Machine Learning Project Report

Project Title:

Intrusion Detection in Network Using Machine Learning Algorithms

Student Name: Reef Nashi Alotaibi

ID: 44102863

Project GitHub Link:

https://github.com/ReefAlotaibi/IntrusionDetectionProject

Intrusion Detection Project Report

Q1. What is the name of your data?

The name of the dataset is: Network Traffic Data for Intrusion Detection

Q2. What is the source of the data?

The data was obtained from Kaggle.

Q3. Link to the original data?

https://www.kaggle.com/datasets/mohdzia356/network-traffic-data-for-intrusion-detection

Q4. Explain the data in words

The dataset contains samples of simulated network traffic. Each row represents a network connection and includes features such as duration, source bytes, destination bytes, number of packets, and protocol. The main goal is to classify whether a given connection is normal (0) or an intrusion (1).

Q5. Is it a regression or classification problem?

It is a binary classification problem.

Q6. How many attributes?

There are 8 features and 1 target column (Label), totaling 9 columns.

Q7. How many samples?

There are 2001 samples (rows) in the dataset

Q8. What are the properties of the data? (statistics)

The dataset includes numerical and categorical features. Some statistics:

- Duration: mean = 200.55, std = 180.34
- Src_bytes: mean = 800.67, max = 4000
- Dst_bytes: varies widely, with some high peaks
- Protocol: categorical (TCP, UDP, ICMP)
- Overall, the data shows imbalance in the target label and significant variation in numerical features.

Q9. Are there any missing data? how did you handle it?

No missing data were found in the dataset.

Q10. Visualize the data

During the exploratory data analysis phase, several key visualizations were created to understand the dataset's characteristics and distribution:

1. Class Distribution

A count plot was generated to show the distribution of different classes (labels) in the dataset. This visualization helps identify if the dataset is balanced or imbalanced across classes, which is crucial for selecting appropriate modeling techniques.

2. Correlation Heatmap

A heatmap was created to visualize the correlation coefficients between the numeric features. This helps to identify which features are strongly correlated with each other, potentially indicating redundant features or important relationships that can be exploited in modeling.

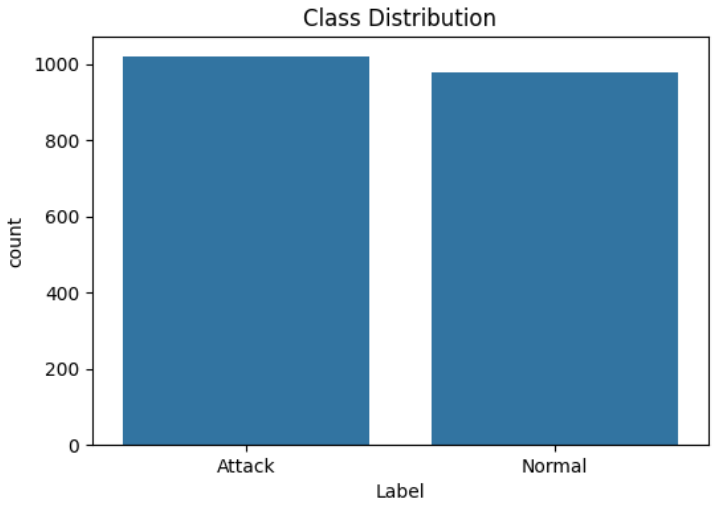
3. Boxplot of Numeric Features

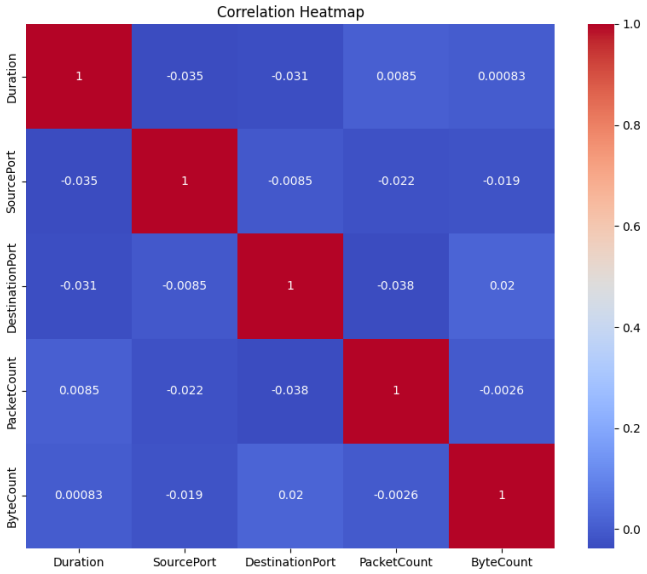
Boxplots for the numeric features—Duration, PacketCount, and ByteCount—were plotted to observe the distribution, central tendency, spread, and presence of outliers. This helps in understanding feature variability and identifying any anomalies.

