Doubtfire University

Security and Privacy Issues in Analytics

OnTrack Submission

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

Submitted By:
Vinit Karunakar Shetty
vkshetty
2022/01/17 19:33

Tutor:
Ann George

Outcome	**Weight**
ULO1	♦♦♦♦♦
ULO2	♦◊◊◊◊
ULO3	♦♦♦♦♦
ULO4	♦♦♦♦♦
ULO5	♦♦♦♦♦

End to End deployment of the task was done.

January 17, 2022

SIT719 Security and Privacy Issues in Analytics

Assessment Task 5.1

End-to-end project delivery on cyber-security data analytics

Student Name: Vinit Shetty

Student ID - 221426969

Table of Contents

Contents

Results Report

Background

For two ideal datasets for anomaly detection in the network, the project intends to construct multi-class machine learning-based classification models to detect diverse network traffic types. Identifying the data source, filtering anomalous data, analysing data and its qualities, Feature Engineering, Predictive Modelling, data visualisation, and eventually the distribution of findings are all processes that must be taken to achieve this[1]. Predictive modelling is a way of employing data modelling to anticipate future consequences. Given that our problem statement necessitates the categorization of many sorts of attack classes in the network, we will employ several benchmark classification models.

Algorithms

1. Random Forest.

The RF classifier is an ensemble approach that uses bootstrapping and aggregation to train many decision trees simultaneously. The RF classifier has strong generalisation and outperforms most of the other classification algorithms in terms of accuracy while avoiding overfitting concerns.

The bagging technique is utilised in Random forest, where random records from the data set are chosen and individual decision trees are constructed from bootstrap samples. Each decision tree is trained individually that generates results, with the outcome based on Majority Voting or Averaging after all of the models' findings have been combined.

Some parameters that can be tuned for optimizing the results[2]

- *Max_features.*

  This is the maximum amount of features that Random Forest may try in a single tree. Increasing max features increases the model's overall performance since each node now has a larger number of possibilities to examine, but it slows down the procedure.

- ❖ *n_estimators.*

  This represents the total number of trees to construct before using maximum voting or prediction averages. While having a larger number of trees improves efficiency, it also increases computational time. A high value of the estimator should be chosen since it makes predictions more stable.

- ❖ *Min_sample_leaf.*

  When the number of observations in one of the child nodes is less than the minimum leaf size, it is not possible to divide the node. The model is more likely to capture noise in train data when the leaf is smaller. Hence, for optimal performance, the minimum value should be greater than 50.

2. Decision Tree(CART).

A Decision Tree is a tree-structured classifier in which the internal nodes represent individual columns, branches indicate decision rules, and each leaf node represents the intended conclusion. The decision tree, on the other hand, is prone to overfitting and can be used in conjunction with other ensemble approaches. The CART algorithm is used in the decision trees and the decisions are made depending on conditions relating to any of the features. The criteria are represented by the internal nodes, while the decisions are represented by the leaf nodes.

We start at the root of the tree when predicting a class label for a record using decision trees. The values of the root attribute and the record's attribute are compared. We follow the branch that corresponds to that value and go to the next node based on the comparison. We keep comparing the attribute values of our record with those of other internal nodes in the tree until we find a leaf node with the expected class value.

Some parameters that can be tuned for optimizing the results -

- ❖ *Criterion.*

  The splitting of the decision tree nodes is done by measuring the impurity. The function allows us to choose the measure between the Gini Index or the information gain represented by "Entropy".

- *Splitter.*

  It defines the method for selecting the split at each node. "Best" is a supported strategy for selecting the best split that assesses all splits, while "random" is a supported strategy for selecting the best random split that employs a random uniform function.

- *Max_depth.*

  It reflects the maximum depth of the tree. If nothing is specified, nodes are stretched until all leaves are pure or contain fewer samples than the min samples split samples. If the number is higher the classifier will become prone to overfitting.

- *Max_features*

  When a split occurs, the decision tree algorithm examines many features and selects the one with the best metric and divides the tree into two branches. The maximum number of features can be mentioned to reduce the computational time to check all of the features every time and instead just check a subset of them.

3. Naive Bayes.

The Naive-Bayes classifier is based on the Bayes Theorem and employs a probabilistic approach to classification issues. The Naive-Bayes algorithm assumes that all characteristics are independent of one another and that each feature has the same weighting or value.

It works on the principle of conditional probability where the occurrence of the dependent feature Y can be calculated by using multiple independent features X[3].

Some parameters that can be tuned for optimizing the results -

- *Var_smoothing.*

  The Gaussian distribution gives more weightage to samples that are closer to the mean of the distribution. To make sure it follows a normal distribution curve, The variable var smoothing widens the curve and compensates for additional samples that are further distant from the distribution mean by arbitrarily adding a user-defined value to the variance of the distribution.

4. K-Nearest Neighbour.

KNN is a non-parametric and lazy learning algorithm which means that there is no assumption for underlying data distribution. The KNN classifier is based on the concept of instance-based learning. KNN calculates the distance between two points and divides them into classes depending on that distance. The KNN method maintains all of the relevant data and then uses similarity to categorise fresh data points. As a result, if the new data is surrounded by training data of Class A, it may be assumed that the new data is part of Class A.

The Euclidean distance between an unlabeled instance and every labelled instance is determined for unlabeled instance. To categorise the unlabeled instance the top labelled samples with the shortest path are chosen.

Some parameters that can be tuned for optimizing the results -

- *n_neighbors.*

  This defines the number of neighbours or classes to use while performing the algorithm. An optimum way of finding this is by using the 'elbow' method.

- *Leaf Size.*

  The size of the leaf is the size passed to the KDTree. This can have an impact on the computational speed and memory with which the tree is built and the best value is determined by the nature of the situation.

- *Metric.*

  This represents the distance metric used by the algorithm and depending on the value of 'p' the metric can be changed to euclidean, Minkowski or manhattan distance.

- *Weights.*

  This depicts the prediction weight function, which can be uniformly distributed i.e. 'uniform' or weighted by the inverse of their distance i.e. 'distance'.

5. Adaboost Classifier.

Adaboost (Adaptive Boosting) is an iterative ensemble algorithm. It combines many classifiers to improve classifier accuracy. Adaboost's core principle is to establish the weights of classifiers and train the data sample in each iteration so those correct predictions of uncommon observations are guaranteed. Decision trees are commonly employed as a base classifier for classification purposes[4].

The procedure begins by selecting a random training subset on which the AdaBoost machine learning model is iteratively trained. It gives incorrectly categorised observations a larger weight so that they have a higher chance of being classified in the next iteration. Finally, a vote is taken across all learning algorithms for classification.

Some parameters that can be tuned for optimizing the results[4]

- *base_estimator*.

  This defines the base classifier on which the boosted ensemble will be built. If left blank, the Decision Tree Classifier will be chosen as the default base estimator.

- *n_estimators*

  Before employing maximum voting or prediction averages, this is the total number of trees to build. While having more trees enhances efficiency, it also lengthens the calculation time. The estimator should be set to a high value since it makes predictions more stable.
- *learning_rate*

  At each boosting iteration, the learning rate indicates the weight given to each classifier. There is a trade-off between the learning rate and n estimators parameters and a balance is required for optimum performance.

6. Logistic Regression.

Logistic regression, which is used to estimate the likelihood of a categorical dependent variable, is based on the probability notion. Instead of using a linear function, it uses the sigmoid function, which produces a probability value between 0 and 1. If a sigmoid function's output is more than or equal to 0.5, the output is classed as 1; if it is less than 0.5, the output is classified as 0. If the graph is negative, the anticipated value for y is 0 and vice versa. It's usually employed as a binary classifier, but in multiclass cases, the training algorithm employs the 'one-vs-rest' method.

