Statistical analysis on the UNSW-NB15 dataset and performance evaluation of AI models for network intrusion detection

Gabriele Volani
Dept. of Information Engineering and Computer Science
University of Trento
Trento, Italy
gabriele.volani@studenti.unitn.it

Sanasar Hamburdzamyan
Dept. of Information Engineering and Computer Science
University of Trento
Trento, Italy
s.hamburdzamyan@studenti.unitn.it

***Abstract*—Collecting high-quality and representative network traffic datasets is crucial to properly train Machine Learning based Intrusion Detection Systems. In this work, we provide a statistical analysis of the UNSW-NB15 dataset, a modern benchmark with more than 2 000 000 records about network traffic data. Through an in-depth feature analysis, we identified the most relevant attributes for attack detection and discrimination, providing insights on them with summary statistics. Then we evaluated the performances of five AI models for intrusion detections proving that, the most significant and informative features, are actually helpful in the classification task.**

***Index Terms*—performance evaluation, network intrusion, anomaly detection, network traffic dataset**

I. Introduction

In this era of rapid digital transformations, security of network infrastructures has become a critical concern, due to the constant evolution in frequency and complexity of modern cyberattacks. Without proper protection, networks can serve as an open gateway in computer systems for malicious actors.

To avoid network breaches, it is important to rely on powerful Intrusion Detection Systems (IDS) that can rapidly analyze the traffic flow and detect as fast as possible the network intrusion and mitigate the tremendous impact of an attack.

Traditional IDS are designed to detect possible threats by looking for specific patterns and inspecting content of network packets with a rule-based approach, following predefined and heuristic-based algorithms.

Unfortunately, this could not be enough when slightly different variations of similar attacks are performed on the network or when we need to analyze vast amounts of data to identify those patterns. For this reason, Machine Learning and Deep Learning models serve as effective tools for detecting network anomalies, thereby facilitating the identification of potential attacks.

However, it is important to note that Machine Learning models require proper training. Obtaining a high-quality network traffic dataset for cybersecurity application can be challenging due to various reasons, including privacy concerns, overwhelming data volume, difficult representativeness of the real-world network traffic scenarios and raw level of the acquisition. A common solution is to rely on public datasets generated in a controlled environment, as the **UNSW-NB15** dataset, that includes network traffic data comprising both benign activity and traffic associated with nine of the most common network attacks. It contains more than 2 millions record and 47 features.

In this context, our work aims to provide a statistical analysis about the most significative features that can help in anomaly detection and a performance evaluation and comparison of some ML and DL models for classify normal or malign situations.

Unlike many studies that directly apply models without a dataset analysis about correlation, feature importance and statistical aspects, we are providing novel results to identify the variables whose values are more representative of the network situation.

Our work then evaluates the performances of the AI models both with all the features and also with just the ones which are considered more statistically significant for the classification task. This can help reduce the size of the datasets and provide a more uniform way to characterize the raw data collected from a low-level network traffic analysis, focusing only on the attributes that are most important for the classification.

What distinguishes this work is its systematic approach to feature analysis prior to model training and the evaluation of multiple ML algorithms to validate the hypothesis made in the first analytical phase. This dual emphasis on explainable feature selection and robust model evaluation aims to provide a deeper understanding of the trade-offs between model complexity and detection accuracy, which is vital for deploying ML-based IDS in safety critical environments.

II. Related Work

In the literature there are many similar works that are evaluating IDS on this dataset or are just providing some statistical analysis.

The article [1] is providing a statistical analysis of the UNSW-NB15 dataset and a comparison with the KDD99 dataset (another dataset that contains network flow information used for anomaly and intrusion detection).

The analysis they are performing involves the evaluation of the distribution of the training set and the test set provided by the creators of the UNSW-NB15. In fact, for AI systems evaluation is important that both the training and the testing are done on data generated by the same probability distribution. They have used the Kolmogorov-Smirnov Test to compare the two distributions and then the Pearson's Correlation Coefficient and the Gain Ratio to measure the feature correlations between the two sets.

The result they achieved was that features of the training and testing sets are a highly statistical correlated and the UNSW-NB15 dataset is reliable to evaluate ML-based IDS.

To conclude, they evaluate also few ML models on the two datasets specified above, gaining better performances (in the accuracy and false alarm rates metrics) in the UNSW-NB15 dataset.

Even if the statistical work to analyze feature was not similar to ours (we just want to provide insights about the most important features for the classification), it was important to provide hints in pre-processing of data and results with which we can compare the metrics of our ML models.

Another important work is [2], which provides a visual analysis of the dataset.

It is showing additional techniques to convert nominal features into numerical format and for scaling the initial values.

Moreover, it showed us the two major problems of this dataset: the class imbalance and the class overlap. In fact, by using visualization techniques as PCA, t-SNE and k-means clustering they show how the various data points are somehow overlapped in low dimensions, meaning that many attack class records mimic the behavior of the Normal records.

Furthermore, it is shown that some classes are composed by smaller clusters themselves, meaning that for any attack can exist very different variations.

Additionally, they showed that the original dataset is very imbalanced: 87% of records identify normal activities in the network and just the rest 17% is associated with one of the 9 intrusion attacks. To deal with it, we have utilized the provided training set that has 175 341 records and is much more balanced than the original one.

Finally, it is important to cite the official paper [3] provided by the creators of the dataset, that is giving good insights and information about the structure of the dataset itself.

III. The UNSW-NB15 dataset

Before going into the actual description of our work and our results, it is appropriate to provide a brief description of the dataset on which we based our work.

The UNSW-NB15 dataset has been created by the Australian Centre for Cyber Security (ACCS) in 2015, in response to the unavailability of comprehensive network based data set able to reflect modern network traffic scenarios (existing benchmarks at the time realized just a limited number of attacks and information of packets which were outdated, so were ineffective for evaluating IDS). The key characteristic of this dataset is its hybrid nature combining the real modern normal behaviors and the synthetical attack activities (the nine families of attacks taken into account are: Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms). The team of the ACCS used the *IXIA PerfectStorm* tool to generate this binomial traffic. They generated also abnormal network traffic, thanks to the fact that tool contains all the information about the new attacks that are updated continuously on the CVE site. Overall, 2 000 000 traffic records are stored.

