

Attack Classification using Naïve Bayes Algorithm

Step 1:

Downloading the dataset and checking class distribution.

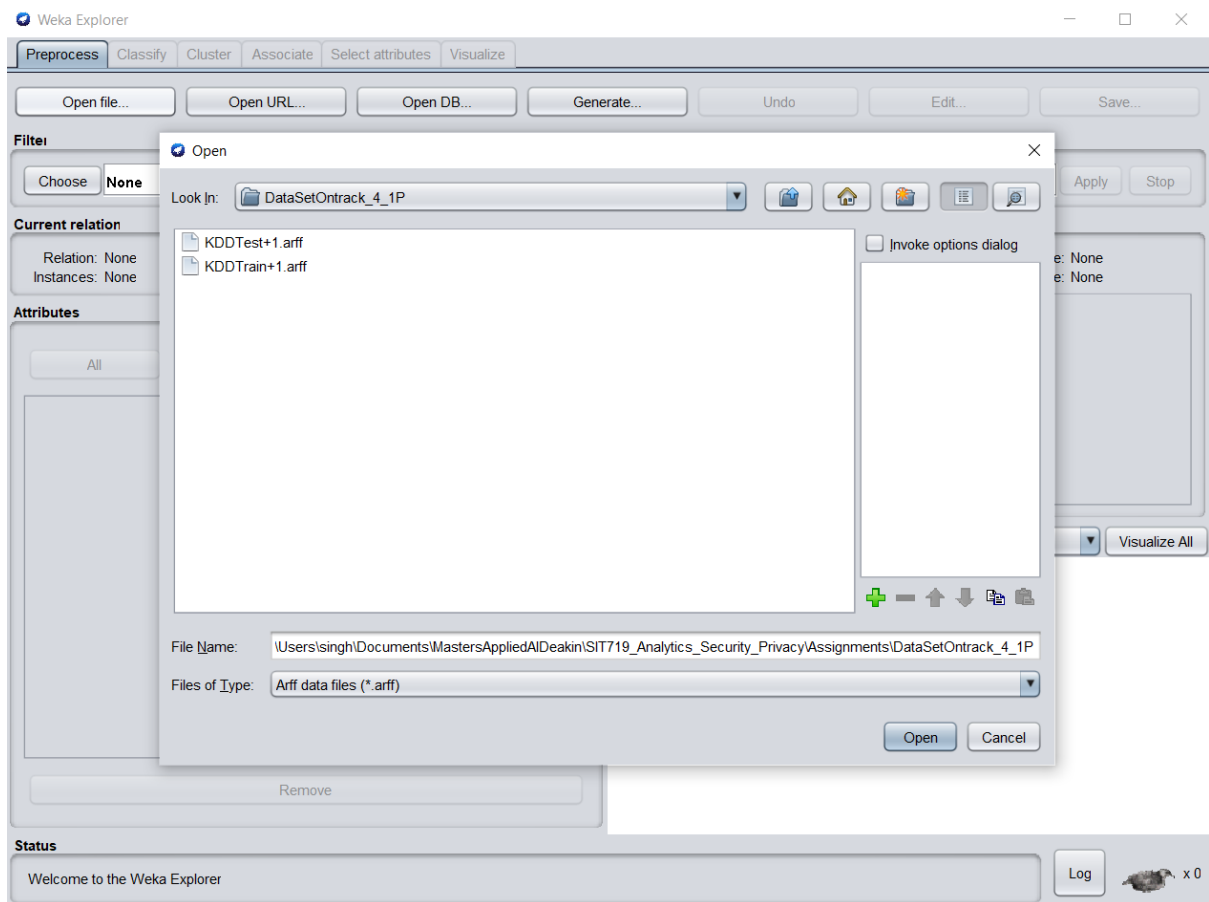


Fig: Downloaded data, train and test.

