

UNIVERSITY OF WALES TRINITY SAINT DAVID(UWTSD)

Assignment – ML Model

Md.Nabil Rahman

ID: 2504501

Course Code: ACCA7026

Course Name: L7-Artificial Intelligence

Lecturer: Dr. Seena Joseph

MSc Software Engineering and Artificial Intelligence

December 9th 2025

Abstract

The creation and assessment of machine learning-based intrusion detection models utilizing the NSL-KDD dataset are presented in this research. Two ensemble approaches, Random Forest and Gradient Boosting, were trained to conduct multiclass classification of network intrusions, including Dos, Probe, R2L, U2R, and Normal traffic. The dataset underwent two preparation steps: encoding category variables and scaling numerical characteristics. Metrics like these were used to assess the models accuracy, F1-score, and confusion matrices. The results demonstrate the potent detection capabilities of ensemble learning, with Random Forest exhibiting exceptional performance and Gradient Boosting providing greater sensitivity to minority attack classes. Feature importance analysis, which identified important factors influencing intrusion detection, generated suggestions for enhancing IDS design. The outcomes demonstrate the effectiveness of ensemble-based machine learning algorithms in modern cybersecurity applications.

Contents

Abstract	i
List of Figures	iii
1 Introduction	1
1.1 Introduction	1
1.2 Dataset Description	1
2 Methodology	2
2.1 Data Preprocessing	2
2.2 Model Development	2
2.2.1 Evaluation Metrics	2
3 EXPERIMENTS AND RESULTS ANALYSIS	3
3.1 Load the NSL-KDD dataset	3
3.2 Show all the information about dataset	4
3.3 Generate summary statistics for the numeric columns	4
3.4 Generate Histograms for the numeric columns	5
3.5 Generate bar chart for categorical column (except protocol type)	5
3.6 Create heatmap for numeric columns	5
4 RESULTS AND DISCUSSIONS	6
4.1 Random Forest Classifier	6
4.2 Gradient Boosting	9
4.3 Discussion	11
4.3.1 Random Forest (RF) Classifier	11
4.3.2 Key Limitations (Overfitting and Imbalance):	11
4.4 Gradient Boosting Classifier	11
4.4.1 Key Limitations:	12
5 Conclusion	12
6 REFERENCES	13

List of Figures

3.1	Test dataset	3
3.2	Train dataset	3
3.3	Train dataset information	4
3.4	summary statistics for the numeric columns	4
3.5	Generate Histrograms	5
3.6	Generate bar char	5
3.7	Generate Heatmap	5
4.8	Random Forest classification report	6
4.9	Trained the model	6
4.10	Classification report	7
4.11	Random Forest Confusion Matrix	7
4.12	Normalized Confusion Matrix	8
4.13	Random Forest Feature Importances	8
4.14	Gradient Boosting classification report	9
4.15	Gradient Boosting Confusion Matrix	9
4.16	Gradient Boosting Normalized Confusion Matrix	10
4.17	Feature Importances	10
4.18	Model Accuracy Comparison	10

1 Introduction

1.1 Introduction

Protecting digital infrastructures from changing cyberattacks is a critical function of cybersecurity. Intrusion Detection Systems (IDS) help identify malicious or abnormal network activities. Traditional signature-based IDS approaches often fail to detect unknown attacks, whereas machine learning techniques—especially ensemble methods—have shown strong performance and adaptability in IDS tasks [6], [7].

“For this project, I selected the NSL-KDD dataset because it addresses several shortcomings of KDD’99, such as excessive redundancy. The dataset offers more balanced attack categories, which makes model evaluation more reliable. [1].

1.2 Dataset Description

The NSL-KDD dataset is widely recommended for IDS research due to its balanced samples, removal of duplicates, and enhanced representativeness compared to the original KDD 99 dataset [1], [5]. Network connection records classified as either normal traffic or one of several attack categories make up the NSL-KDD dataset. Each record contains 41 features describing properties of the connection, such as protocol type, bytes transferred, number of failed logins, and error rates.

