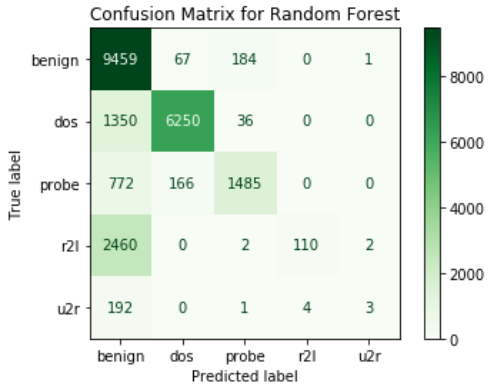
SIT719 Security and Privacy Issues in Analytics

Higher Distinction Task 5.1 End-to-end project delivery on cyber-security data analytics

Classification is the process of dividing the data elements into specific classes based on their values. It is a type of supervised learning which means data are labeled. This project “Cyber-attack classification in the network traffic database” uses commonly used machine learning classification algorithms to solve the problem by identifying the normal network traffics and attack classes. We know there are 5 different classes so this comes under the classification problem where classification algorithms can be used to identify different classes based on attributes. In this report, I will discuss the various classification algorithms like logistic regression, naive bayes, decision trees, random forests and many more. In addition, I will be comparing their performance and time complexity. I am using Macintosh with 2.9 GHz Quad-Core Intel Core i7 CPU and 16 GB RAM.

Random Forest is a supervised learning algorithm that can be used for both classification and regression. It is an easy and flexible algorithm. This algorithm is based on randomly selected sets of decision trees. It creates decision trees based on randomly selected data samples and gets a prediction from each tree and selects the best solution by means of voting. It uses the majority wins theory. In order to use this algorithm, we need to import RandomForestClassifier from the scikit-learn library. Scikit-learn library provides free supervised and unsupervised machine learning algorithms for python. n_estimater is an important parameter where you need to specify the number of trees in the forest. This is an optional parameter and default is 100. The confusion matrix is used to check the quality and performance of a model and below is the confusion matrix chart from this algorithm.



This confusion matrix of the Random Forest algorithm clearly shows that 9459 items are correctly identified as a benign class. Similarly, 6050 are correctly classified as dos, 1485 as a probe, 110 as r2l and 3 as u2r.

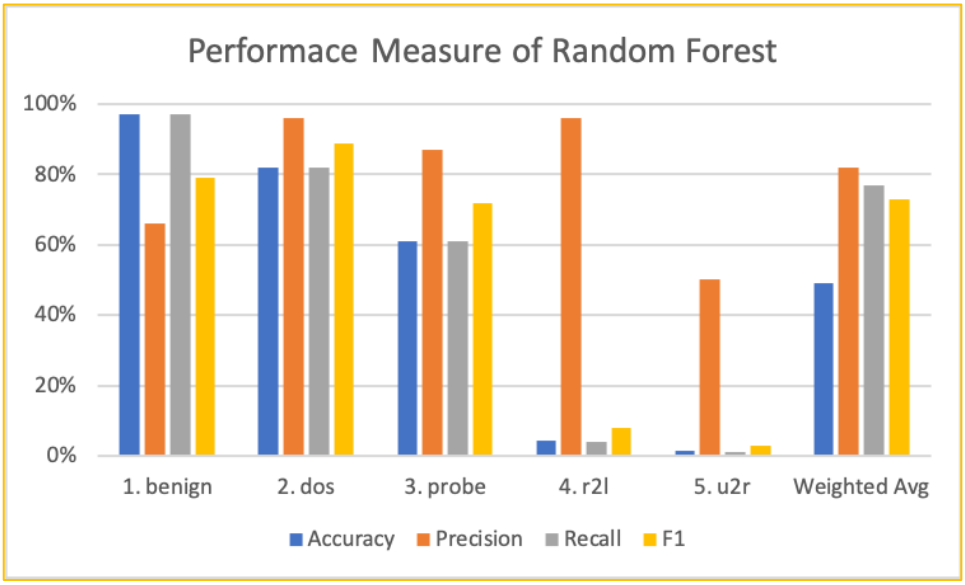
The performance measure of Random Forest Classifier algorithm is as follow:

Algorithm	Accuracy	Precision	Recall	F1
1. benign	97%	66%	97%	79%
2. dos	82%	96%	82%	89%

3. probe	61%	87%	61%	72%
4. r2l	4.3%	96%	4%	8%
5. u2r	1.5%	50%	1%	3%
Weighted Avg	49.16%	82%	77%	73%

From the above table, we can clearly see that the benign class has a higher accuracy rate and u2r has a lower accuracy rate. Similarly, dos has the highest precision rate and u2r has the lowest precision rate. Benign has a higher recall rate and u2r has the lowest recall rate. Dos has the highest f1 rate and u2r has the lowest f1 rate. Overall, average weighted accuracy is around 49%, precision is 82%, recall is 77% and F1 is 73%.

