Network Traffic Classification with Machine Learning for Network Intrusion Detection Systems

Marvin Newlin
Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH, USA
marvin.newlin@afit.edu

***Abstract*—Network Intrusion Detection Systems (NIDS) are an important aspect of cybersecurity in modern networks. One impor-tant task of many NIDS is being able to detect malicious traffic entering or exiting the network among the benign traffic. The large amounts of data that NIDS must analyze make this a suitable area of interest for Machine Learning. This research seeks to answer two questions: first, can we accurately classify network traffic as benign or malicious? Second, can we determine what features are important for making this classification? We utilize a subset the CICIDS2017 dataset to perform supervised binary classification of network traffic flows. We evaluate performance of Logistic Regression, K-Nearest Neighbors, and Balanced Random Forest classifiers on training data using the balanced accuracy score. We find that the Balanced Random Forest classifier performs the best and obtains a 95% balanced accuracy score and 0.9 recall score on the test set and additionally identify the four most important features in the classification. We hope that this will aid efforts to improve the ability of NIDS to classify network traffic.**

***Index Terms*—Machine Learning, Intrusion Detection Systems**

I. Introduction

Network Intrusion Detection Systems (NIDS) are an important aspect of cybersecurity in modern networks. One important task of many NIDS is being able to detect malicious traffic entering or exiting the network among the benign traffic. NIDS are a critical aspect to network security to prevent systems from being infected by malware or subjected to attack [1]. An important area of research for NIDS traffic classification is the utilization of Machine Learning (ML) methods for performing this traffic classification.

We utilize the CICIDS2017 dataset presented by Sharafaldin et al. in [2]. This dataset is in the form of a network packet capture and contains both legitimate and malicious traffic. The dataset contains both packet details (Source/Destination IP address, Source/Destination port number, timestamp, and protocol) and NetFlow statistics (packet rate, packet size, etc.). Each of the dataset entries is keyed as a network flow. We focus on a subset of these statistical features for the task of classifying network traffic as benign or malicious. The features are the following:

- Flow duration (seconds)
- Flow bytes per second
- Number of forward packets sent
- Number of backward packets sent
- Forward Packet Length Mean
- Backward Packet Length Mean
- Number of Forward Packets per Second
- Number of Backward Packets per Second

The problem of traffic classification presents two research questions (RQ).

1) *RQ1*: Can we successfully classify traffic as benign or malicious using the features from the list above?
2) *RQ2*: What features from the list above are most important for classifying traffic as benign or malicious?

The formal ML task to answer these questions is classification. Classification involves predicting a qualitative response for each observation, thereby separating observations into different categories [3]. Due to the immense size of the dataset, we utilize a subset which contains only one malicious type of traffic, making this task a binary classification. Additionally, the dataset is labelled, meaning that each observation has an assigned value (i.e. benign or malicious). Thus, the formal ML task is supervised binary classification [3].

We show that a Balanced Random Forest classifier is able to classify traffic in the test set with a 95% balanced accuracy score. Additionally, the classifier obtains a 0.9 recall score on the test set and we identify the top four features involved in the classification. More specific details are covered in Section IV.

The structure of this paper is organized as follows. In section II, related works and areas of research are discussed. Section III provides more detail on the dataset, the models used, methods for selecting the models, and performance measures. Section IV details the results of the classification. Section V provides conclusions and recommendations for future work.

II. Related Work

Supervised classification is a task that involves a qualitative prediction based on input features. Classification of network traffic has been discussed in previous research in a variety of forms. In this section we discuss some previous work in the areas of Supervised Classification, Unsupervised Classification, and Classification with Imbalanced Datasets.

A. Supervised Traffic Classification

Along with generating the CICIDS2017 dataset, [2] performed traffic classification on the dataset along with feature selection. They utilized several techniques including K-

Nearest Neighbors (KNN), Random Forest (RF), Quadratic Discriminant Analysis (QDA), and several others. In addition to classification analyze the important features for detecting the different attacks within the dataset. With several of the algorithms they applied they achieved high accuracy, precision, and recall. However, they did not evaluate the significance of the weights of the selected features and in many cases the weights were almost zero which made it difficult to interpret the value of the selected features.

B. Unsupervised Traffic Classification

Mukkamala et al. [4] explored utilizing neural networks and Support Vector Machines (SVM) for NIDS. They utilized the KDD-CUP 1999 dataset and developed both a neural network based IDS and an SVM IDS. With both of these systems, they were able to achieve high classification accuracy, over 99% accuracy on the test set. However, they do not discuss any other metrics such as precision, recall, or F1 measure. Additionally, they do not discuss any of the pertinent features involved in the classification. The KDD-CUP 1999 dataset is also outdated since it was generated in 1999 and is not representative of today's network traffic.

