Department of Electrical and Computer Engineering
University of Massachusetts Dartmouth
ECE 489/549 Network Security Prof. Hong Liu

Project Progress Report

Group Coordinator: Jeevan Kumar Banoth
Members: Pulkit Kalra, Samuel D Guillaume, Namratha Sumeda

A. Is there a need to modify the Work Plan in your proposal?

Yes, there is a need to modify the Work Plan in the original proposal. The initial plan was based on using NS-3 or Cooja for simulation, but our implementation has focused on using the NSL-KDD dataset and Python-based machine learning techniques. This shift in approach necessitates changes to the Work Plan. Here's an elaboration on why and how the Work Plan should be modified:

a) Change in simulation approach: The original plan included tasks for setting up NS-3 or Cooja simulations. Since we're now using a pre-existing dataset (NSL-KDD), these tasks are no longer relevant and should be replaced with data preprocessing and feature engineering tasks.

b) Focus on machine learning: The current implementation heavily emphasizes machine learning techniques, which weren't explicitly mentioned in the original Work Plan. We need to add tasks related to model selection, training, and evaluation.

c) Real-world application considerations: While the original plan focused on simulation, our current approach is closer to real-world application. We should add tasks related to integrating our model with actual SHM WSN systems.

d) Timeline adjustment: Some tasks, like data preprocessing and feature engineering, have already been completed. The timeline needs to be adjusted to reflect the current state of the project and allocate time for remaining tasks.

B. **Show your work following the schedule and/or the modified Work Plan**

Based on the need for modification, here's a presentation of the work done so far and a proposed modified Work Plan:

Work Completed:

1. Data Acquisition and Preprocessing
   - Obtained NSL-KDD dataset
   - Cleaned and preprocessed data
   - Encoded categorical variables
   - Normalized numerical features

2. Feature Engineering
   - Created new features relevant to SHM WSN (e.g., bytes_ratio, is_long_connection)
   - Implemented rolling window statistics (count_100, srv_count_100)

3. Advanced Feature Selection
   - Applied Recursive Feature Elimination
   - Selected 20 most important features for attack detection

4. Handling Imbalanced Data

   - Implemented SMOTEENN technique. (It is a powerful technique of handling imbalanced dataset.)
   - Balanced the dataset for improved model training

Modified Work Plan for Remaining Tasks:

5. Model Development and Training

   - Implement ensemble model (Random Forest, Gradient Boosting, SVM)
   - Train models on balanced dataset
   - Implement cross-validation for robust evaluation

6. Model Evaluation and Optimization

   - Evaluate model performance using various metrics (accuracy, precision, recall, F1-score)
   - Fine-tune hyperparameters
   - Analyze feature importance and model interpretability

7. Real-world Application Design
   - Design integration framework for existing SHM systems
   - Develop real-time data processing pipeline
   - Create alert system for detected anomalies

8. Testing and Validation

- Conduct thorough testing with various attack scenarios
- Validate model performance against real-world SHM data (if available)
- Address any identified issues or limitations

9. Documentation and Reporting

- Prepare comprehensive project documentation
- Write final report detailing methodology, results, and conclusions
- Develop presentation materials for project demonstration

