Machine Learning Models for a Network Intrusion Detector

Dane Jew, Sirish Prabakar

Machine Learning Models for a Network Intrusion Detector

Abstract

Machine learning can be used to predict the existence of a network intrusion or attack based on the conditions or attributes of a network. The question is which learning model is the best at this task. Using a dataset of network attributes, the goal is to find out which of the four models, Logistical Regression, Nearest Neighbor, Support Vector Machines, or a Fully Connected Neural Network can best accurately predict a network intrusion or attack.

I. Problem Statement

The objective of this project is to design a network intrusion detector by using 4 different machine learning models and comparing their performances in terms of accuracy, precision, recall and F-scores. This is to determine whether or not an intrusion or attack has occurred in various networks. This is a classification problem to be solved by taking network data of known connections and determining whether they are good “normal” connection or “bad” connections, which is an intrusion or attack. The network data is extensive and can include attributes such as number of packets transferred, type of service used, or response time, to name a few. Based on these and other conditions it can be determined whether a connection is “normal”, or an “attack”.

II. Methodology

The four models that will be used to analyze the network data and determine a network intrusion or attack are Logistical Regression, Nearest Neighbor, Support Vector Machine, and Fully Connected Neural Networks. Although these models are required by this project, they are also probably chosen because they are commonly used in classification problems.

Dataset

Using machine learning, in order to approach any of the proposed models it is important to determine which attribute of the data is the target for classification and which are the features. In this case the classification target is binary, i.e. good normal connections versus bad connections via intrusion or attack. The features are the rest of the network data, such as “service”, “state”, packets, (transfer) rate, etc. The network data used for this project is the UNSW-NB 15 dataset. (Moustafa n.d.)

One problem with the dataset is that the training and testing data is separated into two files with an extra column in the test data. Any model that tries to fit the data in this manner will result in a value error. In order to resolve this it is necessary to preform a merge between the training and test data before any preprocessing can be attempted.

Preprocessing the data includes transforming the categorical data into “dummy” variables, or “One-Hot” encoding. This involves taking a categorical column, such as “service” and creating a binary column for each text item that exist within that column. An example of this is if “udp” is a service used, the function “encode_text_dummy” will automatically create a binary column for “service-udp”. This function is called for all category columns.

Columns that contain numeric data, such as “dur” (duration) may require normalization in order to prevent overfitting. The function used for normalizing numeric data is called “encode_numeric_zscore”. Normalization is necessary to prevent outlier data from skewing the approximation of the true model.

The models used for this project are Logistical Regression, Nearest Neighbor, Support Vector Machines, and Fully-Connected Neural Networks. All four models require the library “SKLearn” while the FCNN in addition uses “Tensor Flow”.

Non-Neural Network Machine Learning Models

The classifier used for Logistical Regression model is “LogisticRegression” from the “SKLearn” library. The “solver” parameter is the “lbfgs” optimizer and is set for 1000 max iteration.

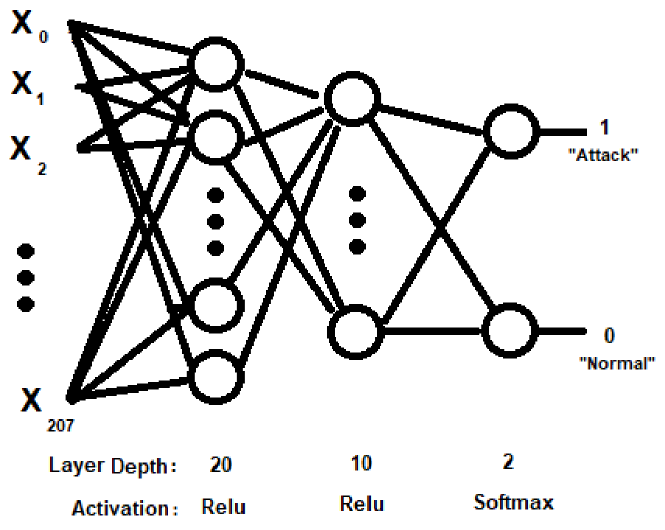
“KNeighborClassifier” is a classifier used for the Nearest Neighbor model. This classifier has only one argument, “n_neighbors”. This argument in this case is set to two meaning it will check the two nearest neighbors to determine the classification..

The SVM model uses the LinearSVC method from the SKLearn library. The reason why this is chosen instead of SVC is because this version provides more flexibility in terms of penalties and loss functions and is able to handle a larger dataset. Only the basic parameters were used for this classifier. These include “random_state=0”, “tol=1e-5”, and “max_iter=2500”.

Fully-Connected Neural Network (FCCN).

