Analysing different machine learning techniques to separate network intrusion traffic from safe traffic and classifying the type of intrusion

Diptanshu Mittal
2018232
diptanshu18232@iiitd.ac.in

Manavjeet Singh
2018295
manavjeet18295@iiitd.ac.in

Rhythm Gupta
2018082
rhythm18082@iiitd.ac.in

Abstract

With increasing use of internet especially after unfolding of global pandemic it is important to review security of cyber-systems. Network intrusion attacks can lead to loss of leaking of personal and sensitive data of the target organisation as well as it's clients, and in worst case cause financial losses. To tackle modern day attacks it is important to make systems that can learn from the available data to classify between safe and unsafe network traffic. We are using UNSW-NB15 dataset to overcome the problem of detecting intrusions and classifying them into different types. In this project until this point we have performed feature selection from given more than forty features using filter and wrapper techniques. We performed training and hyper-parameter tuning for DT, SVM, KNN and Random Forest models. For binary classification as safe or unsafe traffic random forest performed best and for classifying the type of attack KNN.

Code is available ***here****.*

1. Introduction

The ever increasing sophistication of network intrusion attacks raises a serious issue in network security and privacy. Configuring a fixed set of rules, for example firewalls, might fail to detect these modern attacks. To tackle this problem a more adaptive, dynamic and accurate approach is required. Machine Learning techniques can satisfy all three requirements.

The goal of this project is to create two classifiers, such that, given the information about network traffic, first one is the binary classifier that can classify between safe traffic and intrusion traffic and the second is multi-class classifier that can classify between different types of attacks. In this project until this point we used KNN, DT and SVC models and tuned them to get best possible results by selecting best hyper-parameters and training on best split.

2. Literature Survey

There are a number of pre-build datasets available to train models for network intrusion detection,[4] compares two other datasets with the UNSW-NB15 data set used in this project. First is KDDCup99, in this data set the probability distribution of the testing set is different from the probability distribution of the training set, this leads to skew or bias classification methods to be toward some records. Also, KDDCup99 is not a comprehensive representation of recently reported low footprint attack projections. The other data set is NSLKDD, This is a newer and modified version of KDDCup99 dataset, which solves the problem of duplication of records in test and the unqual probability distribution in KDCup99. But, it still lacks records showing low footprint attacks.

Recent works in this area highlight the usage of Deep Learning to learn the features of incoming network packets to determine the kind of traffic. [5] gives an insight into general process for feature selection. It compares filter and wrapper methods for feature selection. Finally it uses DT for wrapper based feature selection because of its ability to handle both categorical and numerical data and also it is much more prone to under-fitting. [3] reviews various datasets available in NID and conducted a comparative study of several DL approaches on CICIDS2018 dataset and the Bot-IoT dataset. [2] uses a hybrid model consisting of AutoEncoder and Deep Neural Network for classification of network traffic. [1] examines auto-encoder and PCA for dimensionality reduction and the use of classifiers towards designing an efficient network intrusion detection system on the CICIDS2017 dataset.

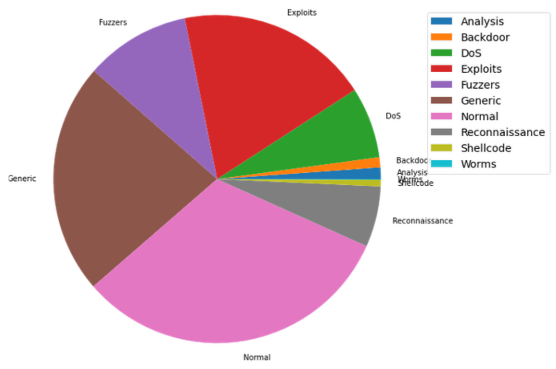
3. Dataset

On the basis of results of [4] we chose to work with UNSW-NB15 dataset.
The dataset is given in the form of pcap files and two CSV files of features extracted from pcap files for training and testing purpose. We have decided to directly use the CSV files for training and testing.

3.1. Distribution

The distribution of training set is represented in figure 1.