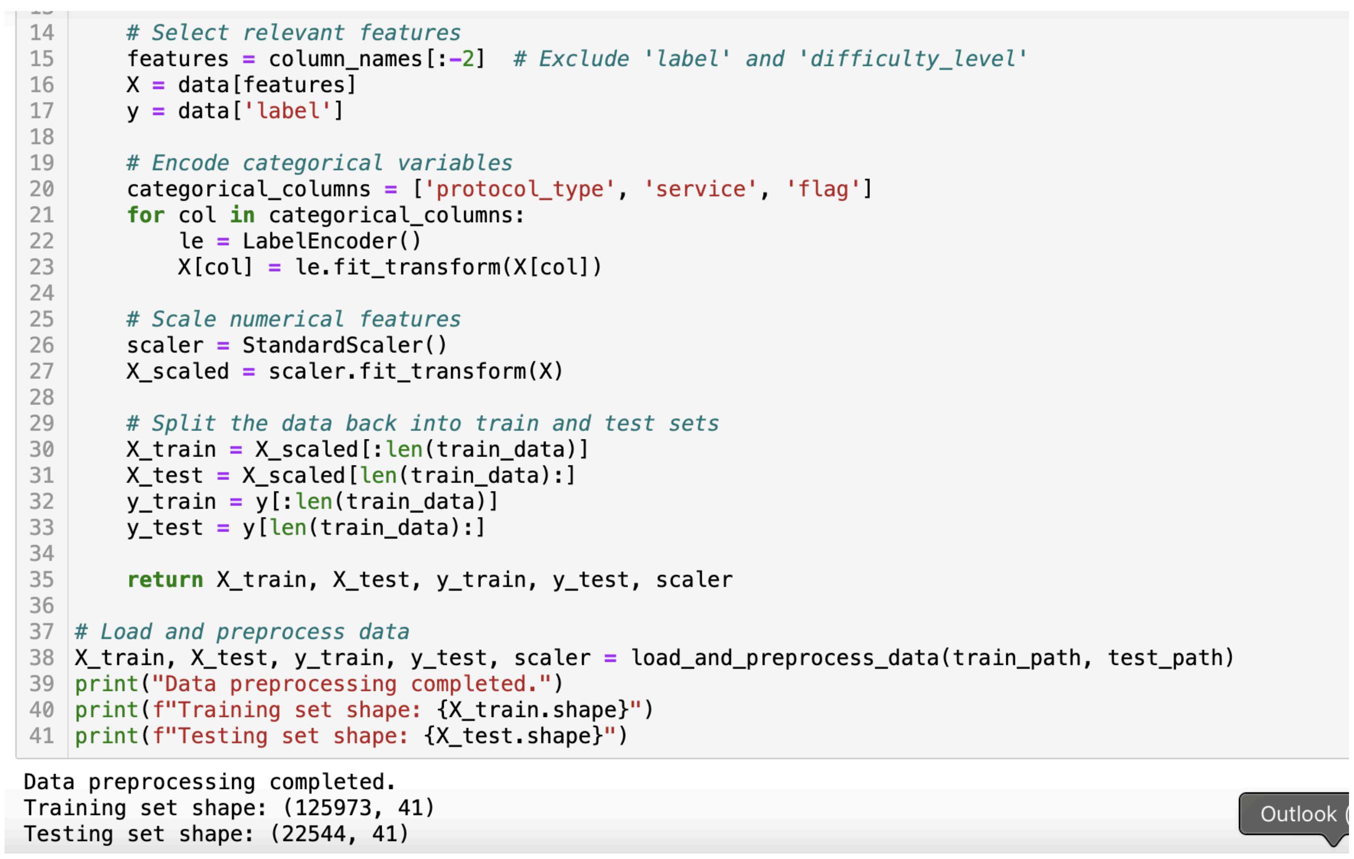
10. Future Work and Recommendations

- Outline potential improvements and future research directions
- Discuss potential for real-world deployment and challenges
- Prepare final presentation and project defense

This modified Work Plan reflects the actual progress made and aligns with the current project direction. It maintains the original timeline but redistributes the tasks to focus on the machine learning approach and **real-world application** considerations for SHM WSN attack detection.

<u>The work that we have done so far is:</u>

Data Preprocessing:

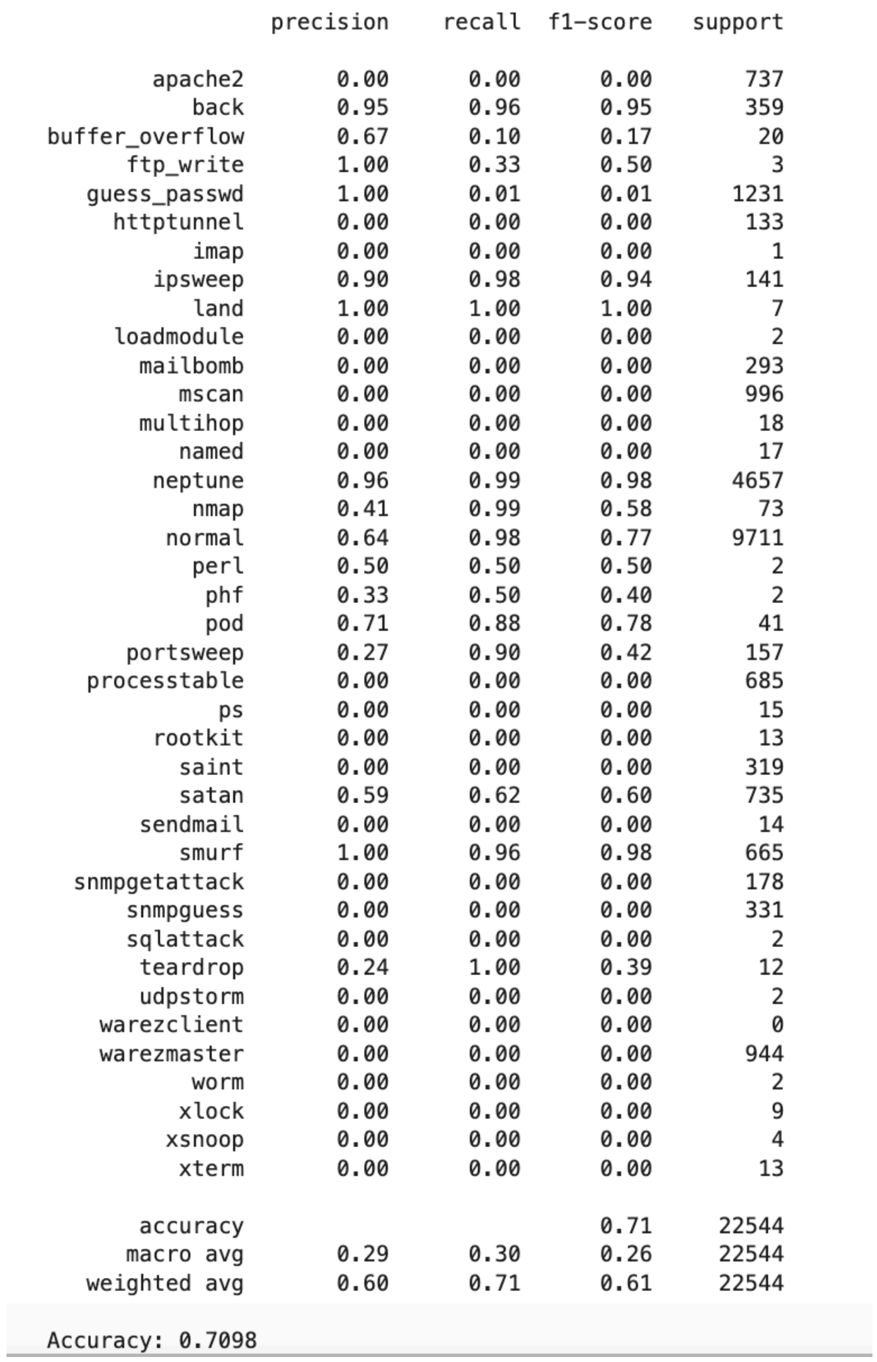


```
    # Select relevant features
    features = column_names[:-2]  # Exclude 'label' and 'difficulty_level'
    X = data[features]
    y = data['label']

    # Encode categorical variables
    categorical_columns = ['protocol_type', 'service', 'flag']
    for col in categorical_columns:
        le = LabelEncoder()
        X[col] = le.fit_transform(X[col])

    # Scale numerical features
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # Split the data back into train and test sets
    X_train = X_scaled[:len(train_data)]
    X_test = X_scaled[len(train_data):]
    y_train = y[:len(train_data)]
    y_test = y[len(train_data):]

    return X_train, X_test, y_train, y_test, scaler

# Load and preprocess data
X_train, X_test, y_train, y_test, scaler = load_and_preprocess_data(train_path, test_path)
print("Data preprocessing completed.")
print(f"Training set shape: {X_train.shape}")
print(f"Testing set shape: {X_test.shape}")
```

```
Data preprocessing completed.
Training set shape: (125973, 41)
Testing set shape: (22544, 41)
```

As you can see here, this is a very large dataset containing (125973, 41) values and the testing one is comparatively small. That is the reason why It took us so long to sample this dataset and perform the preprocessing.