Attacks fall into four classes:

- Denial of Service (DoS): Overloading a system to prevent services from being provided.
- Probe: Information-gathering surveillance and scanning.
- Unauthorized access from a distant machine is known as R2L (Remote-to-Local).
- U2R (User-to-Root): Attempt to gain superuser privileges.

A variety of network traffic characteristics, including TCP/IP packet contents, error rates, connection counts, and behavior-based indications, are represented by these features. By removing redundancy and balancing samples for a more trustworthy assessment, the dataset outperforms KDD’99.

2 Methodology

2.1 Data Preprocessing

The dataset was cleaned, categorical characteristics were one-hot encoded, and numerical features were standardized in accordance with best practices in IDS preprocessing [5]. To provide a fair assessment, train/test splits were developed.

- One-Hot Encoding was used to translate categorical features (protocol type, service, flag).
- Numerical features were standardized using StandardScaler.
- Attack labels were mapped to the four attack classes and "normal".
- Training and testing sets were prepared using KDD Train and KDD Test

2.2 Model Development

Because of their shown effectiveness in cybersecurity applications, two ensemble learning algorithms were chosen:

- Random Forest (RF): Uses bagging and decision tree ensembles [3]
- Gradient Boosting (GB): Sequential boosting approach minimizing residual errors [4]

Both algorithms are known to outperform traditional ML models in IDS tasks [6], [7].

2.2.1 Evaluation Metrics

- Accuracy
- F1-Score (Macro)
- Classification Report (Precision, Recall)
- Confusion Matrix

3 EXPERIMENTS AND RESULTS ANALYSIS

3.1 Load the NSL-KDD dataset

Import And Explore test and train dataset, Each dataset has 20 rows $\times$ 42 columns.

```
# Import test dataset
tf=pd.read_csv('kdd_test.csv')
```

```
#Explore the dataset
tf.head(20)
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate
0	5	tcp	smtp	SF	2429	475	0	0	0	0	...	57	0.11	0.02
1	0	udp	domain_u	SF	45	134	0	0	0	0	...	155	0.80	0.02
2	0	udp	domain_u	SF	45	80	0	0	0	0	...	255	1.00	0.00
3	1979	udp	other	SF	145	105	0	0	0	0	...	2	0.01	0.84
4	14462	tcp	other	RSTR	1	0	0	0	0	0	...	2	0.01	0.68
5	0	tcp	private	S0	0	0	0	0	0	0	...	9	0.04	0.07
6	0	tcp	http	SF	294	5499	0	0	0	0	...	255	1.00	0.00
7	0	tcp	http	S0	0	0	0	0	0	0	...	44	0.17	0.03
8	0	icmp	ecr_i	SF	1032	0	0	0	0	0	...	255	1.00	0.00
9	0	udp	private	SF	1	0	0	0	0	0	...	32	0.13	0.45
10	0	tcp	mtp	REJ	0	0	0	0	0	0	...	6	0.02	0.08
11	0	tcp	netstat	RSTR	0	0	0	0	0	0	...	1	0.01	0.17
12	0	tcp	private	S0	0	0	0	0	0	0	...	6	0.02	0.09
13	0	tcp	private	S0	0	0	0	0	0	0	...	9	0.04	0.07

Figure 3.1: Test dataset

```
# Import test dataset
df = pd.read_csv('kdd_train.csv')
```

```
#Explore the dataset
df.head(20)
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	25	0.17	0.03
1	0	udp	other	SF	146	0	0	0	0	0	...	1	0.00	0.60
2	0	tcp	private	S0	0	0	0	0	0	0	...	26	0.10	0.05
3	0	tcp	http	SF	232	8153	0	0	0	0	...	255	1.00	0.00
4	0	tcp	http	SF	199	420	0	0	0	0	...	255	1.00	0.00
5	0	tcp	private	REJ	0	0	0	0	0	0	...	19	0.07	0.07
6	0	tcp	private	S0	0	0	0	0	0	0	...	9	0.04	0.05
7	0	tcp	private	S0	0	0	0	0	0	0	...	15	0.06	0.07
8	0	tcp	remote_job	S0	0	0	0	0	0	0	...	23	0.09	0.05
9	0	tcp	private	S0	0	0	0	0	0	0	...	13	0.05	0.06
10	0	tcp	private	REJ	0	0	0	0	0	0	...	12	0.05	0.07
11	0	tcp	private	S0	0	0	0	0	0	0	...	13	0.05	0.07
12	0	tcp	http	SF	287	2251	0	0	0	0	...	219	1.00	0.00
13	0	tcp	ftp_data	SF	334	0	0	0	0	0	...	20	1.00	0.00
14	0	tcp	name	S0	0	0	0	0	0	0	...	1	0.00	0.07