Some parameters that can be tuned for optimizing the results[5]

- *Penalty.*

  High-valued regression coefficients are penalised by regularisation to prevent overfitting. Regularization norms are grouped into three categories. The absolute value of the magnitude of coefficients is added by the L1 penalty. The square of the magnitude of the coefficients is added by the L2 penalty. L1 and L2 techniques are combined in elastic nets.

- *Max_iter.*

  For the solvers to converge, the maximum number of iterations is used. The value is set to 100 by default. However, a greater value can be utilised to attain optimal performance, but this would increase computing complexity.

- *C (Regularization Parameter).*

  C is the regularization parameter that controls the trade-off between achieving a low training error and a low test error that is to ensure the generalisability of the model.

7. Support Vector Machines.

Support vectors machines or SVM does not use a probabilistic model like other classifiers but it creates hyperplane lines that separate and classifies the data points into different feature spaces.

We developed SVM models using the RBF kernel function to predict attack classes. As it resembles the Gaussian distribution, RBF kernels are the most generic type of kernelization and one of the most extensively utilised kernels. The RBF kernel function computes the similarity, or how near two points X1 and X2 are to one another, given two points X1 and X2.

Some parameters that can be tuned for optimizing the results -

- *Kernel.*

  This field specifies the kernel to be used in the algorithm. The kernel can be linear, polynomial, radial(RBF) or sigmoid. For our algorithms, we have used only a single RBF kernel to reduce the computational complexity.

- *Gamma.*

  The Gamma parameter defines how far the single training sample is influenced and used only when using the RBF kernel. A Higher Gamma value corresponds to a higher value of the hyperplane curvature.

- *C.*

  The regularisation parameter C is used to control error and is set before the training model. To punish the cost of misclassification, a big value of C provides a low bias and high variance. When C is small, the penalty for misclassified points is small enough that a wide-margin decision on the boundary is chosen at the expense of a greater number of misclassifications.

8. LDA Classifier.

Linear Discriminant Analysis, or LDA, is a classification approach that reduces dimensionality. The Bayes' Theorem is used in LDA models to estimate probabilities and make predictions about the likelihood that a fresh input dataset belongs to a specific class. The output class is the one with the highest probability, after which LDA makes the prediction. They also utilise the likelihood of each class as well as the probability of the data in each class. This approach, however, is only viable if the dataset has a Normal distribution. Using the transform approach, the fitted model may be utilised to minimise the dimensionality of the input by projecting it to the most discriminative directions.

Some parameters that can be tuned for optimizing the results -

- *Solver.*

  To perform the classification, multiple solvers can be used like singular value decomposition(svd), Least square solution(lsqr) and Eigenvalue decomposition.

- *n_components.*

  This represents the number of components we require for dimensionality reduction.

- *Shrinkage.*

  Shrinkage is a type of regularisation that is used to enhance covariance matrix estimation. It can be left as 'auto' or changed to any float value between 0 and 1. Also, shrinkage works only with Eigen and lsqr solvers.

Evaluation Metrics

Different metrics should be created for analysing the algorithms, based on which judgments will be made. Performance parameters such as Accuracy, Precision, Recall, F1-Score, and False Alarm-FPR must be calculated for this reason. All the above measures can be calculated using the confusion matrix of the model. The confusion matrix, a N x N matrix used to evaluate the performance of a classification model, where N is the number of target classes[6], is passed to a function called "metrics," which returns all performance assessment metrics. Consider the following representation of the confusion matrix -

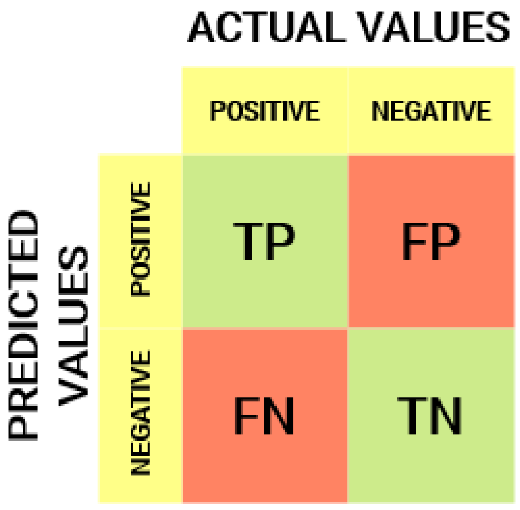


fig. 2 x 2 Confusion matrix with 4 values[6].

Using the obtained confusion matrix we could define the following metrics:

❖ Accuracy.
Accuracy is a good metric when the data is balanced. Hence, it is not an ideal scenario to use Accuracy for our problem.

Accuracy = (TP+TN) / (TP+FP+FN+TN)

❖ Precision.
Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actually positive.

Precision = (TP) / (TP + FP)

❖ Recall/ True Positive Rate (TP RATE).

TP RATE or Recall is the measure of our model correctly classifying the True Positive values.

Recall = (TP) / (TP + FN)

❖ F1-score.

F1-score is another ideal metric for Imbalanced classification. The range is between 0 to 1 and can also be defined as the harmonic mean of precision and recall.

F1-score = 2 * (Precision * Recall / Precision + Recall)

❖ False Positive Rate (FPR)

FPR is defined as the proportion of negative cases incorrectly identified as positive cases in the data i.e. the total number of negative cases incorrectly identified as positive cases divided by the total number of negative cases.

False Positive Rate (FPR) = (FP) / (FP + TN)

Experiment Protocol

For Dataset 1 the attack class was divided into five categories: benign, dos, r2l, u2r, and probe. We used six supervised algorithms to tackle the multiclass problem: Random Forest, Naive Bayes Classifier, K-Nearest Neighbour, Support Vector Machine, Adaboost Classifier, and Logistic Regression. On all numeric characteristics, the Standard Scaler() method was used to standardise them. Algorithm optimization is critical since it considerably improves performance. A very novel approach called Halving RandomSearchCV was used out of all the hyperparameter tuning strategies. They employ cross-validation to identify ideal hyperparameters, same as the conventional GridSearchCV and RandomizedSearchCV. Rather than searching the hyperparameter set candidates separately, their successive halving search approach begins by assessing all candidates with fewer resources and iteratively picking the top candidates.

The attack class in the second IoT dataset is solely separated into two labels. Because the test dataset is not available, the train-test split technique should be used to ensure that the algorithms are generalizable. With the split in the ratio of 70:30, Sklearn's 'train_test_split' function is employed. Random Forest, Naive Bayes, K-Nearest Neighbour, CART(Decision Tree), Logistic Regression, and LDA are six novel algorithms used for classification. RandomizedSearchCV is used to fine-tune the algorithms. In comparison to grid search, the randomised search model can be trained on the optimum parameters in a considerably shorter period. This results in significantly more efficient computational power being used.

Results

Dataset 1

1. Random Forest.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
Normal	76.74%	65.45%	97.46%	78.31%	38.93%
Dos	91.59%	96.42%	78.05%	86.26%	1.48.%
Probe	94.75%	83.77%	63.48%	72.22%	1.48%
R2L	88.83%	95.16%	2.29%	44.72%	0.00%
U2R	99.11%	40.00%	1.00%	19.51%	0.00%

- When Random Forest was used, the u2r attack class had the highest accuracy rate of 99.11 per cent   With 96.42 per cent precision, the dos class was found to be the most precise. The recall value for the normal label was found to be the highest at 97.46%, while the F1 score for dos was determined to be the top at 86.26%. The Normal class had the maximum false positive rate (38.93%), whereas the r2l and u2r attack classes had negligible false positivity rates.

