Accuracy F-Measure	89.00%
(Zhang et al., 2013b)	flow statistical features	No	No	Transport Layer	Supervised	k-NN	~	k-NN		>90%

APPENDIX B – DATA DICTIONARY

Data dictionary for output from experiment implementation. .

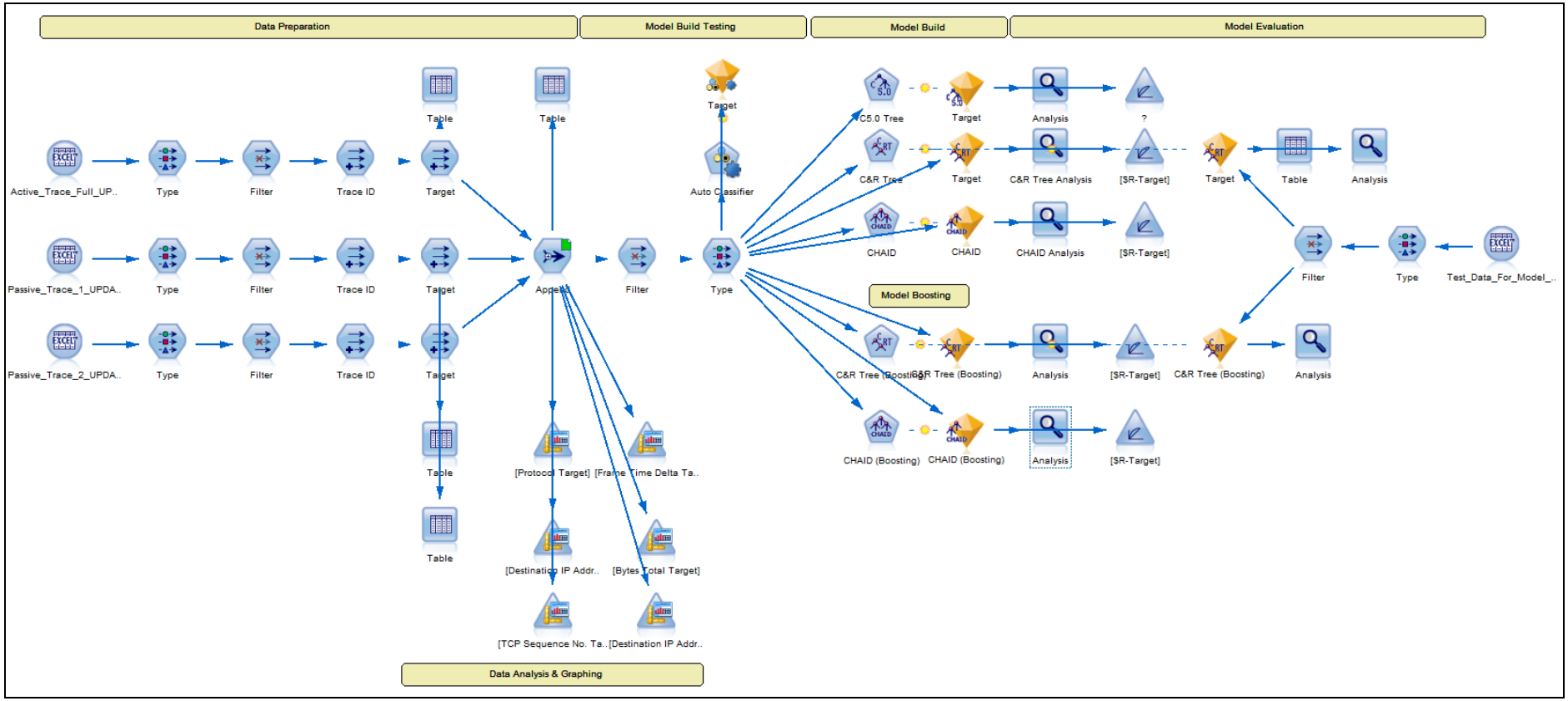
Feature Name	Example	Data Type	Wireshark Attribute	Description
IP Frame Number	10	integer	frame.number	Number to identify each frame (sequential)
TCP Flow No.	3	integer	tcp.stream	Number to identify each TCP flow (sequential)
TCP Sequence No.	28	integer	tcp.seq	TCP sequence number used to keep track of how much data has been sent.
Protocol	TCP	string	protocol	The protocol of the network frame
Time	12.17324	seconds	frame.time_relative	Time since reference or first frame
Bytes Total	1484	integer	frame.len	Frame length on the wire in bytes
Source IP Address	192.168.1.14	string	source.address	Source IP address of the client
Source Port	38346	integer	tcp.srcport	Source port number of the client
Destination IP Address	74.125.24.156	string	destination.address	Destination IP address of the server
Destination Port	443	integer	tcp.dstport	Destination port number of the server
Combined Ports	38346 , 443	string	tcp.port	Source and destination port numbers separated by a comma
Info	Server Hello	string	information	Frame information string
TCP Flag Syn	Set	string	tcp.flags.syn	Flag to indicate if TCP Syn was set
TCP Flag Ack	Set	string	tcp.flags.ack	Flag to indicate if TCP Ack was set
TCP Flag Fin	Not Set	string	tcp.flags.fin	Flag to indicate if TCP Fin was set
TCP Flags	0x0018	string	tcp.flags	TCP Flags expressed in hexadecimal
DNS Answer Count	5	integer	dns.count.answers	The number of answers in the DNS query