Let's see the above result in a chart:



Hyperparameter tuning of Random Forest can be done using GridSearchCV package where we initialize a dictionary of hypermeter values. The max_depth and min_samples_leaf values are similar to those of the decision tree. However, n_estimators is a new parameter, covering the total number of trees that you want your random forest algorithm to consider while making the final prediction. We then build and fit the gridsearch object to the training data and extract the optimal parameters. The best model is then extracted using these optimal hyperparameters. The n_jobs parameter tells Scikit-Learn the number of CPU cores to use for training and predictions where –1 tells Scikit-Learn to use all available cores.

k-nearest neighbors Classification (KNN) is a simple algorithm that classifies values based on similarity measures such as distance. It is used for both classification and regression predictive problems. However, it is mostly used in classification problems. This is simple and popular because it is easy to interpret the output. Similarly, calculation time is faster. It works by finding the distance between a point and data with the selected specific number closet to the point then votes for the most frequent label. Parameter n_neighbors is very important as it changes the accuracy rate varies. n_neighbors are a number of neighbors to use by default for neighbor's queries. There is no specific method to choose the best value for it. The

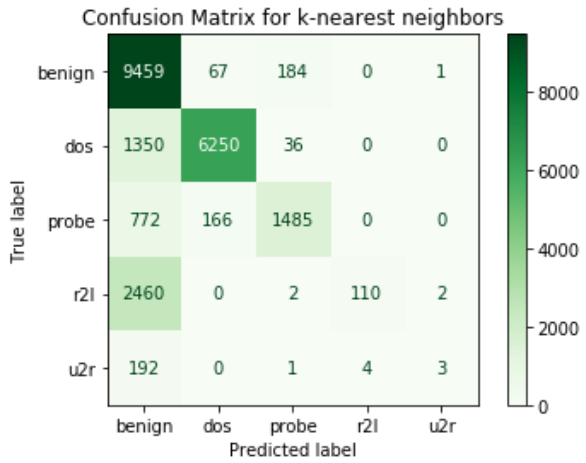
default value of it is 5. One of the ways to find out better value for n_neighbors is to iterate n_neighbors value with some range (1 to 40) and check the accuracy. Once you know the accuracy of different values of n_neighbors, you can choose the higher accuracy value.

Now let me find out the better value for the n_neighbors parameter by running 1 to 20 and calculate the accuracy rate for each value. Here is the result when running value from 1 to 20.

```
accuracy for k value  1 :  0.7579400283889283
accuracy for k value  2 :  0.7505766501064585
accuracy for k value  3 :  0.7551898509581263
accuracy for k value  4 :  0.7511089425124201
accuracy for k value  5 :  0.7619322214336409
accuracy for k value  6 :  0.7601579134137686
accuracy for k value  7 :  0.7631742370475515
accuracy for k value  8 :  0.7519073811213627
accuracy for k value  9 :  0.7532824698367636
accuracy for k value  10 :  0.7512420156139106
accuracy for k value  11 :  0.752971965933286
accuracy for k value  12 :  0.752750177430802
accuracy for k value  13 :  0.7558552164655784
accuracy for k value  14 :  0.7553229240596168
accuracy for k value  15 :  0.755722143364088
accuracy for k value  16 :  0.7556777856635912
accuracy for k value  17 :  0.7563431511710433
accuracy for k value  18 :  0.7560770049680624
accuracy for k value  19 :  0.7566980127750177
accuracy for k value  20 :  0.7564762242725337
```

k (n_neighbors) value with 7 has a slightly higher accuracy rate than others so we will be using this value to train the model and see the difference.

I have used an arbitrary value of 7 for n_neighbors and here is my confusion matrix.



