CRYPTOGRAPHY AND NETWORK SECURITY

PROJECT REPORT

"PERFORMANCE EVALUATION OF INTRUSION DETECTION SYSTEMS USING MACHINE LEARNING TECHNIQUE IN APACHE SPARK"

COURSE CODE: ITE 5003

SLOT: A1

FACULTY: DR.SUMAIYA THASEEN

TEAM MEMBERS:

AARTHI N (17BCE0893)

MOTURU NIHARIKA (17BCE0951)

PERFORMANCE EVALUATION OF INTRUSION DETECTION SYSTEMS USING MACHINE LEARNING TECHNIQUES IN APACHE SPARK

1.Abstract

In recent times, in the context of network communications, network intrusion has become a major problem. To overcome this intrusion many intrusion detection systems have been developed aiming to identify unauthorized access and malicious attacks in the network environment. Among them Machine Learning is one of the efficient analysis framework to detect threats in the network traffic flow. Our project evaluates the performance of Intrusion detection system based on Logistic regression ,Naïve Bayes, Decision Tree and Random Forest algorithms of Machine Learning using Apache Spark, a big data processing tool .UNSW-NB15,a recently updated dataset is used to carry out the evaluation. The overall performance is evaluated in terms of detection accuracy ,building time(or) training time and prediction time.

2. Introduction

Intrusion is defined as the act of wrongfully entering upon, seizing, or taking possession of the property of another. In general, an intrusion detection system (IDS) is a device or software application that monitors a network or systems for malicious activity or policy violations. When any malicious intrusion or attack is manifest in a network, computers and information systems may suffer serious consequences when defined computer security policies are violated. Several security strategies have been implemented over the years to safeguard networks. Basic packet filter i.e. firewall is not that accurate in the network environment . Firewall working in conjunction with a intrusion detection may provide an improved and safer network.

In the literature, some IDSs are based on single-classification techniques, while others (hybrid/ensemble) include more than one classification techniques. IDS are categorized by misuse and anomaly strategies. The misuse approach seeks known attacks referred to as attack signatures, while the anomaly approach is based on normalcy models, where any significant deviation from these reference models indicates a potential threat. However, both the approaches suffer from a number of weaknesses. Misuse detection requires frequent updates of the signatures to ensure ample detection, while anomaly detection presents a high false positive rate. Thus, the current challenge is to deal these two shortcomings toward the provision of a solution with characteristics such as superior accuracy with low false positive rates.

The main purpose of this paper is to address the issue of low accuracy and prediction time in IDS. We employed multiple classifiers with different learning paradigms to evaluate different classifier model. The organization of this paper is as follows: The majority of Section 3 discusses the background and related works. Section 4 presents the various classification techniques and dataset employed in this work, and Section 5 provides experimental results and a discussion on the findings. Finally, Section 6 concludes the paper.

3.Related Work

Many research papers describing the use of machine learning methods for anomaly detection have reported the attainment of a very high detection rate of 98%, while the false positive rate was less than 1% [8]. However, when surveying state-of-art IDS solutions and commercial tools, there is no evidence for the utilization of anomaly detection approaches, presumably because experts still consider anomaly detection to be an immature technology. To discover the reason for this contrast [9] concentrated studies on the details of the research done in anomaly detection were considered from different angles, for example learning and detection approaches, training data sets, testing data sets, and evaluation methods. These studies indicated that there are some issues in the KDDCUP'99 data set [10], which is broadly used as one of the rare publicly accessible data sets for network based anomaly detection systems. The primary critical deficiency in the KDD data set was the enormous number of redundant records. Approximately 78% and 75% of the train and test set records, respectively, are duplicated in KDD [9]. These redundant records in the train set cause learning algorithms to be one sided toward the more frequent records, and prevent it from learning infrequent records, which are typically more damaging. On the other hand, the presence of these redundant records in the test set causes the evaluation results to be biased by the methods that have better detection rates on the frequent records [9]. New NSL-KDD datasets were generated in order to solve the issues of the original KDD dataset [9], with new train and test sets (KDDTrain+ and KDDTest) consisting of selected records of the complete KDD data set. This new version of the KDD data set (NSL-KDD) is publicly available for researchers [11].In 2015, Moustafa et al. generated a new dataset UNSW-15 in order to counter the unavailability of network benchmark data set challenges [5]. This data set contained a fusion of actual modern normal network traffic and contemporary synthesized attack activities thereof. The authors were able to find only one paper that described the use of different existing machine learning classifiers to evaluate complexities in terms of accuracy and false positive rate algorithms on the UNSW-15 dataset [4]. The results indicated that the Decision Tree classifier accomplished the highest accuracy of 85.56% and the lowest FAR at 15.78% compared to other classifiers (e.g., Logistic Regression, Naive Bayes, Artificial Neural Network, Expectation-Maximization Clustering). However, the UNSW-NB15 dataset was considered as complex and may be used to evaluate existing and novel methods of Network Intrusion Detection Systems [6].

