**Name – Santosh Pandeya**

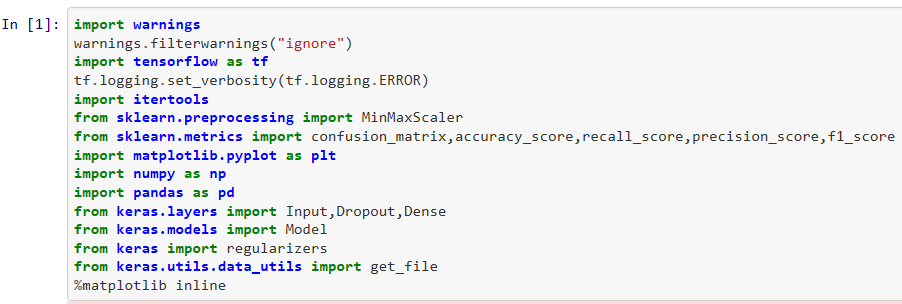
**Email –** [**skpandey@techmahindra.com**](mailto:skpandey@techmahindra.com)

**Improving Network Intrusion Detection using a Denoising Auto encoder with Dropout**

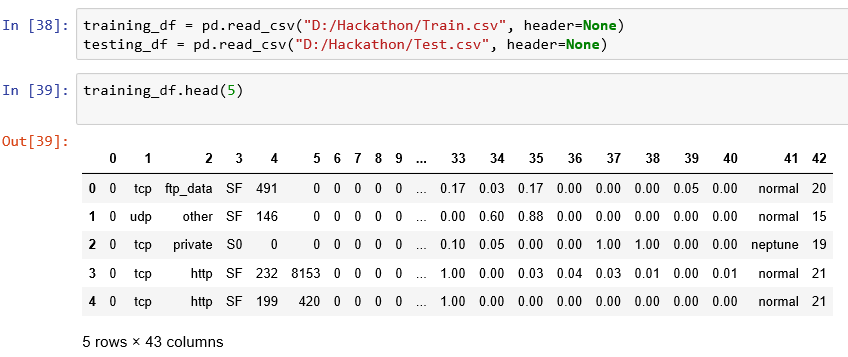
This document presents an auto encoder based anomaly detection model for intrusion detection, I used the KDDCUP’99 dataset. In the KDDCUP’99 training dataset only 0.04% of the samples belong to the u2r attack type making it severely underrepresented, the case is similar for the r2l and probe attack types whereas the majority of attack records are representing the DDOS attack type, this fact made it difficult for classifiers to detect these underrepresented types resulting in poor accuracy.

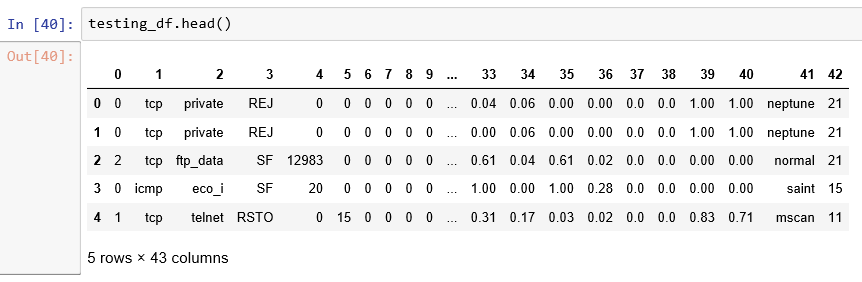
Another issue is that this dataset is unrealistic, in reality most traffic in a network is benign and only a small percentage might be malicious, while in the KDDCUP’99 training set for example, attack samples compose 80% of the entire dataset which makes the models trained using this dataset ineffective in real life situations. Out auto encoder based approach attempts to overcome these problems.