Figure 1. Attack Category Distribution in Dataset



3.2. Features

Features present in the dataset are extracted from packets and are of following categories:
1. **Basic Features :** These are the basic information extracted from packet headers like TTL value, packet size, etc.
2. **Content Based Features :** These are based on the packet content like data size, window size, mean flow, etc,
3. **Time Related Features :** These are based on time attributes like TCP setup time, total time taken, etc
4. **Additionally generated features :** These are generated based on additional features like similarity/dissimilarity in IP address over a period of time
In total there are 5 categorical and 34 numerical features.

3.3. Selected features

sttl, sload, dload, swin , dwin, dmean, sinpkt, ct-state-ttl, ct-srv-src, ct-srv-dst, ct-dst-ltm, ct-src-ltm, ct-src-dport-ltm, ct-dst-sport-ltm, ct-dst-src-ltm , proto, state, is-sm-ips-ports

3.4. Data Visualization

We used two visualization techniques PCA and tSNE to visualize the UNBSW-15 dataset. From figure 2, we can see that labels like normal, exploits, generic are divided into multiple clusters of small sizes. Moreover, the decision boundaries for individual classes are not noticeable from the plots. Clusters of different classes appear to be overlapping with each other and the dataset is having 'class overlap' problem. From the figure 4, we can see that classes like Generic, Exploits, Dos are overlapping with each other and Fuzzers, Reconnaissance, Normal are overlapping with each other.

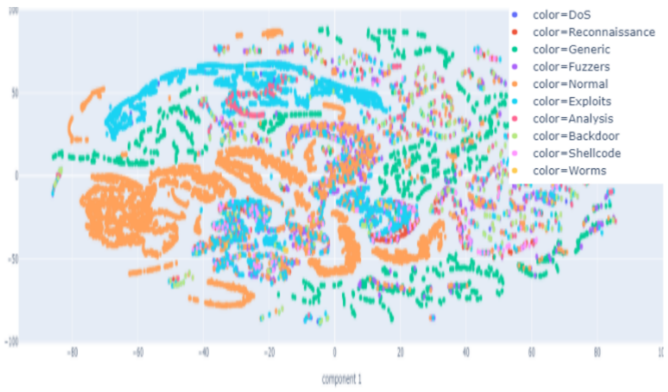


Figure 2. 2-D tSNE plot of the dataset

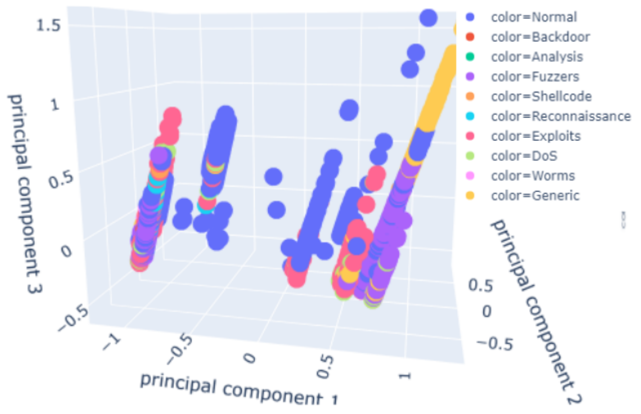


Figure 3. 3-D PCA plot of the dataset

4. Methodology

4.1. Preprocessing

1. **Feature Selection:** We have used three different techniques for feature selection viz Wrapper Based, Filter Based and Embedded and then took union of all feature sets we got, to obtain features important from atleast one aspect. In **Wrapper** method we trained a Naive Bayes and decision tree model with all the features and moved backwards by removing the least significant feature after each iteration. In **Filter** method we used Anova F1 score for numerical feature selection and mutual-info-classif

