Unusual Network activity Detector

Jagmeet Singh(MT2018042), Karanbir Singh(MT2018047)
and Sonveer Tomar(MT2018118)

December 7, 2018

1 Introduction

Intrusion prevention technologies have been built in the past decade to detect and prevent various attacks, but the complexity of attack are also increasing which are not easily detectable on such intrusion detection mechanism. The security mechanism can be implemented using an Intrusion Detection System (IDS) which can be describe as a collection of software or hardware device able to collect, analyse and detect any unwanted, suspicious or malicious traffic either on a particular computer host or network. Therefore to achieve its task, an IDS should use some statistical or mathematical method to read and interpret the information it collects and subsequently reports any malicious activity to the network administrator.

2 Problem Statement

We want to Predict whether a particular Request on a System is Denial of Service attack or not, the tasks involved are the following.

1. Download and pre-process the NSL-KDD dataset.
2. Training various classifier using various Model.
3. Predicting the whether there is intrusion or not.
4. Measuring the performance and accuracy of Model.

3 Metrics

3.1 Accuracy

Accuracy is common metric for binary classifiers, it considers both true positives and true negatives with equal weight.

$$Accuracy = \frac{Truepositive + Truenegatives}{datasetsize} \tag{1}$$

This metric is used when evaluating the classifier because the false negatives and false positive both have impact on whether the model has detect intrusion or not.

- False Negative result when there is attack request and classifier predict that it is not attack request.
- False Positive result when there is attack request and classifier unclassified that request into the non-attack request.

3.2 Confusion Matrix

Confusion Matrix is a table used for measuring the performance of Various Classification Model on a set of test data for which true values are known. Confusion matrix is 2X2 Matix where two Row represent the Actual value and 2 Column tells the Predict value.There are 4 different cell defining the number of test data classified value. The 4 different cell are.

- True Positive: These are the request which are predicated as denial of service request and they are actually denial of service request.
- True Negative: These are the request classified as the normal request and they are actually normal request
- False Positive: We predict the request as denial of service request but it not actually denial of service request
- False Negative :We predict the request as normal request but it is denial of service request.

4 Approach

As we have to predict whether a particular request is Denial of service attack request or not, we use different binary classification models like Logistic Regression, Decision Tree and Naïve bayes classifier.

To decide which model to use in predicting the dataset we used confusion matrix and accuracy as evaluation criteria.

5 Analysis

For intrusion detection system to be intelligent enough we need an effective dataset to trained them for detecting various attack. NSL KDD dataset contains information about various based on different features like duration, type of protocol, type of service, Flag, Src_bytes, Dst_bytes, Urgent, No_ failed login, Count etc. The reason behind use of this dataset is

- Training Dataset doesn't contain redundant records, so our classifier will be unbiased towards more frequent records.
- Test dataset also contain unique records therefore our classifier performance will not be baised by the techniques which have better detection rates on the frequent records.

NSL KDD dataset contains two main files as KDDTrain+.TXT and KDDTest+.TXT both of these contains information about various classes of attacks, as we are interested in predicting whether a attack is DoS or not so we will look feature vectors related to the DoS attack.

The labels in the NSL KDD datatset are divided into 4 categories which represent the attack class and one as normal traffic:

- Denial of Service.
- Probeh.
- Remote to local.
- User to root

6 Algorithms and Technique

In this work we have used 3 different type of classifier :

1. Logistic regression
2. Decision Tree
3. Naïve bayes classifier

6.1 Logistic regression

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X. Logistic Regression is a Machine Learning classification algorithm that is used to from sklearn.linear_model import Logistic Regression

```
class sklearn.linear_model.LogisticRegression
(
    penalty='l2',
    dual=False,
    tol=0.0001,
    C=1.0,
    fit_intercept=True,
    intercept_scaling=1,
    class_weight=None,
    random_state=None,
    solver='warn',
    max_iter=100,
    multi_class='warn',
    verbose=0,
    warm_start=False,
    n_jobs=None
)
```

6.2 Decision Tree

Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables. Scikit learn contain class which implement decision tree classifier. We can regulate the performance of decision tree by tuning various parameter.

```
class sklearn.tree.DecisionTreeClassifier
(
    criterion='gini',
    splitter='best',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features=None,
    random_state=None,
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    min_impurity_split=None,
    class_weight=None,
    presort=False
)
```

6.3 Naïve Bayes classifier

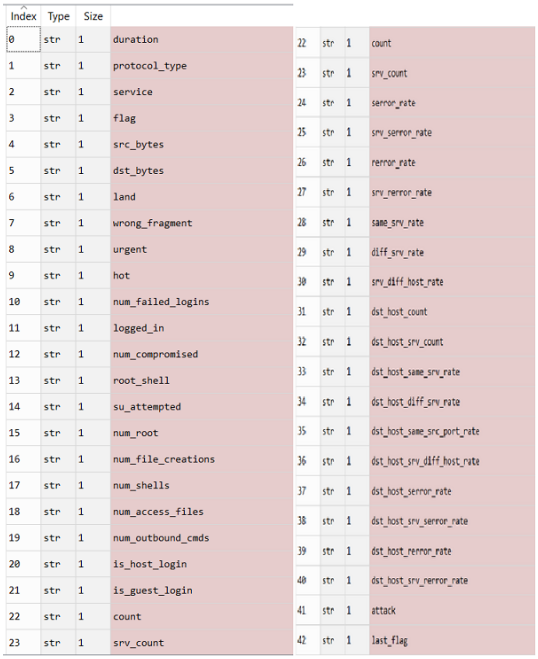
It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a feature in a class is unrelated to the presence of any other feature. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

