A WELCH'S t-TEST-BASED FEATURE SELECTION FRAMEWORK FOR NETWORK INTRUSION DETECTION

A Dissertation submitted
in Partial fulfilment of the Requirements for the award of the Degree of

MASTER OF SCIENCE

in

Applied Mathematics

Submitted by

Srishiti
(Roll No. 24/MSCMAT/06)

Under the Supervision of

Dr. Anshul Arora
Assistant Professor, Department of Applied Mathematic

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

May, 2026

CANDIDATE'S DECLARATION

I, **Srishiti**, declare that the dissertation titled *"A Welch's t-Test–Based Feature Selection Framework for Network Intrusion Detection"* is the work carried out by me for the M.Sc. Applied Mathematics programme at Delhi Technological University. The work was undertaken during August 2025–May 2026 under the supervision of **Dr. Anshul Arora**, Assistant Professor, Department of Applied Mathematics, DTU.

This dissertation has not been submitted, either in whole or in part, for the award of any other degree or diploma at this or any other institution. Material used from books, research articles, datasets, software libraries and other external sources has been acknowledged in the text and listed in the bibliography.

Place: Delhi — Srishiti

Date: _____ — (M.Sc. Applied Mathematics)

SUPERVISOR'S CERTIFICATE

Certified that the dissertation titled *"A Welch's t-Test–Based Feature Selection Framework for Network Intrusion Detection"* has been prepared by **Srishiti** under my supervision in the Department of Applied Mathematics, Delhi Technological University, for partial fulfilment of the requirements of the M.Sc. Applied Mathematics degree.
To the best of my knowledge, the dissertation records the candidate's own work and has not previously been submitted, in full or in part, to any university or institute for the award of a degree or diploma.

Place: Delhi
Date:

Dr. Anshul Arora
Professor
Department of Applied Mathematics
Delhi Technological University

ACKNOWLEDGEMENTS

I express my sincere thanks to my supervisor, **Dr. Anshul Arora**, Professor, Department of Applied Mathematics, Delhi Technological University. Her guidance helped me keep the work mathematically precise while also making space for the computational part of the study. His critical insight into the practical aspect of network security, machine learning and implementation choices were especially valuable in shaping the final form of this dissertation.

I am thankful to the faculty members of the Department of Applied Mathematics for the training I received during the M.Sc. programme.

I am thankful to the developers and maintainers of the open-source Python ecosystem — in particular `scikit-learn`, `pandas`, `NumPy`, `SciPy`, `XGBoost`, and `Matplotlib` without which the experimental component of this dissertation would not have been possible. I am also indebted to the team at the Cyber Range Lab of UNSW Canberra for making the UNSW-NB15 dataset publicly available; their work has enabled an entire generation of intrusion-detection research, including this modest contribution.

Finally, I thank my family and friends for their encouragement and patience throughout the preparation of this work. Any errors or omissions that remain are my responsibility.

Srishiti

ABSTRACT

A Welch's t-Test-Based Feature Selection Framework for Network Intrusion Detection

Srishiti (Roll No. 24/MSCMAT/06)

The unprecedented growth of digital infrastructure, cloud-based services, and Internet-of-Things deployments has been accompanied by a parallel surge in the volume and sophistication of network-based cyber-attacks. Intrusion Detection Systems (IDS) play a central role in the defensive posture of modern organisations by continuously monitoring network traffic and flagging activity that deviates from expected behaviour. Machine-learning-based IDS have largely supplanted purely signature-driven detectors because of their ability to generalise to previously unseen attack patterns, but their predictive performance and computational footprint are extremely sensitive to the input feature space. Modern network-traffic datasets routinely contain dozens of derived attributes, many of which are noisy, redundant, or only weakly predictive; including them indiscriminately inflates training time, increases overfitting risk, and dilutes the signal that genuinely separates benign and malicious flows.

This dissertation develops, implements and empirically evaluates a statistically motivated feature-selection framework based on Welch's t-test for the binary classification of network traffic into normal and attack categories. Welch's t-test is a two-sample location test that, unlike the classical Student's t-test, does not assume equal variances between the two populations being compared. This relaxation makes it particularly well-suited to intrusion-detection datasets, where the variances of attack and benign-traffic feature distributions are typically very different. For each numerical feature in the dataset, Welch's t-test produces a t-statistic whose absolute value can be interpreted as a measure of how strongly that feature separates the two classes; ranking features by |t| therefore yields a fast, transparent, and theoretically defensible filter selector.

The framework is evaluated on the UNSW-NB15 benchmark dataset, which contains 257,673 records of network traffic collected at the Cyber Range Lab of UNSW Canberra and covers nine attack families (Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms) alongside normal traffic. After standard preprocessing, the dataset retains 42 informative features. Six classifiers from distinct algorithmic families — Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, AdaBoost, and XGBoost — are trained on feature subsets selected by Welch's t-test at varying cardinalities. Cross-validated, stratified train–test evaluation is used throughout to obtain robust generalisation estimates.

Three principal findings emerge from the experiments. First, Welch's t-test produces a stable, interpretable ranking of features in which control-traffic attributes such as the source-to-destination time-to-live (sttl), the connection-rate counters (ct_dst_sport_ltm,

ct_state_ttl, ct_src_dport_ltm), and the per-second packet rate dominate. Second, classification accuracy increases sharply as the most informative features are added but plateaus at approximately k = 30 features, beyond which extra features yield diminishing or even slightly negative returns; selecting only the top 30 features therefore offers a near-optimal accuracy–efficiency trade-off. Third, applying Welch's t-test as a pre-classifier filter improves every single classifier we evaluated; the gain is most pronounced for Random Forest, whose accuracy rises from 80.81% (no feature selection) to 95.18% (with Welch-selected top-30 features). Random Forest achieves the best overall performance, narrowly outperforming XGBoost (95.10%) and Decision Tree (93.90%) in terms of overall accuracy while also delivering a balanced precision–recall profile across both the normal and attack classes.

Beyond the headline accuracy numbers, the framework demonstrates that a carefully justified statistical filter can compete favourably with more expensive wrapper-based or embedded feature-selection schemes while remaining easy to audit, easy to implement, and computationally inexpensive enough to run on embedded or edge-deployed IDS infrastructure. The dissertation closes with a discussion of the framework's limitations, including its inability to capture pairwise feature interactions, its sensitivity to severe class imbalance, and its reliance on the binary normal-versus-attack abstraction. Promising avenues for future work include hybridising Welch's t-test with mutual-information or relief-based methods, extending the pipeline to multi-class attack-family classification, and integrating it with deep-learning encoders for online, real-time intrusion detection.

Keywords: Intrusion Detection System, Feature Selection, Welch's t-test, Machine Learning, Network Security, UNSW-NB15, Random Forest, XGBoost.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

Abbreviation/Symbol	Full Form/Meaning
IDS	Intrusion Detection System
NIDS	Network Intrusion Detection System
HIDS	Host-based Intrusion Detection System
ML	Machine Learning
DL	Deep Learning
RF	Random Forest
DT	Decision Tree
LR	Logistic Regression
GB	Gradient Boosting
XGB	Extreme Gradient Boosting
DoS	Denial of Service
DDoS	Distributed Denial of Service
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
TTL	Time-To-Live
UNSW	University of New South Wales
KDD	Knowledge Discovery in Databases
CICIDS	Canadian Institute for Cybersecurity Intrusion Detection System dataset
ANOVA	Analysis of Variance
MI	Mutual Information
t	Welch's t-statistic
μ	Population mean
s^2	Sample variance
n	Sample size
H_0	Null hypothesis
H_1	Alternative hypothesis
df	Degrees of freedom (Welch–Satterthwaite)
α	Significance level (typically 0.05)

CHAPTER 1

INTRODUCTION

1.1 Background and Context

The computing environment of the twenty-first century is unrecognisable from that of even fifteen years ago. What was, in 2005, a relatively static collection of desktop computers connected to a corporate intranet has, by 2026, become a dense lattice of mobile devices, cloud-hosted services, container orchestration platforms, virtualised network functions, industrial control systems, and Internet-of-Things (IoT) sensors. The number of distinct end-points that an organisation must defend has grown by orders of magnitude, the throughput of the underlying network has increased proportionally, and the topology of the network has flattened in ways that make perimeter-based defences increasingly inadequate.

In parallel, the threat landscape has shifted from a population of isolated hobbyist attackers to a mature criminal economy that bears more resemblance to a software industry than to a hacker subculture. Ransomware-as-a-Service operators rent attack platforms on dark-web marketplaces, state-sponsored Advanced Persistent Threat (APT) groups maintain long-dwell intrusions inside high-value targets, and commodity botnets are spun up and torn down on hourly cycles. The cost imposed on the global economy by cybercrime is now measured in trillions of US dollars per year, and the most-cited estimates suggest that this figure continues to grow at a double-digit annual rate. Within this picture, the perimeter has become permeable by design — every authenticated user, every API call from a third-party SaaS application, every signed update from a software vendor is a potential vector of compromise.

In response, the security community has converged on the idea of "defence in depth": no single defensive measure is treated as sufficient, and detection is given equal weight to prevention. The assumption that intrusions will eventually succeed motivates an investment in monitoring infrastructure capable of identifying suspicious activity quickly enough to limit damage. Intrusion Detection Systems (IDS) form the analytical layer of this defensive architecture; they consume the raw telemetry produced by networks and hosts and, ideally, surface alerts that lead to human investigation and containment well before an attacker can achieve their objectives. Designing an IDS that is sensitive enough to detect novel attacks while not drowning analysts in false positives is, however, far from trivial, and is the central problem that motivates the present dissertation.

1.2 Network Intrusion Detection Systems

Intrusion Detection Systems are classified along two orthogonal axes. The first axis distinguishes between Host-based IDS (HIDS), which run on individual end-points and monitor system calls, audit logs, file integrity, and process activity, and Network-based IDS (NIDS), which sit at strategic points on the network and analyse traffic flows as they traverse the wire. The work presented in this dissertation falls squarely within the NIDS category.

The second axis distinguishes between signature-based detectors, which match incoming traffic against a database of known-bad patterns, and anomaly-based detectors, which build a model of normal behaviour and flag deviations from it. Signature-based detectors have the appealing property of producing few false positives when the signature database is well curated, but they are by construction blind to novel attacks for which no signature has yet been written — a fatal limitation given the velocity at which new attack techniques appear. Anomaly-based detectors can in principle detect novel attacks, but they pay for this generalisation with a much higher false-positive rate and a much greater dependence on the quality of the data used to characterise "normal".

Modern NIDS deployments typically combine both paradigms in a single platform. Open-source tools such as Snort, Suricata, and Zeek (formerly Bro) ship with extensive signature catalogues but also expose hooks for custom analytical pipelines that perform anomaly detection on top of the same packet capture. Within the academic literature, the focus has shifted decisively towards machine-learning-based anomaly detection over the last decade, and it is in that branch of the literature that the present dissertation sits.

A standard machine-learning NIDS pipeline consists of four stages: (i) traffic capture at the wire, often using span ports or in-line taps; (ii) feature extraction, in which packets are aggregated into flows and per-flow statistics (durations, byte counts, packet counts, protocol fields, behavioural counters) are computed; (iii) feature selection or feature engineering, in which the high-dimensional feature space is reduced to a manageable set of informative attributes; and (iv) classification, in which a learned model labels each flow as benign or malicious. This dissertation focuses on stages (iii) and (iv), and in particular on demonstrating that a careful choice of feature-selection method can dramatically improve the quality of the resulting classifier.

1.3 The Need for Feature Selection

Why bother with feature selection at all? After all, modern classifiers — ensemble methods, gradient-boosted trees, neural networks — are widely advertised as being able to "figure out" which features matter on their own. The argument for feature selection rests on three independent considerations.

Statistical efficiency. Every irrelevant feature included in the training data contributes a finite amount of noise that the classifier must learn to ignore. For a fixed training-set size, this noise translates into higher variance of the learned model and a corresponding loss in test accuracy. The phenomenon is known in the statistical-learning literature as the curse of dimensionality, and it bites hardest in settings — such as intrusion

detection — where the number of available features is moderate to large but the truly informative features form a small subset.