To capture the network traffic in form of packets and generate the pcap files they used the *tcpdump* tool. Furthermore, the 47 features (49, if we consider also the attack category and the label - 0 for normal traffic, 1 for anomalous traffic) of the dataset have been extracted using *Argus*, *Bro-IDS* tools and twelve C# algorithms to analyze in-depth the flows of the connection packets.

The extracted features can be divided into five categories:

- FLOW features (from 1 to 5): simply the IP addresses and the ports of source and destination and the network protocol used.
- BASIC features (from 6 to 18): general information about the traffic recorded (such as bytes sent and received, time to live, service requested, packets sent or dropped, etc.).
- CONTENT features (from 19 to 26): information about headers and payload of the packets.
- TIME features (from 27 to 35): information about the inter-arrival time and the source and destination jitter.
- ADDITIONAL GENERATED features (from 36 to 47): the most important features, whose intent is to identify some patterns in multiple network flows; they are generated starting from 100 connections sequentially ordered in order to extract common characteristic and reveal anomalous behaviors.

More details about dataset generation, simulation setup and the features meaning are available in [3].

IV. Problem definition

Our work is mainly divided into two interconnected phases: an initial *dataset analysis*, where we compute some summary statistics and where we inspect correlation among features, and a following *performance evaluation* phase, where we train some Machine Learning and Deep Learning models on our dataset and we compare different metrics.

A. Dataset analysis

In this primal phase we aimed to analyze statistically the data and variables inside the dataset.

To begin with, we analyze the correlations among the features. Using the correlation formula:

$$\rho = Corr(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}} \tag{1}$$

we have computed the correlation metric for every pair of feature. This helped us to identify the redundant features in the dataset.

Eliminating them during the training phase of AI models is crucial to avoid the **multicollinearity** problem, which can distort the interpretation of model coefficients and reduce the capabilities of the model to generalize properly on unseen data. In fact, highly correlated features provide overlapping information, leading to redundancy and overfitting.

By removing these features, we are able to ensure that the model learns only from distinct and meaningful signals: this allows us to get better performances, more stable predictions and also an improvement in the interpretability of the results.

The second part of dataset analysis is primarily focused on identifying the features that are statistically more relevant for the classification task.

We start by addressing the following question: "Which attributes and characteristics of the network flows are most strongly associated with anomalies and potential intrusions?".

Our objective is to gain a deeper understanding of the features that a prediction model should prioritize when classifying a network flow as anomalous. In particular, we aim to highlight those aspects of network activity that act as early "red flags" of non normal behaviors.

This targeted analysis can help to develop more effective classification models, which concentrate on extracting meaningful patterns only from the most informative signals. In our view, this feature selection strategy not only enhances model accuracy, but also improves generalization and robustness by focusing on the truly discriminative aspects of network traffic.

To support our intuition, we are going then to provide summary statistics of the features identified as "relevant for the classification" and show how they deeply change between a normal and a anomalous situation. We rely on mean, standard deviation, quantiles and coefficient of variation, providing also confidence interval within a 95% confidence level, to show the huge differences between different executions.

A similar approach was adopted in the *multiclass* setting. However, in this case, we could not directly compute simple correlation coefficients between features and attack categories, since the attack type is a nominal attribute.

To address this "problem", we turned to an exploratory data analysis using summary statistics and visualization techniques. Precisely, we computed descriptive statistics (similar to the previous case) for each feature across the different attack categories.

B. Performance evaluation of Machine Learning models

In this part, we are evaluating the performances of the Machine Learning and Deep Learning models used on the UNSW-NB15 dataset by related works. The chosen models are:

- Gaussian Naive Bayes
- Decision Tree
- Logistic Regression
- Random Forest
- Neural Network (standard multi layer perceptron)

The metrics used to analyze their performances are:

- Accuracy: *how many examples have been predicted correctly*.
- Precision: *of all the positive predictions we have done, how many are really positive*.
- Recall: *of all positive real cases, how many are predicted positive*.
- F1 score: a sort of *harmoinc mean* of precision and recall.
- ROC-AUC: a typical metric for binary classification.

The models were trained both on the complete training set and also on a "filtered" version of it, created based on the correlation analysis conducted during the initial phase of the study. Features that were highly correlated with each other were remove to avoid the multicellularity problem, whereas only the ones most relevant for the attack detection were retained.

This approach helped in simplifying the model and also in improving generalization and interpretation capabilities. At the end, by comparing model performances across both datasets, we validated the effectiveness of the feature selection strategy and the impact of eliminating redundant and less informative features.

All our work can be founded on our GitHub repository: https://github.com/gabri17/Project-SPE.git.

V. Dataset analysis results

In this section, we are going to see which features are highly correlated and represent redundant information, which

features are more meaningful for attack prediction and which features are more meaningful for discrimination across attack categories.

All these results and computation can be founded in `/analysis`, where three different directories are providing the codes and the results we got.

A. Redundant features

In appendix A there is the heatmap showing all the correlations among the variables inside the dataset are shown, computed using the Pearson correlation coefficient ρ.

We can see that that all the additional generated features (*ct_**) have a big positive correlations. This is expected, since they are featured derived through similar operations aggregating events happening between source and destination.

Then *sbytes* and *dbytes* (representing bytes sent and received) are correlated with the number of packets sent and received (*spkts* and *dpkts*) and with the source and receiver traffic loads (*sload* and *dload*). This is expected too and shows consistency between traffic and load on the network.

An interesting results is that the number of packets retransmitted or dropped (*sloss*) is more correlated with the number of bytes sent (*sbytes*) rather than to the number of packets sent (*spkts*, even if the correlation with it is clearly anyway high).

Then *spkts* and *dpkts* are a bit correlated with the duration of the connection *dur* suggesting an expected behavior of the network: the longer is a session and the more are the packets exchanged.

Lots of features are redundant and can be removed during the classification task: for instance all data about bytes and packets sent and load on the network are expressing similar information and we can remove some of those features to avoid multicollinearity.

B. Meaningful features for attack prediction

Mapping to *label = 0* a normal activity and to *label = 1* an anomaly, we can identify which features are most correlated with the fact of having an anomaly.

We summarize in the two tables in appendix B the top 10 features mostly correlated with the label, showing also as summary statistics change between each category.

The *sttl*, *ct_dst_src_ltm*, *Sload*, and *ct_srv_dst* features show significant variations between scenarios with and without an attack.