4.Background

4.1 UNSW-NB15 Data Set

As mentioned earlier, the UNSW-NB15 data set was created at the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) using the AXIA Perfect Storm tool to create a hybrid of modern normal and abnormal network traffic . This data set included 49 features and nine attack families: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shell code, and Worms. The UNSW-NB15 dataset was decomposed into

two partitions the training and the testing sets, including 175,340 and 82,000 records respectively. The two partitions are available online for research purposes.

4.2 Apache Spark

Apache Spark is a cluster computing platform designed to be fast [21], Spark extends the popular Map Reduce model to efficiently support more types of computations, including interactive queries and stream processing. Speed is important in processing large datasets, as it means the difference between exploring data interactively and waiting minutes or hours. One of the main features Spark offers for speed is the ability to run computations in memory, but the system is also more efficient than Map Reduce for complex applications running on disk. Spark is also designed to cover a wide range of workloads that previously required separate distributed systems, including batch applications, iterative algorithms, interactive queries, and streaming. By supporting these workloads in the same engine, Spark makes it easy and inexpensive to combine different processing types, which is often necessary in production data analysis pipelines. In addition, it reduces the management burden of maintaining separate tools.

5. Proposed Models:

5.1Random forest

The random forest is an ensemble learning method for unpruned classification, regression or other tasks, that consists of building multiple decision trees [18]. A bootstrap sample of the original data subsets is utilized to construct multiple decision trees (Forest). Each tree in the forest gives a decision about the class of the new object that needs to be classified. The class which obtains the most votes for the object is selected by the forest.

5.2 Decision tree

Decision trees are developed by recursive partitioning . A univariate split is selected for the root of the tree according to some criterion, and the process repeats recursively. This process known as pruning, which decreases the tree size, is performed once a full tree has been built . The most popular representative of decision trees is C4.5.A bootstrap sample of the original data subsets is utilized to construct multiple decision trees (Forest). Each tree in the forest gives a decision about the class of the new object that needs to be classified. The class which obtains the most votes for the object is selected by the forest.

5**.3 Naïve Bayes**

The Naive Bayes algorithm is an intuitive method that uses Bayes rule to compute the probabilities of each attribute belonging to each class to make a prediction .It simplifies the calculation of probabilities by assuming that the attributes are independent, given the label

of all other attributes. Numerous studies have shown that Naive Bayes algorithms were unexpectedly accurate for classification tasks, albeit only with small databases. For some larger databases, the accuracy of decision trees was better than Naive Bayes.

5.4 Logistic Regression

Logistic regression is a statistical method for analysing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

5.5 Gradient Boosted Trees

Gradient boosting is a machine learning technique
for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. Gradient boosting is typically used with decision trees (especially CART trees) of a fixed size as base learners. For this special case, Friedman proposes a modification to gradient boosting method which improves the quality of fit of each base learner.

6.WORKING OF MODELS

In our project, five well-known machine learning algorithms are used, namely Naïve Bayes, Logistic Regression ,Decision Tree, and Random Forest are used for performance evaluation. The classification measures used are :

TP, TN, FP and FN.