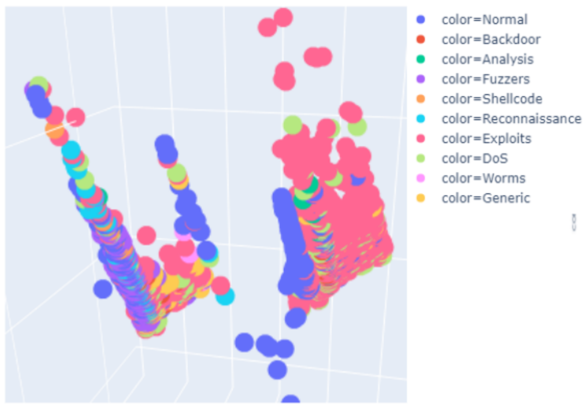
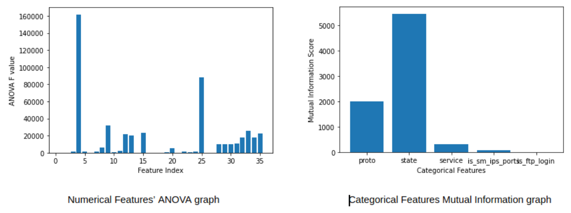


Figure 4. Subset of zoomed in version of PCA

for categorical feature selection. In **Emebedded** methods LinearSVC model was trained and using SelectFromModel important feature were selected from the trained model.

Figure 5.



2. **Encoding Categorical Features:** To encode categorical features, we used target encoding.

3. **Feature Scaling:** We scaled the features using min-max scaler. Because the inter-cluster overlap was minimized, the results improved drastically after min-max scaling.

4. **Dataset balancing:** Distribution of classes in attack category was highly unbalanced. Thus, dataset balancing was done. SMOTE-NC was used for over-sampling minority classes as it is capable of handling mixture of categorical and numerical attributes. RandomUnderSampling and Tomek Links was used for under-sampling the majority classes.

4.2. Evaluation Metrics

Most commonly used metrics for evaluating performance of a classifier are:

```
Accuracy = (TP+TN) / (TP+TN+FP+FN)
Recall(rec) = TP / (TP+FN)
```

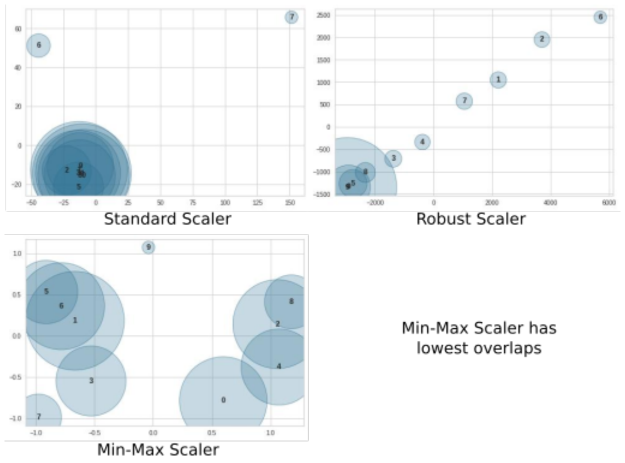


Figure 6. Inter-cluster distance map for multiclass data

```
Precision(pre)  = TP / (TP+TN)
F1 score = (2*pre*rec) / (pre+rec)
```

Here TP, TN, FP, FN refer to True Positive, True Negative, False Positive, False Negative. We have taken intrusion as positive class. Accuracy gives the fraction of samples correctly classified by model and has been used as a metric in previous works. Recall in this case will give the fraction of intrusion traffic correctly classified. Precision will give the fraction of traffic classified as intrusion out of all correctly classified traffic. F1 score is the harmonic mean of precision and recall.

We use accuracy and recall to evaluate binary classifier. Since the distribution of intrusion and normal traffic is almost equal in testing data, accuracy can be used for evaluation. But among two models of almost equal accuracy, one with more recall value is selected. This is because a classifier which doesn't predict intrusion traffic as normal is more favourable.

For the purpose of multi-class classification however the aim is to identify the type of attack. Also the distribution of classes is uneven in testing data. Therefore we use weighted F1-score alongside accuracy for evaluation purpose.

4.3. Models

Support Vector Classifier -

It is a supervised machine learning algorithm capable of classifying datasets. SVC finds the best fitting hyper-plane which can divide the dataset into various classes. SVC use a function to transform the given input into a required form such that they are linearly-separable. These functions are called as kernels. These kernels can be different types likes polynomial, linear, radial basis function and sigmoid.