The following section makes the necessary imports:

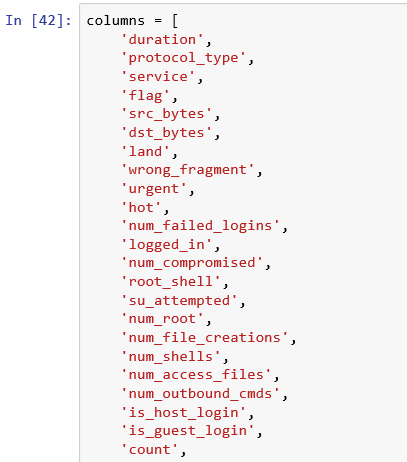


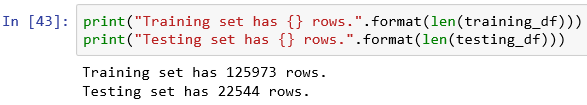
**Loading the data**

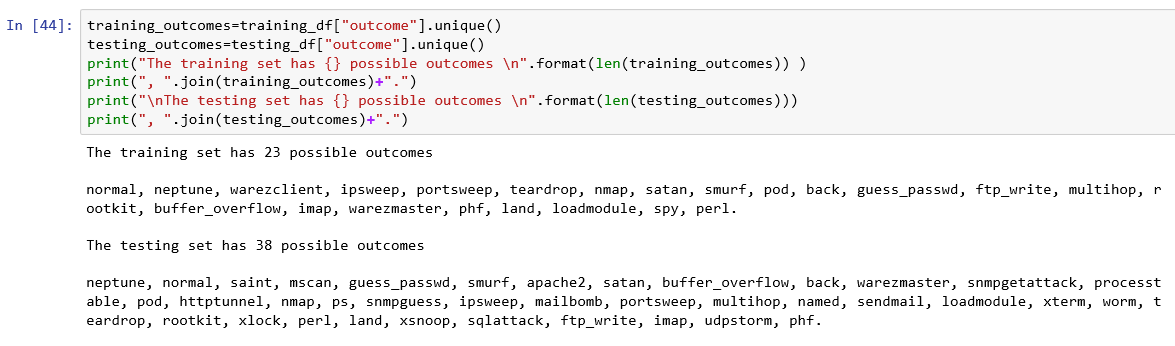




Since the CSV files don't contain a header we'll need to assign column names ourselves.







**Preprocessing**

**Extracting the labels**

As depicted previously the testing set has an additional 15 attack types that are not available in training data hence we will need more general labels to train the model for the classification task.

The 37 attack types available in the dataset can be clustered into four general attack types

Denial of service attacks

Remote to Local attacks

User to Root

Probe attacks

Our model will perform binary classification of the data to two classes indicating whether the traffic is normal or an Attack, however we will use the four attack types to analyze the results and calculate performance metrics for each general attack type.

The next section replaces the current outcome field with a Class field that has one of the following values:

Normal

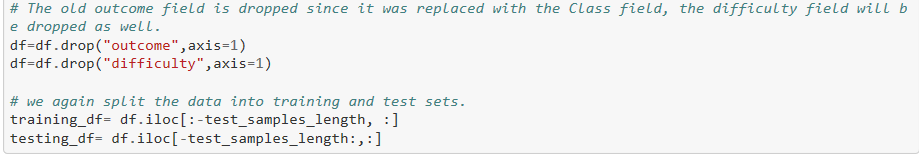
Dos

R2L

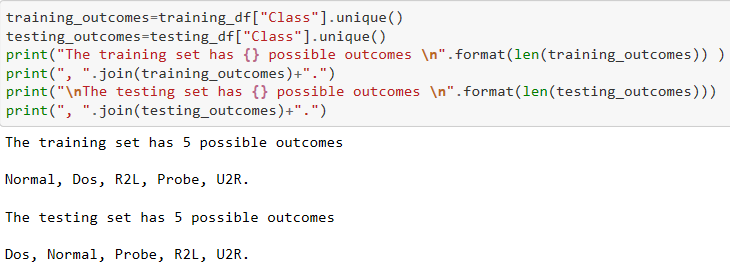
U2R

Probe





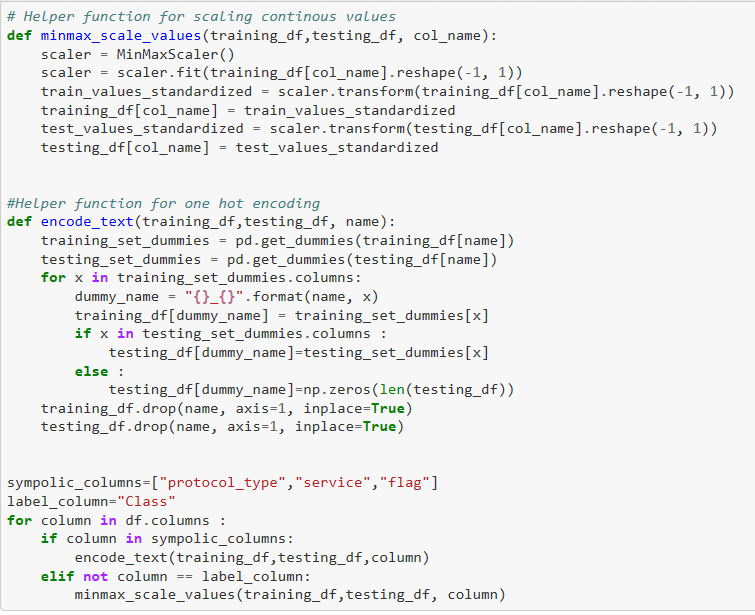
Take a look at the new labels

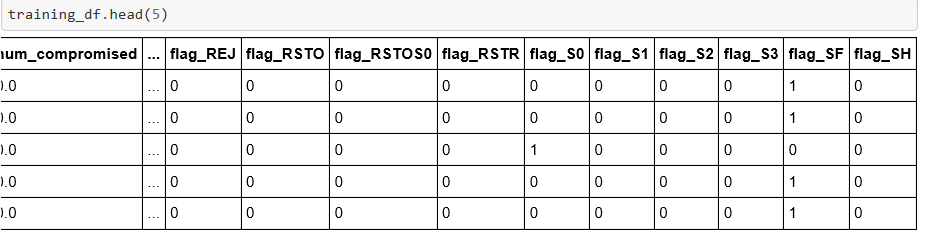


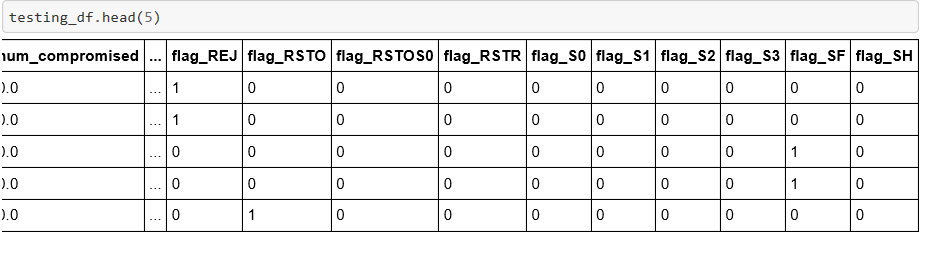
**Preparing the Features**

For continuous features we use the MinMaxScaler provided by the scikit-learn library, we only allow the scaler to fit the training set values and then we use it to scale both the training and testing sets. The minmax\_scale\_values helper function does this task.

As for the discrete features we use one hot encoding. The encode\_text function achieves this.