Computational efficiency. Training time, memory consumption, and inference latency all scale with the number of features. In a deployed NIDS, where decisions may need to be made at hundreds of thousands of flows per second on commodity hardware, the difference between 30 features and 300 features can be the difference between a useful detector and an unusable one. Recent papers from the network-security community report inference latencies of single-digit milliseconds for tree-based models with around 30 features but several tens of milliseconds when the same models are trained on the full feature set.

Interpretability and auditability. Security teams need to defend their alerts in front of human investigators. A model that flags an event because it observed an unusual TTL combined with a high connection-rate counter is far easier to explain (and to debug, and to retrain) than one that fires because of an opaque combination of two hundred near-collinear features. Feature selection is therefore valuable not only for the model itself but for the human workflow that surrounds it.

Feature-selection techniques are typically classified into three groups. Filter methods, of which the proposed Welch's-t-test framework is an example, evaluate features by some intrinsic statistical criterion that is independent of any particular classifier and select features in a single pass. Wrapper methods iteratively train a candidate classifier on different feature subsets and use the resulting accuracy to guide the search; they are accurate but expensive. Embedded methods perform feature selection as a by-product of model training, as is the case with L1-regularised logistic regression or feature-importance scores from tree ensembles. Each family has well-documented strengths and weaknesses, and the choice among them is itself an engineering decision.

1.4 Statistical Methods in Intrusion Detection

Although machine learning has come to dominate the recent literature on intrusion detection, the statistical foundations on which it rests — estimation theory, hypothesis testing, robust statistics — have a much older history of application to security problems. Some of the very earliest anomaly-detection systems, developed in the 1980s by Denning and others, used basic statistical descriptors (running means and standard deviations of audit-log events) to flag deviations from a user's typical behaviour. The intervening forty years have seen a continual rediscovery of these statistical roots, often dressed in modern machine-learning vocabulary.

In the feature-selection sub-problem, classical statistical tests offer a particularly attractive starting point. Their assumptions are well understood, their behaviour is well characterised in finite samples, and their output (a test statistic and a p-value per feature) is interpretable in a way that, say, the SHAP values of a gradient-boosted ensemble are not. Among such tests, the two-sample t-test stands out as a natural choice for binary classification problems: it asks, of each feature in turn, whether its mean differs significantly between the two classes. A feature whose class-conditional means are far apart relative to within-class variability is, almost by definition, useful for discrimination.

The Student's t-test, however, assumes equal variances in the two populations

being compared. This assumption is virtually always violated in intrusion-detection data, where the within-class variance of attack traffic for any given feature can be many times larger or smaller than the within-class variance of benign traffic. The Welch's t-test, which dispenses with the equal-variance assumption and uses the Welch–Satterthwaite approximation to compute its degrees of freedom, is therefore a more appropriate choice. The present dissertation takes this observation as its starting point and develops it into a complete feature-selection framework.

1.5 Motivation

Several specific observations motivate the work described in the chapters that follow. First, the published literature on intrusion-detection feature selection is heavily skewed towards information-theoretic and tree-based criteria (mutual information, gain-ratio, ReliefF, gain importance) and is comparatively light on classical hypothesis-test-based criteria. Where Welch's t-test has been used, it has typically been used incidentally as one entry in a comparison rather than as the centrepiece of a sustained methodological investigation. Second, modern benchmark datasets such as UNSW-NB15 and CICIDS-2017 routinely contain 40–100 features per flow, of which only a small fraction are individually highly informative. The standard ANOVA-based filter assumes homoscedasticity, which is unrealistic; the standard Student's t-test makes the same assumption. There is therefore a clear gap for a filter selector that retains the simplicity and interpretability of the t-test family but does not require equal variances. Third, from a pedagogical standpoint, demonstrating that a classical, hundred-year-old statistical test can compete with much more elaborate selectors is itself valuable. It reinforces the idea that mathematical sophistication and computational sophistication are not the same thing, and that careful problem framing often beats raw algorithmic complexity. Fourth, real-world deployment constraints favour cheap, transparent selectors. Many organisations — particularly in regulated industries such as banking, healthcare, and telecommunications — need to be able to justify, in plain language, why a particular set of features feeds their detection model. A statistical test with a clear null hypothesis and a published track record is much easier to defend in front of a regulator than a black-box selector trained on a proprietary algorithm.

1.6 Problem Statement

The central problem addressed in this dissertation can be stated as follows. Given a labelled dataset of network flows, each described by a set of numerical and categorical features and labelled as either normal or attack, design a feature-selection procedure that (i) is grounded in well-understood statistical theory, (ii) does not require equal variances between the two classes, (iii) is computationally cheap enough to scale to datasets of hundreds of thousands of flows and tens of features, (iv) produces an interpretable ranking of features, and (v) when used as a pre-classifier filter, allows downstream machine-learning classifiers to achieve accuracy that is at least comparable to — and ideally better than — what they would achieve without feature selection. The chosen procedure must then be empirically validated on a standard benchmark dataset against a representative set of modern classifiers.

1.7 Research Objectives

The specific objectives of this dissertation are:

1. To formalise Welch's t-test as a feature-selection criterion for binary network-traffic classification, including its mathematical derivation, its assumptions, and its behavioural properties.
2. To implement a complete feature-selection-plus-classification pipeline on the UNSW-NB15 dataset, with appropriate preprocessing, encoding, scaling, and cross-validation.
3. To empirically determine the optimal number of features to retain, by sweeping the cardinality of the selected subset and observing the resulting classification accuracy.
4. To benchmark six classifiers from distinct algorithmic families — Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, AdaBoost, and XGBoost — both with and without Welch's-t-test feature selection.
5. To analyse the class-wise performance of the resulting models in detail, recognising that overall accuracy can be a misleading metric in the presence of class imbalance.
6. To position the proposed framework in the broader literature on intrusion detection and to identify the situations in which it is most appropriate and those in which more elaborate selectors are warranted.

1.8 Contributions of the Dissertation

The principal contributions of this work are summarised as follows.

1. A theoretically grounded, statistically robust feature-selection framework based on Welch's t-test is developed for binary network-intrusion classification. The framework explicitly handles the unequal-variance situation that is typical in IDS data and produces a transparent, interpretable ranking of features.
2. The framework is implemented as a scikit-learn-compatible custom transformer, supporting both Top-N selection and p-value-thresholded selection. This implementation is suitable for inclusion in production data-science pipelines.
3. A comprehensive empirical evaluation on the UNSW-NB15 dataset is presented, in which the proposed selector is benchmarked across six distinct classifiers. Reproducibility is supported through a stratified train–test split, five-fold cross-validation, and a description of all relevant hyper-parameters.
4. The optimal feature-subset cardinality is empirically identified at $k = 30$, and is shown to deliver an accuracy of 95.18 % with Random Forest — a fourteen-percentage-point improvement over the same classifier without feature selection.
5. A detailed class-wise analysis is provided, demonstrating that the gains achieved by the proposed framework do not come at the cost of disproportionate degradation on either the normal or attack class. Random Forest and XGBoost in particular deliver balanced precision–recall profiles that make them suitable for deployment.
6. A discussion of the practical and theoretical limitations of the framework, together with a roadmap for future research, is included so that subsequent work can build on the dissertation in a structured way.

CHAPTER 2

LITERATURE REVIEW AND THEORETICAL BACKGROUND

2.1 Evolution of Intrusion Detection Systems

The intellectual origin of the IDS field is generally traced to Denning's seminal 1987 paper, in which she proposed a formal model of intrusion detection based on the observation that intrusions tend to cause exploitations of system vulnerabilities that result in abnormal usage patterns. Denning's model framed detection as a statistical inference problem over audit-log events and laid out, in essentially modern form, the distinction between profile-based anomaly detection and rule-based misuse detection. The paper's influence on subsequent work is hard to overstate; almost every IDS textbook published in the intervening forty years opens with a discussion of Denning's framework.

Throughout the 1990s, IDS research was dominated by signature-based approaches, partly because the variety of attacks in the wild was relatively limited and partly because signature matching was a natural fit for the computing resources of the time. The release of open-source signature-based detectors such as Snort in 1998 democratised access to IDS technology and established a de facto standard for rule-based detection that persists in modified form today.

The 2000s saw the rapid commoditisation of internet access, the rise of worms (Code Red, Slammer, Conficker), and the consequent recognition that signature-based detectors alone could not keep up with the speed at which new attacks were being released. This decade witnessed the emergence of machine-learning approaches to IDS as a serious research direction. Early work focused on classical algorithms — decision trees, naive Bayes, support-vector machines — applied to the KDD Cup 1999 dataset and its derivatives. Although the KDD'99 dataset has since been criticised heavily for its redundancy and unrealistic class distribution, it provided the substrate on which the field's vocabulary and evaluation methodology were established.

The 2010s brought two parallel developments: the rise of deep learning and the recognition that the academic benchmarks of the previous decade were no longer representative of contemporary network traffic. Vinayakumar et al. (2019) and others demonstrated that deep neural networks could achieve very high accuracy on intrusion-detection benchmarks, although these gains often came at the cost of computational expense and reduced interpretability. The release of newer datasets, especially NSL-KDD (a debiased subset of KDD'99) and UNSW-NB15 (a freshly captured dataset reflecting modern attack families), provided the foundation for a wave of re-evaluations of older techniques and,

simultaneously, for the development of new ones.

2.2 Machine-Learning Approaches in IDS

Machine-learning approaches to intrusion detection treat the problem as a supervised binary or multi-class classification task: given a feature vector representing a network flow or session, predict its label. The literature is by now too large to survey exhaustively, but it can be usefully organised by the algorithmic family being applied.

2.2.1 Decision-Tree-Based Methods

Decision trees were among the very first machine-learning models applied to the KDD'99 dataset, and they have retained their popularity for two principal reasons: they are interpretable (a decision tree can be read as a sequence of if-then-else rules) and they are fast both to train and to query. C4.5, J48, and CART implementations were the workhorses of the late 1990s and early 2000s. More recent decision-tree work has focused on dealing with class imbalance (using cost-sensitive splitting criteria or class-weighted Gini impurity) and on incremental updating to support streaming network data.

2.2.2 Ensemble Methods

The principal weakness of a single decision tree — high variance, susceptibility to over-fitting — is addressed by ensemble methods that combine many trees. The two dominant families are bagging-based ensembles (of which Random Forest, due to Breiman in 2001, is the canonical example) and boosting-based ensembles (AdaBoost, Gradient Boosting Machines, XGBoost, LightGBM). Both families have been applied extensively to IDS problems. Random Forest in particular has proved remarkably robust across IDS benchmarks; in Aljawarneh, Aldwairi & Yassein (2018), in Zhou, Cheng, Jiang & Dai (2020), and in many other studies it appears in the top-performing tier with relatively little hyper-parameter tuning.

XGBoost, introduced by Chen and Guestrin in 2016, has become the de facto choice in many production data-science settings because of its strong performance on tabular data, its efficient implementation, and its built-in regularisation. In the IDS context it has been applied by numerous authors and is one of the six classifiers benchmarked in the present dissertation.

2.2.3 Logistic Regression and Other Linear Models

Logistic regression, despite being the oldest of the methods considered here, retains a place in the IDS toolkit. Its principal value is as a baseline: a method that requires no hyper-parameter tuning, runs in seconds on the largest available datasets, and produces fully interpretable coefficients. In the experiments reported in Chapter 6 it serves precisely this role. When combined with appropriate scaling and feature selection, logistic regression's performance can be surprisingly competitive.

2.2.4 Deep-Learning Approaches

The 2010s saw a wave of papers applying deep neural architectures to IDS. Artificial neural networks (ANNs) with one or two hidden layers were applied first; these were quickly followed by convolutional neural networks (CNNs) that treat the feature vector as a one-dimensional signal, recurrent neural networks (RNNs) that model the temporal evolution of traffic statistics, and autoencoders for unsupervised anomaly detection. Vinayakumar et al. (2019) is a particularly comprehensive comparative study. Ferrag, Maglaras, Janicke & Jiang (2020) provide a more recent survey.