For intrusion detection, SVC with kernel - RBF, Poly, Sigmoid and decision boundary as one-vs-one was used. Polynomial kernel computes the relationship between the datapoints using function $(a * b + r)^d$. RBF kernel computes the relationship in infinite dimensions. Sigmoid kernel uses sigmoid function to compute the relationship between datapoints. For attack categorization SVC with kernel - RBF and Sigmoid and decision boundary as one-vs-one was used. Other kernels were not used as they were computationally expensive.

Stochastic Gradient Descent -
The SGD algorithm is similar to Logistic Regression algorithm and passes weighted sum of inputs to sigmoid function which in turn converts it to some class labels based on the threshold. Only difference comes in optimization function as SGD is using Stochastic Gradient descent instead of Gradient descent. Thus is converges much faster and is computationally cheaper. After optimization hinge loss function with l2 penalty gave best results.

Linear Support Vector Classification -
LinearSVC is similar to Support vector classification with linear kernel, but it is implemented in liblinear instead of libsvc. Due to this is scales better and is computationally viable in large datasets. After optimization squared hinge loss function with l2 penalty gave best results.

K-Nearest Neighbor -
The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. Fitting a KNN model is alias of learning all the data. The output is based upon the label of K nearest neighbors.

We ran a grid search to find the optimal number of nearest neighbors for both binary and multi-class classification. As evident in figure 3 and 4, we chose 21 and 16 as nearest neighbors for binary and multi-class distribution.

Because of the large number of entries in the data-set we used 10-fold cross validation to train the model on the best fold available.

Decision Tree Classifier -
Decision tree is tree based classifier which is able to handle wide variety of use-cases due to its ability to handle non-linearly separable data. However it is prone to over-fitting. For this purpose we did fine tuning on various

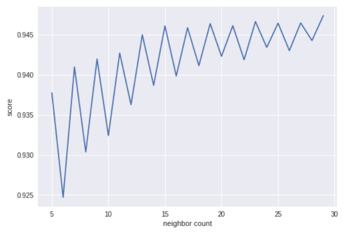


Figure 7. Neighbors for binary

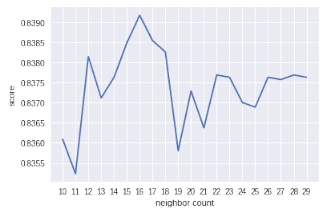


Figure 8. Neighbors for multi-class

values of max-depth and as per the results, we chose 20 as max-depth. We also used minimal cost complexity pruning for protecting Decision Tree from overfitting. For this decision tree takes an input parameter $ccp_\alpha \geq 0$ called complexity parameter. In minimal cost-complexity pruning, for each node, the $\alpha_{effective}$ of the node with each of its children nodes is calculated. And the subtree corresponding to the $\alpha_{effective} \leq ccp_\alpha$ are pruned out. Here, $\alpha_{effective}$ is defined by the following equation:
$\alpha_{effective}$ (t) = (R(T) - R(t_t)) / ($|T|$ -1)
R(T) = Total misclassification rate in node T.
$|T|$ = Number of terminal nodes in subtree with root node T.

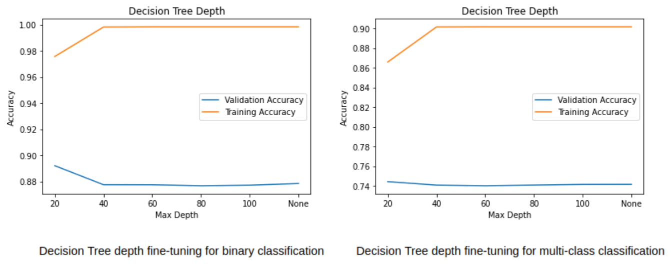


Figure 9.

Ensemble Methods -
Random Forest Classifier - It trains multiple decision trees and maximum voting for declaring the output and therefore it is more robust than decision tree. We have used forest of 100 decision trees.