Chitrakar and Chuanhe [5] explored utilizing k-medioids clustering and naive Bayes classification for IDS traffic classification. This method uses the k-medioids clustering algorithm to cluster the traffic based on similarity metrics and then performs the classification with a naive Bayes classifier. Overall, they achieved high accuracy with this classification method and had a low false positive rate and a high true positive rate. However, they also do not discuss any features that are important to the classification.

C. Classification on Imbalanced Datasets

The CICIDS2017 dataset contains a severe class imbalance. In the subset we explore here, there are approximately 170,000 observations of which only 2,000 are malicious. This means that 99% of the data belongs to the benign class, qualifying it as severely imbalanced [6]. The problem with imbalanced data is that the traditional accuracy metric may reflect the underlying class distribution rather than correct classifications [7]. In this section, we explore some of the work related to classification on imbalanced datasets.

Haibo and Garcia in [6] explored learning from imbalanced data in the binary case. They discuss the ideas of random oversampling and undersampling. Random oversampling is done by sampling from the minority class of the dataset with replacement, thereby increasing the overall size of the minority class by the number of samples [6]. Conversely, undersampling randomly selects a smaller number of observations from the majority class without replacement and then combines that with the original minority class to increase the percentage of the minority class [6].

Some problems occur with these methods however. Oversampling leads to multiple copies of observations from the minority class in the data, thus tying the observations together and this can lead to overfitting [6]. Similarly, with undersampling, data that could be important is removed, leading to reduced performance [7].

Batista et al. [7] discuss some heuristic methods that can be used to improve classification on imbalanced datasets. One method they discuss is the One Sided Selection (OSS) heuristic. This heuristic removes borderline examples from the the majority class while maintaining the entire minority class, thus creating better separation between the classes [7]. Additionally, they discuss the Neighborhood Cleaning Rule heuristic. This rule finds the three nearest neighbors of a given observation using KNN with k=3. If the observation belongs to the majority class and the predicted class based on KNN is not the majority class, then that observation is removed. If the observation belongs to the minority class and the results of KNN misclassify it as the majority class, then the three nearest neighbors are removed [7].

Imbalanced-learn (imblearn) is an open source python package that improves on existing scikit-learn methods in order improve usability with imbalanced datasets [8]. For this research, we utilize the `BalancedRandomForestClassifier`[1] from this package. At a high level, the BRF model works by changing the sampling method used during the bootstrapping process to make the class distributions more equal. The BRF model is built on top of the scikit-learn `RandomForestClassifier` but includes extra hyperparameters to improve working with imbalanced data. These hyperparameters are `sampling_strategy` and `replacement`. The `sampling_strategy` hyperparameter allows for specification of how to sample during the bootstrapping process. This option can be specified as a float value α and when provided, the BRF model will resample during bootstrapping such that the ratio of majority to minority class is: $\alpha = \frac{\text{size of majority class}}{\text{size of minority class}}$. The `replacement` hyperparameter is a boolean for sampling with or without replacement.

Moving forward with the CICIDS2017 dataset, we utilize methods to overcome the imbalance in the data to produce a classifier that performs well. As we have discussed, accuracy alone is not a good performance measure with imbalanced datasets and we discuss the methods used in Section III.

III. Methodology

In this section, we discuss the dataset in detail, model fitting decisions, model evaluation decisions, and the analysis strategy for the classification task.

A. Dataset Details

We are utilizing a subset of the CICIDS2017 dataset. As described earlier, this dataset was synthetically generated via simulation in a sandbox and contains both benign and malicious traffic. The original dataset contains examples of Distributed Denial of Service (DDoS), Port Scanning, Web

[1] Imbalanced-learn BalancedRandomForestClassifier - https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.ensemble.BalancedRandomForestClassifier.html

Attacks, Infiltration, Heartbleed, and other types of malicious traffic. Due to the large size of the total dataset, for this work, we selected the subset of the dataset that contained only benign traffic and the web attacks. This subset allows us to perform a binary classification task between benign traffic and the malicious web attack traffic. For the rest of this work, unless specifically mentioned, references to the dataset refer to this subset and not the entire CICIDS2017 dataset.

The dataset comes in a comma separated value (CSV) file and as such, is easily imported in code. The dataset contains about 80 features that are both packet level features (IP Addresses, port numbers, etc.) and statistical features like the ones listed in Table I. Table I includes the features that were selected from the original features. Since these selected features are all statistical features, they are real valued inputs (data type `float64`) and the range for each feature is displayed in Table I. The dataset contains approximately 170,000 observations, of which 2,000 are labelled as malicious, leaving about 168,000 observations belonging to the benign class. This means that the distribution of the classes is 99% benign and 1% malicious traffic. This is a heavily imbalanced dataset, so in a later section we will discuss our strategy for handling this imbalance.