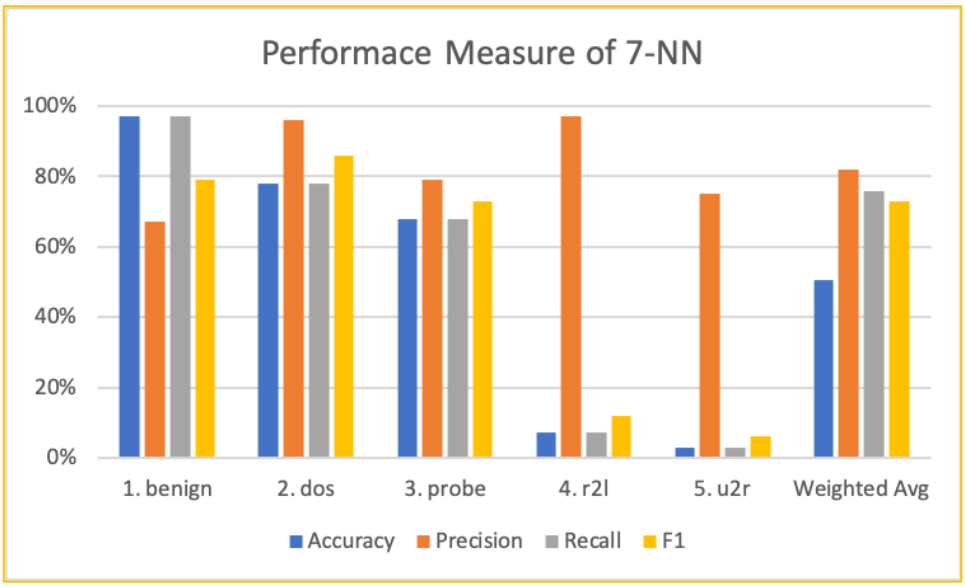
This confusion matrix of the k-nearest neighbors algorithm clearly shows that 9459 items are correctly identified as a benign class. Similarly, 6250 are correctly classified as dos, 1485 as a probe, 110 as r2l and 3 as u2r.

The performance measure of k-nearest neighbors Classifier algorithm is as follow:

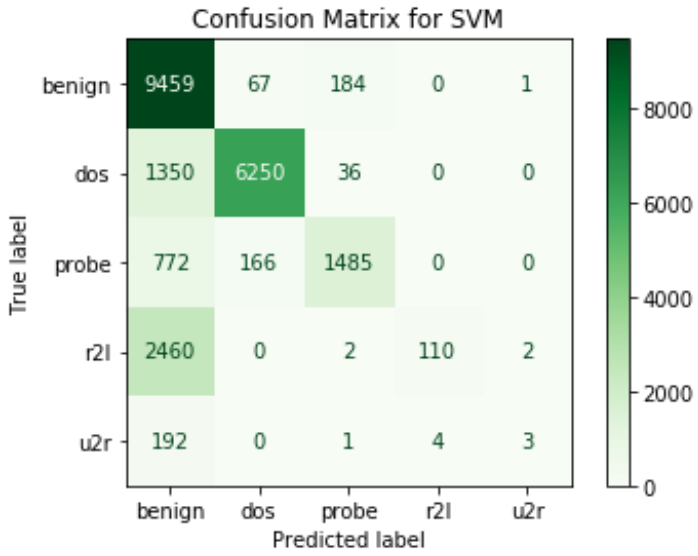
Algorithm	Accuracy	Precision	Recall	F1
1. benign	97%	67%	97%	79%
2. dos	78%	96%	78%	86%
3. probe	68%	79%	68%	73%
4. r2l	7%	97%	7%	12%
5. u2r	3%	75%	3%	6%
Weighted Avg	50.6%	82%	76%	73%

From the above table, we can clearly see that the benign class has a higher accuracy rate and u2r has a lower accuracy rate. Similarly, r2l has the highest precision rate and benign has the lowest precision rate. Benign has a higher recall rate and u2r has lowest recall rate. Dos has the highest f1 rate and u2r has the lowest f1 rate. Overall, accuracy has around 51% accuracy, precision has 82%, recall has 76% and F1 has 73%.

Let's see the above result in a chart:



A **Support vector machine (SVM)** is a supervised machine learning model that uses classification algorithms for two-group classification problems. It is a fast and dependable classification algorithm that performs very well with a limited amount of data. The support vector machine takes a pair of (x,y) coordinates and outputs the hyperplane that best separates the tags which is the decision boundary. The decision boundary is anything that falls to one side of it we will classify as class A and anything that falls to other as class B. It uses kernel as a parameter where it specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. The default kernel parameter value is rbf. I have used a 'poly' for kernel, degree as 1, C=3 and here is my confusion matrix.



