Problem statement:

i) Idea

These different types of neural networks are at the core of the deep learning revolution, powering applications like unmanned aerial vehicles, self-driving cars, speech recognition, etc. Neural networks are used to identify and predict unusual activities in the system. Inspired by the performance of the neural networks in intrusion detection system we proposed to design a model.

ii) Scope

The world’s need on the internet is growing. Information is the most precious resource in the world. To protect information from unwanted hands is of utmost importance. Information gets stolen when a network gets breached. An intrusion detection system finds known and unknown attacks that facilitate breaching of a network. we model an intrusion detection system trained to detect such types of attack. We do so using neural networks which is a machine learning approach. Furthermore, we study the performance of our model in binary and multiclass classification environments. Our experiments show that using Convolutional Neural Networks and Recurrent neural networks are better approaches to detect network intrusions.

iii) Novelty

We have focused on Increasing the hidden Layers, Change Activation function, Change Activation function in Output layer, Increase number of neurons, Weight initialization, Normalizing/Scaling data expecting to obtain a better F1-score and accuracy.

iv) Comparative statement

In a need for the efficiency in finding ideal solutions within a finite amount of data, deep learning has gathered significant research attention. Javaid et al. use a deep learning method in the context of a deep neural network for flow-based anomaly detection, and it is seen through the results that deep learning is able to be used for anomaly detection in software-defined networks (SDNs). Tang et al. suggest a deep learning-based approach involving the use of self-taught learning (STL) within the benchmark NSL-KDD dataset, in the context of a network intrusion detection system. In the work of paper, an RNN-based model is implemented for the purposes of classification instead of pre-training. In addition, the NSL-KDD dataset is employed for independent training and testing sets, in order to appraise performance in pinpointing network intrusions in both binary and multiclass classifications. The results are then contrasted with J48, ANN, RF, SVM, and other machine learning methods suggested in earlier research. The study of Zhao etal. offers a cutting-edge survey of deep learning applications in the context of machine health monitoring. Experiments were conducted to contrast conventional machine learning methods with four widely employed deep learning methods (autoencoders, restricted Boltzmann machine (RBM), CNN, and RNN). This study found that deep learning methods provide greater accuracy over their conventional counterparts. In the work of Alrawashdeh and Purdy, it is suggested that using a RBM with a single hidden layer can undertake unguided feature reduction.

The weights are then transferred to another RBM in order to create a deep belief network (DBN), and the pretrained weights are moved into a fine-tuning layer made up of a logistic regression classifier (trained with 10 epochs) with multiclass SoftMax. In the study of Kim et al., a DNN using 100 hidden units is put forward, in conjunction with the rectified linear unit (RLU) activation function and the ADAM optimizer.

The study by Cordero et al. suggested another unsupervised method to train models the normal network flows. RNN, autoencoder, and dropout concepts of deep learning are employed to achieve this. The performance of these suggested methods is not fully released. Along the same lines, Tang et al. suggest a way of overseeing network flow data. In addition, Kang and Kang put forward the notion of using an unsupervised DBN to train certain features to initialize the DNN, offering greater classification performance, even though specific details of the approach are not provided. Their appraisal depicts superior outcomes when it comes to classification error detection.

In the study by Bontemps et al., a real-time collective anomaly detection model using neural network learning and feature operating was described. Here, a LSTM-RNN is trained using normal time series data, prior to making a live prediction for every time step. Furthermore, Ma et al. used the method of spectral clustering (SC) to find the key properties of network traffic, and a multilayer DNN was used to pinpoint attack types. The findings denote that superior performance was seen with the SC-DNN over the SVM, backpropagation neural network (BPNN), random forest (RF), and Bayesian methods, with the highest level of accuracy. On the contrary, weight parameters and thresholds for every DNN layer must be established experimentally and not theoretically. Erfani et al. put forward a mixed model, which used a DBN alongside a one-class SVM. An unsupervised DBN was trained to pinpoint common properties, and a one-class SVM was trained using features taken through the DBN.

A NIDS using a supervised CNN-IDS has been proposed, in which a datapreprocessing step normalizes the dataset; the CNN is trained, optimal features are extracted, and, finally, a SoftMax classifier is used to classify attacks [8]. To decrease computational costs, the traffic input vector is reconfigured into an image format. This model is evaluated using the KDD-CUP99 dataset. Although the study sees a reduction in detection time, the detection rate should be increased and feature learning should be improved for the model to learn the features with a small number of attack categories.

In, a hybrid model leverages a grey wolf optimizer (GWO) to propose a CNN for network anomaly detection, and the GWO improves initial population generation, exploration, exploitation, and revamped dropout functionality. In the first step, the GWO selects desired features to establish optimal trade-off between the two main objectives of a minimized feature set and reduced false-alarm rate. In the second step, an improved CNN (ImCNN) is utilized for anomaly classification, and the proposed model is subsequently evaluated on the DARPA98, KDD-CUP99, and synthetic datasets.

To discriminate between normal and abnormal traffic, and to auto-profile traffic patterns, D-PACK has been proposed [9]. This approach integrates an unsupervised CNN model to investigate just the first few bytes of the first few packets in each flow, therefore detecting abnormal traffic early using raw packet-level data. D-PACK is assessed using the USTC-TFC2016 dataset.

A combination of bidirectional long short-term memory (BLSTM), attention mechanism, and multiple convolutional (MC) layers has been suggested as the BAT-MC model. This approach uses the structured network traffic information to generate time series features. The MC layers extract the local features, the BLSTM generates the packet vectors, the attention mechanism screens the network flow composed of packet vectors, and a SoftMax classifier is used for final classification. This model is tested with the NSL-KDD and KDD-CUP99 datasets.

Zheng propose two convolution and pooling layers with batch normalization appended to each convolution layer to reduce computational costs and speed up detection. To determine the optimal model, different numbers of convolution and pooling layers are examined and a SoftMax classifier assesses the CNN-extracted features. Evaluation is conducted using the KDD-CUP99 dataset.

An improved CNN for wireless network intrusion detection has been proposed using stochastic gradient descent (SGD) classification and KDD-CUP99-based evaluation although this method demonstrated problems with gradient dispersion and local optima. An alternative CNN model that uses a SoftMax classifier on the KDD-CUP99 dataset is proposed and shows that increasing the number of epochs improves the accuracy of the model. In addition, this approach demonstrates that a CNN model achieves better performance as compared to SVM and DBN.

With the continuous development of big data and computing power, deep learning methods have blossomed rapidly, and have been widely utilized in various fields. Following this line of thinking, a deep learning approach for intrusion detection using recurrent neural networks (RNN-IDS) is proposed in this paper. Compared with previous works, we use the RNN-based model for classification rather than for pretraining. Besides, we use the NSL-KDD dataset with a separate training and testing set to evaluate their performances in detecting network intrusions in both binary and multiclass classification, and we compare it with J48, ANN, RF, SVM and other machine learning methods proposed by previous researchers.

v) Dataset

NSL-KDD is a dataset suggested to solve some of the inherent problems of the KDD'99 dataset. Although, this new version of the KDD data set still suffers from some of the problems discussed by McHugh and may not be a perfect representative of existing real networks, because of the lack of public data sets for network-based IDSs, we believe it still can be applied as an effective benchmark data set to help researchers compare different intrusion detection methods.

Furthermore, the number of records in the NSL-KDD train and test sets are reasonable. This advantage makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research work will be consistent and comparable.

The NSL-KDD dataset has the following advantages over the original KDD data set:

* It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
* There is no duplicate records in the proposed test sets;
* The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.
* The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable.

There are four different attack classes within the data set: User to Root (U2R), Denial of Service (DoS), Probe, and Remote to Local (R2L). Here is a brief description of each attack below:

DoS is an attack which attempts to shut down the flow of traffic from and to the target system. An abnormal amount of traffic is flooded with the IDS, which the system can not handle, and shuts down to protect itself. This prevents a network from being visited by normal traffic. An example of this could be that on a day with a big sale, an online retailer gets flooded with online orders, and because the network can not handle all the requests, it will shut down to avoid paying customers to buy anything. In the data set, this is the most prevalent attack.

• A probe is an attack that attempts to get data from a network. The objective here is to behave like a thief and steal critical data, whether it be personal customer information or banking information.

• U2R is an attack that starts with a regular user account and tries, as a super-user (root), to gain access to the system or network. To gain root privileges/access, the attacker attempts to exploit the vulnerabilities in a system.