TABLE I
VALUE RANGES OF NON-TEST DATA FOR THE SELECTED FEATURES

Feature	**Data Range**
Flow duration (seconds)	$[0, 1.2 \times 10^8]$
Flow bytes/second	$[-2.61 \times 10^8, 2.07 \times 10^9]$
Number of forward packets sent	$[1, 200755]$
Number of backward packets sent	$[0, 270686]$
Forward Packet Length Mean (bytes)	$[0, 4183]$
Backward Packet Length Mean (bytes)	$[0, 3495]$
Number of Forward Packets per Second	$[0, 3 \times 10^6]$
Number of Backward Packets per Second	$[0, 2 \times 10^6]$

Due to the wide separation of each range of values for each feature from Table I, we have performed standard scaling, normalizing all values so as not to negatively impact the classification. Additionally, we have changed the labels on the label column to be a 0 for the benign class and a 1 for the malicious class. This approach is consistent with other classification tasks with a large benign class and small positive class such as cancer detection.

Exploring the dataset visually is difficult due to the high imbalance between classes and large number of data points. However, based on the scatter matrix we have selected three scatterplots to attempt to show some of the class distributions. To generate these scatterplots, we sample 5% of the data points from each class at random to display in the plot. For better interpretation, the points in the plot are not scaled and represent the original units.

In Figure 1 we see a distinct separation between the majority of the benign traffic and a few of the malicious points. However, near the (0,0) point, there is a large cluster of malicious points there as well. In Figure 2 we see a larger distribution of the benign traffic with some malicious points scattered around. In Figure 3 we also see a larger distribution of the benign traffic and two main clusters of malicious traffic.

The main thing to note is that there do not appear to be any distinct boundaries between the malicious and benign traffic. This indicates that the more flexible models may perform better at this classification task.

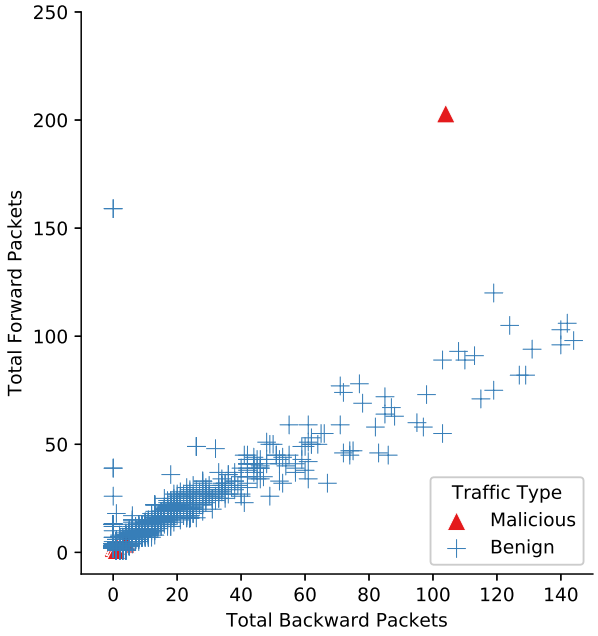

Fig. 1. Scatterplot of number of packets sent in each direction.

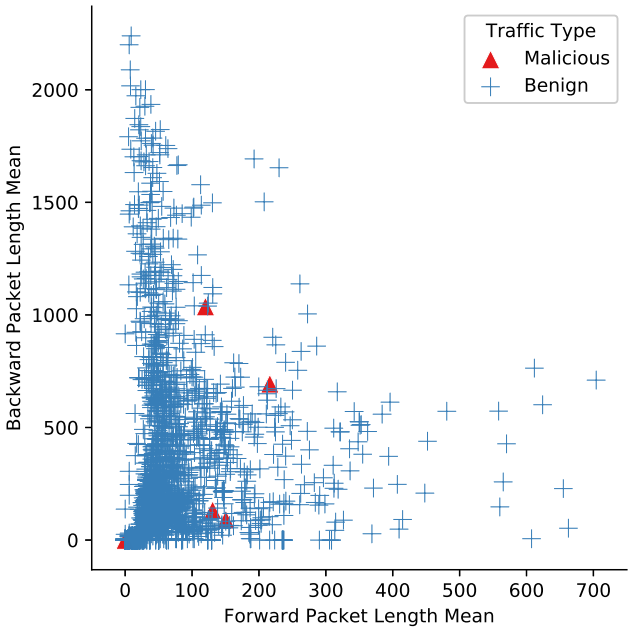

Fig. 2. Scatterplot of Packet Length Mean in each direction.

