JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

PERFORMANCE EVALUATION OF FEATURE SELECTION TECHNIQUES FOR MACHINE LEARNING-BASED INTRUSION DETECTION

HARISH VARMA J, S. RAGHUL, K.N.SIVAKUMAR
M.E. Computer Science and Engineering Student
M.P. Nachimuthu M. Jaganathan Engineering College

சுருக்கம்

வலையலமப்புப் பபோக்குவரத்தினைஃ ஏற்படும் தீங்கிலைக்கும் செயைப்போடுகலைக் கண்டறிவதினைஃ வலையலமப்பு ஊடுருவைப் கண்டறிதைத் முக்கியப் பங்கு வகிக்கிறF. இயந்திரக் கற்றைற் வலகப்படுத்திகைக் தோக்குதைத் முலறகலைக் கண்டறிய உதவுகின்றன. இருப்பினும், உைஃைீட்டு அம்ெங்கைின் எண்ணிக்லக கண்டறிதைத் Fைஃலியம் மற்றும் கணக்கீட்டுெஃ செைலவப் போதிக்கிறF. எனபவ, சபோருத்தமோன அம்ெத் பதர்வு முலறகலை மதிப்பிடுவபத இத்திட்டத்தின் பநோக்கமோகும். இந்த ஆய்வினைஃ யு.என்.எஸ்.டபினைஃயூ-என்.பி15 தரவுத்சதோகுப்பு பயன்படுத்தப்படுகிறF. இதினைஃ 1,75,341 பயிற்சிப் பதிவுகளும் 82,332 பெோதலனப் பதிவுகளும் உைஃைன. இரும இைக்கு வலகப்போடு மூைம் இயைப்போன பபோக்குவரத்F (0) மற்றும் தோக்குதைத் பபோக்குவரத்F (1) பவறுபடுத்தப்படுகின்றன. அலடயோைங்கோட்டி மற்றும் தோக்குதைத் வலகப் புைங்கைக் நீக்கப்பட்டு, வலகப்போட்டுத் தரவுகைோன சநறிமுலற, பெலவ மற்றும் நிலை ஆகியலவ ஒபர குறியீட்டு முலறயினைஃ மோற்றப்படுகின்றன. இதன் மூைம் 194 எண்

அம்ெங்கைச் சபறப்படுகின்றன. முடிசவடுக்கும் மரம், சீரற்ற கோடு மற்றும் பநரியைப் ஆதரவு திலெயன் இயந்திரம் ஆகிய மூன்று வலகப்படுத்திகைக் மதிப்பிடப்படுகின்றன. பகுப்போய்வு மோறுபோட்டுெ் பபோதலன, பரஸ்பரத் தகவை மற்றும் சீரற்ற கோட்டு அம்ெ முக்கியத்Fவம் ஆகிய முலறகைக் மூைம் 30 அம்ெங்கைப் பதர்ந்சதடுக்கப்படுகின்றன. அடிப்பலட முடிசவடுக்கும் மரம் 86.13% Fை்லியத்லதயும் 0.8844 எஃப்1 மதிப்சபண்லணயும் சபற்றF. சீரற்ற கோடு 87.03% Fை்லியத்லதயும் 0.8933 எஃப்1 மதிப்சபண்லணயும் சபற்றF.

30,000 பயிற்சிப் பதிவுகலைப் பயன்படுத்திய பநரியைப் ஆதரவு திலெயன் இயந்திரம் 83.84% Fை்லியத்லதப் சபற்றF. சீரற்ற கோட்டு அம்ெ முக்கியத்Fவ முலறயிை் பதர்ந்சதடுக்கப்பட்ட 30 அம்ெங்கைக் 87.14% Fை்லியத்லத வைங்கின. முடிசவடுக்கும் மரத்திை் பரஸ்பரத் தகவை முலற 86.20% Fை்லியத்லதப் சபற்றF. ஆனோை், இபத அம்ெத் பதர்வு முலறலயப் பயன்படுத்திய ஆதரவு திலெயன் இயந்திரம் 79.29% Fை்லியத்லத மட்டுபம சபற்றF. முடிவுகைக், அம்ெத் பதர்வின் தோக்கம் ஒவ்சவோரு வலகப்படுத்திக்கும் மோறுபடும் என்பலதக் கோட்டுகின்றன. அம்ெங்கலைக் குலறப்பF சிை மோதிரிகிை் பயிற்சி பநரத்லதக் குலறத்Fப் பபோFமோன செய்திறலனப் பரோமரிக்க உதவும். இருப்பினும், நம்பகமோன முடிவுகலை உறுதிப்படுத்த ஒபர மோதிரியோன பரிபோதலன அலமப்புகைக், பைமுலற பபோதலனகைக் மற்றும் கூடுதைத் மதிப்பீடுகைப் பதலவப்படுகின்றன.

ABSTRACT

Network intrusion detection is important for identifying malicious activities in network traffic. Machine learning classifiers can detect attack patterns, but the number of input features affects detection performance and computational cost. This project evaluates feature selection techniques to identify suitable feature subsets for network intrusion detection. The experiments use the UNSW-NB15 dataset, containing 175,341 training records and 82,332 testing records. The binary target label distinguishes normal traffic (0) from attack traffic (1). Identifier and attack-category fields are removed, while the categorical features proto, service, and state are one-hot encoded, producing 194 numeric features. Three classifiers—Decision Tree, Random Forest, and Linear Support Vector Machine (SVM)—are evaluated. ANOVA F-test, Mutual

Information, and Random Forest feature importance are used to select 30 features. The baseline Decision Tree achieved 86.13% accuracy and an F1-score of 0.8844. Random Forest achieved 87.03% accuracy and an F1-score of 0.8933. Linear SVM achieved 83.84% accuracy and an F1-score of 0.8647 using a stratified sample of 30,000 training records and the full test set. The Random Forest importance-selected configuration achieved 87.14% accuracy and an F1-score of 0.8942. For Decision Tree, Mutual Information selection achieved 86.20% accuracy and an F1-score of 0.8848. However, the SVM with Mutual Information-selected features achieved 79.29% accuracy and an F1-score of 0.8316. The results demonstrate that feature selection affects classifiers differently. Reducing features can decrease training time while maintaining similar performance for some models. However, performance may decline for others. Further validation using consistent experimental settings, repeated runs, and additional comparisons is necessary to confirm these findings.

Keywords: Network intrusion detection, UNSW-NB15, feature selection, ANOVA, Mutual Information, Random Forest, Decision Tree, Linear SVM.

CHAPTER 1 - INTRODUCTION

1.1 General introduction

Computer networks support communication and data exchange across organizations, institutions and digital services. The traffic carried by a network can include legitimate activity as well as attempts to disrupt services, gain unauthorized access or misuse network resources. An intrusion detection system (IDS) examines activity and identifies records or events that may correspond to malicious behaviour. This project studies a data-driven approach to binary network traffic classification using machine learning.

The project focuses on the effect of feature selection on classifier performance. A dataset may contain many measured attributes describing packets, bytes, timing, protocol state and connection history. Some features may be highly informative for the target label, while others may contribute less to a particular classifier. Feature selection ranks or scores the available predictors and retains a smaller subset. The aim is not simply to minimize the feature count, but to measure the trade-off between detection metrics and computational cost.

1.2 Background of network intrusion detection

In a binary intrusion-detection setting, each record is assigned to one of two classes: normal traffic or attack traffic. A supervised learning algorithm uses labelled training examples to learn a decision function and then predicts labels for previously unseen test records. The reliability of the evaluation depends on separating training and testing data, avoiding target leakage and reporting the metrics that reflect the operational objective.

In this implementation, label 0 represents Normal and label 1 represents Attack. The test set is evaluated after training, and the same test partition is used across the reported configurations. The resulting measurements describe performance on this supplied test file; they do not by themselves establish performance on every network, time or deployment environment.

1.3 Problem definition

The project investigates how three feature-selection methods—ANOVA F-test, Mutual Information and Random Forest feature importance—affect the performance of Decision Tree, Random Forest and Linear SVM classifiers on the UNSW-NB15 binary classification task. The principal comparison is between the full 194-feature input and a reduced set of 30 selected features. Performance is evaluated using accuracy, precision, recall, F1-score, false-positive rate and training time where recorded.

1.4 Need for study

Using a large input representation can increase model fitting cost and complicate interpretation. A reduced set of features may shorten training time and simplify the model input, but it can also remove information useful for classification. The effect may vary across algorithms because different classifiers respond differently to feature distributions, interactions and scaling. An empirical comparison is therefore required rather than assuming that feature reduction always improves performance.

1.5 Aim and objectives

The aim is to evaluate selected feature-selection techniques for machine learning-based binary intrusion detection using the UNSW-NB15 dataset.

- Prepare the supplied training and testing files for binary classification.
- Encoding categorical network-traffic attributes into a numeric feature matrix.
- Establish baseline performance for Decision Tree, Random Forest and Linear SVM.
- Select 30 features using ANOVA F-test, Mutual Information and Random Forest feature importance.
- Compare classification metrics and recorded training time for full-feature and reduced-feature configurations.
- Discuss the observed trade-offs and identify limitations and future validation needs.