2. Naive Bayes.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
Normal	84.78%	76.78%	92.70%	83.99%	21.22%
Dos	80.29%	88.2%	48.27%	62.39%	3.31%
Probe	85.17%	37.21%	55.22%	44.46%	11.22%
R2L	87.08%	41.67%	32.83%	36.72%	5.92%
U2R	95.09%	5.41%	27.50%	9.04 %	4.30%

- The u2r class has the highest accuracy rate for Naive Bayes, at 95.09 per cent. The dos class had the highest precision rate of 88.2 per cent, while the u2r class had the lowest precision rate of 5.41 per cent. The normal class had the greatest recall and f1 score, with 92.70 per cent and 83.99 per cent, respectively. The normal class had the greatest false positive rate of 21.22 per cent, while the dos and u2r attack classes had lower rates of 3.31 per cent and 4.30 per cent, respectively.

3. K-Nearest Neighbour.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
Normal	77.63%	66.41%	97.25%	78.92%	37.22%
Dos	91.36%	96.17%	77.59%	85.89%	1.58%
Probe	94.95%	82.52%	67.23%	74.09%	1.71%
R2L	89.30%	94.97%	6.60%	12.34%	0.05%
U2R	99.15%	90.00%	4.50%	8.57%	0.00%

- For the K-Nearest Neigggbour Classifier, the u2r attack class obtained the best accuracy of 99.15 per cent. At 96.17 per cent, the accuracy rate for the dos class was the highest. The normal class had the highest recall rate (97.25%), whereas the dos class had the best f1 score (85.89%). The normal class had the maximum false positive rate (37.22%), while the r2l and u2r assault classes had the lowest.

4. Support Vector Machines.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
Normal	77.74%	66.73%	96.38%	78.86%	36.36%
Dos	92.35%	96.39%	80.42%	87.68%	1.54%
Probe	94.54%	82.05%	63.02%	71.29%	1.66%
R2L	89.72%	97.42%	10.26%	18.56%	0.0%
U2R	99.17%	87.5%	7.0%	12.96%	0.0%

- The u2r attack class has the highest accuracy of 99.17 per cent while employing support Vector Machines. Against the present trend, the r2l attack class had the highest accuracy rate of 97.42 per cent. The normal class had the highest recall rate of 96.38 per cent, while the u2r class had the lowest recall rate of just 7%. The maximum f1 score for the dos class was 87.68 per cent. The normal class had a high false-positive rate of 36.36, whereas the r2l and u2r assault classes had none.

5. AdaBoost.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
Normal	77.21%	65.99%	97.19%	78.61%	37.91%
Dos	88.69%	92.19%	72.79%	81.35%	3.16%
Probe	91.71%	62.51%	57.04%	59.65%	4.12%
R2L	88.59%	100%	0.00%	0.00%	0.0%
U2R	99.11%	0.00%	0.00%	0.00%	0.0%

- The u2r attack class has the highest accuracy rate of 99.11 percent for the Adaboost Classifier. The r2l attack class had a precision rate of 100%, where as the u2r attack class had a precision rate of 0%. The recall rate was highes t in the normal class, at 97.19 percent, and lowest in the r2l and u2r classes, at 0%. Similarly, the falsepositive rate and f1 scores for u2r and r2l attacks were also 0%. With a false positivity rate of 37.91 percent, the normal class had the highest percentage of false positives.

6. Logistic Regression.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
Normal	74.79%	64.41%	92.69%	76.00%	38.75%
Dos	91.17%	96.35%	76.85%	85.5%	1.49%
Probe	93.88%	72.72%	68.96%	70.79%	3.12%
R2L	88.35%	33.94%	2.18 %	4.1 0%	0.0%
U2R	99.14%	64.71%	5.5%	10.14%	0.0%

- The u2r attack class achieved the highest accuracy of 99.14 per cent when Logistic Regression was applied. The dos class had the highest precision rate, at 96.35 per cent. The recall value for the regular class was 92.69 per cent, while the r2l attack had the lowest recall value of 2.18 per cent. The top f1 score for the dos class was 85.5 per cent. The false-positive rate for r2l and u2r attacks was zero, whereas the normal class had a rate of 38.75 per cent.

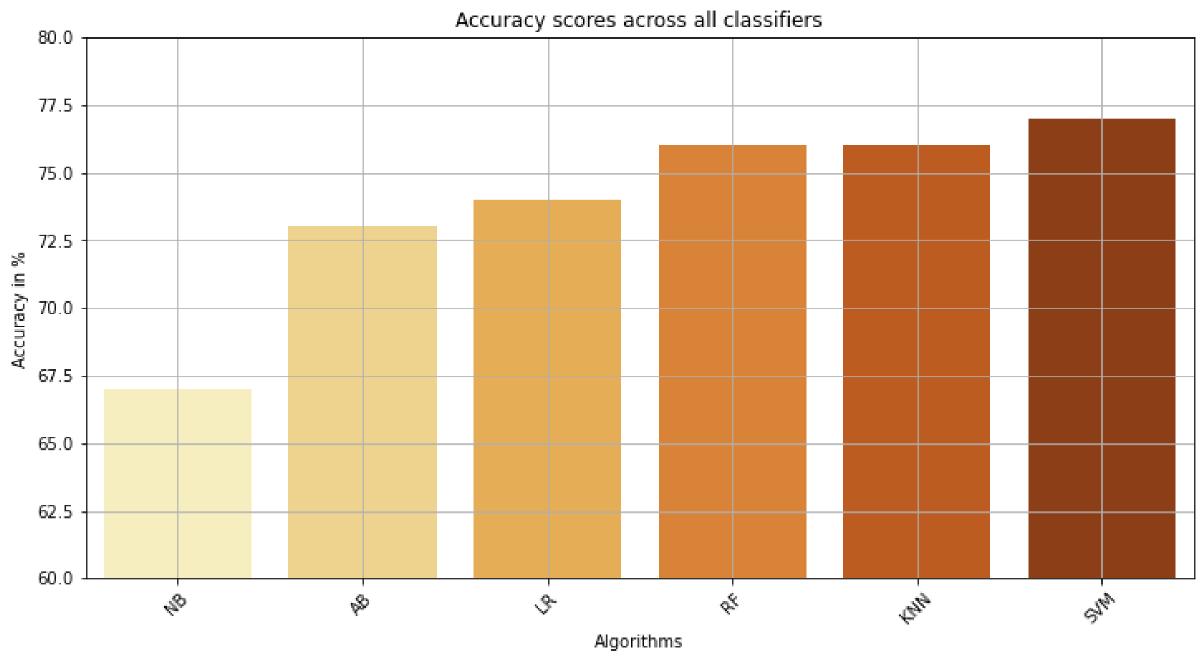
Overall weighted average for all the novel algorithms.

Algorithms	Accuracy	Precision	Recall	F1 Score	Train Time(s)	Test Time(s)
Random Forest	**76%**	**81%**	**76%**	**71%**	**85.006**	**0.50**
Naive Bayes	**66%**	**72%**	**66%**	**66%**	**63.19**	**0.77**
KNN	**76%**	**82%***	**76%**	**73%**	**5.79***	**166.35**
SVM	**77%***	**82%***	**77%***	**74%***	**212.91**	**74.35**
AdaBoost	**73%**	**78%**	**73%**	**68%**	**1490.37**	**3.53**
Logistic Regression	**74%**	**73%**	**74%**	**70%**	**217.56**	**0.094***

*** Represent the highest Score across all algorithms**

When we examine the weighted averages of all six methods, we can see that the SVM classifier surpasses all other classifiers in all four performance metrics. SVM, Random Forest, and KNN had good accuracy rates, but Naive Bayes Classifier had a poor accuracy rate. The precision rate for both SVM and KNN was determined to be 82 per cent, with Random Forest coming in second at 81 per cent. SVM, KNN, and Random Forest were the top three performers for Recall and F1 scores, respectively. In every metric, Naive Bayes fared the worst of all the algorithms. Because a different system is employed to accomplish the classification, as well as a different hyperparameter tuning procedure, the computing time differed significantly from the original IEEE publication. K-Nearest Neighbor had the quickest training time of 5.79 seconds, but the slowest testing time of 166.35 seconds. The shortest testing time was 0.094 seconds for Logistic Regression.

