Implementation Details – kNN Classifier:

In pattern recognition, k-nearest neighbors’ algorithm (k-NN) is a non-parametric method used for classification and regression purposes. In both cases, the inputs consits of k closest training examples in the feature space, in the case of classification, the output is a class membership. An object is classified by plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

In this classification task, Fire-wall Action is determined by 11 inputs such as NAT source ports, elapsed time, and packet size; and outputs 1 out of 4 possible action by the firewall which is ‘Allow’, ‘Deny’, ‘Drop’ or ‘Reset-Both’. All the inputs given by the datasets are considered into the training model as the goal of the Firewall is to be absolute strict in considering anonymous packet transfers to avoid malicious packets from getting pass. As such, all inputs are counted as vital in deciding the action that will taken by the firewall.

After data initialization, we can check the datasets for any existance of null or NoneType values, thankfully the datasets come in whole. The datasets have in total 65532 entries with the majority of the action given by the firewall as ‘Allow’.

Before processing the data, it must be known that the output is not a binary value, as it comes with 4 unique values. Thus, we must consider a multioutput or multilabeling approaches.

The only categorical value is the ‘Action’ attribute, which is also our output for this dataset. Thus, we will have to encode it into machine-readable value. One of the methods of encoding used is the ‘One Hot Encoding’ algorithm in which indivual actions are categorize into a matrix format, the reason OHE is used is to avoid the training-learning algorithm to take the value as a predictive measure. After normalizing the data with OHE encoded ‘Action’ attribute, we can then proceed to train the model.

The datasets X is the 11 attribute that counts towards the training model and y is the output actions determined by the firewall, with y\_cat as its categorical counterpart for easy reading. The datasets are then splitted into 7:3 training-testing ratio, which is an optimal value recommend by many papers.

The model, action\_model is an instance of the kNearestnNeighborsClassifer with parameters at its default settings. After running a prediction, the model shown have an impressive accuracy of 0.9947, as expected from a Firewall System. However, to trutly validate the data we must run a k-fold cross validation test and construct a confusion matrix from it.

Upon running a 5-fold validation test, the scores given by the test are very close to the accuracy of the action\_model. From the looks of the confusion matrix, the recall and precision also mark in at around 0.994, thus leaving the f1 score of the overall system also at 0.9944.

Normally, both the ROC and PR graphs lacks the supports for multioutput models, thus we must define our own visualization, which can be implemented by reading several papers from the internet. The PR curve shows an average of 0.99 precision while the ROC AUC score clocks in at 0.8756.

Next, our model can be fine tuned to optimize on the ROC AUC score by using a Grid Search on some of parameters of the kNN-Classifier: n\_neighbors, the amount of neighbors that the model will sample from; p which is either in 1 or 2 to determine the distance calculating method for the model; and the weights used for sampling.

After grid searching, the hyperparameters setting for the best model are as followed, with n\_neighbors at 5, p settings at 1 which is the mahattan distance and a uniform weight used for sampling.

Finally, we can evaluate our model using the testing sets, which gives us an overall ROC AUC score of 0.8722, which is expected to be slightly worse than the training sets as the model is fine-tuned to the validation data.