1.6 Scope of the project

The implemented scope is offline, supervised, binary classification using the supplied UNSW-NB15 training and testing files. The predictors are prepared from the fields remaining after removing id, attack_cat and label. The categorical predictors proto, service and state are one-hot encoded. The implementation evaluates three classifiers and three feature-selection

approaches. The notebook includes visualizations for class distribution, numeric-feature correlation, feature importance, a Random Forest baseline confusion matrix, classifier metrics, feature-selection metrics and training time.

The project does not implement a live network sensor, streaming inference service, alert dashboard or automated response mechanism. Consequently, the report describes a machine-learning evaluation prototype rather than a production intrusion-detection deployment.

1.7 Organization of the thesis

Chapter 1 introduces the problem, aim, objectives and scope. Chapter 2 organizes the concepts and research context relevant to intrusion detection and feature selection; verified bibliographic sources must be added before submission. Chapter 3 explains the dataset, preprocessing, classifiers, feature-selection methods and metrics. Chapter 4 documents the implementation. Chapter 5 presents the recorded experimental results and their interpretation. Chapter 6 summarizes the findings, limitations and future work.

CHAPTER 2 - LITERATURE REVIEW

2.1 Introduction

This chapter frames the project in the areas of network intrusion detection, supervised machine learning and feature selection. The supplied notebook contains implementation evidence and experimental results, but it does not contain a complete collection of research papers. Therefore, this draft explains the concepts needed to understand the experiments and leaves the final paper-by-paper comparison to be completed with verified sources and in-text citations.

2.2 Intrusion-detection approaches

Intrusion detection can be organized broadly around rule or signature matching and behaviour-based anomaly identification. Signature-oriented methods compare observed activity with known patterns. Anomaly-oriented methods identify deviations from a learned or specified representation of expected activity [1]. Supervised machine learning belongs to the data-driven family when training records carry known labels. The current project evaluates supervised binary classification and does not compare a signature-based system.

2.3 Machine learning-based intrusion detection

A supervised classifier is trained from examples whose target labels are known. After training, it assigns a predicted label to each test record. Decision Trees form predictions through feature-based splits; Random Forest combines multiple decision trees; and Linear SVM learns a linear decision boundary in the feature space. These models provide different inductive biases, making them useful candidates for a comparative experiment.

2.4 Feature selection

Feature selection retains a subset of the existing predictors rather than transforming all predictors into a new latent representation. In this project, each selection method ranks features and retains the top 30. The reduced-feature model is then evaluated on the same supplied test set. Feature selection is distinguished from feature extraction: the selected names correspond to original encoded columns.

2.5 ANOVA F-test

The ANOVA F-test scoring function used by SelectKBest evaluates each feature's association with the class labels through a between-class versus within-class variation statistic. The implementation fits SelectKBest with f_classif and k=30 on the training features and labels only, then transforms the test matrix using the fitted selector [2]. The selected columns include numeric traffic attributes and one-hot encoded protocol, service and state indicators.

2.6 Mutual Information

Mutual Information estimates the statistical dependence between an input feature and the target. The notebook uses mutual_info_classif through SelectKBest with k=30. The selector is fitted on the training partition and applied to the test data. The resulting feature list contains traffic-volume, timing, TCP-related and connection-count attributes, together with the state_INT indicator.

2.7 Random Forest feature importance

A Random Forest model is fitted on all training features and its feature_importances_ values are used to rank the predictors. The top 30 columns are then passed to a Decision Tree experiment and a separate Random Forest experiment. This procedure measures the predictive results of a reduced subset ranked by a tree ensemble. The importance scores are model-dependent and should not be interpreted as causal effects.

2.8 Review of related research

For the final submitted thesis, this section should contain a table of verified research papers. For each paper, record the author and year, dataset, classification algorithm, feature-selection approach, evaluation metrics, main reported findings and limitations [3]. Do not insert unverified author names, paper titles or performance values. The supplied notebook alone does not substantiate claims about the numerical results of external studies.

2.8 Comparative Analysis of Feature Selection Techniques

Feature selection is an important preprocessing technique in machine learning-based intrusion detection because network traffic datasets contain multiple

attributes describing communication patterns, packet statistics, protocol information and

connection behaviour. Some attributes provide useful information for identifying malicious activity, while others may contribute limited predictive value or overlap with other features. Selecting a suitable subset can reduce the input dimensionality and potentially improve computational efficiency. However, the effectiveness of feature selection depends on the dataset and classification algorithm.

The ANOVA F-test, Mutual Information and Random Forest feature importance methods use different approaches to identify relevant features. The ANOVA F-test evaluates the relationship between individual features and class labels by comparing between-class and within-class variation. It is computationally straightforward and useful for ranking numerical predictors. However, because it evaluates features individually, it may overlook useful relationships that become apparent only when multiple attributes are considered together.

Mutual Information measures the statistical dependence between an individual feature and the target variable. It can identify statistical relationships that are not necessarily linear, making it useful for analysing complex network traffic characteristics. Nevertheless, the effectiveness of its ranking depends on the estimation procedure, and the highest-ranked individual features do not always form the best subset for every classifier.

Random Forest feature importance uses the importance scores generated by an ensemble of decision trees. It provides a model-based approach that can reflect relationships useful to tree-based classifiers. However, importance rankings may be influenced by feature characteristics and correlations among predictors. Therefore, the selected features should be evaluated empirically rather than assuming that the highest-ranked subset will always produce superior results.

The present study compares these three techniques by reducing the original 194-feature representation to 30 selected features. This approach provides a direct way to examine whether different feature-ranking strategies preserve useful classification information while reducing the number of input attributes.

2.9 Feature Selection and Classifier Compatibility

Feature selection and classifier choice should be considered together because different machine learning algorithms learn patterns differently. A feature subset that performs effectively with a Decision Tree may not produce the same results with a Random Forest or Linear Support Vector Machine (SVM). Consequently, evaluating feature selection across multiple classifiers provides a broader understanding of its effectiveness.

A Decision Tree creates predictions through successive feature-based splits. Removing less informative predictors can simplify the learning process and reduce training time. However, removing a feature that supports an important decision boundary may reduce classification

performance. In this study, the Mutual Information-selected Decision Tree achieved 86.20% accuracy and an F1-score of 0.8848, compared with 86.13% accuracy and an F1-score of 0.8844 for the baseline model.

Random Forest combines multiple decision trees to improve predictive capability. Feature selection can reduce the number of candidate predictors available to the ensemble, but the outcome depends on the information retained. In the present experiment, the Random Forest model using its own importance-ranked features achieved 87.14% accuracy and an F1-score of 0.8942. Its baseline achieved 87.03% accuracy and an F1-score of 0.8933. These results indicate a modest improvement in the recorded experiment.

Linear SVM learns a linear decision boundary and can be affected by feature representation and scaling. Removing useful predictors may affect its ability to distinguish normal traffic from attacks. The Mutual Information-selected SVM achieved 79.29% accuracy and an F1-score of 0.8316, compared with 83.84% accuracy and an F1-score of 0.8647 for the baseline. Its false-positive rate also increased.

These results demonstrate that feature selection does not have a uniform effect across classifiers. Therefore, a suitable selection method should be identified by evaluating its performance with the intended classification algorithm rather than relying exclusively on feature-ranking scores.

2.10 Evaluation Criteria and Practical Implications

The evaluation of intrusion detection systems requires multiple performance measures. Accuracy represents the proportion of correctly classified records, but it does not independently explain how effectively a model identifies attacks or avoids unnecessary alerts. Precision, recall, F1-score and false-positive rate provide additional information about classification behaviour.

Precision measures the proportion of predicted attacks that are genuinely attacks. Higher precision can reduce unnecessary security alerts and the workload associated with investigating false alarms. Recall measures the proportion of actual attacks correctly identified by the model. A model with low recall may fail to detect malicious activity. The F1-score combines precision and recall, while the false-positive rate measures the proportion of normal records incorrectly classified as attacks.

These metrics may produce different assessments of a model. For example, a classifier may achieve high recall while also generating numerous false positives. Although such a model detects many attacks, excessive false alarms can make operational monitoring difficult. Therefore, the selection of a suitable model should consider the security requirements and the relative consequences of missed attacks and false alerts.

Computational efficiency is another important consideration. Reducing the number of features can decrease the amount of data processed during model training and prediction. However, the

time required to calculate feature rankings should also be considered when evaluating the complete feature-selection pipeline. Training time may vary according to hardware, software configuration and system load.

The present study evaluates classification metrics alongside recorded training time. Nevertheless, the experiments are conducted offline and do not measure live detection latency, network throughput or resource consumption during continuous monitoring. These factors would require additional experiments before the models could be considered for practical deployment.

2.9 Research gap and motivation

The study addresses the need to compare different feature-selection techniques across multiple classifiers within a common binary intrusion detection task. Although feature selection can reduce dimensionality, its effect on classification performance varies according to the selected features and the algorithm being used. A comparative investigation of ANOVA F-test, Mutual Information and Random Forest feature importance therefore provides a practical basis for identifying suitable configurations.