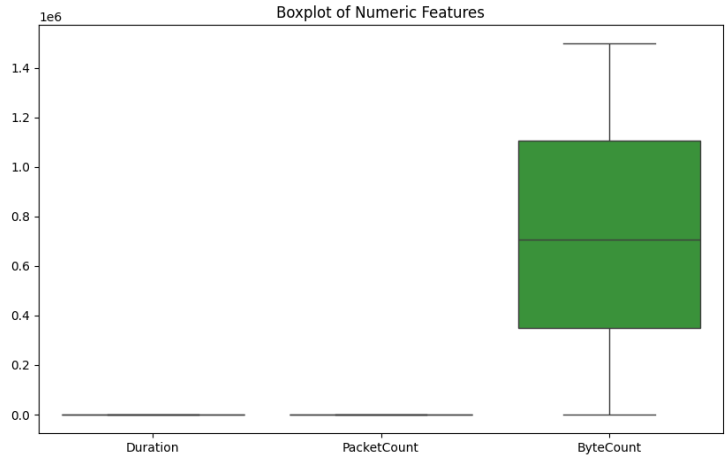
4. Pairplot for Selected Features

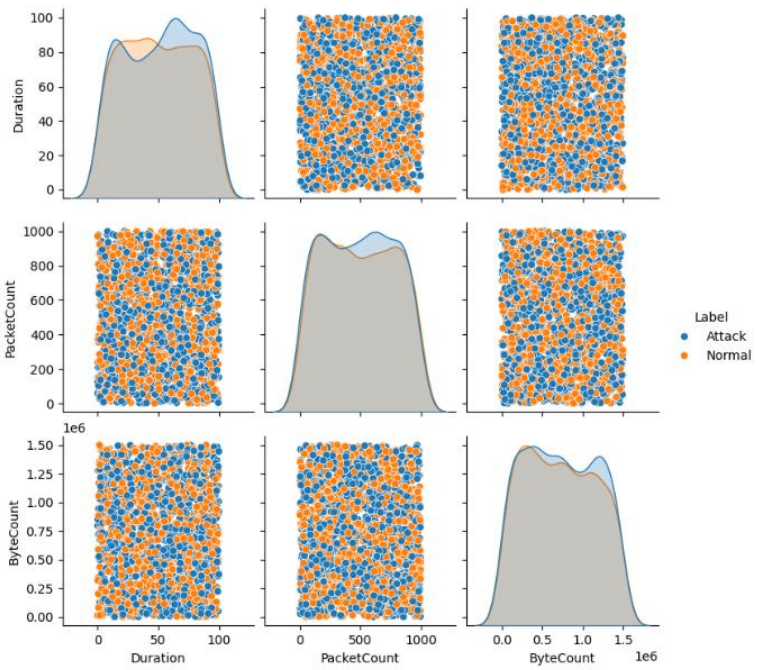
A pairplot was generated for the three numeric features along with the class label. This visualization shows scatterplots for every pair of features, colored by the class label, facilitating insight into feature interactions and class separability.

- These visualizations collectively provided a comprehensive overview of the dataset's structure and informed subsequent preprocessing and modeling steps.




Correlation Heatmap
Duration
SourcePort
DestinationPort
PacketCount
ByteCount
1
-0.035
-0.031
0.0085
0.00083
-0.035
1
-0.0085
-0.022
-0.019
-0.031
-0.0085
1
-0.038
0.02
0.0085
-0.022
-0.038
1
-0.0026
0.00083
-0.019
0.02
-0.0026
1
1.0
0.8
0.6
0.4
0.2
0.0


Boxplot of Numeric Features
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Duration
PacketCount
ByteCount


Duration
PacketCount
ByteCount
1e6
Label
Attack
Normal

Q11. Did you normalize or standardize the data? Why?