The CoV (coefficient of variation) is generally lower during attacks, indicating that the values in those situations are more concentrated, so attacks tend to have more regular patterns than normal traffic.

It is interesting to observe that standard deviations typically increase with an attack, while CoV does not. This is happening because the absolute values of the metrics increase a lot (so their "absolute dispersion" increases, also), while variability is much lower over the mean, conferming our intuition that attacks are typically **automated and repetitive**.

In appendix C it is provided a graphical representation in one feature case, with the boxplot of *ct_dst_ltm* (number of connections to the same destination address). We can graphically see that is different when an attack is happening, where it tends to increase: this is a typical behavior with DDoS attacks, brute-force or port scanning attacks that try to overwhelm a specific server and to target repeatedly the host.

C. Meaningful features for each attack category

As specified in the problem section, for analyzing most important features for predicting each attack type we have performed an exploratory data analysis using the *ANOVA* analysis: this statistical test allowed us to determine whether there are significant differences between the means of our independent groups.

The p-values we obtained from it were extremely small for all of our features. We could be mislead to think that all the features of our dataset are very valuable for all the attacks categories. But this is not totally true: these results are "inflated" by the huge sample size (more than 2 million). Therefore, any small difference in group means is "empathized" and the p-values are very small. For this reason, we also adopted another strategy: measuring the *eta squared*.

The eta-squared is a measure of the effect size and tells us more deeply about the strength of the relationship between the various groups.

An eta-squared value greater than 0.15 suggests a large effect size. The highest values were obtained for the generated variables *ct_** (always > 0.40): this means that not only they are useful to distinguish between anomalous and non anomalous situations, but also to discriminate which specific attack is performed in the network.

Other features for which we got a high effect size were *sttl* (0.31) (already identified as important for the binary classification task), *dttl* (0.44) and some time features as *tcprtt*, *synach* and *ackdat* (all > 0.30).

Furthermore we can say that some features relevant for the binary classification task, such as *Sload*, provided small effect size with the eta squared metric (0.006), suggesting that all the attacks type are similar in the ratio of bits per second sent by the source.

This analysis can be a starting point for further considerations about the attack types and their characteristics: it could be extended to identify which are the most relevant features to discriminate the various categories of attack.

Results and summary statistics computed across the 9 attack groups are available in the repository in the directory `/analysis/correlation_with_attack_cat/result`.

VI. Methodology

Before we delve into the performance evaluation of AI models, we summarize the methodology followed for the training and the evaluation phases.

A. Data Preprocessing

Our preprocessing pipeline included:

1) **Data Integration**: Instead of using an imbalanced dataset where the normal activity data entries are almost 7 times more than abnormal activities, we have used a separate, more close to the be called balanced training (175,341 records) and testing sets (82,332 records)
2) **Target Engineering**: We moved on by creating a binary target, counting all entries which are not explicitly mentioned as 'Normal' to be an attack one:

$$is\_attack = \begin{cases} 1 & \text{if } attack\_cat \neq \text{'Normal'} \\ 0 & \text{otherwise} \end{cases}$$

3) **Temporal Features**: We have also derived duration, hour, and day-of-week from timestamps of the given data.
4) **IP Removal**: We then eliminated FLOW features from our dataframe *srcip*, *dstip*, *sport* and *dsport* to prevent overfitting.
5) **Categorical Encoding**: To be able to work with the dataset we have also transformed proto, service, state using integer encoding.
6) **Normalization**: Finally we have applied Min-Max scaling for normalization purposes:

$$X_{norm} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

B. Model Framework

In the `/performance_evaluation/models.py` file in the GitHub repository, one can observe the model training pipeline used for the project. For performance evaluation we evaluated five models representing different learning paradigms:

- Gaussian Naive Bayes (probabilistic)
- Decision Tree (max_depth=8)
- Logistic Regression (L2 regularization)
- Random Forest (150 trees)
- Neural Network (MLP with two hidden layers, with 100 and 50 hidden units)

C. Evaluation Protocol

For the models' evaluation, we have put in practice the following computations:

- 5-fold stratified cross-validation.
- Metrics: Accuracy, Precision, Recall, F1, ROC-AUC.
- Statistical tests: paired t-tests.
- Confidence intervals: 95% CI for F1 and Accuracy metrics.

VII. Performance evaluation results

A. Performance analysis with the full feature set

To begin with, we trained and evaluated models with the entire feature set.

After the evaluation of the models, tree-based models (Random Forest and Decision Tree) combined with the Neural Network model have shown top performance, while Naive Bayes and Logistic Regression demonstrated lower performance metrics.

If we take a look to the Table I, at the appendix D we can see that **Random Forest** has shown very discriminative ability with close to perfect ROC-AUC (0.98). Although it has a lower F1 score than Decision Tree and Neural Network models, its cross-validated F1 (0.971 ± 0.0008) was the highest among all models, which gives us a ground to claim that it has a very high generalization capability.

The **Decision Tree** model has achieved the highest F1 test score (0.902) among all models, although it was one of the simplest models among the other ones. If we take a look at its confusion matrix, we can see that it shows precision-focused design. To end with the positive evaluation result, we can state that this model's training time was just 0.98 seconds.

The **Neural Network** model has proved to be among the top performing models. It has a detection capability with its second-highest F1 = 0.901 score, but it exhibited concerns about operational limitations. The confusion matrix reveals its key strength: second-best overall attack detection with a 97.3% recall with only 1,219 false negatives.

Logistic Regression and **Naive Bayes** models are the worst in the sense of performance. They both struggled with the complex decision boundaries in network traffic data, which becomes more evident when we look at their ROC-AUC scores (0.87 and 0.83 respectively). For very fast training and initial analysis the Naive Bayes model may be suitable as it is the fastest model in terms of running time (0.07s), but its 26% false negative rate renders it impractical for production security systems.

B. Class-specific performance patterns

Analysis of classification reports uncovered consistent patterns across models, with particularly interesting findings in attack detection performance.

We can take some considerations about the models' **recall** (how many attacks the model is able to detect) and **precision** (considering all the situations the model has predict an attack, how many of them are actually an attack; precision can be also interpreted also as *1-False Alarm Rate*).

Table II shows some key security trade-offs in models' performances:

Random Forest achieves the highest attack recall (98.50%) but it comes with the cost of a 19.7% increase in false positives compared to Decision Tree, this means that it catches the most attacks at the cost of more false alarms.