				response
DNS Query Name	www.googleadservices.com	string	dns.qry.name	DNS query name for DNS lookup
DNS IP Addresses	74.125.24.156, 74.125.24.157,	string	dns.a	The IP addresses returned by the DNS server
SSL Handshake Type	Client Hello	string	ssl.handskake.type	Details of the SSL handshake type
Certificate SSL String	*.rovio.com	string	x509sat.uTF8String	Details of the SSL string in the security certificate from the host
Certificate DNS Name	*.g.doubleclick.net	string	x509ce.dNSName	DNS details from the security certificate from the host
Certificate String	DigiCert Inc,www.digicert.com,	string	x509sat.printableString	Details of the security certificate from the host
Host	i.instagram.com	string	http.host	The host string
User Agent	Instagram 6.11.2 Android (16/4.1.1…..	string	http.user_agent	The user agent is the software that has created the network traffic
Referer URI	www.rte.ie/news	string	http.referer	The full referrer URI
Requested URI	/apps/YouTube	string	http.request.uri	The full URI requested
Frame Time Epoch	1417110806.000000	seconds	frame.time_epoch	Epoch time is a system for describing instants in time
Frame Time Delta	0.081276	seconds	frame.time_delta	Time delta from previous captured frame
TCP Window Size	43648	integer	tcp.window_size	Calculated window size
TCP Window Size Value	1024	integer	tcp.window_size_value	Window size value
TCP Window Size S-factor	64	integer	tcp.window_size_scalefactor	Window size scaling factor
TCP Bytes in Flight	122	integer	tcp.analysis.bytes_in_flight	Bytes in flight
TCP Ack RTT	0.0345234	float	tcp.analysis.ack_rtt	The Round Trip Time to ACK the segment

TCP Options	0101080afff	string	tcp.options	TCP Options header section
TCP Options MSS	0	integer	tcp.options.mss	TCP MSS Option
TCP Options Kind	Timestamp	string	tcp.option_kind	TCP Option kind
TCP Option Len	4,2,10,3	string	tcp.option_len	TCP Option length
TCP Options Sack Perm	True	String	tcp.options.sack_perm	TCP SACK Permitted Option
TCP Options Sack Count	3	integer	tcp.options.sack.count	TCP SACK Count
TCP Options Sack Len	10075338	Long Int	tcp.options.sack_le	TCP SACK Left Edge
TCP Options Time tsec	167128	integer	tcp.options.timestamp.tsecr	Timestamp echo reply
TCP Options Time tsval	687677493	Long Int	tcp.options.timestamp.tsval	Timestamp value
TCP Options Type	1	integer	tcp.options.type	TCP Option type
TCP Options Type Class	control	string	tcp.options.type.class	Class
TCP Options Type Number	No-Operation	string	tcp.options.type.number	TCP Type number
TCP Options WScale Shift	9	integer	tcp.options.wscale.shift	TCP window scale option shirt count
TCP Options WScale Multi	512	integer	tcp.options.wscale.multiplier	TCPwindow scale option multiplier
TCP Window Size	43648	integer	tcp.window_size	Calculated window size

APPENDIX C - SPSS STREAM

Overview of stream built using IBM SPSS software.