• R2L is an attack that attempts to gain a remote machine’s local access. An attacker does not have access to the system/network locally and attempts to "hack" their way intothe network.

We can see that DoS is different from the other three attacks. DoS tries to shut down a system to stop traffic flow completely. In contrast, the other three attempts to infiltrate the system undetected.

There are:

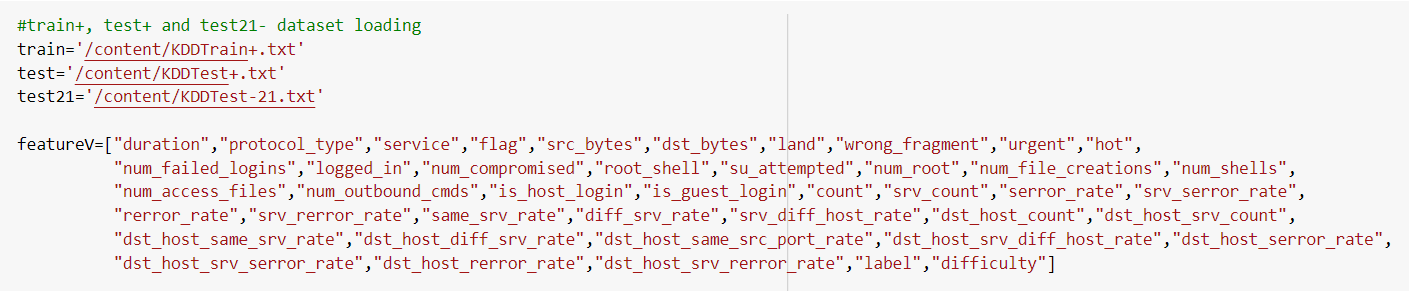
• 11 types of DoS attack

• 6 types of Probe attacks

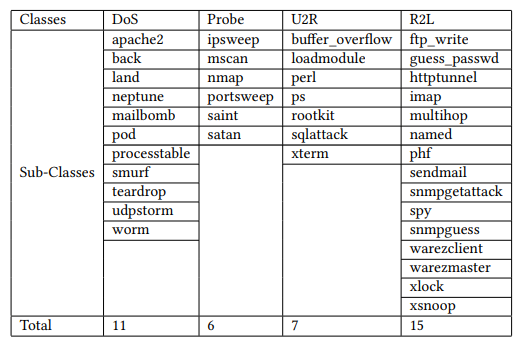
• 7 types of U2R attacks

• 15 types of R2L attacks

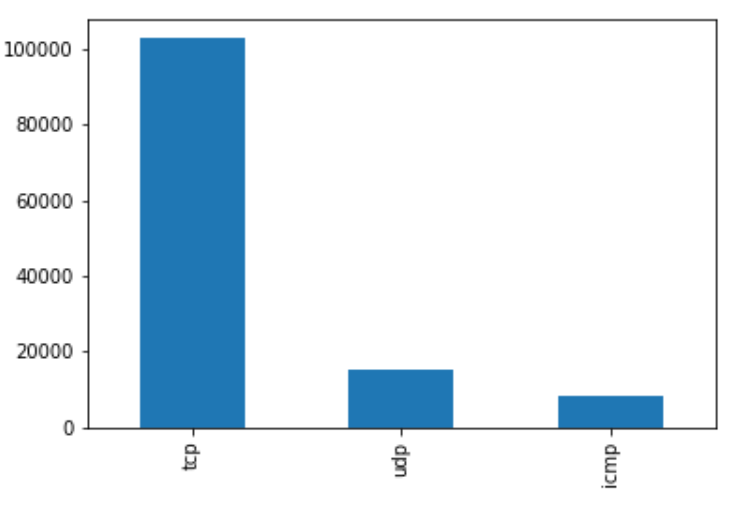
Features of NSL-KDD dataset



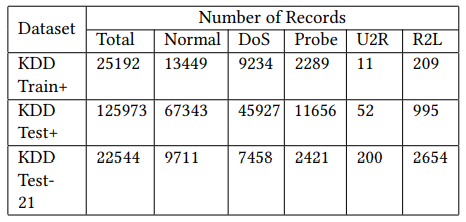
Multiclass Attacks in NSL-KDD



Plotting bar-graph of data points in feature protocol\_type



Number of records in NSL-KDD



vi) Test bed

The NSL-KDD dataset is a successor of KDD99 dataset. It consists of 1,25,973 training samples and 22,544 testing samples. Each of these samples consists of 41 features and represents some form of network activity. The 42nd column in the dataset contains the label for each sample: normal or an attack type. While the label is a categorical value, the values of features range from binary to numeric to categorical type. The attack types in the last attribute can be broadly divided into 4 categories: Denial of Service (DoS), Probe, Remote to Local (R2L) and User to Root (U2R). NSL-KDD offers several advantages over the traditional KDD dataset. The exclusion of duplicate and redundant samples from NSL-KDD allows classifiers to give unbiased results for both frequent as well as infrequent records. Also, the number of records for each category in this dataset has been balanced to improve the detection capabilities of the classifiers. Moreover, NSL-KDD contains a reasonable number of records as compared to its predecessor and that’s why this dataset can be directly used for experiments without the need to randomly select a subset of a large dataset. Since random selection of dataset may lead to inconsistencies in the results, therefore the use of NSL-KDD ensures a consistent test-bed for various researches in anomaly based network intrusion detection.

vii) Expected result

The result we acquire is that to check if the accuracy of our CNN model is better than the accuracy of the RNN model. With doing the RNN model, we expect it to have a better accuracy than the CNN model.

Architecture for CNN

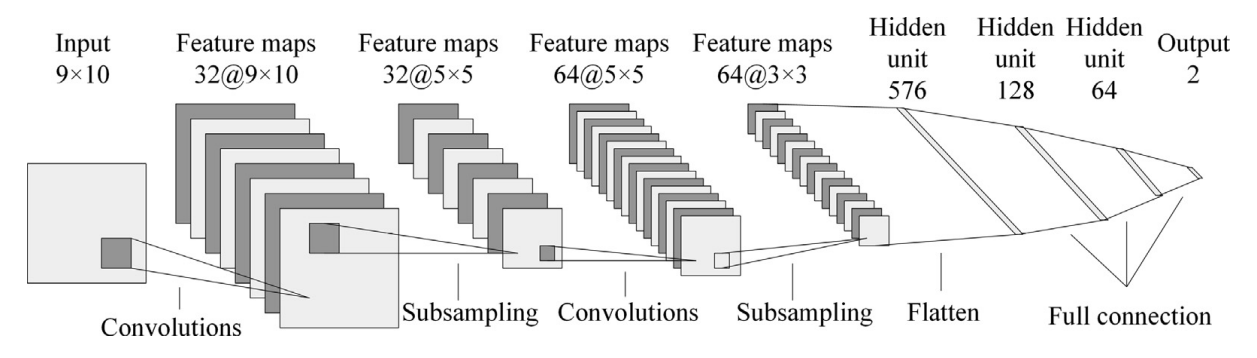
i) High level design

Society and the internet integrate deeply. This dependency is ever growing, and for that enormous quantity of data is being produced. Security of information is a significant concern and to identify attacks on networks, especially attacks that are entirely new and unseen, is a necessary need. We need An intrusion detection system that can (IDS) can identify such attacks, whether on going or an attack that has already occurred. We propose an intrusion detection system using Concurrent neural network.

Convolutional Neural Network (CNN)

CNN has two operations which are convolution and pooling. Convolution changes input data to output using a set of kernels or filters. The produced output showcases the features of the input data. That is why the output is known as the feature map. An activation function processes the convolution output further and down-sampling trims off irrelevant data using pooling. Pooling removes glitches in the data. That is how the learning improves for the following layers. CNN adjusts the kernels/filters using rounds after rounds of learning so that the feature map can functionally represent the input data. We use ID convolution since the network packet in the dataset is represented in an ID format.