Figure 3.2: Train dataset

3.2 Show all the information about dataset

```
#Show all the information about dataset
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 125973 entries, 0 to 125972
Data columns (total 42 columns):
#   Column                Non-Null Count  Dtype
---  -
0   duration              125973 non-null  int64
1   protocol_type         125973 non-null  object
2   service               125973 non-null  object
3   flag                 125973 non-null  object
4   src_bytes             125973 non-null  int64
5   dst_bytes             125973 non-null  int64
6   land                 125973 non-null  int64
7   wrong_fragment       125973 non-null  int64
8   urgent               125973 non-null  int64
9   hot                  125973 non-null  int64
10  num_failed_logins     125973 non-null  int64
11  logged_in            125973 non-null  int64
12  num_compromised      125973 non-null  int64
13  root_shell          125973 non-null  int64
14  su_attempted         125973 non-null  int64
15  num_root             125973 non-null  int64
16  num_file_creations   125973 non-null  int64
17  num_shells           125973 non-null  int64
18  num_access_files     125973 non-null  int64
19  num_outbound_cmds    125973 non-null  int64
20  is_host_login        125973 non-null  int64
21  is_guest_login       125973 non-null  int64
22  count                125973 non-null  int64
23  srv_count            125973 non-null  int64
24  serror_rate          125973 non-null  float64
25  srv_serror_rate      125973 non-null  float64
26  rerror_rate          125973 non-null  float64
27  srv_rerror_rate      125973 non-null  float64
28  same_srv_rate        125973 non-null  float64
```

Figure 3.3: Train dataset information

After Showing all the information about dataset we can find numeric and categorical data. and explore the unique types of data in each columns.

3.3 Generate summary statistics for the numeric columns

```
# Generate summary statistics for the numeric columns
df[num_columns].describe().T
```

	count	mean	std	min	25%	50%	75%	max
duration	125973.0	287.144650	2.604515e+03	0.0	0.00	0.00	0.00	4.290800e+04
src_bytes	125973.0	45566.743000	5.870331e+06	0.0	0.00	44.00	276.00	1.379964e+09
dst_bytes	125973.0	19779.114421	4.021269e+06	0.0	0.00	0.00	516.00	1.309937e+09
logged_in	125973.0	0.395736	4.890101e-01	0.0	0.00	0.00	1.00	1.000000e+00
count	125973.0	84.107555	1.145086e+02	0.0	2.00	14.00	143.00	5.110000e+02
srv_count	125973.0	27.737888	7.263584e+01	0.0	2.00	8.00	18.00	5.110000e+02
serror_rate	125973.0	0.284485	4.464556e-01	0.0	0.00	0.00	1.00	1.000000e+00
srv_serror_rate	125973.0	0.282485	4.470225e-01	0.0	0.00	0.00	1.00	1.000000e+00
rerror_rate	125973.0	0.119958	3.204355e-01	0.0	0.00	0.00	0.00	1.000000e+00
srv_rerror_rate	125973.0	0.121183	3.236472e-01	0.0	0.00	0.00	0.00	1.000000e+00
same_srv_rate	125973.0	0.660928	4.396229e-01	0.0	0.09	1.00	1.00	1.000000e+00
diff_srv_rate	125973.0	0.063053	1.803144e-01	0.0	0.00	0.00	0.06	1.000000e+00
srv_diff_host_rate	125973.0	0.097322	2.598305e-01	0.0	0.00	0.00	0.00	1.000000e+00
dst_host_count	125973.0	182.148945	9.920621e+01	0.0	82.00	255.00	255.00	2.550000e+02
dst_host_srv_count	125973.0	115.653005	1.107027e+02	0.0	10.00	63.00	255.00	2.550000e+02
dst_host_same_srv_rate	125973.0	0.521242	4.489494e-01	0.0	0.05	0.51	1.00	1.000000e+00
dst_host_diff_srv_rate	125973.0	0.082951	1.889218e-01	0.0	0.00	0.02	0.07	1.000000e+00

