NETWORK INTRUSION DETECTION

Report by – Himanshu Chauhan

Table of Contents

(1) 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.

(2) Methodology

Used the following models to detect bad connections (intrusions). Compared the accuracy, recall, precision and F1-score of ALL the following models.

1. Logistic Regression
2. Nearest Neighbor
3. Support Vector Machine
4. Gaussian Naive Bayes
5. Fully-Connected Neural Networks
6. Convolutional Neural Networks (CNN)

- Used the entire dataset
- Loaded the dataset into the dataframe
- Added column headers to the data in the dataframe
- Dropped null rows
- Dropped redundant rows
- Studies and understood each column values and accordingly performed – normalization,
- label encoding and one hot encoding on them
- Understood the attack types and divided (label encoded) them into two categories – Normal (0) and Malicious (1).

Below (Fig -1) is the visualization of the count for 0 and 1 category.

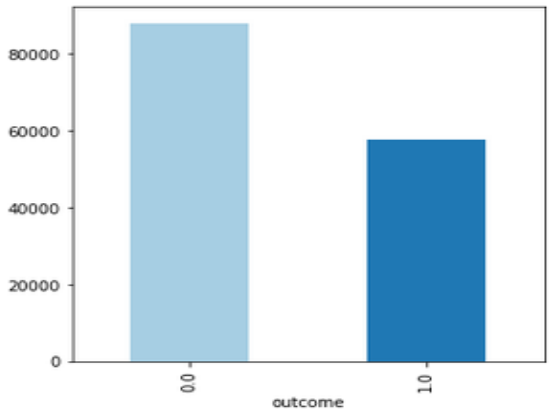


Fig- 1

- Split the data into 80 – 20 ratio for training and testing the models.
- Implemented the Sklearn models for Logistic Regression, SVM, Gaussian Naïve Bayes, KNN.

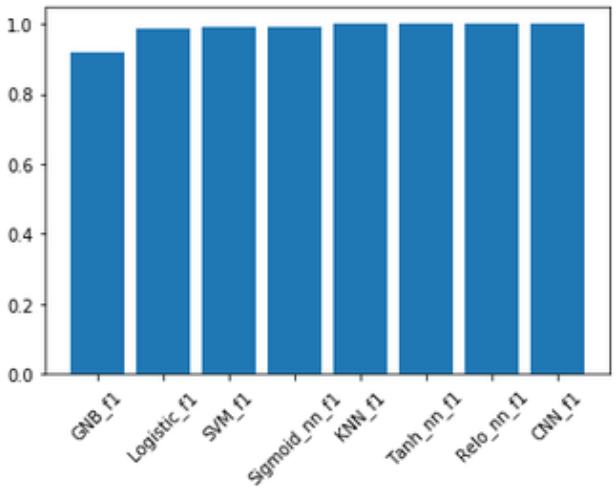
- Implemented Tensorflow models with activation function ReLU, Sigmoid and Tanh and compared their performances. Implemented Early Stopping and checkpointing and 4 hidden layers in these models.
- Implemented CNN model to handle numeric data. Transformed the shape of the data to make it look like an 2D image to be used for Conv2D.
- Calculated the accuracy, recall, precision and F1-score of all the models.
- Also plotted the Confusion Matrix and ROC curve for each model to analyze the performance of the models for the given data.
- In the end plotted a graph to compare F1 score for all these models.

