Building Machine Learning Intrusion Detection System with NSL-KDD Dataset

Hong Gong Lizhong Wang Jiajun Bao Puhan Li

ABSTRACT

This project aims to build an intrusion detection systems(IDS) and select the best binary classification model to determine whether an access might be malicious. We found a dataset containing five million server connection records in a simulated network environment from KDD. In terms of target variable, the original dataset contains a feature categorizing connections into either normal or 24 types of attack, and we plan to create a binary variable indicating if the connection is an attack as the target variable. Given that original KDD dataset is generated in 1999, there have been some analysis and paper published based on this dataset. After reading through some of the paper for reference and conducting our own baseline analysis, we decided to proceed forward with a certain type of machine learning models, including logistic regression, random forest, knn, decision tree and neural network. For each model, we conducted cross validation method as well as tuning certain hyperparameters to improve every model's predictive ability and reduce common issues such as overfitting. We ultimately found out that our baseline model random forest is able to achieve the highest AUC score.

BUSINESS UNDERSTANDING

Internet security is one of our biggest concerns and challenges in the modern world. According to the Internet Security Threat Report by Symantec Corporation, the number of cyberattacks increased by over 50% in 2019, and cyberattacks caused vast loss for business. “Organizations spend more than ever to deal with the costs and consequences of more sophisticated attacks— the average cost of cybercrime for an organization increased from US$1.4 million to US$13.0 million. Improving cybersecurity protection can decrease the cost of cybercrime and open up new revenue opportunities—we calculate a total value at risk of $US5.2 trillion globally over the next five years.” In conclusion, a successful IDS model could prevent companies from losing millions of dollars by creating a safer and more reliable firewall for their connections.

RELEVANT WORK ANALYSIS

Due to the popularity of the original KDD dataset, there have been some papers on IEEE or other journals analyzing this dataset. Some of them utilize self-organized maps combined with deep learning algorithms, trying to achieve a high accuracy. Some models can achieve an accuracy of as high as 99%. However, some other papers address that the high accuracy might due to the unbalancedness and redundancy of the original KDD dataset. We will address this issue in detail under our data understanding section. After thorough considerations and experiments of our dataset, we decided to try out supervised machine learning models, including Random Forest, Logistic Regression, knn, decision tree and neural networks combined with cross-validation techniques to search for the best model.

DATA UNDERSTANDING

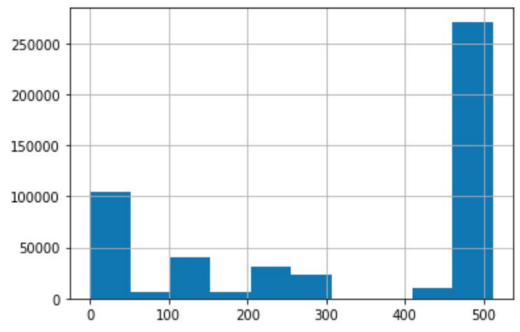
1. Data Source

The dataset we used is an improved version of KDD’99 Computer Network Intrusion Dataset. The original KDD’99 is a synthesized dataset created by MIT Lincoln Lab. The data contains nine weeks of raw connection log data of a simulated U.S. Air Force local-area network. Each instance is a connection to the server.

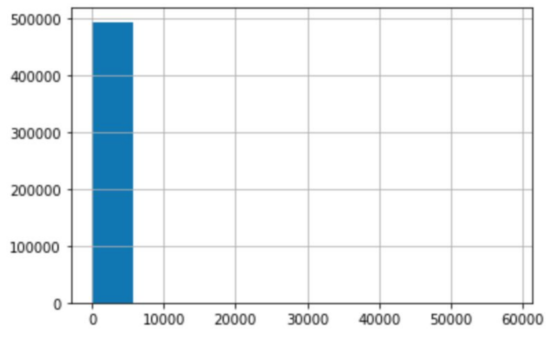
2. Abnormality

After trying out first few machine learning models on our KDD dataset, we surprisingly observed that most of the algorithms can achieve an accuracy rate of above 95%, with some of them scoring as high as over 99%. Although the result is more than ideal, we believe that there might be some problems in the datasets that cause such a peculiar accuracy. We further dig into more literature on this dataset combined with our own analysis and found out the issues with our original KDD dataset: 1. KDD data set has a huge number of redundant records, with about 78% and 75% of the records duplicated in the train and test set, respectively. 2. Almost every machine learning algorithm, no matter it is simple or complex, will achieve an accuracy between 86% and

100%, making the comparison among different models quite difficult. 3. Many of Internet connection attributes are categorical and highly skewed, making the standardization and normalization task almost unlikely.