APPENDIX D – C5.0 TREE MODEL SETUP

Standard Implementation: Model Setup

Analysis

- Tree depth: 21
- Cross Validation
  - Mean: 99.0
  - Standard Error: 0.0
- Analysis of Append (23-Feb-2015 22:50:58)
  - Number of records: 157,066
  - Analysis Accuracy: 95.566%

Fields

- Target
  - Target
- Inputs
  - Frame Time Delta
  - TCP Window Size Value
  - Bytes Total
  - TCP Options Time tsec
  - TCP Options Time tsval
  - Protocol
  - Destination Port
  - Source Port
  - TCP Window Size Scalefactor
  - TCP Window Size
  - TCP Ack RTT
  - TCP Bytes in Flight
  - TCP Options Type Class
  - TCP Options WScale Shift

Build Settings

- Use partitioned data: false
- Calculate predictor importance: true
- Calculate raw propensity scores: true
- Calculate adjusted propensity scores: false

Use weight: false

Output type: Decision tree

Group symbolics: false

Use boosting: false

Cross-validate: true

Number of folds: 10

Mode: Expert

Pruning severity: 75

Minimum records per child branch: 2

Winnow attributes: false

Use global pruning: true

Use misclassification costs: false

Training Summary

Algorithm: C5

Model type: Classification

Stream: C:\Users\IBM_ADMIN\Desktop\Dissertation\SPSS Models\Model 2 - Packets\Stream5 - Packets All Data.str

User: IEI76422

Date built: 03/03/15 22:50

Application: IBM® SPSS® Modeler 15

Elapsed time for model build: 0 hours, 0 mins, 54 secs

Boosting Implementation: Model Setup

Fields

Target

Target

Predictors(Inputs)

Use partitioned data: false

Build Options

Objectives

What is your main objective?: Enhance model accuracy (boosting)

Basics

Maximum Tree Depth: 5

Prune tree to avoid overfitting: true

Maximum difference in risk (in standard Errors): 1.0

Maximum surrogates: 5

Stopping Rules

What to use for node size requirement: Use percentage

Minimum records in parent branch(%): 2.0

Minimum records in child branch(%): 1.0

Minimum records in parent branch: 100

Minimum records in child branch: 50

Costs & Priors

Use misclassification costs: false

How to use priors: Based on training data

Adjust priors using misclassification costs: false

Ensemble

Number of component models for boosting and/or bagging: 10

Advanced

Overfit prevention set(%): 30.0

Replicate Results: true

Random seed: 681644031

Significance level for splitting: 0.05

Significance level fro merging: 0.05

Adjust significance values using Bonferroni method: true

Allow resplitting of merged categories within a node: false

Chi-Square for categorical targets: Pearson

Minimum change in expected cell frequencies: 0.0010

Maximum iterations for convergence: 100

Minimum change in impurity: 1.0E-4

Impurity measure for categoriacl targets: Gini

Training Summary

Method: Decision Trees

Records used in training: 1,884,792

Model type: Classification

User: IEI76422

Application: IBM SPSS Modeler Common 15.0.0.0

Date built: 04 March 2015 16:45:28 GMT

Predictors used in model

- TCP Options Sack Len
- TCP Window Size
- TCP Window Size Value
- TCP Options Time tsec
- Destination Port
- TCP Window Size Scalefactor
- TCP Bytes in Flight
- Bytes Total
- TCP Flags
- Source Port
- TCP Options Time tsval
- Frame Time Delta
- TCP Options WScale Shift
- TCP Options WScale Multi
- Protocol
- TCP Ack RTT
- Source IP Address
- TCP Options Sack Count
- TCP Flag Ack
- TCP Flag Fin
- TCP Flag Syn
- TCP Options Kind
- TCP Options Type
- Destination IP Address

APPENDIX E – C&R TREE MODEL SETUP

Standard Implementation: Model Setup

Analysis

- Tree depth: 5

Fields

- Target
  - Target
- Inputs
  - Protocol
  - Bytes Total
  - Source IP Address
  - Source Port
  - Destination Port
  - TCP Flag Fin
  - TCP Flags
  - Frame Time Delta
  - TCP Window Size
  - TCP Window Size Value
  - TCP Window Size Scalefactor
  - TCP Bytes in Flight
  - TCP Ack RTT
  - TCP Options Sack Len
  - TCP Options Time tsec
  - TCP Options Time tsval
  - TCP Options WScale Shift
  - TCP Options WScale Multi

Build Settings

- Use partitioned data: false
- Calculate predictor importance: true
- Calculate raw propensity scores: false
- Calculate adjusted propensity scores: false
- Use frequency: false
- Use weight: false