7 Methodology

7.1 Data Preprocessing

Data preprocessing is done in various steps

1. We created a list datacols to give feature name to our dataset since original dataset doesn't contains Training label, it contains only value.Figure [1]
2. Loading the Training Dataset and Test Dataset Figure [2], Figure [3]
3. First we drop column which is not relevant in our work.In NSL KDD dataset there are column num_outbound_cmds, attack which is drop because it contains all zero value. Figure 4



Index	Type	Size		Index	Type	Size	
0	str	1	duration	22	str	1	count
1	str	1	protocol_type	23	str	1	srv_count
2	str	1	service	24	str	1	serror_rate
3	str	1	flag	25	str	1	srv_serror_rate
4	str	1	src_bytes	26	str	1	rerror_rate
5	str	1	dst_bytes	27	str	1	srv_rerror_rate
6	str	1	land	28	str	1	same_srv_rate
7	str	1	wrong_fragment	29	str	1	diff_srv_rate
8	str	1	urgent	30	str	1	srv_diff_host_rate
9	str	1	hot	31	str	1	dst_host_count
10	str	1	num_failed_logins	32	str	1	dst_host_srv_count
11	str	1	logged_in	33	str	1	dst_host_same_srv_rate
12	str	1	num_compromised	34	str	1	dst_host_diff_srv_rate
13	str	1	root_shell	35	str	1	dst_host_same_src_port_rate
14	str	1	su_attempted	36	str	1	dst_host_srv_diff_host_rate
15	str	1	num_root	37	str	1	dst_host_serror_rate
16	str	1	num_file_creations	38	str	1	dst_host_srv_serror_rate
17	str	1	num_shells	39	str	1	dst_host_rerror_rate
18	str	1	num_access_files	40	str	1	dst_host_srv_rerror_rate
19	str	1	num_outbound_cmds	41	str	1	attack
20	str	1	is_host_login	42	str	1	last_flag
21	str	1	is_guest_login				

Figure 1: Datacols list is used to give feature label to dataset. It contains 42 different label names.

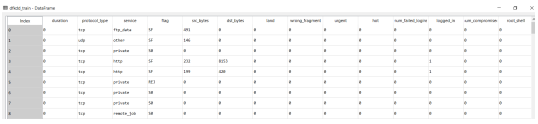


Figure 2: Dataframe dfkdd_Train contains the dataset value.

4. Scaling of data is done using StandardScaler() function.We applied standar Scaler on those features having Float and Integer value.

5. One hot encoder is fit on the feature column Service of dataset.

8 Implementation

The implementation Process can be split into two main stages.

1. Training the classifier.

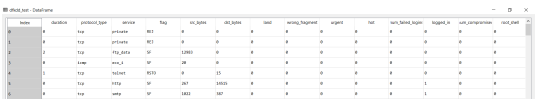


Figure 3: Dataframe dfkdd_Test contains the dataset value for testing the our Model.

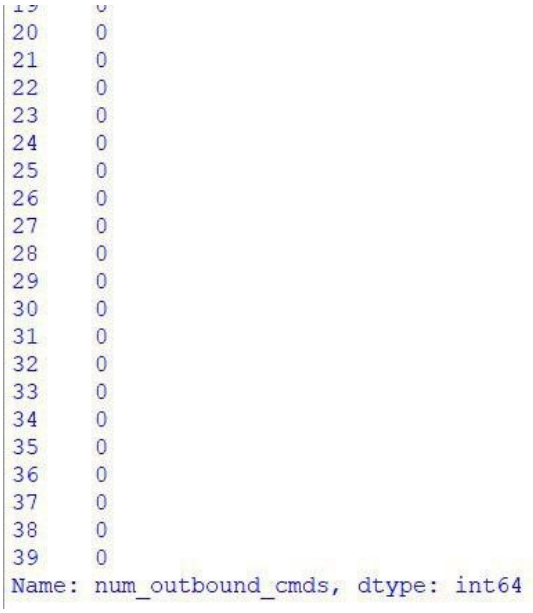


Figure 4: num_outbound_cmds Column is Drop as it contains all 0 value which is not relevant for Training the model.

2. Running the cross validation test for measuring the accuracy. Figure [5]

During the first stage, the classifier is trained on the Preprocessed Training data. We used the Anaconda Navigator Spider for doing the implementation.

