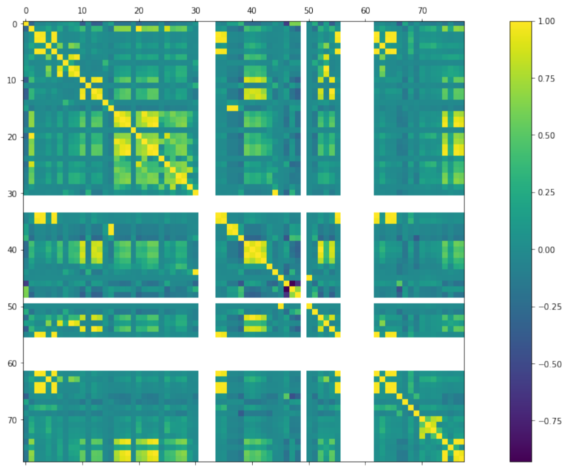
Binary Problem:

Data Overview:

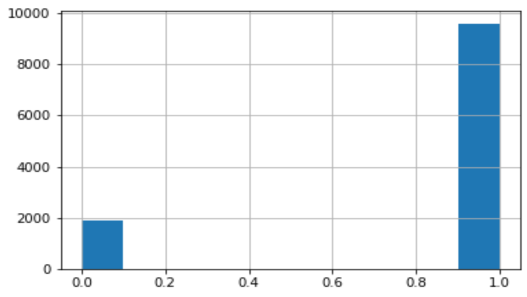
It’s a binary labeled data that classifies the network intrusion,
whether it’s an attack or benign.
Data has many records,
as it has 25K records with 79 features,
almost the majority of the features are correlated.
We need to reduce the complexity of the model.
We have cleaned the data and dropped both of NANs and duplicated values , after that the data dimensions became 11K.



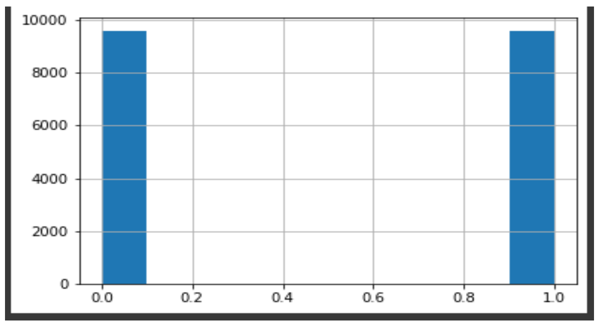
Correlation matrix

We have observed the distribution of our class, and we saw that the problem is an unbalanced learning problem, hence we have used techniques for oversampling such as SMOTE.

Class labels before smote

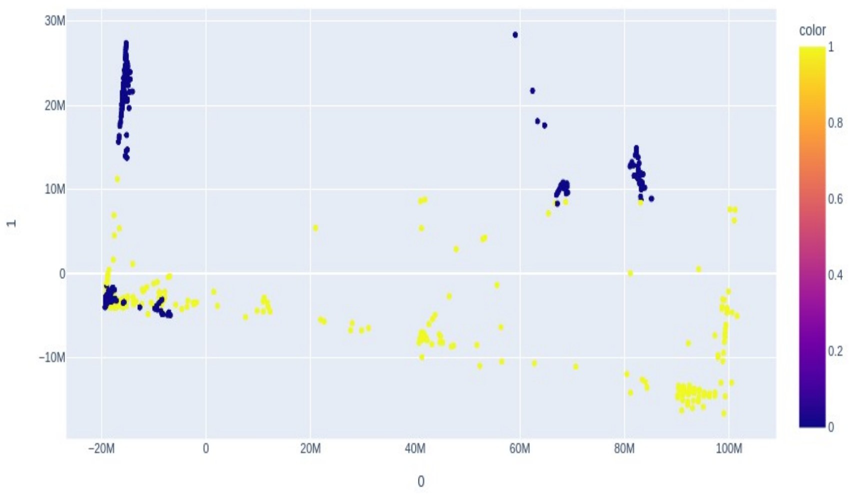


Class labels after smote



In order to reduce the model complexity and reduce the dimensions of the data, we have used the forward method for feature selection.
This method outputs exactly the features that seem to be more useful than others.