3. Experimental Results and Analysis

3.1 Model Comparison Table

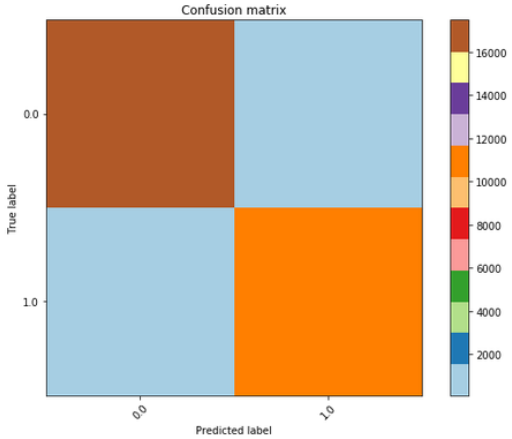
Model & Tuning	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Logistic Regression	0.9876	0.9876	0.98760433	0.987657498
KNN	0.998317	0.99831849	0.99831713	0.9983171343
SVM	0.990795	0.990796	0.9907957	0.9907957550
Gaussian Naïve Bayes	0.9185	0.92696	0.91850	0.9165476101
Tensor flow Fully Connected neural network models				
ReLU with Early stopping & checkpoint + adam + 4 hidden Layers	0.99866057	0.99866055	0.99866057	0.9986605662
Tanh with Early stopping & checkpoint + adam + relu + checkpoint + 4 hidden layers	0.998488	0.998488	0.998488	0.9984887421
Sigmoid with Early stopping & checkpoint + adam + 4 hidden layers	0.99179	0.99188	.991791	0.9917801405
CNN conv2d (32 , ReLU, (1,3), (1,1) + max_pooling2d_1(1,2))+ conv2d (64 , ReLU, (1,3))+ max_pooling2d_1(1,2) + dropout_1 (0.25)+ flatten Layer + dense_1 (128, ReLU) + dropout_2 (0.3) + dense_2 (64, ReLU) + dropout_2 (0.50) + dense_3 (2 , softmax) output.	0.9987979	0.9987979	0.9987979	0.9987978721

3.2 Comparing F1 Score for all models Listed in above Table

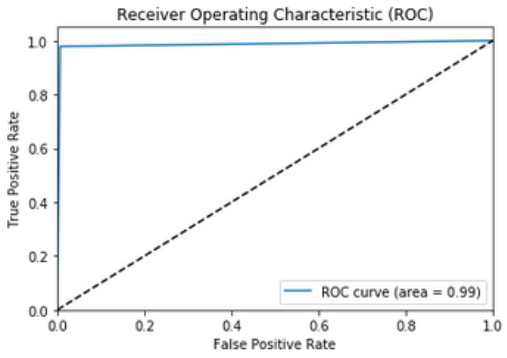


3.3 Confusion Matrix and Roc

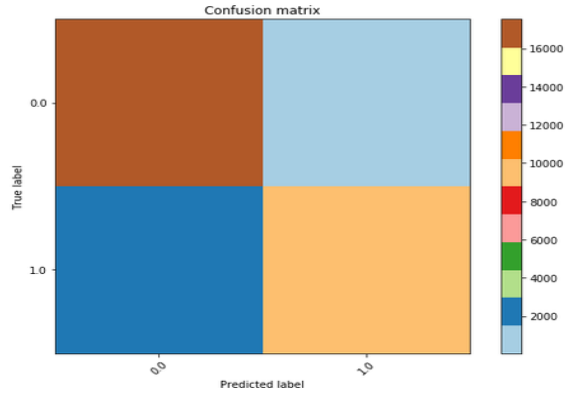
Logistic Regression

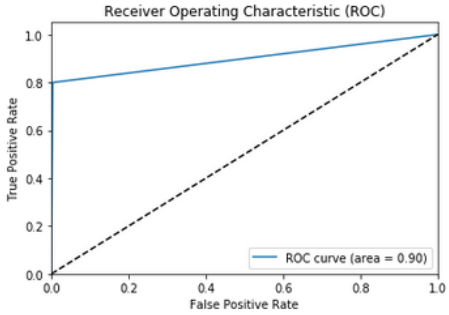


	precision	recall	f1-score	support
0.0	0.99	0.99	0.99	17589
1.0	0.99	0.98	0.98	11528
avg / total	0.99	0.99	0.99	29117