Figure 3.4: summary statistics for the numeric columns

After Identify numeric and categorical data Generate a summary of all numerical data.

3.4 Generate Histograms for the numeric columns

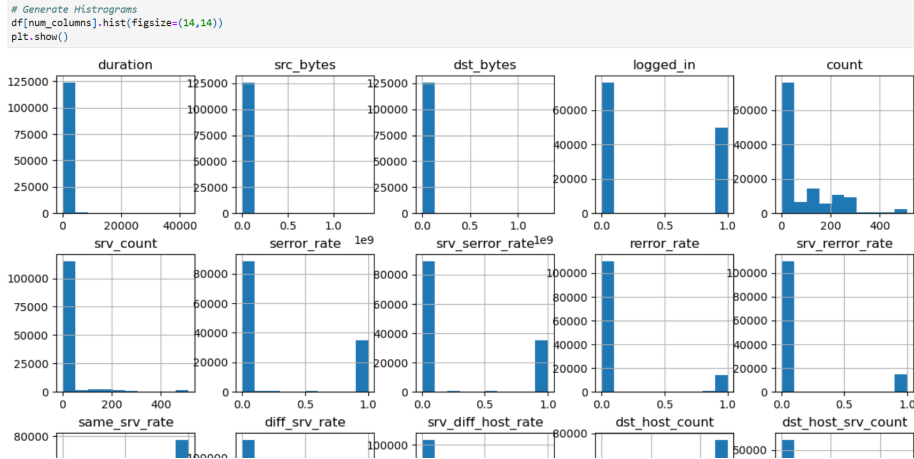


Figure 3.5: Generate Histograms

3.5 Generate bar chart for categorical column (except protocol type)

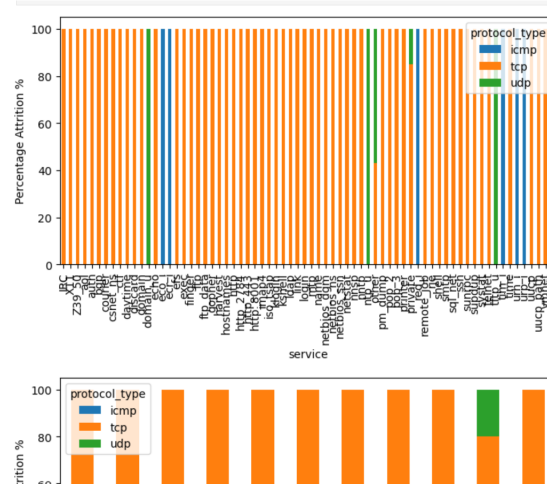


Figure 3.6: Generate bar chart

3.6 Create heatmap for numeric columns

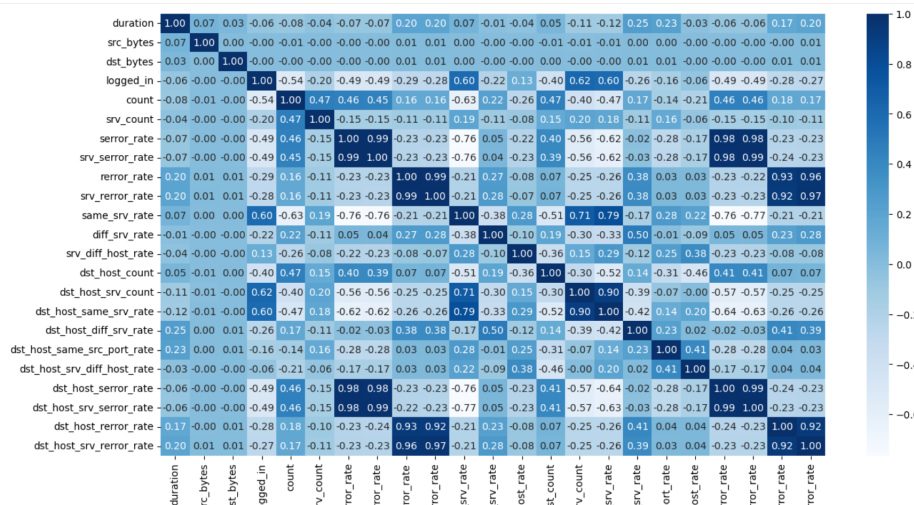


Figure 3.7: Generate Heatmap

The first figure contains multiple histograms, each corresponding to one numerical feature in the dataset. The second figure shows bar charts for categorical variables. And the third figure is a correlation heatmap.

4 RESULTS AND DISCUSSIONS

4.1 Random Forest Classifier

```

: #Model 1: Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42, class_weight='balanced')
rf_model.fit(X_train_scaled, y_train)
y_pred = rf_model.predict(X_test_scaled)
print(classification_report(y_test, y_pred))

```

	precision	recall	f1-score	support
back	1.00	1.00	1.00	287
buffer_overflow	1.00	0.44	0.62	9
ftp_write	1.00	0.50	0.67	2
guess_passwd	1.00	0.88	0.93	16
imap	1.00	0.67	0.80	3
ipsweep	0.99	1.00	1.00	1000
land	0.60	0.60	0.60	5
loadmodule	0.00	0.00	0.00	3
multihop	0.00	0.00	0.00	2
neptune	1.00	1.00	1.00	12364
nmap	0.99	0.99	0.99	448
normal	1.00	1.00	1.00	20203
perl	0.00	0.00	0.00	1
phf	0.00	0.00	0.00	1
pod	0.98	1.00	0.99	60
portsweep	1.00	1.00	1.00	879
rootkit	0.00	0.00	0.00	3
satan	1.00	0.99	0.99	1090
smurf	1.00	1.00	1.00	794
spy	0.00	0.00	0.00	1
teardrop	1.00	1.00	1.00	268
warezclient	0.99	0.97	0.98	267
warezmaster	1.00	0.67	0.80	6