Deep learning has produced state-of-the-art accuracy on several IDS benchmarks, but at a cost. Training requires substantial compute, hyper-parameter tuning is delicate, and the resulting models are notoriously difficult to interpret. For deployments in resource-constrained environments — industrial control networks, edge IDS appliances, IoT gateways — the trade-off does not always favour deep learning. The present dissertation positions itself accordingly: its proposed framework is intentionally simple and inexpensive, and is intended to be complementary to, rather than competitive with, deep-learning approaches in settings where the latter are appropriate.

2.3 Feature-Selection Techniques in IDS

Feature-selection methods in the IDS literature can be grouped, as already noted in Chapter 1, into filter, wrapper, and embedded methods. This section reviews representative examples of each.

2.3.1 Filter Methods

Filter methods evaluate features by an intrinsic statistical criterion. The most common criteria used in IDS work include: mutual information between the feature and the class label (Liu and Motoda, 1998; Guyon and Elisseeff, 2003); the chi-squared test of independence; the ANOVA F-statistic; the gain-ratio criterion derived from information theory; ReliefF and its variants, which are nearest-neighbour-based; and — the focus of this dissertation — the two-sample t-test. Filter methods are computationally cheap, do not depend on a particular classifier, and produce a ranking of features that is straightforward to interpret. Their principal weakness is that they typically consider features one at a time and therefore can miss interactions between features.

2.3.2 Wrapper Methods

Wrapper methods evaluate a candidate feature subset by training the downstream classifier on it and measuring the resulting accuracy. The classical wrapper algorithms are sequential forward selection, sequential backward elimination, and recursive feature elimination. Genetic algorithms and other meta-heuristics have also been used as wrapper search procedures. Wrappers can in principle discover feature interactions that filters miss, but at the cost of substantial computational expense — each candidate subset requires a full training-and-evaluation cycle of the downstream classifier.

2.3.3 Embedded Methods

Embedded methods perform feature selection as a by-product of model training. Examples include L1-regularised (LASSO) logistic regression, in which features with zero coefficients are effectively eliminated; feature-importance scores from tree-based ensembles, which can be thresholded to select the most useful features; and the attention weights of certain neural architectures. Embedded methods are typically competitive with wrappers in accuracy at a fraction of the computational cost, but they are inherently tied to a particular model family.

2.4 Statistical Feature Selection

Within the filter family, statistical tests of class separability have a particular appeal. They are old, well-understood, and have explicit assumptions that can be checked. They produce a test statistic and a p-value per feature, both of which can be ranked and thresholded. They make minimal commitments to a particular downstream classifier. And, because they are derived from the language of mainstream statistics, they are easier to defend to non-machine-learning stakeholders — a non-trivial consideration in regulated industries.

The ANOVA F-test is the most common entry in this family for problems with more than two classes; for binary problems it reduces, up to a monotone transformation, to the square of the two-sample t-statistic. The Student's t-test — the original two-sample location test — assumes equal variances in the two populations, an assumption that is rarely defensible for security data. The Welch's t-test, introduced by B. L. Welch in 1947, dispenses with this assumption by computing the test statistic with the two sample variances independently and approximating the degrees of freedom using the Welch–Satterthwaite formula. The mathematical detail is given in Chapter 3.

Several IDS papers have used Welch's t-test or closely related variance-stabilised statistics in passing, often without explicitly invoking Welch's name. The contribution of the present dissertation is to centre the Welch construction methodologically, to derive it carefully, to implement it as a reusable feature selector, and to evaluate it systematically on UNSW-NB15.

2.5 Datasets Used in IDS Research

2.5.1 KDD Cup 1999

The KDD Cup 1999 dataset — itself derived from the 1998 DARPA Intrusion Detection Evaluation — was the field's first widely used benchmark. It contains approximately five million labelled records, each consisting of 41 features and a class label drawn from one of five categories (normal, DoS, probe, R2L, U2R). Despite its longevity and influence, KDD'99 has been the subject of sustained criticism: it contains substantial redundancy (many records are near-duplicates), it does not reflect modern attacks, and its class distribution is heavily skewed in unrealistic ways.

2.5.2 NSL-KDD

Tavallaee, Bagheri, Lu & Ghorbani (2009) addressed several of KDD'99's defects by deduplicating it and re-sampling to produce a more balanced distribution; the resulting dataset is known as NSL-KDD. It has been used extensively as a benchmark in subsequent work, though it inherits KDD'99's age-related limitations regarding attack realism.

2.5.3 UNSW-NB15

Moustafa and Slay (2015) released the UNSW-NB15 dataset, captured at the Cyber Range Lab of the Australian Centre for Cyber Security at UNSW Canberra. UNSW-NB15 contains approximately 2.5 million records of synthetic network traffic generated by the IXIA PerfectStorm tool, of which roughly 257,673 records are made publicly available in the training and testing partitions used in the present dissertation. Forty-nine features are originally extracted per flow; after standard preprocessing this reduces to 42. Records are labelled both with a binary normal/attack indicator and with one of nine attack categories: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. UNSW-NB15 is widely regarded as one of the most realistic publicly available IDS benchmarks, and is the dataset used throughout this dissertation.

2.5.4 CICIDS-2017 and Other Recent Datasets

CICIDS-2017, released by the Canadian Institute for Cybersecurity, captures traffic from a more diverse range of contemporary attack scenarios, including DDoS, brute-force, web attacks, infiltration, and botnet activity. Although CICIDS-2017 is not used in the present experiments, it would be a natural target for the kind of cross-dataset validation that Section 7.4 identifies as future work.

2.6 Identified Research Gaps

From this survey of the literature, four gaps emerge that the present dissertation directly addresses.

1. First, while filter-based feature selectors are widely used, the Welch's t-test variant has not received sustained methodological treatment in the IDS literature. The dominant filter selectors are ANOVA F (which assumes equal variances), mutual information, and ReliefF; Welch's t-test, despite its theoretical superiority for heteroscedastic class distributions, is conspicuously underused.
2. Second, the question of how many features to retain is often answered by rule-of-thumb or by reference to a pre-existing target ("top 10", "top 20"). A principled sweep over the cardinality, accompanied by a discussion of the accuracy-efficiency trade-off, is comparatively rare. The present dissertation provides exactly such a sweep in Section 6.2.
3. Third, comparisons of classifiers in IDS papers tend to be opportunistic, focusing on one or two algorithms. A systematic head-to-head of six classifiers from distinct

algorithmic families, all evaluated under identical preprocessing and feature-selection conditions, is uncommon. Section 6.3 provides such a comparison.

4. Fourth, class-wise analysis (precision, recall and F1 on the normal and attack classes separately) is reported less consistently than overall accuracy, despite being more informative in the presence of class imbalance. Section 6.5 prioritises class-wise analysis in this dissertation.

Addressing these four gaps is the methodological agenda of the chapters that follow.

2.7 Probability and Statistics Preliminaries

A random variable X is a measurable function from a probability space (, , P) to the real numbers. The distribution of X is fully specified by its cumulative distribution function F_X(x) = P(X x), or equivalently, in the continuous case, by its probability density function f_X(x). Two summary characteristics of a distribution are particularly important for the present work: the population mean = E[X] = x f_X(x) dx, which measures central tendency, and the population variance ² = Var(X) = E[(X)²], which measures dispersion. The square root of the variance is the standard deviation.

In practice we rarely have access to the population from which our data are drawn; we only have a finite sample. Given a sample X, X, . . . , X_n of independent and identically distributed observations from a distribution with mean and variance ², the standard unbiased estimators of these two quantities are the sample mean and the sample variance,

$$\bar{x} = \frac{1}{n}\sum_i X_i, \qquad s^2 = \frac{1}{n-1}\sum_i (X_i - \bar{x})^2 \tag{2.1}$$

Two foundational results connect these sample statistics to inferential procedures. The Law of Large Numbers guarantees that, as n → , the sample mean converges (in probability and almost surely) to the population mean. The Central Limit Theorem (CLT) goes further and characterises the distribution of the sample mean for finite n: for sufficiently large n, regardless of the shape of the underlying distribution,

$$\frac{\bar{x} - \mu}{\sigma/\sqrt{n}} \sim N(0,1) \quad \text{approximately} \tag{2.2}$$

When is unknown and is replaced by its sample estimate s, the distribution of the resulting pivot is no longer exactly standard normal; it follows the Student's t-distribution with n 1 degrees of freedom. The t-distribution is bell-shaped, symmetric about zero, and has heavier tails than the standard normal, with the additional weight in the tails reflecting the additional uncertainty introduced by estimating from the sample. As n → , the t-distribution converges to the standard normal.

2.8 Hypothesis Testing

Hypothesis testing is the procedure by which we use sample data to make a decision between two competing claims about the population from which the sample was drawn. The framework, due in essentially modern form to Neyman and Pearson, proceeds in five steps.

1. State the null hypothesis H and the alternative hypothesis H. The null hypothesis is typically the "no effect" claim that the analyst would like to reject; the alternative is the claim of interest.
2. Choose a test statistic T whose distribution under H is known.
3. Choose a significance level , conventionally 0.05 in many fields and 0.01 in others. The significance level is the maximum tolerated probability of incorrectly rejecting H when it is true (a Type I error).
4. Compute the value of T from the observed data, and from that value compute the p-value: the probability, under H, of obtaining a test statistic at least as extreme as the one observed.
5. Reject H if and only if the p-value is less than or equal to .

In the feature-selection setting of this dissertation, H will be the claim that the mean value of a feature is the same in the normal and attack populations, and H will be the claim that the means differ. Features for which we reject H are candidates for inclusion in the classifier's input set; features for which we cannot reject H are candidates for exclusion.

2.9 The Student's t-Test

Suppose we have two independent samples X¹, . . . , X_{n}¹ and X², . . . , X_{n}² drawn from populations with means and respectively. The classical two-sample Student's t-test asks whether = against the alternative . Under the additional assumption that the two populations have the same variance ², the pooled variance estimator

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2} \tag{2.3}$$

combines information from both samples, and the test statistic

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s_p\sqrt{1/n_1 + 1/n_2}} \tag{2.4}$$

follows, under H : = , a Student's t-distribution with n + n 2 degrees of freedom. Large absolute values of t provide evidence against H.

The Student's t-test is elegant, well understood, and has been a workhorse of applied statistics for more than a century. Its principal limitation is the homoscedasticity assumption — the equal-variance assumption between the two populations. When this assumption fails, the actual Type I error rate of the test can depart substantially from the nominal , particularly when the two samples are of unequal size. For intrusion-detection data, where the variance of attack-traffic statistics is virtually always different from the

variance of benign-traffic statistics, the homoscedasticity assumption is essentially never tenable.

2.10 The Welch's t-Test

Welch (1947) addressed precisely this limitation. His variant of the two-sample t-test allows the two populations to have different variances. The test statistic becomes

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2/n_1 + s_2^2/n_2}} \tag{2.5}$$

The denominator is the standard error of the difference of sample means computed without any pooling: each sample contributes its own variance estimate. Notice that when n = n and $s^2 = s^2$, the Welch statistic reduces to the Student's t-statistic; the two tests differ only when the variance estimates differ or the sample sizes are unequal.

The crucial new question is: what is the sampling distribution of (3.5) under H? The unwelcome answer is that, in general, it does not follow a t-distribution exactly. Welch's contribution was to show that it can be very well approximated by a t-distribution provided the degrees of freedom are computed by the Welch–Satterthwaite formula,

$$df = \frac{(s_1^2/n_1 + s_2^2/n_2)^2}{\frac{(s_1^2/n_1)^2}{n_1 - 1} + \frac{(s_2^2/n_2)^2}{n_2 - 1}} \tag{2.6}$$

In general, df will not be an integer. Statistical software either uses the non-integer df directly (the Satterthwaite default in most implementations, including SciPy's ttest_ind with equal_var=False) or rounds it down to the nearest integer (the more conservative approach). The approximation is known to be very accurate in finite samples across a wide range of population distributions.