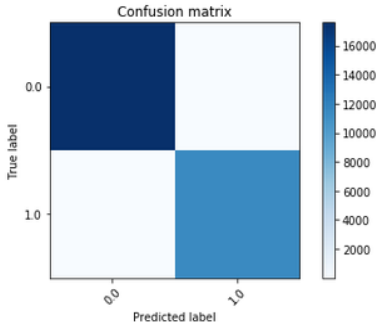
Gaussian Naive Bayes

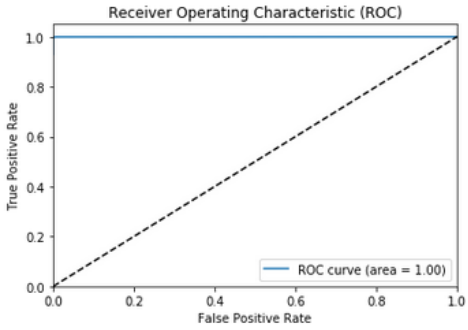




	precision	recall	f1-score	support
0.0	0.88	1.00	0.94	17589
1.0	0.99	0.80	0.89	11528
avg / total	0.93	0.92	0.92	29117

Convolutional Neural Networks (CNN)





	precision	recall	f1-score	support
0	1.00	1.00	1.00	17589
1	1.00	1.00	1.00	11528
avg / total	1.00	1.00	1.00	29117

4. Task Division: N/A (Individually completed the project)

5. Project Reflection

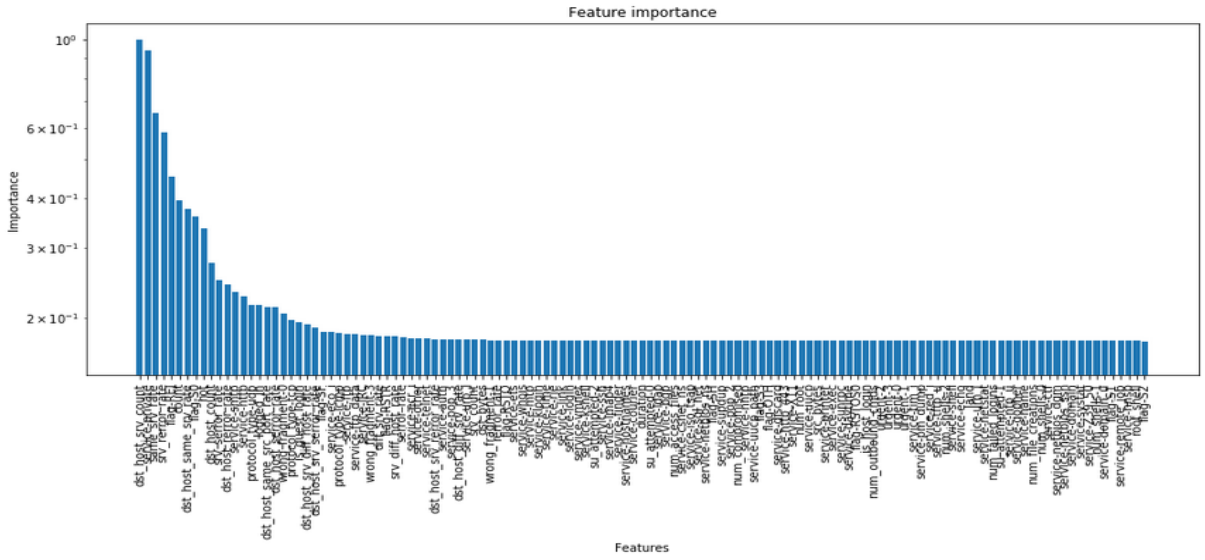
- Learnt to implement CNN Model. Learn how to convert text input as image to implement CNN.
- All models gave accuracy/F1 score greater than 97% except Gaussian Naïve Model which gave 91%
- There were no null values in the given dataset.
- The dataset contained lot of redundant rows and those were removed so that the learning algorithms are not biased towards the more frequent records
- Studied and researched about the columns and attack types. Some columns had just 2 values as 0 and 1. Did not normalize or encode those columns. Researched about the attack types and divided them into 4 categories.
- Feature Importance analysis gave really good results. The number of features were reduced from 127 to 16 and still gave accuracy above 98%
- In this project I learnt how to implement CNN model with different number of layers: CNN layer, max pooling layer, dropout layers, flatten, dense fully connected layers, output layer.
- Also, it was exciting to see the results of feature important analysis.