Decision Tree maintains the best precision (84.20%) and lowest false positive count. It misses just 1.41% more attacks than Random Forest.

Neural Network strikes a middle ground with 97.31% recall and just 3.1% more false positives than Decision Tree.

Logistic Regression and **Naive Bayes** nearly double the false alarms of Decision Tree while also missing a larger fraction of attacks, making them less suitable for production security systems. For environments sensitive to false alarms, Decision Tree offers the optimal balance—investigating 19.7% fewer false incidents than Random Forest while sacrificing only a minimal 1.41% in recall.

Analysis of confusion matrices in the Figure 3, in Appendix D, reveals critical operational patterns:

- **Decision Tree**: high false positives (8,261) would create significant alert workload.
- **Naive Bayes** and **Logistic Regression**: dangerous false negatives (16,494 missed attacks).
- **Random Forest**: *best balance* with moderate false positives (9,886) and fewest false negatives (682).
- **Neural Network**: Competitive detection with 8,516 false positives and 1,219 false negatives.

So overall, random forests should be the default choice for security applications using similar network traffic data, as maximum detection accuracy is critical in intrusion detection systems.

On the other hand, linear models are unsuitable for production-ready network security applications.

Some extra details about the models' computational efficiency and operational impact is provided in the appendix E.

C. Performance analysis across feature sets

To validate the statistical and feature analysis done in the first part of this project, we have properly selected **three** feature sets in which we have progressively remove redundant features and just maintained the most correlated ones with the attack label.

In this way, we could be able to verify if the statistical results we obtained are really significant in a real scenario or not.

The three features sets are provided in the Appendix F.

In this section, we will explore how their performances vary, with particular focus on accuracy, stability and feature set sensitivity.

Key observations from Table III in the Appendix F:

- **Random Forest** has shown a very high stability across different sets, having the smallest accuracy range (Δ=0.031) and lowest standard deviation (σ=0.016). This consistency is remarkable given the fact that the features in set 3 and set 4 were drastically decreased in size. This confirms its robustness for production environments where feature availability may vary.
- **Decision Tree** showed alarming sensitivity to feature selection with the largest accuracy fluctuation (Δ=0.074). The 8.4% performance drop from Set 1 (0.884) to Set 3 (0.810) indicates critical dependency on specific features, making it unreliable for dynamic environments.
- **Neural Network** illustrated the biggest performance degradation (9.2% accuracy drop from Set 1 to Set 4). The high standard deviation (σ=0.044) confirms its instability when features are modified or reduced.
- **Naive Bayes** maintained consistent but very low accuracy throughout the sets. The performance (mean 0.744), with its limited learning capacity, prevents both significant improvements and severe degradation across configurations.

Figure 4 visually confirms these patterns, showing Random Forest's minimal performance variation compared to other models' dramatic swings. To quantify these differences, we conducted rigorous pairwise statistical comparisons between feature sets (Table IV).

The statistical analysis proved to be very useful, with the most important insights discussed below:

- **Set 3 limitations**: the significant differences between Set 3 and both Set 1 (p=0.043) and Set 2 (p=0.010) represent operationally meaningful accuracy gaps (4.6% and 4.1% respectively). In security contexts, this difference could translate to thousands of undetected attacks monthly.
- **Set 2 efficiency**: the non-significant difference between Set 1 and Set 2 (p=0.553) despite Set 2 having fewer features makes it a compelling option. Organizations could reduce feature collection costs by 30% without statistically significant accuracy loss.
- **Set 4 anomaly**: while Set 4 showed non-significant differences versus Set 1 (p=0.065), its significant difference versus Set 2 (p=0.013) suggests that In this scenario, quantity means also quality, as Set 2's carefully selected features outperformed Set 4's smaller set.
- **Consistency patterns**: the non-significant Set 3 vs Set 4 comparison (p=0.336) indicates these configurations share similar limitations, with neither providing adequate detection capability.

Figure 5 in Appendix F provides a strategic visualization of the stability-accuracy tradeoff:

- **Random Forest** occupies the optimal upper-left quadrant, achieving the ideal balance of high accuracy (x-axis) and low variation (y-axis position) between all features sets.
- **Decision Tree** shows high peak performance but extreme variance, making it a high-risk option.
- **Neural Network** demonstrates the worst stability profile, with both low average accuracy and high fluctuation.
- **Linear models**, even though they are more stable, they proved to have very low performance across all the metrics.

Based on our comprehensive analysis, we can provide some operational recommendations:

- **High-security environments**: set 1 with Random Forest maximizes detection (accuracy 0.872) with minimal performance risk ($\sigma = 0.016$).
- **Cost-sensitive deployments**: set 2 with Random Forest provides statistically equivalent accuracy to Set 1 (p=0.553) with 30% fewer features.
- **Edge computing**: set 2 with Decision Tree balances accuracy (0.874) and efficiency, though with higher operational risk due to instability.
- **Legacy systems**: set 1 with Naive Bayes provides minimal hardware requirements but at significant detection cost (26% false negatives).
- **Set avoidance**: set 3 should be excluded from consideration due to significant accuracy drops across all models ($p < 0.05$ vs. richer sets).

These findings fundamentally change feature selection strategy for intrusion detection. Rather than maximizing feature quantity, optimal performance comes from strategic selection validated through rigorous statistical testing. The significant performance differences ($p < 0.05$) between feature configurations demonstrate that feature quality directly determines detection capability, with Set 1 and Set 2 providing the only viable options for production systems.

VIII. Conclusion

In this work we conducted a comprehensive statistical analysis to the dataset prior to the training and the evaluation of Machine Learning and Deep Learning-based Intrusion Detection System.

To begin with, we analyzed the features correlations to identify the most irrelevant and redundant information that could introduce multicollinearity problems in the models. As it could be expected, features representing bytes sent, load of the source and number of packets sent were highly correlated within each other.

Next, we investigated the most relevant features in the classification task, by verifying the correlations of the numerical features with the target label (discriminating anomalous and non-anomalous traffic scenarios). The features showing stronger correlation were actually the ones with the most observable differences between summary statistics (e.g., mean, median and standard deviation) when computed separately for attack and normal cases. One particularly important finding was that the coefficient of variation tended to be lower during attacks, suggesting that attacks patterns are typically repetitive and automated.

Finally, we also provided some insights on the most and the least discriminative features to distinguish between attacks. This analysis lays a foundation for more granular future search into attack categorization.