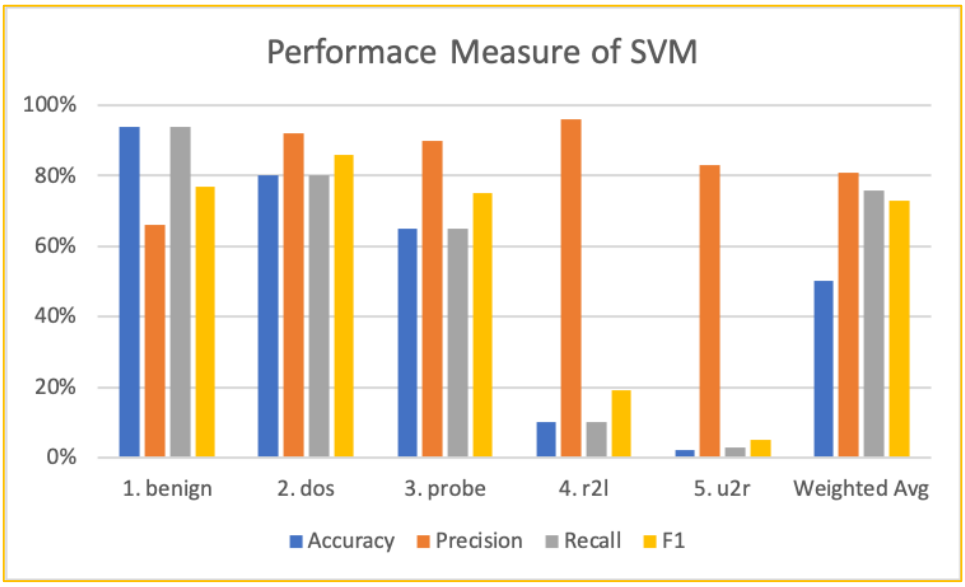
This confusion matrix of the SVM algorithm clearly shows that 9459 items are correctly identified as a benign class. Similarly, 6250 are correctly classified as dos, 1485 as a probe, 110 as r2l and 3 as u2r.

The performance measure of SVM Classifier algorithm is as follow:

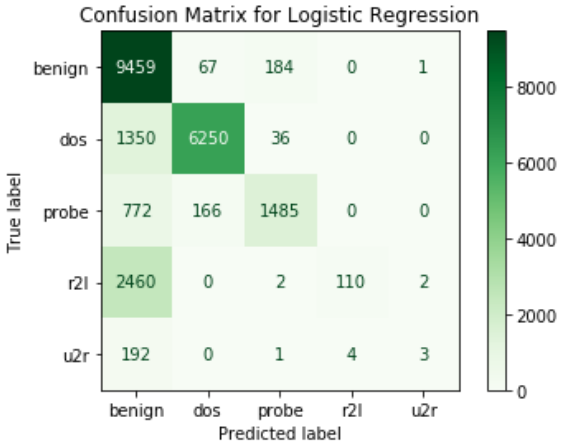
Algorithm	Accuracy	Precision	Recall	F1
1. benign	94%	66%	94%	77%
2. dos	80%	92%	80%	86%
3. probe	65%	90%	65%	75%
4. r2l	10%	96%	10%	19%
5. u2r	2%	83%	3%	5%
Weighted Avg	50.2%	81%	76%	73%

From the above table, we can clearly see that the benign class has a higher accuracy rate and u2r has a lower accuracy rate. Similarly, r2l has the highest precision rate and benign has the lowest precision rate. Benign has a higher recall rate and u2r has the lowest recall rate. Dos has the highest f1 rate and u2r has the lowest f1 rate. Overall, accuracy has weighted average of 50.2%, precision has 81%, recall has 76% and f1 has 73%.

Let's see the above result in a chart:



Logistic Regression is used for classification. This algorithm determines the probability of observation to be part of a certain class or not. The probability is expressed with a value between 0 and 1 in which 1 means the observation is very likely to be part of that category and 0 means the observation is not the part of that category. When it comes to classification, we are determining the probability of observation to be part of a certain class or not. Therefore, we wish to express the probability with a value between 0 and 1. The sigmoid function is used to generate these values between 0 and 1. Random_state is one of the parameters that this algorithm takes which is a random number generator. The default random state is none. I have used random_state to be 0 and here is my confusion matrix.



This confusion matrix of the Logistic Regression algorithm clearly shows that 9459 items are correctly identified as a benign class. Similarly, 6250 are correctly classified as dos, 1485 as a probe, 110 as r2l and 3 as u2r.