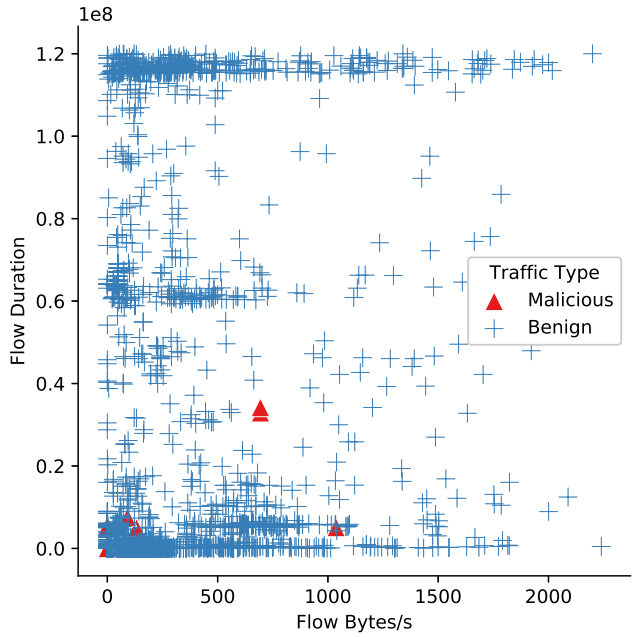

Fig. 3. Scatterplot of Flow Bytes/s vs. Flow Duration.

B. Model Fitting

The ML task being performed is a supervised binary classification task. We have partitioned the data into non-test and test sets, and since the dataset is large we partition $\frac{2}{3}$ of the data as the non-test set and $\frac{1}{3}$ of the data as the test set. We utilize the using the scikit-learn `train_test_split` function to perform the partitioning. For classification models we evaluate Logistic Regression (LR), K-Nearest Neighbors (KNN), and Balanced Random Forest (BRF). For all model tuning and hyper-parameter decisions we use standard 10-fold cross validation to select the best performing model on the non-test set and then evaluate that best model on the test set.

For fitting LR, we examine the different values for the cost parameter `C` with cross validation and utilize the L1 penalty for LASSO regularization with the '`saga`' solver. For KNN, we explore values of k using cross validation to determine the best value of k by selecting the k for which the average cross validation balanced accuracy score is maximized for the model. With the chosen value of k we then compare its average cross validation balanced accuracy score with the other models to select the best classifier with respect to the balanced accuracy score. For BRF, we set the hyperparameters `sampling_strategy` and `replacement` to 1.0 and `True` respectively. We utilize `sampling_strategy` $=$ 1.0 in order to balance the class distributions by sampling with replacement during the bootstrapping process in BRF. Further, with cross validation we manually tune the hyperparameters `num_estimators`, `max_features`, and `max_depth` based on highest balanced accuracy.

For performance evaluation, since we have an imbalanced dataset we utilize the scikit-learn `balanced_accuracy_score` function[2]. This function reports the average accuracy for each class weighted according to the inverse of the class frequency [9], [10]. For model evaluation, the scikit-learn classification functions support class weights in order to balance the classification[3]. Additionally, to answer RQ2, we utilize a feature selection method based on the best performing model. If LR or KNN is the best performing model, we will utilize Forward Stepwise Subset Selection (FSS) to select the top four features. If BRF is the best performing model, we will utilize the `feature_importances_` attribute to select the top four features.

C. Model Evaluation

Since we are comparing the performance of several different models, we use cross validation to make the selection of the best performing model. After choosing this best classifier we retrain the selected model on the entire non-test set before evaluating test set performance. Additionally, to answer RQ2, we will utilize a feature selection method based on the best performing model. If LR or KNN is the best performing model, we will utilize Forward Stepwise Subset Selection (FSS) to select the top four features. If BRF is the best performing model, we will utilize the `feature_importances_` attribute to select the top four features.

We also utilize classification evaluation metrics to evaluate performance. We use the Precision Recall (PR) curve and the Area Under the Curve (AUC) metric to determine the informedness of the classifiers. We utilize PR curve instead of the Receiver Operater Characteristic (ROC) curve because the PR curve provides better information in imbalanced datasets when we are dealing with a minority positive class (e.g. cancer detection) and are more concerned with correctly identifying elements of the positive class [6], [11]. We also leverage confusion matrices and the precision and recall metrics to ensure that our classification is correctly classifying the data and not just reflecting the underlying data distribution.

IV. Results

In this section, we explore the results of the classification task. The results of hyperparameter tuning for each model are discussed in Subsection IV-A. The comparison and evaluation of the three models is discussed in Subsection IV-B. The test set performance and evaluation is discussed in Subsection IV-C. Finally, feature importance in the test set is discussed in Subsection IV-D.

A. Model Fitting

In this section, we explore the results of tuning the hyperparameters for the three models.

[2]scikit-learn Balanced Accuracy Score - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html

[3]scikit-learn Class Weights - https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html

1) Logistic Regression: For fitting an LR model to the data, we explored 10 linearly spaced values of the cost parameter `C` in $[0.0001, 0.0025]$. These values were chosen as they were the only values for which the LR model would converge when fitting on the non test set in cross validation. The cost parameter is the inverse of regularization strength so these small values correspond to a large regularization penalty. Additional hyperparameters for the LR model we used were the following:

- `penalty` = '`l1`' - for LASSO regularization
- `solver` = '`saga`' - Stochastic Average Gradient solver for L1 penalty
- `max_iter` = 1000 - Maximum number of iterations allowed during fitting, chosen because no smaller number would converge

In Figure 4 we see the cost values plotted against the average cross validation balanced accuracy. The black 'x' mark annotates the cost value for the maximum accuracy. This maximum balanced accuracy occurs at a cost of 0.0006333. We then use this value for cost in the best LR model for performance comparison.

