Mini-Project 3: Network Intrusion Detection

Version 1.0

CSC215, Fall 2018

Oct 22nd, 2018

Prepared by: Chandini Nagendra

Siddharth Chittora

Contents

[1. Problem Statement 3](#_Toc527957908)

[2. Methodology 3](#_Toc527957909)

[3. Experimental Results and Analysis 3](#_Toc527957910)

[4. Task Division 5](#_Toc527957911)

[4.1. Chandini Nagendra: 5](#_Toc527957912)

[4.2. Siddharth Chittora 5](#_Toc527957913)

[5. Project Reflection 5](#_Toc527957914)

[6. Additional Features 6](#_Toc527957915)

[7. References 8](#_Toc527957916)

# Problem Statement

This project aims to build a network intrusion detector, a predictive model capable of distinguishing between bad connections, called intrusions or attacks, and good normal connections.

# Methodology

Here we are using dataset containing a wide variety of intrusions simulated in a military network environment. We have implemented Sklearn models for Logistic Regression, SVM, Gaussian Naïve Bayes, KNN, TensorFlow models and CNN.

* We are using the entire dataset.
* The dataset lacked column headers, so we manually added them.
* We normalized numeric features and encoded categorical features.
* We dropped rows with missing values.
* We encode the Outcome column with 0 for normal connection and 1 for all the other intrusion.
* We then split the data for training and testing.
* We implemented the Sklearn models for Logistic Regression, SVM, Gaussian Naïve Bayes, KNN.
* We created three Tensorflow models with activation function ReLU, Sigmoid and Tanh.
* We also used Early Stopping and checkpointing and 4 hidden layers in these models.
* We implemented CNN model to handle numeric data.
* Here we transformed the shape of the data to make it look like an 2D image to be used for Conv2D.
* We plotted the Confusion Matrix and ROC curve for each model to analyze the performance of the models for the given data.
* We obtained the accuracy, recall, precision and F1-score of all the models, and the comparison can be seen in the table below.

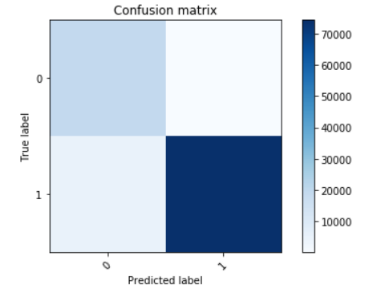
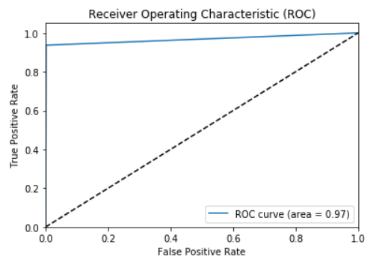
# Experimental Results and Analysis

* We observed that when we consider the full data set, for every model that we run, we got 99% accuracy.
* We tried to analyze whether it is because of the model tuning or the redundancy in the data set.
* We are showing below the confusion matrix and the ROC curve of the Gaussian Naïve Bayes and CNN.
* The confusion matrix and ROC curve of all the other models are like CNN.