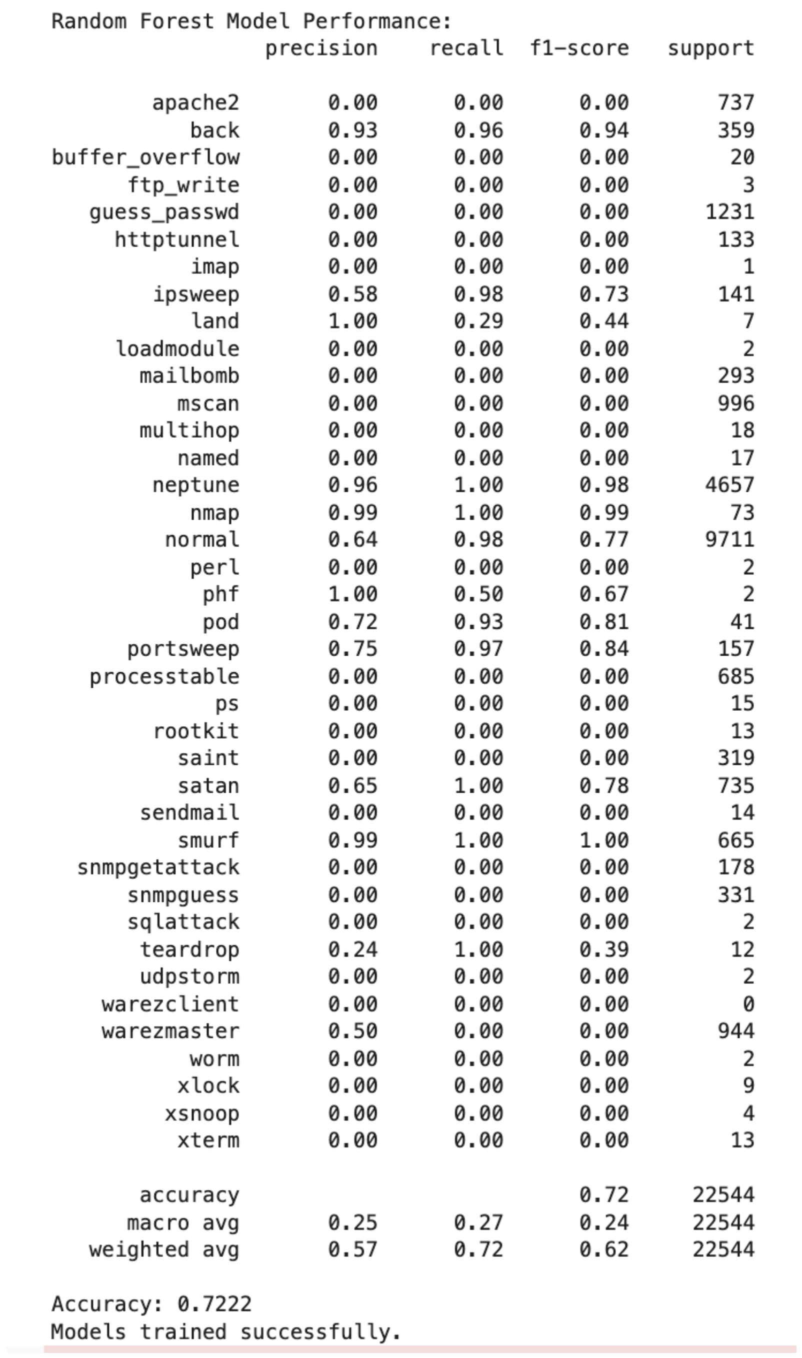
SVM Model Performance and accuracy:



```
                 precision    recall  f1-score   support

        apache2       0.00      0.00      0.00       737
           back       0.95      0.96      0.95       359
buffer_overflow       0.67      0.10      0.17        20
      ftp_write       1.00      0.33      0.50         3
   guess_passwd       1.00      0.01      0.01      1231
     httptunnel       0.00      0.00      0.00       133
           imap       0.00      0.00      0.00         1
        ipsweep       0.90      0.98      0.94       141
           land       1.00      1.00      1.00         7
     loadmodule       0.00      0.00      0.00         2
       mailbomb       0.00      0.00      0.00       293
          mscan       0.00      0.00      0.00       996
       multihop       0.00      0.00      0.00        18
          named       0.00      0.00      0.00        17
        neptune       0.96      0.99      0.98      4657
           nmap       0.41      0.99      0.58        73
         normal       0.64      0.98      0.77      9711
           perl       0.50      0.50      0.50         2
            phf       0.33      0.50      0.40         2
            pod       0.71      0.88      0.78        41
      portsweep       0.27      0.90      0.42       157
   processtable       0.00      0.00      0.00       685
             ps       0.00      0.00      0.00        15
        rootkit       0.00      0.00      0.00        13
          saint       0.00      0.00      0.00       319
          satan       0.59      0.62      0.60       735
       sendmail       0.00      0.00      0.00        14
          smurf       1.00      0.96      0.98       665
   snmpgetattack       0.00      0.00      0.00       178
      snmpguess       0.00      0.00      0.00       331
      sqlattack       0.00      0.00      0.00         2
       teardrop       0.24      1.00      0.39        12
       udpstorm       0.00      0.00      0.00         2
    warezclient       0.00      0.00      0.00         0
    warezmaster       0.00      0.00      0.00       944
           worm       0.00      0.00      0.00         2
          xlock       0.00      0.00      0.00         9
         xsnoop       0.00      0.00      0.00         4
          xterm       0.00      0.00      0.00        13

       accuracy                           0.71     22544
      macro avg       0.29      0.30      0.26     22544
   weighted avg       0.60      0.71      0.61     22544

Accuracy: 0.7098
```