Levels below root: 5

Mode: Expert

Maximum surrogates: 5

Minimum change in impurity: 0.0

Impurity measure for categorical targets: Gini

Stopping criteria: Use percentage

Minimum records in parent branch (%): 2

Minimum records in child branch (%): 1

Prune tree: true

Use standard error rule: false

Prior probabilities: Based on training data

Adjust priors using misclassification costs: false

Use misclassification costs: false

Training Summary

Algorithm: C&R Tree

Model type: Classification

Stream: C:\Users\IBM_ADMIN\Desktop\Dissertation\SPSS Models\Model 2 - Packets\Stream4 - Packets All Data.str

User: IEI76422

Date built: 04/03/15 16:39

Application: IBM® SPSS® Modeler 15

Elapsed time for model build: 0 hours, 0 mins, 14 secs

Boosting Implementation: Model Setup

Fields

Target

Target

Predictors(Inputs)

Use partitioned data: false

Build Options

Objectives

What is your main objective?: Enhance model accuracy (boosting)

Basics

Maximum Tree Depth: 5

Prune tree to avoid overfitting: true

Maximum difference in risk (in standard Errors): 1.0

Maximum surrogates: 5

Stopping Rules

What to use for node size requirement: Use percentage

Minimum records in parent branch(%): 2.0

Minimum records in child branch(%): 1.0

Minimum records in parent branch: 100

Minimum records in child branch: 50

Costs

Use misclassification costs: false

How to use priors: Based on training data

Adjust priors using misclassification costs: false

Ensemble

Number of component models for boosting and/or bagging: 10

Advanced

Overfit prevention set(%): 30.0

Replicate Results: true

Random seed: 701499504

Significance level for splitting: 0.05

Significance level fro merging: 0.05

Adjust significance values using Bonferroni method: true

Allow resplitting of merged categories within a node: false

Chi-Square for categorical targets: Pearson

Minimum change in expected cell frequencies: 0.0010

Maximum iterations for convergence: 100

Minimum change in impurity: 1.0E-4

Impurity measure for categoriacl targets: Gini

Training Summary

Method: Decision Trees

Records used in training: 1,884,792

Model type: Classification

User: IEI76422

Application: IBM SPSS Modeler Common 15.0.0.0

Date built: 24 February 2015 23:52:43 GMT

Predictors used in model

- Frame Time Delta
- TCP Flag Fin
- TCP Window Size Value
- Source Port
- Destination Port
- TCP Options Kind
- Bytes Total
- TCP Bytes in Flight
- TCP Options Time tsec
- Protocol
- TCP Window Size
- TCP Options Time tsval
- TCP Flags
- Source IP Address
- Destination IP Address
- TCP Window Size Scalefactor
- TCP Options Type Class
- TCP Flag Syn

APPENDIX F – CHAID TREE MODEL SETUP

Standard Implementation: Model Setup

Analysis

- Tree depth: 4
- Analysis of Append (23-Feb-2015 19:31:46)
  - Number of records: 157,066
  - Analysis Accuracy: 88.308%
- Analysis of Append (23-Feb-2015 20:25:42)
  - Number of records: 157,066
  - Analysis Accuracy: 88.308%
- Analysis of Append (23-Feb-2015 20:30:21)
  - Number of records: 157,066
  - Analysis Accuracy: 88.308%

Fields

- Target
  - Target
- Inputs
  - Protocol
  - Bytes Total
  - Source Port
  - Destination Port
  - TCP Flag Fin
  - Frame Time Delta
  - TCP Window Size
  - TCP Window Size Value
  - TCP Bytes in Flight
  - TCP Options Kind
  - TCP Options Time tsec

Build Settings

- Use partitioned data: false
- Calculate predictor importance: true
- Calculate raw propensity scores: false
- Calculate adjusted propensity scores: false

Continue training existing model: false

Use frequency: false

Use weight: false

Levels below root: 5

Alpha for Splitting: 0.05

Alpha for Merging: 0.05

Epsilon For Convergence: 0.001

Maximum iterations for convergence: 100

Use Bonferroni adjustment: true

Allow splitting of merged categories: false

Chi-Square method: Pearson

Stopping criteria: Use percentage

Minimum records in parent branch (%): 2

Minimum records in child branch (%): 1

Use misclassification costs: false

Training Summary

Algorithm: CHAID

Model type: Classification

Stream: C:\Users\IBM_ADMIN\Desktop\Dissertation\SPSS Models\Model 2 - Packets\Stream4 - Packets All Data.str

