Attack Classification using Naïve Bayes Algorithm

Step 1:

Downloading the dataset and checking class distribution.

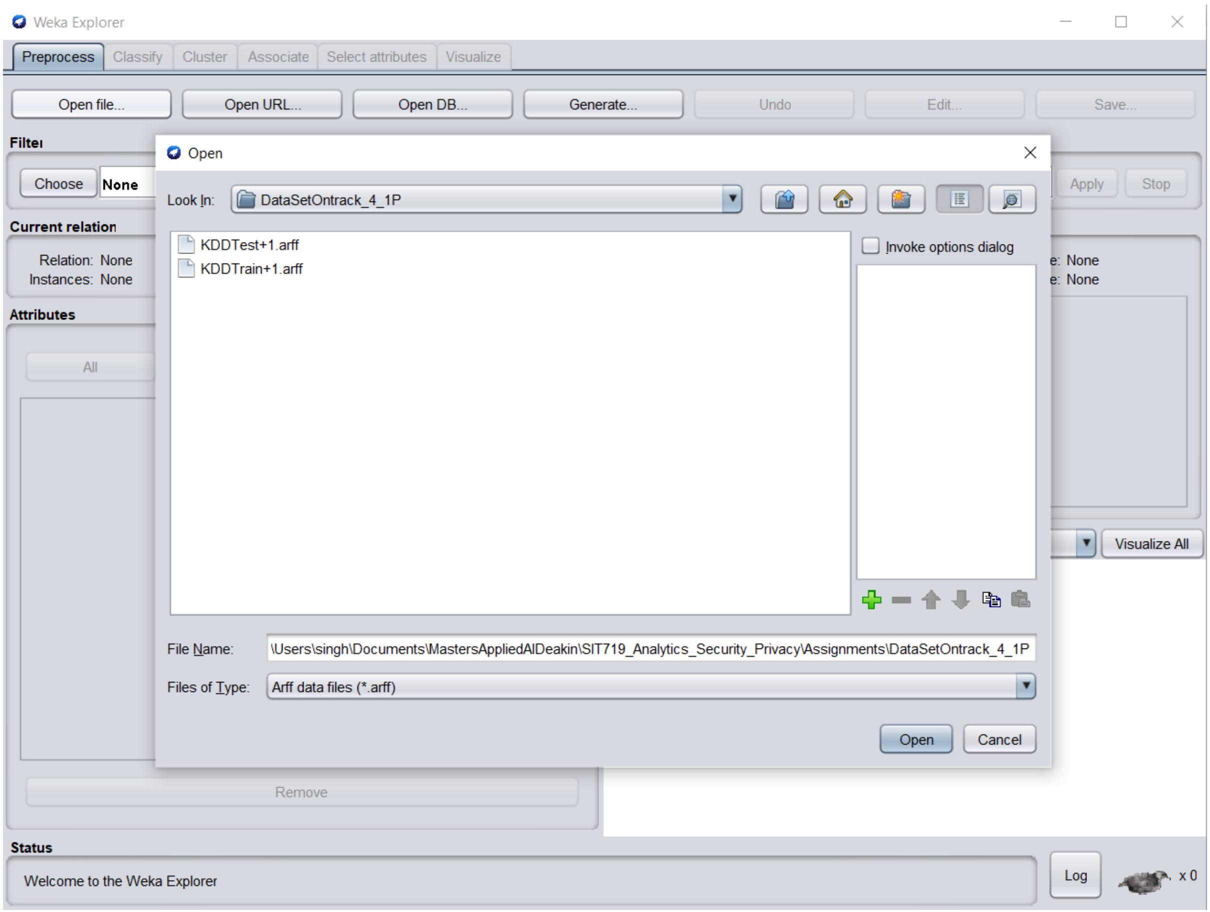

Fig: Downloaded data, train and test.