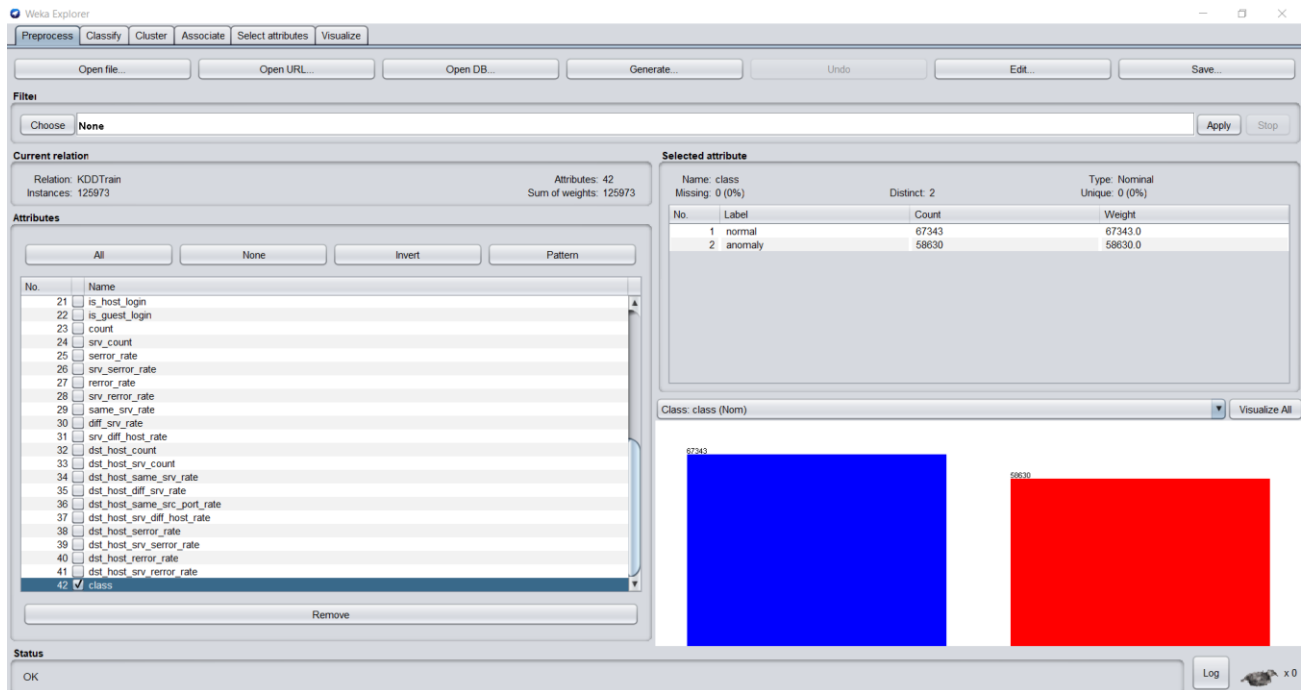


Fig: Data distribution

Step 2:

Applying Naïve Bayes Classifier

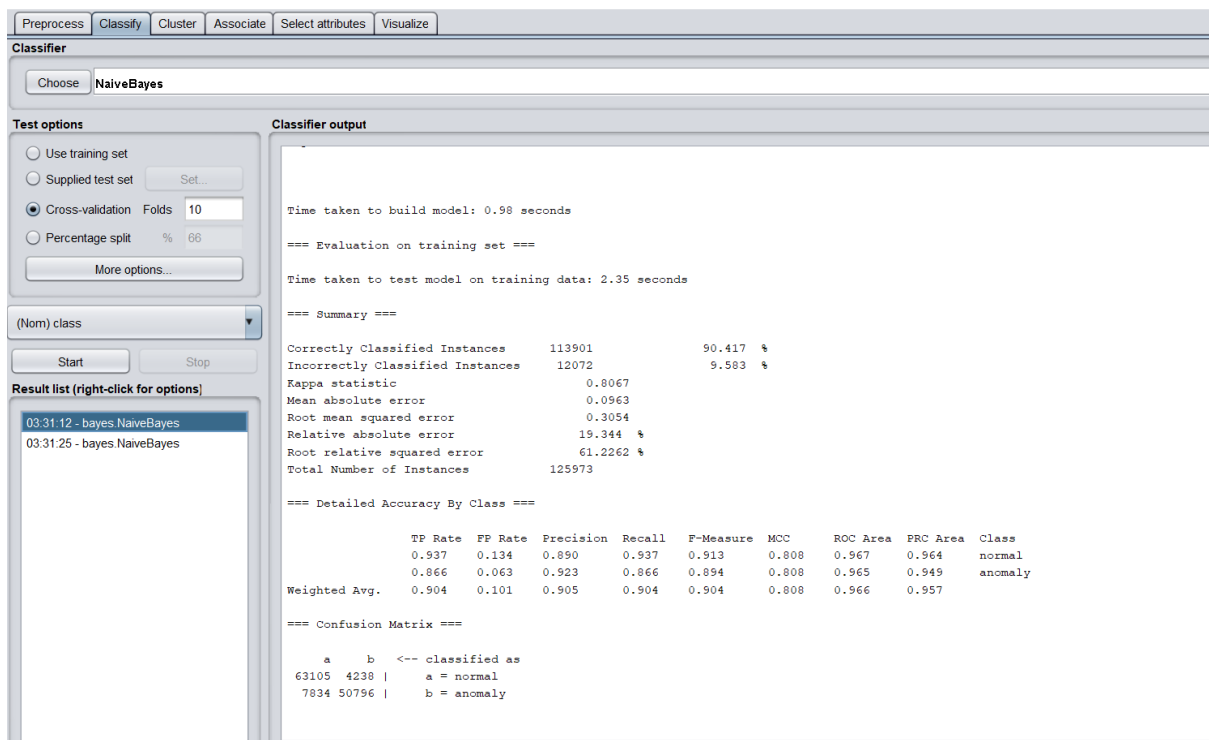


Fig: Classification summary after applying Naïve Bayes classifier.

Step 3:

Performing 10-fold cross validation

Test options

☐ Use training set

☐ Supplied test set

☒ Cross-validation Folds

☐ Percentage split %

(Norm) class

Result list (right-click for options)

- 03:31:12 - bayes.NaiveBayes
- 03:31:25 - bayes.NaiveBayes

Classifier output

std. dev.	0.1922	0.4034
weight sum	67343	58630
precision	0.01	0.01

Time taken to build model: 0.83 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	113856	90.3813 %
Incorrectly Classified Instances	12117	9.6187 %
Kappa statistic	0.8059	
Mean absolute error	0.0965	
Root mean squared error	0.3058	
Relative absolute error	19.3981 %	
Root relative squared error	61.312 %	
Total Number of Instances	125973	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.936	0.134	0.890	0.936	0.912	0.807	0.967	0.964	normal
	0.866	0.064	0.922	0.866	0.893	0.807	0.965	0.949	anomaly
Weighted Avg.	0.904	0.101	0.905	0.904	0.904	0.807	0.966	0.957	

=== Confusion Matrix ===

a	b	<-- classified as
63058	4285	a = normal
7832	50798	b = anomaly

Fig: Summary of 10-fold cross validation

Step 4:

Upload test data and checking classification result

The screenshot shows the 'Classifier' application window. The 'Test options' panel on the left has 'Supplied test set' selected. The 'Classifier output' panel on the right displays the following text:

```
Time taken to build model: 0.92 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.6 seconds

=== Summary ===

Correctly Classified Instances      17161      76.1222 %
Incorrectly Classified Instances    5383      23.8778 %
Kappa statistic                    0.5366
Mean absolute error                 0.2386
Root mean squared error             0.4862
Relative absolute error             47.2755 %
Root relative squared error         96.0968 %
Total Number of Instances          22544

=== Detailed Accuracy By Class ===

              TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
              -----  -----  -
              0.931   0.367   0.657     0.931   0.771     0.572   0.895    0.844    normal
              0.633   0.069   0.924     0.633   0.751     0.572   0.917    0.911    anomaly
Weighted Avg.   0.761   0.197   0.809     0.761   0.759     0.572   0.908    0.882

=== Confusion Matrix ===

  a    b  <-- classified as
9041  670 |   a = normal
4713 8120 |   b = anomaly
```

Fig: Summary on uploaded test data

Step 5:

Compare results between 10-fold cross validation and test dataset.

Ground Truth\Classification	Normal - pred	Anomaly - pred
Normal - gt	63058	4285
Anomaly - gt	7832	50798

Table 1: 10-fold Cross validation result

Ground Truth\Classification	Normal - pred	Anomaly - pred
Normal - gt	9041	670
Anomaly - gt	4713	8120

Table 2: Test set result

Here, we can see the difference between cross validation and test set results.

Table 1 shows that during cross validation a total of 113856 samples were correctly classified (63058 + 50798) and 12117 were incorrectly classified (4285 + 7832).

Table 2 shows that during classification on test data a total of 17161 samples were correctly classified (9041+ 8120) and 5383 were incorrectly classified (4285 + 670).

Here, correct classification is constituted of two elements, True Positives and True Negatives, similarly, misclassification constitutes of two elements, False Positives and False Negatives.

True Positives: When sample is normal and classified as normal.

True Negatives: When sample is anomaly and classified as anomaly.

False Positives: When sample is anomaly and classified as normal.

False Negatives: When sample is normal and classified as anomaly.

(This is when normal is considered as positive and anomaly as negative, if we interchange the label assigned to these classes then the meaning will change accordingly).