Before modeling, we applied PCA to our data to convert the data dimensions into 2D, hence we can observe data complexity and purity.

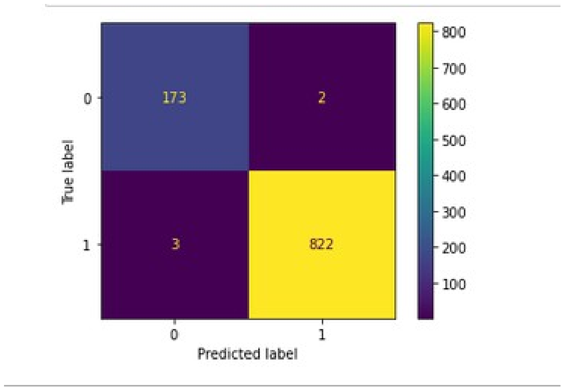


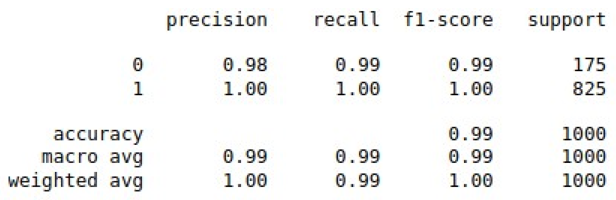
From the figure in right, we can see that the outliers in the data are too many, hence we kept them to avoid over-fitting as we are going to model with random forest and tree algorithms. In addition to almost not having an overlap between the two classes shows that the data is after feature selection is not complex, hence we excepect the model performance to be a good fit.

Algorithms:

We are modeling with random forest algorithm.
Random Forest Baseline:

Experiments:





```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       175
           1       1.00      1.00      1.00       825

    accuracy                           0.99      1000
   macro avg       0.99      0.99      0.99      1000
weighted avg       1.00      0.99      1.00      1000
```

Metrics selected:

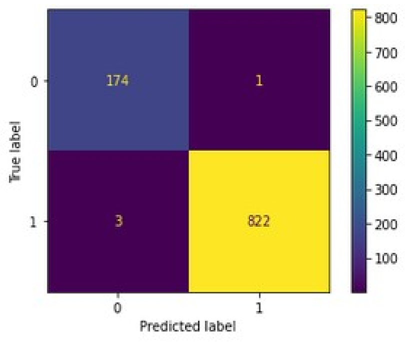
We selected the accuracy after resolving the imbalanced learning problem in addition to the precision and recall to evaluate the binary classification in each class.

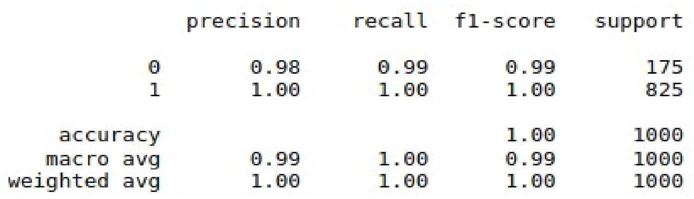
Hyper-parameters tuning:

We have applied the random search on a wide range of parameters, main parameters were to set the max-depth to 20 and the min-samples-leaf to 4 and more.