Another important consideration is the balance between detection performance and computational efficiency. A smaller feature set may reduce processing requirements, but this benefit is valuable only when classification performance remains acceptable. The present work compares the original 194 features with subsets containing 30 features using the UNSW-NB15 dataset.

The study also highlights the importance of consistent experimental design. Decision Tree and Random Forest models use the complete training dataset, whereas the Linear SVM experiments use a stratified sample of 30,000 training records. Consequently, training-time comparisons across different classifiers must be interpreted cautiously. Repeated experiments and cross-validation would provide stronger evidence about the stability of the results.

The current investigation is limited to the supplied dataset, implemented preprocessing procedure and selected model configurations. It does not establish that any feature-selection method is universally superior. Instead, it identifies promising configurations and provides a foundation for further validation.

2.10 Summary

This chapter discussed the principles of feature selection and its application to machine learning-based intrusion detection. ANOVA F-test, Mutual Information and Random Forest feature importance provide different approaches to ranking predictors, while Decision Tree, Random Forest and Linear SVM respond

differently to reduced feature subsets. Evaluating these methods requires consideration of accuracy, precision, recall, F1-score, false-positive rate and computational cost. These concepts

establish the theoretical foundation for the methodology and experimental results presented in the following chapters.

CHAPTER 3 - THEORETICAL BACKGROUND AND METHODOLOGY

3.1 Proposed workflow

The experimental workflow consists of loading the supplied train and test files, checking dimensions and missing values, separating the binary target, removing identifier and attack-category fields, encoding categorical predictors, fitting baseline models, selecting features using training data, fitting reduced-feature models, evaluating predictions on the test set and recording metrics. The notebook also generates figures for exploration analysis and performance comparison [4].

3.2 Dataset description

The training spreadsheet contains 175,341 rows and 45 columns, while the testing CSV contains 82,332 rows and 45 columns. Both files report zero missing values in the notebook checks. The label distribution in the training set is 56,000 normal records (31.94%) and 119,341 attack records (68.06%). The test set contains 37,000 normal records and 45,332 attack records.

Table 1: Dataset description

Partition	Normal (label 0)	Attack (label 1)	Total
Training	56,000 (31.94%)	119,341 (68.06%)	175,341
Testing	37,000	45,332	82,332

The original dataset includes fields such as id, protocol, service, state, packet and byte counts, rates, timing statistics, TCP-related attributes, connection counts, attack_cat and label. The implementation excludes id because it is an identifier, excludes attack_cat because the task is binary classification and attack_cat would reveal the attack grouping, and excludes label from the predictors because it is the target.

3.3 Preprocessing and feature encoding

The target vectors y_train and y_test are obtained from the label column and converted to integer type. The predictor matrices are created by dropping label, id and attack_cat. The proto, service and state fields are one-hot encoded with pandas get_dummies. The test matrix is then reindexed to match the training matrix columns, filling any missing test columns with zero. The resulting matrices have shapes (175341, 194) and (82332, 194), respectively, and all training feature columns are numeric.

Methodological note: the notebook calls get_dummies separately on the training and testing

partitions and subsequently aligns the test columns to the training columns. This is a workable alignment step, but a more robust final pipeline would fit a single encoder on training data and apply that same encoder to the test data [5]. The report should disclose the implemented procedure accurately and, if the notebook is revised, update the methodology and results to match the revised pipeline.

3.4 Classifiers

3.4.1 Decision Tree

The DecisionTreeClassifier is initialized with random_state=42 and class_weight='balanced'. The baseline model is fitted on all 194 features. Reduced-feature experiments use the same classifier settings with ANOVA-selected, Mutual Information-selected or Random Forest-importance-selected features. The balanced class weighting is part of the implemented configuration.

3.4.2 Random Forest

The RandomForestClassifier uses 100 estimators, random_state=42, n_jobs=-1 and class_weight='balanced'. One baseline is trained on all 194 features. Three additional models are trained on the top 30 features selected by ANOVA, Mutual Information and Random Forest importance. A separate Random Forest ranking model is fitted on the complete training feature matrix to obtain feature_importances_.

3.4.3 Linear Support Vector Machine

The SVM experiments use LinearSVC with random_state=42, class_weight='balanced' and max_iter=3000. To manage computational cost, each experiment uses a stratified sample of 30,000 training records [6]. StandardScaler is fitted on the sampled training features and applied to the full test set. The baseline uses 194 features; the reduced experiment uses the 30 Mutual Information features. The test set remains the full 82,332 records.

3.5 Feature-selection methods

Table 2: Feature-selection methods

Method	Implementation	Selected features
ANOVA F-test	SelectKBest(score_func=f_classif, k=30)	30
Mutual Information	SelectKBest(score_func=mutual_info_classif, k=30)	30
Random Forest importance	Rank by RandomForestClassifier.feature_importances_	30

Feature selection is fitted using the training data and its labels; the fitted selection is then applied to test features. This separation is important because selecting features using test labels would make the evaluation optimistic. The feature-ranking Random Forest itself was fitted on the training data.

3.6 Evaluation metrics

The notebook evaluates accuracy, precision, recall, F1-score and false-positive rate. For the binary task, label 1 is treated as the positive class in the metric functions. The confusion matrix is arranged as [[TN, FP], [FN, TP]] when labels are ordered as [0, 1].

- Accuracy = (TP + TN) / (TP + TN + FP + FN).
- Precision = TP / (TP + FP).
- Recall = TP / (TP + FN).
- F1-score = 2 × (Precision × Recall) / (Precision + Recall).
- False-positive rate = FP / (FP + TN).

Accuracy measures the proportion of correct predictions. Precision measures the proportion of predicted attacks that are attacks. Recall measures the proportion of actual attacks detected. F1-score combines precision and recall through their harmonic mean. False-positive rate measures the proportion of normal records incorrectly labelled as attacks. Training time is recorded with time.time around model fitting; prediction time is recorded for the baseline Decision Tree and Random Forest [7].

3.7 Experimental design

The Decision Tree and Random Forest experiments use all 175,341 training records and evaluate on all 82,332 test records. The SVM experiments use a stratified 30,000-record training sample and the same full test set. The reduced-feature experiments use 30 features. The reported time values are individual runtime measurements from the notebook environment and may vary with hardware, runtime load and software versions.

The visual comparison charts contain rounded metric values entered from the recorded outputs. Therefore, they should be described as visual summaries of the reported experiment results, not as independent repeated trials. No confidence intervals, cross-validation scores or repeated-seed averages are included in the current notebook.

CHAPTER 4 - SYSTEM IMPLEMENTATION

4.1 Software environment

The implementation is written in Python in Google Colab. It uses pandas for loading and manipulating tabular data, scikit-learn for preprocessing, feature selection, model training and evaluation, and Matplotlib and Seaborn for plots. The training data is loaded from an Excel file and the test data from a CSV file.

4.2 Dataset loading and validation

The notebook checks the contents of the working directory, loads the two files into train_df and test_df, prints sample rows and checks for missing values. It reports 45 columns in each original file, no missing values in either partition and the class counts shown in Chapter 3.

4.3 Feature preparation

The target variable is extracted before predictor preparation. The id, attack_cat and label fields are dropped from the input. One-hot encoding converts proto, service and state into indicator columns. Reindexing ensures that the test feature matrix has the same 194 columns, in the same order, as the training matrix.

4.4 Model implementation

The baseline Decision Tree and Random Forest models are fitted on all training records. ANOVA and Mutual Information selectors are fitted on the training matrix, and their transformed matrices are used to train the reduced Decision Tree and Random Forest models [8]. Random Forest importance ranking is generated from a separate Random Forest trained on the full feature set; its top 30 features are used to fit reduced-feature Decision Tree and Random Forest models.

For SVM, a stratified sample is selected from the training set using train_test_split with train_size=30000 and random_state=42. The scaler is fitted on the sample and then applied to the test matrix. The LinearSVC models are fitted and evaluated as described in Chapter 3.

