Applied Machine Learning Model Comparison

Samantha Riggs

Old Dominion University

CYSE 420: Applied Machine Learning

Professor Wael Khallouli

December 9th, 2025

Project Overview:

Training and evaluating models on intrusion datasets like the one utilized in this project is integral to predicting and reducing breaches within systems and networks. Being able to evaluate the performance of applied machine learning models that are trained on a dataset can help determine their specific strengths and weaknesses and also help determine which models would perform the best if implemented in a real-world scenario.

For this project, an intrusion detection dataset was chosen. This dataset consisted of 9,537 samples and 10 features plus the target. The features included numeric network and session statistics: network packet size, login attempts, session duration, IP reputation score, failed logins, and unusual time access. The categorical features consisted of protocol type, encryption used, and browser type. Binary classification is used to detect attack (attack detected = 1) versus normal traffic (0). The objective of this project was to implement four models with the addition of ensemble methods and compare performance using Accuracy and F1 score as primary metrics. Precision, Recall, and Confusion Matrices were also used as secondary metrics in order to evaluate model performance in more detail.

Methodology:

The dataset was cleaned up in order to handle missing values using Simple Imputer as well as dropping invalid rows. Features were then separated into numerical and categorical features. The numerical features were scaled with StandardScaler and the categorical features were encoded using OneHotEncoder. Finally, an 80/20 stratified split was used for the train/test split in order to ensure a fair evaluation.

Four machine learning models were implemented in this project. Firstly, a Logistic Regression model was included as it provides a simple, interpretable baseline. Second, a Support Vector Machine, or SVM, model was implemented as it performs well with scaled data. Third, a Random Forest Classifier model was chosen due to how robust it is to noise and outliers in the case they are present. Finally, an MLP Classifier Neural Network as it is able to capture nonlinear relationships well.

Three ensemble learning approaches were added as well. Bagging was added to help reduce variance and work to increase stability. AdaBoost was added since it performs well on tabular data. Finally, a stacking ensemble with Logistic Regression as the meta-learner was implemented as this ensemble leverages diverse model strengths and works to mitigate weaknesses.

Each model in this project was trained using the same preprocessed training data via pipelines as well as default hyperparameters. Pipelines were chosen to ensure consistent preprocessing and work to prevent data leakage. Each model used the following metrics: Accuracy, Precision, Recall, F1 Score, and Confusion Matrices. Accuracy and F1 are notable primary metrics as F1 help balance both false positives and negatives. This metric is particularly useful for cybersecurity since attacks can be caught without excessive false alarms. In order to clearly visualize the results and comparisons using a table summarizing all of the metrics, a horizontal bar chart comparing accuracy and F1 across all four models, and the identification of the best performing models.

Results and Analysis:

The following tables detail the results for each model:

```
==LogisticRegression ==
Accuracy: 0.7222222222222222
Precision: 0.6828992072480181
Recall: 0.7069167643610785
F1: 0.6947004608294931

              precision    recall  f1-score   support

           0     0.7561    0.7346    0.7452      1055
           1     0.6829    0.7069    0.6947       853

    accuracy                         0.7222      1908
   macro avg     0.7195    0.7208    0.7199      1908
weighted avg     0.7234    0.7222    0.7226      1908
```

```
==SVM ==
Accuracy: 0.8736897274633124
Precision: 0.9707692307692307
Recall: 0.7397420867526378
F1: 0.8396540252827678

              precision    recall  f1-score   support

           0     0.8235    0.9820    0.8958      1055
           1     0.9708    0.7397    0.8397       853

    accuracy                         0.8737      1908
   macro avg     0.8971    0.8609    0.8677      1908
weighted avg     0.8894    0.8737    0.8707      1908
```