Architecture of a single CNN model



In CNN, there are:

* One or more convolutional layers

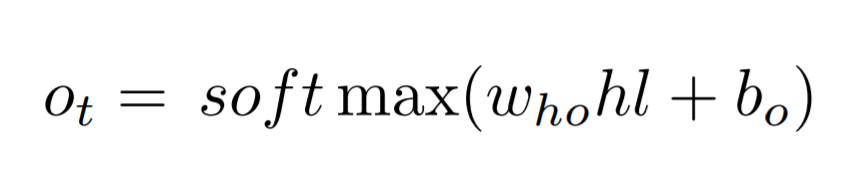
Convolution is a primary building block of CNN. Given a 1D data of network traffic time series events as input vector. Convolution1D constructs a feature map fm by applying the convolution operation on the input data with a filter w ∈ Rfd where f denotes the features in TCP/IP packets that results in a new set of features. A new feature map from a set of features is obtained where b ∈ R denotes a bias term. The filter is employed to each set of features in a TCP/IP connection records to generate a feature map.

* Pooling layers at the top

next we apply the max-pooling operation on each feature map.This obtains the most significant features in which a feature with highest value is selected.

* Fully connected layers

However, multiple features obtain more than one features and those new features are fed to fully connected layer. A fully connected layer contains the softmax function that gives the probability distribution over each class. A fully connected layer is defined mathematically as



* Dropout layers which serve as regularization layers

The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Note that the Dropout layer only applies when training is set to True such that no values are dropped during inference.

ii) Low level Design

Convolutional network or convolutional neural network or CNN is an extension to tradional feed forward networks (FFN) in the context of inspiring the biological factors. These were initially studied for image processing using Convolution 2D layers, pooling 2D layers and fully connected layer. Followed this, applied on natural language processing with Convolution 1D layer, pooling 1D layers and fully connected layer. This infers that the CNN takes image data in the form of 2D mesh and time series data in the form of 1D mesh in which the data are arranged in systematic time interval. Based on this, we model network traffic events as a time series data with a million of benign and malware connections and apply CNN and hybrid of CNN, and recurrent approaches on the same. The CNN is composed of Convolution 1D layer, pooling 1D layer, fully connected layer and non-linear activation function as ReLU.

Convolutional Neural Network (CNN) can take advantage of the 2D structure of the input data. As such a network can take an image as an input. Therefore, we avoid complicated feature extraction and unnecessary data reconstruction of traditional recognition algorithms. The modelling efficiency can be increased, and the difficulty of processing data manually can be decreased through pooling, shared weights and sparse connectivity. CNN can learn from various levels of features from a vast amount of data that is unlabeled. Therefore, the ways CNN can be used in a field such as a network intrusion detection are comprehensive

Architecture for RNN-LSTM

i) High level design

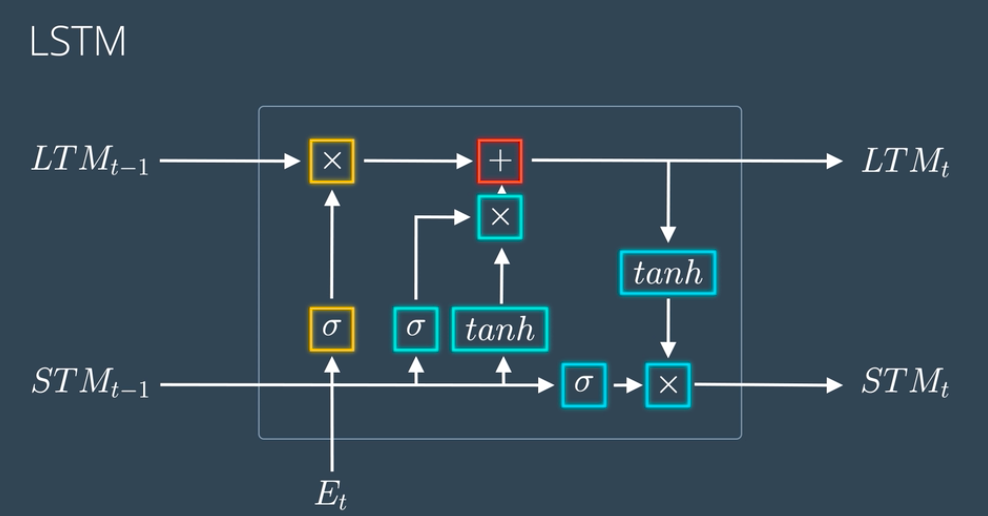
With the increasingly deep integration of the Internet and society, the Internet is changing the way in which people live, study and work, but the various security threats that we face are becoming more and more serious. How to identify various network attacks, especially unforeseen attacks, is an unavoidable key technical issue. An Intrusion Detection System (IDS), a significant research achievement in the information security field, can identify an invasion, which could be an ongoing invasion or an intrusion that has already occurred. We propose an intrusion detection system using Recurrent neural network.

Recurrent Neural Network (RNN-LSTM)

Recurrent neural networks include input units, output units and hidden units, and the hidden unit completes the most important work. The RNN model essentially has a one-way flow of information from the input units to the hidden units, and the synthesis of the one-way information flow from the previous temporal concealment unit to the current timing hiding unit. We can regard hidden units as the storage of the whole network, which remember the end to-end information. When we unfold the RNN, we can find that it embodies the deep learning. A RNNs approach can be used for supervised classification learning.

Recurrent neural networks have introduced a directional loop that can memorize the previous information and apply it to the current output, which is the essential difference from traditional Feed-forward Neural Networks (FNNs). The preceding output is also related to the current output of a sequence, and the nodes between the hidden layers are no longer connectionless; instead, they have connections. Not only the output of the input layer but also the output of the last hidden layer acts on the input of the hidden layer.

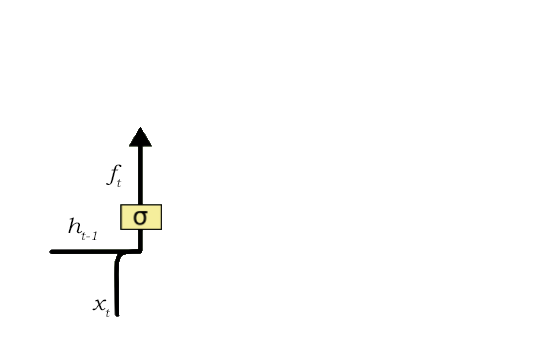
Architecture of a RNN-LSTM model



The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called gates.

* Forget Gate

A forget gate is responsible for removing information from the cell state. The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter. This is required for optimizing the performance of the LSTM network.



This gate takes in two inputs; h\_t-1 and x\_t.

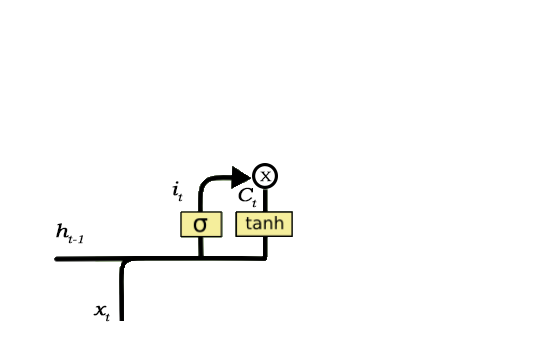
h\_t-1 is the hidden state from the previous cell or the output of the previous cell and x\_t is the input at that particular time step. The given inputs are multiplied by the weight matrices and a bias is added. Following this, the sigmoid function is applied to this value. The sigmoid function outputs a vector, with values ranging from 0 to 1, corresponding to each number in the cell state. Basically, the sigmoid function is responsible for deciding which values to keep and which to discard. If a ‘0’ is output for a particular value in the cell state, it means that the forget gate wants the cell state to forget that piece of information completely. Similarly, a ‘1’ means that the forget gate wants to remember that entire piece of information. This vector output from the sigmoid function is multiplied to the cell state.

* Input Gate

The input gate is responsible for the addition of information to the cell state. This addition of information is basically three-step process as seen from the diagram above.

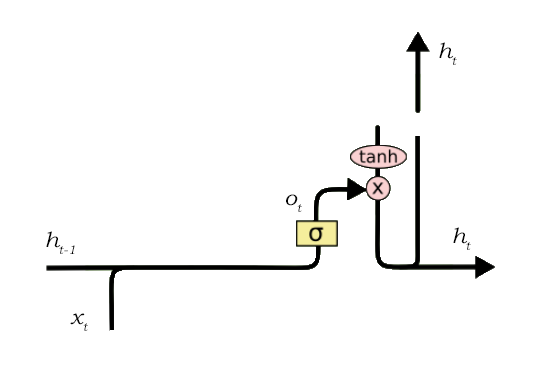
1. Regulating what values need to be added to the cell state by involving a sigmoid function. This is basically very similar to the forget gate and acts as a filter for all the information from h\_t-1 and x\_t.
2. Creating a vector containing all possible values that can be added (as perceived from h\_t-1 and x\_t) to the cell state. This is done using the **tanh**function, which outputs values from -1 to +1.
3. Multiplying the value of the regulatory filter (the sigmoid gate) to the created vector (the tanh function) and then adding this useful information to the cell state via addition operation.