The design of the Fully-Connected Neural Network, or FCCN involves a three layer neural network. The first layer has 20 neurons with a “Relu” activation function. The second layer is a hidden layer with 10 neurons with an activation function which is also based on the “Relu” algorithm. The third is the output layer with 2 neurons. The reason for this is that it represents the two class, “Normal” and “Attack”. In order to prevent overfitting the activation function used for the output layer is Softmax. The problem with this is that the resulting prediction value results in a floating point value, and in some cases larger than 1. Numpy argmax is used to rectify the prediction value to generate a binary value.

Figure 1
Model
Fully-Connected Neural Network (FCNN)



III. Experimental Results and Analysis

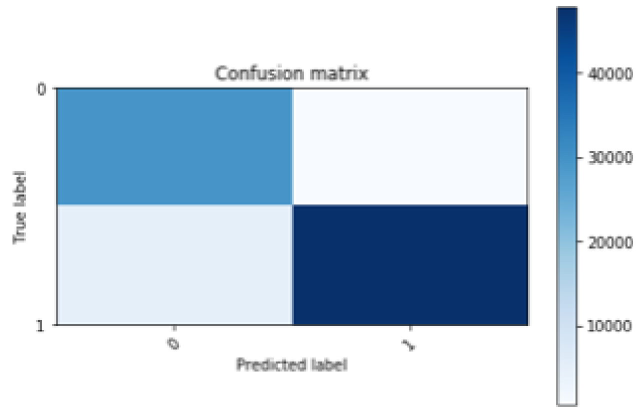
Logistical Regression

When executing the "fit" function for the logistical regression model, a few observations were made. The parameters passed for this classifier was basic. The accuracy was close to 99.99 percent. This was the same for the precision, recall and F1 scores. When compared to other scores this model performed very well.

Selecting an optimizer also has an effect on the accuracy. The default optimizer is "adam" which produces a 99.99% accuracy. When changing to "sgd", however the accuracy goes down to 50%.

Figure 2
Performance Measures & Confusion Matrix
for Logistical Regression

```
Accuracy score: 0.99996356216295
Precision score: 0.999963562216754
Recall score: 0.99996356216295
F1 score: 0.9999635620300538
Plotting confusion matrix
```

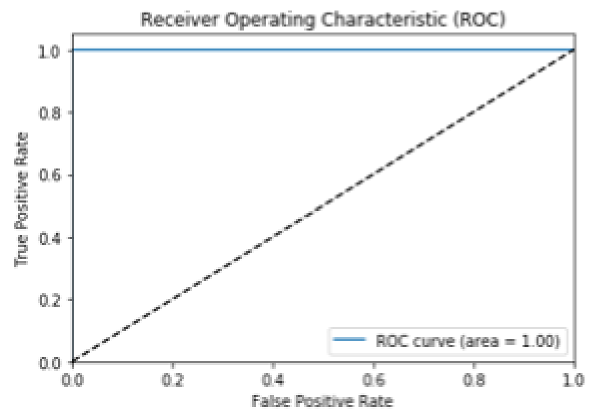


```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     29755
         1.0       1.00      1.00      1.00     52577

    accuracy                           1.00     82332
   macro avg       1.00      1.00      1.00     82332
weighted avg       1.00      1.00      1.00     82332
```

Figure 3
Logistical Regression ROC Curve

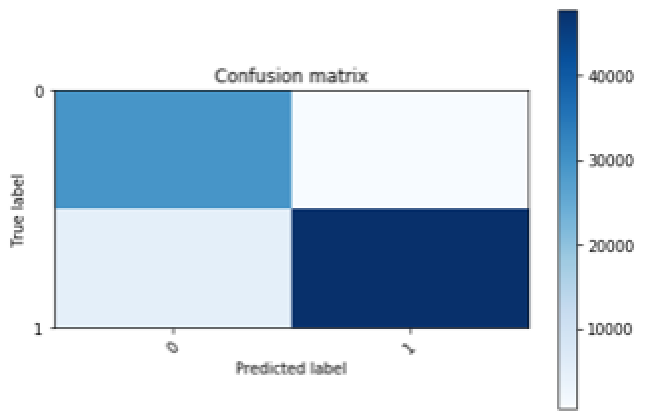


Nearest Neighbor

The KNN classifier performed fairly well. Although the difference is under 7% which is minute, it is not as precise or accurate as the logistical regression classifier.