XGBoost -
XGBoost is a decision tree based ensemble model using

gradient boosting framework. In boosting techniques, each tree learns from its predecessors and update the residual errors. Hence, the tree in the next generation will learn from an update version of itself. Boosting makes use of tree with fewer splits compared to other ensemble techniques like random forest which grows their classifier to maximum extent.

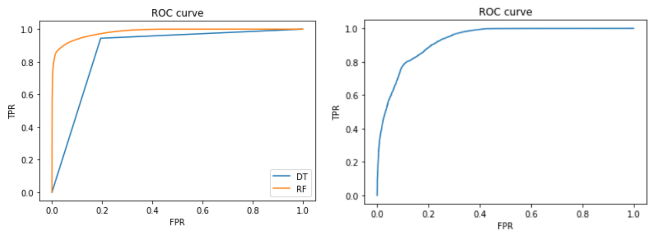
Extreme Gradient Boosting aims at minimizing the errors of weak estimators using a sequence of decision trees utilizing gradient descent algorithm. The objective function consists of training loss and a regularization term
$\mathcal{L}(\varphi) = \sum_i l(y_{i_i}, \hat{y}_i) + \sum_k \Omega(f_k)$
where $\sum_i l(y_{i_i}, \hat{y}_i)$ indicates the loss function and $\sum_k \Omega(f_k)$ represents the regularization term which is equal to $\gamma T + 1/2\lambda \parallel w \parallel^2$. Regularization helps in making the final weights gained by training the model smoother.

Stacking Models for Multiclass Classification:
We noticed that all the classifiers in binary classification performs much better than the classifiers in multiclass classification. We decided to use its performance in multiclass classification by making a custom classifier consisting of two models. The first model classifies the input into intrusion and non intrusion traffic and the entries labelled intrusion by the first model are then forwarded to next model which classifies the type of intrusion.

We chose decision tree for binary classification model in this case and made two such stacking models with decision tree and XGBoost as multiclass classifiers respectively.

5. Results and Analysis

Results of different models are given in Table 1 to Table 6. We have taken Intrusion to be positive class, hence recall in binary classification measures the fraction of intrusion attacks that were detected. DT, RF, SVC, KNN have been used as abbreviations of Decision Tree, Random Forest, Support Vector Classifier and K Nearest Neighbours respectively.



Decision tree and Random Forest Roc curve

SVM(RBF) Roc curve

Figure 10.

Models	Accuracy(%)	F1	Recall
DT	92.09	0.93	0.95
SVC(RBF)	81.47	0.87	0.99
SVC(Poly)	81.44	0.86	0.99
SVC(Sigmoid)	68.02	0.74	0.81
SGD Classifier	78.54	0.84	1.00
LinearSVC	78.49	0.82	0.89
KNN	89.30	0.89	0.89

Table 1. Results of various models for intrusion detection.

Models	Accuracy(%)	F1	Recall
DT	76.87	0.77	0.77
SGD Classifier	64.80	0.66	0.65
SVC(RBF)	65.27	0.67	0.65
SVC(Sigmoid)	54.49	0.57	0.54
KNN	75.67	0.77	0. 76

Table 2. Results of various models for attack classification.

Models	Accuracy(%)	F1	Recall
XGBoost	91.34	0.92	0.95
Voting	91.42	0.92	0.96
RF	91.31	0.91	0.91

Table 3. Results of Ensemble Classifiers for attack detection

Models	Accuracy(%)	F1	Recall
XGBoost	77.01	0.78	0.77
Bagging (DT)	75.59	0.76	0.76
RF	74.93	0.79	0.75

Table 4. Results of Ensemble Classifiers for attack classification

Models	Accuracy(%)	F1	Recall
DT+DT	80.68	0.79	0.81
DT + XGBoost	**81.33**	**0.80**	**0.81**

Table 5. Results of Custom Model for attack classification

Models	SVC(RBF)	DT	RF
ROC-score	0.92	0.87	0.98

Table 6. ROC Score of various models for intrusion detection.