Distribution of Variable 'Duration'



Distribution of Variable 'count'

3. Solution

Luckily, there is an improved version of the original KDD'99 dataset, called NSL-KDD, published by Mahbod Tavallaee's research team. NSL-KDD dataset tackles the duplicated-rows and low-level-difficulty problems by picking randomly sampled records from the 'successfulPrediction' value groups in such a way that the number of records selected from each group is inversely proportional to the percentage of records in the original successfulPrediction value groups. The resulting dataset has a much diverse average accuracy on different models. The entire dataset of nine weeks connection log has about 149K instances. The data generated in the first seven weeks is used as training dataset (125,973 instances, around 84.82% of total connections), and the data generated in the last 2 weeks is used as testing dataset (22,544 instances, around 15.17%of total connections). The data contains 41 predictors and one feature indicating the attack type. A connection is either normal or one of the 21 types of attack. Compared with the original KDD'99, NSL-KDD dataset is rather balanced: the training set has

46.54% of attack instances, and the attack instance ratio of the testing set is 56.92%. Furthermore, this dataset does not have any missing values or abnormal values, because the data is automatically generated by computer programs ------ no human error was involved in the data generation process. We believe that NSL-KDD dataset is representative of what a company can access when a user is trying to connect to the company's internal server. Therefore, this dataset can be regarded as a benchmark to represent the real-world network attack.

DATA PREPARATION/ DATA CLEANING

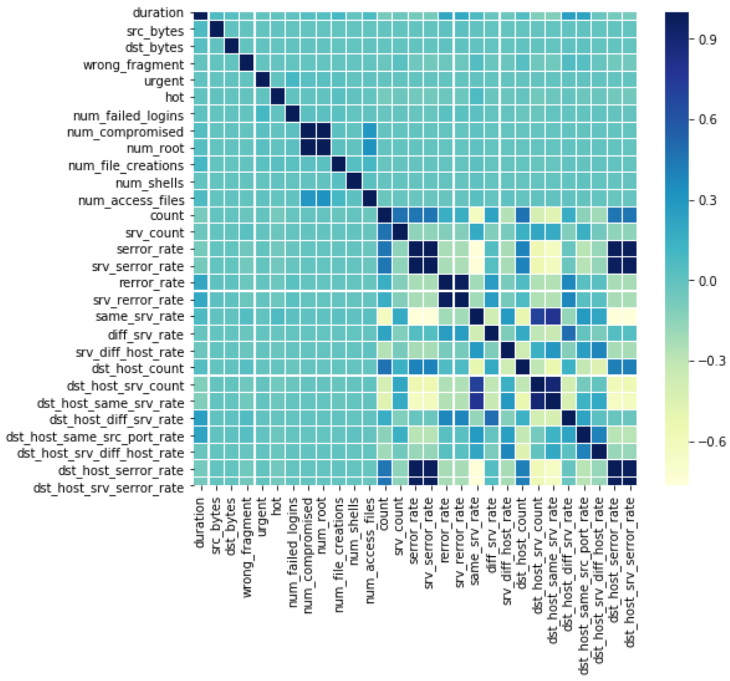
1.Target Variable

First, we created a binary target variable. The target variable is equal to 1 if the connection is an attack and 0 if the connection is normal. The reason that we did not differentiate the types of attack is that in business context, a company cares more about whether the connection is malicious rather than the types of the attack.

2. Feature Engineering

(1) First, we dropped two features that have 0 variance since they do not provide any prediction power. (2) We created dummy variables for all categorical features. (3) We mitigated the multicollinearity among the predictors by examining the correlation matrix (shown in the next page) of all non-indicator independent variables. We spotted that four sets of features have correlation coefficient greater than 0.9, and four sets of features have correlation coefficient greater than 0.7. It turns out that these set of features basically measure the same attribute of a connection. For example, one set of features-rerror_rate, srv_rerror_rate, dst_host_rerror_rate,

and dst_host_srv_rerror_rate ------ any pair of them have >0.9 correlation. They all measure the frequency of a connection being rejected by the server.



High correlation among the independent variables is problematic because the correlated features provide redundant information. Therefore, we kept only one feature for each set of highly correlated variables. If the features provide identical information, we kept the one with the highest variance. Otherwise, if the correlated features have different measurements, we used domaine knowledge to decide which one to keep. For instance, between num_root and num_compromised, we kept the former since it’s more interpretable.

3. Feature Selection

After we finished feature engineering, the dataset has 112 features. Out of the 112 predictors, 89 are indicator variables.We ranked all 112 features by their importance measured by normalized information gain computed with sklearn’s decision tree classifier. We dropped 62 features with 0 information gain accordingly.Finally, we have a dataset with 49 features, 125,973 training samples, and 22,544 testing samples.