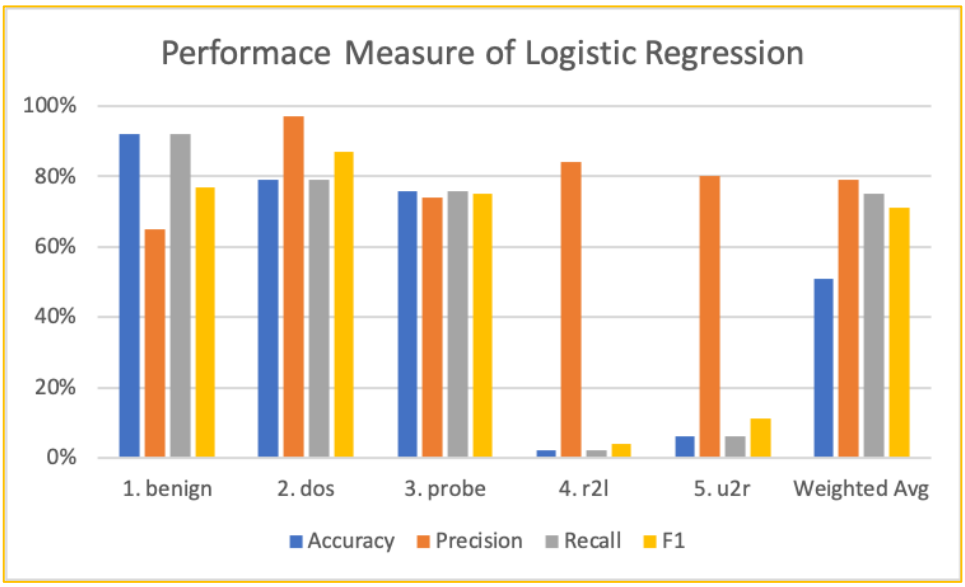
The performance measure of Logistic Regression Classifier algorithm is as follow:

Algorithm	Accuracy	Precision	Recall	F1
1. benign	92%	65%	92%	77%
2. dos	79%	97%	79%	87%
3. probe	76%	74%	76%	75%
4. r2l	2%	84%	2%	4%
5. u2r	6%	80%	6%	11%

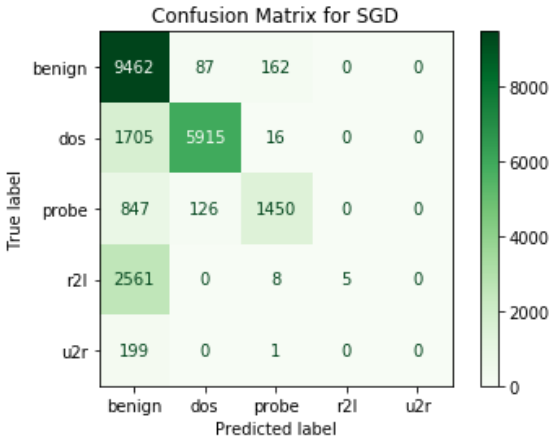
Weighted Avg	51%	79%	75%	71%

From the above table, we can clearly see that the benign class has a higher accuracy rate and r2l has a lower accuracy rate. Similarly, dos have the highest precision rate and benign has the lowest precision rate. Benign has a higher recall rate and r2l has the lowest recall rate. Dos has the highest f1 rate and r2l has the lowest f1 rate.

Let's see the above result in a chart:



Stochastic Gradient Descent (SGD) is a simple and efficient approach to discriminative learning of linear classifiers under convex loss function such as support vector machine and Logistic Regression. Efficiency and ease of implementation are the main benefits of SDG. SDG classifier is part of a linear model. Loss, penalty, max_iter are some of the parameters for this algorithm. Loss takes one of the values from "hinge", "modified_huber" and "log" where the hinge is linear support vector machine, modified_huber is smoothed hinge loss and a log is logistic regression. Similarly, penalty supports three different values which are l2, l1, and elasticnet. The default setting is penalty="l2". max_iter is the number of maximum iterations. In this algorithm, I have use loss='hinge', penalty='l2' and max_iter=5 and here is the confusion matrix.



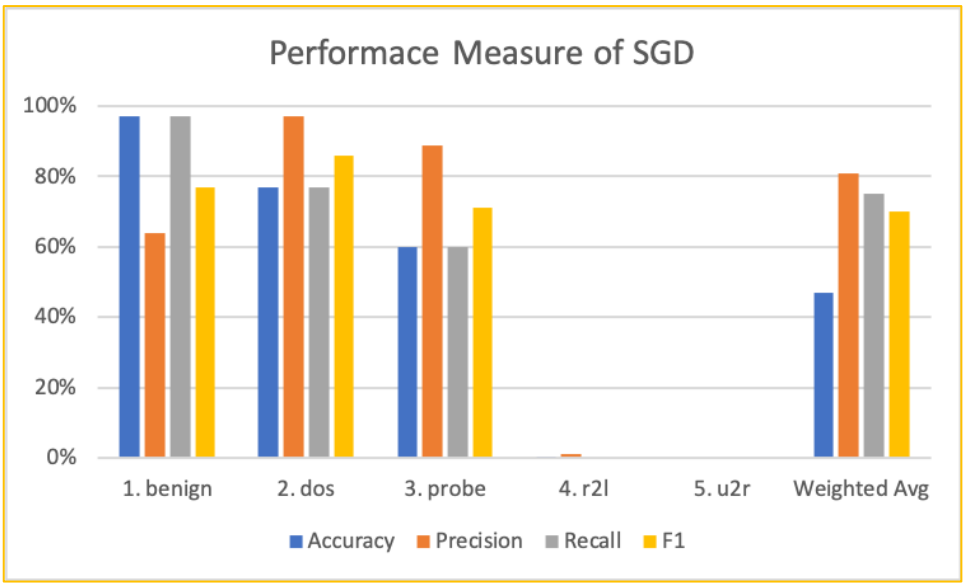
This confusion matrix of the SGD algorithm clearly shows that 9462 items are correctly identified as a benign class. Similarly, 5915 are correctly classified as dos, 1450 as a probe, 5 as r2l and did not identify any for u2r.