Accuracy of SGD classifier improved from 0.449 to 0.785 after feature scaling. Results degraded when SMOTE-NC was used to over-sample the minority classes. SMOTE-NC creates artificial data points, which can differ from actual data-points. Thus, SMOTE-NC algorithm was disregarded. RandomUnderSampling was performing better in the binary classification but Tomek links performed better in

Multi-Class classification. RandomUnderSampling randomly removes the instances of majority classes whereas Tomek Links finds the pair of majority-minority data points having smallest euclidean distance and removes the majority class data points. Tomek links algorithm also leads to noise removal. Due to this Tomek might be performing better in multi-class classification

Decision Tree and KNN performed better than other classifiers on binary classification because they are able to make non-continuous decision boundaries and form clusters for classes. SVC makes continuous hyper planes due to which it is not able to resolve class overlap problem and is under performing.

Random Forest is an ensemble classifier with decision tree as base classifier, and it will be having lower variance and less than equal to bias when compared to decision tree model.

Although the accuracy improved slightly by using ensemble classifiers, these classifiers performed better in the minority classes as DT was giving 0 F1 score in "Analysis" class, but in XGBoost the score improved to 0.4. Thus, this indicates that the decision trees in XGBoost were able to learn from previous misclassification and improve on next generation.

As evident in figure 11, the cluster of binary data is distanced apart, hence by applying under-sampling the accuracy was further increased. But for multiclass classification, the data is overlapping as seen in figure 12, and hence by undersampling the majority neighbors of similar category are removed which decreases the accuracy.

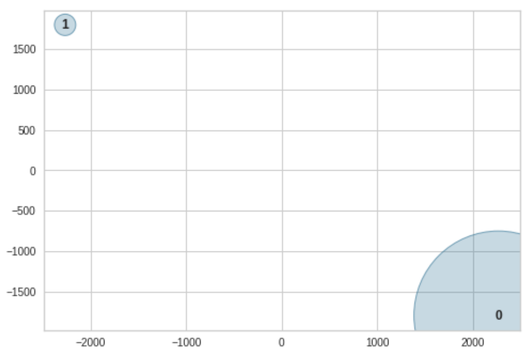


Figure 11. Inter-cluster distance map for binary class data

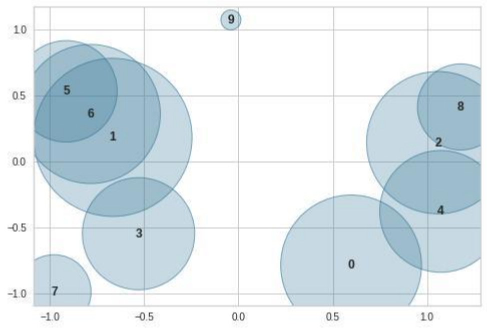


Figure 12. Inter-cluster distance map for multiclass data

6. Conclusion

6.1. Learning

We learnt about various preprocessing techniques. These include techniques related to under-sampling and oversampling, noise removal, category encoding, scaling and most importantly feature selection. We learnt about various machine learning classifiers and their advantages and disadvantages over one another. We learnt about how to fine tune hyperparameters in classifiers. We learnt about various ensemble learning methods like bagging, boosting, voting, etc.

6.2. Individual Contribution

Diptanshu: Literature Review, Preprocessing, Support Vector Machine, Ensemble Classifiers

Manavjeet: Literature Review, Preprocessing, K-Nearest Neighbour, Ensemble Classifiers

Rhythm: Literature Review, Preprocessing, Decision Tree, Ensemble Classifiers

References

[1] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid. Features dimensionality reduction approaches for machine learning based network intrusion detection. *Electronics*, 8(3):322, 2019.

[2] V. Dutta, M. Choraś, R. Kozik, and M. Pawlicki. Hybrid model for improving the classification effectiveness of network intrusion detection. In *Conference on Complex, Intelligent, and Software Intensive Systems*, pages 405–414. Springer, 2020.

[3] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50:102419, 2020.

[4] N. Moustafa and J. Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6, Nov 2015.

[5] Y. L. Mubarak Albarka Umar, Chen Zhanfang. Network intrusion detection using wrapper-based decision tree for feature selection. *Commun. ACM*, (1):13, Jan. 2020.