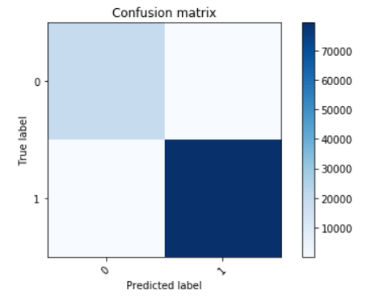
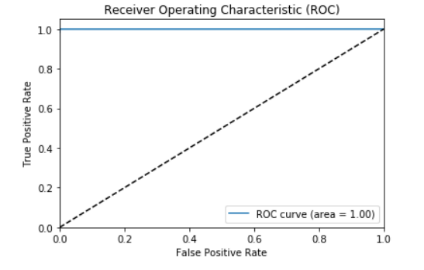
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model & Tuning** | **Accuracy** | **Precision** | **Recall** | **F1** **Score** |
| Logistic Regression | 0.9946257236549128 | 0.9946828323036672 | 0.9946257236549128 | 0.9946408290044061 |
| KNN | 0.9995546738998421 | 0.9995549353996092 | 0.9995546738998421 | 0.9995547520894394 |
| SVM | 0.9983300271244079 | 0.9983315628157213 | 0.9983300271244079 | 0.9983305969065448 |
| Gaussian NB | 0.9489089510546131 | 0.9593017637420566 | 0.9489089510546131 | 0.9509965159983609 |
| **Fully Connected NN** |  |  |  |  |
| ReLU + adam + early stopping and checkpoint + 4 hidden layers | 0.9992814056111089 | 0.999281343572176 | 0.9992814056111089 | 0.9992813705348386 |
| Sigmoid + sgd + early stopping and checkpoint + 4 hidden layers | 0.9949394761345695 | 0.9950494730768633 | 0.9949394761345695 | 0.9949616324501922 |
| Tanh + adagrad + early stopping and checkpoint + 4 hidden layers | 0.9992510424679163 | 0.9992514105204905 | 0.9992510424679163 | 0.9992511739686027 |
| **CNN** conv2d\_1 (41 , ReLU, (1,3), (1,1))+  conv2d\_2 (82 , ReLU, (1,3))+ max\_pooling2d\_1(1,2) +  dropout\_1 (0.25)+  flatten\_8 +  dense\_1 (164, ReLU) +  dropout\_2 (0.5) +  dense\_2 (2) output. | 0.9994129792316101 | 0.9994134960875988 | 0.9994129792316101 | 0.999413128064702 |

**Confusion Matrix and ROC**

* Gaussian Naïve Bayes



* CNN



# Task Division

## Chandini Nagendra:

* Logistic Regression
* Gaussian NB
* Fully Connected Neural Networks
* Report

## Siddharth Chittora

* SVM
* KNN
* CNN
* Report

Discussed together on how to improve the model and came up with the solution discussed in the additional features section.

# Project Reflection

* All the models have an outstanding accuracy of 99%.
* We tried to remove null values and found out that there were no null values in the dataset.
* We removed redundancy which reduced the dataset significantly.
* Even after the removing redundancy the accuracy stayed at 99%.
* Though the accuracy remained the same we observed that the dataset with redundant record are highly skewed towards intrusion or attacks as an outcome.
* After the removal of redundant records, we observed that maximum of the connections was normal.
* Since we did not have complete domain knowledge, we researched existing papers on data mining for network intrusion detection.
* We were surprised to find that features suggested in the papers were almost similar to the features selected after Feature Importance Analysis using tree-based estimators.
* It was challenging to perceive the data as an image, whether to use it as greyscale image of size 1X41 or color image of size 1X1 with 41 channels.
* Therefore, we chose to go with the feature selected by the tree-based estimator.
* We experimented with a new CNN architecture to see how it would impact the accuracy of the model, because every other model that we worked with gave 99% accuracy.

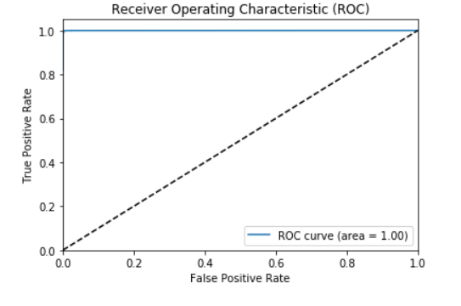
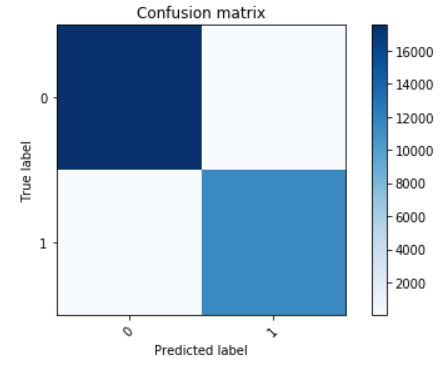
# Additional Features