Once this three-step process is done with, we ensure that only that information is added to the cell state that is *important*and is not *redundant.*



* **Output Gate**

This job of selecting useful information from the current cell state and showing it out as an output is done via the output gate.



The functioning of an output gate can again be broken down to three steps:

1. Creating a vector after applying **tanh**function to the cell state, thereby scaling the values to the range -1 to +1.
2. Making a filter using the values of h\_t-1 and x\_t, such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.
3. Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as a output and also to the hidden state of the next cell.

ii) Low level Design

Recurrent neural networks (RNNs) suffer from the vanishing and exploding gradient problem due to repeated multiplications of Jacobians during backpropagation. Due to the absence of a stable gradient signal during training, RNNs are unable to model long-term sequential dependencies. In contrast, long short-term memory networks (LSTMs) do additive––not multiplicative––updates to the cell state, and therefore avoid vanishing gradients. They can still however suffer from exploding gradients––strategies such as gradient clipping are often used to mitigate this problem.

LSTMs (and GRUs) can model long-term sequential dependencies, so they are generally used over RNNs unless there is a compelling reason not to. As a side note, temporal convolutional networks (TCNs) have been shown to be competitive or even better than LSTMs and GRUs at sequence modeling tasks. This is because TCNs have altered backpropagation paths in comparison to those in RNNs; the shortest path between an input unit xi and output unit oi is logarithmic instead of linear due to the use of dilated convolutions.

IMPLEMENTATION FOR CNN

DATA PREPROCESSING

Numericalization

In the NSL-KDD dataset, there are three features which are nonnumeric and 38 numeric features. Since the input values must be numeric, we convert the non-numeric features into numeric. For example, the feature ’protocol\_type’ can have three different types of attributes which are ’tcp’, ’udp’, and ’icmp’. We encode them as binary vectors (0,0,1), (0,1,0) and (1,0,0). This way, we convert the 41-dimensional feature map into a 122-dimensional feature map.

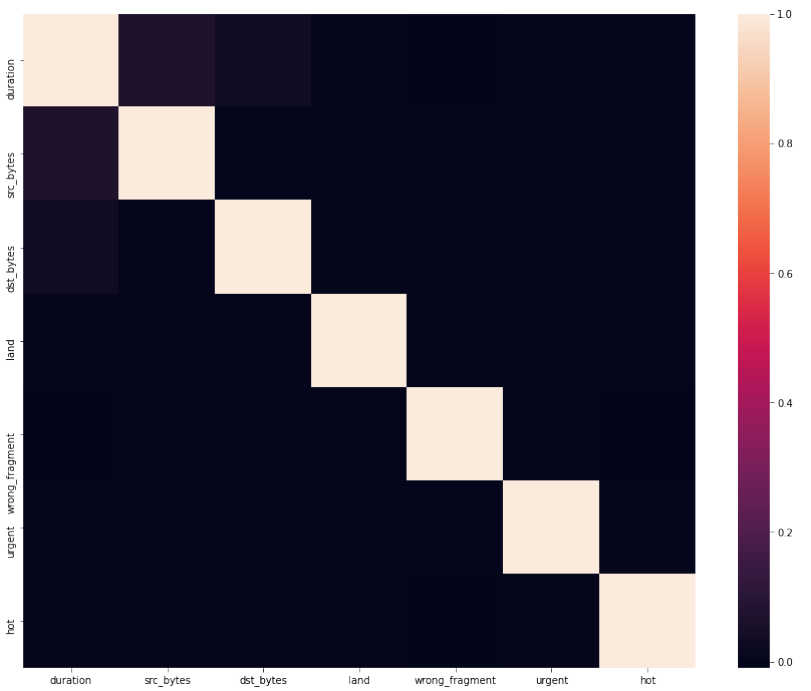
Normalization

There are several features in the dataset in which the difference between the max and min values are large. Such features are dst\_bytes [0,1.3 × 109], src\_bytes [0,1.3 × 109] and duration[0,58329]. We apply the logarithmic scaling method to lower the differences and then use the formula below to map them to the [0,1] range: xi = (xi - Min) / (Max - Min)

Feature Selection

The features in a traffic record provide the information about the encounter with the traffic input by the IDS and can be broken down into four categories: Intrinsic, Content, Host-based, and Time-based. We have considered the whole dataset for training as a four part dataset for classifying the attacks. There are many highly correlated features which hampers the model in classifying in the validation and testing time. So, we split the dataset in four part and took the first part as hybrid features.

Selecting the most relevant features of NSL-KDD



Intrinsic features can be derived from the header of the packet without looking into the payload itself, and hold the basic information about the packet. First 10 features of each packet gives the model more generalization and inference ability which helped us to go through all this way with a better accuracy than other models.

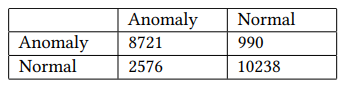
Evaluation Metrics

We use Accuracy (AC) to measure the performances of our model. Furthermore, we also introduce false positive rate and detection rate. True Positive (TP) denotes the number of records that rejects correctly and identifies as anomalies. Whereas, True Negative (TN) denotes the opposite. True Negative (TN) denotes the correct records that are normal and False Negative (FN) denotes the opposite.

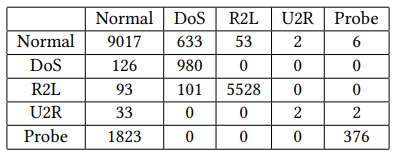
Binary Classification

We have mapped 41-dimensional features into 83-dimensional features. Therefore, the CNN-IDS model has 122 input nodes and 2 output nodes in the binary classification experiment. We take the number of epochs as 100 and the learning rate as 0.01. To find the better model, let the number of hidden nodes be 60, 80 and 120, respectively. The no. of hidden layers is 2, and the batch size is 64. From the table below, we have determined that a hidden node value of 64 achieves the best result.

Confusion matrix of 2-category classification on KDDTest+



Confusion Matrix for Multiclass Classification



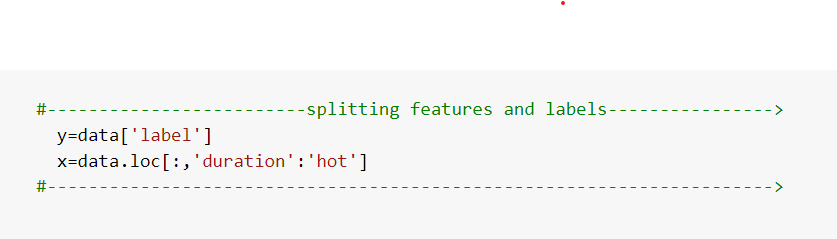
In this experiment, the detection rate of the CNN-IDS model gets higher accuracy testing dataset, not only higher than the detection rate on the NSL-KDD dataset, but also higher than other neural network models. The experimental results show that the fully connected model has stronger modelling ability and higher detection rate than the reduced-size CNN model.

Data validation

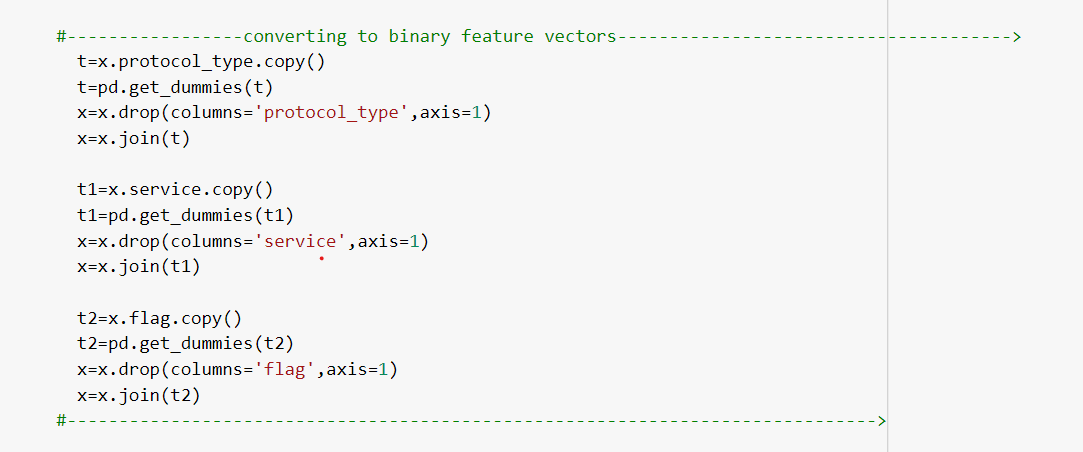
We are given two types of classification: binary classification and multiclass classification. We also set the attack category as the label column since that is the main accuracy we need to determine.