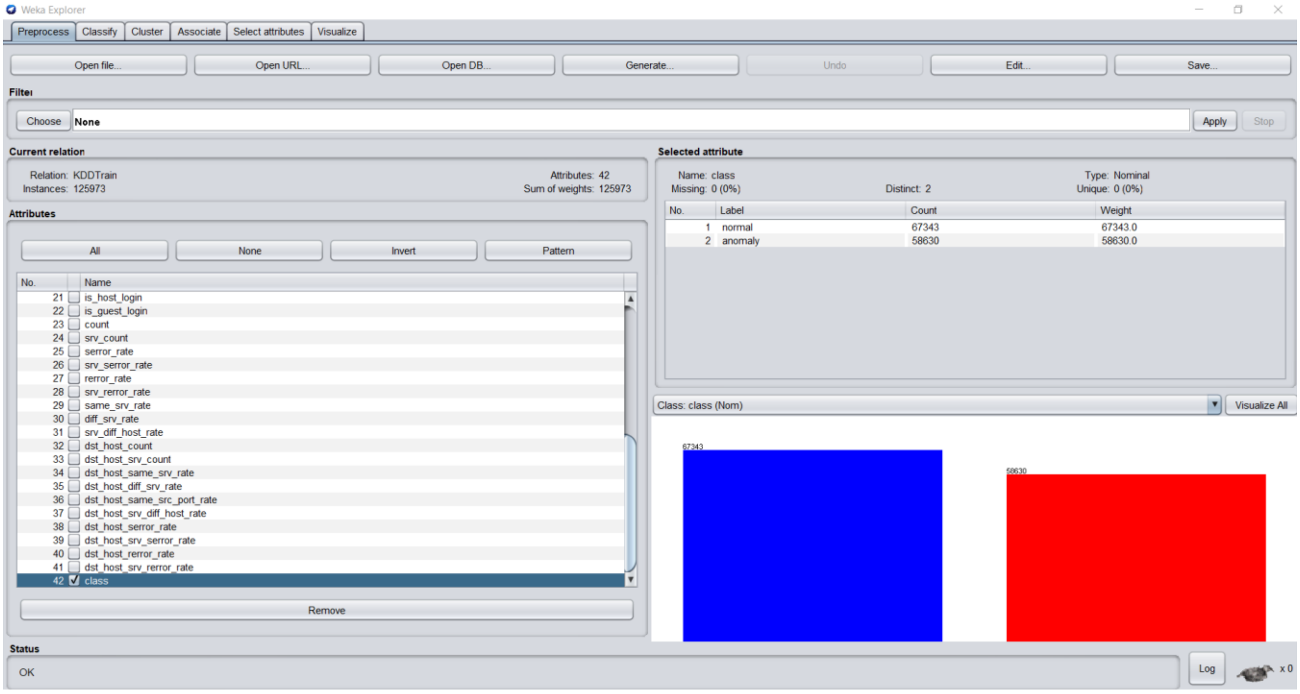

Fig: Data distribution

Step 2:

Applying Naïve Bayes Classifier

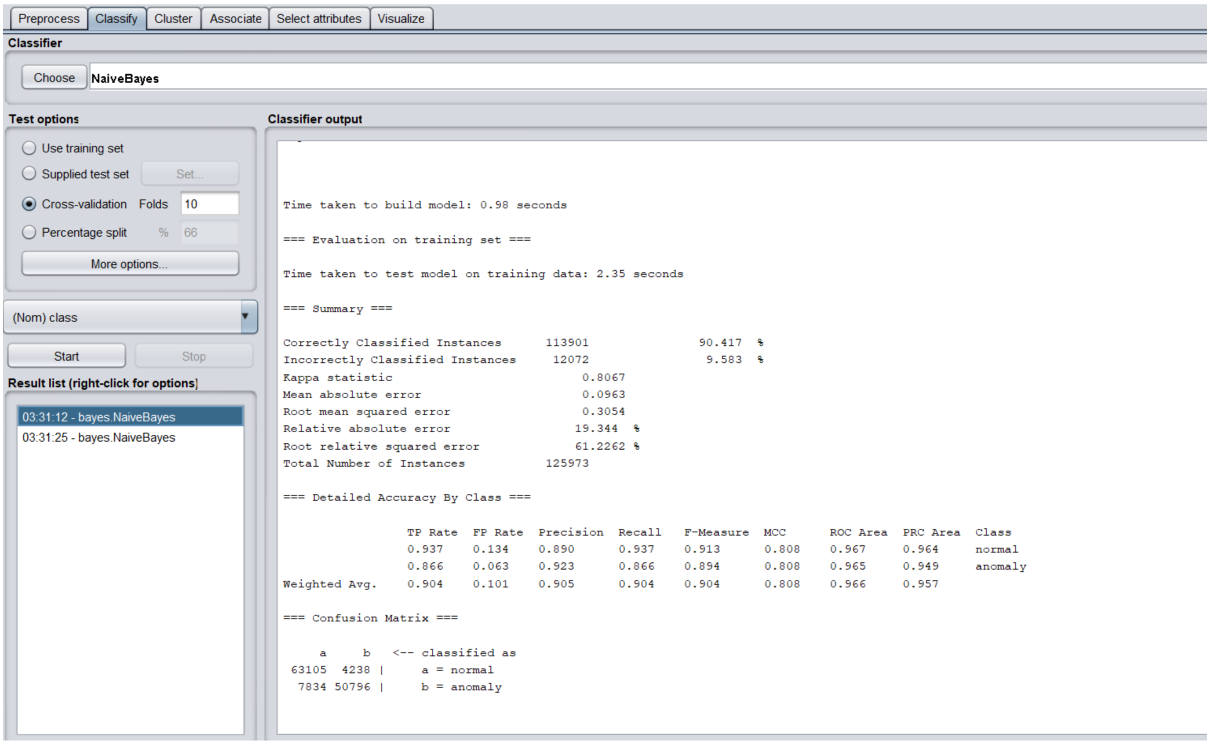

Fig: Classification summary after applying Naïve Bayes classifier.