User: IEI76422

Date built: 25/02/15 13:51

Application: IBM® SPSS® Modeler 15

Elapsed time for model build: 0 hours, 0 mins, 14 secs

Boosting Implementation: Model Setup

Analysis

Tree depth: 40

Cross Validation

Mean: 99.3

Standard Error: 0.0

Fields

Target

Target

Inputs

Frame Time Delta

TCP Window Size Value

Bytes Total

TCP Options Time tsec

TCP Options Time tsval

Protocol

Destination Port

Source Port

TCP Window Size Scalefactor

TCP Window Size

TCP Ack RTT

TCP Bytes in Flight

TCP Options Type Class

TCP Options WScale Shift

TCP Options Kind

TCP Options WScale Multi

TCP Options Sack Len

TCP Options Type

Build Settings

Use partitioned data: false

Calculate predictor importance: true

Calculate raw propensity scores: true

Calculate adjusted propensity scores: false

Use weight: false

Output type: Decision tree

Group symbolics: false

Use boosting: true

Number of trials: 10

Cross-validate: true

Number of folds: 10

Mode: Expert

Pruning severity: 75

Minimum records per child branch: 2

Winnow attributes: false

Use global pruning: true

Use misclassification costs: false

Training Summary

Algorithm: C5

Model type: Classification

Stream: C:\Users\IBM_ADMIN\Desktop\Dissertation\SPSS Models\Model 2 - Packets\Stream5 - Packets All Data.str