Based on these values we have the following metrics:

	10 fold cross validation	Test data
Accuracy	90.38 %	76.12 %
Precision (Weighted avg)	90.5 %	80.9 %
Recall (Weighted avg)	90.4 %	76.1 %

We can see that there is a performance drop in test data as compared to 10-fold cross validation results. This indicates that the model does not generalize well on unseen data and is possibly overfitted.

Step 6: Five supervised classification algorithms

1. Decision Tree (J48):

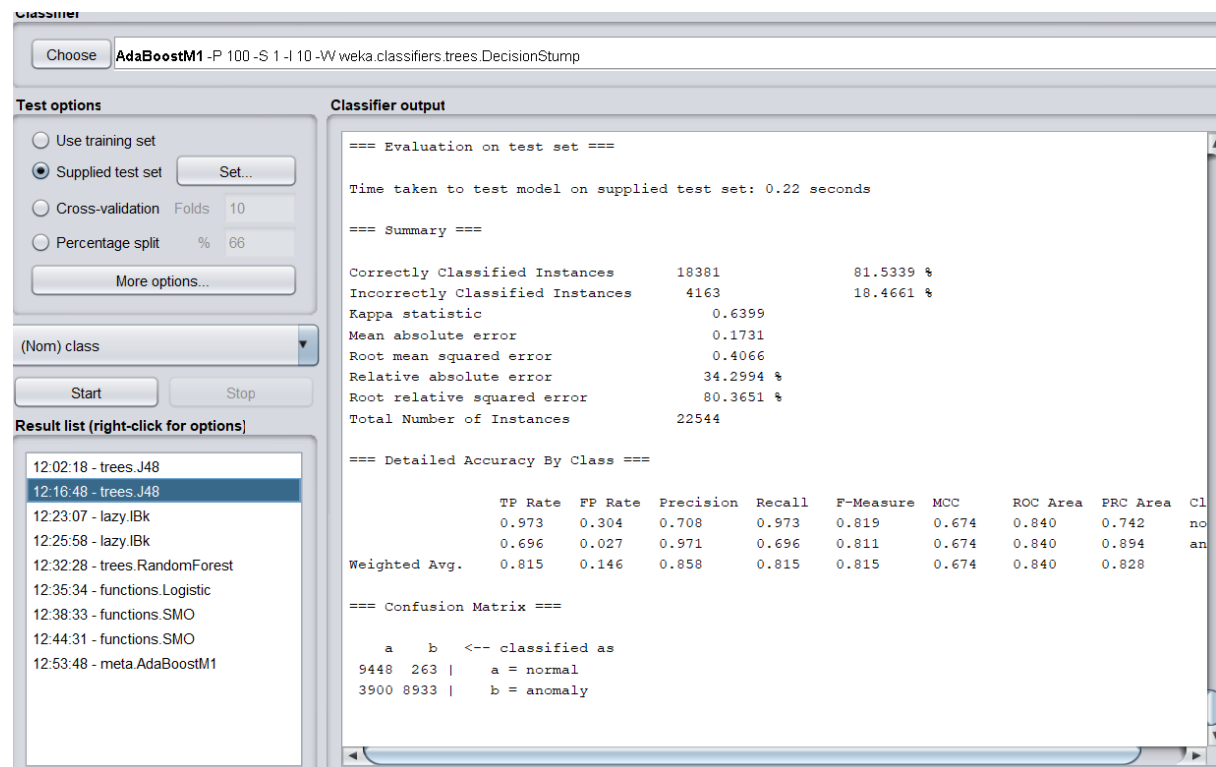


Fig: Training summary of Decision Tree (J48)