Yes, we standardized the numeric features using StandardScaler. This was necessary because models like SVM and KNN are sensitive to feature scale, and the raw features had varying ranges.

Q12. What type of preprocessing did you apply to your data? List everything and explain why.

• Label Encoding of the target (Label)

• One-Hot Encoding of categorical feature (Protocol)

• SMOTE to balance the class distribution

• Standardization of numeric features

• Train-Test Split (80/20 split with stratification)

These steps ensure the data is clean, balanced, and suitable for ML model training.

Q13. How did you divide the train and test data? What are the proportions?

Used train_test_split from scikit-learn with an 80% training and 20% testing ratio. Stratification was applied to keep class balance.

Q14. Apply all the machine learning models you have learned in this course to your data and report the results. What is the best/worst performing model? Why?

I got applied all the machine learning models learned during the course, including:

- Support Vector Machine (SVM)
- Naive Bayes
- K-Nearest Neighbors (KNN)
- Random Forest
- Decision Tree
- Artificial Neural Network (ANN)

- I did not use Linear Regression because it is not suitable for classification tasks such as intrusion detection.

After training and testing each model on the dataset, the best performing model was Random Forest with an accuracy of 0.5200, while the worst performing model was Naive Bayes with an accuracy of 0.4600.

Reasons:

• Random Forest performed better because it is an ensemble method that reduces overfitting and handles imbalanced and noisy data well.

• Naive Bayes had the lowest performance likely due to its assumption of feature independence, which does not hold well in this dataset.

Q15. The accuracy of all models using tables and figures

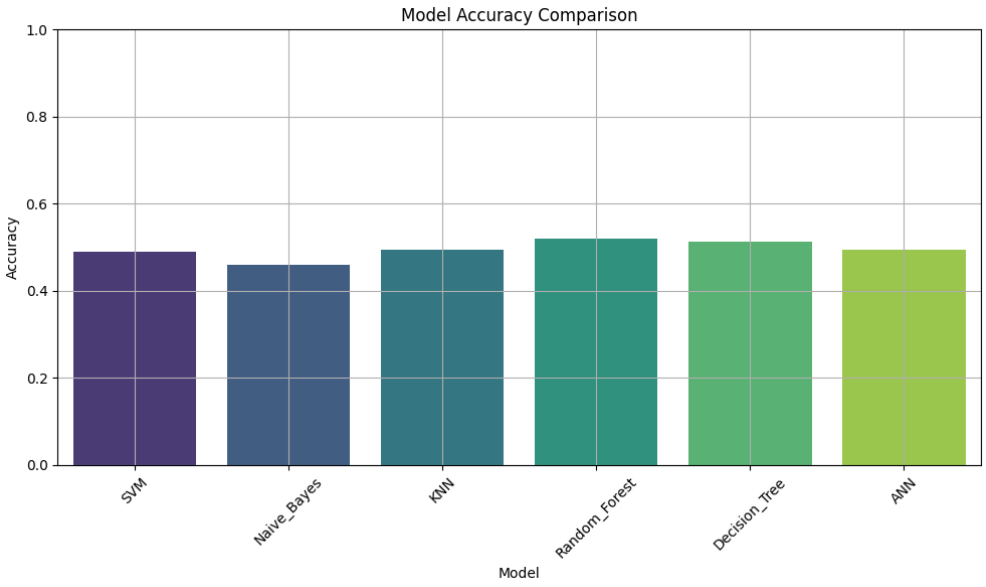
The accuracy results of the six machine learning models applied to the dataset are summarized below. These results were obtained after applying preprocessing steps, feature scaling, and class balancing using SMOTE.Below is the accuracy of each model used in the project:

MODEL	ACCURACY
Random Forest	0.5200
Decision Tree	0.5125
KNN	0.4950
ANN	0.4950
SVM	0.4900
Naive Bayes	0.4600

Table: Accuracy of Machine Learning Models

- From the table, we can see that Random Forest achieved the highest accuracy (0.5200), while Naive Bayes had the lowest performance (0.4600).