Step 3:

Performing 10-fold cross validation

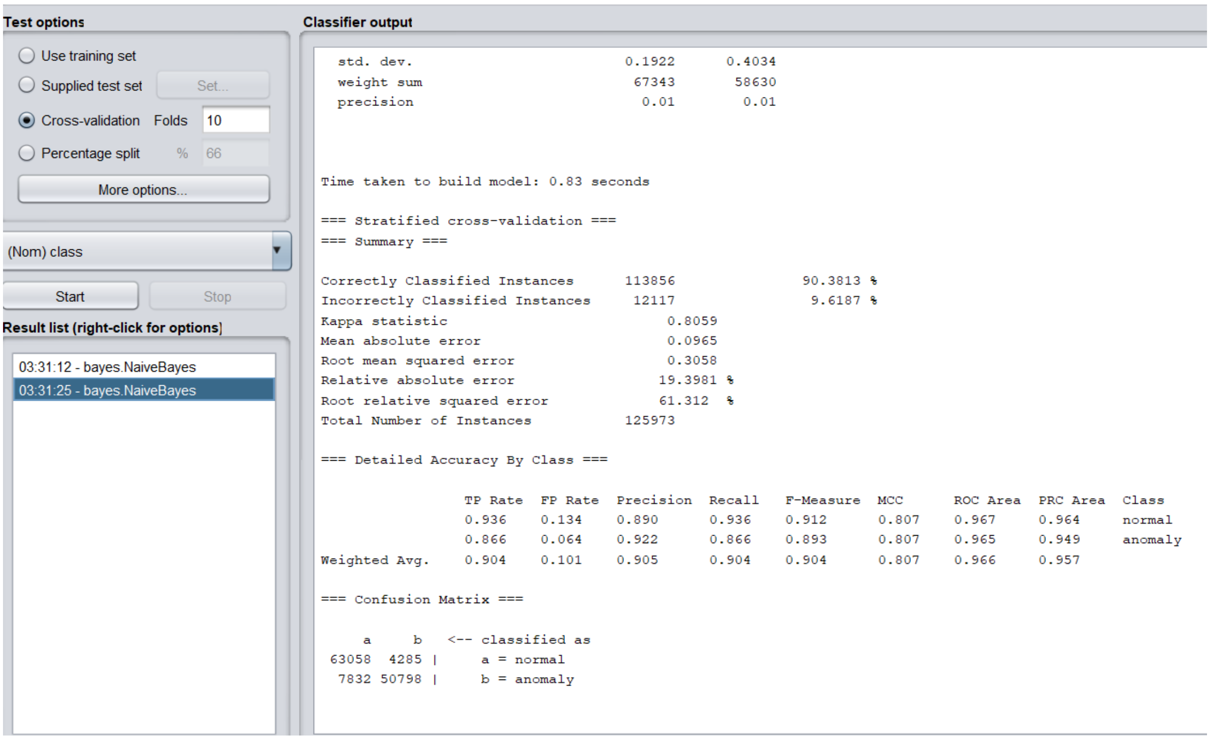

Fig: Summary of 10-fold cross validation

Step 4:

Upload test data and checking classification result

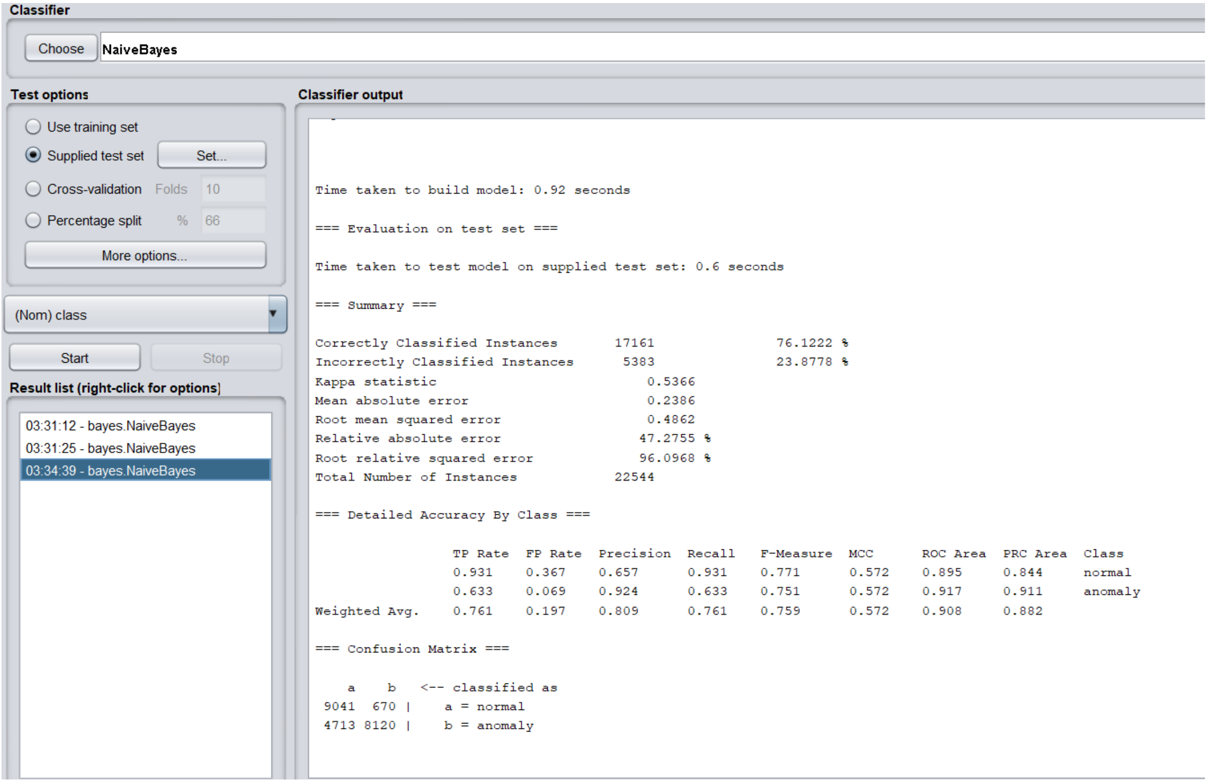

Fig: Summary on uploaded test data

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):