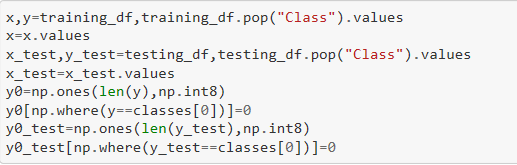


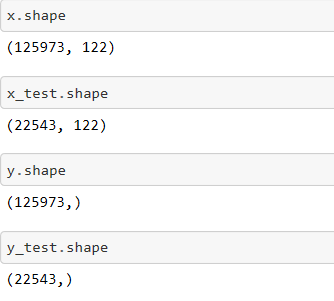




I will extract the values from the pandas dataframes as Numpy arrays, where:

* x holds the features of the training dataset
* y holds the classification of the training dataset to one of the five possible values
* x\_test holds the features of the testing dataset
* y\_test holds the classification of the testing dataset to one of the five possible values
* y0 holds the classification of the training dataset to one of two possible labels, 0 for normal traffic or 1 for an attack
* y0\_test holds the classification of the testing dataset to one of two possible





**The Model**

In order to avoid the imbalance of the samples representing each attack type in the training data, and to avoid the model’s inability to learn about new attack types by observing existing ones, we present an approach that utilizes auto encoders and reconstruction error to detect anomalies.

**Architecture and Training**

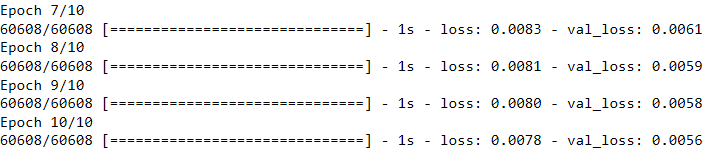
In this approach I implemented a sparse autoencoder with dropout on the inputs, it consists of an input layer of 122 neurons due to the fact that the number of features for each sample is 122 followed by a dropout layer and a hidden layer of 8 neuron units so the hidden representation of the autoencoder has a compression ratio of 122/8 forcing it to learn interesting patterns and relations between the features, finally there is an output layer of 122 units, the activation of both the hidden layer and the output layer is the relu function.

The autoencoder was trained to reconstruct its input, in other words it learns the identity function, the model was trained using only the samples labeled “Normal” in the training dataset allowing it to capture the nature of normal behavior, this was accomplished by training the model to minimize the mean squared error between its output and its input.

The regularization constraints enforced over the autoencoder prevent it from simply copying the input to the output and overfitting the data, furthermore the dropout presented on the inputs makes the autoencoder a special case of a denoising autoencoder, this kind of autoencoders is trained to reconstruct the input from a distorted corrupted version of itself, forcing the autoencoder to learn even more properties of the data.

The model is trained for 10 epochs using an Adam optimizer with a batch size of 100, furthermore we held out 10% of the normal training samples to validate the model.





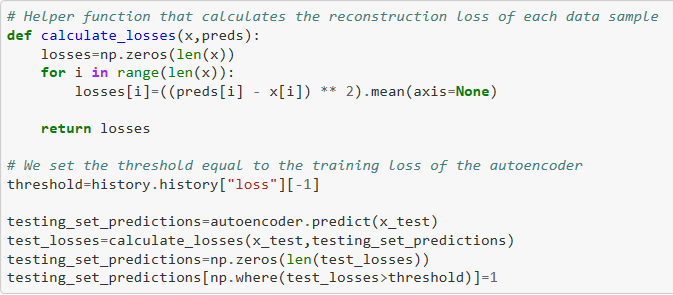
**Prediction**

The model performs anomaly detection by calculating the reconstruction error of samples, since the model was trained using normal data samples only the reconstruction error of samples that represent attacks should be relatively high compared to the reconstruction error of normal data samples, this intuition allows us to detect attacks by setting a threshold for the reconstruction error, if a data sample has a reconstruction error higher than the preset threshold then the sample is classified as an attack, otherwise it’s classified as normal traffic.

For the choice of a threshold two values can be helpful for guiding the process, the model loss over the training data and over the validation data, we found by experiment that a choice around these values produces acceptable results, for our experiments we use the model loss over the training data as a threshold.

Due to the nature of this approach it can only be used for 2-Class classification as it is purely for anomaly detection and not classification.