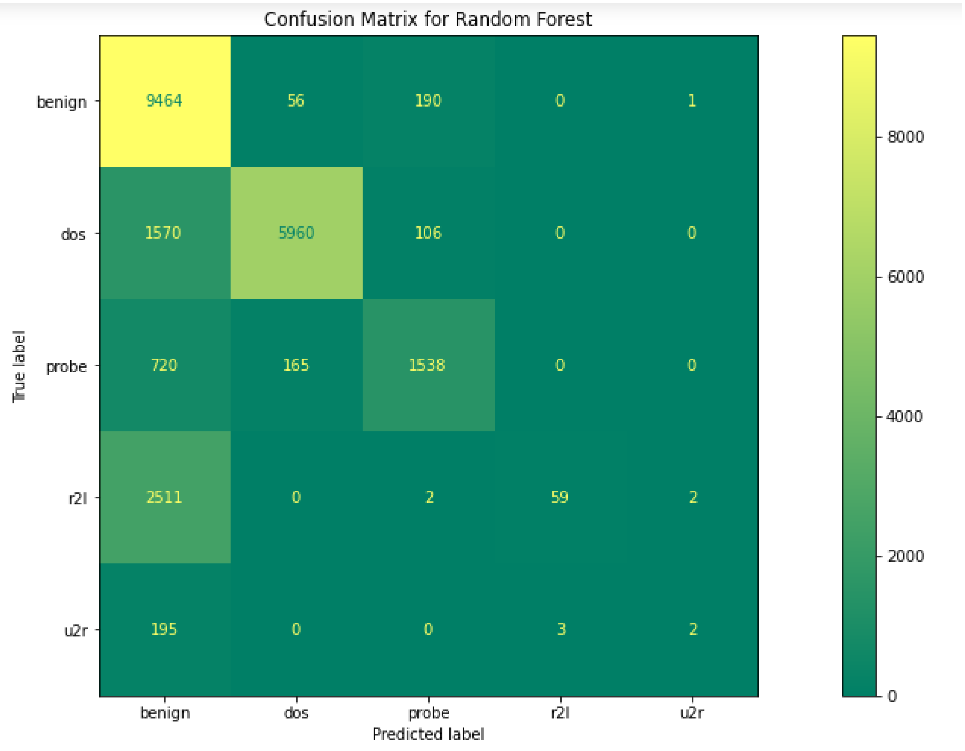
Graph 1



The comparison of the accuracy of the six classifiers used for dataset 1 is depicted in the bar chart above. The bars are ordered in ascending order, starting with Naive Bayes (less than 67.5%) followed by Adaboost (more than 72.5%). The accuracy rates of Random Forest, K-Nearest Neighbor, and SVM classifiers were all over 75%.

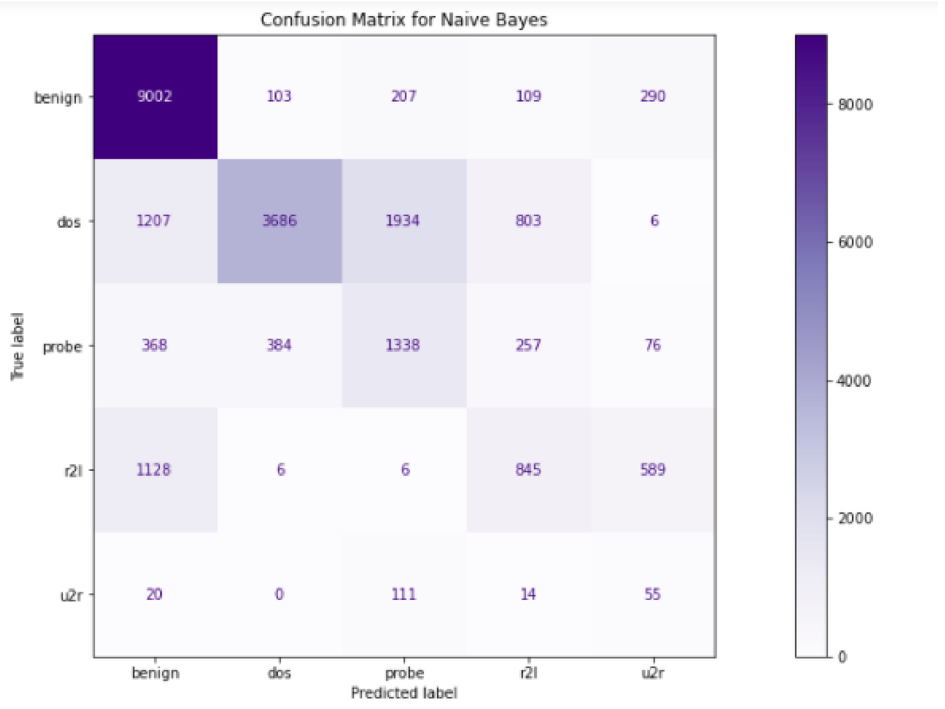
Confusion Matrix

- **Random Forest.**



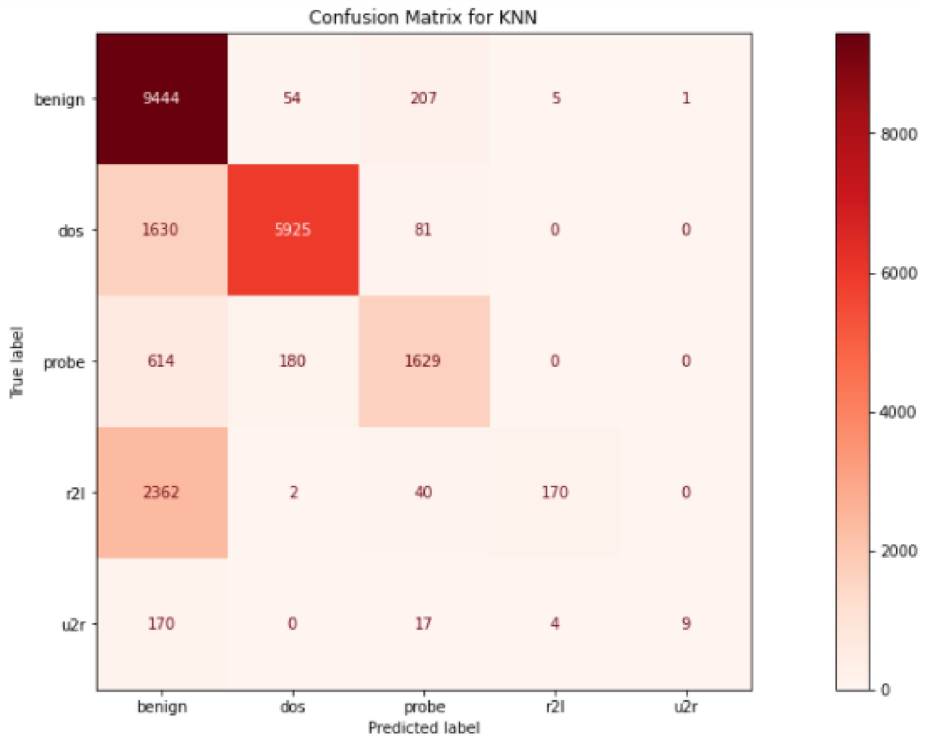
	benign	dos	probe	r2l	u2r
benign	9464	56	190	0	1
dos	1570	5960	106	0	0
probe	720	165	1538	0	0
r2l	2511	0	2	59	2
u2r	195	0	0	3	2