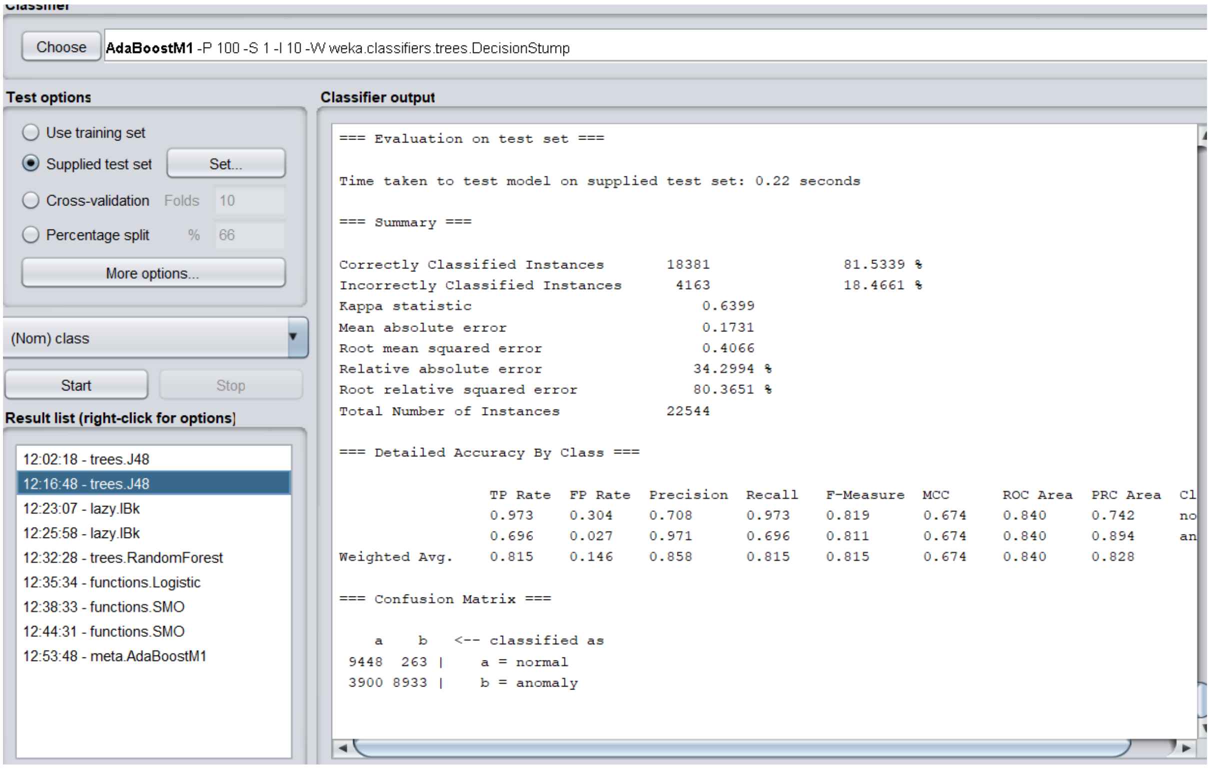

Fig: *Training summary of Decision Tree (J48)*

2. Instance based classifier (IBK):

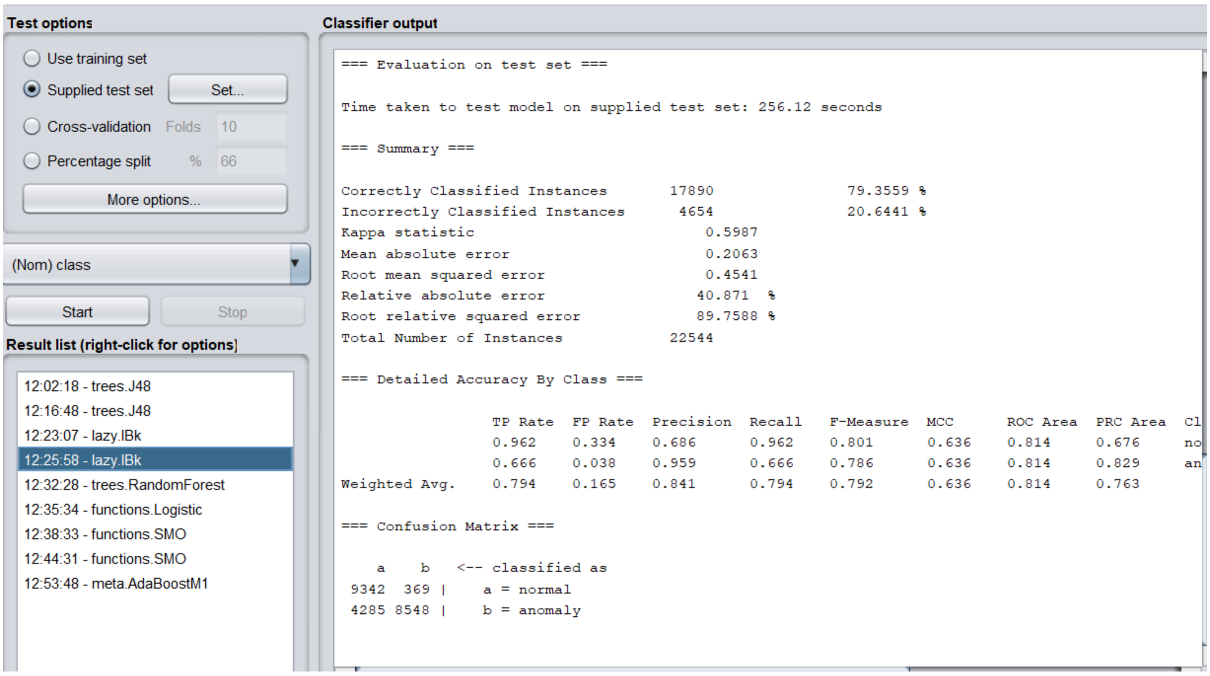

Fig: *Training summary of IBK*