2. Instance based classifier (IBK):

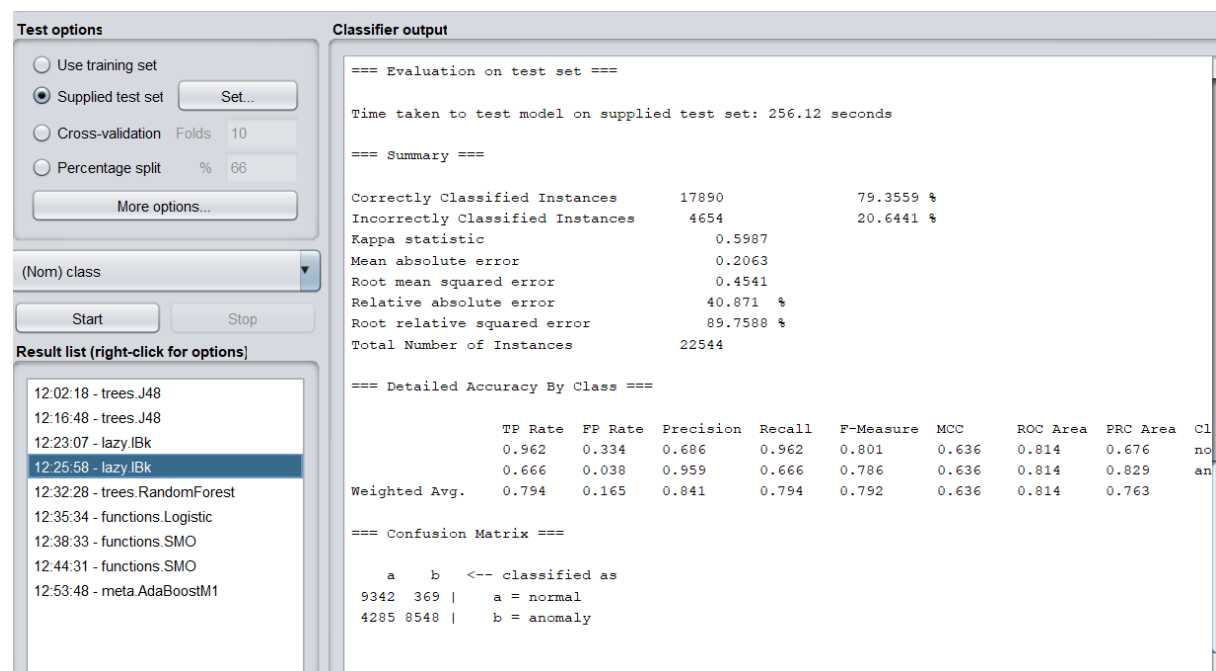


Fig: Training summary of IBK

3. Logistic Regression:

The screenshot shows the Weka Classifier window with the AdaBoostM1 classifier selected. The Test options are set to 'Supplied test set'. The Classifier output pane displays the following information:

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.22 seconds

=== Summary ===

Correctly Classified Instances	17045	75.6077 %
Incorrectly Classified Instances	5499	24.3923 %
Kappa statistic	0.5266	
Mean absolute error	0.2437	
Root mean squared error	0.4725	
Relative absolute error	48.2774 %	
Root relative squared error	93.388 %	
Total Number of Instances	22544	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.926	0.372	0.653	0.926	0.766	0.562	0.777	0.581	normal
	0.628	0.074	0.918	0.628	0.746	0.562	0.777	0.863	anomaly
Weighted Avg.	0.756	0.203	0.804	0.756	0.754	0.562	0.777	0.742	

=== Confusion Matrix ===

a	b	<-- classified as	
8988	723	a	= normal
4776	8057	b	= anomaly

Fig: Training summary of logistic regression

4. Random Forest:

The screenshot shows the Weka Classifier window with the AdaBoostM1 classifier selected. The Test options are set to 'Supplied test set'. The Classifier output pane displays the following information:

=== Evaluation on test set ===

Time taken to test model on supplied test set: 1.08 seconds

=== Summary ===

Correctly Classified Instances	18137	80.4516 %
Incorrectly Classified Instances	4407	19.5484 %
Kappa statistic	0.6199	
Mean absolute error	0.1968	
Root mean squared error	0.3794	
Relative absolute error	38.9795 %	
Root relative squared error	74.9868 %	
Total Number of Instances	22544	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.973	0.323	0.695	0.973	0.811	0.658	0.959	0.945	normal
	0.677	0.027	0.971	0.677	0.798	0.658	0.959	0.959	anomaly
Weighted Avg.	0.805	0.155	0.852	0.805	0.803	0.658	0.959	0.953	

=== Confusion Matrix ===

a	b	<-- classified as	
9447	264	a	= normal
4143	8690	b	= anomaly

Fig: Training summary of Random Forest

5. AdaboostM1:

Classifier

Choose **AdaBoostM1 -P 100 -S 1 -I 10 -W weka.classifiers.trees.DecisionStump**

Test options

☐ Use training set

☒ Supplied test set **Set...**

☐ Cross-validation Folds **10**

☐ Percentage split % **66**

More options...