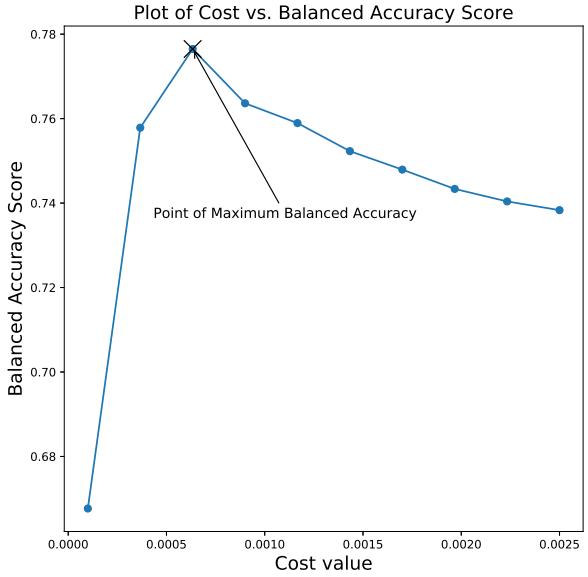

Fig. 4. Plot of Logistic Regression cost value vs. 10-fold cross validation balanced accuracy.

2) K-Nearest Neighbors: For KNN we tuned the k hyperparameter. We tested the balanced accuracy of a KNN model for all k in $[1, 40]$. The plot of the k values versus balanced accuracy score are displayed in Figure5. As we can see and we expect, the balanced accuracy for lower k values is fairly low and increases up until the maximum balanced accuracy at $k = 36$ and then begins to decrease after that. Interestingly, the balanced accuracy for odd numbers is generally higher than for the even values of k. We conjecture that this is due to a 50/50 vote in the even k values resulting in higher misclassification during cross validation.

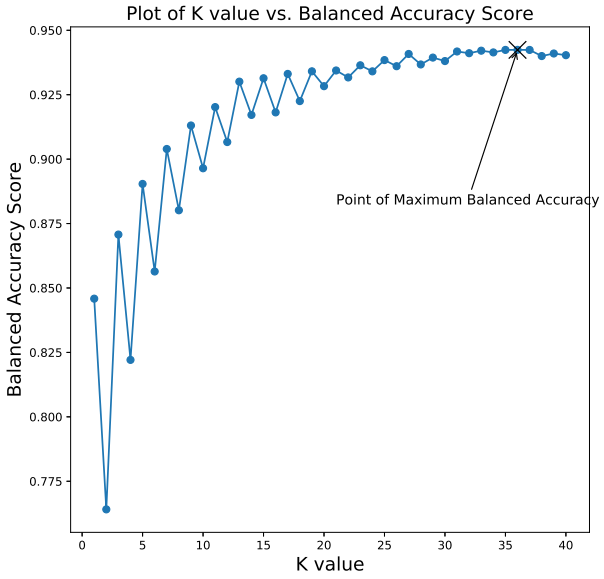

Fig. 5. Plot of KNN k vs. average 10-fold cross validation balanced accuracy.

3) Balanced Random Forest: For BRF, we explored tuning three hyperparameters: `num_estimators`, `max_features`, and `max_depth`. Additional hyperparameters chosen and not tuned were the following:

- `random_state` = 1 - Seed to control randomness
- `sampling_strategy` = 1.0 - For equal ratio of minority and majority class (details of how this is accomplished are discussed in Section II
- `class_weight` = '`balanced`' - Each class weighted according to the inverse of class frequency

Due to computational complexity and time limits, we were unable to tune the three hyperparameters together. Thus, we chose to first tune `num_estimators` and then tune `max_features` and `max_depth` together. The results of the number of trees in the BRF model versus balanced accuracy are plotted in Figure 6. The balanced accuracy for 100 trees is significantly lower than the rest and then the number of trees with the highest balanced accuracy is 600. We then tuned features and depth with `num_estimators` = 600.

For tuning the number of features we explored numbers of features between 1 and 5. We know that RF works best with a subset of the features so we didn't go all the way to 8. Additionally, we tried depths between 1 and 30 resulting in a total number of $5 \times 30 = 150$ different configurations tested with 10-fold cross validation.