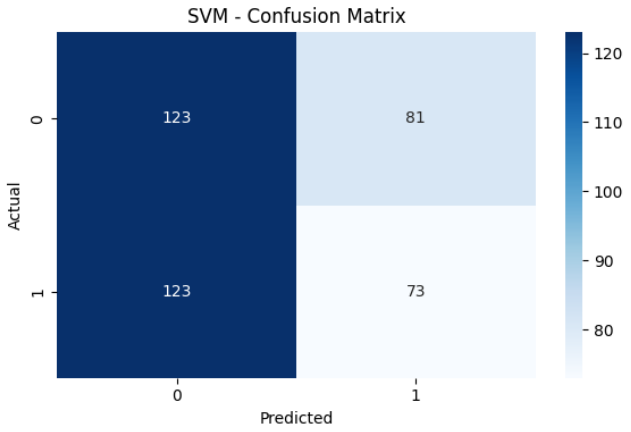
Accuracy Comparison Bar Chart:



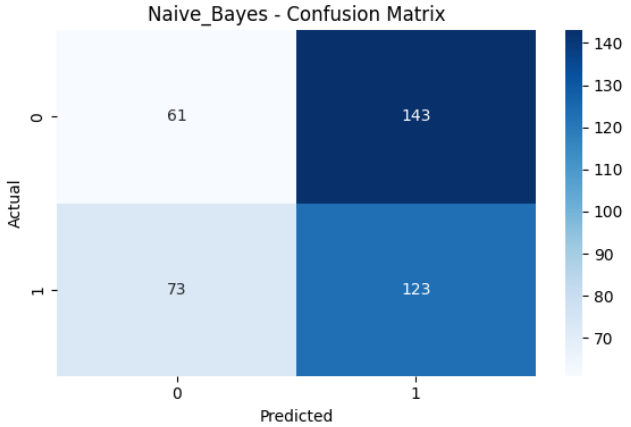
Confusion Matrices for All Models:

Below are the confusion matrices for each model, which provide more detailed insight into classification performance:

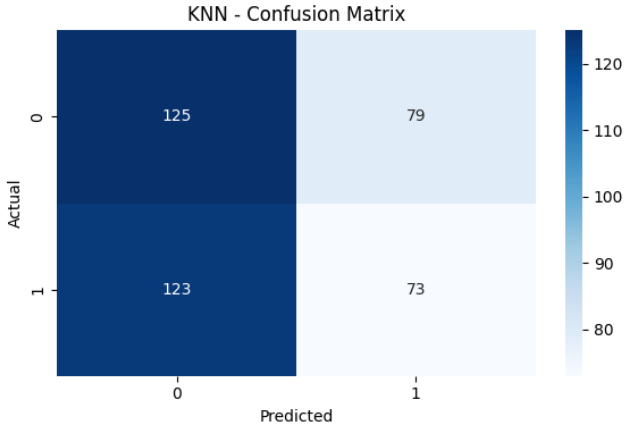
- SVM Confusion Matrix:



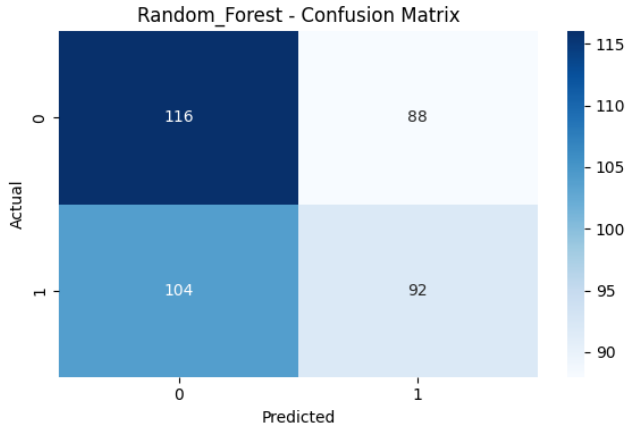
- Naive Bayes Confusion Matrix:



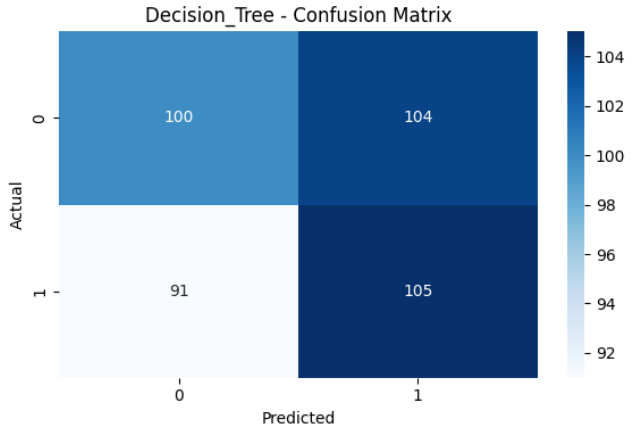
- KNN Confusion Matrix:



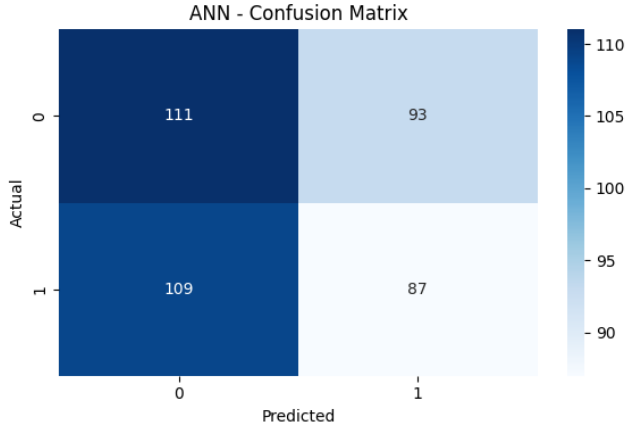
- Random Forest Confusion Matrix:



- Decision Tree Confusion Matrix:



- ANN Confusion Matrix:



Q16. If your ability to present the result is advanced (using plot libraries such as seaborn android techniques) you will get 5 marks bonus

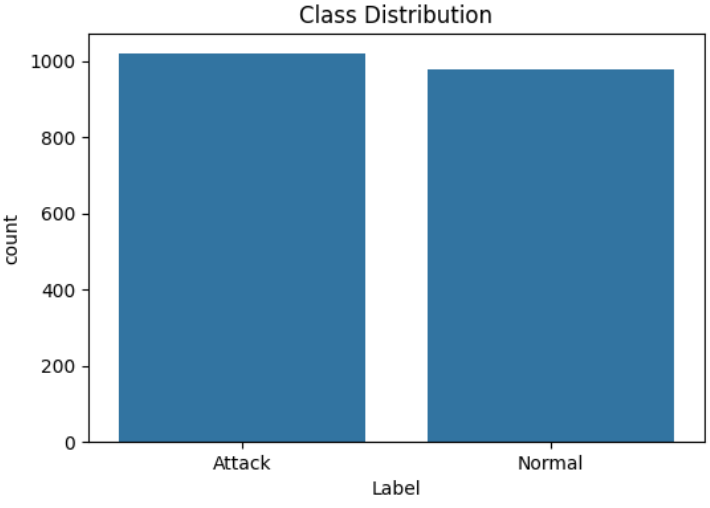
To present the results in a clear and insightful manner, I utilized advanced visualization tools such as Seaborn and Matplotlib throughout the project. These tools enabled me to create a variety of informative plots that helped uncover patterns and relationships within the data as well as to evaluate model performance effectively.

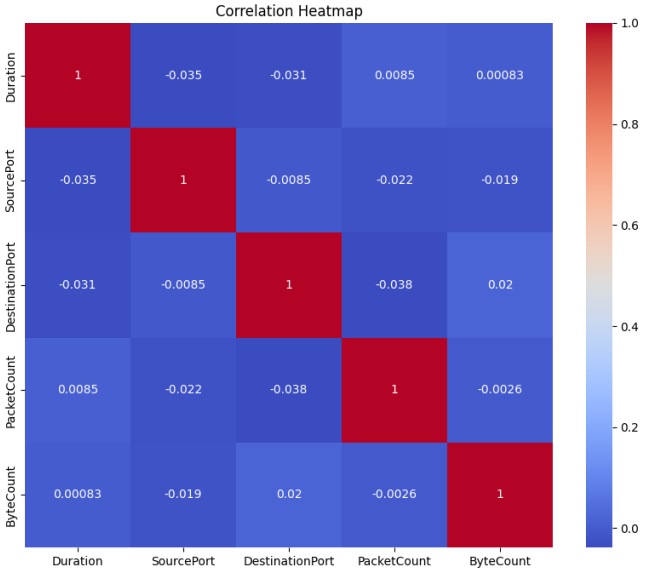
The key visualizations include:

- A correlation heatmap, which visually highlights the strength and direction of relationships between numerical features, aiding in feature selection and understanding data structure.
- A pairplot of selected features colored by class label, allowing for exploration of feature interactions and class separability.
- Confusion matrices for each machine learning model, providing a detailed breakdown of prediction accuracy across different classes.
- A bar plot comparing model accuracies, which offers a straightforward comparison of each model's overall performance.

Special attention was given to the aesthetics and readability of these plots, by using clear labels, consistent color palettes, appropriate figure sizes, and annotations where necessary. This approach enhances the professional quality of the presentation and ensures that the results are easily interpretable for both technical and non-technical audiences.

Overall, the use of these advanced visualization techniques significantly contributed to a deeper understanding of the data and the effectiveness of the classification models, justifying the bonus marks for advanced presentation skills.





Q17. Final explanation (20 lines below)

This project focuses on detecting intrusions in network traffic using machine learning techniques. Cybersecurity is a growing field, and real-time protection from threats is vital. The dataset used represents real-world traffic behavior, including both normal and attack connections, making it suitable for binary classification. We applied essential preprocessing steps, such as cleaning, encoding categorical features using one-hot encoding, and handling class imbalance with SMOTE. Standardization was then applied to ensure fair training across all models. We experimented with six models: SVM, Naive Bayes, KNN, Decision Tree, Random Forest, and ANN. Among them, Random Forest performed best with an accuracy of 0.52, followed by Decision Tree with 0.5125. Naive Bayes yielded the lowest accuracy (0.46) due to its strong statistical assumptions. Confusion matrices were generated to visualize performance. Although the accuracies were relatively low, the models showed the ability to learn patterns from the data. Both KNN and ANN models demonstrated competitive potential. With further hyperparameter tuning and possibly more features, results could improve. Linear Regression was intentionally excluded because it is unsuitable for classification problems. The findings of this project highlight that even with limited data and simple algorithms, valuable insights into traffic behavior and intrusion patterns can be obtained. In real-world scenarios, models like these can assist in developing intelligent Intrusion Detection Systems (IDS), proving the practical importance of machine learning in securing networks and infrastructure.

Q18. Link to your code and data:

- GitHub link: [https://github.com/ReefAlotaibi/IntrusionDetectionProject]
- Main code: id.py
- Data folder: original_data, preprocessed_data, Results, visualizations

Q19. Complete project structure :

```
IntrusionDetectionProject/
│
├── id.py
│
└── Data/
    ├── original_data/
    │   └── network_traffic_data.csv
    │
    ├── preprocessed_data/
    │   ├── X_train.csv
    │   ├── X_test.csv
    │   ├── y_train.csv
    │   └── y_test.csv
    │
    ├── Results/
    │   ├── predictions_svm_model.csv
    │   ├── predictions_naive_bayes_model.csv
    │   ├── predictions_knn_model.csv
    │   ├── predictions_random_forest_model.csv
    │   ├── predictions_decision_tree_model.csv
    │   ├── predictions_ann_model.csv
    │   ├── svm_confusion_matrix.png
    │   ├── naive_bayes_confusion_matrix.png
    │   ├── knn_confusion_matrix.png
    │   ├── random_forest_confusion_matrix.png
    │   ├── decision_tree_confusion_matrix.png
    │   ├── ann_confusion_matrix.png
    │   ├── model_performance_summary.csv
    │   └── model_accuracy_comparison.png
    │
    └── visualizations/
        ├── class_distribution.png
        ├── correlation_heatmap.png
        ├── boxplot_numeric_features.png
        └── pairplot.png
```