MODEL & EVALUATION

1. Model Evaluation Metric

The common evaluation metrics used for comparing different classifiers are accuracy and area under ROC curve (AUC). Accuracy score is popular because it is easy to compute and intuitive to understand. However, accuracy score is not a comprehensive metric. In the case of intrusion detection, we care more about the recall because the consequences of missing an attack is more severe than the consequences of miss-classify a normal connection. However, recall can not be used as a metric to compare different models since we can always get perfect recall score by lowering the threshold. In contrast, AUC is a better metric. AUC is the area under the ROC curve, which is plotted with true positive rate (TPR or recall), against false positive rate (FPR). AUC measures the capability of how well a model could separate different classes, and it is invariant of the value of threshold and under scaling.

2. Baseline: Random Forest

First we want to quickly establish a baseline model to get a general picture. We selected random forest as our baseline model here. Random forest is robust and very unlikely to overfit over our training set due to its nature of combining the results of a large amount of trees. Furthermore, random forest does not require a heavy hyperparameter selection to give a fair result. With 500 trees and all default settings, we have our first model with an AUC of 0.957. After carefully selected the best parameters of max depth=20 and minimum number of records for a split=2, we have an AUC of 0.999 on the validation data, and an AUC of 0.963 on the testset. Due to the heavy time cost building each random forest model, We used a simple train-validation split instead of cross validation for random forest. Minimum number of records for a split is really small, we suspected that we might overfitted the model. However, both AUCs of validation and test dataset are high and really close, so the model is not overfitting.

3. Logistic Regression

Logistic regression is a simple classifier, yet it can produce accurate prediction if the relationship between features and target is linear. In addition, logistic regression is interpretable, robust, and has low computational costs. The out-of-box logistic regression yields a 0.861 AUC score. Then, we used five-folds cross validation to tune the regularization strength parameter, C. The highest mean AUC score is achieved when C=1e-5. Finally, we re-trained a logistic regression model with the selected best C and 'L2' regularization with all training data. The test AUC score is slightly improved to 0.866, and it does not deviate much from the train AUC score (0.95), so the model does not suffer from overfitting. However, the AUC score of logistic regression seems pretty low comparing to the AUC score of random forest (0.963). This might be because the problem is not linear in nature.

4. Neural Networks: Multilayer Perceptron (MLP)

MLP is a supervised neural network that is very commonly used on classification problems. With hidden layers and different number of neurons, it can capture non-linear relationships. MLP requires tuning a number of parameters, such as learning rate, number of neurons, and the number of hidden layers. First we picked 'adam' solver since we have a large dataset. While 'lbfgs' can converge faster and give better performance on a small dataset, 'Adam' usually performs better when training set has thousands or more records. Then we try to find the best parameters. There are more hyperparameters when we use the solver 'adam'. After a very time consuming search, we found that with epsilon = 1e-8, alpha = 0.0001, learning rate of 0.001, and two hidden layers of 25 and 20 neurons on each layer, we reached our highest AUC of 0.997 on our validation testset. We got an AUC of 0.902 on our actual testset with this tuned MLP model. There is a small difference between two AUCs, but we believe that it is in a normal range.

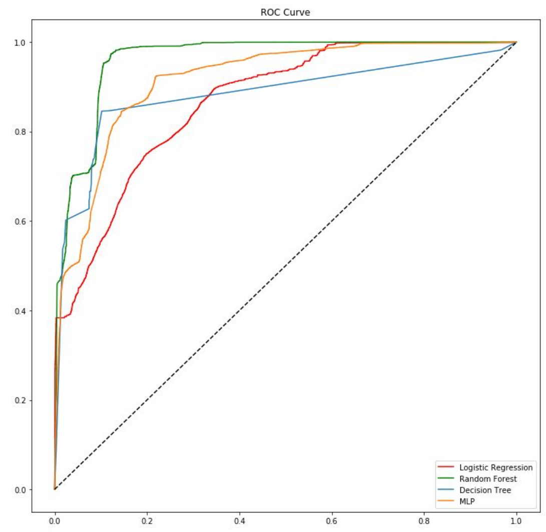
5. KNN (KNeighborsClassifier)