The results of this search are displayed in Figure 7. Here we present a contour plot of `max_features` vs. `max_depth` vs. balanced accuracy score. For low features and low depth, the balanced accuracy score is very low. However, once past

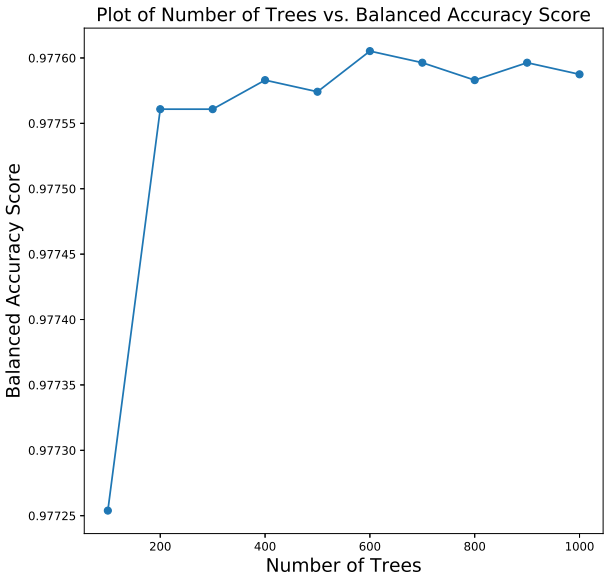

Fig. 6. Plot of BRF `num_estimators` hyperparameter vs. balanced accuracy.

a depth of 10, the balanced accuracy scores are very close together. The maximum balanced accuracy is indicated by the orange 'x' and occurs at `max_features` = 5 and `max_depth` = 19. With these parameters chose we had a BRF model with `num_estimators` = 600, `max_features` = 5, and `max_depth` = 19.

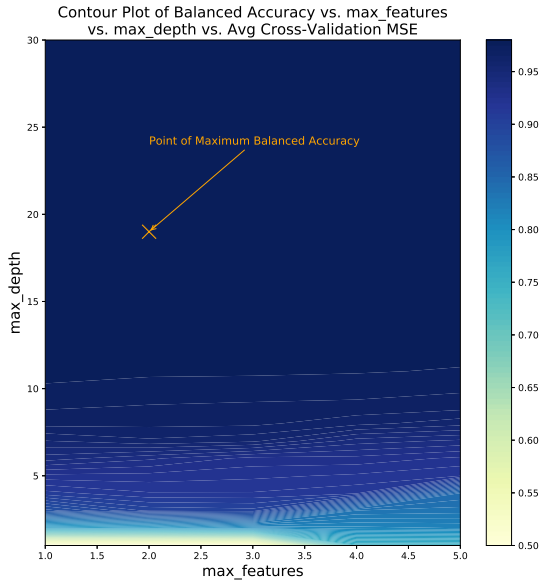

Fig. 7. Contour plot of BRF `max_features` and `max_depth` hyperparameters vs. balanced accuracy.

B. Model Evaluation

With the hyperparameters for each model selected, we compare the performances of each model using the balanced accuracy score. The results of balanced accuracy score for each cross validation fold are displayed in Figure 8. Balanced Random Forest has the highest balanced accuracy at around 97.5%. KNN has the second highest balanced accuracy at around 93% and Logistic Regression is the worst at around 81%. Since it has the highest average cross validation balanced accuracy, we select BRF as the best performing model and evaluate its performance on the test set.

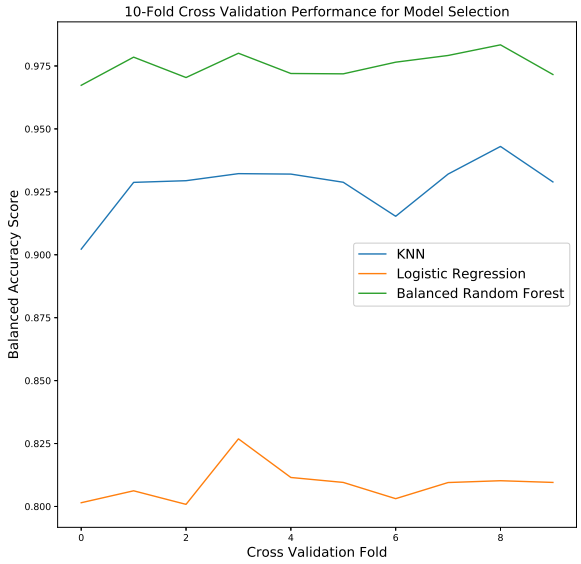

Fig. 8. Cross validation balanced accuracy scores for LR, KNN, and BRF.

Before evaluating performance, we had to make one final decision. We needed to decide the probability threshold that we would use to make our classification after predicting the class probabilities with our model. This time, we partition the non test set into a training set and a validation set with $\frac{2}{3}$ training and $\frac{1}{3}$ validation. We then fit the BRF model on the training partition and then predict class probabilities on the validation partition. For thresholds, we create 100 linearly spaced thresholds between the minimum predicted probability and the maximum probability. The PR curve for the validation set is displayed in Figure 9. Like the ROC curve, we want the area under the curve to be as close to 1 as possible. For the validation set we obtain an AUC of 0.785, which could be better but on the whole is not bad.