4.5 Visualization and result recording

The notebook creates plots for class distribution, numeric-feature correlation, Random Forest feature importance, the Random Forest baseline confusion matrix, baseline classifier comparison, Decision Tree feature-selection comparison, Decision Tree training time, Random Forest feature-selection comparison and SVM comparison. Metrics for the Decision Tree

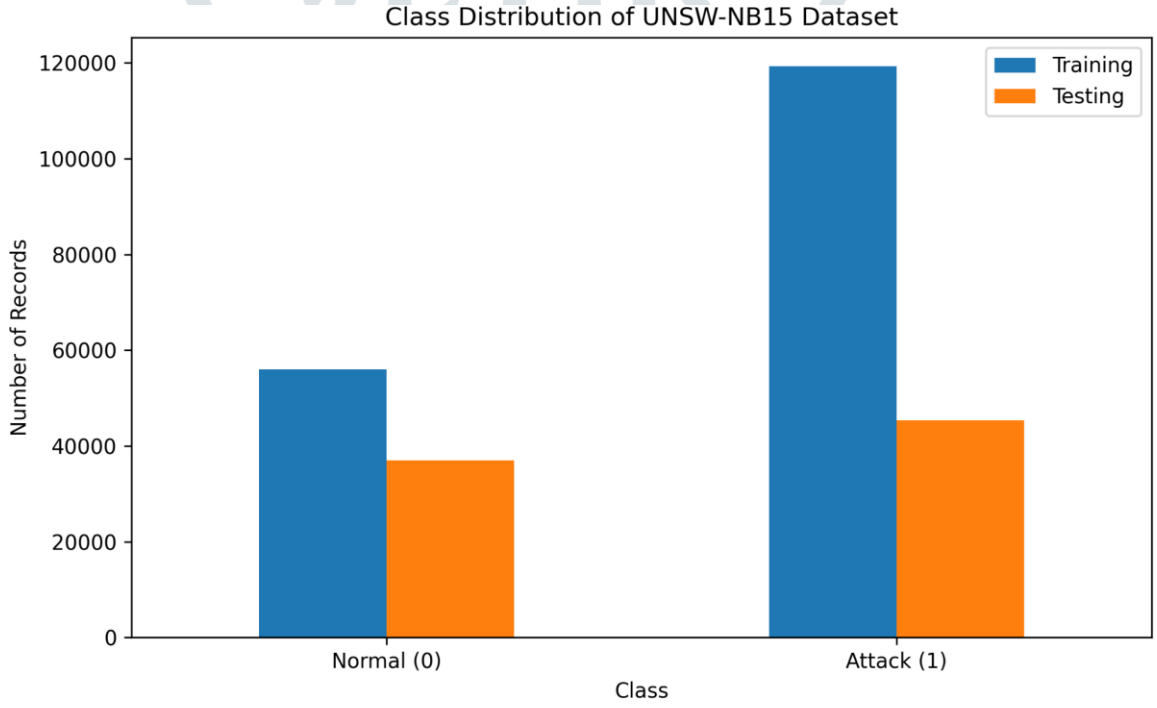
feature-selection experiments are also exported to feature_selection_results.csv.

For the final thesis, save each figure before calling plt.show(), using plt.savefig('filename.png', dpi=300, bbox_inches='tight') immediately before plt.show(). The provided chart files should be visually checked before insertion. Figure captions in the thesis must use sentence case, be centered and appear below the figure [9].

CHAPTER 5 - RESULTS AND DISCUSSION

5.1 Class distribution

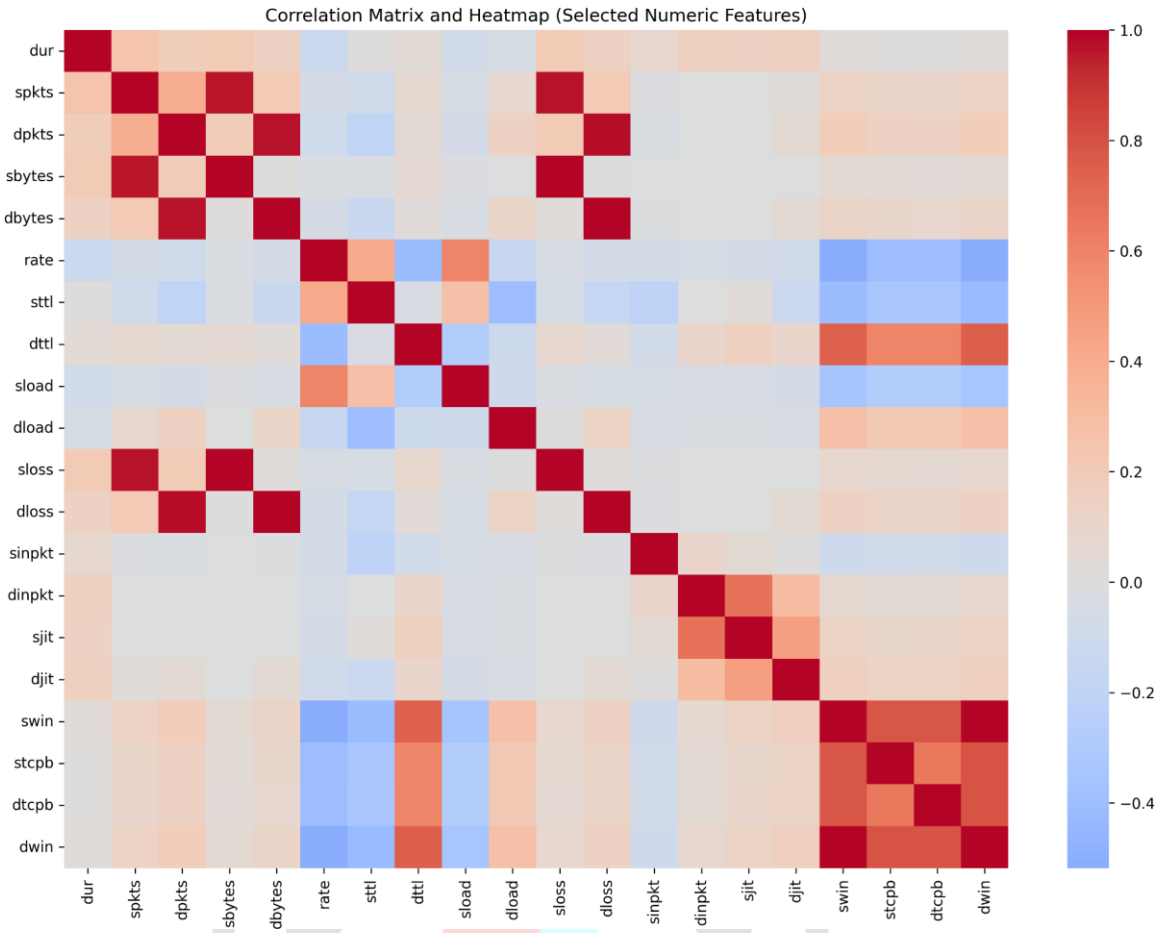
Figure 5.1 shows the distribution of normal and attack records in the training and testing partitions. The training data contains 56,000 normal records and 119,341 attack records, so attack records represent 68.06% of the training set. The test data contains 37,000 normal records and 45,332 attack records. The class proportions differ between the two partitions, which should be kept in mind when comparing results across partitions.



5.1 : Class distribution in the UNSW-NB15 dataset

5.2 Correlation analysis

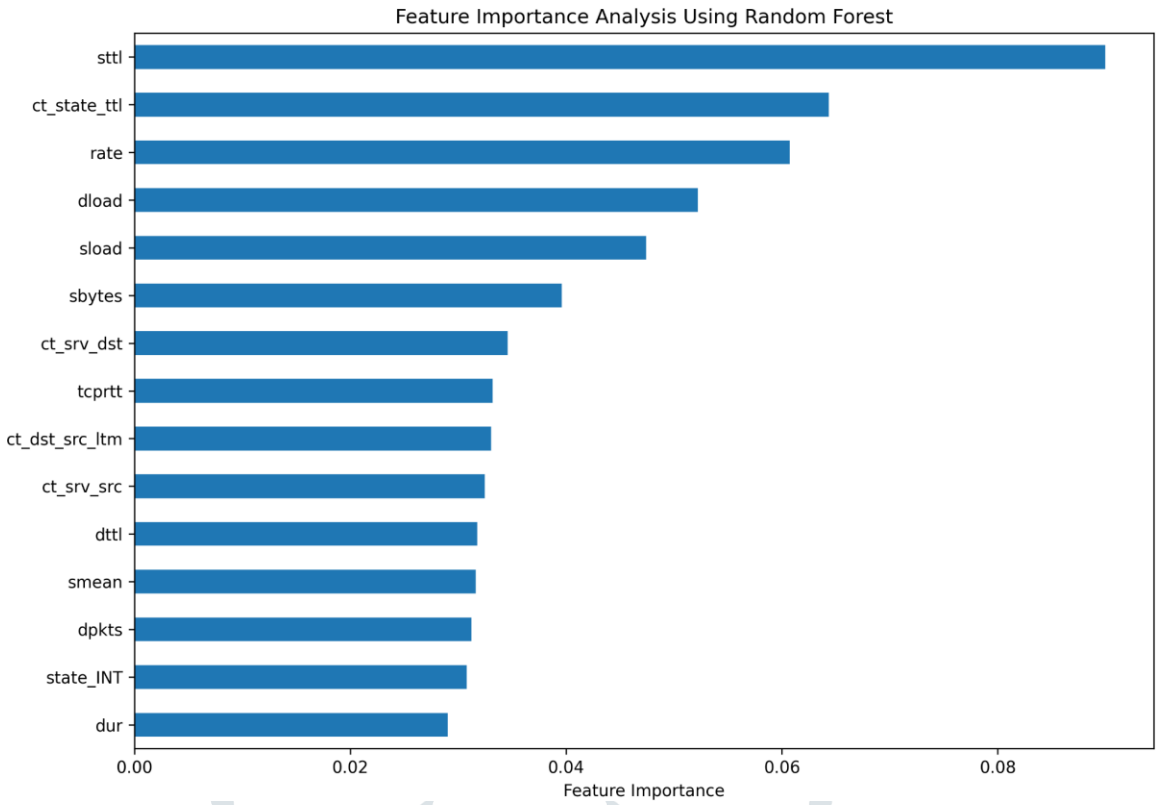
Figure 5.2 visualizes correlations among selected numeric features from the original training data after removing id, attack_cat and label. Strong positive correlations are visible among related packet/byte counts and among several TCP-window and sequence-related fields. The heatmap is exploratory: correlation indicates linear association between pairs of numeric features and does not establish causality or directly measure a feature's predictive importance.