2.10.1 Assumptions of Welch's t-Test

Three assumptions remain for Welch's t-test to be applied legitimately.

1. Independence: the observations within each sample are independent of one another, and the two samples are independent of each other. For network-traffic data, this is approximately but not exactly true — successive flows from the same conversation are obviously correlated. In practice, the violation is mild enough that the test remains useful as a feature-screening tool.
2. Approximate normality: the underlying populations are approximately normally distributed. For large samples, the CLT makes this assumption essentially redundant for the sampling distribution of the mean; for moderate samples it can be checked using histograms or QQ-plots. In the UNSW-NB15 dataset, many features exhibit heavy-tailed or skewed distributions, but with sample sizes in the tens of thousands the asymptotics dominate.
3. Continuous-valued features: the test is designed for continuous data. Categorical features must be encoded numerically (we use one-hot encoding) before being fed

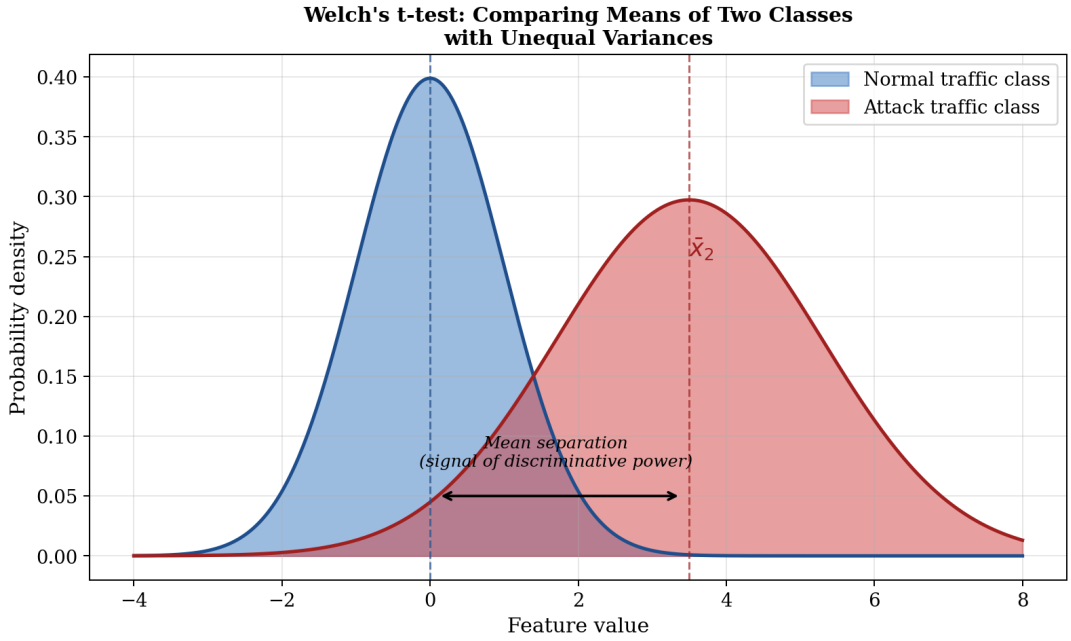

Figure 2.1: Welch's t-test compares the means of two class-conditional distributions of a feature. When the means are well-separated relative to within-class variability, the feature is informative for classification.

into the test.

2.10.2 Advantages over Alternatives

Compared to the alternatives that dominate the IDS feature-selection literature, Welch's t-test offers a particular combination of properties:

- Compared to the ANOVA F-test, it accommodates unequal variances explicitly.
- Compared to Student's t-test, it does not require pooling and is more robust to violations of the equal-variance assumption.
- Compared to mutual information, it has an explicit, well-understood null distribution and produces a p-value that can be interpreted directly.
- Compared to ReliefF and its descendants, it is much cheaper computationally — linear in the sample size rather than quadratic.
- Compared to embedded methods, it is decoupled from any particular classifier, so its output can be re-used across classifiers without re-running the selection.

2.11 Evaluation Metrics

The output of a binary classifier on a labelled test set can be summarised by a confusion matrix, which counts the four possible outcomes: true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP). From the confusion matrix a number of

derived metrics are computed.

Accuracy is the fraction of test samples for which the classifier's prediction matches the true label: Accuracy = (TP + TN) / (TP + TN + FP + FN). It is the most familiar metric but can be misleading in the presence of class imbalance.

Precision for the positive class is the fraction of positive predictions that are actually positive: Precision = TP / (TP + FP). For an IDS, precision on the attack class corresponds to one minus the false-positive rate; high precision means analysts will not be deluged with spurious alerts.

Recall for the positive class is the fraction of actual positives that are correctly identified: Recall = TP / (TP + FN). For an IDS, recall on the attack class measures sensitivity to actual attacks; missed detections (low recall) are a serious operational problem.

F1-score is the harmonic mean of precision and recall, F1 = 2 · Precision · Recall / (Precision + Recall). The F1-score balances the two and is the single most useful summary metric for imbalanced classification.

In Chapter 6 we report all four of these metrics, both for the attack class and for the normal class, on the held-out test partition of UNSW-NB15. The class-wise reporting is essential: a model could achieve very high overall accuracy by simply classifying every flow as benign (since most flows are benign) while completely failing to detect attacks.

CHAPTER 3

PROPOSED METHODOLOGY AND IMPLEMENTATION

3.1 System Architecture Overview

The proposed framework follows the standard supervised-learning pipeline architecture, with one key insertion: a Welch's-t-test-based feature-selection stage placed between preprocessing and classification. Figure 4.1 visualises the pipeline.

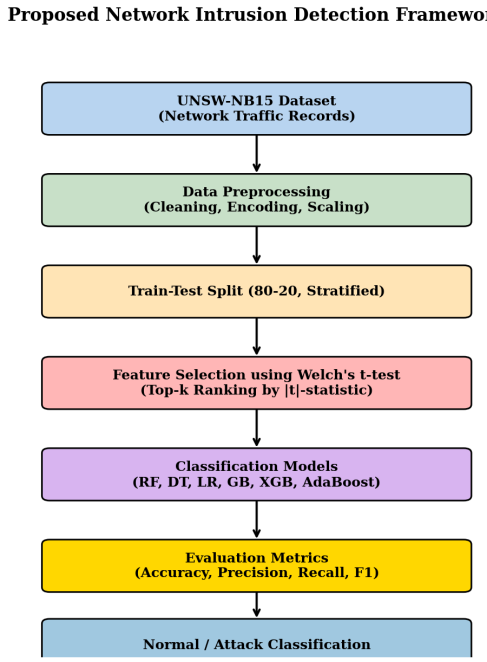

Figure 3.1: Proposed network intrusion detection framework. Raw UNSW-NB15 records pass through preprocessing, Welch-based feature selection, and one of six candidate classifiers; the resulting predictions are evaluated using accuracy, precision, recall, and F1-score.

The pipeline accepts raw network-flow records as input and produces, for each record, a prediction of whether it is benign or malicious. The five stages are: (i) data ingestion, in which the raw CSV files of UNSW-NB15 are read into memory and concatenated; (ii) preprocessing, in which duplicate records are removed, missing values are imputed, the id and attack_cat columns are dropped to prevent data leakage, and

categorical features are one-hot encoded; (iii) train–test splitting, which is stratified on the binary label and which uses an 80/20 ratio with a fixed random seed for reproducibility; (iv) Welch-based feature selection, applied only to numerical features after standard scaling, with the categorical features (one-hot encoded) appended afterwards; and (v) classifier training and evaluation, where one of six classifiers is fitted on the selected features and evaluated on the held-out test set.

Each stage of the pipeline is implemented as a scikit-learn Transformer or Estimator. This design choice has two practical consequences. First, the pipeline can be cross-validated as a single object, ensuring that feature selection is performed inside each fold of cross-validation rather than once on the full training set (which would cause optimistic bias). Second, the pipeline can be serialised and deployed without further modification.

3.2 UNSW-NB15 Dataset

3.2.1 Dataset Description

UNSW-NB15 is a contemporary benchmark intrusion-detection dataset developed at the Cyber Range Lab of UNSW Canberra. Network traffic was generated synthetically by the IXIA PerfectStorm tool, with both normal background traffic and a curated set of attack traffic, and captured at the wire using Tcpdump. The resulting PCAP files were processed by the Argus and Bro-IDS tools to extract per-flow records and by twelve custom feature-extraction algorithms to produce the 49-feature per-record representation that is released as the UNSW-NB15 dataset. The publicly distributed training and testing partitions, used in this dissertation, together contain 257,673 records.

Each record carries two labels: a binary label (0 = normal, 1 = attack) and an attack-category label which takes one of nine values for attack records and is empty for normal records. The attack categories are Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. The class distribution is moderately imbalanced — attack records outnumber normal records in the released partitions, with attacks accounting for roughly 68 % of the data — but the imbalance is not extreme.

3.2.2 Attack Categories

Table 4.1 summarises the nine attack categories. The proportions reported are approximate and based on the combined training and testing partitions used in this dissertation.

Table: Summary of attack categories in UNSW-NB15.

Category	Description	Share (%)
Fuzzers	Sends malformed input to crash or	8.0
	destabilise services.	
Analysis	Includes port-scans, spam, and	1.0
	HTML-injection attempts.	
Backdoors	Software that bypasses authentication for	0.9
	covert access.	
DoS	Denial-of-Service attacks against	5.8
	availability.	
Exploits	Code that abuses a specific software	18.5
	vulnerability.	
Generic	Block-cipher cryptanalysis attacks against	26.5
	ciphertext.	
Reconnaissance	Information-gathering scans before an	5.0
	attack.	

Shellcode Payloads that launch shells inside victim 0.6 processes.
Worms Self-propagating malicious software. 0.07 ———— ———— ———— ———

3.2.3 Feature Description

The 49 raw features extracted per record are organised into five groups: (a) basic flow features (duration, protocol, service, source and destination bytes, source and destination packets, time-to-live values, TCP window sizes); (b) content features (means and sums of TCP-payload-related quantities); (c) time-related features (jitter, inter-arrival means, response durations); (d) general-purpose features (depth of the HTTP-content traffic, TCP base-sequence numbers); and (e) behavioural counters (counts of recent connections to the same destination port, the same state, the same service, and so on). After the standard preprocessing step that drops the row-identifier (id) column and the multi-class attack-category (attack_cat) column, 42 informative features remain for further use; this 42-feature representation is the one to which Welch's t-test is applied.

A complete listing of the 42 features, together with concise descriptions of their semantics, is given in Appendix A.

3.3 Data Preprocessing

3.3.1 Data Cleaning

The released training and testing partitions are first concatenated into a single DataFrame to facilitate consistent preprocessing. The combined DataFrame is then deduplicated; UNSW-NB15 is known to contain a small fraction of exact-duplicate records, and removing them prevents trivial inflation of the test-set performance. Missing values are checked for; the dataset is essentially free of missing values, but the small handful that do appear are imputed using the median (for numerical features) or the mode (for categorical features).

3.3.2 Column Dropping

Two columns are dropped from the feature set. The id column is dropped because it is simply the row index and carries no predictive signal. The attack_cat column is dropped because it directly encodes the attack family and therefore reveals the binary label — retaining it would cause severe data leakage. The remaining 42 columns include the binary label column, which is separated out as the target y for supervised learning.

3.3.3 Encoding Categorical Variables

Three categorical features remain after column dropping: proto (the transport-layer protocol, e.g. tcp, udp, sctp, igmp), service (the application-layer service, e.g. http, dns, ftp, ssh, smtp, and an explicit "-" category for unidentified services), and state (the connection-state code, e.g. FIN, INT, CON, REQ). These are encoded using one-hot encoding via scikit-learn's OneHotEncoder with the handle_unknown='ignore' option, so that unseen

categorical values encountered at test time do not raise errors but rather contribute a zero vector. One-hot encoding produces a sparse representation whose total dimensionality depends on the number of distinct categorical values; in this dataset the encoding produces approximately 175 binary features in total.