To choose our operating point, which is the probability threshold that we will make our classification on, we select the probability threshold for which the F1 score is maximized. We select this threshold because F1 score provides a balance between precision and recall. This is desirable because on the one hand we do not want to classify malicious traffic as benign

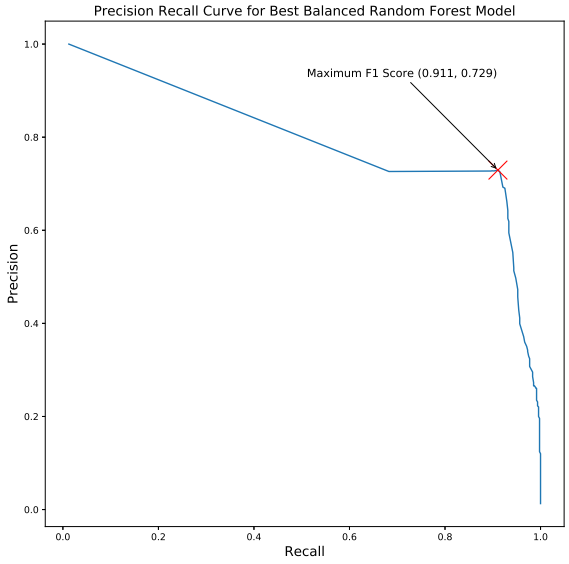

Fig. 9. Precision Recall curve for validation set. Point of maximum F1 score is denoted with a red 'x'. AUC: 0.785

but we also do not want to classify benign traffic as malicious. Thus, the probability threshold for maximum F1 score is the desired threshold. The red 'x' in Figure 9 represents the point where the maximum F1 score is reached and the corresponding probability threshold is $p = 0.89$. With this operating point established, we move on to test set performance.

C. Test Set Performance

After determining the operating point, we refit the BRF model on all of the non test data. We then predict class probabilities and then based on the threshold established in Subsection IV-B, we make classifications from the predicted probabilities. To evaluate our performance, we create a confusion matrix of the test set with predicted classes versus the actual classes. The confusion matrix is displayed in Table II. The results of the classification with the BRF model are the following:

- Balanced Accuracy: 0.9514
- Recall: 0.9068
- Precision: 0.7512
- F1 Score: 0.8217

The two most important scores are the balanced accuracy and the recall scores. The 0.95 balanced accuracy is fairly good, however, the 0.9 recall is really good as well. Even though we are balancing precision and recall, we would prefer higher recall because this means that fewer false negatives occur, i.e. less malicious traffic is misclassified. We are okay with a lower precision rate of 0.75 because false positives are not as harmful to the network as false negatives are. Figure 10 displays the PR curve for the test set and the red 'x' indicates the precision and recall scores as an (x,y) pair. Additionally, the AUC for the test set PR curve is 0.804 which is an increase from the validation set AUC, meaning that our classifier has improved on the test set.

TABLE II
CONFUSION MATRIX FOR TEST SET RESULTS

$p = 0.89$	**Predict Benign**	**Predict Malicious**	**Total**
Actual Benign	55242	216	55458
Actual Malicious	67	652	719
Total	55309	868	56177

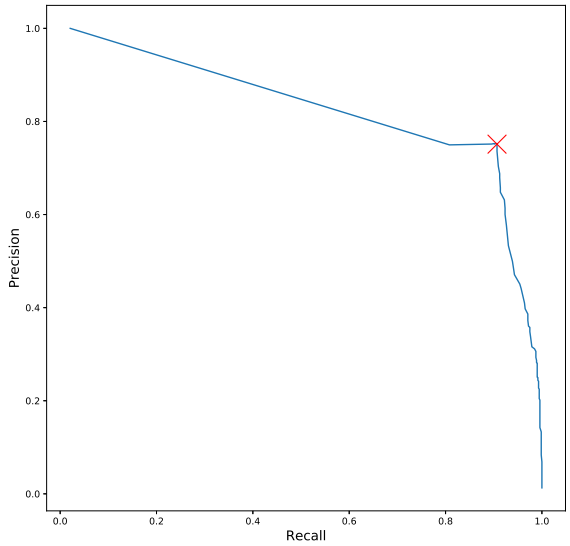

Fig. 10. Precision Recall curve for test set. Red 'x' is test set recall and precision as (x,y) pair. AUC: 0.804

D. Feature Importance

Test set results in hand, we examine feature importance using the `feature_importances_` attribute of our BRF model. This is one of the nice features of the Random Forest framework, in that we can see what features are important based on the higher level decisions from each tree in the forest [12]. These feature importances are displayed in Figure 11. All scores are normalized to be between 0 and 1 and the sum of all the weights add up to 1. The higher the score, the more important the feature is in making the predictions [12]. The top four features for making the classification are:

1) `fwd_packet_length_mean`: 0.26
2) `flow_bytes/s`: 0.17
3) `flow_duration`: 0.14
4) `fwd_packets/s`: 0.13

With these results, we are able to answer RQ2 definitively that we can indeed determine the important features for classification.