3. Logistic Regression:

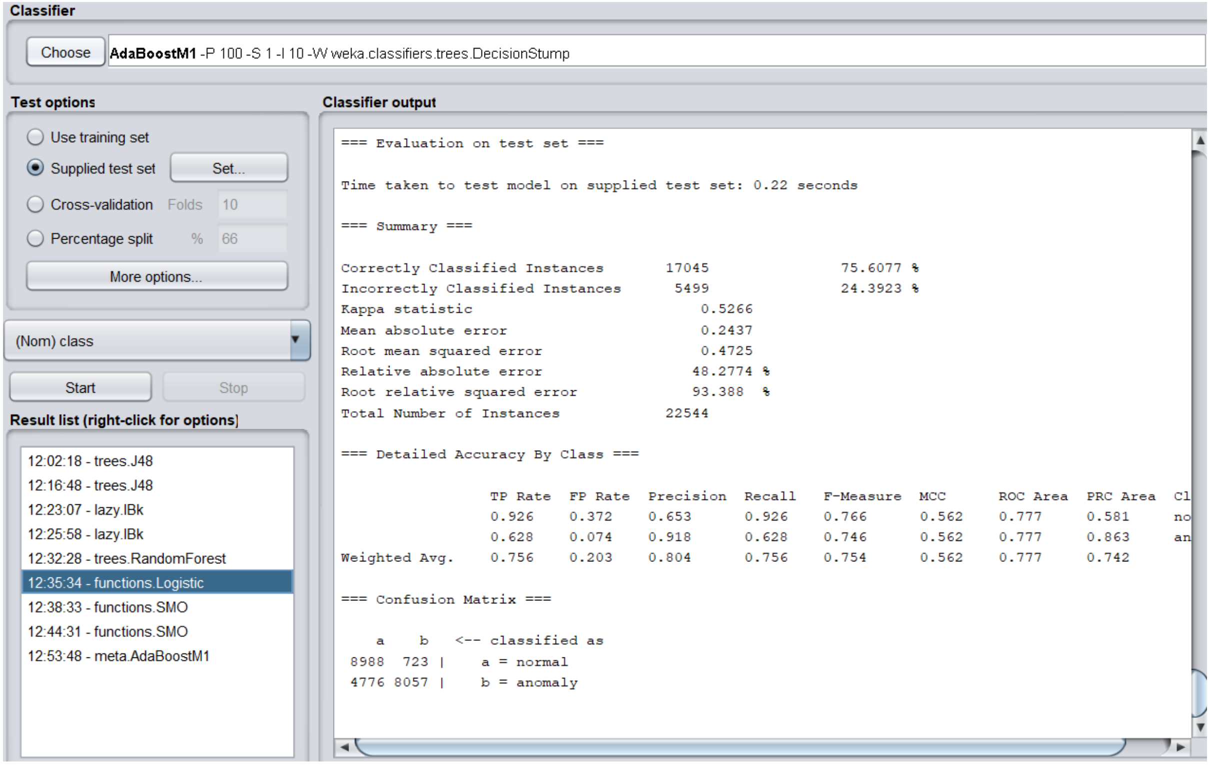

Fig: *Training summary of logistic regression*

4. Random Forest:

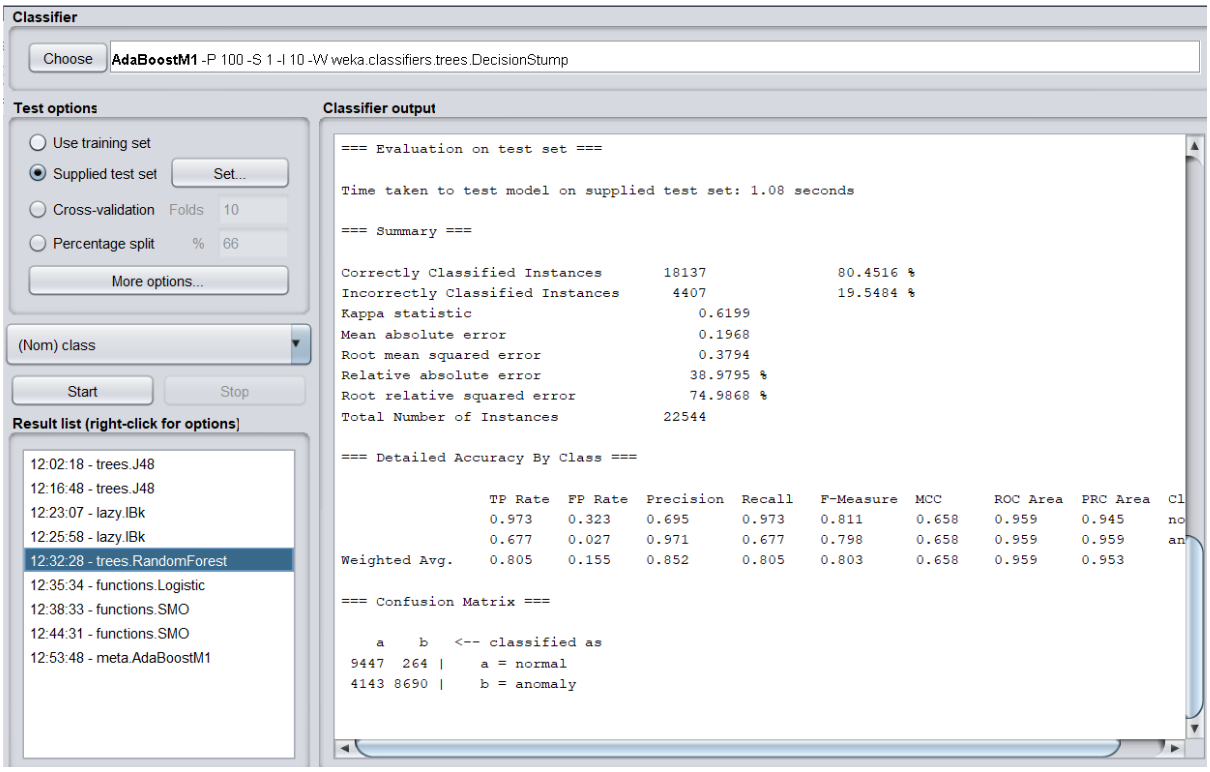

