Task 3: Building Classification Models

Diana Pham

Progress Report:

For Task 3 of my final project, I used the seven feature groups that were created during Task 2 to build five classification models: Random Forest (RF), Bayesian Network (BN), Random Tree (RT), Naive Bayes (NB), and J48. The authors of the research paper briefly mentioned that these classifiers were chosen based on their research of previous works and found that they had good performance in terms of accuracy, learning ability, scalability, and speed. For each classifier, I also printed the accuracy associated with each of them so I could get a preliminary understanding of its performance, but the evaluation metrics that will be used in Task 4 will be True Positive Rate, False Positive Rate, Precision, Recall, F1 Score, Accuracy, Percentage of Incorrectly Classified, and Execution Time.

Challenges:

While following the implementation of the original research paper titled, CICIDS-2017 Dataset Feature Analysis with Information Gain for Anomaly Detection, I ran into a few issues that led to my results differing from the research paper's results:

1. The research paper built the five classification models in Weka software instead of python, so I had to look up alternative functions that were similar to Weka's functions for the models.
2. The Random Forest classifier can't handle NaN or Infinity values and the paper didn't specify how they processed these values, so I decided to do mean imputation with the training data's mean value for each column to prevent data leakage, and converted infinity values to the maximum floating point number. The difference between how these data points were handled may lead to my results being different from those in the paper.
3. Since they built their Bayesian Network in Weka and did not describe the network structure in the paper, I was unable to recreate it in python due to memory issues while processing the data.
    a. I tried two different estimators for the model, Maximum Likelihood Estimation and Bayesian Estimator, but both of them resulted in my jupyter notebook kernel crashing.
    b. I also tried using Hill Climbing Search and Bayesian Inference Criteria (BIC) Scoring to find the best Bayesian Network structure while estimating all of the parameters, but this also failed due to memory constraints.