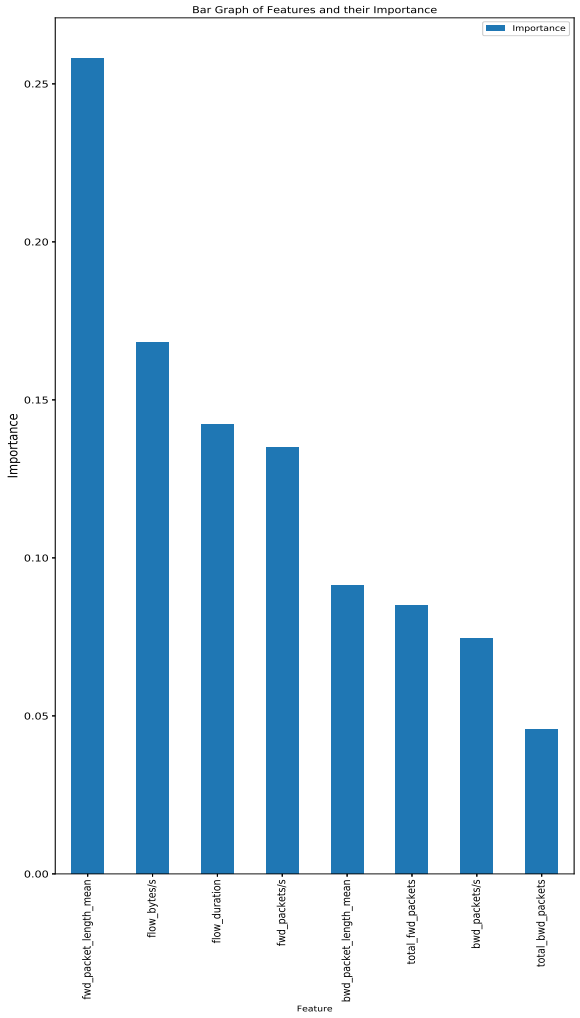

Fig. 11. Feature importance graph from BRF model.

V. Conclusions and Future Work

This research has sought to answer two research questions. First, can we accurately classify network traffic using a set of the statistical features of a network flow dataset? Second, can we identify the features that are important to this classification? To answer RQ1, we were successful in classifying malicious network traffic with a Balanced Random Forest classifier and achieved the following scores on the test set:

- Balanced Accuracy: 0.9514
- Recall: 0.9068
- Precision: 0.7512
- F1 Score: 0.8217

To answer RQ2, we were also able to identify the four most important features from the Balanced Random Forest model. They are (in order):

1) `fwd_packet_length_mean`
2) `flow_bytes/s`
3) `flow_duration`
4) `fwd_packets/s`

Even though this classification was mostly successful, there are two areas where performance could be improved in future work. The first would be to utilize the full feature set which contains approximately 80 features. Incorporating these we believe would help further improve the classification of malicious network traffic. Additionally, due to the highly non-linear relationships present in the data, deep learning methods may be another avenue of future work that could improve performance on this classification task.

With this success in classifying malicious network traffic, hopefully this can be implemented on NIDS. With the high accuracy and recall rates, utilizing ML to assist in this task may help improve USAF cybersecurity in the future.

References

[1] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7307098/

[2] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018, pp. 108–116. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[3] G. James, D. Witten, T. Hastie, and R. Tibshirani, "Classification," in *An Introduction to Statistical Learning*. New York: Springer Science+Business Media, 2013, ch. 4, pp. 127–173.

[4] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proceedings of the 2002 International Joint Conference on Neural Networks*. IEEE, 2002.

[5] R. Chitrakar and H. Chuanhe, "Anomaly based Intrusion Detection using Hybrid Learning Approach of combining k-Medoids Clustering and Naïve Bayes Classification," in *8th International Conference on Wireless Communications, Networking and Mobile Computing*, Shanghai, China, 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6478433

[6] Haibo He and E. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 9 2009. [Online]. Available: http://ieeexplore.ieee.org/document/5128907/

[7] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, 2004. [Online]. Available: https://dl-acm-org.afit.idm.oclc.org/citation.cfm?id=1007735

[8] G. LematreLematre, F. Nogueira, and C. K. Aridas char, "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning," Tech. Rep., 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://scikit-learn.org/stable/index.html

[10] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The Balanced Accuracy and Its Posterior Distribution," in *2010 20th International Conference on Pattern Recognition*. IEEE, 8 2010, pp. 3121–3124. [Online]. Available: http://ieeexplore.ieee.org/document/5597285/

[11] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in *Proceedings of the 23rd international conference on Machine learning - ICML '06*. Pittsburgh, PA: ACM Press, 2006, pp. 233–240. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1143844.1143874

[12] G. James, D. Witten, T. Hastie, and R. Tibshirani, "Variable Importance Measures," in *An Introduction to Statistical Learning*. New York: Springer Science+Business Media, 2013, ch. 8.2.1, p. 319.