Experiments:

Random Forest with hyper parameters tuning:
The model performance has slightly improved , as in the first class the f1 score increased up till 100%.





```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       175
           1       1.00      1.00      1.00       825

    accuracy                           1.00      1000
   macro avg       0.99      1.00      0.99      1000
weighted avg       1.00      1.00      1.00      1000
```

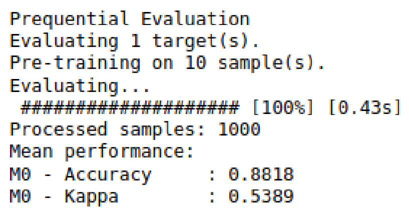
Adaptive Learning:

We have applied the HoeffdingTreeClassifier on the data streamed using skmultiflow library.
And here is the output.

metrics selected:
We are applying the kappa and the accuracy as an evaluation metric. Hence we can use the accuracy after oversampling and resolving the problem of the imbalanced learning.

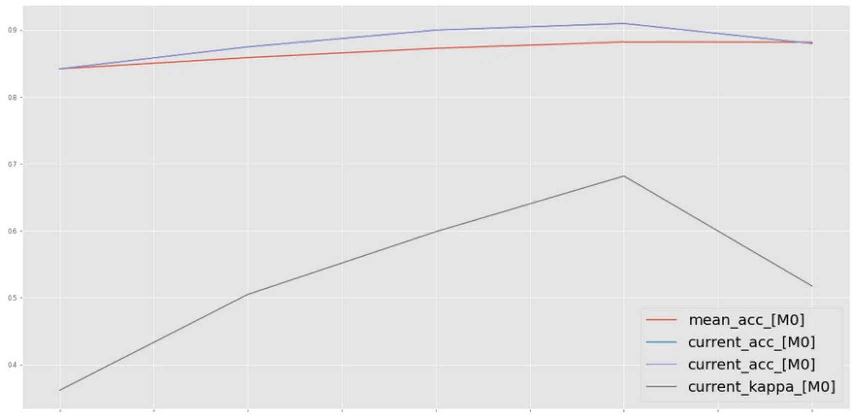
Experiments:

The accuracy is decreased as the mean accuracy within streams was only 0.88.



```
Prequential Evaluation
Evaluating 1 target(s).
Pre-training on 10 sample(s).
Evaluating...
 #################### [100%] [0.43s]
Processed samples: 1000
Mean performance:
M0 - Accuracy     : 0.8818
M0 - Kappa        : 0.5389
```

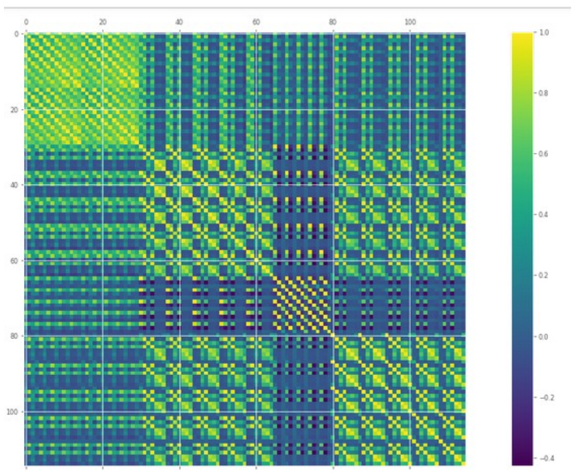
Comparing between mean and current metrics results:



Multi-class Problem:

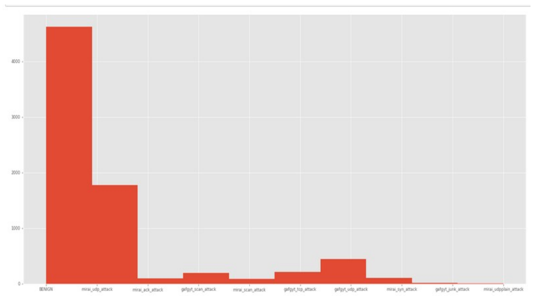
Data Overview:

It’s a multi-class data that classifies IOT Botnet Attack.
Data has many records,
as it has 25K records with 117 features,
almost the majority of the features are correlated.
We need to reduce the complexity of the model.
We have cleaned the data and dropped both of NANs and duplicated values , after that the data dimensions became 7K.