(Nom) class

Start **Stop**

Result list (right-click for options)

- 12:02:18 - trees.J48
- 12:16:48 - trees.J48
- 12:23:07 - lazy.IBk
- 12:25:58 - lazy.IBk
- 12:32:28 - trees.RandomForest
- 12:35:34 - functions.Logistic
- 12:38:33 - functions.SMO
- 12:44:31 - functions.SMO
- 12:53:48 - meta AdaBoostM1**

Classifier output

```

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.17 seconds

=== Summary ===

Correctly Classified Instances      17684      78.4422 %
Incorrectly Classified Instances    4860      21.5578 %
Kappa statistic                    0.5815
Mean absolute error                 0.2363
Root mean squared error             0.4308
Relative absolute error             46.8199 %
Root relative squared error         85.1385 %
Total Number of Instances          22544

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Cl
          -----  -
normal    0.957    0.346    0.677    0.957    0.793    0.620    0.903    0.857    no
anomaly   0.654    0.043    0.953    0.654    0.775    0.620    0.903    0.918    an
Weighted Avg.    0.784    0.174    0.834    0.784    0.783    0.620    0.903    0.892

=== Confusion Matrix ===

  a  b  <-- classified as
9295 416 |  a = normal
4444 8389 | b = anomaly
  
```

Fig: Training summary of AdaboostM1

Comparison of evaluation metrics on Test Set

Algorithms	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Decision Tree (J48)	81.5 %	14.6 %	85.8 %	81.5 %	81.5 %	84 %
Instance based classifier (IBK)	79.4 %	16.5 %	84.1 %	79.4 %	79.2 %	81.1 %
Logistic Regression	75.6 %	20.3 %	80.4 %	75.6 %	75.4 %	77.7 %
Random Forrest	80.5 %	15.5 %	85.2 %	80.5 %	80.3 %	95.9 %
AdaboostM1	78.4 %	17.4 %	83.4 %	78.4 %	78.3 %	90.3 %

Table: Comparison of all the algorithms used. Best value for each metric is highlighted.

Step 7: Resampling data to 20% of its original size

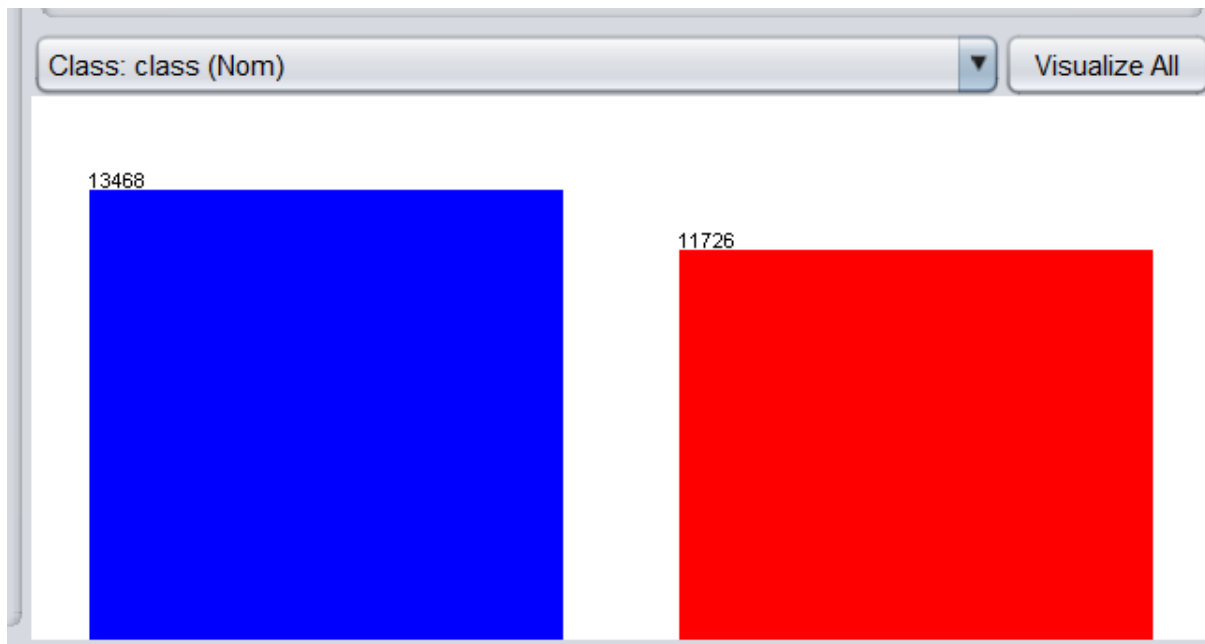


Fig: Resampled Data

Step 8: Training SVM classifier with RBF and POLY kernels on resampled data

SVM – RBF:

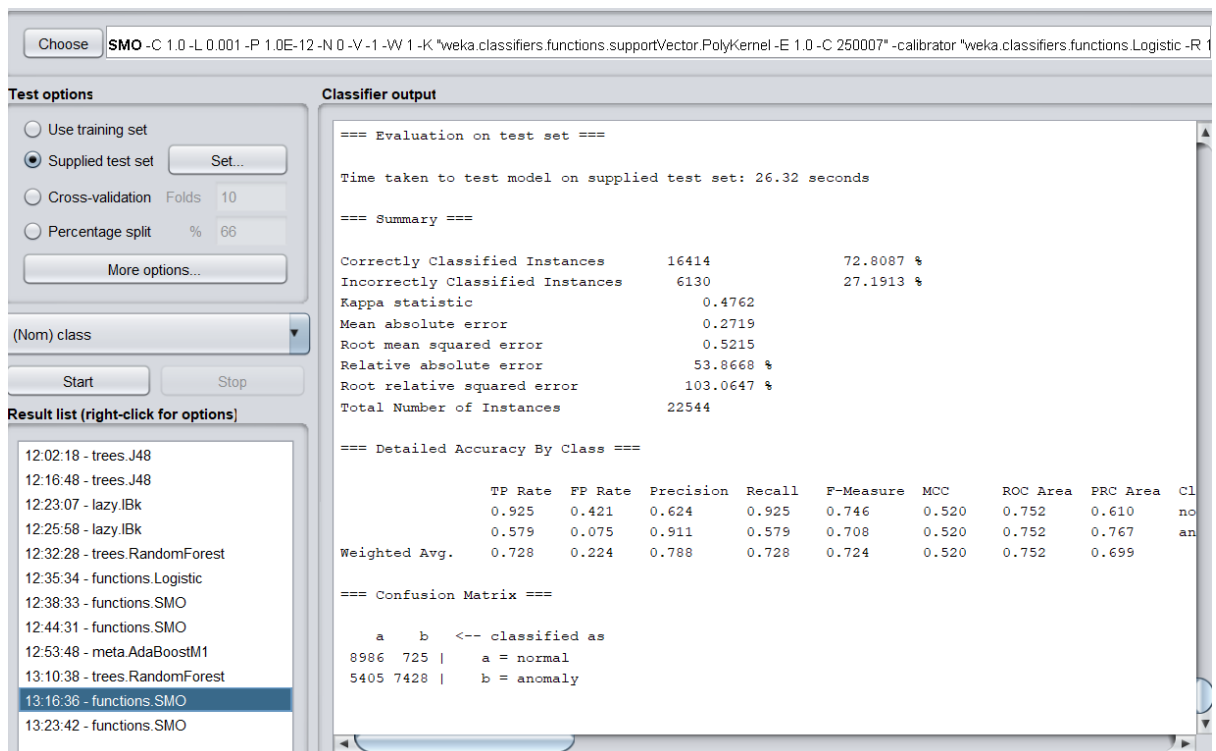


Fig: Training summary of SVM – RBF kernel

Model training time: 198.3 seconds

Prediction Time: 26.32 seconds

Confusion Matrix:

```
=== Confusion Matrix ===

      a    b  <-- classified as
8986  725 |    a = normal
5405 7428 |    b = anomaly
```

Evaluation Metrics:

Algorithm	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
SVM - RBF	72.8 %	22.4 %	78.8%	72.8 %	72.4 %	75.2 %

SVM – POLY:

The screenshot shows the Weka GUI with the 'Classifier output' pane selected. The output text is as follows:

```
Choose SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic -R "
```

Test options

- ☐ Use training set
- ☒ Supplied test set Set...
- ☐ Cross-validation Folds 10
- ☐ Percentage split % 66
- More options...

(Norm) class

Start Stop

Result list (right-click for options)

- 12:02:18 - trees.J48
- 12:16:48 - trees.J48
- 12:23:07 - lazy.IBk
- 12:25:58 - lazy.IBk
- 12:32:28 - trees.RandomForest
- 12:35:34 - functions.Logistic
- 12:38:33 - functions.SMO
- 12:44:31 - functions.SMO
- 12:53:48 - meta.AdaBoostM1
- 13:10:38 - trees.RandomForest
- 13:16:36 - functions.SMO
- 13:23:42 - functions.SMO

Classifier output

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.2 seconds

=== Summary ===

Correctly Classified Instances      16810           74.5653 %
Incorrectly Classified Instances    5734           25.4347 %
Kappa statistic                    0.5077
Mean absolute error                 0.2543
Root mean squared error             0.5043
Relative absolute error             50.387 %
Root relative squared error         99.6801 %
Total Number of Instances          22544