6. Additional Features:

6.1 Feature importance analysis

Implemented input perturbation and extra tree classifier

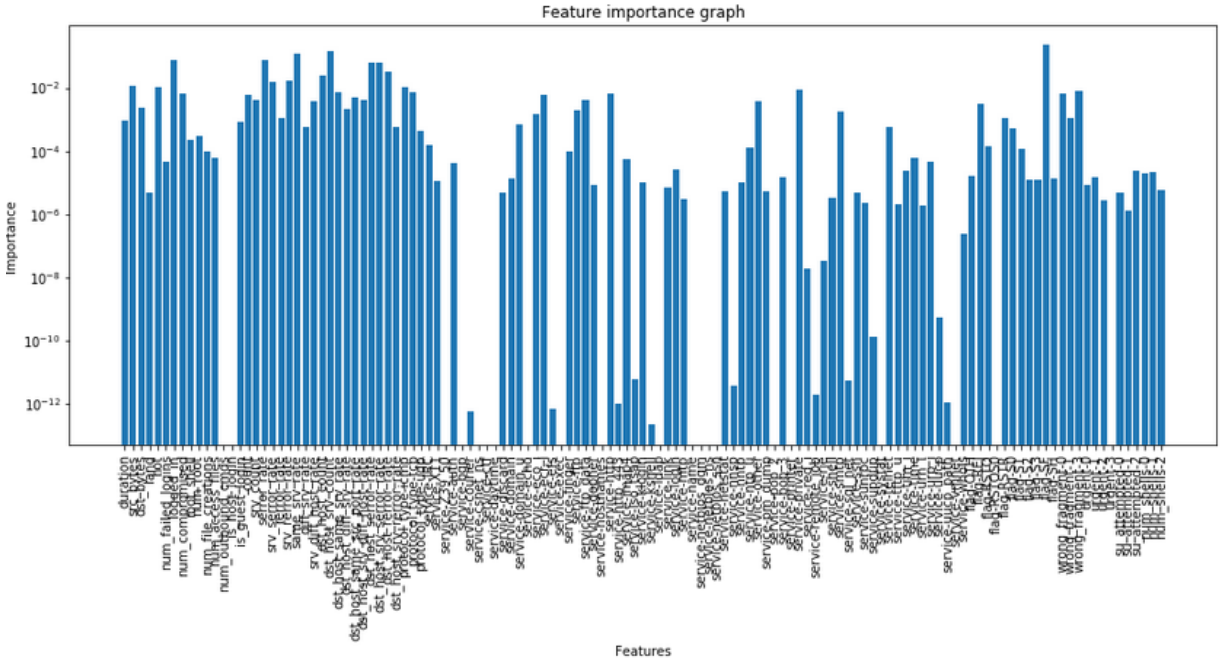
6.1.1 Input Perturbation:



- It uses log loss to evaluate a classification problem.
- Gave an already implemented logistic regression model as an input to evaluate the rank and importance for the features. Total number of features given as input were 127.
- It gave the error and importance for each feature.

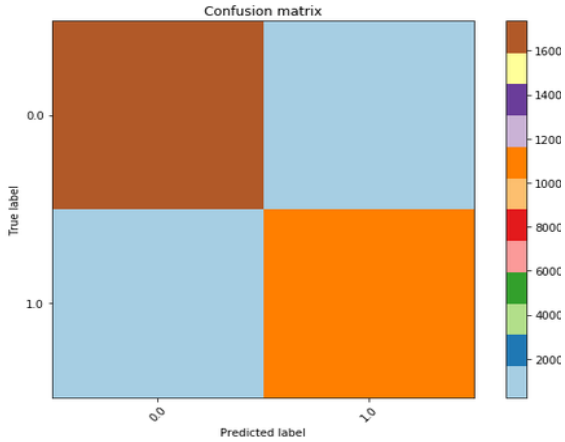
- Removed the features from the data frames with importance less than 0.20.
- Total number of features were reduced from 127 to 19.
- Split the data into train and test with 80:20 ratio, and again implemented the logistic regression model.
- The new logistic model (with 19 input features) gave F1 score as a 0.9855. The old F1 score was (with 127 input features) 0.9876