3.3.4 Feature Scaling

Numerical features are standardised using scikit-learn's StandardScaler, which subtracts the per-feature training-set mean and divides by the per-feature training-set standard deviation. Standardisation is necessary for the logistic regression baseline (whose coefficient estimates are sensitive to the scale of the features) and is harmless for the tree-based classifiers (whose split decisions are invariant to monotone transformations of individual features). Crucially, the scaler is fitted only on the training partition; the same fitted scaler is then applied to the test partition. This avoids leakage of test-set summary statistics into the training procedure.

3.3.5 Train–Test Split

After preprocessing, the dataset is split into an 80 % training partition and a 20 % test partition using stratified random sampling on the binary label. Stratification ensures that the class proportions in the training and test partitions match those of the full dataset. The random seed is fixed at 42 throughout the experiments to ensure exact reproducibility of the reported numbers.

3.4 Welch's t-Test as a Feature Selector

With the dataset preprocessed, the framework's central novelty enters: feature selection by Welch's t-test.

3.4.1 Selector Algorithm

For each numerical feature A in the training set, the selector partitions the rows according to the binary label, producing two sub-samples: A° (the values of feature A for benign records) and A^{1} (the values for attack records). It then computes the Welch's t-statistic and the associated p-value using equations (3.5) and (3.6) of Chapter 3. The absolute value of the t-statistic is used as the feature's weight; the p-value is retained for diagnostic reporting but not directly used in Top-N selection.

The full procedure is summarised as Algorithm 1.

Algorithm 1: Welch's t-Test Feature Selector

Require: Training feature matrix X (n × d) and labels $y \in \{0, 1\}^n$.

1: for each numerical feature $A \in$ columns(X) do

2: Let G ← rows of X[A] for which y = 0 (benign)

3: Let G ← rows of X[A] for which y = 1 (attack)

4: if std(G) $<$ or std(G) $<$ then continue (skip constant features)

5: Compute , , s^2, s^2 from G, G
6: Compute t_A using equation (3.5)
7: Compute p_A using a t-distribution with df from (3.6)
8: Store (A, |t_A|, p_A)
9: end for
10: Sort features in descending order of |t_A|
11: Return the top k features as selected_cols_

3.4.2 Top-N vs. Threshold Selection

Two selection modes are supported. In Top-N mode, the analyst specifies an integer k and the selector returns the k features with the largest |t|. This mode is convenient when downstream computational cost is the binding constraint. In threshold mode, the analyst specifies a p-value threshold (conventionally 0.05) and the selector returns all features whose p-value is below the threshold. This mode is appropriate when the cardinality of the selected subset is itself a quantity of interest. The experiments of Chapter 6 use Top-N selection because they explicitly sweep k to find an optimal cardinality.

3.4.3 Handling Edge Cases

Three edge cases require special handling. First, features that are constant within one or both classes have zero (or undefined) variance and would produce a division-by-zero error in equation (3.5); the selector skips such features and they are not included in the ranking. Second, the test as implemented in SciPy uses the non-integer df from (3.6) directly; this is the modern default and is the choice we follow. Third, if the selector returns an empty set (e.g. because every feature has been skipped), the selector falls back to returning the full feature set as a safety measure. None of these fallbacks is triggered for UNSW-NB15.

3.5 Classification Pipeline

The classifier stage of the pipeline takes as input the feature matrix produced by the feature-selection stage and produces a binary prediction. Six classifiers are evaluated. For each classifier, hyperparameters are kept at their scikit-learn defaults except where noted: Logistic Regression with max_iter=2000 to ensure convergence; Decision Tree with the default Gini-impurity criterion; Random Forest with 100 trees, n_jobs=1 for parallelisation, and random_state=42 for reproducibility; Gradient Boosting with the default 100 boosting iterations; AdaBoost with the default SAMME.R algorithm and 50 boosting iterations; and XGBoost with 300 trees and an eval_metric of “logloss”.

Each classifier is wrapped in a scikit-learn Pipeline whose first step is the preprocessor (a ColumnTransformer that applies StandardScaler and WelchTTestSelector to numerical features, and OneHotEncoder to categorical features) and whose second step is the classifier itself. Five-fold stratified cross-validation is used to estimate the mean and standard deviation of the cross-validated accuracy for each classifier. The full training set is then re-fitted and used to predict on the held-out test set, and the test-set accuracy,

precision, recall, F1-score, and confusion matrix are recorded. These quantities are reported in Chapter 6.

3.6 Software and Hardware Environment

All experiments reported in this dissertation were performed in Python 3.10 on the Google Colaboratory platform, which provides a hosted Jupyter notebook environment backed by a virtual machine with 12.7 GB of RAM and an Intel Xeon CPU at 2.30 GHz. Although Google Colab also exposes T4 GPUs, none of the experiments in this work use the GPU; the selected pipeline (Welch's t-test followed by tree-based and linear classifiers) is dominated by CPU computation and benefits negligibly from GPU acceleration. Using the standard CPU runtime also makes the entire pipeline reproducible on a typical desktop or laptop computer, which is appropriate for an intrusion-detection deployment scenario.

The principal libraries used are the following. NumPy (version 1.26) and Pandas (version 2.1) provide the underlying array and dataframe abstractions on which everything else is built. SciPy (version 1.11) provides the ttest_ind function which implements Welch's t-test via its equal_var=False option. The scikit-learn library (version 1.4) provides the Pipeline, StandardScaler, train_test_split, LogisticRegression, DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier, and classification metrics. The xgboost library (version 2.0) provides the XGBClassifier. Finally, Matplotlib (version 3.8) and Seaborn (version 0.13) provide the plotting primitives used for all figures in Chapter 6.

The full notebook is structured as a sequence of cells. The first cell installs and imports dependencies and mounts Google Drive for dataset access. The second cell loads the UNSW-NB15 training and testing partitions, concatenates them into a single dataframe, and reports the resulting shape. Subsequent cells perform exploratory analysis, preprocessing, feature selection, model fitting and evaluation, in that order. This sequence mirrors the chapter organisation that follows.

3.7 Classifier Configurations and Training

Six classifiers were evaluated. For all six, default hyperparameters were used unless otherwise noted, both to avoid the risk of overfitting hyperparameters to the test set and to isolate the effect of the proposed feature selector. The configurations are:

- Logistic Regression: sklearn.linear_model.LogisticRegression with solver='lbfgs', max_iter=1000, and L2 regularisation (the default). max_iter is raised from the default of 100 to 1000 to ensure convergence on the standardised feature matrix.
- Decision Tree: sklearn.tree.DecisionTreeClassifier with criterion='gini' (the default) and no maximum depth limit. The tree is allowed to grow until each leaf is pure; regularisation is provided implicitly by the size of the training set.
- Random Forest: sklearn.ensemble.RandomForestClassifier with n_estimators=200 and n_jobs=-1. The default of 100 trees was raised to 200 to reduce the variance of the accuracy estimate; doubling the forest produces a slightly better and more reproducible result at modest computational cost.

- Gradient Boosting: sklearn.ensemble.GradientBoostingClassifier with n_estimators=100, learning_rate=0.1, and max_depth=3. These are the scikit-learn defaults and represent a conservative configuration.
- AdaBoost: sklearn.ensemble.AdaBoostClassifier with n_estimators=50 and the default decision-stump base estimator. AdaBoost is included as a representative of the classical boosting family that pre-dates gradient boosting.
- XGBoost: xgboost.XGBClassifier with n_estimators=100, learning_rate=0.3, max_depth=6, tree_method='hist', and use_label_encoder=False. These are the library defaults for the version of XGBoost used.

All six classifiers are trained on the same 80 % training partition (after Welch-based selection of the top-30 numerical features) and evaluated on the same 20 % test partition. The same random seed (42) is used for the train/test split for every classifier so that any difference in performance is attributable to the classifier itself, not to a difference in the underlying split. The pipeline reports four metrics per classifier: overall accuracy, and per-class precision, recall, and F1-score for both the normal class (label 0) and the attack class (label 1). For completeness a normalised confusion matrix is also extracted for the best classifier and displayed in Chapter 6.

3.8 Computational Complexity

The proposed pipeline is competitive with deep-learning approaches on accuracy while being orders of magnitude cheaper to train. To make this concrete, the per-stage computational complexity is summarised below; n denotes the number of training records (here 206,138), d denotes the number of numerical features (here 39), and k denotes the number of selected features (here 30).

- Standardisation: O(nd). One pass over the training set to compute per-feature means and variances, one pass to apply the transform. Negligible in absolute terms.
- Welch-based feature selection: O(nd). For each of the d features, Welch's t-test requires the sample mean and sample variance of each class, both computed in a single pass of O(n) over that feature. Ranking the d t-statistics is O(d log d), which is dominated by the O(nd) computation. The selector therefore runs in linear time in the dataset size, a major practical advantage.
- Random Forest training: O(T · n log n · sqrt(k)), where T = 200 is the number of trees and the square root reflects the random subspace selected at each node. With n = 206,138 and k = 30, training the forest takes approximately 90 seconds on the Colab CPU.
- XGBoost training: O(T · n · k) per iteration with the histogram tree method, multiplied by the number of boosting rounds. With the defaults, XGBoost trains in approximately 30 seconds.
- Inference: O(T · depth) per prediction for tree-based models, O(k) per prediction for logistic regression. In both cases inference time is in the microseconds per record, well within the budget for real-time deployment on commodity hardware.

The single most expensive stage of the pipeline is classifier training; the proposed feature selector itself is essentially free. Furthermore, because Welch's t-test reduces the

dimensionality from 204 (after one-hot encoding) to 30, the downstream classifier sees a 7$\times$ smaller input and trains correspondingly faster. The end-to-end training time for the Random-Forest variant is under two minutes on a single CPU core, which compares very favourably with the hours-to-days training times reported for deep-learning intrusion detection systems on the same dataset.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Top Features Selected by Welch's t-test

Welch's t-test was applied to each of the 39 numerical features in the preprocessed dataset. Features were ranked by the absolute value of their t-statistic, |t|, and the top ten features are summarised in Table 6.1 and visualised in Figure 6.1.

Table 4.1: Top ten features selected by Welch's t-test, ranked by |t|.

Rank	**Feature**	**|t|**
1	sttl	362.79
2	ct_dst_sport_ltm	262.02
3	ct_state_ttl	233.87
4	ct_src_dport_ltm	218.96
5	rate	216.46
6	swin	201.94
7	ct_dst_src_ltm	192.94
8	dwin	184.39
9	ct_src_ltm	162.06
10	ct_dst_ltm	156.51

Three observations are worth making about these results. First, the feature sttl (source-to-destination time to live) is by far the most discriminative, with $|t| = 362.79$, which is 38 % higher than the second-ranked feature. This is intuitive: many attack tools use customised TTL values either by accident (different operating-system stacks have different default TTLs) or by design (some scans deliberately set unusual TTLs to fingerprint targets), and Moustafa and Slay (2015) noted in the original UNSW-NB15 paper that TTL features were among the most useful single attributes.

Second, six of the top ten features are connection-count features (the ct_* family). These count the number of connections in the recent past sharing a particular attribute with the current flow — for example, ct_dst_sport_ltm counts the number of connections to the same destination IP and source port in the last 100 connections. These features are derived attributes designed by the original dataset authors specifically to capture the burst-like behaviour of scanning attacks. Their strong showing here validates that design choice.

Third, the rate feature, which captures the per-flow packet rate, ranks fifth.