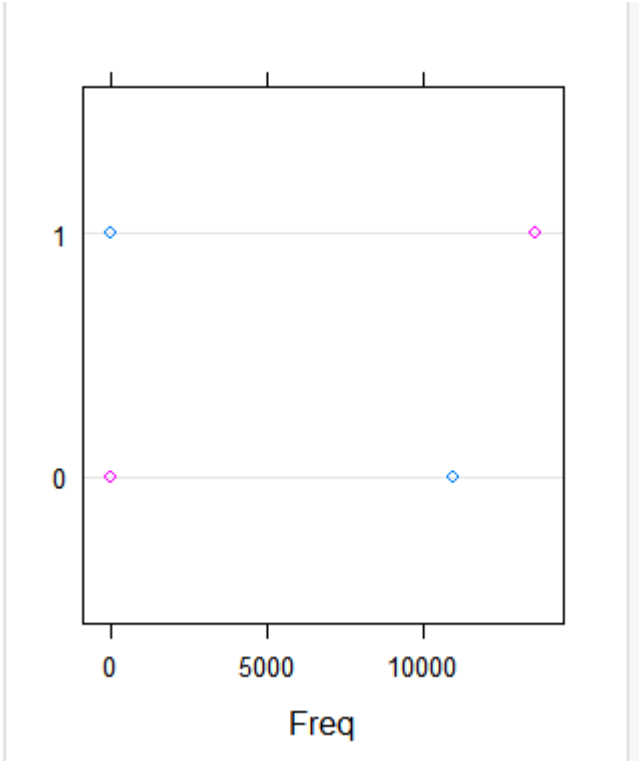
1. TP (true positive) is the number of correctly classified attacks.
2. TN (true negative) is the number of correctly classified normal records.
3. FP (false positive) is the number of misclassified attacks.
4. FN (false negative) denotes the number of misclassified normal records.

6.RESULTS:

6.1 SNAPSHOTS: (WITH CONFUSION MATRIX & DOT PLOTS)

RANDOM FOREST

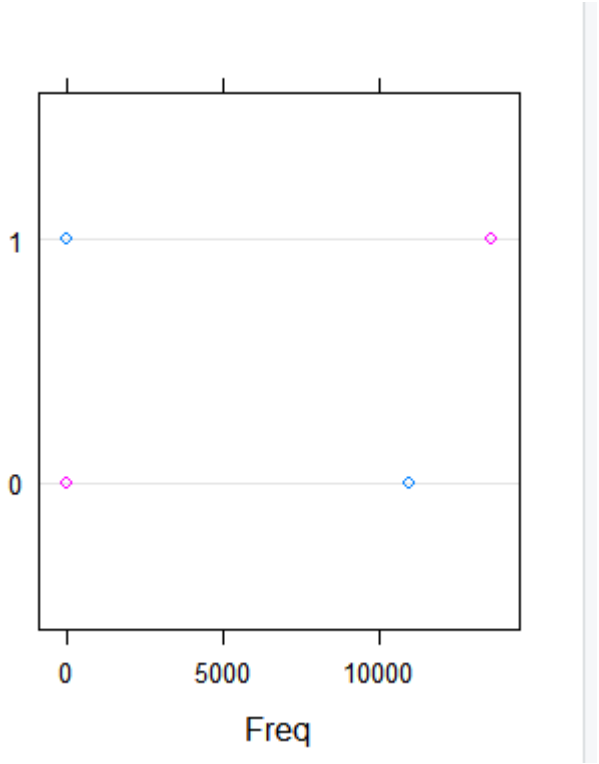
```
> partitions <- UNSW_training_tbl %>%
+ sdf_partition(training=0.7,test=0.3,seed=1111)
> UNSW_trainings<-partitions$training
> UNSW_testing<-partitions$test
> rf_model<-UNSW_trainings %>%
+ ml_random_forest(label ~ ., type="classification")
> pred <- ml_predict(rf_model)
> pred <- ml_predict(rf_model,UNSW_testing)
> ml_multiclass_classification_evaluator(pred)
[1] 0.9998781
> rf_test<-collect(pred)
>
>
> source('C:/Users/Aarthi N/Desktop/semester4/cryptography/project/tri.R')
>
>
>
> rf_test<-collect(pred)
>
> rf_final <- table(rf_test$prediction,rf_test$label)

> confusionMatrix(rf_final)
Confusion Matrix and Statistics


       0     1
  0 11003     0
  1     3 13601

               Accuracy : 0.9999
                 95% CI : (0.9996, 1)
    No Information Rate : 0.5527
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9998
 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9997
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9998
             Prevalence : 0.4473
         Detection Rate : 0.4471
   Detection Prevalence : 0.4471
      Balanced Accuracy : 0.9999

       'Positive' Class : 0


> dotplot(rf_final)
> print(rf_final)

       0     1
  0 11003     0
```