The performance measure of SGD Classifier algorithm is as follow:

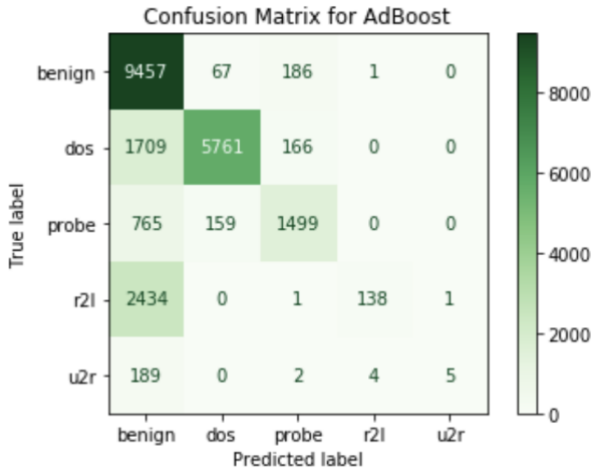
Algorithm	Accuracy	Precision	Recall	F1
1. benign	97%	64%	97%	77%
2. dos	77%	97%	77%	86%
3. probe	60%	89%	60%	71%
4. r2l	0.1%	1%	0%	0%
5. u2r	0%	0%	0%	0%
Weighted Avg	47%	81%	75%	70%

From the above table, we can clearly see that the benign class has a higher accuracy rate and u2r has a lower accuracy rate. Similarly, dos have the highest precision rate and u2r has the lowest precision rate. Benign has a higher recall rate and u2r and r2l have the lowest recall rate. Dos has the highest f1 rate and r2l and u2l have the lowest f1 rate. Overall, weighted average accuracy is 47%, precision is 81%, recall is 75% and f1 is 70%.

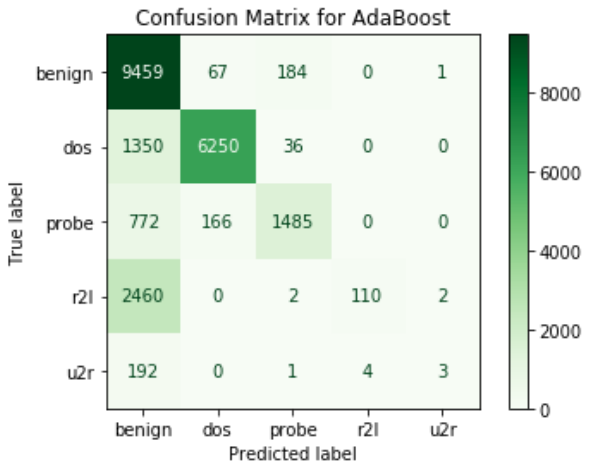
Let's see the above result in a chart:



AdaBoost classification is a popular boosting algorithm. Adaboost or adaptive boosting is one of the ensembles boosting classifiers. It combines multiple classifiers to increase the accuracy of classifiers. It is an iterative ensemble method where it builds a strong classifier by combining multiple poorly performing classifiers so that you will get higher accuracy. I have used this algorithm without any change in parameters and here is my confusion matrix.



This confusion matrix of the AdaBoost algorithm without any parameter being used and it clearly shows that 9457 items are correctly identified as a benign class. Similarly, 5761 are correctly classified as dos, 1499 as a probe, 138 as r2l and 5 as u2r.