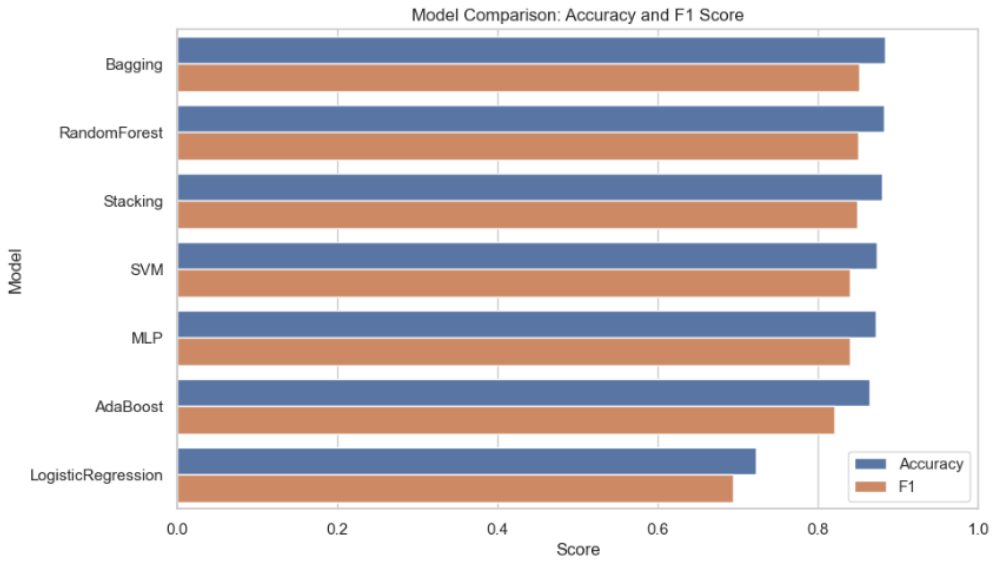
```
==RandomForest ==
Accuracy: 0.8825995807127882
Precision: 0.9906396255850234
Recall: 0.7444314185228605
F1: 0.8500669344042838

              precision    recall  f1-score   support

           0     0.8279    0.9943    0.9035      1055
           1     0.9906    0.7444    0.8501       853

    accuracy                         0.8826      1908
   macro avg     0.9093    0.8694    0.8768      1908
weighted avg     0.9007    0.8826    0.8796      1908
```

```
==MLP ==
Accuracy: 0.8726415094339622
Precision: 0.9621212121212122
Recall: 0.7444314185228605
F1: 0.8393919365499009

              precision    recall  f1-score   support

           0     0.8253    0.9763    0.8945      1055
           1     0.9621    0.7444    0.8394       853

    accuracy                         0.8726      1908
   macro avg     0.8937    0.8604    0.8669      1908
weighted avg     0.8865    0.8726    0.8699      1908


 ==Bagging ==
 Accuracy: 0.8841719077568134
 Precision: 0.9922118380062306
 Recall: 0.7467760844079718
 F1: 0.8521739130434782

               precision    recall  f1-score   support

           0     0.8294    0.9953    0.9048      1055
           1     0.9922    0.7468    0.8522       853

    accuracy                         0.8842      1908
   macro avg     0.9108    0.8710    0.8785      1908
weighted avg     0.9022    0.8842    0.8813      1908


 ==AdaBoost ==
 Accuracy: 0.8642557651991615
 Precision: 1.0
 Recall: 0.6963657678780774
 F1: 0.821008984105045

               precision    recall  f1-score   support

           0     0.8029    1.0000    0.8907      1055
           1     1.0000    0.6964    0.8210       853

    accuracy                         0.8643      1908
   macro avg     0.9014    0.8482    0.8558      1908
weighted avg     0.8910    0.8643    0.8595      1908
```

```
==Stacking ==
Accuracy: 0.8805031446540881
Precision: 0.9785604900459418
Recall: 0.7491207502930832
F1: 0.848605577689243

              precision    recall  f1-score   support

           0     0.8295    0.9867    0.9013      1055
           1     0.9786    0.7491    0.8486       853

    accuracy                         0.8805      1908
   macro avg     0.9040    0.8679    0.8750      1908
weighted avg     0.8961    0.8805    0.8777      1908
```

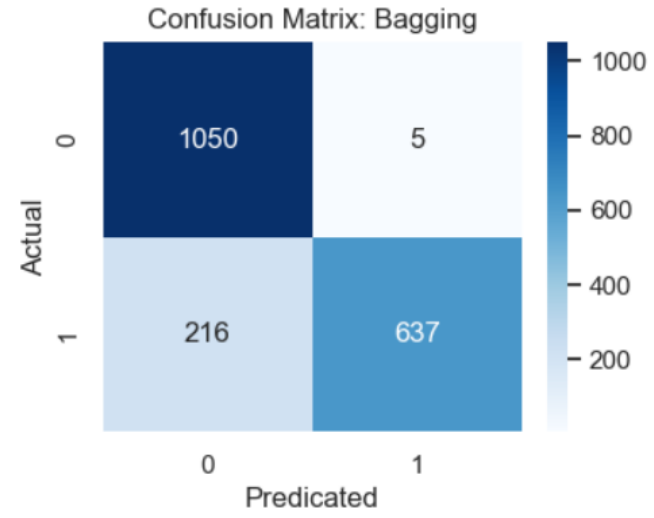
The Random Forest model achieved the highest F1 with 0.850 in addition to having high Precision (~0.991) and Accuracy (~0.883), with the lowest value for the model being Recall (~0.744). This result indicates that the model balances false positives and false negatives better in comparison to the other. It is also notable that the AdaBoost ensemble had a precision of 1.000, indicating that the model had zero false positives. However, the Accuracy and F1 Score, which are both primary metrics, are lower than other models.