- **Naive Bayes.**



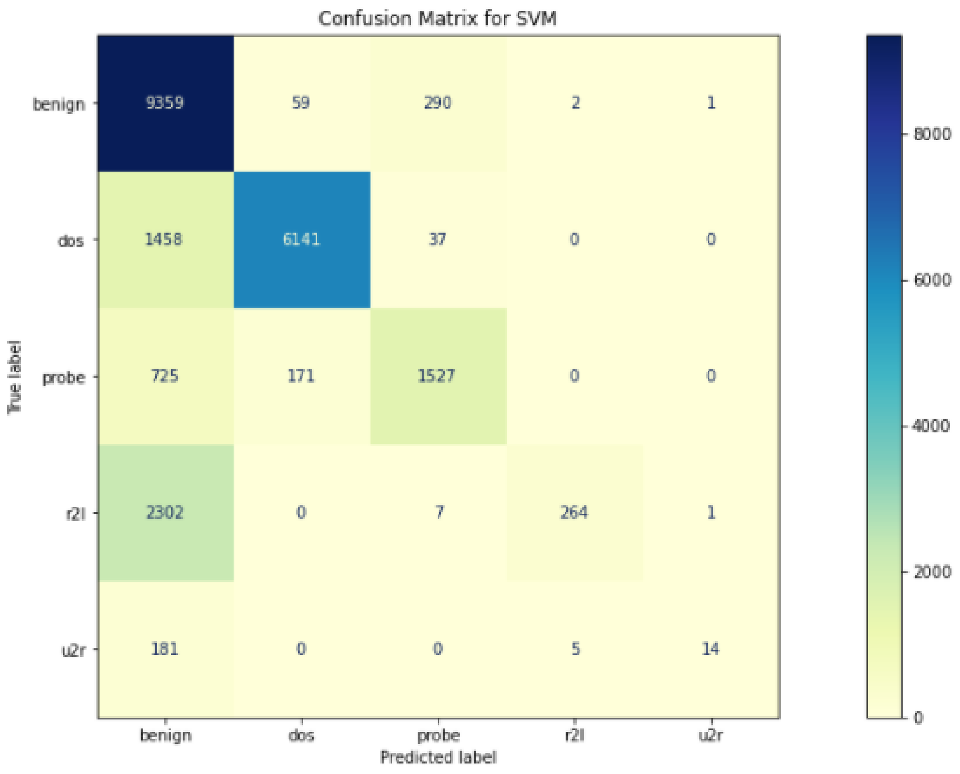
	benign	dos	probe	r2l	u2r
benign	9002	103	207	109	290
dos	1207	3686	1934	803	6
probe	368	384	1338	257	76
r2l	1128	6	6	845	589
u2r	20	0	111	14	55

- **K-Nearest Neighbour.**



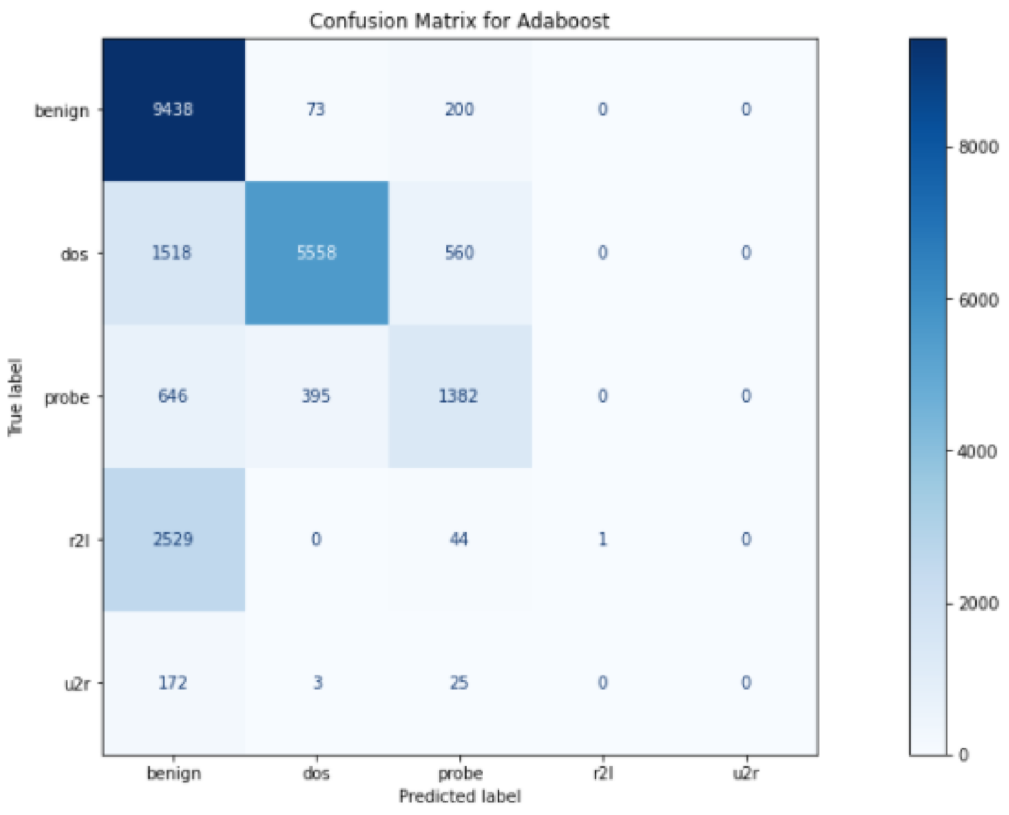
	benign	dos	probe	r2l	u2r
benign	9444	54	207	5	1
dos	1630	5925	81	0	0
probe	614	180	1629	0	0
r2l	2362	2	40	170	0
u2r	170	0	17	4	9

- **Support Vector Machines.**

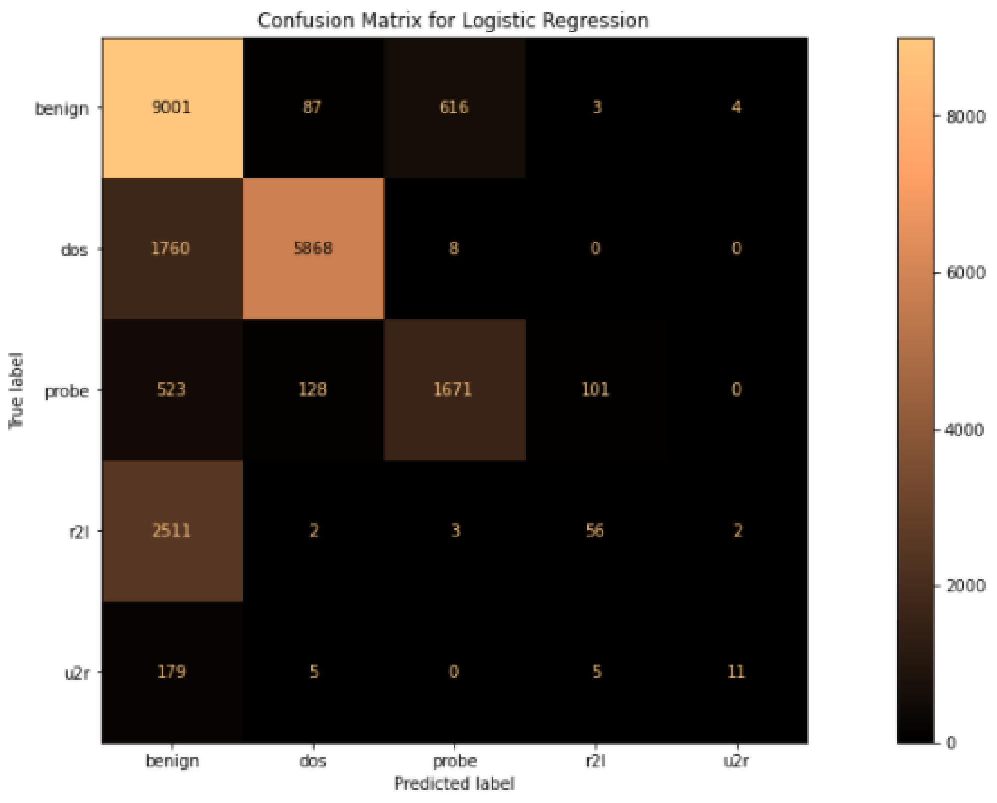


	benign	dos	probe	r2l	u2r
benign	9359	59	290	2	1
dos	1458	6141	37	0	0
probe	725	171	1527	0	0
r2l	2302	0	7	264	1
u2r	181	0	0	5	14