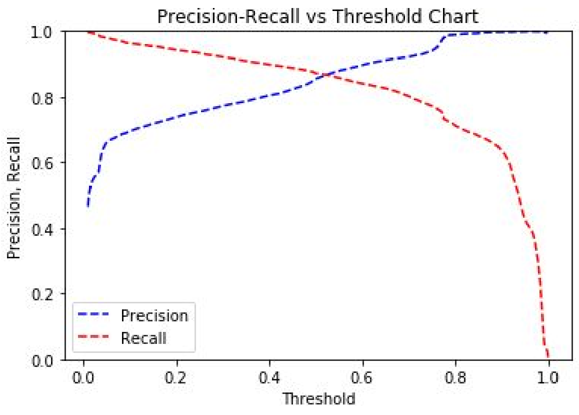
KNN is advantageous for many classification tasks as it neither has any prior assumptions about training dataset nor has an over complicated algorithm. We conducted GridSearchCV with KNN classifier and found out that with increasing K values, the performance of our model does improves. However, the average auc score of KNN model is below most of our other models. More importantly, given the facts that KNN algorithm is computationally expensive and demands high memory requirement, the process of each KNN model training is time-consuming and ineffective. Auc score given k value of 1 Hence, we prefer the other models such as LR and Random forest which provide a better efficiency-performance tradeoff to KNN.

6. Evaluation

AUC for Each Model

	Random Forest	Logistic Regression	MLP	KNN	Decision Tree
Validation	0.999	0.963	0.997	0.967	0.956
Test	0.963	0.866	0.902	0.858	0.915





From the table above, we can see that random forest over performed other models. It has better overall performance as the AUCs of random forest are generally higher than other models. Random forest also has the smallest difference of performance between validation and test. It means that the random forest model is less likely to overfit. The linear graph of AUC curves gives us a better view of each model's performance. Four curves are actually really close to each other. Logistic Regression and KNN here give the worst overall performance. We noticed that the MLP model ranked second on the graph, but it showed huge improvements while we were tuning the parameters. We only explored a fair range of possible values due to limited time and resources. There is a chance for MLP to have similar or better performance if we search deeper into the hyperparameters. Although random forest gives the best result, we might need to consider the time cost for a random forest to make a prediction. It takes more time for a random forest model with 500 trees to make a prediction. For a popular server, connect requests are usually very intense, and we do not want any connection delay.

Above graph shows a threshold analysis of logistic regression. Since we want our recall rate to be very high while the precision is not too bad, we will probably pick a threshold of 0.2. To our understanding, although recall rate is a very important component of the common standard to find the best model, recall rate will be heavily influenced by the threshold. As a result, we do not consider recall rate for any model selection.

DEPLOYMENT / CONCLUSION

Our model does a great job in building a network intrusion detector. Using several attributes, we generated a predictive model capable of distinguishing between 'bad' and 'good' connection. Although this data is a simulation of a military network environment, it is also useful in the commercial area. As we know, web security is important to any business. Web connection vulnerabilities could make a company suffer a lot. Thus, we can provide the web protection model to help web security team fight against scams, intrusions, and attacks better.

In order to deploy our model, we need to cooperate with security team. They will provide us data in several dimensions. For example, in a web-based company like Amazon, there are millions of requests per second. Even simple bugs in the code can result in private information being leaked, and bad people are out there trying to find ways to steal data. The better way to counter network intrusion is to detect them in advance.

The model is not static. We need to re-train it from time to time. Here is a strategy. We can set a threshold, for example, 0.01%, as the largest percentage allowed for intrusion. If the bad connection is beyond the limit, we need to retrain the model by combining all the old data with new data when the bad connection rates trend upwards.

Reference

· Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorban**,** *A Detailed Analysis of the KDD CUP 99 Data Set,* 2009

· L.Dhanabal1, Dr. S.P. Shantharajah2, *A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms,* 2015

· Preeti Aggarwal Sudhir Kumar Sharma, *Analysis of KDD Dataset Attributes Class wise For Intrusion Detection,* 2015

· ZhiSong PAN1 Hong LIAN2 GuYu HU GuiQiang NI, *An Integrated Model of Intrusion Detection Based on Neural Network and Expert System,* 2015

Appendix

Open source code for this project: https://github.com/hg1153-nyu-edu/DS_1001_PROJECT

NSL-KDD data: https://www.unb.ca/cic/datasets/nsl.html

KDD data: http://kdd.ics.uci.edu/databases/kddcup99/task.html

Contribution

Jiajun Bao:

Random Forest | Neural Networks Model

Model Performance Evaluation | Visualization | Proofreading

Hong Gong:

Relevant Work Analysis | Dataset Analysis and Comparison

Evaluation Metric Determining | KNN Model | Report Writing and Formatting

Lizhong Wang:

Data Preprocessing | Feature Engineering | Logistic Regression

Model Evaluation | Report Writing | Visualization

Puhan Li:

Attribute Analyzing | Feature Selection with Domain Knowledge

Report Writing | Visualization