Fig: *Training summary of Random Forest*

5. AdaboostM1:

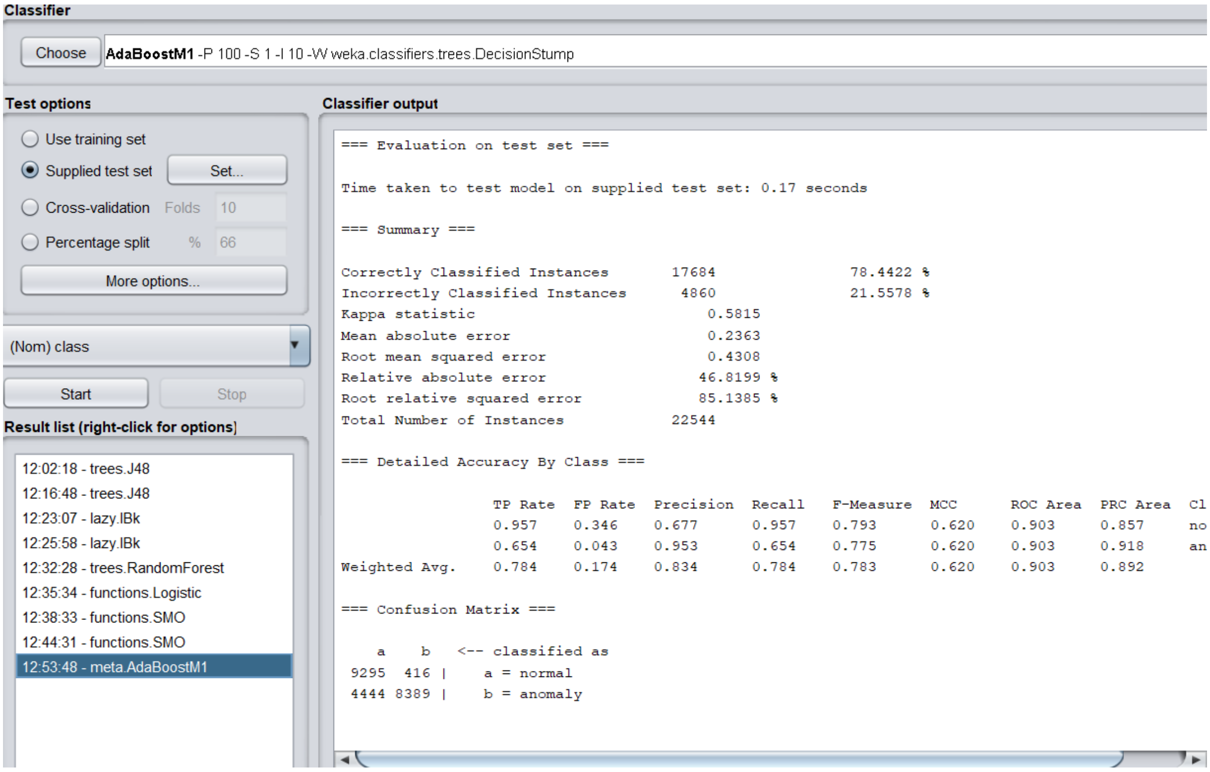

Fig: *Training summary of AdaboostM1*

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:

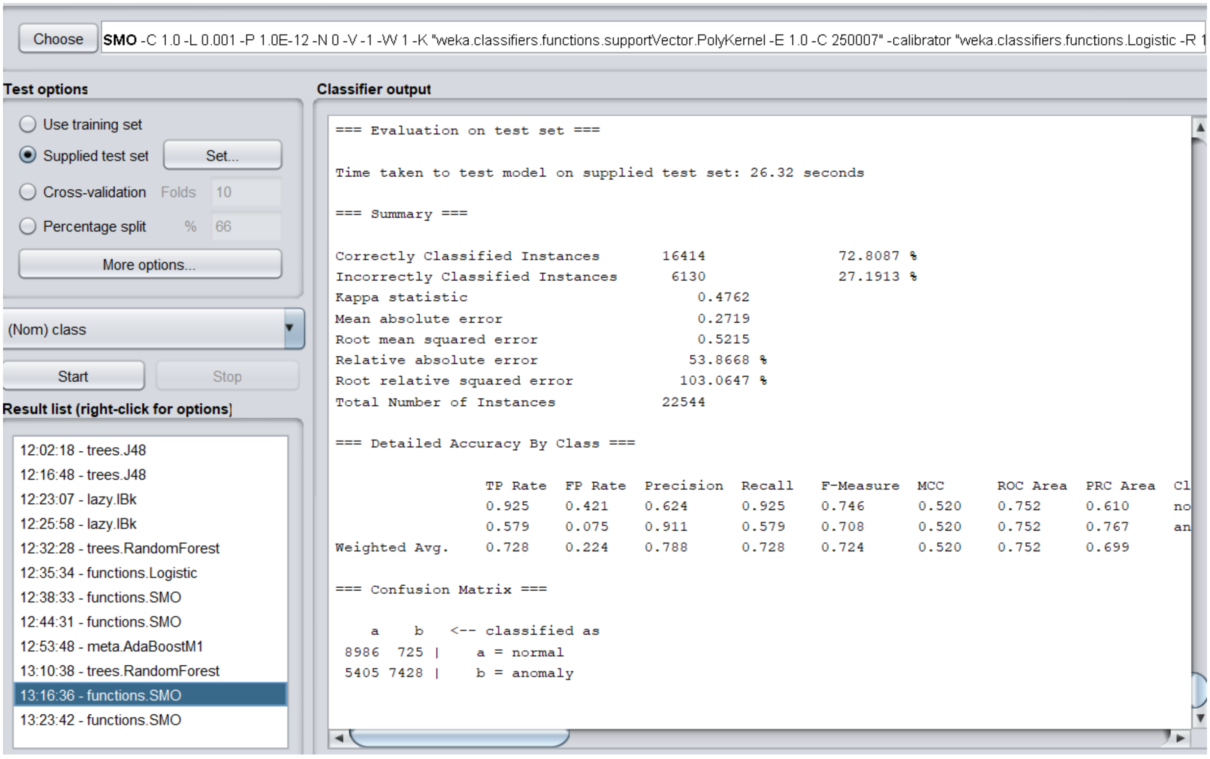

***Fig:** Training summary of SVM – RBF kernel*

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:

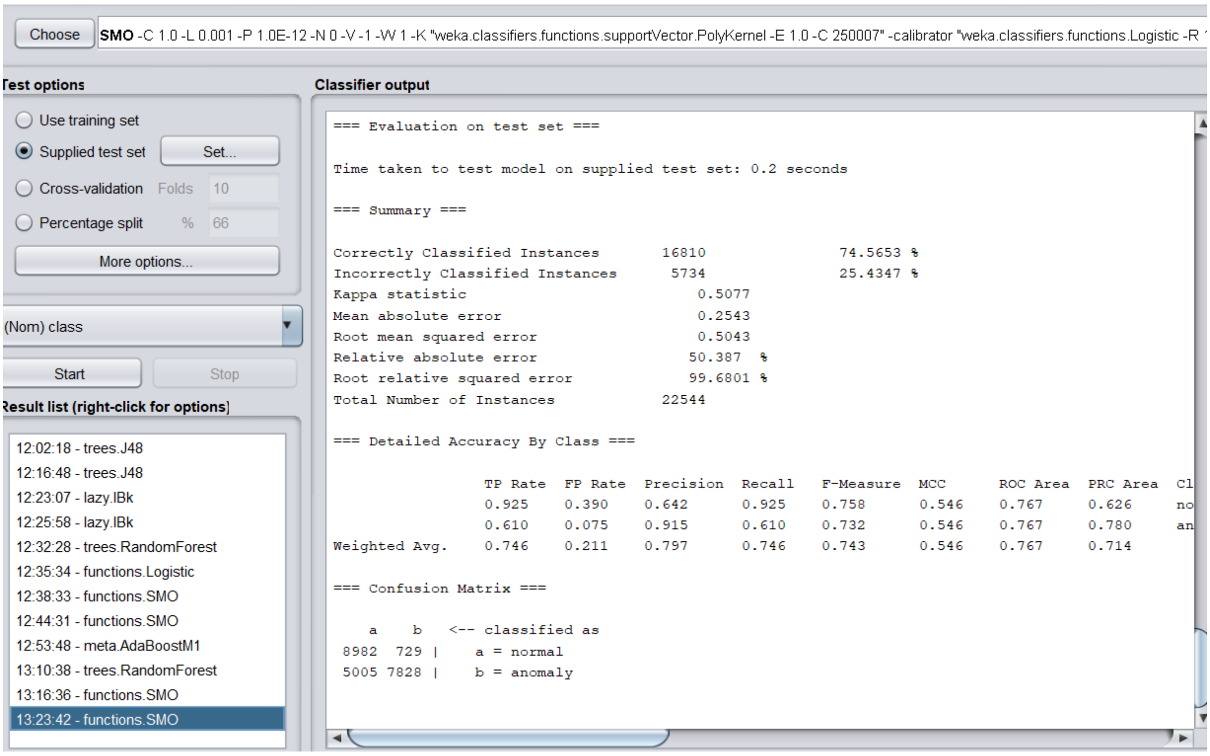

***Fig:** Training summary of SVM – POLY kernel*

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 a 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.