- **Adaboost Classifier.**



- **Logistic Regression.**



DATASET 2

1. Random Forest.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
0	87%	85.28%	95.14%	88.94%	25.77%
1	87%	90.68%	74.23%	81.63%	4.86%

- Both classes have an accuracy percentage of 87 per cent. Label 1 has a better precision rate of 90.68 per cent. Class 0 had a greater recall and F1 scores, with 95.14 per cent and 88.94 per cent, respectively. Class 0 has a false positive rate of 25.77 per cent, whereas class 1 has a low rate of 4.86 per cent.

2. Naive Bayes.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
0	69.99%	68.67%	93.53%	79.19%	66.95%
1	69.99%	76.51%	33.05%	46.16%	6.47%

- For both classes, Naive Bayes had an accuracy of 69.99 per cent. Label 1 has a better precision rate of 76.51 per cent. Recall for class 0 was substantially greater, at 93.53 per cent, than for class 1, which was just 33.05 per cent. Similarly, the F1 score for class 0 was high, at 79.19 per cent. The false-positive rate for class 0 was quite high, at 66.95 per cent.

3. KNN.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
0	83.32%	82.12%	92.92%	87.19%	31.74%
1	83.32%	86.00%	68.26%	76.11%	7.08%

- For both classes, KNN had a high accuracy of 83.32 per cent. Both classes had similar precision rates, about 86 per cent for class 1 and 82.12 per cent for class 0. Class 0 had a greater recall and f1 scores, with 92.92 per cent and 87.19 per cent, respectively. Label 0 has a high false-positive rate of 31.74 per cent.

4. CART.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
0	79.77%	76.84%	95.73%	85.25%	45.28%
1	79.77%	89.1%	54.72%	67.8%	4.27%

- For both attack types, the decision tree classifier has a high accuracy of 79.77 per cent. The precision rate for Class 1 is 89.1 per cent. Label 0 has strong recall and f1 ratings of 95.73 per cent and 85.25 per cent, respectively. The false-positive rate is 45.28 per cent lower than comparable algorithms.

5. Logistic Regression.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
0	68.98%	66.78%	97.91%	79.4%	76.42%
1	68.98%	87.8%	23.58%	37.18%	2.09%

- The accuracy rate for logistic regression is considerably lower, at 68.98 per cent. Class 1 has a high precision rate of 87.8 per cent. The values for Recall and F1 are higher, at 87.91 per cent and 79.4 per cent, respectively. Class 0 has a high false-positive rate of 76.42 per cent, whereas class 1 has a low rate of 2.09 per cent.

6. LDA.

Attack Class	Accuracy	Precision	Recall	F1 Score	FPR
0	68.31%	66.27%	97.99%	79.07%	78.27%
1	68.31%	87.32%	21.73%	34.8%	2.01%

- The accuracy rate for LDA is considerably lower, at 68.31 per cent. Class 1 has a high precision rate of 87.32 per cent. The values for Recall and F1 are higher, at 97.91 per cent and 79.07 per cent, respectively. Class 0 has a high false-positive rate of 78.27 per cent, whereas class 1 has a low rate of 2.01 per cent.

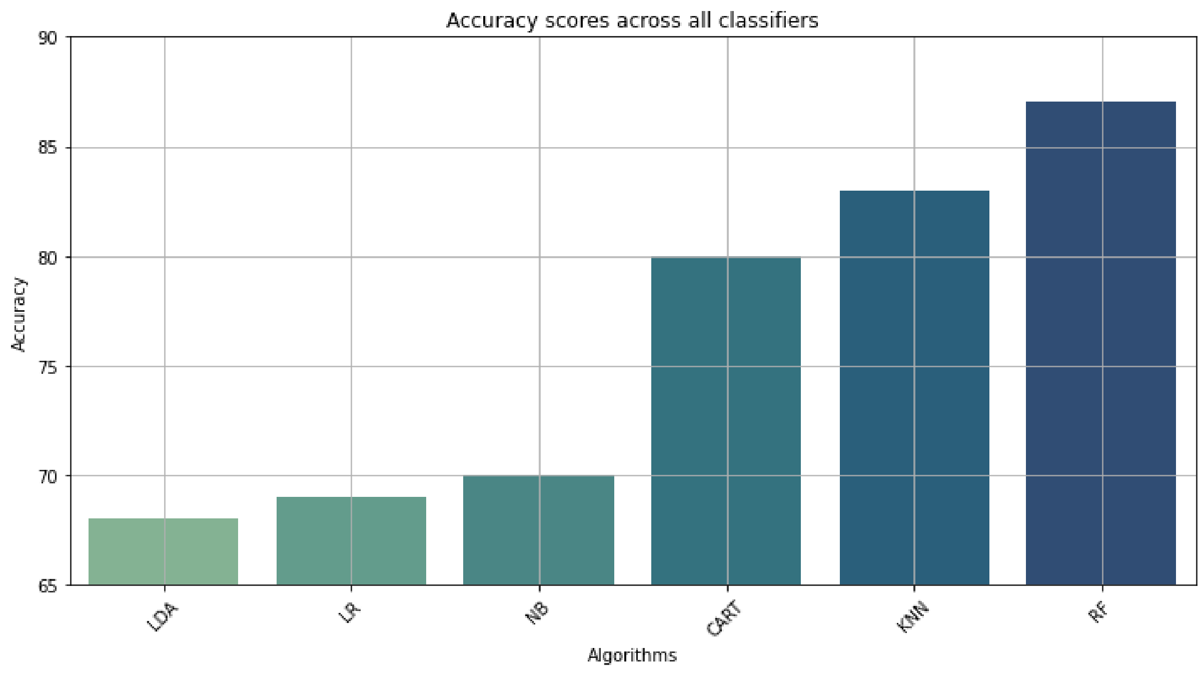
Overall weighted average for all the novel algorithms.

Algorithms	Accuracy	Precision	Recall	F1 Score	Train Time(s)	Test Time(s)
Random Forest	87%*	87%*	87%*	87%*	311.38	15.25
Naive Bayes	70%	72%	70%	66%	37.07	0.28
KNN	83%	84%	83%	83%	0.173*	2190.77
CART	80%	82%	80%	78%	92.84	0.208
Logistic Regression	69%	75%	69%	63%	939.29	0.0832
LDA	68%	74%	68%	62%	32.00	0.056*

* Represent the highest score across all algorithms

- We used six methods for dataset 2, and after assessing the classifiers, we found that the Random Forest Classifier produced the best results. On the dataset, the K-Nearest Neighbour and CART Algorithms also performed well. LDA has the lowest accuracy of all the classifiers, with only 68 per cent accuracy. With scores around 70%, Logistic Regression and Naive Bayes performed admirably. The computing time varies greatly from the original IEEE paper due to the use of the different working systems and a different hyperparameter tuning process. The fastest training time was 0.173 seconds, whereas the slowest testing time was 2910.77 seconds for K-Nearest Neighbor. The LDA classifier has the quickest testing time of 0.056 seconds. The training computational time for Random Forest and Logistic Regression was longer.