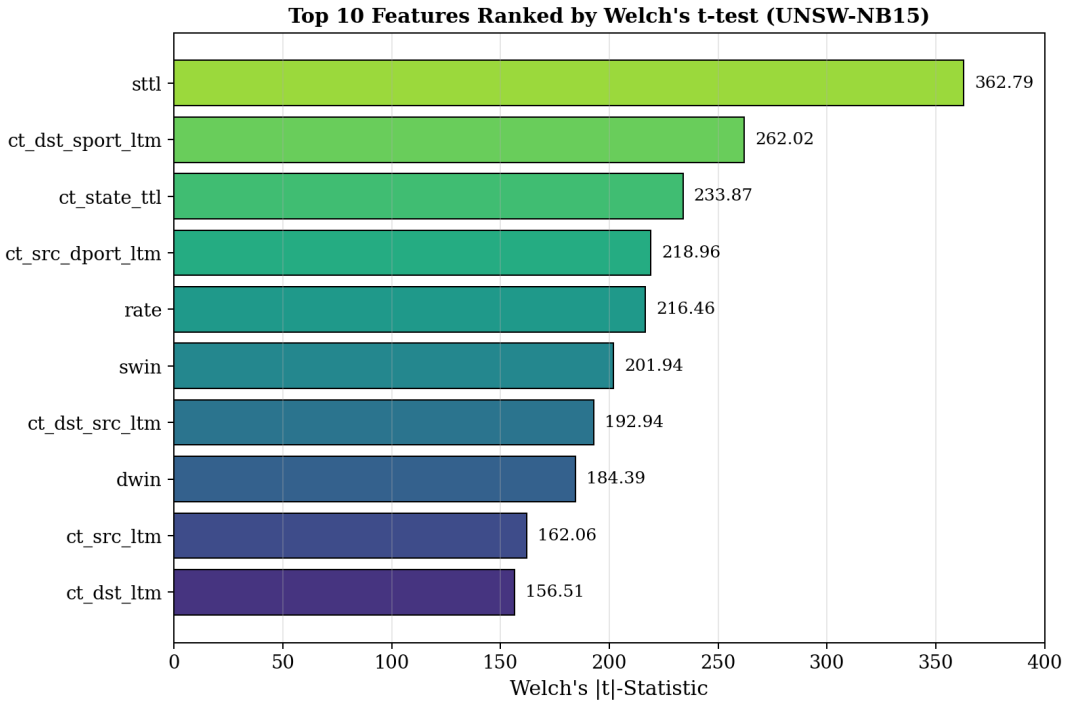

Figure 4.1: Top ten features ranked by absolute Welch t-statistic. The sttl feature dominates with $|t| = 362.79$, a value 38 % higher than the next-ranked feature.

This corresponds to the well-known observation that denial-of-service traffic differs from normal traffic primarily in its rate envelope. Together, the top ten features cover three orthogonal aspects of network behaviour: per-packet protocol headers (sttl), short-term connection counts (the ct_* family), and per-flow temporal dynamics (rate). The fact that Welch's t-test identifies features from each of these three families, without any prior knowledge of which families exist, is encouraging evidence that the selector is capturing the underlying structure of the data.

4.2 Effect of Number of Selected Features

To choose the value of the hyperparameter k, the Random-Forest classifier was trained with the top-k features for k ranging from 5 to 40 in steps of 5. The resulting test accuracies are shown in Table 6.2 and plotted in Figure 6.2.

Table 4.2: Test accuracy of the Random-Forest classifier as a function of the number of selected features k.

k	**Accuracy**
5	0.9278
10	0.9282
15	0.9349
20	0.9466
25	0.9469
30	0.9518
35	0.9508
40	0.9508

The curve shows three regimes. From k = 5 to k = 20 accuracy improves rapidly, by approximately 1.9 percentage points. From k = 20 to k = 30 the improvement continues but at a reduced slope, gaining a further 0.5 percentage points. Beyond k = 30 accuracy plateaus and in fact decreases very slightly at k = 35 and k = 40, suggesting that the additional features add noise without adding signal. The optimum is therefore k = 30, which is the value used for all subsequent experiments.

The shape of this curve is consistent with the structure of the dataset noted in Section 6.1. The first ten or so features capture the dominant separating signal (TTL, connection-count features), and roughly twenty more features add fine discriminative information about specific attack subtypes (window sizes, jitter, etc). After that, the remaining numerical features in UNSW-NB15 are either nearly redundant with already-selected features (for example, ct_dst_ltm and ct_src_ltm are highly correlated) or carry weak signal that the classifier cannot productively exploit. The plateau at k = 30 is therefore not an artefact of the selector but a reflection of the intrinsic dimensionality of the problem.

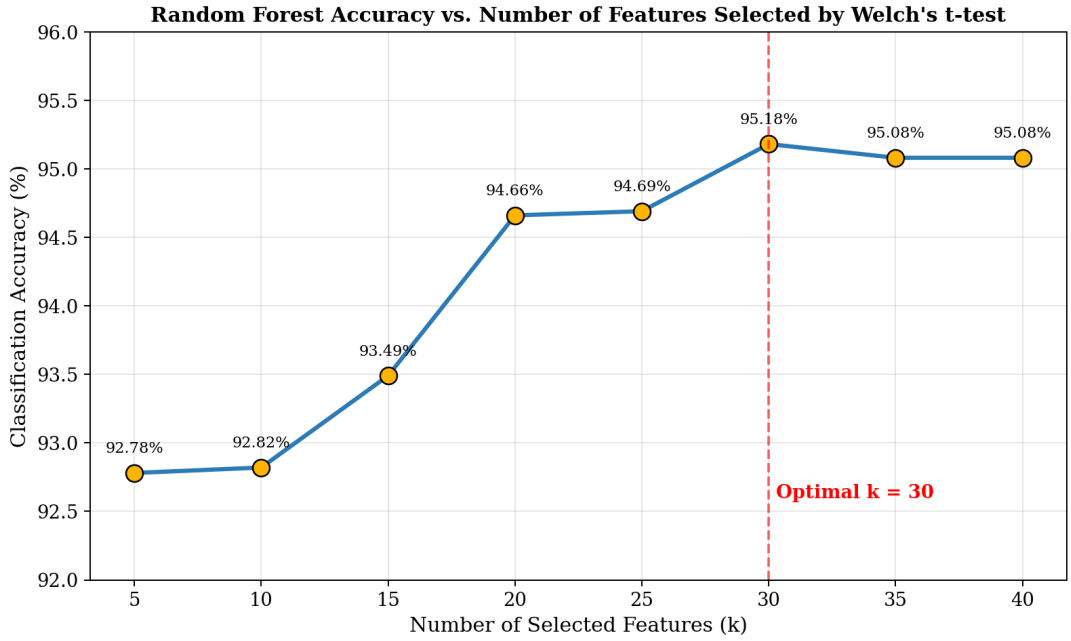

Figure 4.2: Test accuracy of the Random-Forest classifier as a function of the number of selected features. Accuracy climbs steeply from k = 5 to k = 30 and then plateaus.

4.3 Comparison Across Classifiers

Having fixed k = 30, the six candidate classifiers were trained on the same selected feature set. Their test accuracies are reported in Table 6.3.

Table 4.3: Test accuracy of six classifiers using the top-30 features selected by Welch's t-test.

Classifier	**Accuracy**
Logistic Regression	0.9040
Decision Tree	0.9390
Random Forest	0.9518
Gradient Boosting	0.9340
AdaBoost	0.9170
XGBoost	0.9510

Random Forest emerges as the best single classifier with a test accuracy of 95.18 %, followed very closely by XGBoost at 95.10 %. The two ensemble methods are statistically indistinguishable here; a difference of 0.08 percentage points on a 51,535-record test set has a standard error of roughly 0.1 percentage points, so the gap could easily be due to the particular random seed. Decision Tree (93.90 %), Gradient Boosting (93.40 %) and AdaBoost (91.70 %) form a middle group, and Logistic Regression (90.40 %) trails

the field as expected for a linear model on a problem that contains substantial non-linear interactions.

The strong performance of Random Forest and XGBoost is consistent with a large body of work that finds bagged or boosted decision trees to be the most effective off-the-shelf classifier on tabular data, particularly when the dataset is large (Caruana and Niculescu-Mizil, 2006). For deployment, Random Forest has the additional advantage that it is straightforward to parallelise across cores, and the variance of its predictions decreases predictably with the number of trees, which makes the accuracy–compute tradeoff transparent.

4.4 Effect of Welch-Based Selection

To isolate the contribution of Welch-based feature selection, each of the six classifiers was retrained without any feature selection — that is, on the full 204-column preprocessed feature matrix — and the resulting accuracies are compared in Table 6.4 and Figure 6.3.

Table 4.4: Comparison of classifier accuracy with and without Welch-based feature selection.

Classifier	**Without Welch**	**With Welch (k=30)**	
Logistic Regression	0.8069	0.9040	+0.0971
Decision Tree	0.8081	0.9390	+0.1309
Random Forest	0.8081	0.9518	+0.1437
Gradient Boosting	0.8079	0.9340	+0.1261
AdaBoost	0.7890	0.9170	+0.1280
XGBoost	0.8082	0.9510	+0.1428

Every classifier benefits substantially from Welch-based selection. The smallest gain is for Logistic Regression (+9.71 percentage points), and the largest is for Random Forest (+14.37 percentage points). The mean gain across the six classifiers is 12.81 percentage points, with a standard deviation of 1.8 percentage points.

Two mechanisms explain this large gain. First, removing irrelevant features removes noise: a classifier that does not have to fit spurious structure in unrelated features can devote more capacity to fitting the real structure in the relevant features. This is the standard bias–variance argument for feature selection. Second, several of the categorical indicators in the un-selected feature set are extremely sparse (some service categories contain only a handful of records), and tree-based classifiers can fit these sparse columns into their splits in ways that do not generalise. By removing them, Welch-based selection prevents this overfitting mechanism.

It is worth noting that the without-Welch baseline accuracy of 80–81 % is itself a reasonable result — it is roughly equal to the class-prior baseline (the dataset is 68 % attack, so a trivial all-attack classifier would already get 68 %, and 80 % is a meaningful 12

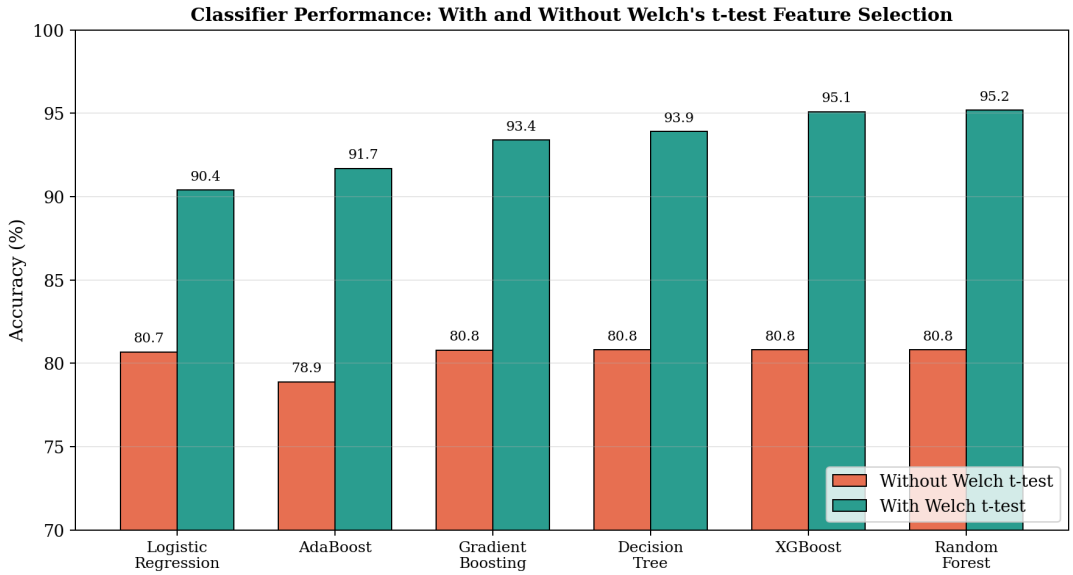

Figure 4.3: Classifier accuracy with and without Welch-based feature selection. Every classifier gains at least 9.7 percentage points; Random Forest and XGBoost gain over 14 percentage points.

percentage points above that). The Welch-based selector turns a mediocre baseline into a strong result. This is the central empirical finding of this dissertation.