DECISION TREE

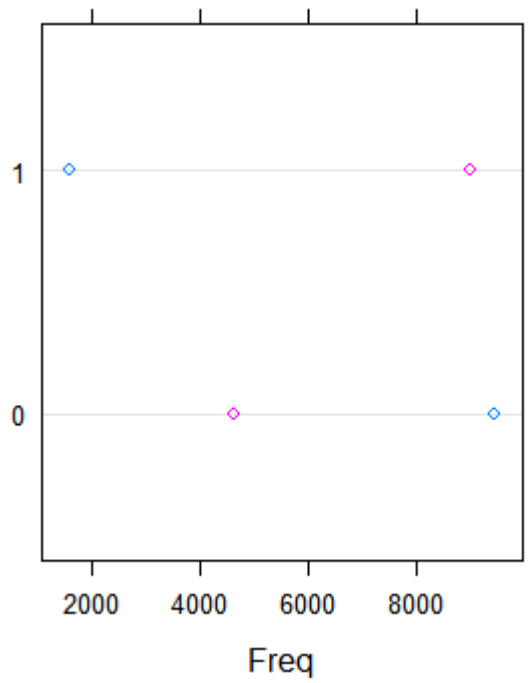
```
  1     3 13601
> decision_tree_model<-UNSW_trainings %>%
+ ml_decision_tree(label ~ ., type="classification")
> pred <- ml_predict(decision_tree_model)
> pred <- ml_predict(decision_tree_model,UNSW_testing)
> ml_multiclass_classification_evaluator(pred)
[1] 1
> decision_tree_test<-collect(pred)
> decision_tree_final <- table(decision_tree_test$prediction,decision_tree_test$label)
> confusionMatrix(decision_tree_final)
Confusion Matrix and Statistics


       0     1
  0 11006     0
  1     0 13601

               Accuracy : 1
                 95% CI : (0.9999, 1)
    No Information Rate : 0.5527
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.4473
         Detection Rate : 0.4473
   Detection Prevalence : 0.4473
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```

```
> print(decision_tree_final)

        0     1
  0 11006     0
  1     0 13601
> dotplot(decision_tree_final)
>
```



NAÏVE BAYES

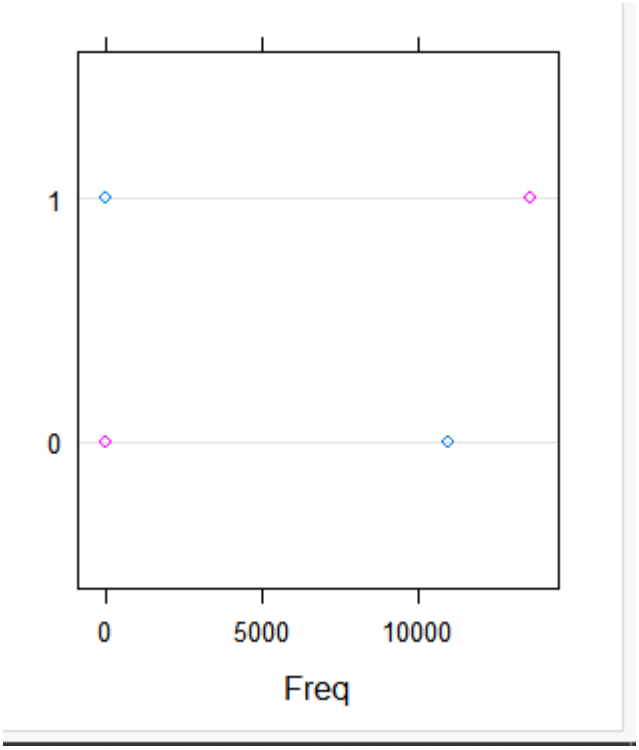
```
> naive_bayes_model<-UNSW_trainings %>%
+ ml_naive_bayes(label ~ ., type="classification")
Warning: Some components of ... were not used: type
> pred <- ml_predict(naive_bayes_model)
> pred <- ml_predict(naive_bayes_model,UNSW_testing)
> ml_multiclass_classification_evaluator(pred)
[1] 0.7466615
> naive_bayes_test<-collect(pred)
> naive_bayes_final <- table(naive_bayes_test$prediction,naive_bayes_test$label)
> confusionMatrix(naive_bayes_final)
Confusion Matrix and Statistics