In the performance evaluation part, we have seen that in general, tree-based models, especially Random Forest, offer the best balance of accuracy and stability, outperforming other models. Decision Tree is a fast alternative with competitive F1, while Neural Network achieves similar detection at a much higher computational cost. Logistic Regression and Naive Bayes lag in both accuracy and reliability.

Regarding feature selection, Set1 (full features) and Set2 (reduced subset) yield equivalent detection performance, with Set2 offering reduced overhead, thanks to the reduction of just over 20 features. Set3 consistently underperforms and should be avoided. Overall, Random Forest with either Set1 or Set2 is recommended for production deployments, combining high detection rates with operational efficiency.

To sum up, we can say we have validated our hypotheses regarding the removal of redundant features and the retention of only those most closely related to the target label. We have identified a smaller feature set that still guarantees performances comparable to the full set of features.

Future works could be done to improve the analysis of the features that are helpful to discriminate between attack categories. Additionally, it is possible to do an evaluation of AI models in a multiclass classification setting, by considering only the most relevant features (for inter-attack category discrimination) and verify if we get positive results as in the binary setting.

References

[1] Nour Moustafa & Jill Slay, (2016), "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set", Information Security Journal: A Global Perspective, 25:1-3, 18-31, DOI: 10.1080/19393555.2015.1125974.

[2] Zeinab Zoghi & Gursel Serpen, (2021), "UNSW-NB15 Computer Security Dataset: Analysis through Visualization", DOI: https://arxiv.org/abs/2101.05067

[3] Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015. DOI: 10.1109/MilCIS.2015.7348942.

APPENDIX A
HEATMAP OF CORRELATIONS

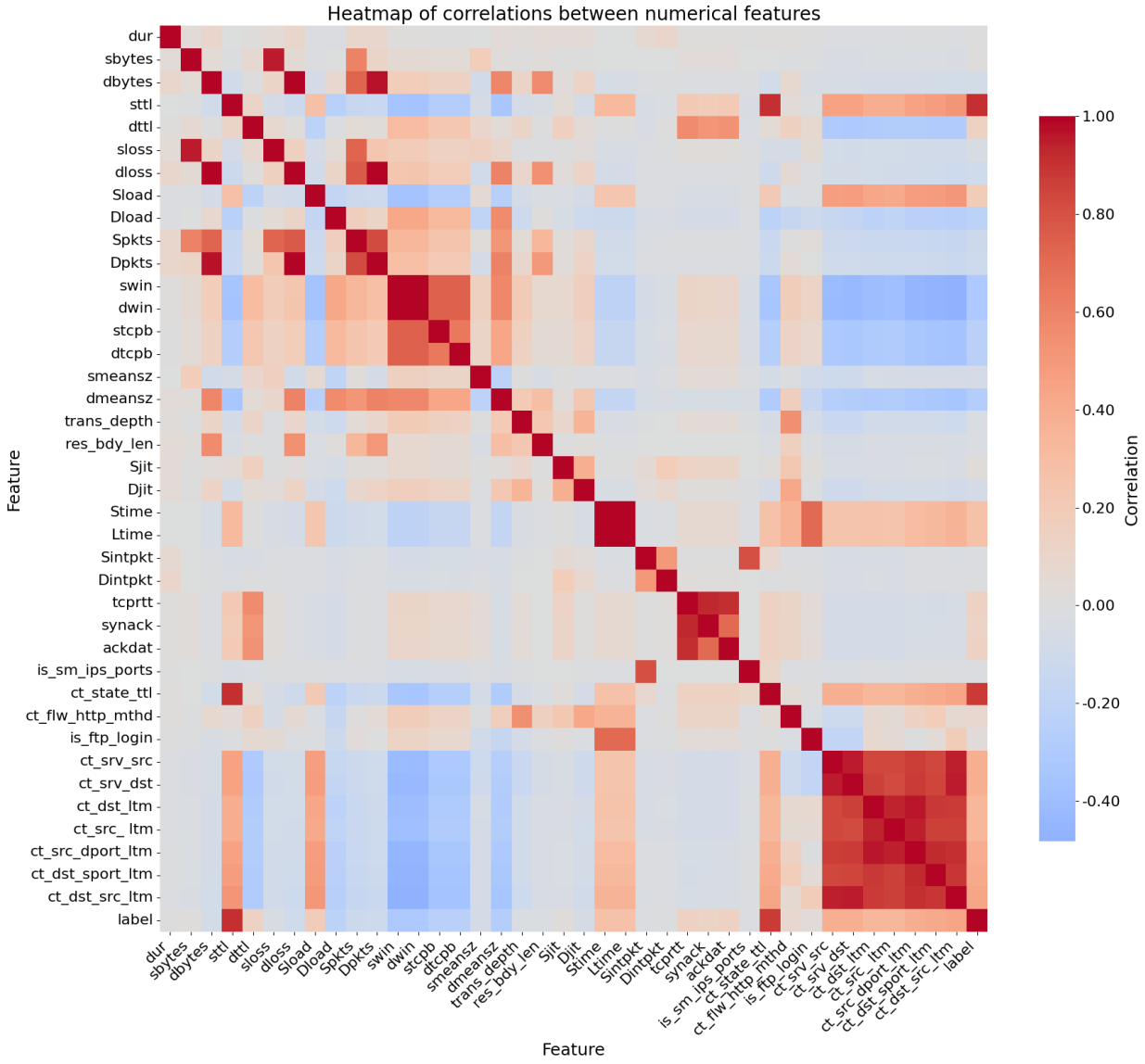

Fig. 1. Heatmap of correlations

APPENDIX B
SUMMARY STATISTIC OF TOP 10 FEATURES CORRELATED WITH ATTACK LABEL

First, the table to show the summary statistics when NO ATTACK is performed.