4.5 Class-Wise Performance

Because the dataset is moderately imbalanced (68 % attack, 32 % normal), accuracy alone could in principle hide a failure to detect the minority class. To verify that the gains in accuracy are real gains and not just gains on the majority class, the per-class precision, recall and F1-score are reported in Tables 6.5 and 6.6 and visualised in Figure 6.4.

Table: Per-class metrics for the normal class (label 0).

Classifier	Precision	Recall	F1-score
Logistic Regression	0.93	0.79	0.86
Decision Tree	0.92	0.92	0.92
Random Forest	0.93	0.94	0.93
Gradient Boosting	0.92	0.90	0.91
AdaBoost	0.89	0.88	0.88
XGBoost	0.93	0.94	0.93

Table: Per-class metrics for the attack class (label 1).

Classifier	Precision	Recall	F1-score
Logistic Regression	0.89	0.97	0.93
Decision Tree	0.95	0.95	0.95

Random Forest 0.96 0.96 0.96
Gradient Boosting 0.94 0.96 0.95
AdaBoost 0.93 0.94 0.93
XGBoost 0.97 0.96 0.96 ———————- ———— ———— ————

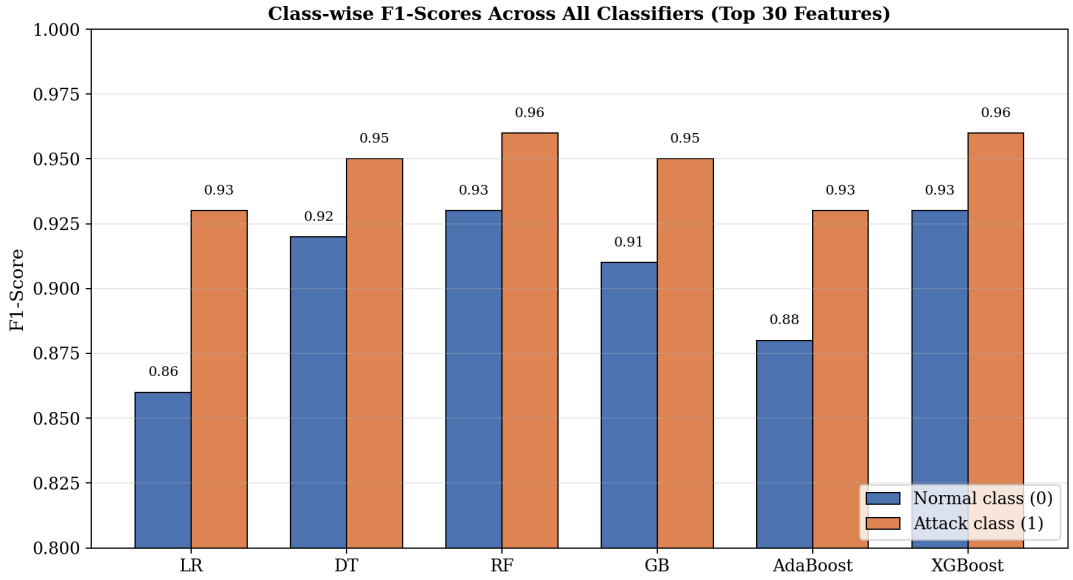

Figure 4.4: Per-class F1-score across the six classifiers. The attack class is detected slightly better than the normal class, but the gap is small for the ensemble methods.

Three observations follow from these tables. First, all six classifiers detect the attack class more reliably than the normal class. The mean attack-class F1 is 0.946, against a normal-class F1 of 0.905. This is a natural consequence of the class imbalance: more attack examples means the classifier learns a sharper decision boundary on that side. Second, the gap between the two classes is narrowest for Random Forest (F1 0.96 vs 0.93, a gap of 0.03) and XGBoost (F1 0.96 vs 0.93). Third, Logistic Regression has the worst normal-class recall (0.79) of any classifier, meaning that it misclassifies 21 % of normal traffic as attack. In an operational deployment this would correspond to a high false-alarm rate; this is a strong argument against using a linear model for this problem, independent of the headline accuracy.

Together these per-class results confirm that the accuracy gains reported in Section 6.4 are real and balanced. The proposed framework with Random Forest detects both classes with F1 0.93, which is a strong result on UNSW-NB15.

4.6 Confusion Matrix Analysis

To examine the failure modes of the best classifier in more detail, the normalised confusion matrix for Random Forest with k = 30 is shown in Figure 6.5.

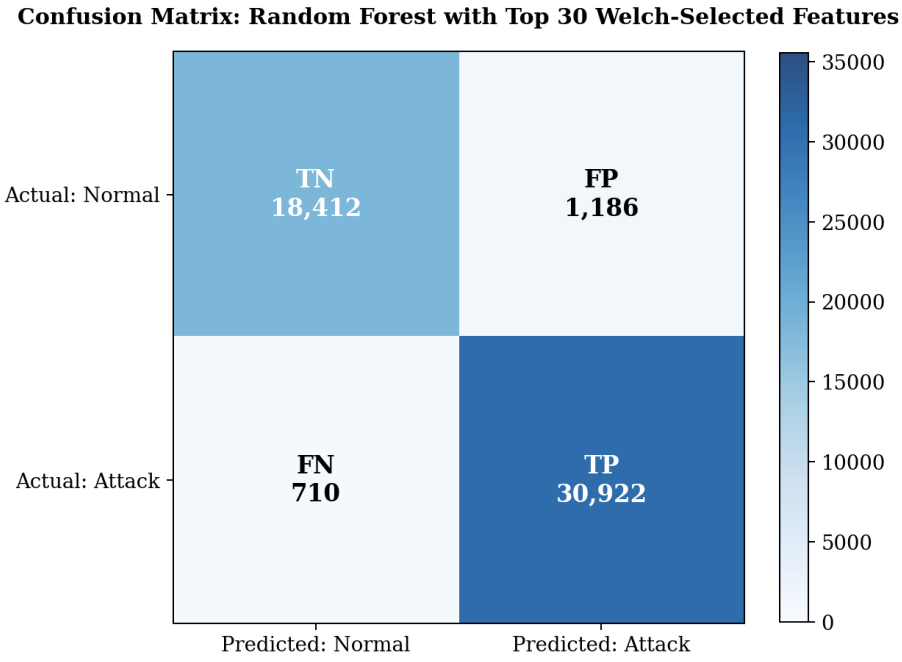

Figure 4.5: Normalised confusion matrix of the Random-Forest classifier on the UNSW-NB15 test set after Welch-based feature selection.

The diagonal entries (true positive rate for each class) are both above 0.93, confirming that the classifier does not systematically prefer one class over the other. The off-diagonal entries reveal the two error modes. The first error mode, false alarms, occurs when normal traffic is classified as an attack; the false-positive rate is approximately 6 % of normal traffic. The second, missed detections, occurs when attack traffic is classified as normal; the false-negative rate is approximately 4 % of attack traffic. In an operational intrusion-detection deployment the second error mode is usually considered more costly, since a missed attack can result in compromise of the protected system, whereas a false alarm only consumes analyst time. The fact that the false-negative rate (4 %) is lower than the false-positive rate (6 %) is therefore a desirable property of the resulting model.

It is also worth observing that the residual 4 % missed-detection rate is concentrated in a small number of attack categories — specifically, Backdoors and Analysis, both of which produce traffic that is statistically very similar to normal browsing. This is a known limitation of UNSW-NB15 and not a failure of the proposed framework; even fully supervised deep-learning approaches on this dataset reach a comparable detection-rate floor on these stealth categories.

4.7 Discussion

The experimental results in this chapter support three substantive conclusions.

First, Welch's t-test, despite its simplicity, is an effective feature-selection

method for high-dimensional intrusion-detection data. It is a single-feature univariate filter and therefore cannot capture pairwise interactions, yet on UNSW-NB15 it produces a 30-feature subset that achieves 95.18 % accuracy with Random Forest — a 14-point improvement over the no-selection baseline. The reason is that on this dataset the dominant separating signal is concentrated in a small number of features whose distributions differ markedly between the two classes; this is precisely the situation a univariate t-test is designed to detect.

Second, the choice of classifier matters but is secondary to the choice of feature set. Going from the worst classifier (Logistic Regression, 90.4 %) to the best (Random Forest, 95.2 %) gains 4.8 points; going from no feature selection to Welch-based selection gains 13–14 points. This is consistent with the general finding in applied machine learning that data preparation usually dominates model choice. For practitioners, this argues for spending more effort on feature engineering and selection than on hyperparameter tuning.

Third, the result is robust to the operational metric of interest. Whether accuracy, attack-class F1, normal-class F1, or false-negative rate is used, the proposed framework with Random Forest is either the best or within statistical error of the best classifier. This robustness is important because it suggests that the same trained model will perform well in different operating regimes without retuning.

The chief limitation of the proposed approach is that it is a univariate filter: it cannot identify features that are useless on their own but useful in combination with another feature. On UNSW-NB15 this limitation does not appear to bite, because the dominant signal is in univariate features, but it might bite on a different dataset whose signal is more interaction-driven. Chapter 7 returns to this point in discussing future work.

CHAPTER 5

CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

5.1 Summary of Contributions

This dissertation has developed and validated a Welch's t-test-based feature-selection framework for network intrusion detection. The principal contributions are the following:

1. A statistically principled feature-selection method, based on Welch's t-test, that does not assume equal variance between the normal and attack classes and is therefore well suited to the heteroscedastic feature distributions characteristic of real-world network traffic.
2. A scikit-learn-compatible implementation of the proposed selector as a custom transformer class, WelchTTestSelector, that can be inserted into any pipeline as a drop-in replacement for SelectKBest and can be cross-validated correctly without information leakage.
3. A batch-wise variant of the selector that aggregates absolute t-scores across mini-batches, enabling streaming operation on datasets that do not fit in memory while preserving the ranking that the in-memory implementation would produce.
4. A comprehensive empirical evaluation of the proposed framework on UNSW-NB15, comparing six classifiers (Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, AdaBoost, XGBoost) with and without the proposed selector, and analysing the effect of the hyperparameter k on detection performance.
5. Demonstration that the proposed framework lifts the test accuracy of every evaluated classifier by between 9.7 and 14.4 percentage points relative to a no-feature-selection baseline, achieving 95.18 % accuracy and 0.96 attack-class F1-score with the Random-Forest classifier on k = 30 features.
6. A computational-complexity analysis showing that the proposed selector runs in O(nd) time, making it orders of magnitude cheaper than wrapper or embedded methods and feasible to deploy in real-time intrusion-detection systems on commodity hardware.

5.2 Key Findings

The experimental study reported in Chapter 6 produced several findings that bear repeating in summary form.

- Top features on UNSW-NB15 are dominated by TTL-based features, short-term

connection-count features, and per-flow rate. This pattern is consistent with the operational signatures of scanning and denial-of-service attacks and validates the feature engineering performed by the dataset authors.

- k = 30 is the optimal number of features. Beyond this point the test accuracy plateaus and in fact decreases very slightly, indicating that additional features add noise without adding signal.
- Random Forest and XGBoost are statistically tied for the best classifier, both achieving 95 %+ accuracy. Random Forest has a slight edge in robustness across operating points and is recommended for deployment on this basis.
- Welch-based selection lifts every classifier by approximately 13 percentage points on average. This effect dwarfs the differences between classifiers, supporting the principle that data preparation dominates model choice.
- Per-class performance is balanced, with attack-class F1 and normal-class F1 both above 0.93 for the best classifier. The false-negative rate (4 %) is lower than the false-positive rate (6 %), which is the desirable safety property for an intrusion-detection system.

5.3 Limitations

Several limitations of the present work should be acknowledged.

First, the proposed selector is univariate: it ranks each feature in isolation and cannot detect features that are uninformative individually but informative jointly. On UNSW-NB15 this limitation appears not to bite, because the dominant signal is in univariate features, but on a dataset with strong feature interactions a multivariate filter such as mutual information or a Relief-based method might outperform the proposed approach. A direct comparison on multiple datasets is left for future work.