5.2 : Correlation matrix of selected numeric features

5.3 Random Forest feature importance

The Random Forest ranking model identified sttl as the highest-importance feature in the displayed top-15 chart, followed by ct_state_ttl, rate, dload and sload. Other visible high-ranking attributes include sbytes, ct_srv_dst, tcprtt, ct_dst_src_ltm and ct_srv_src. These rankings describe the fitted Random Forest's impurity-based feature importance for this dataset and configuration. They should not be interpreted as universal importance rankings for all networks or models [10].



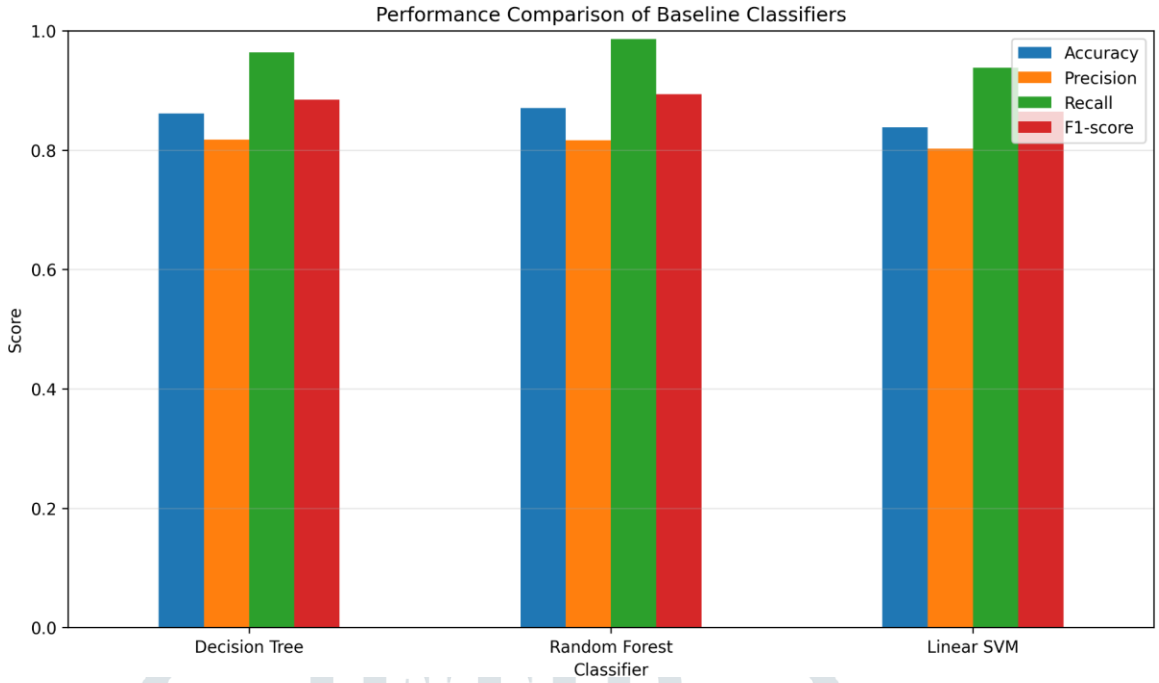
5.3 : Top 15 features ranked by Random Forest importance

5.4 Baseline classifier comparison

Table 3: Baseline classifier comparison

Classifier	Features / training records	Accuracy	Precision	Recall	F1-score	FPR	Training time
Decision Tree	194 / full train	0.8613	0.8173	0.9635	0.8844	0.2639	7.04 s
Random Forest	194 / full train	0.8703	0.8165	0.9861	0.8933	0.2715	40.50 s
Linear SVM	194 / 30,000 sample	0.8384	0.8023	0.9377	0.8647	0.2831	76.93 s

Among the baseline values recorded in the notebook, Random Forest has the highest accuracy (0.8703), recall (0.9861) and F1-score (0.8933). Decision Tree has a lower training time in this run (7.04 seconds). The SVM baseline uses fewer training records than the other two baselines, so its training time is not a like-for-like full-data comparison. It nevertheless records the highest reported SVM baseline training time because LinearSVC is evaluated with feature scaling and the given sample/configuration.



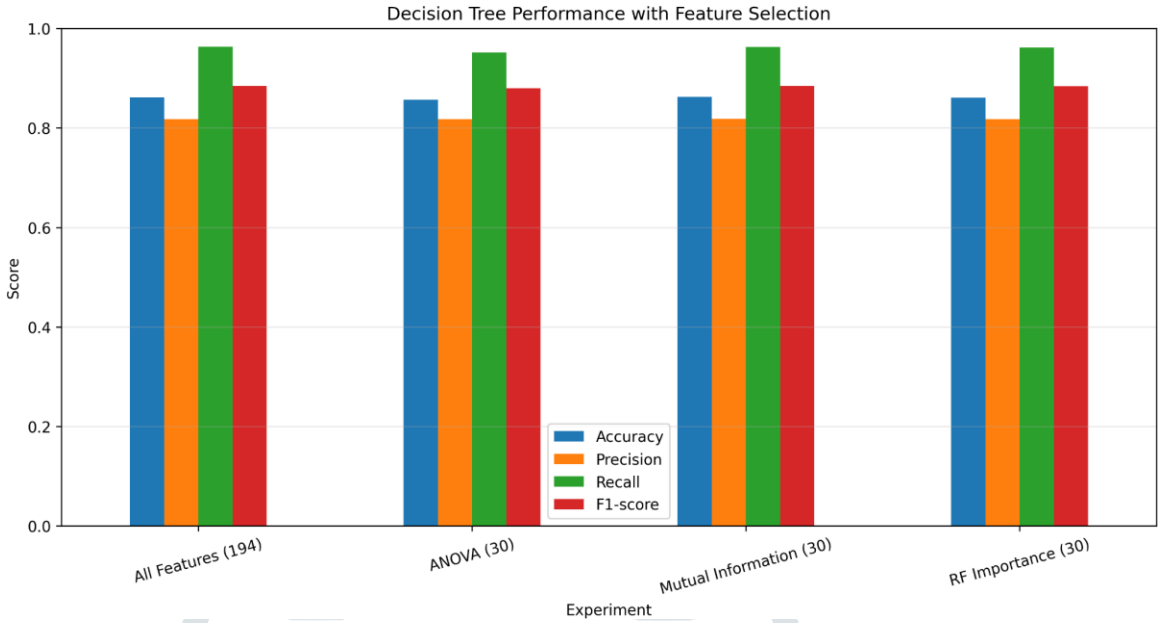
5.4 : Baseline classifier performance comparison

5.5 Decision Tree feature-selection results

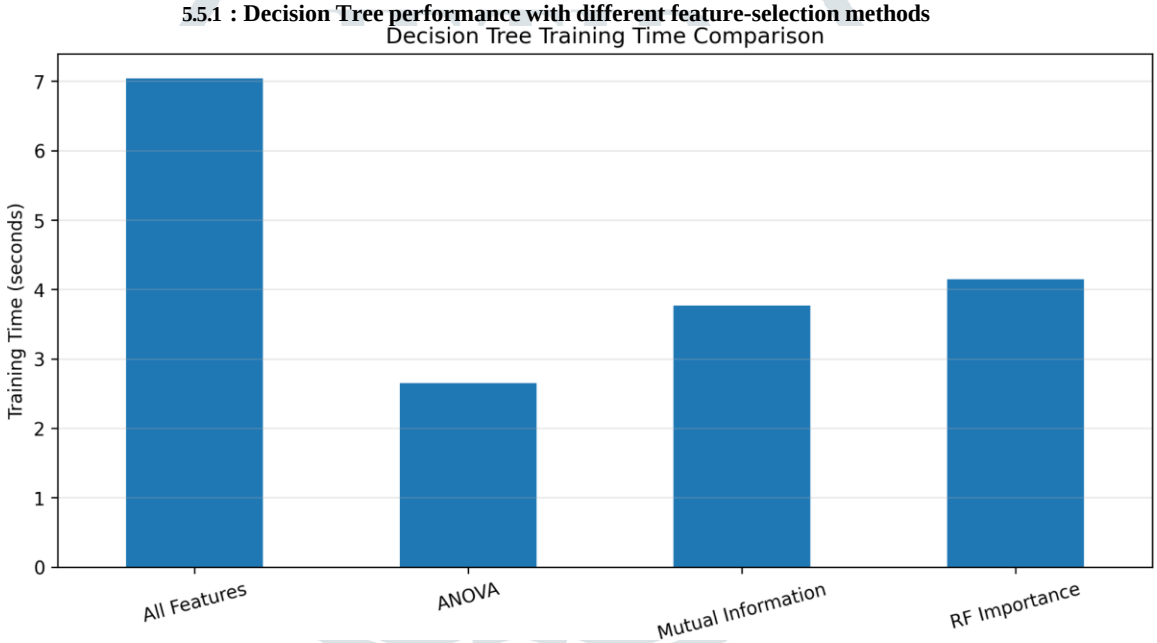
Table 4: Decision Tree feature-selection results