* The data set had a lot redundancy which skewed the results.
* We removed redundancy from the entire dataset and ran all the models.
* We performed feature importance analysis using tree-based estimators which computes the feature importance
* We coupled it with the Sklearn feature selection meta-transformer to discard the irrelevant features and reduced the number of features to train the models.
* We also researched to see if there was an existing proven CNN architecture for this type of problem.
* We found an architecture which we have implemented in the additional features section.
* The implemented model has sixteen layers, which is a mix of eight Conv2d layers, three Max-pooling layers, one flatten layer, two hidden layers and one dropout layer.

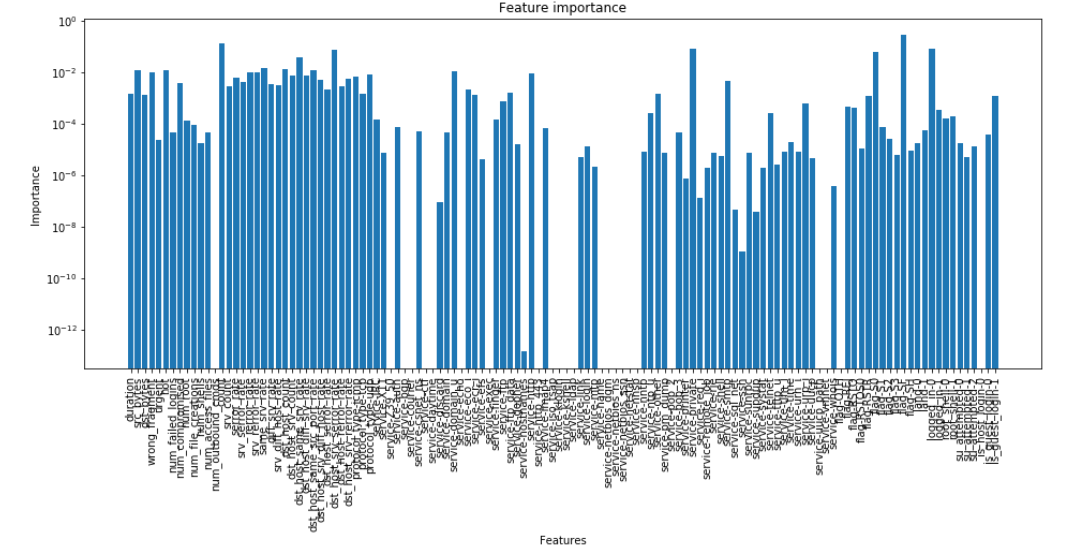
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model & Tuning** | **Accuracy** | **Precision** | **Recall** | **F1** **Score** |
| **Redundancy Removal** |  |  |  |  |
| Logistic Regression | 0.9876360888827832 | 0.9876555198142505 | 0.9876360888827832 | 0.9876232148457863 |
| KNN | 0.9983514785177044 | 0.9983514386974491 | 0.9983514785177044 | 0.9983513798219226 |
| SVM | 0.9922038671566439 | 0.9922025866386067 | 0.9922038671566439 | 0.9922020510243128 |
| Gaussian NB | 0.918295154033726 | 0.926633232685079 | 0.918295154033726 | 0.9163503763232373 |
| **Fully Connected NN** |  |  |  |  |
| ReLU + adam + early stopping and checkpoint + 4 hidden layers | 0.9985231995054435 | 0.9985233281256618 | 0.9985231995054435 | 0.9985230557019816 |
| Sigmoid + sgd + early stopping and checkpoint + 4 hidden layers | 0.986331009375966 | 0.986336941032662 | 0.986331009375966 | 0.9863197626089032 |
| Tanh + adagrad + early stopping and checkpoint + 4 hidden layers | 0.9980767249373218 | 0.9980766831171078 | 0.9980767249373218 | 0.998076580954231 |
| **CNN** conv2d\_1 (41 , ReLU, (1,3), (1,1))+  conv2d\_2 (82 , ReLU, (1,3))+ max\_pooling2d\_1(1,2) +  dropout\_1 (0.25)+  flatten\_8 +  dense\_1 (164, ReLU) +  dropout\_2 (0.5) +  dense\_2 (2) output | 0.9994129792316101 | 0.9994134960875988 | 0.9994129792316101 | 0.999413128064702 |
| **Feature Selection based on Feature Importance Analysis** |  |  |  |  |
| Logistic Regression | 0.9837208503623313 | 0.9837784158032781 | 0.9837208503623313 | 0.9836938095343001 |
| KNN | 0.9981110691348697 | 0.9981118993313256 | 0.9981110691348697 | 0.9981107146517584 |
| SVM | 0.9805268399903836 | 0.9809747688267506 | 0.9805268399903836 | 0.980447975698628 |
| Gaussian NB | 0.9782944671497751 | 0.9785952641646312 | 0.9782944671497751 | 0.9782192710782204 |
| **Fully Connected NN** |  |  |  |  |
| ReLU + adam + early stopping and checkpoint + 4 hidden layers | 0.997458529381461 | 0.9974605511391783 | 0.997458529381461 | 0.9974589455556856 |
| Sigmoid + sgd + early stopping and checkpoint + 4 hidden layers | 0.9869148607342789 | 0.9869227666119975 | 0.9869148607342789 | 0.986903789403367 |
| Tanh + adagrad + early stopping and checkpoint + 4 hidden layers | 0.9970463990108871 | 0.9970511600234251 | 0.9970463990108871 | 0.9970471884585685 |
| **CNN** conv2d\_1 (41 , ReLU, (1,3), (1,1))+  conv2d\_2 (82 , ReLU, (1,3))+ max\_pooling2d\_1(1,2) +  dropout\_1 (0.25)+  flatten\_8 +  dense\_1 (164, ReLU) +  dropout\_2 (0.5) +  dense\_2 (2) output | 0.9977332829618436 | 0.9977349414187415 | 0.9977332829618436 | 0.997733620522209 |
| **Experimental CNN Architecture** |  |  |  |  |
| **CNN** 8 Conv2d layers + 3 Max-pooling layers + 1 flatten layer + 2 Dense hidden layers + 1 dropout layer | 0.9986605762956349 | 0.9986607317324747 | 0.9986605762956349 | 0.9986604458692393 |