      0    1
  0 9416 4630
  1 1590 8971

               Accuracy : 0.7472
                 95% CI : (0.7417, 0.7526)
    No Information Rate : 0.5527
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5019
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8555
            Specificity : 0.6596
         Pos Pred Value : 0.6704
         Neg Pred Value : 0.8494
             Prevalence : 0.4473
         Detection Rate : 0.3827
   Detection Prevalence : 0.5708
      Balanced Accuracy : 0.7576

       'Positive' Class : 0


> print(naive_bayes_final)

      0    1
  0 9416 4630
  1 1590 8971
> dotplot(naive_bayes_final)
>
```



GRADIENT BOOST ALGORITHM

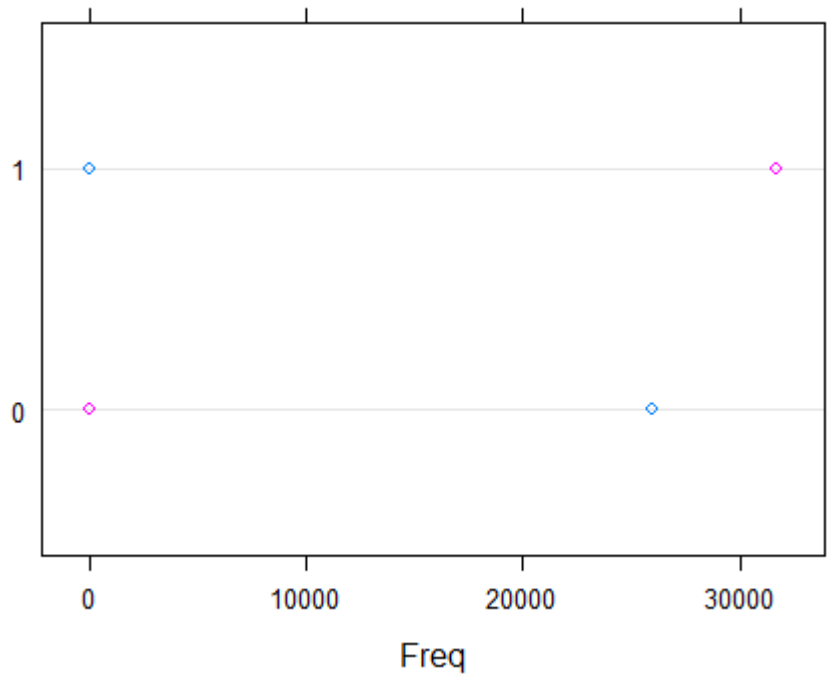
```
> gbost_model<-UNSW_trainings %>%
+ ml_gradient_boosted_trees(label ~ ., type="classification")
> pred <- ml_predict(gbost_model)
> pred <- ml_predict(gbost_model,UNSW_testing)
> ml_multiclass_classification_evaluator(pred)
[1] 1
> gbost_test<-collect(pred)
> gbost_final <- table(gbost_test$prediction,gbost_test$label)
> confusionMatrix(gbost_final)
Confusion Matrix and Statistics


       0     1
  0 11006     0
  1     0 13601

               Accuracy : 1
                 95% CI : (0.9999, 1)
    No Information Rate : 0.5527
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.4473
         Detection Rate : 0.4473
   Detection Prevalence : 0.4473
      Balanced Accuracy : 1.0000

       'Positive' Class : 0

> |
```

```
> print(gbost_final)

        0     1
  0 11006     0
  1     0 13601
> dotplot(gbost_final)
>
```



LOGISTIC REGRESSION

```
Confusion Matrix and Statistics


       0     1
  0 25994     0
  1     0 31731

               Accuracy : 1
                 95% CI : (0.9999, 1)
    No Information Rate : 0.5497
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.4503
         Detection Rate : 0.4503
   Detection Prevalence : 0.4503
      Balanced Accuracy : 1.0000

       'Positive' Class : 0

> print(logistics_final)

       0     1
  0 25994     0
  1     0 31731
> dotplot(logistics_final)
>
```



Correlation And chi-square test without feature selection

Random forest:

```
> cor(rf_analysis$label,rf_analysis$prediction)
[1] 0.9819465
> chisq.test(ans)