As you can see here, the accuracy for the SVM model is: 71%

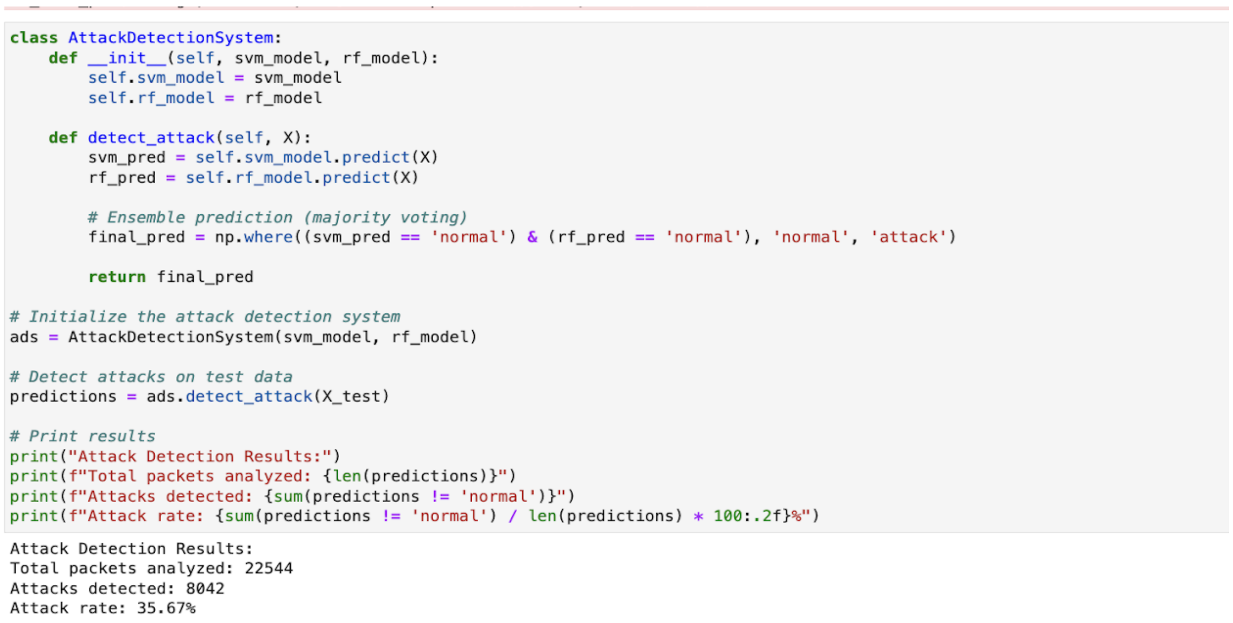
Random Forest Model Performance:



```
Random Forest Model Performance:
                 precision    recall  f1-score   support

        apache2       0.00      0.00      0.00       737
           back       0.93      0.96      0.94       359
buffer_overflow       0.00      0.00      0.00        20
      ftp_write       0.00      0.00      0.00         3
   guess_passwd       0.00      0.00      0.00      1231
     httptunnel       0.00      0.00      0.00       133
           imap       0.00      0.00      0.00         1
        ipsweep       0.58      0.98      0.73       141
           land       1.00      0.29      0.44         7
     loadmodule       0.00      0.00      0.00         2
       mailbomb       0.00      0.00      0.00       293
          mscan       0.00      0.00      0.00       996
       multihop       0.00      0.00      0.00        18
          named       0.00      0.00      0.00        17
        neptune       0.96      1.00      0.98      4657
           nmap       0.99      1.00      0.99        73
         normal       0.64      0.98      0.77      9711
           perl       0.00      0.00      0.00         2
            phf       1.00      0.50      0.67         2
            pod       0.72      0.93      0.81        41
      portsweep       0.75      0.97      0.84       157
   processtable       0.00      0.00      0.00       685
             ps       0.00      0.00      0.00        15
        rootkit       0.00      0.00      0.00        13
          saint       0.00      0.00      0.00       319
          satan       0.65      1.00      0.78       735
       sendmail       0.00      0.00      0.00        14
          smurf       0.99      1.00      1.00       665
    snmpgetattack     0.00      0.00      0.00       178
      snmpguess       0.00      0.00      0.00       331
      sqlattack       0.00      0.00      0.00         2
       teardrop       0.24      1.00      0.39        12
       udpstorm       0.00      0.00      0.00         2
    warezclient       0.00      0.00      0.00         0
    warezmaster       0.50      0.00      0.00       944
           worm       0.00      0.00      0.00         2
          xlock       0.00      0.00      0.00         9
         xsnoop       0.00      0.00      0.00         4
          xterm       0.00      0.00      0.00        13

       accuracy                           0.72     22544
      macro avg       0.25      0.27      0.24     22544
   weighted avg       0.57      0.72      0.62     22544

Accuracy: 0.7222
Models trained successfully.
```