**Confusion Matrix and ROC**

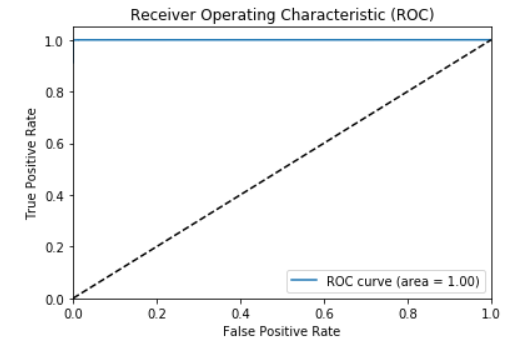
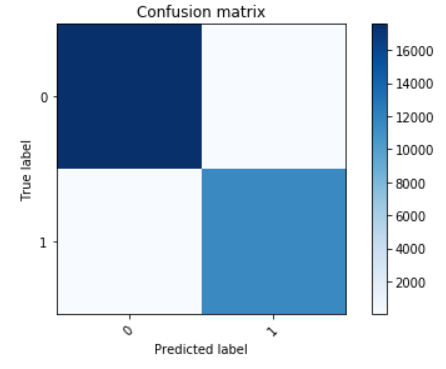
* CNN after Redundancy Removal



* Feature Importance Analysis Plot



* CNN after Feature Selection



# References

[1] Md Moin Uddin Chowdhury and *et. al.* “A Few-shot Deep Learning Approach for Improved Intrusion Detection”, IEEE UEMCOM 2017, October 2017