        Pearson's Chi-squared test with Yates' continuity correction

data:  ans
X-squared = 23582, df = 1, p-value < 2.2e-16
```

Logistic Regression:

```
> cor(log_anal$label,log_anal$prediction)
[1] 0.9998347
> log_table<-table(log_anal$prediction,log_anal$label)
> chisq.test(log_table)

        Pearson's Chi-squared test with Yates' continuity correction

data:  log_table
X-squared = 24449, df = 1, p-value < 2.2e-16
```

Decision Tree:

```
> cor(dt_anal$label,dt_anal$prediction)
[1] 1
> dt_table<-table(dt_anal$prediction,dt_anal$label)
> chisq.test(dt_table)

        Pearson's Chi-squared test with Yates' continuity correction

data:  dt_table
X-squared = 24457, df = 1, p-value < 2.2e-16
```

Naïve Bayes:

```
> cor(nb_anal$label,nb_anal$prediction)
[1] 0.5148501
> nb_table<-table()
Error in table() : nothing to tabulate
> nb_table<-table(nb_anal$prediction,nb_anal$label)
> chisq.test(nb_table)

        Pearson's Chi-squared test with Yates' continuity correction

data:  nb_table
X-squared = 6481.8, df = 1, p-value < 2.2e-16
```

Gradient Boosted Trees:

```
> cor(gbt_anal$label,gbt_anal$prediction)
[1] 1
> gbt_table<-table(gbt_anal$prediction,gbt_anal$label)
> chisq.test(gbt_table)

        Pearson's Chi-squared test with Yates' continuity correction

data:  gbt_table
X-squared = 24457, df = 1, p-value < 2.2e-16
```

STACKING/BLENDING without feature selection

Considering 3 algorithms – NAÏVE BAYES + RANDOM FOREST+ GBM

With NAÏVE BAYES as the final layer of stack

```
Confusion Matrix and Statistics

          Reference
Prediction stayed left
    stayed    766 1004
    left     2662   67

               Accuracy : 0.1852
                 95% CI : (0.1739, 0.1968)
    No Information Rate : 0.7619
    P-Value [Acc > NIR] : 1

                  Kappa : -0.466
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.22345
            Specificity : 0.06256
         Pos Pred Value : 0.43277
         Neg Pred Value : 0.02455
             Prevalence : 0.76195
         Detection Rate : 0.17026
   Detection Prevalence : 0.39342
      Balanced Accuracy : 0.14301

       'Positive' Class : stayed
```

GBM+ RANDOM FOREST+ DECISION TREE

With GBM as the final layer of stack

```
Confusion Matrix and Statistics

          Reference
Prediction stayed left
    stayed    530  903
    left     2898  168

               Accuracy : 0.1551
                 95% CI : (0.1447, 0.1661)
    No Information Rate : 0.7619
    P-Value [Acc > NIR] : 1

                  Kappa : -0.4197
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.15461
            Specificity : 0.15686
         Pos Pred Value : 0.36985
         Neg Pred Value : 0.05479
             Prevalence : 0.76195
         Detection Rate : 0.11780
   Detection Prevalence : 0.31852
      Balanced Accuracy : 0.15574

       'Positive' Class : stayed
```

DECISION TREE+ RANDOM FOREST+LOGISTIC REGRESSION

With DECISION TREE as the final layer of stack

```
Confusion Matrix and Statistics

          Reference
Prediction stayed left
    stayed   2441   35
    left      987 1036

               Accuracy : 0.7728
                 95% CI : (0.7603, 0.785)
    No Information Rate : 0.7619
    P-Value [Acc > NIR] : 0.04422

                  Kappa : 0.5204
 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.7121
            Specificity : 0.9673
         Pos Pred Value : 0.9859
         Neg Pred Value : 0.5121
             Prevalence : 0.7619
         Detection Rate : 0.5426
   Detection Prevalence : 0.5503
      Balanced Accuracy : 0.8397

       'Positive' Class : stayed
```

Feature Selection in UNSW-NB15 data set:

- **dur**
- **proto**
- **state**
- **sload**
- **dload**
- **sloss**
- **dloss**
- **synack**
- **ackdat**
- **is_ftp_login**
- **attack_cat**

Confusion Matrix:

Gradient-boosted trees:

```
> confusionMatrix(gbt_analysis)
Confusion Matrix and Statistics


       0     1
  0 10966     0
  1     0 13495

               Accuracy : 1
                 95% CI : (0.9998, 1)
    No Information Rate : 0.5517
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.4483
         Detection Rate : 0.4483
   Detection Prevalence : 0.4483
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```

Naïve Bayes:

```
> confusionMatrix(nb_analysis)
Confusion Matrix and Statistics


      0    1
  0 9550 4614
  1 1416 8881

               Accuracy : 0.7535
                 95% CI : (0.748, 0.7589)
    No Information Rate : 0.5517
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5149
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8709
            Specificity : 0.6581
         Pos Pred Value : 0.6742
         Neg Pred Value : 0.8625
             Prevalence : 0.4483
         Detection Rate : 0.3904
   Detection Prevalence : 0.5790
      Balanced Accuracy : 0.7645

       'Positive' Class : 0
```

Decision tree:

```
> confusionMatrix(dt_analysis)
Confusion Matrix and Statistics


       0     1
  0 10966     0
  1     0 13495

               Accuracy : 1
                 95% CI : (0.9998, 1)
    No Information Rate : 0.5517
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.4483
         Detection Rate : 0.4483
   Detection Prevalence : 0.4483
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```

Logistic Regression:

```
> confusionMatrix(log_analysis)
Confusion Matrix and Statistics


       0     1
  0 10964     0
  1     2 13495

               Accuracy : 0.9999
                 95% CI : (0.9997, 1)
    No Information Rate : 0.5517
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9998
 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.9998
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9999
             Prevalence : 0.4483
         Detection Rate : 0.4482
   Detection Prevalence : 0.4482
      Balanced Accuracy : 0.9999

       'Positive' Class : 0
```

Random Forest:

```
Confusion Matrix and Statistics


       0     1
  0 10746     0
  1   220 13495

               Accuracy : 0.991
                 95% CI : (0.9897, 0.9922)
    No Information Rate : 0.5517
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9818
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9799
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9840
             Prevalence : 0.4483
         Detection Rate : 0.4393
   Detection Prevalence : 0.4393
      Balanced Accuracy : 0.9900

       'Positive' Class : 0
```

Correlation And chi-square tests with feature selection

Random forest:

```
> rf_anal<-collect(rf_pred)
> cor(rf_anal$label,rf_anal$prediction)
[1] 0.9992444
> rf_table<-table(rf_anal$prediction,rf_anal$label)
> chisq.test(rf_table)

        Pearson's Chi-squared test

data:  rf_table
X-squared = 24461, df = 69, p-value < 2.2e-16
```

Logistic Regression:

```
> log_anal<-collect(log_pred)
> cor(log_anal$label,log_anal$prediction)
[1] 1
> log_table<-table(log_anal$prediction,log_anal$label)
> chisq.test(log_table)

        Pearson's Chi-squared test with Yates' continuity correction

data:  log_table
X-squared = 24457, df = 1, p-value < 2.2e-16
```

Decision Tree:

```
> dt_anal<-collect(dt_pred)
> cor(dt_anal$label,dt_anal$prediction)
[1] 1
> dt_table<-table(dt_anal$prediction,dt_anal$label)
> chisq.test(dt_table)

        Pearson's Chi-squared test with Yates' continuity correction

data:  dt_table
X-squared = 24457, df = 1, p-value < 2.2e-16
```

Naïve Bayes:

```
> nb_anal<-collect(nb_pred)
> cor(nb_anal$label,nb_anal$prediction)
[1] 0.5328349
> nb_table<-table(nb_anal$prediction,nb_anal$label)
> chisq.test(nb_table)

        Pearson's Chi-squared test with Yates' continuity correction

data:  nb_table
X-squared = 6942.6, df = 1, p-value < 2.2e-16
```

Gradient Boosted Trees:

```
> gbt_anal<-collect(gbt_pred)
> cor(gbt_anal$label,gbt_anal$prediction)
[1] 1
> gbt_table<-table(gbt_anal$prediction,gbt_anal$label)
> chisq.test(gbt_table)