The Accuracy of the entire Random Forest Model performance is: 72%

Attack Detection Results:

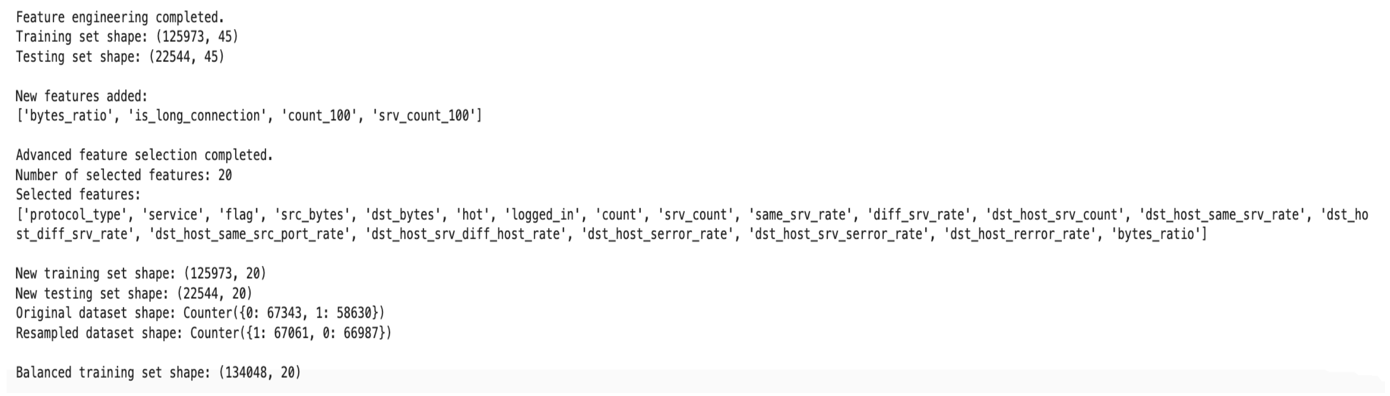


```
class AttackDetectionSystem:
    def __init__(self, svm_model, rf_model):
        self.svm_model = svm_model
        self.rf_model = rf_model

    def detect_attack(self, X):
        svm_pred = self.svm_model.predict(X)
        rf_pred = self.rf_model.predict(X)

        # Ensemble prediction (majority voting)
        final_pred = np.where((svm_pred == 'normal') & (rf_pred == 'normal'), 'normal', 'attack')

        return final_pred

# Initialize the attack detection system
ads = AttackDetectionSystem(svm_model, rf_model)

# Detect attacks on test data
predictions = ads.detect_attack(X_test)

# Print results
print("Attack Detection Results:")
print(f"Total packets analyzed: {len(predictions)}")
print(f"Attacks detected: {sum(predictions != 'normal')}")
print(f"Attack rate: {sum(predictions != 'normal') / len(predictions) * 100:.2f}%")
```

```
Attack Detection Results:
Total packets analyzed: 22544
Attacks detected: 8042
Attack rate: 35.67%
```

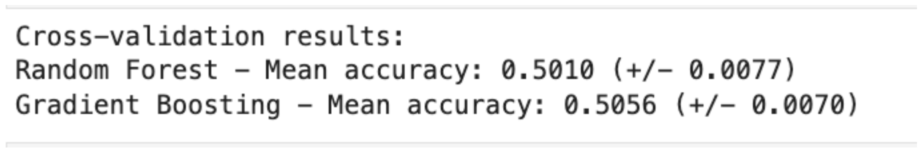
As you can see here, attacks detected are: 8042.
And attack rate is: 35.67%

Feature Engineering:



```
Feature engineering completed.
Training set shape: (125973, 45)
Testing set shape: (22544, 45)

New features added:
['bytes_ratio', 'is_long_connection', 'count_100', 'srv_count_100']

Advanced feature selection completed.
Number of selected features: 20
Selected features:
['protocol_type', 'service', 'flag', 'src_bytes', 'dst_bytes', 'hot', 'logged_in', 'count', 'srv_count', 'same_srv_rate', 'diff_srv_rate', 'dst_host_srv_count', 'dst_host_same_srv_rate', 'dst_ho
st_diff_srv_rate', 'dst_host_same_src_port_rate', 'dst_host_srv_diff_host_rate', 'dst_host_serror_rate', 'dst_host_srv_serror_rate', 'dst_host_rerror_rate', 'bytes_ratio']

New training set shape: (125973, 20)
New testing set shape: (22544, 20)
Original dataset shape: Counter({0: 67343, 1: 58630})
Resampled dataset shape: Counter({1: 67061, 0: 66987})

Balanced training set shape: (134048, 20)
```