1. Load the training and test data into dfkdd_train and dfkdd_test dataframe and preprocessing them as done in previous stage.
2. Feature selection: we have selected the feature that are relevant to the denial of service activity like, selected_features=['duration', 'src_bytes', 'dst_bytes', 'logged_in', 'dst_host_srv_count', 'dst_host_same_srv_rate', 'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate', 'dst_host_serror_rate', 'service']
3. Plotting the frequency of various data feature and Attack class Distribution. Figure [6] Figure [7]
4. We have implemented the training model function for 3 different classifier.
   - Naïve bayes classifier : BNB_Classifier()
   - Logistic Regression: LGR_Classifier()
   - Decision tree classifier: DTC_Classifier()
5. Define the accuracy and confusion Matrix Figure [8]
6. Selecting the best model among the 3 different model used.

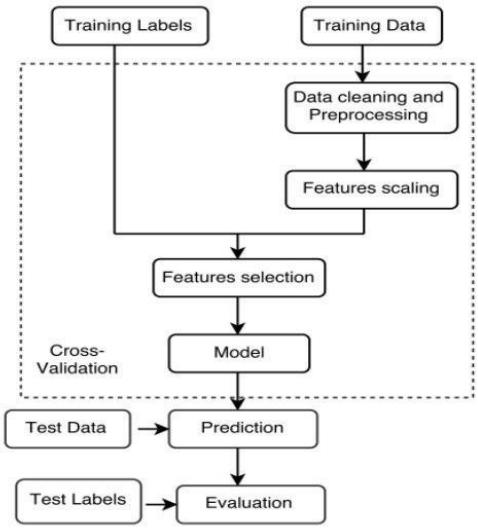


Figure 5: Methodology used in the Implemenation of various Model.

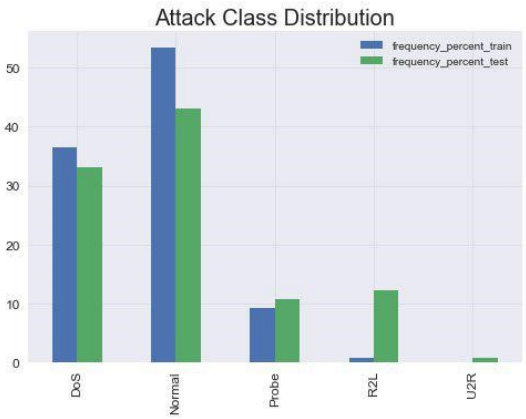


Figure 6: Various Attack Distribution.

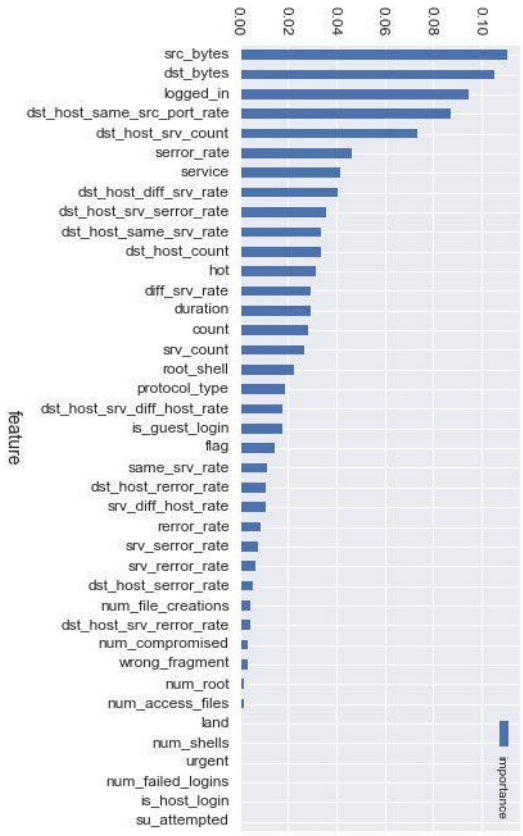


Figure 7: Plotting Feature and there frequency.

```
Cross Validation Mean Score:
 0.9756619060627969
Naive Baye Classifier
Model Accuracy:
 0.8204321742675753
Confusion matrix:
 [[5331 2127]
 [ 956 8755]]
Decision Tree Classifier
Model Accuracy:
 0.6610751936629973
Confusion matrix:
 [[2314 5144]
 [ 675 9036]]
LogisticRegression
Model Accuracy:
 0.8025511095579242
Confusion matrix:
 [[5122 2336]
 [1054 8657]]
```

Figure 8: Model Accuracy and Confusion Matrix Score.

9 Results : Model Evaluation and Validation

During the implementation of various model Validation set is used to evaluate the 3 different classifier used. Cross validation mean score produced is 0.9756. Then we produce the Model accuracy and confusion matrix based on the score of the we selected the best fit model our Problem. see Table[1]

Model	Model Accuracy	Confusion Matrix
Naïve Bayes classifier	0.82043	[[5331,2127] [956,8755]]
Decision Tree Classifier	0.66107	[[2314,5144] [675,9036]]
Logistic Regression	0.80255	[[5122,2336] [1054,8657]]

Table 1: Comparison of models.

The model accuracy of the Naïve Bayes classifier is 0.82 which is better as compared to the Decision Tree (0.66) and Logistic Classifier (0.80).