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall  F-Measure  MCC   ROC Area  PRC Area  Cl
                0.925   0.390   0.642     0.925   0.758     0.546  0.767    0.626    no
                0.610   0.075   0.915     0.610   0.732     0.546  0.767    0.780    an
Weighted Avg.   0.746   0.211   0.797     0.746   0.743     0.546  0.767    0.714

=== Confusion Matrix ===

      a    b  <-- classified as
8982  729 |    a = normal
5005 7828 |    b = anomaly
```

Fig: Training summary of SVM – POLY kernel

Model training time: 54.98 seconds

Prediction Time: 0.2 seconds

Confusion Matrix:

```

=== Confusion Matrix ===

      a      b  <-- classified as
8982  729 |      a = normal
5005 7828 |      b = anomaly

```

Evaluation Metrics:

Algorithm	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
SVM - POLY	74.6 %	21.1 %	79.7 %	74.6 %	74.3 %	76.7 %

Consolidated results of all the algorithms used:

Algorithms	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Naïve Bayes	76.1 %	19.7 %	80.9 %	76.1 %	75.9 %	90.8 %
Decision Tree (J48)	81.5 %	14.6 %	85.8 %	81.5 %	81.5 %	84 %
Instance based classifier (IBK)	79.4 %	16.5 %	84.1 %	79.4 %	79.2 %	81.1 %
Logistic Regression	75.6 %	20.3 %	80.4 %	75.6 %	75.4 %	77.7 %
Random Forest	80.5 %	15.5 %	85.2 %	80.5 %	80.3 %	95.9 %
AdaboostM1	78.4 %	17.4 %	83.4 %	78.4 %	78.3 %	90.3 %
SVM - RBF	72.8 %	22.4 %	78.8 %	72.8 %	72.4 %	75.2 %
SVM - POLY	74.6 %	21.1 %	79.7 %	74.6 %	74.3 %	76.7 %

Table: Comparison of all the algorithms used. Best value for each metric is highlighted (yellow). Last two rows show the difference between SVM (POLY and RBF) kernels.

Note: The SVM models are trained on resampled data (20 % of the entire data). Still yields good performance across all metrics.

Observations:

We can see that while progressing from Naïve Bayes to other more powerful algorithms we get more generalised and robust models. Here, we can also see Decision Tree outperforms other algorithms in 5 out of 6 evaluation metrics. Random Forest outperforms every other algorithm in ROC Area metric. This shows the strength of tree based models and shows how simple Decision Tree can be a powerful tool to design basic pipeline for complex machine learning tasks. They give an idea of a minimum best accuracy possible on any given dataset and allow us to visualize the features that are important for building a robust model. In addition to this, the ability to visualize what is being learnt via Decision Trees also makes them a very handy tool for machine learning tasks. This can be considered as a baseline benchmark accuracy before moving to other complex model for refining the performance metrics.

Other important highlight of this experiment is the use of ensemble techniques (Random Forest and AdaBoost). We can see that how ensemble techniques overcome limitation of individual models by showing a sharp increase in the ROC metric as compared to other models.

Another aspect of this experiment highlights the difference between SVM kernels, we can see that as we move from RBF kernel to a POLY kernel there is a significant gain in training and prediction time as well as gain in performance across all metrics. This highlights the strength of Polynomial kernel and shows why the famous “Kernel Trick” is so effective. The kernel trick highlights the fact that by *transforming the data into higher dimensions we can effectively find a boundary functions for data that otherwise would have been difficult to separate in lower dimensions*. We can also see that how SVM algorithm performs like the best performing models **while being trained on only 20%** of the entire dataset. This shows us that in case when there is a **lack of data, SVM models** should be the go-to approach as they need a smaller data size to yield high performing models.

This experiment also shows that how instance-based learning models (e.g., K Nearest Neighbours) are effective in providing insights into the kind of machine learning solution possible by quick modelling. In these types of models there is no actual learning that takes place, the strength of these models lies in the fact that they can utilize simple information such as distance between two points and perform classification as and when unseen data arrives. This is the reason why they are also called lazy learners.

Apart from the models used, this task also highlights the importance of using different performance or evaluation criteria to test the robustness and generalisability of a machine learning model. Sometimes, a general flaw is to only use accuracy to test the performance of machine learning models, this approach can be very misleading as we could end up with a model which is highly specialised on one dataset (the dominant class) and give us misleading figures. E.g., in the given example we might have used Logistic Regression as a well performing model, but evaluating it across all metrics highlights that the model suffers from a high FP rate which can be counter intuitive and lead to many false alarms in a attack scenario. This establishes that to evaluate a model’s performance, it should be tested across various performance metrics.