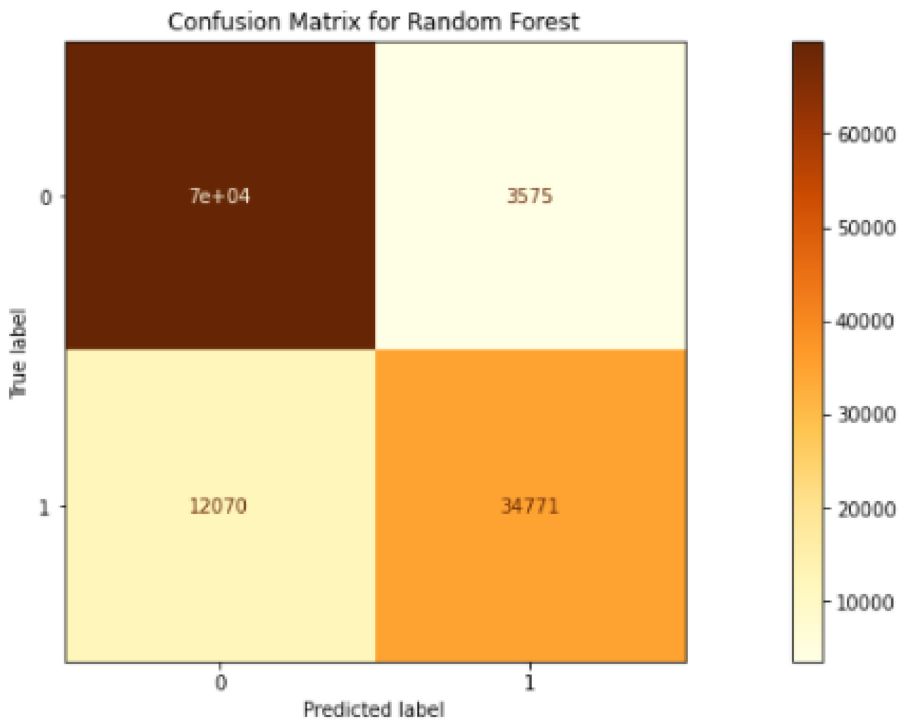
Graph 2



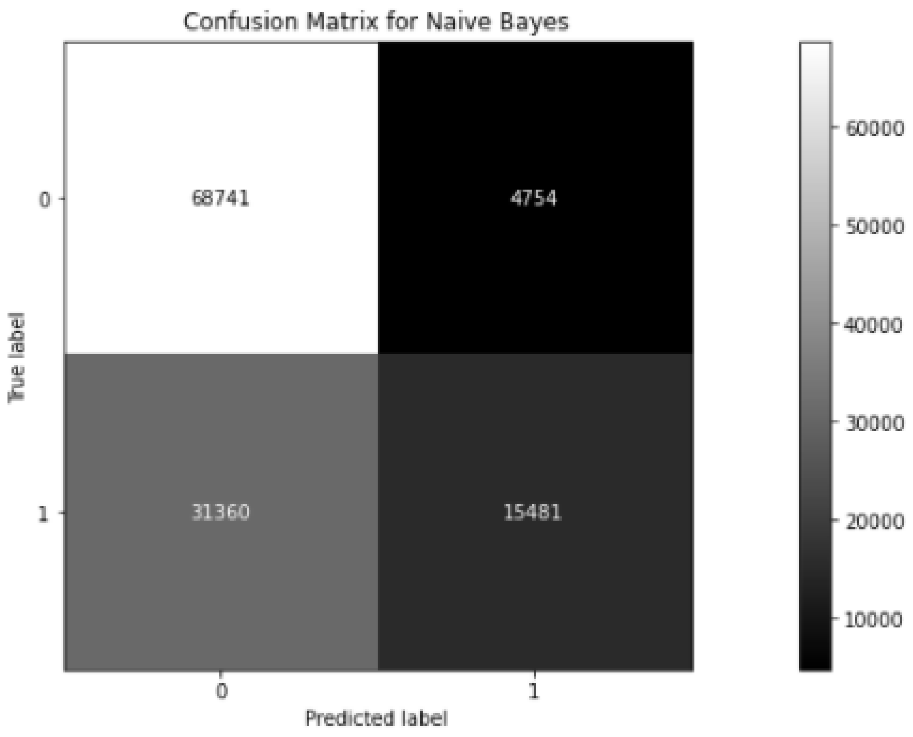
The bar chart above shows a comparison of the accuracy of the six classifiers used for dataset 2. Starting with LDA (less than 68 per cent), Logistic Regression (69 per cent), and Naive Bayes (70 per cent), the bars are arranged in ascending order. Random Forest, K-Nearest Neighbor, and CART classifiers have accuracy rates of 87 per cent, 83 per cent, and 80 per cent, respectively.