Figure 4
Performance Measures & Confusion Matrix
for K-Nearest Neighbor

```
Accuracy score: 0.93345236360103
Precision score: 0.9402466845865659
Recall score: 0.93345236360103
F1 score: 0.9342723810575654
[[29153   602]
 [ 4877 47700]]
Plotting confusion matrix
```

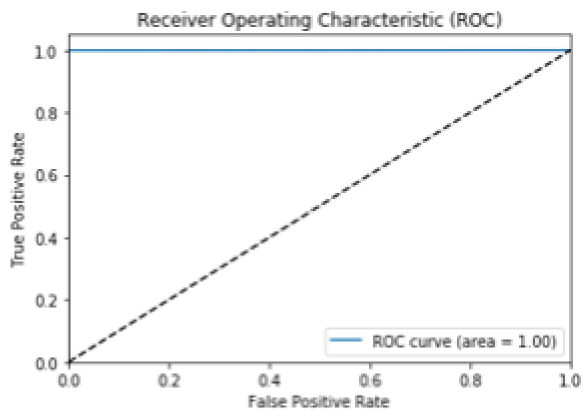


```
              precision    recall  f1-score   support

         0.0       0.86      0.98      0.91     29755
         1.0       0.99      0.91      0.95     52577

    accuracy                           0.93     82332
   macro avg       0.92      0.94      0.93     82332
weighted avg       0.94      0.93      0.93     82332
```

Figure 5
K-Nearest Neighbor ROC Curve

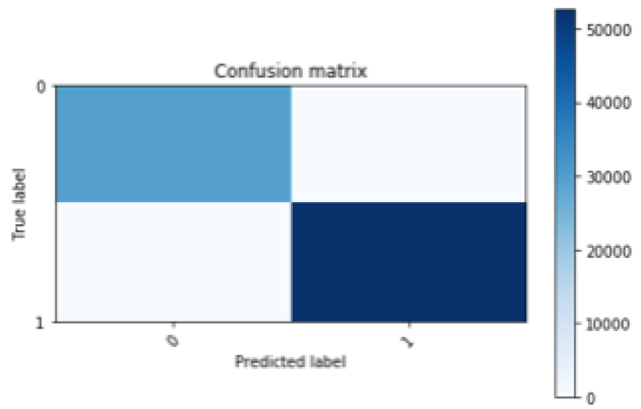


Support Vector Machine

The SVM classifier did surprisingly well. The accuracy, precision, recall, and F1 score is at a full 100%.

Figure 6
Performance Measures & Confusion Matrix
for Support Vector Machine

```
Accuracy score: 1.0
Precision score: 1.0
Recall score: 1.0
F1 score: 1.0
[[29755     0]
 [    0 52577]]
Plotting confusion matrix
```

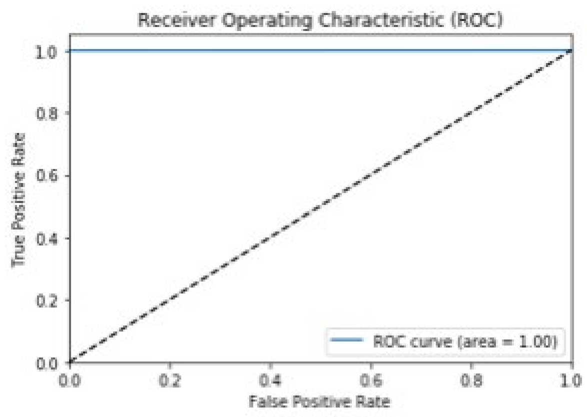


```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     29755
         1.0       1.00      1.00      1.00     52577

    accuracy                           1.00     82332
   macro avg       1.00      1.00      1.00     82332
weighted avg       1.00      1.00      1.00     82332
```

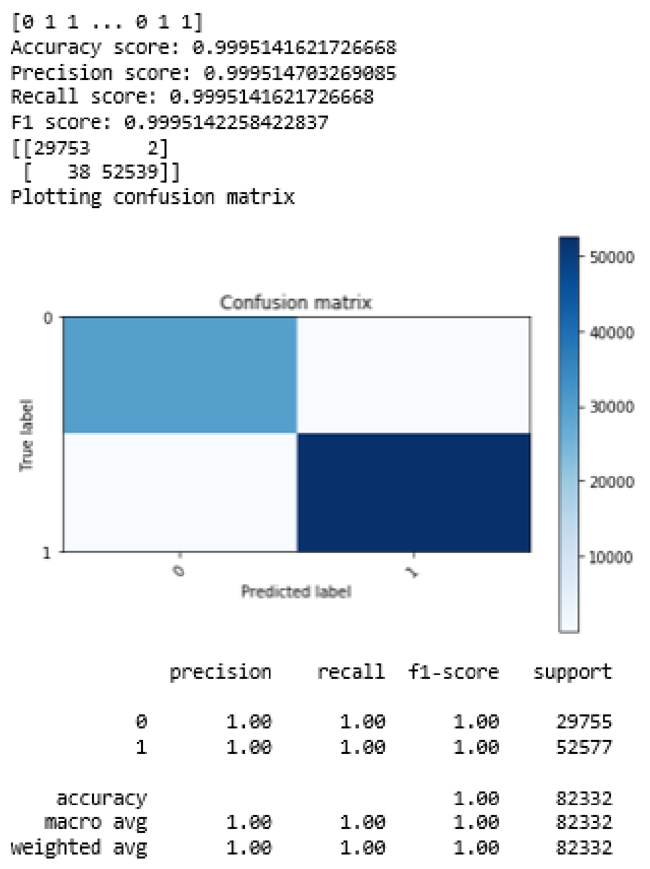
Figure 7
Support Vector Machine ROC Curve



FCNN

In analyzing the performance of FCNN, it is apparent that it is better than the KNN model but not as good as the Logistical Regression or SVM models. This is indicated by the confusion matrix below. The accuracy, precision, recall, and F1 scores are at 99.95%..