Cross Validation Accuracy results:



```
Cross-validation results:
Random Forest - Mean accuracy: 0.5010 (+/- 0.0077)
Gradient Boosting - Mean accuracy: 0.5056 (+/- 0.0070)
```

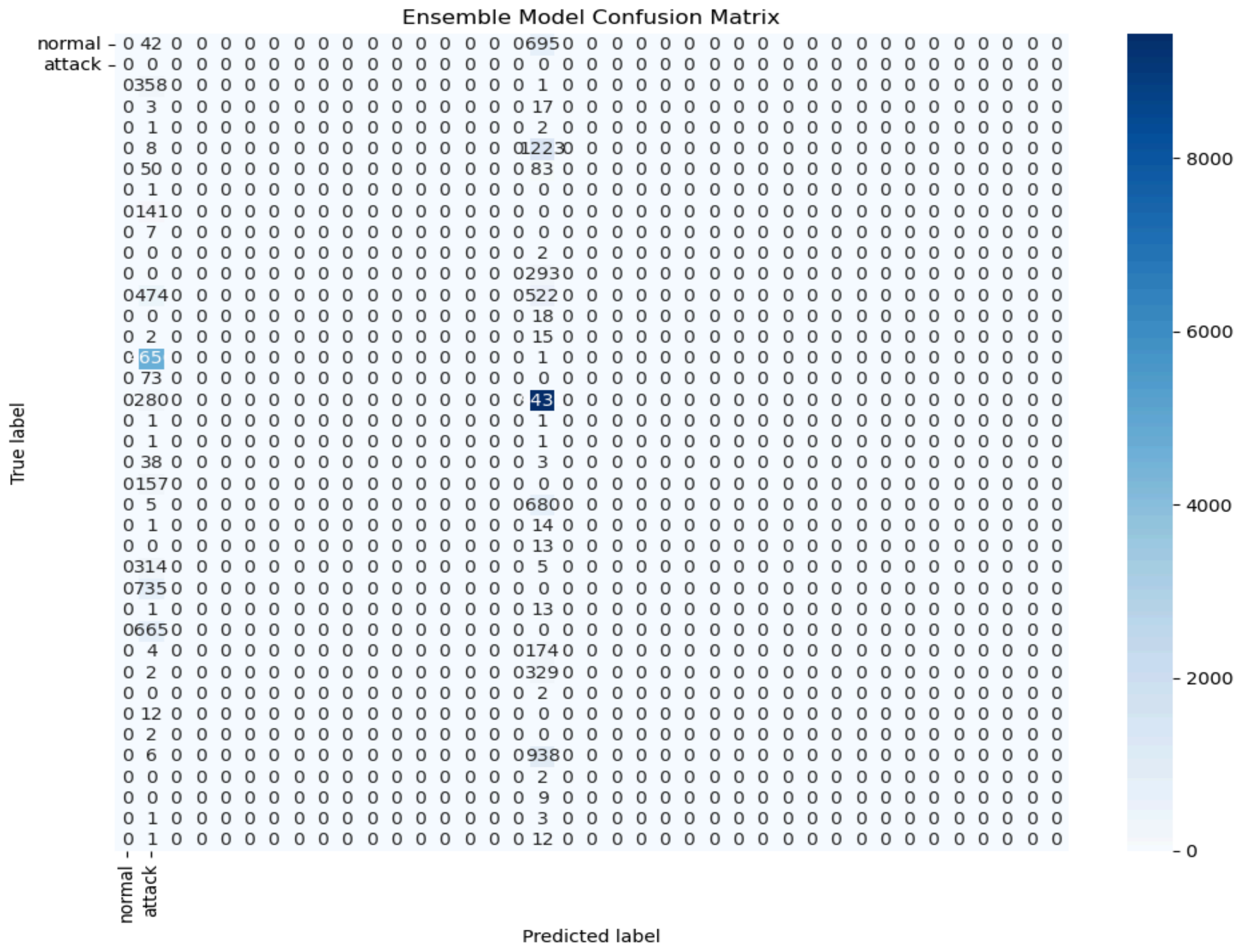
We got the Cross-validation accuracy results as above, which is unacceptable, so we must make some improvement in this like following the different method or possibly using the full feature set or a different preprocessing approach.