Confusion Matrix

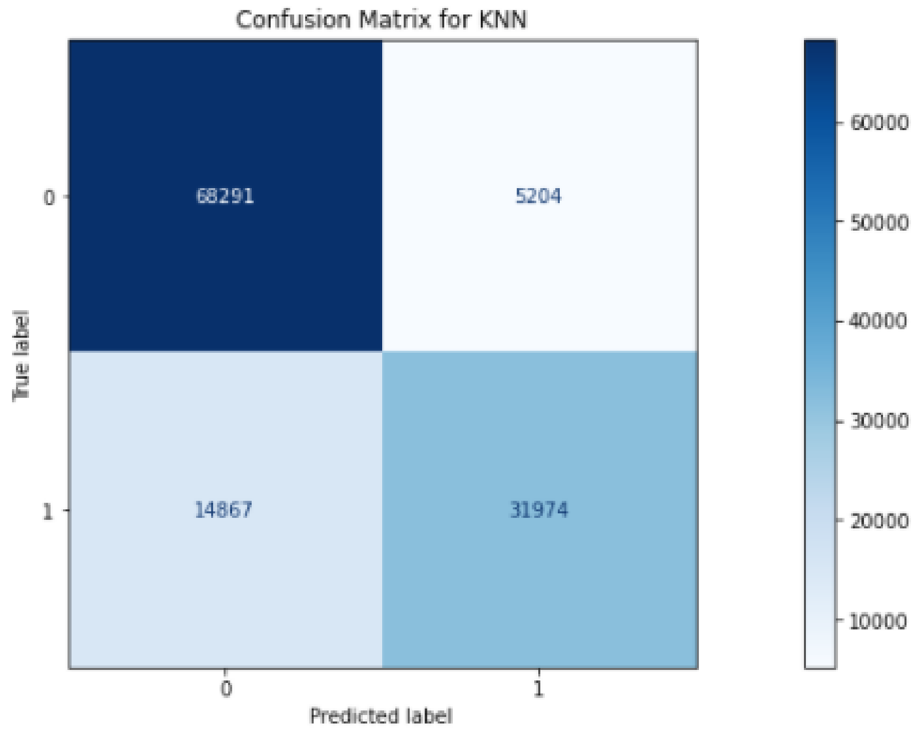
- **Random Forest.**



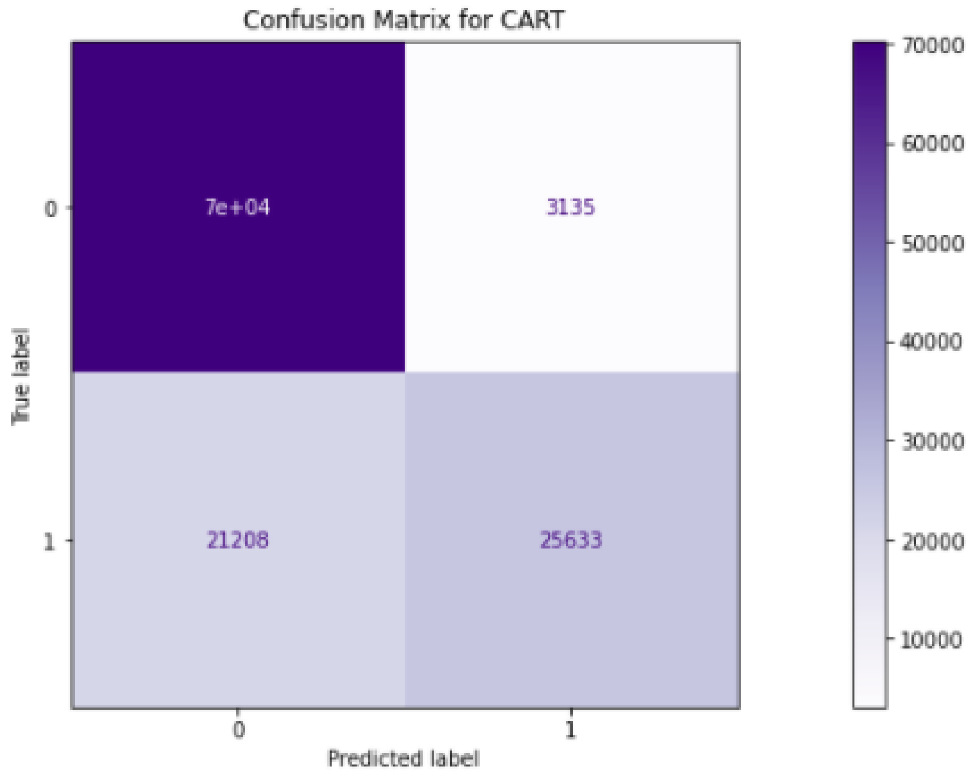
- **Naive Bayes.**



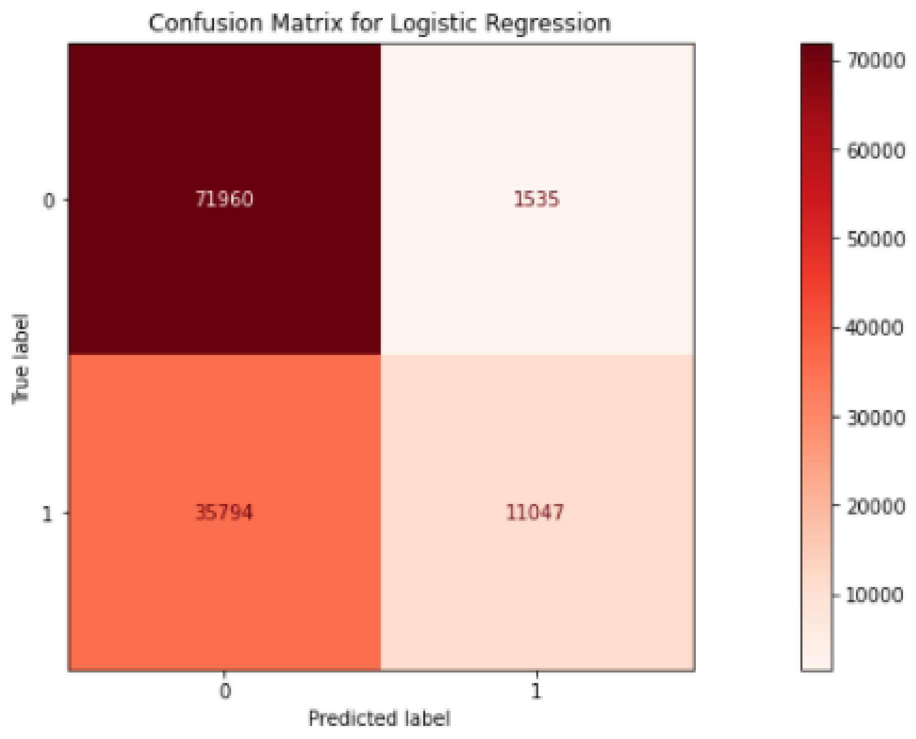
- **K-Nearest Neighbour.**



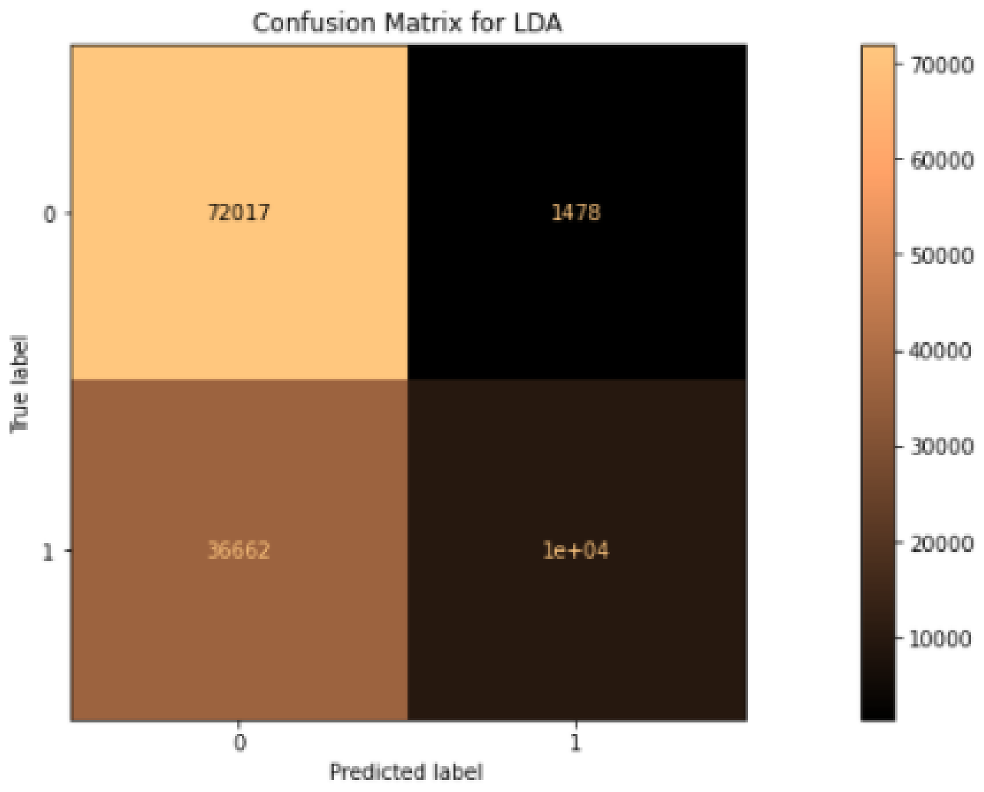
- **Decision Tree (CART)**



- **Logistic Regression.**



- **LDA.**



References

1. Rakesh Maddipati (2021, May 21), " Machine Learning Life-cycle Explained!," **[**Website**],** Available: https://www.analyticsvidhya.com/blog/2021/05/machine-learning-life-cycle-explained/.
2. Saurabh Gupta (2021, Jan 22), "Hyper Parameters of Random Forest Classifier," [Website], Available: https://www.geeksforgeeks.org/hyperparameters-of-random-forest-classifier/
3. Rohith Gandhi (2020, Mar 15), "Naive Bayes Classifier," Website], Available: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c
4. Srijani Chaudhury (2020, Aug 24), "Tuning of Adaboost Classifier with computational complexity," [Website], Available: https://medium.com/@chaudhurysrijani/tuning-of-adaboost-with-computational-complexity-8727d01a9d20
5. Soner Yildrim (2020, Jun 1), "Hyperparameter tuning for Support Vector Machines - C and Gamma Parameters," [Website], Available: https://towardsdatascience.com/hyperparameter-tuning-for-support-vector-machines-c-and-gamma-parameters-6a5097416167
6. Aniruddha Bhandari (2020, Apr 17), "Everything you Should Know about Confusion Matrix for Machine Learning," [Website], Available: https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/