accuracy			1.00	37792
macro avg	0.72	0.64	0.67	37792
weighted avg	1.00	1.00	1.00	37792

Figure 4.8: Random Forest classification report

The classification report produced following Random Forest classifier training is displayed in this table in the picture. Usually, the classification report consists of:

- Precision – of all predicted positives, how many were correct
- Recall – of all actual positives, how many were detected
- F1-score – harmonic mean of precision and recall
- Support – number of true instances of each class

```
3]:
```

	n_estimators	max_depth	accuracy
2	100	20.0	0.998465
4	300	NaN	0.998439
3	200	20.0	0.998386
0	50	10.0	0.992961
1	100	10.0	0.992670

Figure 4.9: Trained the model

This table summarizes the accuracy achieved by the Random Forest model for different combinations of hyperparameters:

- n_estimators — number of trees in the forest
- max_depth — maximum depth of each tree
- accuracy — overall test accuracy achieved

	precision	recall	f1-score	support
back	1.00	1.00	1.00	287
buffer_overflow	1.00	0.67	0.80	9
ftp_write	0.00	0.00	0.00	2
guess_passwd	1.00	0.88	0.93	16
imap	1.00	0.67	0.80	3
ipsweep	0.99	1.00	0.99	1080
land	0.60	0.60	0.60	5
loadmodule	1.00	0.33	0.50	3
multihop	0.00	0.00	0.00	2
neptune	1.00	1.00	1.00	12364
nmap	0.98	0.99	0.99	448
normal	1.00	1.00	1.00	20203
perl	0.00	0.00	0.00	1
phf	0.00	0.00	0.00	1
pod	0.98	0.98	0.98	60
portsweep	1.00	1.00	1.00	879
rootkit	0.00	0.00	0.00	3
satn	1.00	0.99	1.00	1090
smurf	1.00	1.00	1.00	794
spy	0.00	0.00	0.00	1
teardrop	1.00	1.00	1.00	268
warezclient	0.99	0.97	0.98	267
warezmaster	1.00	0.83	0.91	6
accuracy			1.00	37792
macro avg	0.72	0.65	0.67	37792
weighted avg	1.00	1.00	1.00	37792