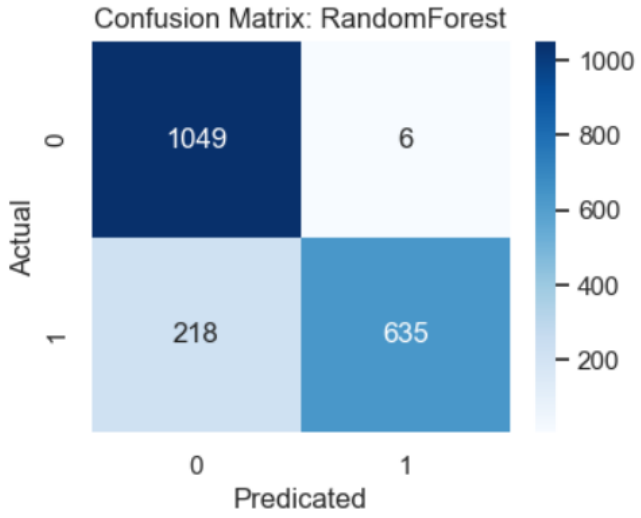
The performance of the models, along with the ensembles, is more clearly visualized using a horizontal bar chart, which presents a comparison using the two primary metrics:

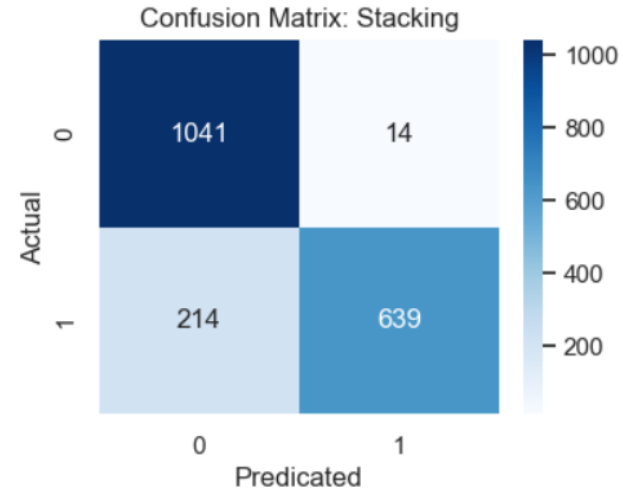


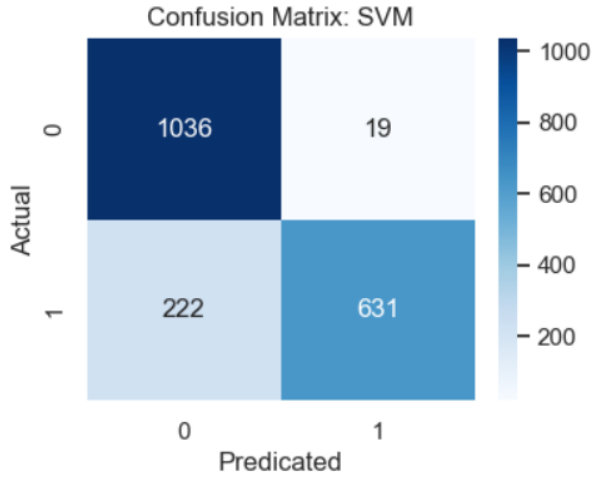
Using this visualization, all of the trained models are relatively similar in performance, with the Logistic Regression model being an exception since it has a noticeably lower number for both Accuracy and F1. In the dataset, intrusion detection patterns are highly nonlinear, which is a weakness of the Logistic Regression model, resulting in poor performance.

The four top performing models had confusion matrices created to visualize false positives and false negatives from each model:









The produced Confusion Matrices indicate that the Bagging ensemble was the highest performing model was the Bagging ensemble, followed closely by Random Forest, Stacking, and SVM. Bagging achieved the strongest accuracy and F1 score because it utilizes many decision tree models trained on different bootstrap samples. This means that variance is reduced and robustness is improved. Intrusion detection datasets often contain noisy or irregular patterns, and Bagging is particularly effective at addressing those inconsistencies.

Conclusion:

Overall, the results indicate that ensemble methods, particularly Bagging and Random Forest, are the most effective approaches for the specific intrusion detection dataset used in this project. They are able to capture nonlinear relationships while remaining resistant to noise and overfitting. Stacking also performed strongly, demonstrating that combining diverse models can enhance predictiveness. SVM proved to also perform well, confirming the presence of nonlinear patterns in the feature space. These results suggest that future intrusion detection systems would benefit from tree-based ensemble methods or hybrid stacked architectures to achieve high detection rates and low error rates.