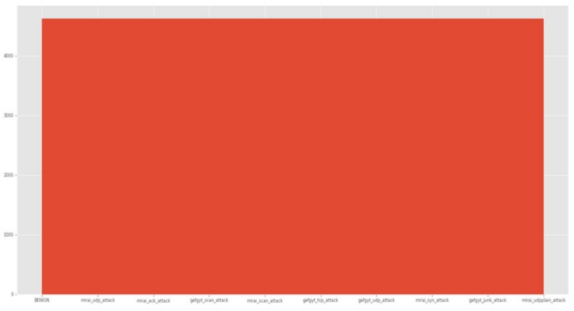

Correlation matrix

We have observed the distribution of our class, and we saw that the problem is an unbalanced learning problem, hence we have used techniques for oversampling such as SMOTE.

Class labels before smote



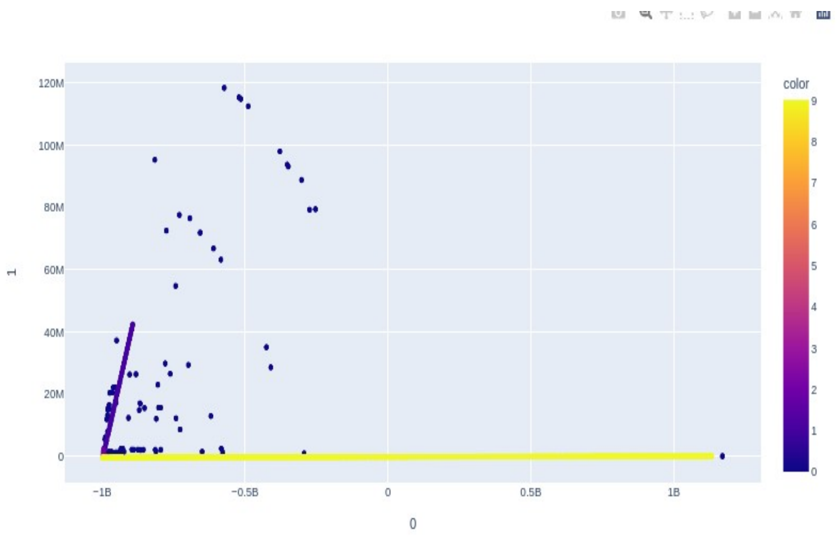
Class labels after smote



In order to reduce the model complexity and reduce the dimensions of the data, we have used the forward method for feature selection.
This method outputs exactly the features that seem to be more useful than others.

Before modeling, we applied PCA to our data to convert the data dimensions into 2D, hence we can observe data complexity and purity.

From the figure in right, we can see that the outliers in all classes are almost zeros except for one class.

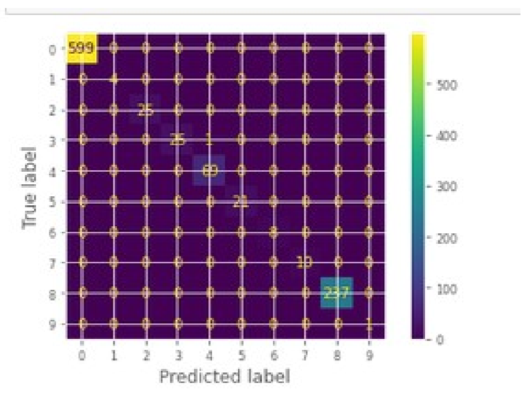


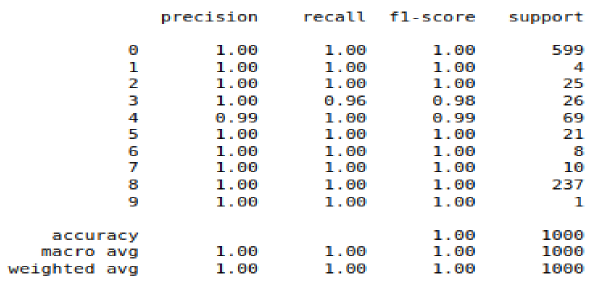
We can estimate that it may be the attack class as it has an anomaly rather that the other classes.
Again we are going to keep the data as it to test the model how it’s going to perform to distinguish between the classes and this full of outliers class.
We don do not have an overlap between the classes shows that the data is after feature selection is not complex, hence we exepect good performance.