Figure 4.10: Classification report

The classification report for the final Random Forest model that was chosen is displayed in this figure.

It shows:

- The model exhibits remarkably high F1 scores, recall, and precision in every class.
- The accuracy at the bottom is around 0.997.
- Macro avg and , weighted avg, also show near-perfect performance.

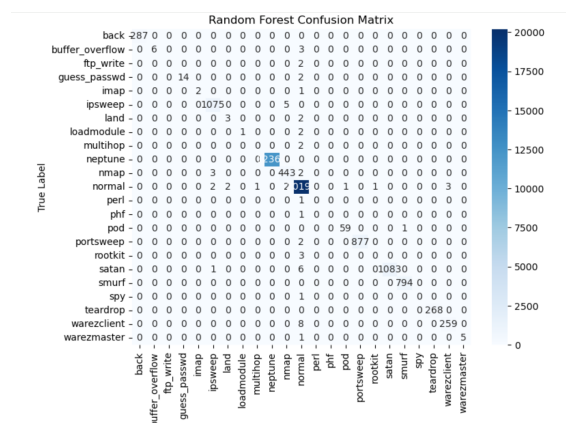


Figure 4.11: Random Forest Confusion Matrix

The confusion matrix shows how well each assault type was predicted by the Random Forest classifier in relation to the actual labels.

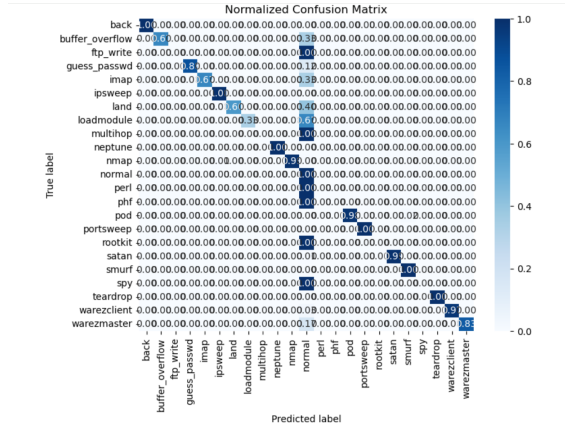


Figure 4.12: Normalized Confusion Matrix

This figure shows how well each attack/traffic type was recognized by your model, and it indicates that:

- Most attack types are identified correctly (strong diagonal).
- A few categories are easily confused with others.
- Normalization ensures each row sums to 1, so values represent percentages.

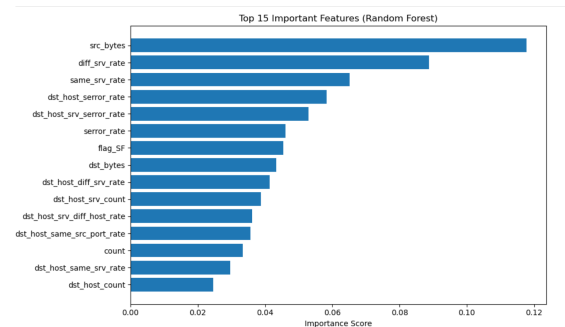


Figure 4.13: Random Forest Feature Importances

The Random Forest classifier's top 15 most significant attributes are displayed in the figure. Each feature's contribution to the model's capacity for prediction is reflected in its feature significance.