        Pearson's Chi-squared test with Yates' continuity correction

data:  gbt_table
X-squared = 24457, df = 1, p-value < 2.2e-16
```

6.2 METRICS USED FOR EVALUATION

The accuracy is the percentage of the correctly classified records over all the rows of the data set, whether correctly or incorrectly classified, as reflected in the following Equation:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Sensitivity: It is also called true positive rate. It is used to measures the proportion of positives that are correctly identified as such.

$$Sensitivity = \frac{TP}{TP+FN}$$

Specificity: It is also called true negative rate. It is used to measures the proportion of negatives that are correctly identified as such.

$$Specificity = \frac{TN}{TN+FP}$$

6.3 COMPARITIVE ANALYSIS ON UNSW-NB Dataset

With Feature Selection

Algorithms	Accuracy	Sensitivity	Specificity	Training time	Prediction time
Logistic Regression	0.9999	0.9998	1.0000	6.656	1.057
Decision Tree	1	1.0000	1.0000	5.568	0.804
Random Forest	0.991	0.9799	1.0000	7.18	0.929
Naïve Bayes	0.7535	0.8709	0.6581	4.352	0.905
Gradient Boosted Trees	1	1.0000	1.0000	22.352	0.584

Without Feature Selection

Algorithms	Accuracy	Sensitivity	Specificity	Training time	Prediction time
Logistic Regression	0.9999	0.9998	1.0000	6.675	1.064
Decision Tree	1	1.0000	1.0000	5.572	0.832
Random Forest	0.991	0.9799	1.0000	7.26	0.943
Naïve Bayes	0.7535	0.8709	0.6581	4.78	0.957
Gradient Boosted Trees	1	1.0000	1.0000	26.542	0.592

6.4 CONCLUSION

It is observed from the above table that Decision tree algorithm and gradient boosting (with feature selection) perform better than all the remaining classifiers in terms of sensitivity having 100%.Random Forest and Logistic Regression have almost the same sensitivity with values 97.99% and 99.98%.

It is observed that specificity for the Random Forest, Logistic Regression, Gradient Boosted tree and Decision Tree based schemes are almost same with 100% specificity. Naïve Bayes is the least ranked amongst all the classifiers in terms of Specificity.

Among the all classifiers, Decision tree and Gradient boosted trees perform better in terms of accuracy with 100% and the accuracy of the Naïve Bayes based scheme is lower among the all schemes with 75.35%.

Naïve Bayes is the fastest of all classifier methods to train, with a training time of just 4.352 seconds. Decision tree is the second most efficient and took a total of 5.568 seconds to train followed by Random Forest which took almost 6.656 seconds.

Gradient boosted trees is the fastest of all classifier methods to predict, with a prediction time of just 0.584 seconds. Decision tree is the second most efficient and took a total of 0.804 seconds to predict.

Considering all the parameters, we can conclude that decision tree algorithm is the best classification algorithm for intrusion detection.

7.REFERENCES

1. J. R. Quinlan, C4. 5: programs for machine learning, 2014

2. H. Karau, A. Konwinski, P. Wendell & M. Zaharia. Learning spark: lightning-fast big data analysis. " O'Reilly Media, Inc.". (2015)

3. UNSW-NB15 data set. Available on: http://www.cybersecurity.unsw.adfa.edu.au/ADFA%20NB15%20Datasets/, 2015

4. N. Moustafa & J. Slay, The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set, in Information Security Journal: A Global Perspective, 25 (2016) p.18–31

5. N. Moustafa and J. Slay, UNSW-NB15: a comprehensive dataset for network intrusion detection systems (UNSW-NB15 network data set), in Military Communications and Information Systems Conference, 2015.

6. B. A. Tama & K. H. Rhee, A combination of PSO-based feature selection and tree-based classifiers ensemble for intrusion detection systems, in Advances in Computer Science and Ubiquitous Computing, (2015) 489–495.

7. S. Choudhury & A. Bhowal, Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection, Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), 2015 International Conference, (2015) 89–95

8.S.X.Wu&W.Banzhaf, The use of computational intelligence in intrusion detection systems: A review, in Applied Soft Computing, 10 (2010) 1–35

9. M. Tavallaee, E. Bagheri, W. Lu & A. A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium, (2009) 1–6.

10. KDD CUP 1999 Data Set Available on: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

11. NSL-KDD Data Set for network-based intrusion detection systems, Available on: http://iscx.ca/NSL-KDD/, 2009.