Algorithms:

We are modeling with random forest algorithm.
Random Forest Baseline:

Experiments:





```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       599
           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00        25
           3       1.00      0.96      0.98        26
           4       0.99      1.00      0.99        69
           5       1.00      1.00      1.00        21
           6       1.00      1.00      1.00         8
           7       1.00      1.00      1.00        10
           8       1.00      1.00      1.00       237
           9       1.00      1.00      1.00         1

    accuracy                           1.00      1000
   macro avg       1.00      1.00      1.00      1000
weighted avg       1.00      1.00      1.00      1000
```

Metrics selected:

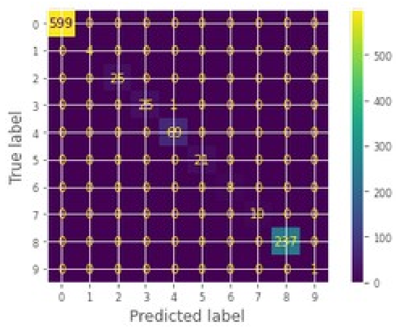
We selected the accuracy after resolving the imbalanced learning problem in addition to the precision and recall to evaluate the binary classification in each class.
It seems that the model can identify between the 10 classes we have, but we doubt that it may be an overfitting as the training examples are reduced to only cleaned 7K rows after they were 25K rows.

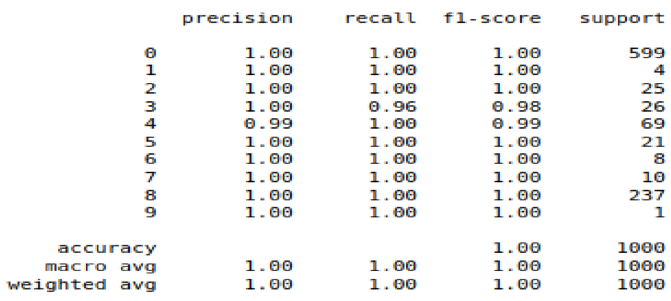
Hyper-parameters tuning:

We have applied the random search on a wide range of parameters, main parameters were to set the max-depth to 59 and the min-samples-leaf to 1 and number of estimators up till 1200 and more.

Experiments:

Random Forest with hyper parameters tuning:
The model performance had no change.





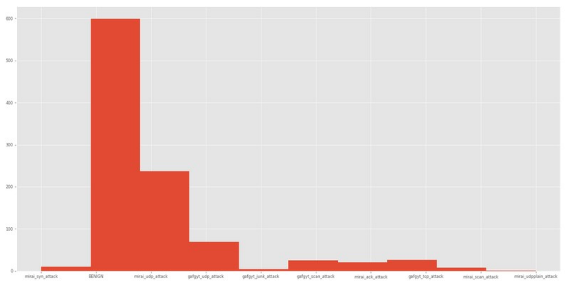
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       599
           1       1.00      1.00      1.00         4
           2       1.00      1.00      1.00        25
           3       1.00      0.96      0.98        26
           4       0.99      1.00      0.99        69
           5       1.00      1.00      1.00        21
           6       1.00      1.00      1.00         8
           7       1.00      1.00      1.00        10
           8       1.00      1.00      1.00       237
           9       1.00      1.00      1.00         1