Feature	Mean with 95% CI	Std Dev	Median	.25 quantile	.75 quantile	CoV
sttl	37.101 [37.063, 37.138]	28.291	31.000	31.000	31.000	0.763
ct_state_ttl	0.034 [0.034, 0.034]	0.306	0.000	0.000	0.000	8.978
ct_dst_src_ltm	4.961 [4.949, 4.973]	9.143	2.000	1.000	4.000	1.843
ct_dst_sport_ltm	2.608 [2.601, 2.615]	5.123	1.000	1.000	1.000	1.964
ct_src_dport_ltm	3.362 [3.353, 3.371]	6.960	1.000	1.000	2.000	2.070
ct_srv_dst	7.397 [7.385, 7.409]	9.024	4.000	2.000	8.000	1.220
ct_srv_src	7.628 [7.616, 7.640]	9.088	5.000	2.000	9.000	1.191
ct_src_ltm	5.829 [5.820, 5.838]	6.894	4.000	2.000	6.000	1.183
ct_dst_ltm	5.385 [5.376, 5.394]	6.832	3.000	2.000	5.000	1.269
Sload	2.83×10^7 [2.81×10^7, 2.84×10^7]	1.10×10^8	5.59×10^5	1.16×10^5	1.45×10^6	3.877

Finally, we show the table with the summary statistics when an ATTACK is performed.

Feature	Mean with 95% CI	Std Dev	Median	.25 quantile	.75 quantile	CoV
sttl	240.136 [239.963, 240.308]	49.881	254.000	254.000	254.000	0.208
ct_state_ttl	1.830 [1.828, 1.831]	0.477	2.000	2.000	2.000	0.261
ct_dst_src_ltm	19.861 [19.809, 19.914]	15.195	20.000	4.000	33.000	0.765
ct_dst_sport_ltm	10.392 [10.364, 10.421]	8.198	10.000	1.000	17.000	0.789
ct_src_dport_ltm	13.481 [13.440, 13.523]	12.008	13.000	2.000	20.000	0.891
ct_srv_dst	19.982 [19.930, 20.033]	15.011	20.000	4.000	33.000	0.751
ct_srv_src	20.114 [20.063, 20.166]	14.896	20.000	5.000	33.000	0.741
ct_src_ltm	14.302 [14.261, 14.343]	11.889	14.000	3.000	21.000	0.831
ct_dst_ltm	13.720 [13.679, 13.762]	11.984	13.000	2.000	20.000	0.873
Sload	9.69×10^7 [9.63×10^7, 9.74×10^7]	1.55×10^8	5.70×10^7	4.56×10^7	1.14×10^8	1.602

APPENDIX C
BOXPLOT OF *ct_dst_ltm*

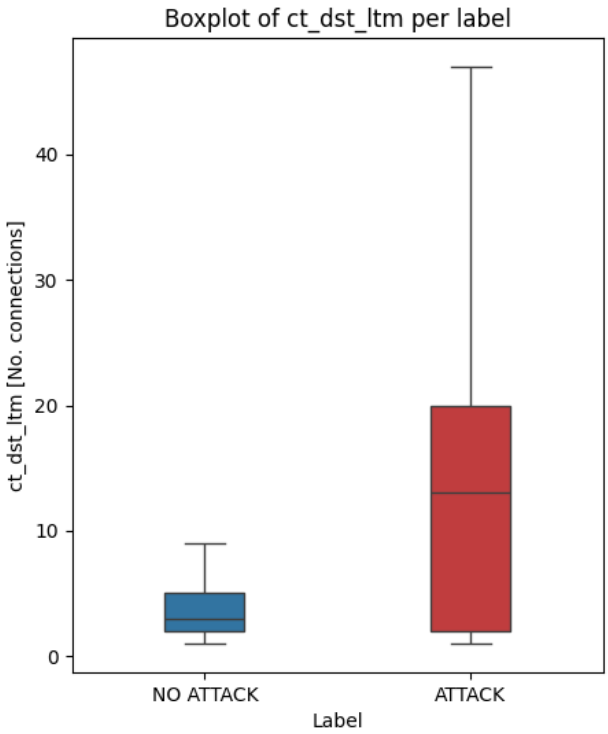

Fig. 2. Boxplot of ct_dst_ltm

APPENDIX D
PERFORMANCE EVALUATION RESULTS WITH FULL FEATURE SET

TABLE I
BINARY CLASSIFICATION PERFORMANCE METRICS WITH FULL FEATURE SET

Model	Acc.	Precision	Recall	F1	CV F1	AUC
DT	0.884	0.842	0.971	0.902	0.955 ± 0.0016	0.98
RF	0.872	0.819	0.985	0.894	0.971 ± 0.0008	0.98
NN	0.882	0.838	0.973	0.901	0.964 ± 0.0011	0.97
LR	0.778	0.725	0.959	0.826	0.948 ± 0.0006	0.87
NB	0.740	0.710	0.892	0.791	0.901 ± 0.0005	0.83

TABLE II
ATTACK DETECTION PERFORMANCE COMPARISON

Model	Attack Recall	Attack Precision	FP Increase
Random Forest	98.50%	81.87%	+19.7% vs. DT
Decision Tree	97.09%	84.20%	—
Neural Network	97.31%	83.82%	+3.1% vs. DT
Logistic Regression	95.89%	72.55%	+99.1% vs. DT
Naive Bayes	89.16%	71.02%	+99.7% vs. DT