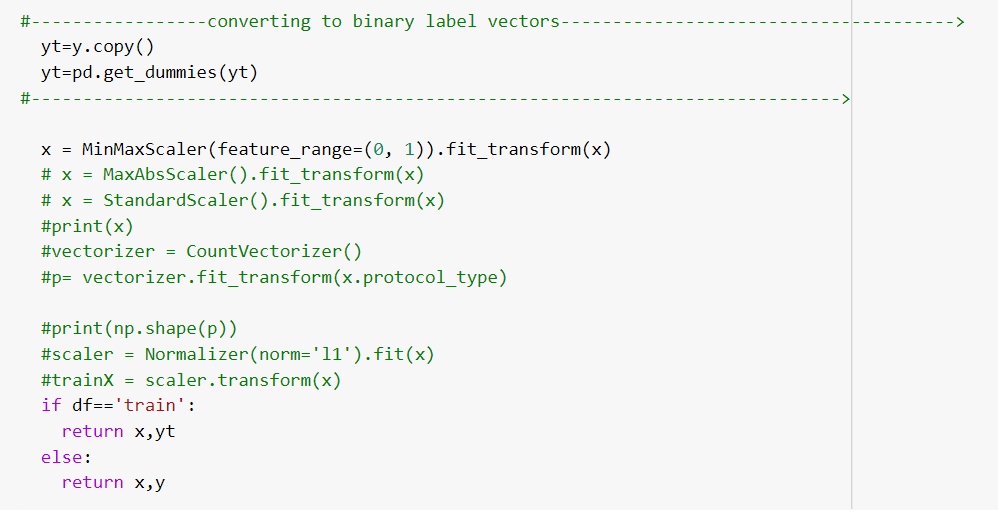


We then split the features and the labels into two separate variables.

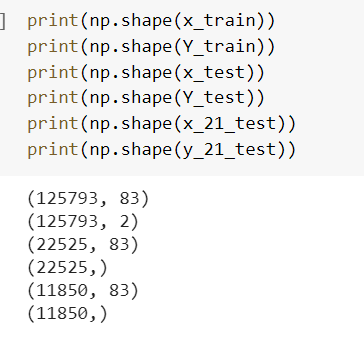


Then we are converting features and labels into binary classification for easy comparison.

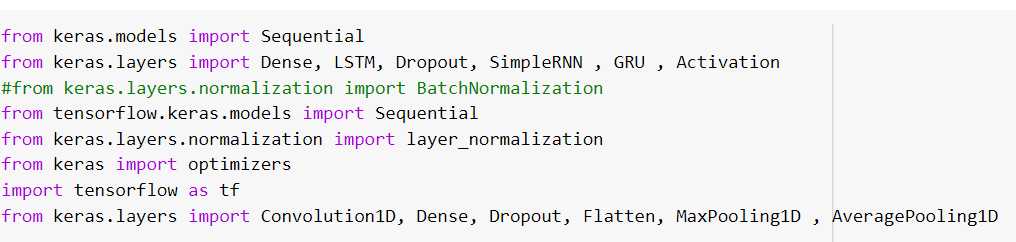




We then examine the three datasets we have taken and find the shape of the dataset (That is how many rows and columns)



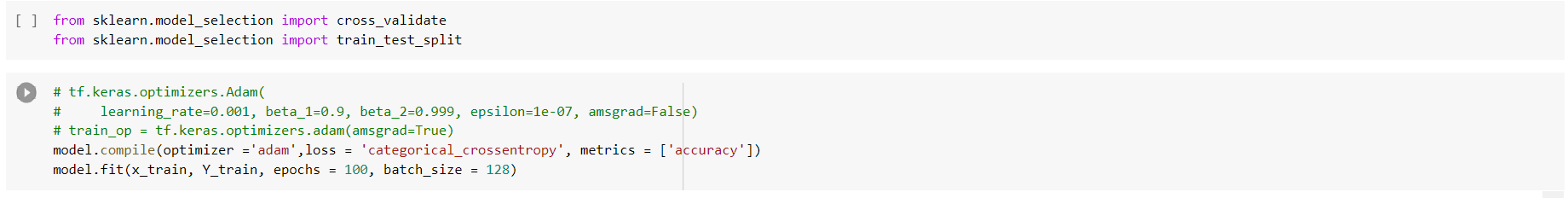
Then we start the creation of the model by importing the headers required to run the model.



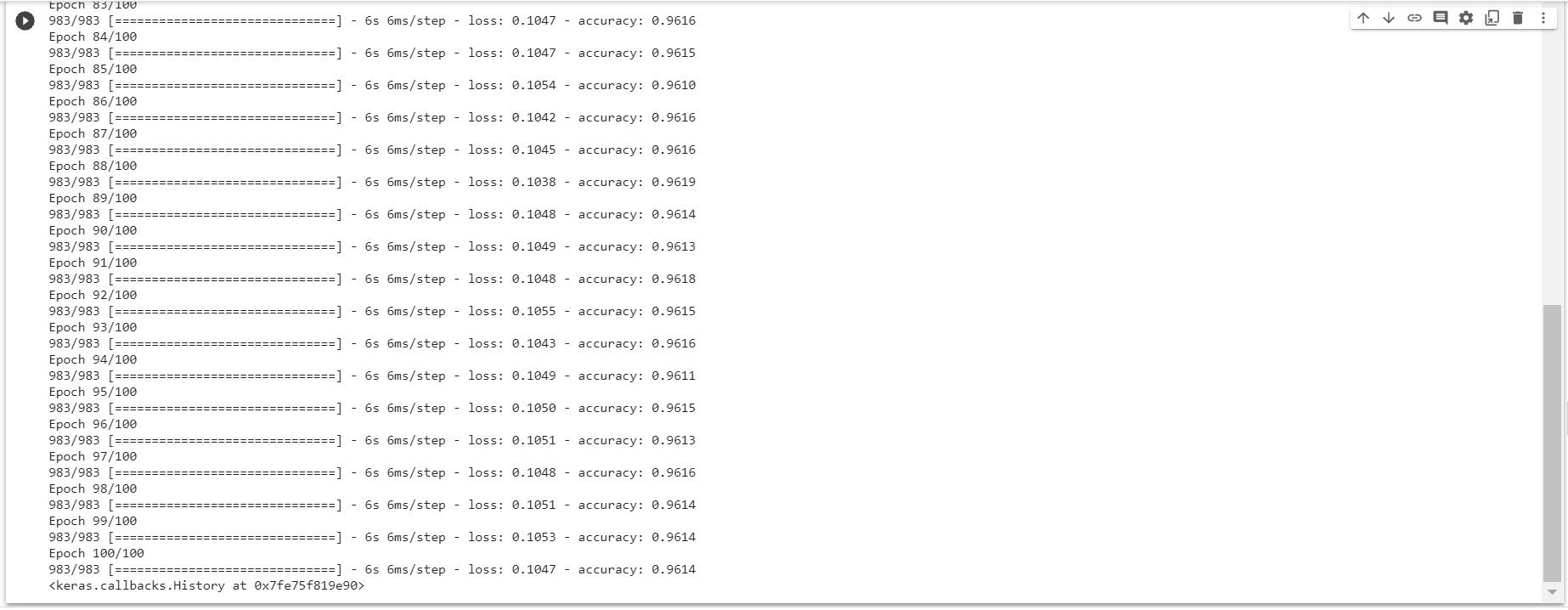
We add MaxPooling 1D and Convolution 1D into our model which has an activation called relu. These are the first layers or the initial layers we add to our model. Then we add the Dense layers into our model with activation as relu and softmax. These are considered as the hidden layers in our model.



We then import the necessary headers and run our train model with an epochs of 100 and a batch size of 128.