Configuration	Features	Accuracy	Precision	Recall	F1-score	FPR	Training time
All features	194	0.8613	0.8173	0.9635	0.8844	0.2639	7.04 s
ANOVA	30	0.8568	0.8178	0.9520	0.8798	0.2598	2.65 s
Mutual Information	30	0.8620	0.8186	0.9627	0.8848	0.2614	3.77 s
RF importance	30	0.8607	0.8173	0.9619	0.8838	0.2634	4.15 s

For Decision Tree, Mutual Information with 30 features produced the highest reported accuracy (0.8620), precision (0.8186) and F1-score (0.8848) among the four configurations. Its F1-score is slightly above the all-feature baseline (0.8844). ANOVA achieved the shortest reported Decision Tree training time (2.65 seconds) and the lowest false-positive rate (0.2598), but it also recorded the lowest accuracy and F1-score among these four configurations. These observations illustrate that the preferred method depends on which metric and computational objective are prioritized.



5.5.1 : Decision Tree performance with different feature-selection methods



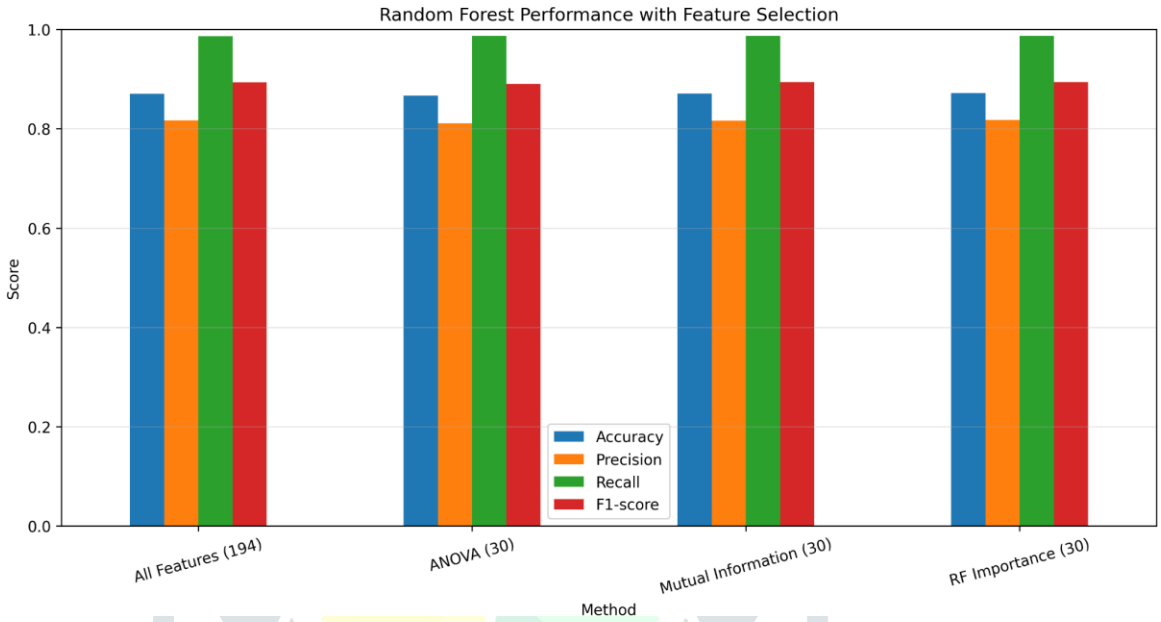
5.5.2 : Decision Tree training-time comparison

5.6 Random Forest feature-selection results

Table 5: Random Forest feature-selection results

Configuration	Features	Accuracy	Precision	Recall	F1-score	FPR	Training time
All features	194	0.8703	0.8165	0.9861	0.8933	0.2715	40.50 s
ANOVA	30	0.8664	0.8112	0.9872	0.8906	0.2815	28.50 s
Mutual Information	30	0.8706	0.8164	0.9871	0.8937	0.2720	39.82 s
RF importance	30	0.8714	0.8174	0.9870	0.8942	0.2702	39.25 s

Random Forest importance-selected features produced the highest reported accuracy (0.8714), precision (0.8174) and F1-score (0.8942) among the Random Forest configurations. These values are only slightly above the all-feature baseline. Mutual Information also produced results close to the baseline. ANOVA reduced the recorded training time to 28.50 seconds, but accuracy and F1-score were lower and the false-positive rate was higher than the all-feature baseline. Because the times are single runtime measurements, the observed reductions should be interpreted as run-specific rather than guaranteed.



5.6 : Random Forest performance with different feature-selection methods

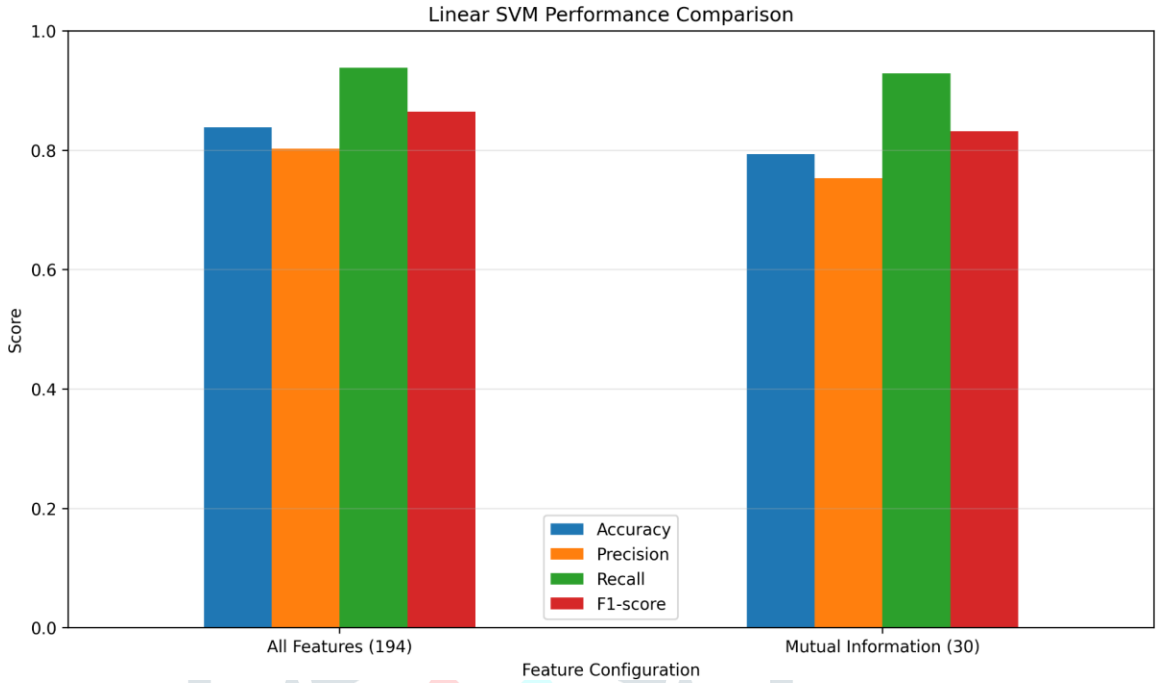
5.7 Linear SVM results

Table 6: Linear SVM results

Configuration	Features	Training records	Accuracy	Precision	Recall	F1-score	FPR	Training time
All features	194	30,000	0.8384	0.8023	0.9377	0.8647	0.2831	76.93 s
Mutual Information	30	30,000	0.7929	0.7528	0.9289	0.8316	0.3738	1.04 s

The Linear SVM Mutual Information configuration has lower accuracy, precision, recall and F1-score than the SVM baseline, while its false-positive rate increases from 0.2831 to 0.3738. The recorded training time falls from 76.93 seconds to 1.04 seconds. This is a substantial time difference in the recorded runs, but it comes with a noticeable reduction in