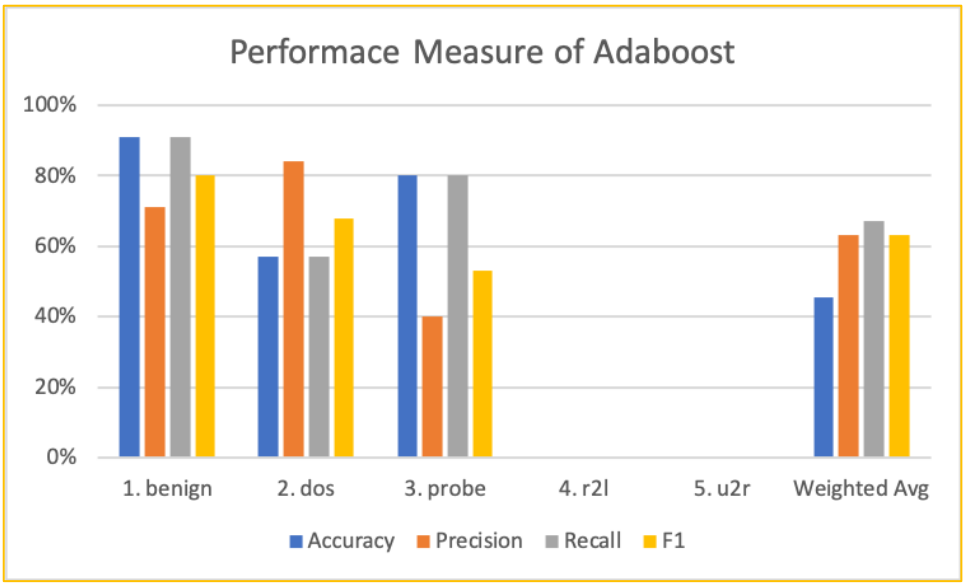
This confusion matrix of the AdaBoost algorithm is using n_estimator = 15 and learning_rate=1 and it clearly shows that 9459 items are correctly identified as a benign class. Similarly, 6250 are correctly classified as dos, 1485 as a probe, 110 as r2l and 3 as u2r.

The performance measure of AdaBoost Classifier algorithm is as follow:

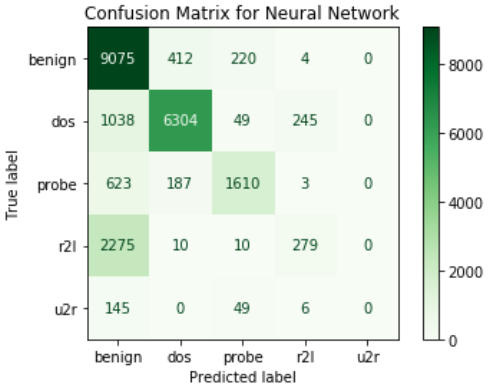
Algorithm	Accuracy	Precision	Recall	F1
1. benign	91%	71%	91%	80%
2. dos	57%	84%	57%	68%
3. probe	80%	40%	80%	53%
4. r2l	0%	0%	0%	0%
5. u2r	0%	0%	0%	0%
Weighted Avg	45.6%	63%	67%	63%

From the above table, we can clearly see that the benign class has a higher accuracy rate and u2r and r2l have a lower accuracy rate. Similarly, dos has the highest precision rate and u2r and r2l have the lowest precision rate. Benign has a higher recall rate and u2r and r2l have the lowest recall rate. benign has the highest f1 rate and r2l and u2l have the lowest f1 rate.

Let's see the above result in a chart:



Neural Network is set of algorithms that can be used to recognise underlying relationships in a set of data through a process that mimics the way the human brain operates. In other words, neural network is systems of neurons. It can adapt to changing input so the network generates the best possible result without needing to redesign the output criteria. Multi-layer Perceptrons (MLP) algorithm can be used to solve the classification problem where this algorithm trains using backpropagation. MLP classifier algorithm takes parameters such as hidden layers, activation, alpha and learning rate.



This confusion matrix of the neural network algorithm clearly shows that 9075 items are correctly identified as a benign class. Similarly, 6304 are correctly classified as dos, 1610 as a probe, 279 as r2l and did not identify any for u2r.

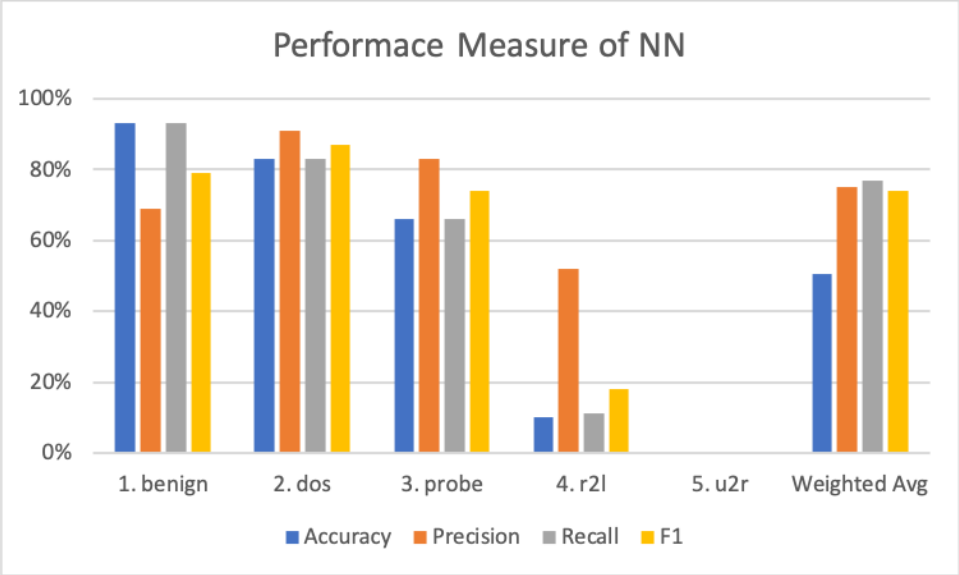
The performance measure of MLP Classifier algorithm is as follow:

Algorithm	Accuracy	Precision	Recall	F1
1. benign	93%	69%	93%	79%
2. dos	83%	91%	83%	87%
3. probe	66%	83%	66%	74%
4. r2l	10%	52%	11%	18%
5. u2r	0%	0%	0%	0%
Weighted Avg	50.4%	75%	77%	74%

From the above table, we can clearly see that the benign class has a higher accuracy rate and u2r has a lower accuracy rate. Similarly, dos has the highest

precision rate and u2r has the lowest precision rate. Benign has a higher recall rate and u2r has the lowest recall rate. benign has the highest f1 rate and u2l has the lowest f1 rate. Overall weighted average of accuracy is just above 50%, precision is 75%, recall is 77% and f1 is 74%.

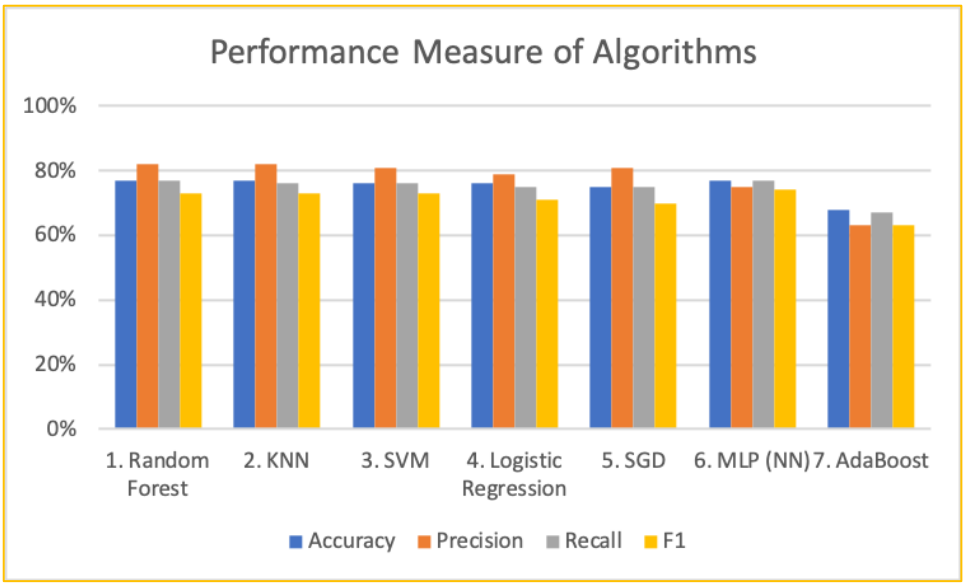
Let's see the above result in a chart:



Summary of algorithms

Algorithm	Accuracy	Precision	Recall	F1	Time (s)
1. Random Forest	77%	82%	77%	73%	5.88
2. KNN	77%	82%	76%	73%	34.23
3. SVM	76%	81%	76%	73%	207.3
4. Logistic Regression	76%	79%	75%	71%	5.97
5. SGD	75%	81%	75%	70%	16.35
6. MLP (NN)	77%	75%	77%	74%	420.20
7. AdaBoost	68%	63%	67%	63%	6.61

Let's visualise the result table into bar graph.



Above table and graph clearly shows that most of the algorithm performed very similar to each other. SVM, Support Vector Machine classification performed well in terms of accuracy. Similarly, AdaBoost classification performed least among the algorithms. Again, SVM performed well in terms of precision and AdaBoost performed worst. KNN performed will in terms of recall and AdaBoost performed worst. KNN and Logistic Regression performed best with higher F1 rate and AdaBoost performed worst.

In order to obtain great accuracy, outside of the scikit learn can be explored such as Xgboost or Catboost which are based on booting algorithms. I also believe using ensemble learning algorithms can boost the performance of the models. Voting classification can also be used to combine multiple better performing algorithms and use voting method which is majority wins method for classification.