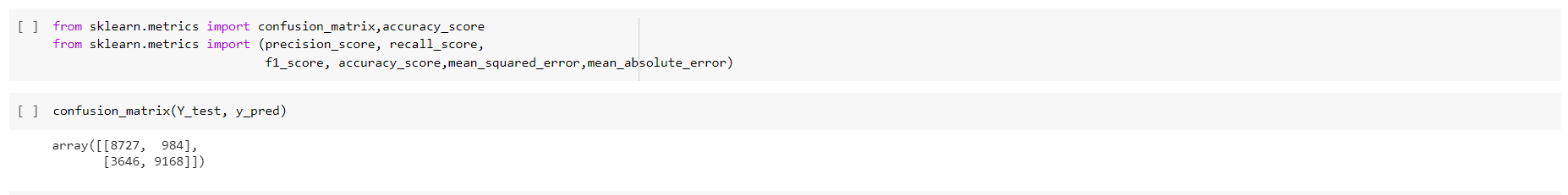




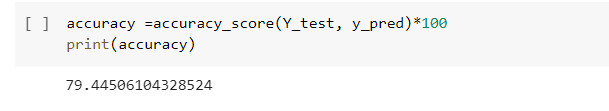


For the first epoch we get a loss of 0.1772 and an accuracy of 0.9416 and for the last epoch (100th epoch) we get a loss of 0.1047 and an accuracy of 0.9614.

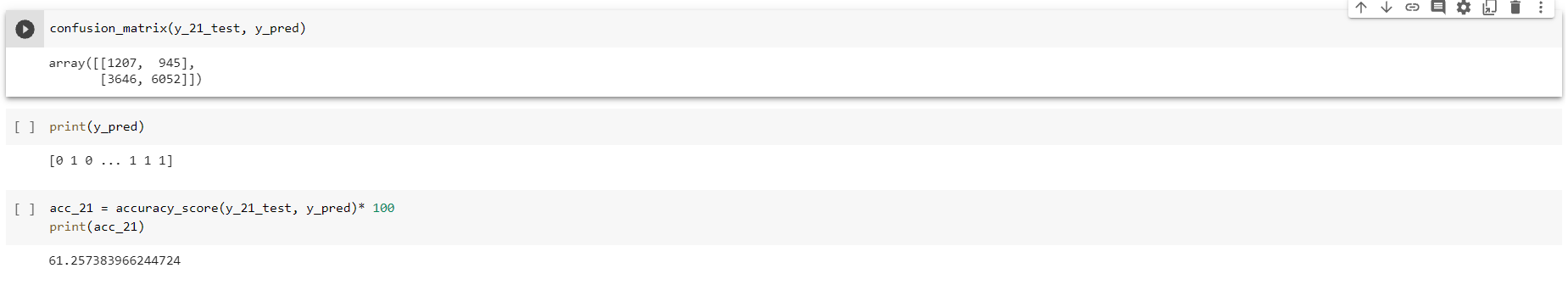
Now we start evaluating our model.



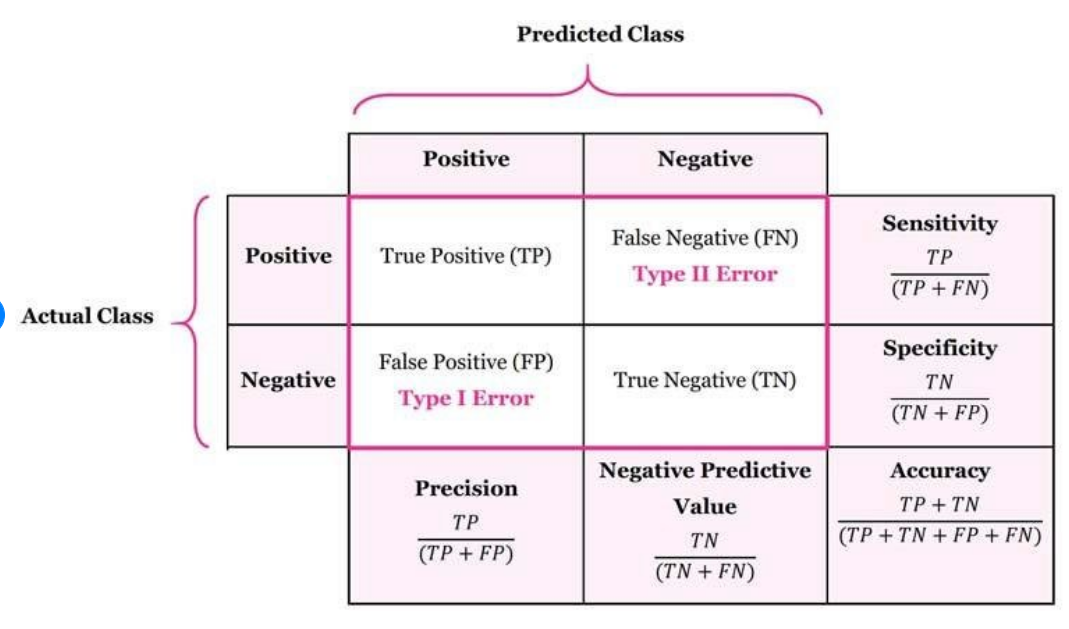
We import the necessary headers and produce a confusion matrix between y\_test and y\_pred and we print the accuracy score of the matrix



We do the similar features for y\_21\_test and y\_pred



We use Accuracy (AC) to measure the performances of our model. Furthermore, we also introduce false positive rate and detection rate. True Positive (TP) denotes the number of records that rejects correctly and identifies as anomalies. Whereas, True Negative(TN) denotes the opposite. True Negative (TN) denotes the correct records that are normal and False Negative (FN) denotes the opposite.

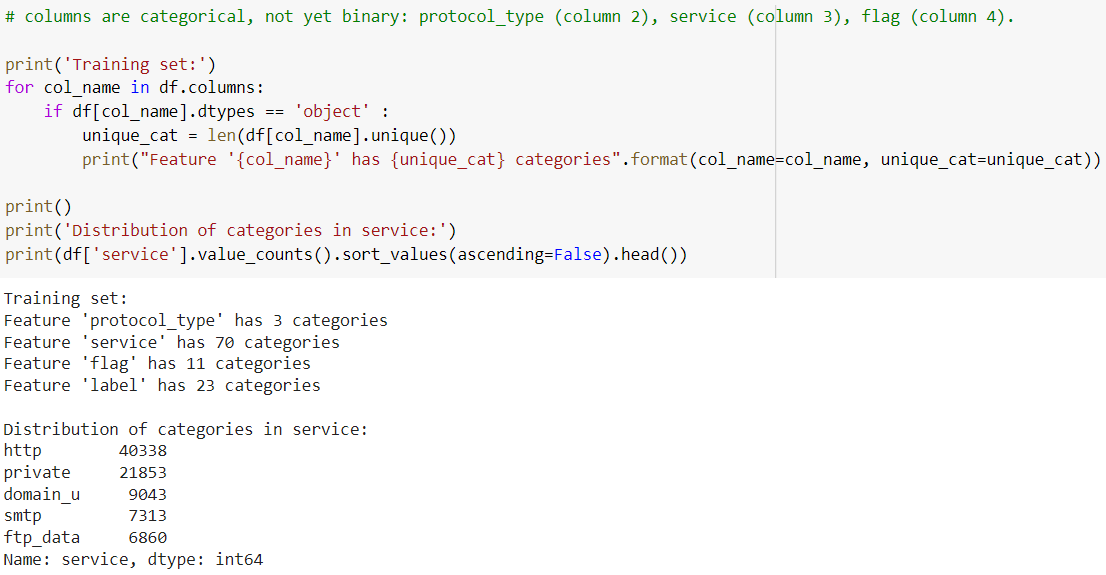


Therefore, our goal is to get high accuracy and better detection rate with low false positive.