    accuracy                           1.00      1000
   macro avg       1.00      1.00      1.00      1000
weighted avg       1.00      1.00      1.00      1000
```

Adaptive Learning:

The data is also unbalanced, but we do not rebalance the data as it’s a streaming data in real-life projects, but here we apply the same techniques of oversampling and feature selection to enable us to compare between the two methods

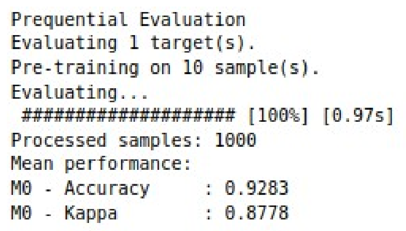


We have applied the HoeffdingTreeClassifier on the data streamed using skmultiflow library.
And here is the output.

metrics selected:
We are applying the kappa and the accuracy as an evaluation metric. Hence we can use the accuracy after oversampling and resolving the problem of the imbalanced learning.

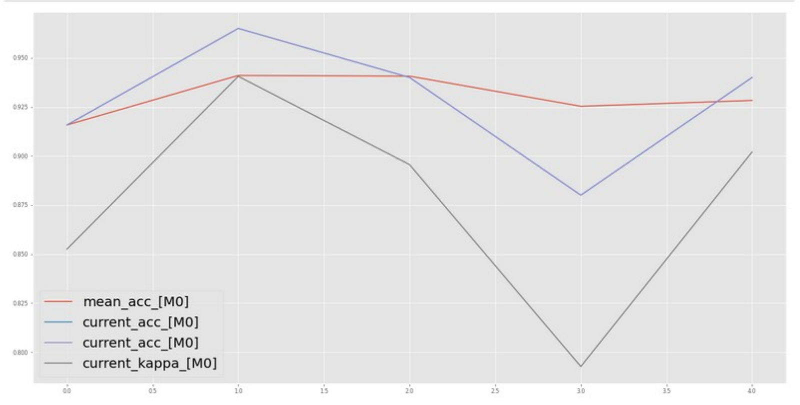
Experiments:

The accuracy is decreased as the mean accuracy within streams was only 0.92.



```
Prequential Evaluation
Evaluating 1 target(s).
Pre-training on 10 sample(s).
Evaluating...
 #################### [100%] [0.97s]
Processed samples: 1000
Mean performance:
M0 - Accuracy     : 0.9283
M0 - Kappa        : 0.8778
```

Comparing between mean and current metrics results:
The mean accuracy is 92.8% which is not far away from the static solution.



In addition to, both of accuracy and kappa are not stable while streaming.

Summary:

We except that the static solution is better than the dynamic one, as the model is being confused from dynamic solutions drawbacks such as the drifting concept. Streaming data is tricky to deal with and needs more intelligent solutions than the traditional machine learning solutions.

References:

[1] https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/
[2] https://towardsdatascience.com/putting-ml-in-production-i-using-apache-kafka-in-python-ce06b3a395c8
[3] https://stackoverflow.com/questions/50141544/kafka-to-pandas-dataframe-without-spark
[4] https://stackoverflow.com/questions/59404567/pandas-how-to-convert-dictionary-to-transposed-dataframe
[5] https://stackoverflow.com/questions/41207014/how-can-i-set-index-while-converting-dictionary-to-dataframe
[6] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.append.html
[7] https://stackoverflow.com/questions/11346283/renaming-column-names-in-pandas
[8] https://riverml.xyz/latest/api/ensemble/AdaptiveRandomForestClassifier/
[9]https://www.geeksforgeeks.org/pandas-dataframe-iterrows-function-in-python/
[10] https://stackoverflow.com/questions/39839034/python-tuple-indices-must-be-integers-not-tuple
[11]https://kafka-python.readthedocs.io/en/master/apidoc/KafkaConsumer.html
[12]https://scikit-multiflow.github.io/
[13]https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.trees.HoeffdingTreeClassifier.html
[14] https://www.kite.com/python/answers/how-to-change-the-font-size-of-a-matplotlib-legend-in-python