Figure 8
Performance Measures & Confusion Matrix
for FCNN



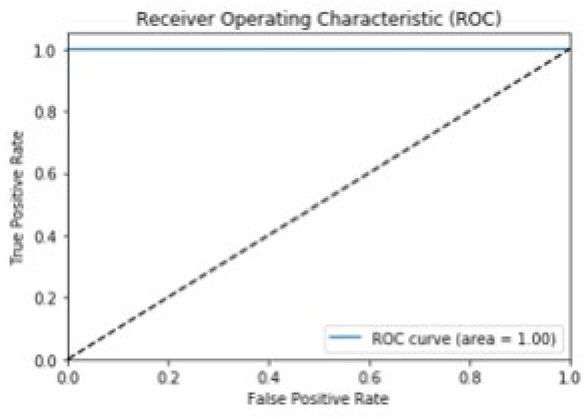
```
[0 1 1 ... 0 1 1]
Accuracy score: 0.9995141621726668
Precision score: 0.999514703269085
Recall score: 0.9995141621726668
F1 score: 0.9995142258422837
[[29753     2]
 [   38 52539]]
Plotting confusion matrix
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     29755
           1       1.00      1.00      1.00     52577

    accuracy                           1.00     82332
   macro avg       1.00      1.00      1.00     82332
weighted avg       1.00      1.00      1.00     82332
```

Table 9
FCNN ROC Curve



IV. Task Division and Project Refletion

The work breakdown is as follow in the table below:

Coding

	Work Pct
Dane Jew	40%
Sirish Prabakar	60%

Documentation

	Work Pct
Dane Jew	60%
Sirish Prabakar	40%

Challenges

Initially both team members worked on the assignment individually, but with mixed progress from each member. At one point one of the team members almost came close to completion of all four models while the other struggled to generate precision data due to improper preprocessing and model design.

The biggest challenge for this project was the preprocessing step. The data required for the target column (y) for the Neural Network is not the same as the data required for the Logistical Regression, Nearest Neighbor, or Support Vector Machine models. The issue was function “to_xy” which is used to separate the test data from training data. At issue was the data type of the target variable. In the FCNN this function was adequate for defining the target classes as “One Hot” or “dummy” values. This was not the case for the other models which required a single numeric value.

The FCNN model also provided another challenge when trying to get the accuracy as close to 100% as possible. Although the epoch value was set to 1000, early stopping was causing the fit to stop short of converging. A loop of 5 iterations was employed to improve the fit. This worked surprisingly well as illustrated by the performance measures on Table 7.

V. Additional Features

(1) “Can you model this intrusion detection problem as a multi-class classification problem so that we can detect the type of each intrusion? How good such predictive model can be in this case?”

Answer:

We created a new model for intrusion detection by using column 'attack_cat' as a target column to perform multi class classification. Since the dataset was unbalanced we were able achieve high F-1 score for this. The dataset requires downsampling by balancing the majority classes and minority classes with appropriate weights.

(3) “Among all the features, can you identify the most important features (this is so called feature importance analysis) and train models only on those important features, e.g., top-10 most important features? What would be the benefits to do that?”

Answer:

We used Extra Trees classifier to select 10 columns which have a strong relation with the target column(label). By having the list of these top 10 features we were able to assign these 10 features to x (as input features) and trained the model with only these 10 features. The obtained F-1 was identical to the F-1 score in the case where all features were considered.

VI. Conclusion

Logistical Regression, Nearest Neighbor, Support Vector Machines, and Fully Connected Neural Networks are indeed tools that can be used to detect network intrusions or attack. The trick is identifying the appropriate features and how they relate to the positive identification of an attack. When an important feature is identified, it is equally important if it is categorical or numeric and if it corelates with the classification of a threat. After this is considered, one must then consider what type optimization, algorithm, kernel function in the case of SVM, and in the case of Neural Networks activation functions to use with the learning model. After that determination it is also necessary to compare the accuracy or precision of that model.

References

Moustafa, Nour and Jill Slay. *The UNSW-NB15 Dataset Description.* n.d. https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/.