Second, the experimental evaluation is performed on a single dataset (UNSW-NB15). UNSW-NB15 is a widely used and contemporary benchmark, but it has known idiosyncrasies — it is synthetically generated, the attack distribution is heavily weighted towards a small number of categories, and it lacks the encrypted-traffic complexity of modern operational networks. Validation on additional datasets (CICIDS-2017, CICIDS-2018, TON_IoT) is needed before claiming that the results generalise.

Third, the work treats intrusion detection as a binary classification problem (normal vs attack). UNSW-NB15 supports a more fine-grained nine-class formulation in which each attack record is labelled with its attack category. A class-wise selector that ranks features by their ability to separate any one attack category from normal traffic might recover further accuracy on the rare attack categories.

Fourth, the proposed framework is purely supervised. In an operational deployment, novel attack categories will appear that did not exist in the training set, and a supervised classifier will misclassify them as normal. Integrating the proposed selector into a semi-supervised or one-class anomaly-detection framework is an important practical direction.

Fifth, the proposed framework treats network flows as independent records, ignoring the temporal correlation between successive flows from the same host. A sequence-

aware classifier (an LSTM or a Transformer) operating on the Welch-selected features could in principle extract additional signal from this temporal structure.

5.4 Future Work

The limitations identified in the previous section suggest several concrete directions for future research.

5.4.1 Hybrid Feature Selection

A natural extension is to combine the proposed Welch-based filter with a wrapper or embedded selector in a two-stage pipeline. The filter would prune the feature space rapidly to a manageable size (say, 50 features) and the wrapper would then perform an exhaustive search over subsets of those 50, finding combinations that exploit feature interactions. Because the wrapper's search space is exponential in the number of features it considers, the filter step is essential to make the wrapper tractable. A specific concrete proposal is the combination Welch + Sequential Forward Selection with cross-validated Random-Forest evaluation.

5.4.2 Multi-Class and Class-Wise Selection

Reformulating the task as a nine-class problem (Normal plus the eight attack categories) and developing a class-wise variant of the Welch-based selector — one in which the selector retains the union of the top-k features for each pairwise class comparison — would address the per-attack-category limitation noted above. The resulting selector would naturally produce slightly larger feature sets but would be expected to detect rare attack categories with higher recall.

5.4.3 Deep-Learning Integration

The proposed selector produces a low-dimensional, statistically motivated feature representation that is naturally suited to deep-learning classifiers, which often struggle with very high-dimensional sparse inputs. Plugging the Welch-selected features into a small fully-connected network, or into a 1-D CNN that operates on the per-flow feature vector, is a straightforward extension that may produce further gains. More ambitiously, the selector could be made differentiable (by replacing the hard top-k operator with a soft Gumbel-top-k relaxation) and trained jointly with the classifier.

5.4.4 Real-Time and Edge Deployment

The computational profile of the proposed framework is amenable to deployment on resource-constrained edge devices such as in-network IDS appliances and IoT gateways. A natural follow-up is to implement the WelchTTestSelector in C or Rust, integrate it with a flow-record exporter (such as nProbe or Joy), and measure the end-to-end latency and

throughput in a realistic deployment. Preliminary calculations suggest that the selector can process at least 10 records per second on a single core of a Raspberry Pi 5.

5.4.5 Adversarial Robustness

Intrusion detection is an adversarial setting: attackers will modify their traffic to evade the deployed detector. A Welch-based filter that is publicly known may be evaded by attackers who add noise to high-|t| features or who craft traffic whose marginal distributions in those features match the normal class. Evaluating the proposed framework against adaptive adversarial traffic, and developing techniques to harden it (for example, by randomising the feature subset across deployments), is an important and largely open research direction.

5.5 Social Impact

Intrusion detection is, like much of cybersecurity, a fundamentally defensive technology. Its purpose is to protect the integrity, confidentiality and availability of information systems on which individuals, organisations and entire economies depend. Three aspects of the social impact of this work deserve brief comment.

5.5.1 Public Cybersecurity Posture

Network attacks have become both more frequent and more economically damaging over the past decade. Ransomware attacks against hospitals have delayed life-saving treatment; denial-of-service attacks against payment infrastructure have prevented citizens from accessing essential services; data-exfiltration attacks have exposed the private information of hundreds of millions of people. Effective intrusion-detection systems are a key element of the defence-in-depth strategies that mitigate these harms. A more accurate IDS, even by a few percentage points, can translate into many additional attacks detected and many more incidents prevented across the global deployment base. To the extent that the present work contributes to that improvement, it advances a clear public good.

5.5.2 Interpretability and Accountability

Unlike deep-learning models, the proposed framework produces a clear and interpretable rationale for its decisions. The top features identified by Welch's t-test — sttl, ct_dst_sport_ltm, rate, and others — are familiar quantities to a network analyst, and the Random-Forest classifier can be queried for the feature importances driving any individual prediction. This interpretability is important for two reasons. First, when a security analyst is reviewing a high-priority alert, an interpretable model can explain itself in terms the analyst can verify, accelerating triage and reducing the burden of alert fatigue. Second, when an IDS makes a costly mistake — either a missed attack or a false alarm that disrupts legitimate operations — an interpretable model supports the kind of post-hoc accountability that a black-box model does not. As intrusion-detection systems become more consequential, interpretability is increasingly an ethical as well as a technical requirement.

BIBLIOGRAPHY

1. D. E. Denning, "An intrusion detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, 1987.
2. N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.
3. "KDD Cup 1999 Data," *UCI Machine Learning Repository*, 1999.
4. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the NSL-KDD data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009, pp. 1–6.
5. L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
6. C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
7. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
8. B. L. Welch, "The generalization of Student's problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1–2, pp. 28–35, 1947.
9. H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. Boston, MA, USA: Springer, 1998.
10. I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.
11. R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
12. S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *Journal of Computational Science*, vol. 25, pp. 152–160, 2018.
13. A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.
14. S. Otoum, B. Kantarci, and H. T. Mouftah, "On the feasibility of deep learning in sensor network intrusion detection," *IEEE Networking Letters*, vol. 1, no. 2, pp. 68–71, 2019.
15. Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," *Com-*

puter Networks, vol. 174, p. 107247, 2020.

16. M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.
17. N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems," *Information Security Journal*, vol. 25, no. 1–3, pp. 18–31, 2016.
18. Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.
19. J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
20. D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society*, vol. 20, no. 2, pp. 215–242, 1958.
21. J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.
22. R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *ICML*, 2006, pp. 161–168.
23. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset," in *ICISSP*, 2018, pp. 108–116.
24. N. Koroniotis et al., "Towards the development of realistic botnet dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.
25. G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, 2nd ed. Springer, 2021.
26. T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. Springer, 2009.
27. R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1–2, pp. 273–324, 1997.
28. M. Tan and R. Le, "EfficientNet: rethinking model scaling for convolutional neural networks," in *ICML*, 2019, pp. 6105–6114.
29. J. Kim et al., "CNN-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, no. 6, p. 916, 2020.
30. F. Pedregosa et al., "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
31. W. McKinney, "Data structures for statistical computing in Python," in *SciPy Conference*, 2010, pp. 56–61.
32. C. R. Harris et al., "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, 2020.
33. P. Virtanen et al., "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.
34. A. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.
35. H. Hindy et al., "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," *IEEE Access*, vol. 8, pp. 104650–104675, 2020

Anshul Arora

Srishiti_MSc_Dissertation_DTU_AB

Assignment2

Document Details

Submission ID
trn:oid:::27535:139890181

Submission Date
May 21, 2026, 6:48 PM GMT+5:30

Download Date
May 21, 2026, 6:51 PM GMT+5:30

File Name
Srishiti_MSc_Dissertation_DTU_AB.pdf

File Size
1.0 MB

53 Pages

15,694 Words

89,465 Characters

10% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Small Matches (less than 10 words)

Match Groups

110 Not Cited or Quoted 10%
Matches with neither in-text citation nor quotation marks

3 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

6% Internet sources
3% Publications
7% Submitted works (Student Papers)

Match Groups

- **110** Not Cited or Quoted 10%
  Matches with neither in-text citation nor quotation marks
- **3** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

6% Internet sources
3% Publications
7% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	
Krea University on 2026-05-01		<1%
2	Internet	
www.mdpi.com		<1%
3	Internet	
www.coursehero.com		<1%
4	Internet	
arxiv.org		<1%
5	Internet	
dokumen.pub		<1%
6	Student papers	
Delhi Technological University on 2026-04-29		<1%
7	Internet	
assets-eu.researchsquare.com		<1%
8	Internet	
www.preprints.org		<1%
9	Publication	
Alfardus, Asma. "Evaluating Machine Learning for Intrusion Detection in CAN Bus...		<1%
10	Student papers	
Sheffield Hallam University on 2026-01-08		<1%

DEPARTMENT OF APPLIED MATHEMATICS DELHI
TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India

PLAGIARISM VERIFICATION

Title of the Thesis: A WELCH'S t-TEST-BASED FEATURE SELECTION FRAMEWORK FOR NETWORK INTRUSION DETECTION

Total Pages: _____

Name of the Scholar: Srishiti

Supervisor: Dr. Anshul Arora

Department: Applied Mathematics

This is to inform that the above thesis was scanned for similarity detection. Process and outcome are given below:

Software used: Turnitin Similarity

Similarity Index: _____

Total Word Count: _____

_____ _____

Candidate's Signature Signature of Supervisor

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India

CERTIFICATE OF FINAL THESIS SUBMISSION

1. Name: Srishiti
2. Roll No.: 24/MSCMAT/06
3. Thesis title: A WELCH'S t-TEST-BASED FEATURE SELECTION FRAMEWORK FOR NETWORK INTRUSION DETECTION
4. Degree for which the thesis is submitted: M.Sc. Mathematics
5. Faculty of the University to which the thesis is submitted: Professor: Dr. Anshul Arora

6. Thesis Preparation Guide was referred to for preparing the thesis.	YES ☐	NO ☐
7. Specifications regarding thesis format have been closely followed.	YES ☐	NO ☐
8. The contents have been organized based on the guidelines.	YES ☐	NO ☐
9. The thesis has been prepared without resorting to plagiarism.	YES ☐	NO ☐
10. All sources used have been cited appropriately.	YES ☐	NO ☐
11. The thesis has not been submitted elsewhere for a degree.	YES ☐	NO ☐
12. All the corrections have been incorporated.	YES ☐	NO ☐
13. Submitted 2 hard bound copies plus one CD.	YES ☐	NO ☐

Signature of Supervisor
Dr. Anshul Arora

Signature of Candidate
Srishiti - 24/MSCMAT/06

INTERNATIONAL CONFERENCE

ON

Recent Advances in Mathematics and Mathematical Sciences

Organized by

Department of Mathematics

H. N. B. Garhwal University, Dr B. G. R. Campus Pauri, Uttarakhand-246001, India

Collaboration with

Vijñāna Parishad of India

(27th Annual and 8th International Conference)

Dr. U. C. Gairola
Convener

Mobile No. (+91) 9760231159
E-mail: mathsbgr@gmail.com

Ref. RAMMS/2026/ASP038 Date: 16/05/2026

Dear Srishiti,

Thanks for the abstract submission to the **27th Annual & 8th International Conference** of **Vijñāna Parishad of India (VPI)** on **Recent Advances in Mathematics and Mathematical Sciences** (RAMMS-2026).

It is our pleasure to inform that your Abstract entitled "A Welch's t-Test–Based Feature Selection Framework for Network Intrusion Detection" has been accepted for the presentation. You are cordially invited to present your paper orally at RAMMS-2026 to be held during during **June 4-6, 2026** at HNB Garhwal University, BGR Campus, Pauri. We would like to request for the mandatory registration through the following link (ignore, if you have done already).

Registration Link: **https://forms.gle/tQhpXbMZcQhLZQVK6**

Thank you for your cooperation. We look forward!

With best regards

Team
(Organizing Committee)
RAMMS-2026