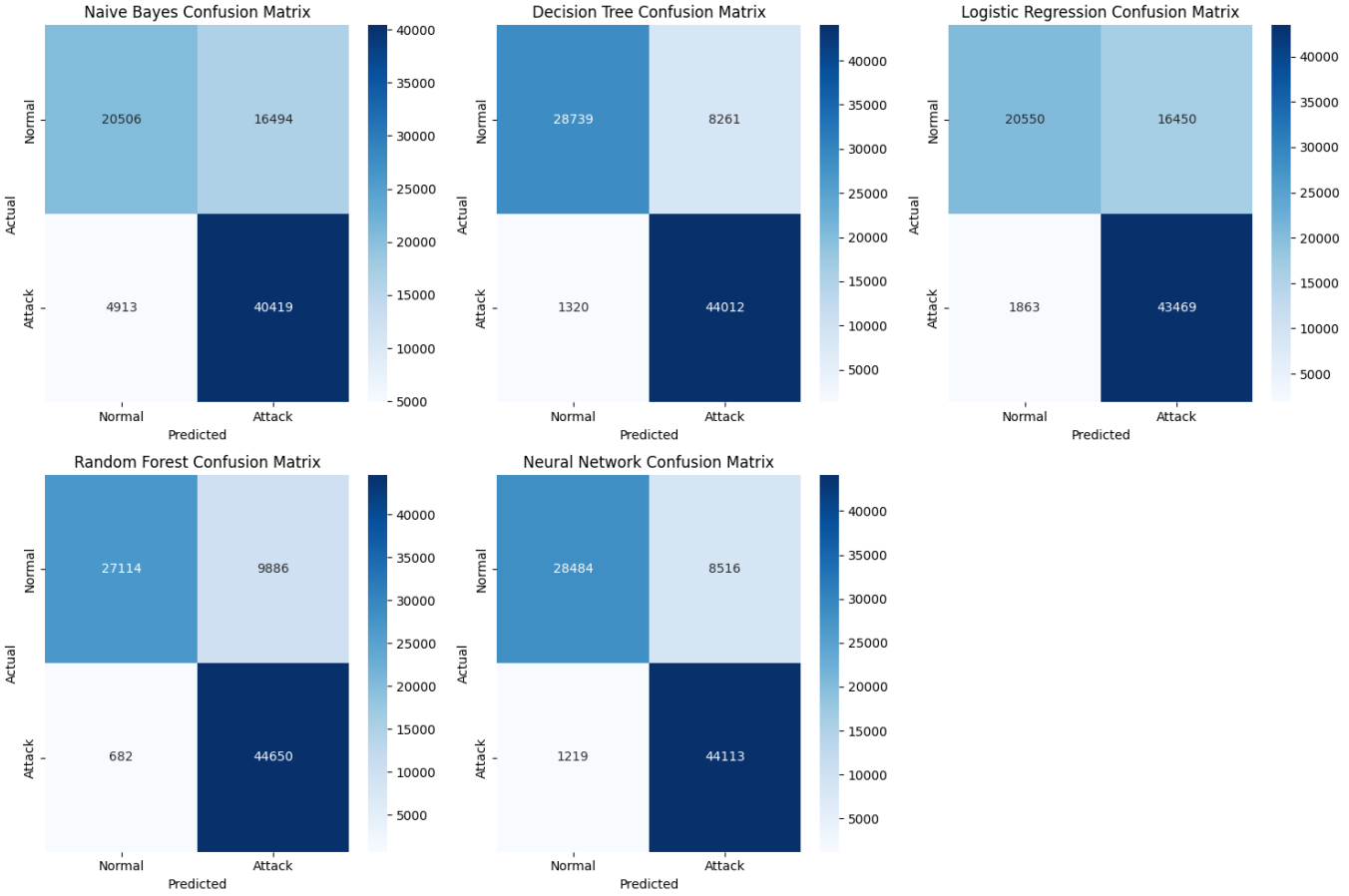

Fig. 3. Confusion matrices showing model comparison

APPENDIX E
COMPUTATIONAL EFFICIENCY AND OPERATIONAL IMPACT

Training time analysis reveals significant practical implications:

- **Training Speed**: Naive Bayes (0.07s) < Logistic Regression (1.05s) < Decision Tree (0.98s) < Random Forest (5.22s) < Neural Network (56.52s).
- **Retraining Feasibility**: Naive Bayes enables near-real-time updates while Neural Network causes 15-20 minute delays in real and larger datasets.
- **Resource Efficiency**: Decision Tree delivers 90% of Random Forest's detection with 19% training time.

Operational impact calculations:

- **False Alarm Cost**: Decision Tree causes 19,700 fewer false alarms daily than Random Forest per 1M connections.
- **Risk Trade-off**: Random Forest detects 1.41% more attacks, justifying overhead in high-risk environments.

APPENDIX F
PERFORMANCE EVALUATION RESULTS WITH PARTIAL FEATURE SETS

Set 1 (Full set)

`All original features used in the study (full feature list)`

Set 2

`'proto', 'service', 'state', 'spkts', 'sloss', 'dloss', 'sload', 'rate', 'sttl', 'dttl', 'swin', 'stcpb', 'dtcpb', 'sinpkt', 'tcprtt', 'dmean', 'trans_depth', 'response_body_len', 'ct_state_ttl', 'ct_dst_src_ltm', 'ct_dst_ltm', 'ct_srv_src', 'ct_flw_http_mthd'`

Set 3

`'sttl', 'ct_dst_src_ltm', 'ct_flw_http_mthd', 'sload', 'tcprtt', 'dttl', 'proto', 'service', 'state'`

Set 4

`'sttl', 'ct_dst_src_ltm', 'sload', 'ackdat', 'is_ftp_login'`

TABLE III
ACCURACY PERFORMANCE ACROSS FEATURE SETS

Model	**Set 1**	**Set 2**	**Set 3**	**Set 4**
Random Forest	0.872	0.867	0.841	0.845
Decision Tree	0.884	0.874	0.810	0.835
Neural Network	0.882	0.851	0.796	0.790
Logistic Regression	0.778	0.780	0.735	0.759
Naive Bayes	0.740	0.757	0.743	0.735

TABLE IV
STATISTICAL COMPARISON OF FEATURE SETS (ACCURACY)

Comparison	**p-value**	**Significant**	**Accuracy Δ**
Set 1 vs Set 3	0.043	Yes	+0.046
Set 2 vs Set 3	0.010	Yes	+0.041
Set 2 vs Set 4	0.013	Yes	+0.033
Set 1 vs Set 4	0.065	No	+0.038
Set 1 vs Set 2	0.553	No	+0.005
Set 3 vs Set 4	0.336	No	-0.008