4.2 Gradient Boosting

	precision	recall	f1-score	support
back	0.99	0.99	0.99	287
buffer_overflow	0.17	0.22	0.19	9
ftp_write	0.00	0.00	0.00	2
guess_passwd	0.00	0.00	0.00	16
imap	1.00	1.00	1.00	3
ipsweep	0.96	0.98	0.97	1080
land	0.33	0.40	0.36	5
loadmodule	0.12	0.33	0.18	3
multihop	0.00	0.00	0.00	2
neptune	1.00	0.96	0.98	12364
nmap	0.99	0.99	0.99	448
normal	0.98	0.99	0.98	20203
perl	0.00	0.00	0.00	1
phf	0.50	1.00	0.67	1
pod	0.97	0.98	0.98	60
portsweep	1.00	0.99	0.99	879
rootkit	0.00	0.00	0.00	3
satan	0.80	0.98	0.88	1090
smurf	1.00	1.00	1.00	794
spy	0.00	0.00	0.00	1
teardrop	1.00	1.00	1.00	268
warezclient	0.99	0.76	0.86	267
warezmaster	0.05	0.67	0.10	6
accuracy			0.98	37792
macro avg	0.56	0.62	0.57	37792
weighted avg	0.98	0.98	0.98	37792

Figure 4.14: Gradient Boosting classification report

The Gradient Boosting classifier's performance for each class is compiled in the classification report. For every attack category, it offers precision, recall, F1-score, and support.

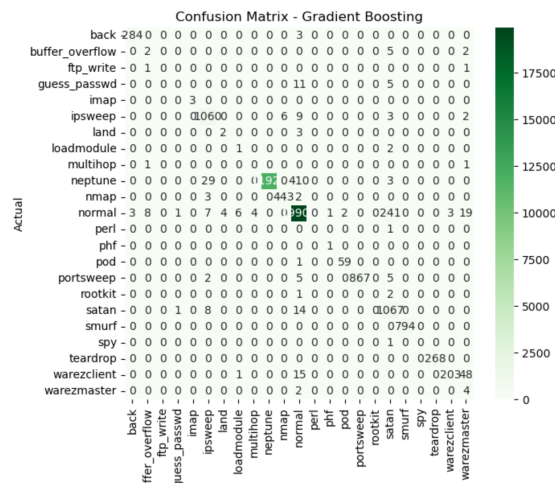


Figure 4.15: Gradient Boosting Confusion Matrix

The Gradient Boosting classifier's predictions are shown in depth in the confusion matrix.

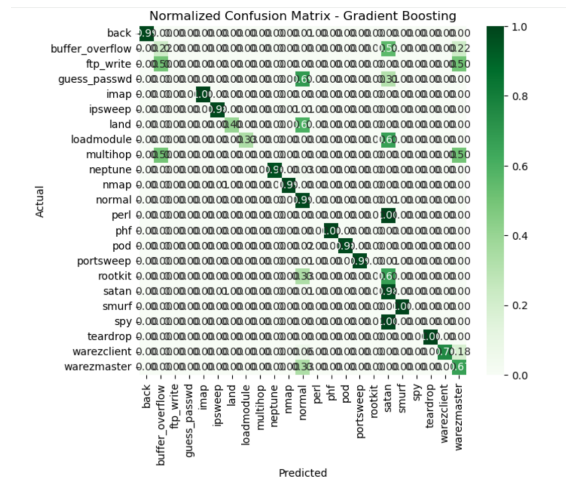


Figure 4.16: Gradient Boosting Normalized Confusion Matrix

This is a raw count of accurate versus inaccurate forecasts. It facilitates the visual comparison of class-level performance by displaying the percentage of accurate predictions, class by class. Strong classifier performance is indicated by the diagonal dominance. Where the model mistakes one class with another is indicated by off-diagonal numbers.

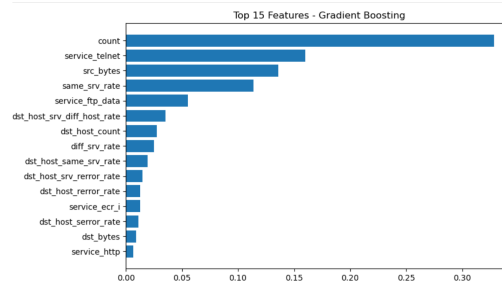


Figure 4.17: Feature Importances

The top 15 features that the Gradient Boosting classifier uses to make predictions are shown in this picture. Each feature's contribution to lowering prediction error during training is indicated by its feature significance.