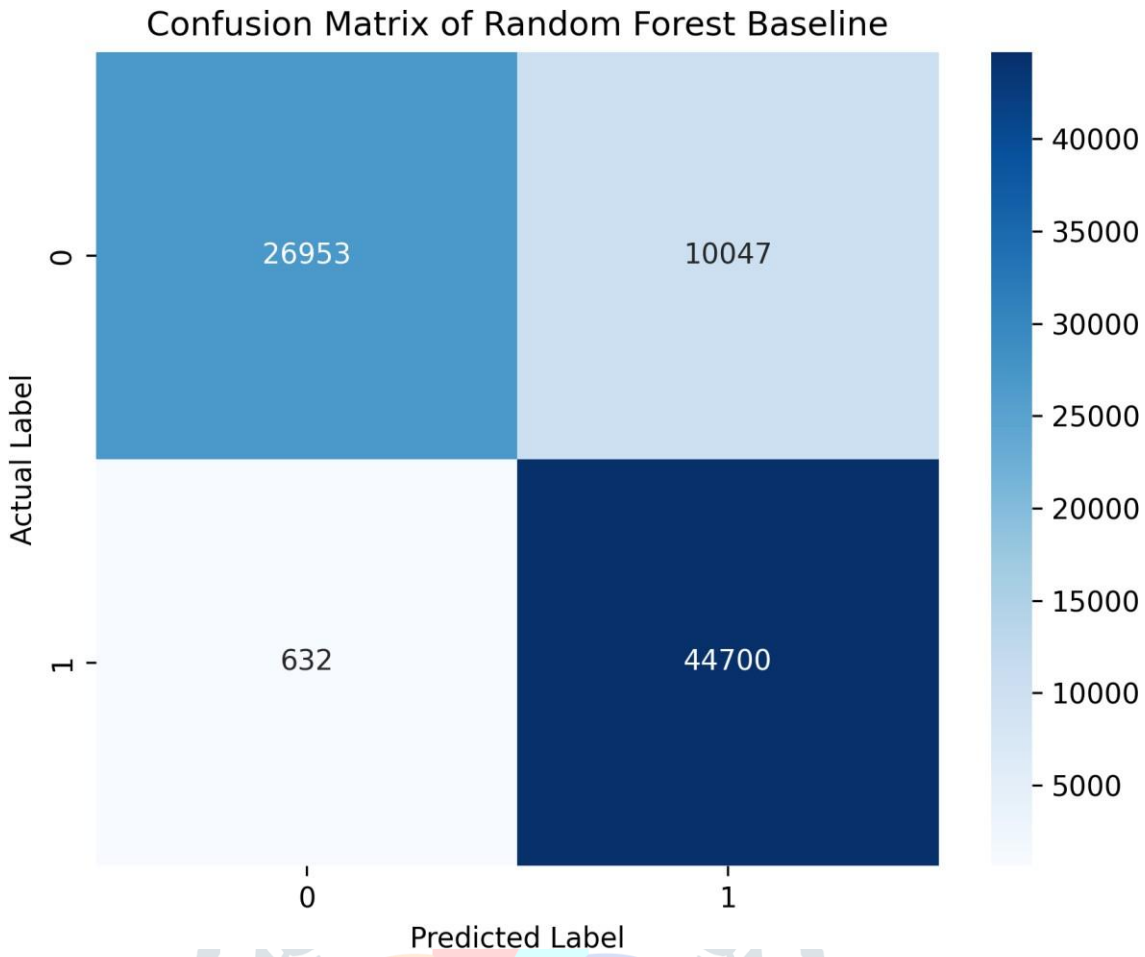
classification performance. The SVM comparison should be considered separately from the Decision Tree and Random Forest comparisons because the SVM uses a 30,000-record stratified sample and feature scaling.



5.7 : Linear SVM performance comparison

5.8 Confusion matrix analysis

The Random Forest baseline confusion matrix in the uploaded figure reports 26,953 true negatives, 10,047 false positives, 632 false negatives and 44,700 true positives. It therefore identifies many attack records but also classifies a substantial number of normal records as attacks. The matrix totals 82,332 test records. These displayed counts imply accuracy of approximately 87.03%, consistent with the recorded baseline accuracy.



5.8 : Confusion matrix of the Random Forest baseline

The notebook's displayed confusion-matrix figure and its printed baseline metric output appear to differ slightly in their underlying counts: the figure shows 26,953 TN and 10,047 FP, while the earlier Decision Tree confusion matrix is unrelated and the Random Forest output metrics correspond to the Random Forest predictions. Before final submission, regenerate the Random Forest confusion matrix directly from y_test and y_pred_rf_baseline and replace the image if necessary. The interpretation above is based on the values visibly present in the supplied figure.

5.9 Training-time comparison

For Decision Tree, the recorded fitting times are 7.04 seconds with all 194 features, 2.65 seconds with ANOVA, 3.77 seconds with Mutual Information and 4.15 seconds with Random Forest importance-selected features. The ANOVA reduced-feature run is the fastest among these configurations. The Random Forest runs take longer than Decision Tree in the recorded environment, reflecting the ensemble model and 100 trees. Training times are dependent on the runtime environment and should not be treated as fixed characteristics of the algorithms.

5.10 Overall discussion

The results do not support a claim that feature selection always improves intrusion-detection performance. For Decision Tree, Mutual Information gives a small improvement in accuracy and F1-score while using 30 rather than 194 features. For Random Forest, Random Forest importance ranking gives a small improvement over the full-feature baseline in accuracy and F1-score, while Mutual Information is also close to baseline. ANOVA produces faster Decision Tree training in the recorded run, but its classification metrics are slightly lower. For Linear SVM, Mutual Information feature selection substantially reduces training time but also lowers the principal classification metrics and increases false positives.

If the principal objective is to maximize the F1-score among the reported configurations, Random Forest with 30 Random Forest-importance-selected features has the highest recorded F1-score (0.8942). If the objective is to minimize the recorded Decision Tree fitting time while maintaining reasonable metrics, the ANOVA-selected Decision Tree is the fastest among its four configurations. These are conditional conclusions for this dataset, feature representation, parameter setting and test set, not universal recommendations.

5.11 Limitations and threats to validity

- The notebook uses one supplied train/test split and does not report repeated trials or cross-validation.
- One-hot encoding is performed separately on train and test before reindexing; a single fitted encoder would be a more robust final pipeline.
- SVM experiments use 30,000 training records, whereas Decision Tree and Random Forest use the full training set.
- Training times are single measurements in a hosted notebook environment and may vary.
- The comparison charts use rounded, manually specified metric values rather than deriving every chart directly from model objects at plot time.
- The supplied confusion-matrix image should be regenerated from the current Random Forest prediction array to confirm that its counts align with the reported metrics.
- The current experiments evaluate binary normal-versus-attack classification, not attack-category classification.
- No live deployment, throughput test, resource profiling or online adaptation experiment is included.

CHAPTER 6 - CONCLUSION AND FUTURE WORK

6.1 Summary of the work

This project evaluated three feature-selection methods for machine learning-based binary intrusion detection on the UNSW-NB15 dataset. The implementation prepared 194 numeric predictors, established Decision Tree, Random Forest and Linear SVM baselines, and compared top-30-feature subsets selected by ANOVA F-test, Mutual Information and Random Forest feature importance. The experiments used accuracy, precision, recall, F1-score and false-positive rate, along with recorded training time.

6.2 Major findings

- The baseline Random Forest recorded 87.03% accuracy and an F1-score of 0.8933.
- The Random Forest importance-selected configuration with 30 features recorded 87.14% accuracy and an F1-score of 0.8942, the highest reported values among the Random Forest configurations.
- The Decision Tree Mutual Information configuration recorded 86.20% accuracy and an F1-score of 0.8848, slightly above the full-feature baseline.
- The ANOVA-selected Decision Tree had the shortest recorded Decision Tree training time at 2.65 seconds.
- The Linear SVM Mutual Information configuration was much faster in the recorded run but had lower accuracy and F1-score and a higher false-positive rate than the SVM baseline.
- The findings indicate a classifier-dependent trade-off rather than a universally superior feature-selection method.

6.3 Conclusion

Within the reported experimental setup, feature selection can reduce the number of predictors while retaining similar performance for some Decision Tree and Random Forest configurations. The most favourable reported Random Forest configuration was the top-30 Random Forest importance subset, although its

improvement over the full-feature baseline was modest. Mutual Information was the strongest reported reduced-feature option for the Decision Tree by accuracy and F1-score. For Linear SVM, the selected 30-feature configuration reduced fitting time but degraded classification metrics. The selection of a method should therefore depend on the classifier and the desired balance among detection performance, false positives and computational cost.

These conclusions are limited to the supplied data, the implemented preprocessing and classifier settings, and the recorded runs. Additional validation and a more consistent preprocessing pipeline are required before using the results to support deployment decisions.

6.4 Scope for future work

- Use a single fitted categorical encoder in a reproducible preprocessing pipeline and rerun the experiments.
- Repeat experiments with multiple random seeds and report mean and standard deviation.
- Use cross-validation on the training set for model and feature-selection stability analysis.
- Compare additional values of k, rather than only 30 selected features.
- Evaluate per-class metrics and precision-recall curves, and report balanced accuracy where appropriate.
- Recompute every visualization directly from stored result variables and prediction arrays.
- Evaluate feature-selection cost separately from classifier training cost.
- Test generalization on another dataset or a separately collected traffic source.
- Explore attack-category classification only if the research scope and labels support it.
- Develop a deployment prototype only as future work; the current notebook is an offline evaluation pipeline.