Reasons for the discrepancy could include:

- Different feature sets used
- Changes in data preprocessing or balancing
- Different cross-validation strategies
- Potential overfitting in the first implementation

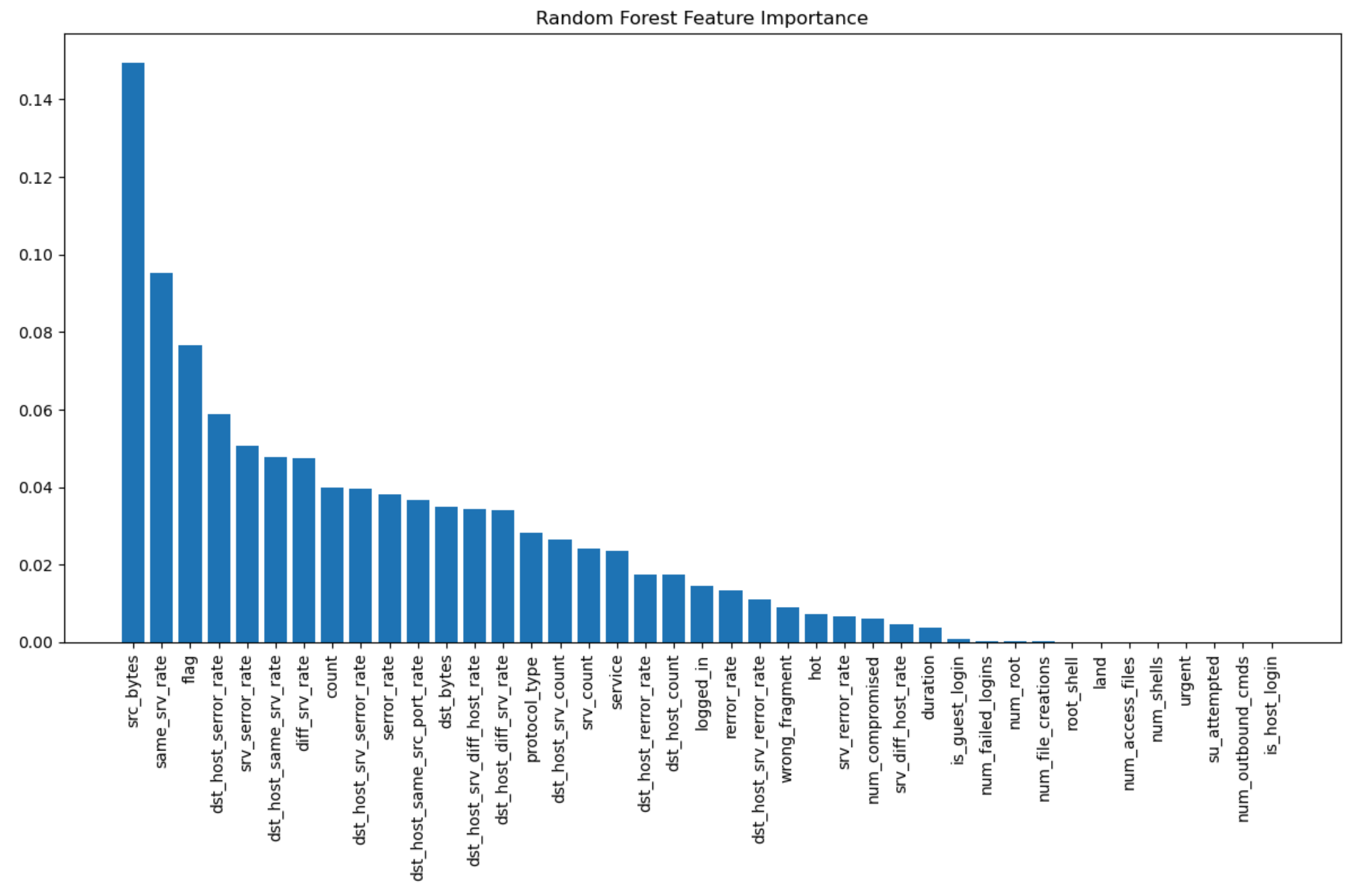
So, we must take this into account for improvement of this.

Confusion matrix:



	normal	attack																																					
normal	0	42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	695	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
attack	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	358	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1223	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	83	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	141	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	293	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	474	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	522	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	73	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	280	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	157	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	680	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	314	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	735	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	665	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	174	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	329	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	938	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Random forest:



References:

https://www.mdpi.com/1424-8220/24/19/6377
https://pmc.ncbi.nlm.nih.gov/articles/PMC11007421/