6.1.2 Extra Tree Classifier:



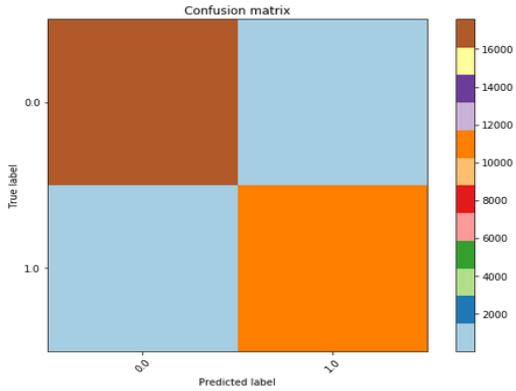
- Implemented extra tree classifier for feature importance analysis.
- It selected 16 input features out of 127.
- Implemented logistic regression model and tensor flow fully connected model with these 16 input features.
- Below is the confusion matrix for these:

Logistic Regression Model



	precision	recall	f1-score	support
0.0	0.97	0.99	0.98	17589
1.0	0.98	0.96	0.97	11528
avg / total	0.97	0.97	0.97	29117

Fully Connected Tensor Flow Model



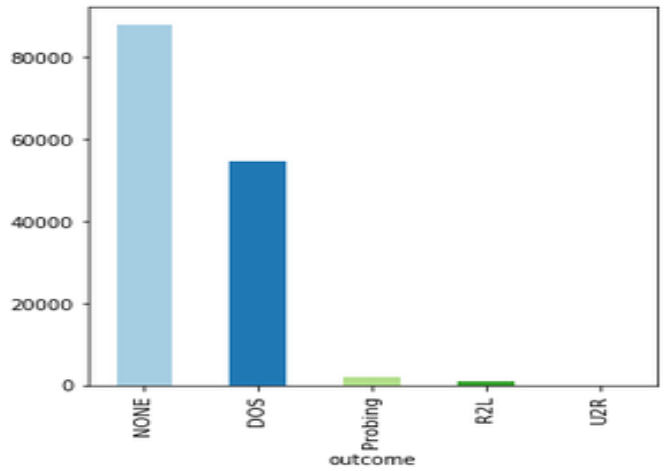
	precision	recall	f1-score	support
0	0.99	1.00	0.99	17589
1	1.00	0.99	0.99	11528
avg / total	0.99	0.99	0.99	29117

6.2 Multi-class classifier

Divided the target output into 5 categories as mentioned below:

1. **DOS**: denial-of-service. e.g. eardrop
2. **R2L**: unauthorized access from a remote machine, e.g. guessing password
3. **U2R**: unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow''
4. **Probing**: surveillance and other probing, e.g., port scanning.
5. **Normal**: No threat

Label encoded each category (0-4). Below is the graph for count(rows) of each category:



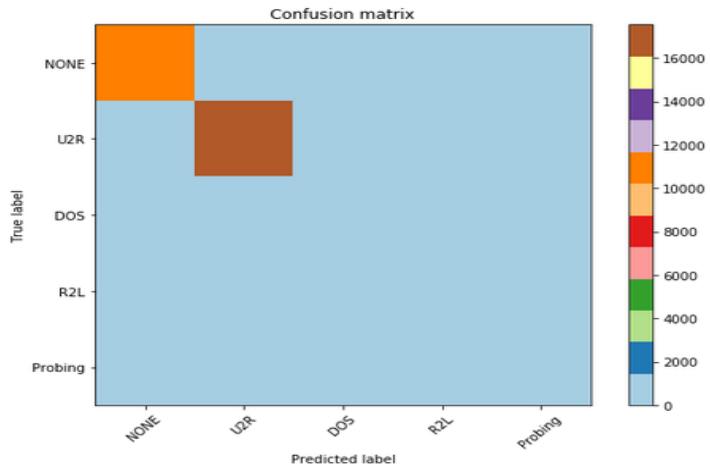
Implemented the following models for the multi class classification problem.