IMPLEMENTATION FOR RNN

DATA PREPROCESSING

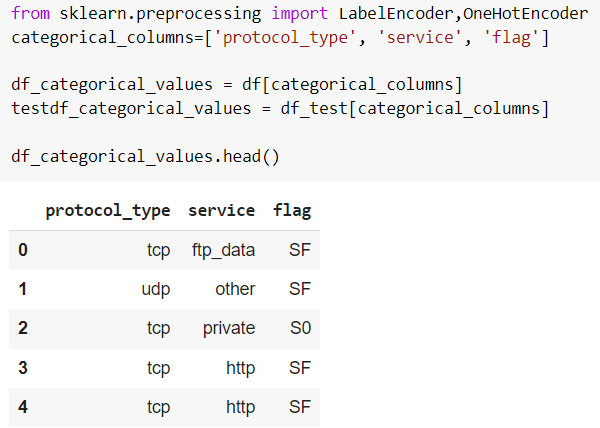
One-Hot-Encoding is used to convert all categorical properties to binary properties. One-Hot-Endcoding requirement, the input to this transformer must be an integer matrix expressing values taken with categorical (discrete) properties. The output will be a sparse matrix in which each column corresponds to a possible value. It is assumed that the input properties have values in the range [0, n\_values]. Therefore, to convert each category to a number, properties must first be converted with LabelEncoder.



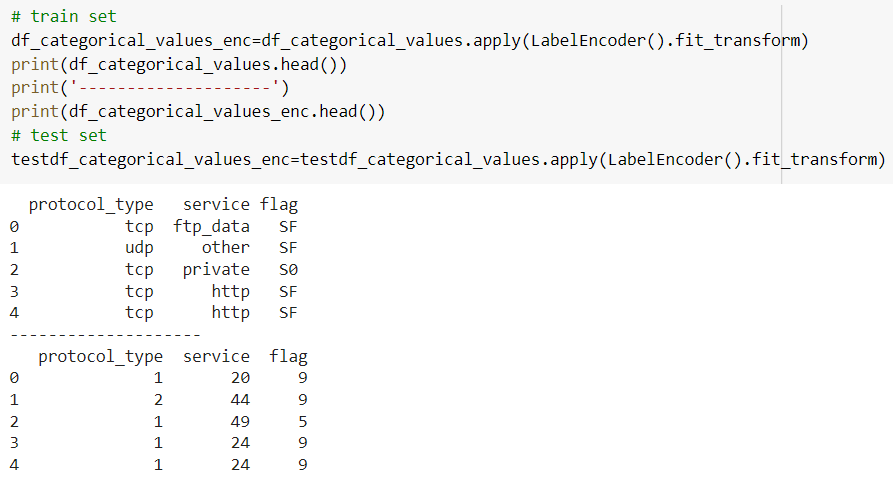


Label Encoder-

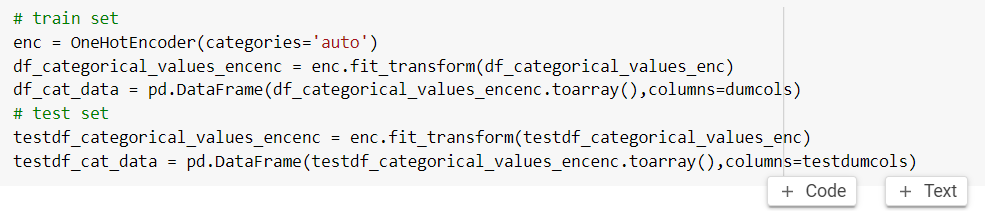
Insert categorical features into a 2D numpy array



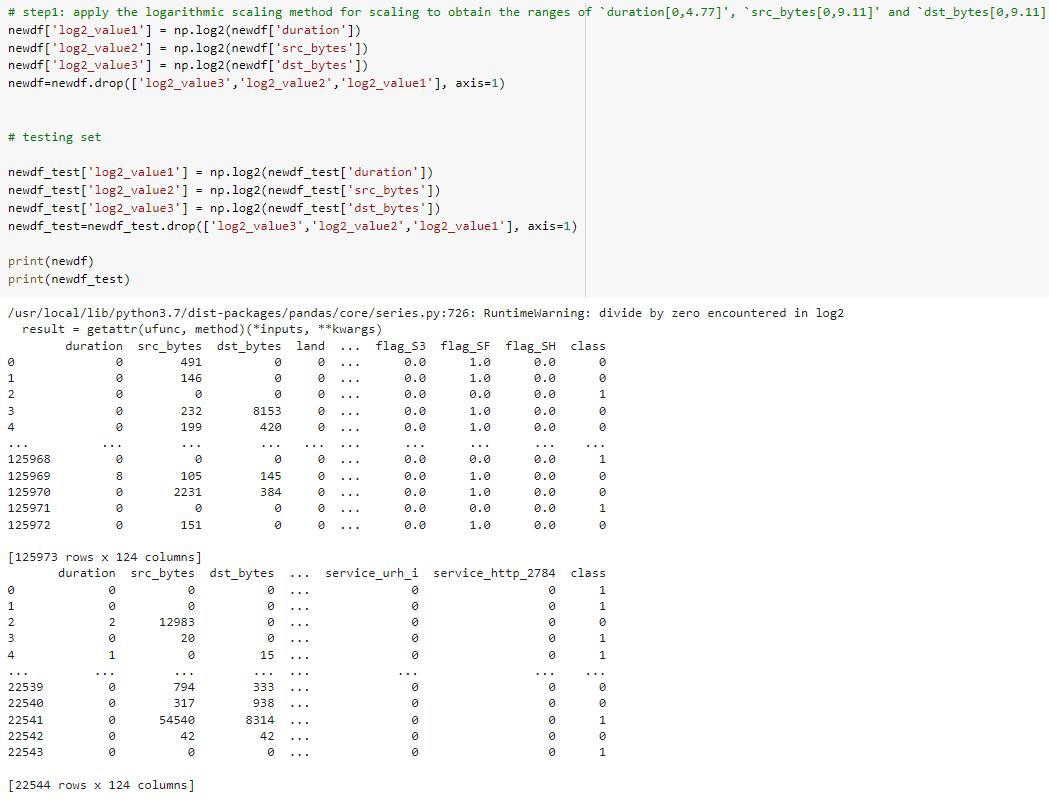
**Transform categorical features into numbers using Label Encoder()**