The following section evaluates the performance over the testing dataset, the calculate losses is a helper function that accepts the original features and the predicted features (the auto encoder’s output) and returns the reconstruction loss of each data sample, afterwards each data sample is classified according to its reconstruction error and the preset threshold.



**Evaluation**

To evaluate the model we calculate the following performance metrics:

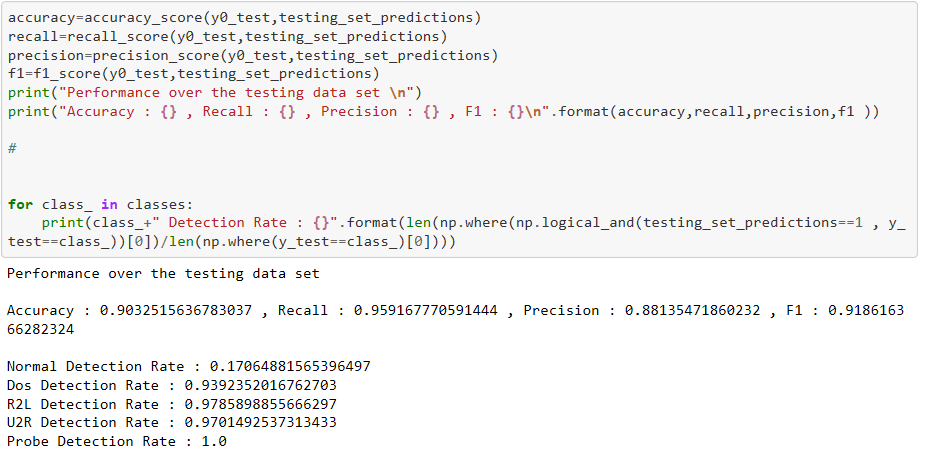
Accuracy

Recall

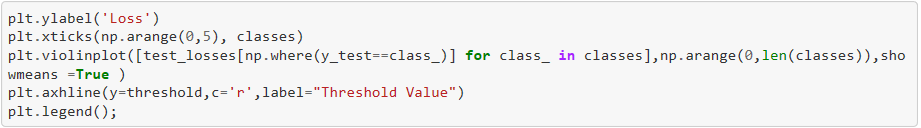
Precision

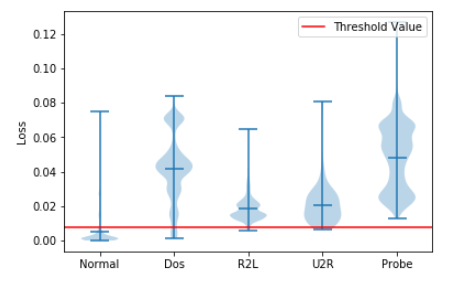
F1 Score

Detection rate for each of the five possible labels.



The following violin plot shows the distribution of reconstruction loss values for all data samples in the testing set, it clearly shows that the loss values of attacks are mostly higher than the threshold value, the opposite is true for normal samples.





**Conclusion**

In this approach I attempted to overcome the problems that exists in the KDD99 datasets, namely the class imbalance issue and the data being unrealistic, by avoiding the attacks data during training, the model was trained only using normal traffic, so it was not affected by the class imbalance of the dataset, in addition the fact that it only uses normal traffic data for training makes it more valuable in real world applications and more viable for use in real networks.

Another strength of this approach is its simplicity, it consist of only a single hidden layer of 8 neurons making it very easy to train. During evaluation we avoided human manipulation of the threshold in order to achieve reproducible results without human interference, however in real networks when the system is deployed the network administrator can manually adjust the threshold which allows a tradeoff between sensitivity and specificity according to the network requirements which is a huge advantage over other existing approaches. In terms of detection rates our approach outperforms every method in the existing literature that we are aware of.

The obvious limitation of our approach is that it can only differentiate between normal and attack traffic, so classifying attacks to the different attack types is not possible, future work can be done to overcome this limitation by building an ensemble of the model alongside others that extend its functionality in order to achieve 5-class classification.