User: IEI76422

Date built: 04/03/15 15:57

Application: IBM® SPSS® Modeler 15

Elapsed time for model build: 0 hours, 12 mins, 46 secs

APPENDIX G – GAIN CHARTS

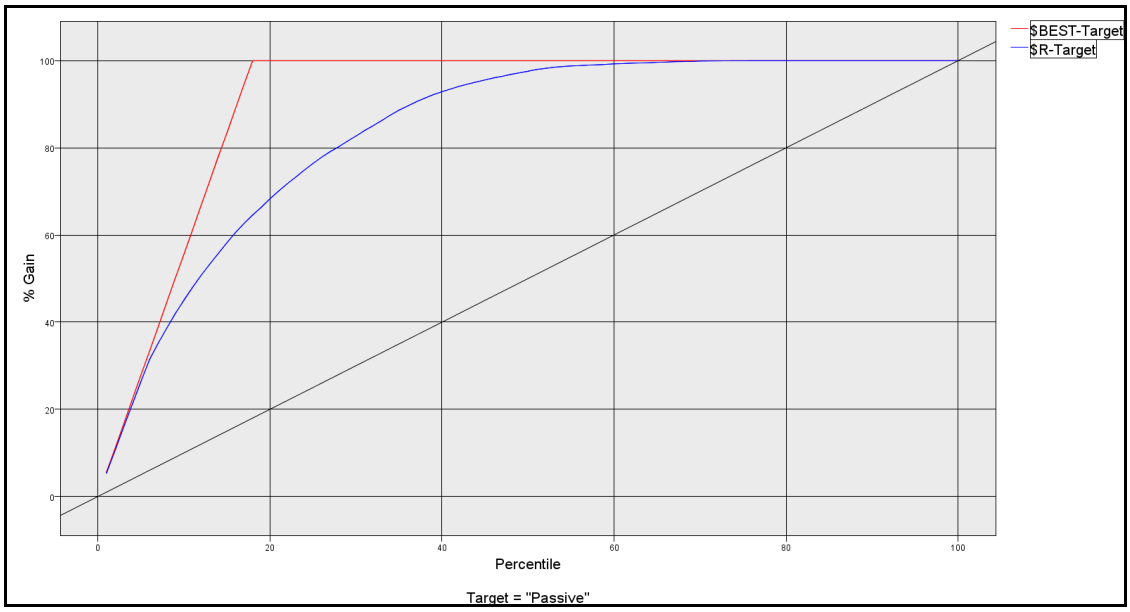


Figure 6-2 : CHAID Tree Gain Chart

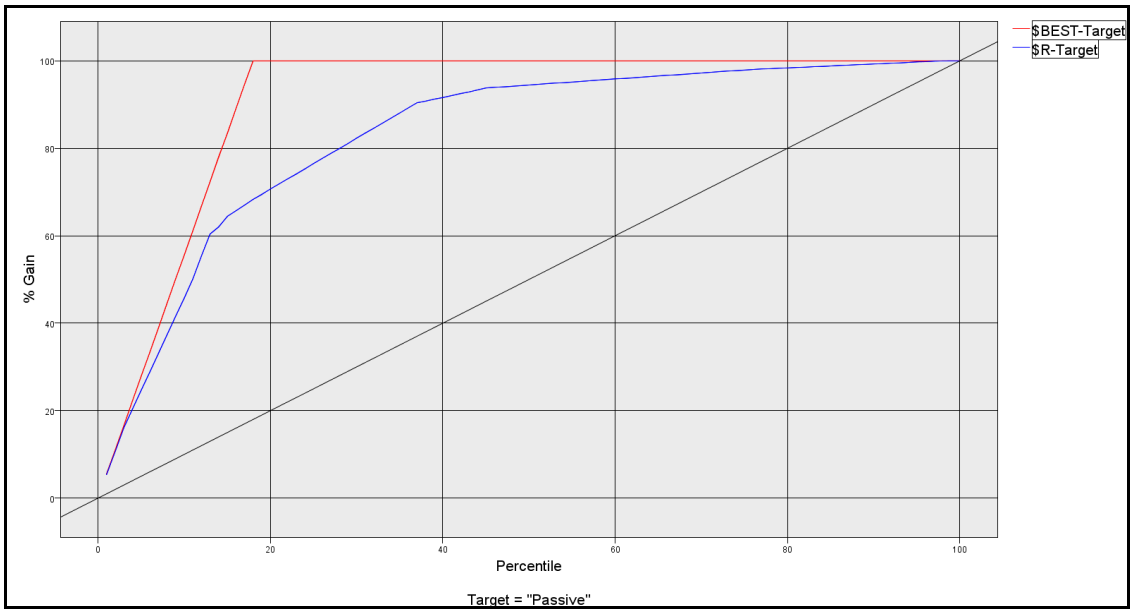


Figure 6-3 : C&R Tree Gain Chart

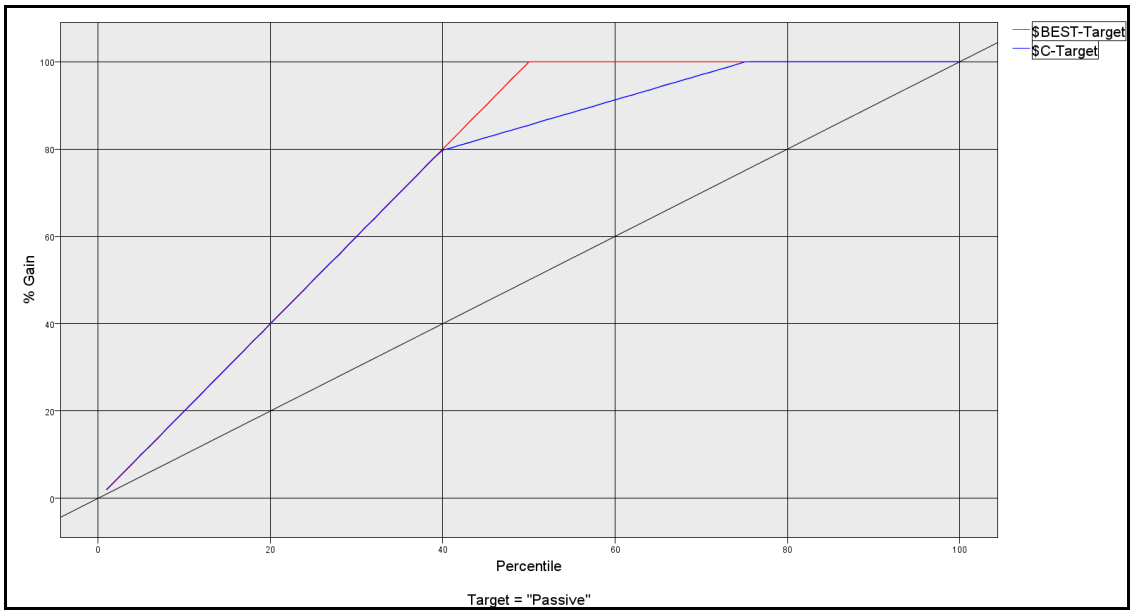


Figure 6-4 : C5.0 Tree Gain Chart