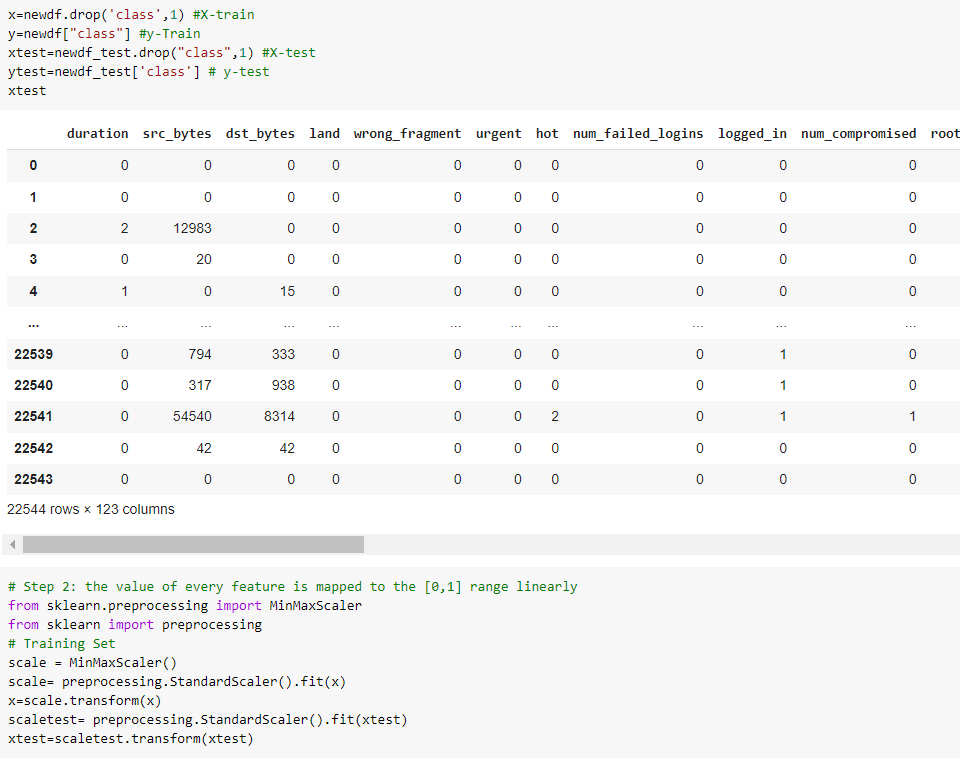
One-Hot-Encoding



Feature scaling

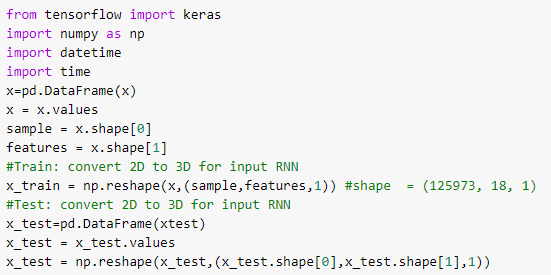


Split the training split and testing split



Input Layer

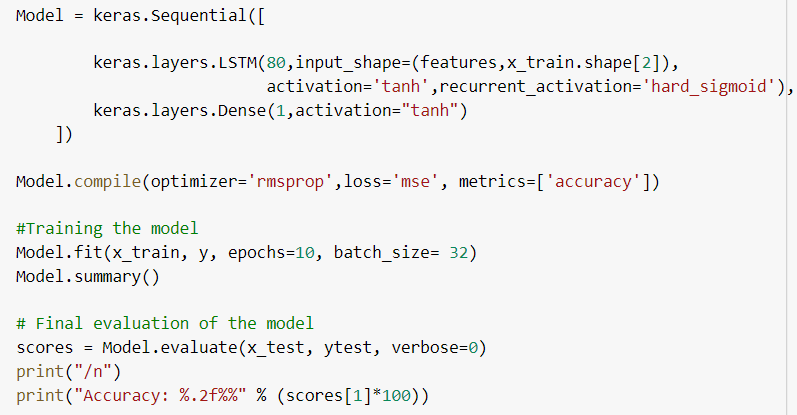
LSTM input layer must be 3D the meaning of the 3 input dimensions are: samples, time steps, and features. The number of samples is assumed to be 1 or more. reshape() function takes a tuple as an argument that defines the new shape. number\_of\_rows\_to\_process\_each\_loop, the\_time\_interval\_for\_next\_move(e.g. per day, per month), column.

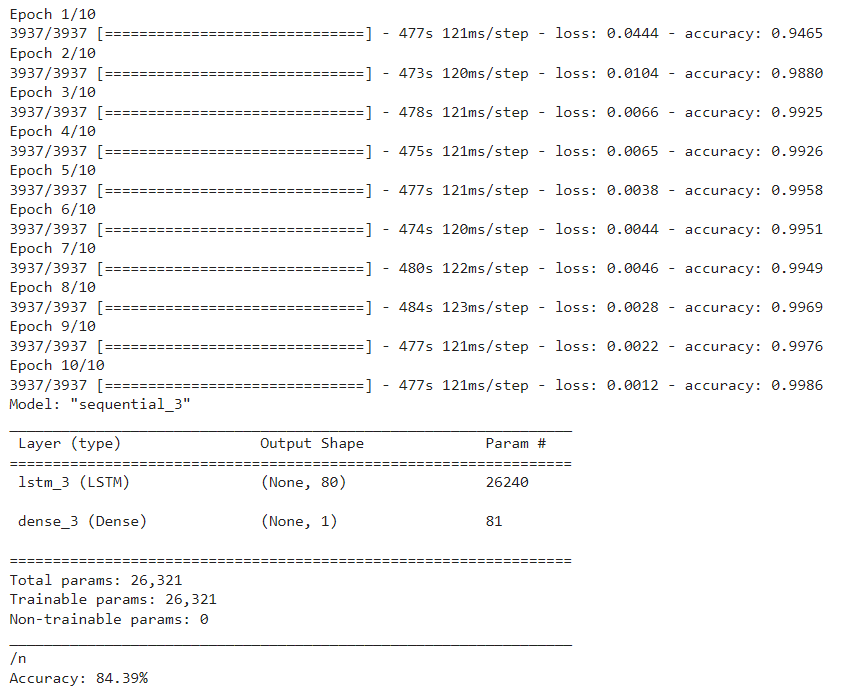


RNN-LSTM model and training

number of neurons in the hidden layer is 80 neurons

Activation function: I used the hard\_sigmoid in the hidden layer and tanh in the output layer





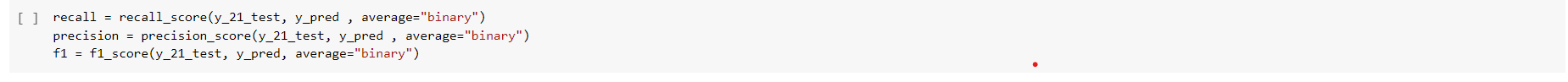
Results and Discussions for CNN

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

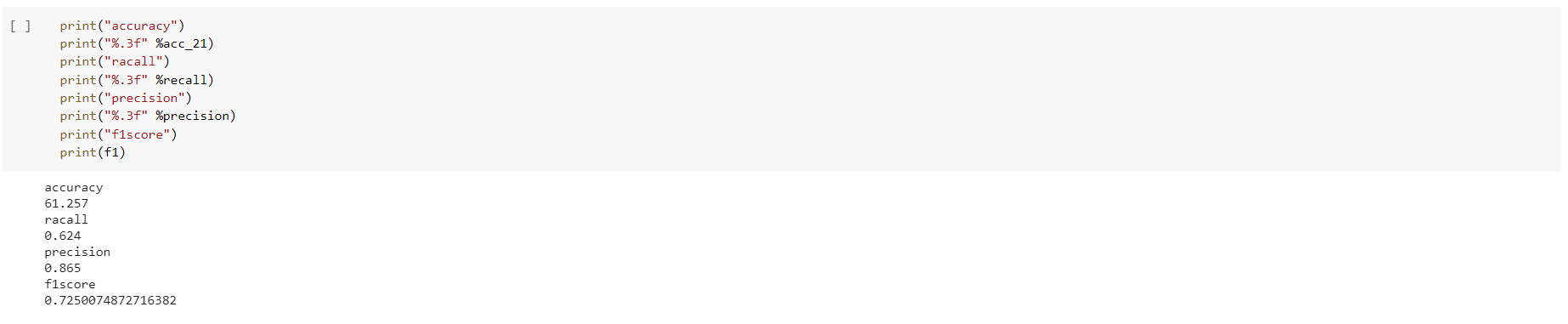
Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it’s better to look at both Precision and Recall.

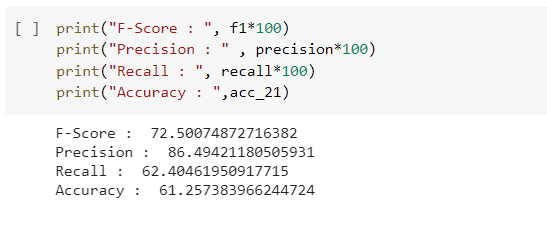
F1 Score = 2\*(Recall \* Precision) / (Recall + Precision)



We print the accuracy, recall, precision and the f1 scores.



We then filter the scores with a two decimal point to understand the scores better with our model.



The f-score, precision score, recall score and the accuracy helps us understand the model using Convolution Neural Network (CNN). We find that the accuracy is 61.257383.

Future works:

We have performed the model using CNN and we need to create and model using Recurrent neural network (RNN) with the NSL-KDD dataset. This helps us understand both the CNN and RNN models in intrusion detection and we are able to compare the two accuracies to find out which has the better accuracy.

References -

<https://www.researchgate.net/publication/338455935_An_Intrusion_Detection_Model_based_on_a_Convolutional_Neural_Network>

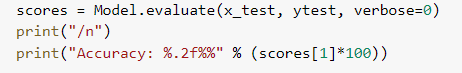
<https://www.hindawi.com/journals/scn/2020/8891185/>

<https://onlinelibrary.wiley.com/doi/full/10.1002/ett.4150>

<https://www.liebertpub.com/doi/10.1089/big.2020.0263>

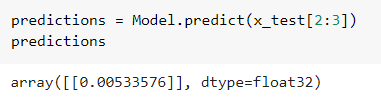
Results and Discussions for RNN

Final evaluation of the model

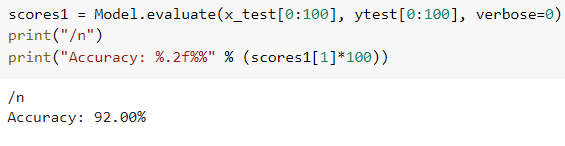




Prediction for 3rd label



evaluation of the model for first 100 labels



REFERENCES:

<https://ieeexplore.ieee.org/abstract/document/8066291>

<https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/>

<https://www.quora.com/When-should-I-use-an-LSTM-over-an-RNN>