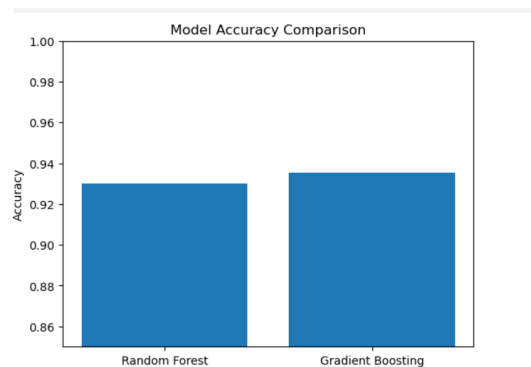


Figure 4.18: Model Accuracy Comparison

compares the Random Forest and Gradient Boosting classifiers' model accuracies. Both models performed well on the dataset, with high accuracy ranging from 92% to 94%. Gradient Boosting's capacity to better capture underlying patterns through its successive boosting mechanism is demonstrated by its slightly superior accuracy compared to Random Forest. Gradient Boosting offers a slight increase in predictive capability despite the small performance difference, making it a solid contender for the final model.

4.3 Discussion

The study describes the creation and assessment of two ensemble machine learning models for multiclass network intrusion detection using the NSL-KDD dataset: Random Forest (RF) and Gradient Boosting (GB). When compared to conventional signature-based approaches, ensemble methods have demonstrated great performance and adaptability in Intrusion Detection System (IDS) tasks, which is why they were selected.

4.3.1 Random Forest (RF) Classifier

The Random Forest model exhibited strong overall performance with high accuracy and robust handling of noisy data. Key Results:

- Achieved comparable accuracy to Gradient Boosting.
- Demonstrates strong generalization due to its ensemble averaging approach (bagging)
- Showed excellent performance for majority classes like Normal traffic and DoS (Denial of Service) attacks.
- Is robust to noise due to bootstrap sampling.

4.3.2 Key Limitations (Overfitting and Imbalance):

- The model displayed near-perfect training accuracy, which is considered a sign of overfitting and is noted as a normal behavior for this model on NSL-KDD data.
- It struggled with minority attack categories such as U2R (User-to-Root) and R2L (Remote-to-Local), consistent with the dataset imbalance challenge. Feature Importance: The Random Forest model's most significant characteristics were associated with connection statistics, particularly

4.4 Gradient Boosting Classifier

The Gradient Boosting model performed competitively, showing a slight edge in overall predictive capability. Key Results:

- Achieved a slightly higher accuracy (typically between 92% and 94%) compared to Random Forest, indicating a better ability to capture underlying patterns through its sequential boosting mechanism.

- Enhanced performance and sensitivity on the minority attack classes (R2L and U2R), which frequently provide a problem for intrusion detection.
- Its sequential learning approach effectively reduces bias.

4.4.1 Key Limitations:

- The model had a longer training time
- It was more sensitive to parameter adjustments.
- Showed slightly lower performance on high-volume attack classes compared to RandomForest.

5 Conclusion

Using the NSL-KDD dataset, this paper provides a thorough assessment of ensemble machine learning models for multiclass intrusion detection. While both Random Forest and Gradient Boosting produced excellent detection results, Gradient Boosting is a good contender for the final model because it offers a slight increase in predicting power.

6 REFERENCES

- [1] M. Tavallaei, E. Bagheri, W. Lu and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009.
- [2] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," ICISSP 2018, 2018.
- [3] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
- [4] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," Annals of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001.
- [5] S. Revathi and A. Malathi, "A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques," IJERT, 2013.
- [6] T. Tang et al., "Deep Learning Approach for Network Intrusion Detection in SDN," WINCOM, 2016.
- [7] C. Song, K. Xia, and J. Shen, "A Survey of Machine Learning in Network Security," IEEE Access, 2020.