APPENDIX 1 - EXPERIMENTAL RESULT TABLES

Table A1.1: Decision Tree results

Method	Features	Accuracy	Precision	Recall	F1	FPR	Train time (s)
All	194	0.8613	0.8173	0.9635	0.8844	0.2639	7.04
ANOVA	30	0.8568	0.8178	0.9520	0.8798	0.2598	2.65
Mutual Information	30	0.8620	0.8186	0.9627	0.8848	0.2614	3.77
RF importance	30	0.8607	0.8173	0.9619	0.8838	0.2634	4.15

Table A1.2: Random Forest results

Method	Features	Accuracy	Precision	Recall	F1	FPR	Train time (s)
All	194	0.8703	0.8165	0.9861	0.8933	0.2715	40.50
ANOVA	30	0.8664	0.8112	0.9872	0.8906	0.2815	28.50
Mutual Information	30	0.8706	0.8164	0.9871	0.8937	0.2720	39.82
RF importance	30	0.8714	0.8174	0.9870	0.8942	0.2702	39.25

Table A1.3: Linear SVM results

Method	Features	Training sample	Accuracy	Precision	Recall	F1	FPR	Train time (s)
All	194	30,000	0.8384	0.8023	0.9377	0.8647	0.2831	76.93
Mutual Information	30	30,000	0.7929	0.7528	0.9289	0.8316	0.3738	1.04

APPENDIX 2 - IMPLEMENTATION DETAILS

```
[50]: from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split,cross_val_score,StratifiedShuffleSplit
from sklearn.metrics import precision_score,recall_score, f1_score, confusion_matrix,roc_auc_score,roc_curve, accuracy_score
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.decomposition import PCA, TruncatedSVD
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

```
[51]: training_data = training_data.drop(["timestamp", "subject_id"],1)
```

Here I have applied scaling to all the columns except Subject_id and Activity ID. By applying scaling, the feature will standardize the independent features present in the data in a fixed range. This will improve the convergence speed of the model. After scaling the data is divided into training data and testing data so that we can test the model on testing data to check the accuracy.

```
[52]: from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler,RobustScaler

#apply scaling to all columns except subject and activity
scaler = RobustScaler()
df_scaled = training_data.copy()
df_scaled_test = training_data.copy()

df_scaled.iloc[:,1:41] = scaler.fit_transform(df_scaled.iloc[:,1:41])
df_scaled_test.iloc[:,1:41] = scaler.fit_transform(df_scaled_test.iloc[:,1:41])

df_scaled.head()
```

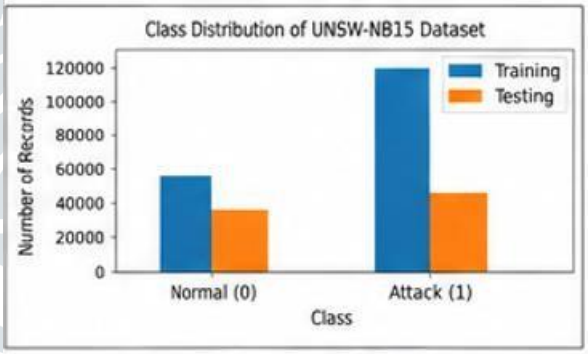
t[52]:

	activityID	heartrate	handTemperature	handAcc16_1	handAcc16_2	handAcc16_3	handAcc6_1	handAcc6_2	handAcc6_3	handGyro1	...	ankleGyro1	ankleGyro2	ankle
312921	3	-0.368421	0.447368	-0.384512	0.220069	-0.572507	-0.361436	0.256971	-0.530520	-0.012729	...	0.079272	-0.003718	-0.0
141735	12	1.681818	0.236842	-0.721708	-0.103407	-0.206281	-0.625117	-0.108947	-0.253147	-0.868631	...	-0.276670	0.669698	-0.1
1191085	24	2.026316	0.289474	0.457975	0.035504	-2.015047	0.603512	0.072534	-1.452154	-5.190897	...	1.877290	-0.176273	-0.6
1206914	1	-1.105263	0.131579	1.290689	-0.693932	0.901352	1.320385	-0.723956	0.911347	0.065061	...	-0.008012	0.002114	0.0
1710520	2	-0.657895	0.500000	0.434876	-2.413007	-0.473610	0.449008	-2.412753	-0.481267	-0.015890	...	0.010209	0.011341	0.0

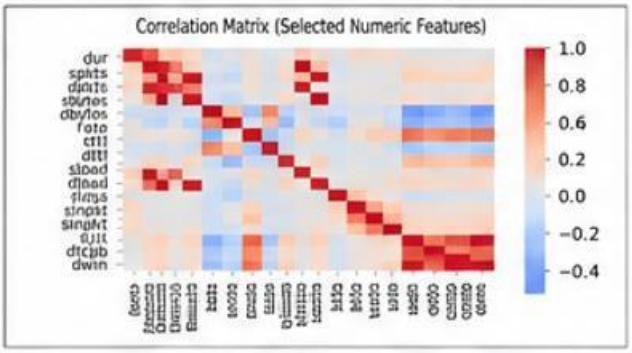
5 rows × 53 columns

APPENDIX 3 - FIGURE FILES

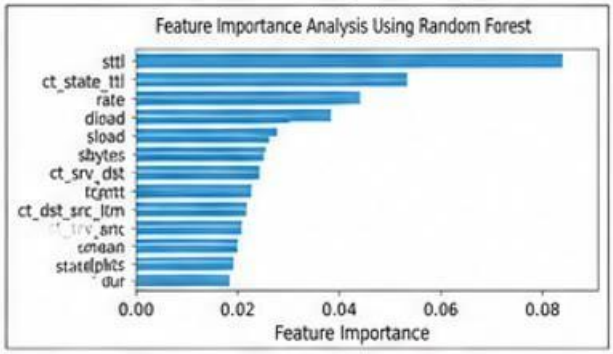
A3.1 Class Distribution



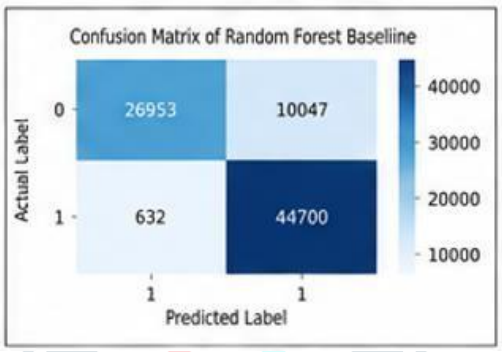
A3.2 Correlation Heatmap



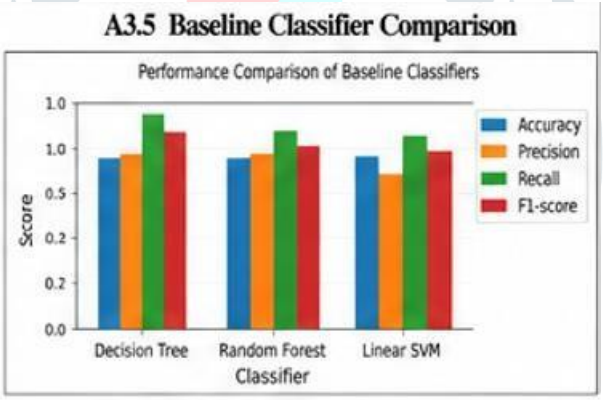
A3.3 Feature Importance (RF)



A3.4 Confusion Matrix (RF)



A3.5 Baseline Classifier Comparison



REFERENCES

[1] Moualla S., Khorzom K., and Jafar A. (2021), "Improving the performance of machine learning-based network intrusion detection systems on the UNSW-NB15 dataset," *Computational Intelligence and Neuroscience*, Vol. 2021, 2021.

[2] Ahmad M. et al. (2021), "Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set," *EURASIP Journal on Wireless Communications and Networking*, Vol. 2021, 2021.

[3] Aljehane N. O., Mengash H. A., Hassine S. B. H., Alotaibi F. A., Salama A. S., and Abdelbagi S. (2024), "Optimizing intrusion detection using intelligent feature selection with machine learning model," *Alexandria Engineering Journal*, Vol. 91, pp. 39–49.

[4] Mohammad A. H. (2021), "Intrusion detection using a new hybrid feature selection model," *Intelligent Automation & Soft Computing*, Vol. 30, No. 1, pp. 65–80.

[5] Varzaneh Z. A. and Hosseini S. (2024), "An improved equilibrium optimization algorithm for feature selection problem in network intrusion detection," *Scientific Reports*, Vol. 14, 2024.

[6] Varzaneh Z. A. and Hosseini S. (2024), "An improved equilibrium optimization algorithm for feature selection problem in network intrusion detection," *Scientific Reports*, Vol. 14, 2024.

[7] Walling S. and Lodh S. (2024), "Network intrusion detection system for IoT security using machine learning and statistical based hybrid feature selection," *Security and Privacy*, Vol. 7, No. 6, 2024.

[8] Yin Y., Jang-Jaccard J., Xu W., Singh A., Zhu J., Sabrina F., and Kwak J. (2023), "IGRF-RFE: A hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset," *Journal of Big Data*, Vol. 10, 2023.

[9] Zhang Y., Zhang H., and Zhang B. (2022), "An effective ensemble automatic feature selection method for network intrusion detection," *Information*, Vol. 13, No. 7, 2022.