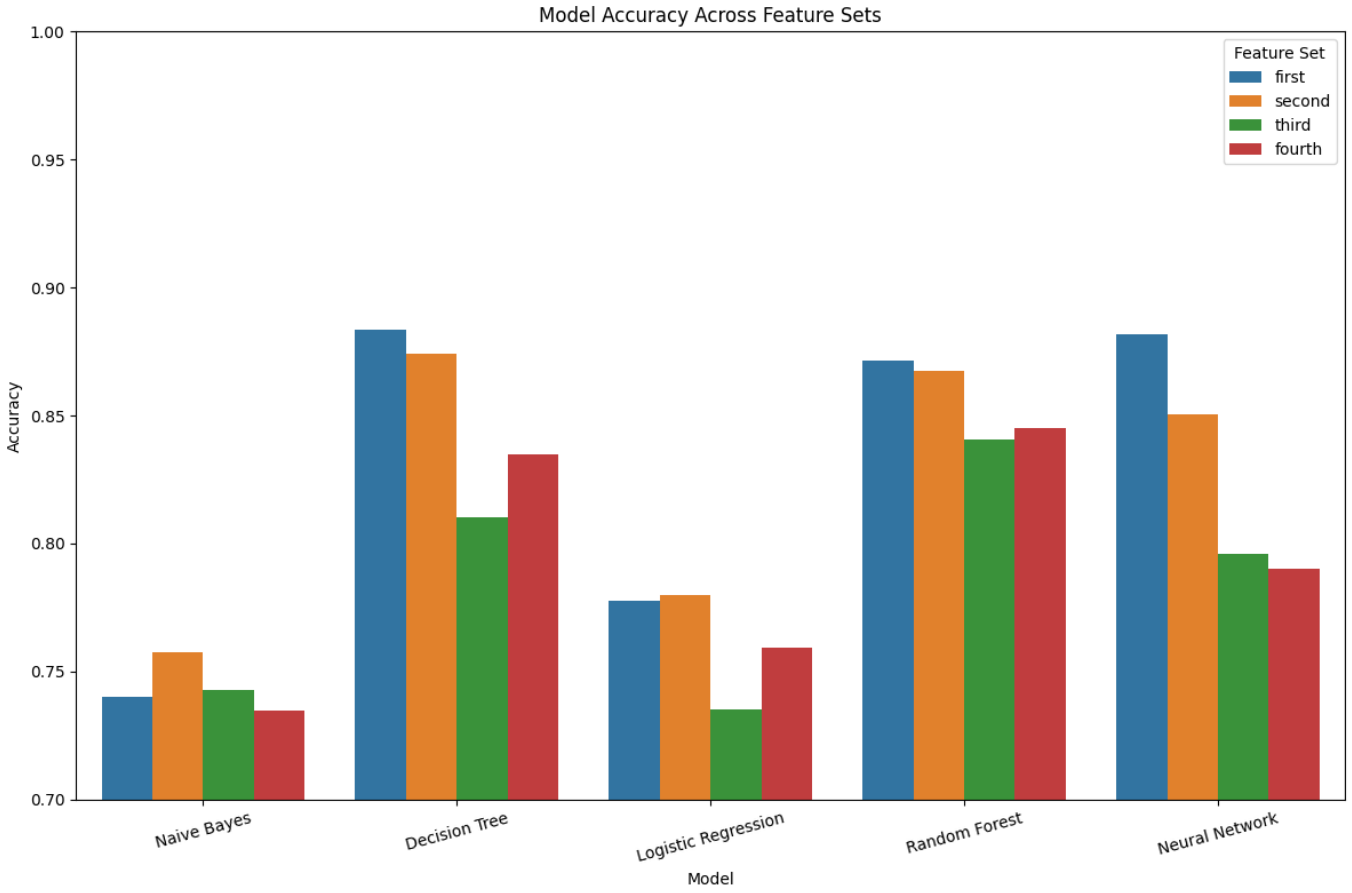

Fig. 4. Accuracy comparison across feature sets

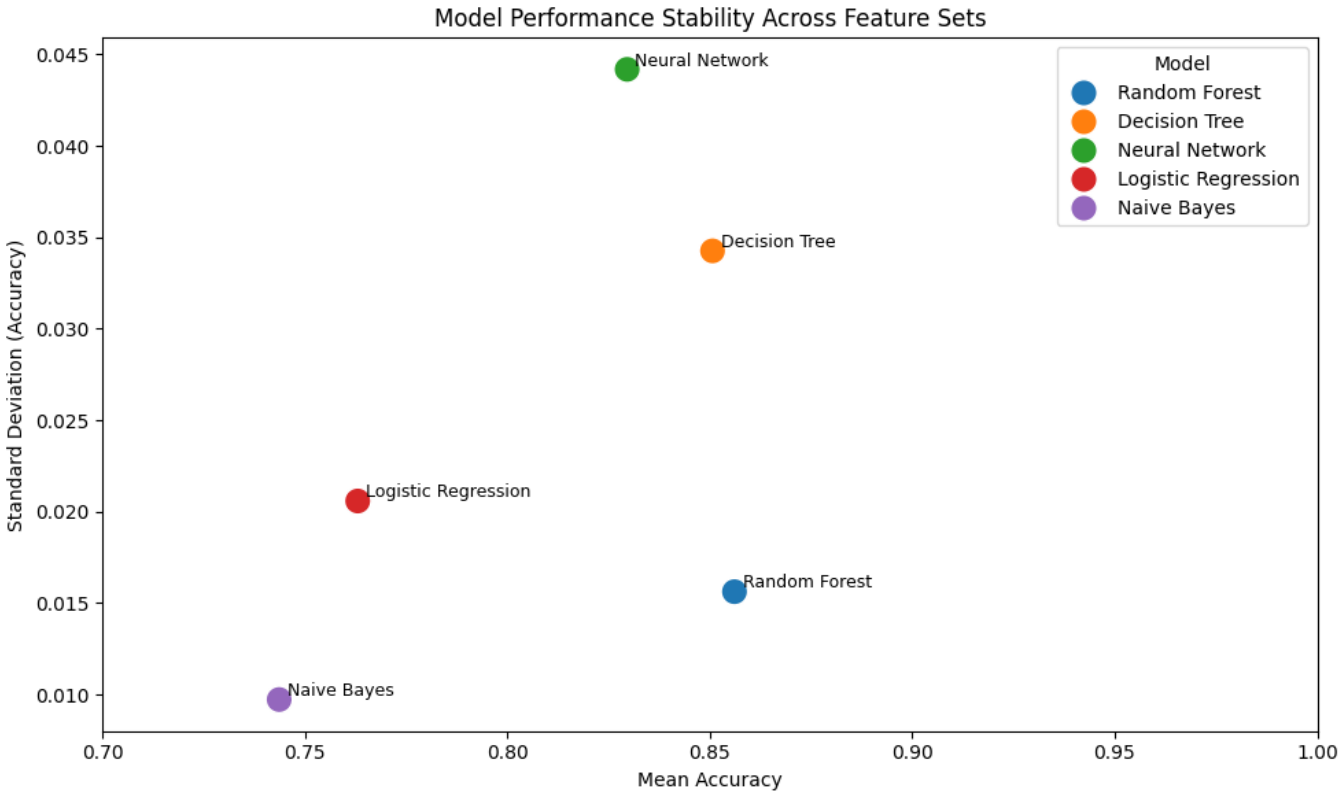

Fig. 5. Model performance stability Across feature sets