1. Logistic : Label encoded the output features
2. Fully connected, CNN, One-Hot encoded the output feature.

Model & Tuning	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.99017	0.99025	0.99017	0.99010
ReLU with Early stopping & checkpoint + adam + 4 hidden Layers	0.998694	0.998679	0.998694	0.998683
CNN conv2d (32 , ReLU, (1,3), (1,1) + max_pooling2d_1(1,2))+ conv2d (64 , ReLU, (1,3))+ max_pooling2d_1(1,2) + dropout_1 (0.25)+ flatten Layer + dense_1 (128, ReLU) + dropout_2 (0.3) + dense_2 (64, ReLU) + dropout_2 (0.50) + dense_3 (2 , softmax) output.	0.998454	0.998466	0.998454	0.998451

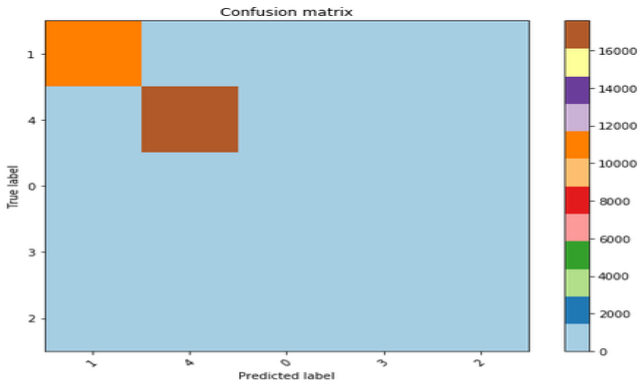
Below are the confusion matrices for these:

Logistic:



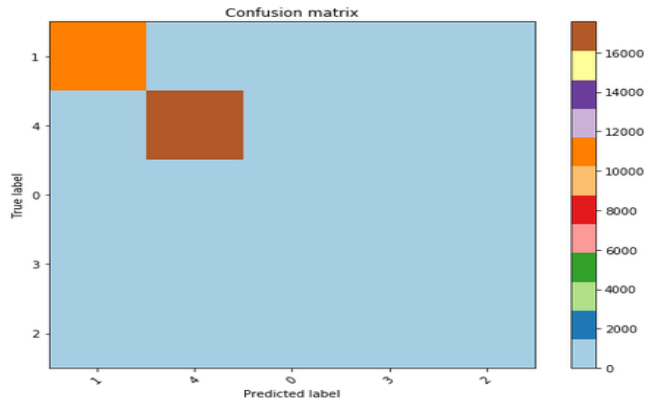
	precision	recall	f1-score	support
0	1.00	0.98	0.99	10924
1	0.99	1.00	0.99	17589
2	0.98	0.92	0.95	425
3	0.85	0.91	0.88	172
4	1.00	0.14	0.25	7
avg / total	0.99	0.99	0.99	29117

Fully_ Connnected:



	precision	recall	f1-score	support
0	1.00	1.00	1.00	10924
1	1.00	1.00	1.00	17589
2	1.00	0.98	0.99	425
3	0.95	0.92	0.93	172
4	0.83	0.71	0.77	7
avg / total	1.00	1.00	1.00	29117

CNN



	precision	recall	f1-score	support
0	1.00	1.00	1.00	10924
1	1.00	1.00	1.00	17589
2	1.00	0.98	0.99	425
3	0.88	0.94	0.91	172
4	0.60	0.43	0.50	7
avg / total	1.00	1.00	1.00	29117