Task-1 : Network Intrusion Detection problem (Binary Classification)

1. **Static way:** In which we take the data and perform on it the needed

   preprocessing to be safely passed to the baseline model and predict the initial model(static) using Pandas, NumPy and Matplotlib.

   First, the data consists of **79** columns containing the “Label” column which have “BENIGN” or “ATTACK” values.

   The data had few **inf** and **-inf** values hence it was necessary to drop all these records to be passed smoothly to the model, Finally, the data is ready to be trained by the baseline model.

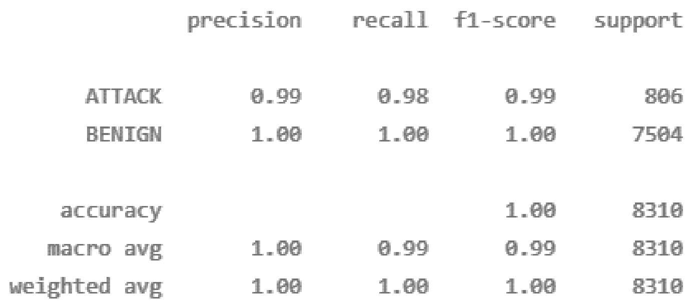
   The data was split by using Sklearn function called **train_test_split,** the test data length was 0.33 from the whole dataset, I have used **Random Forest** which was imported from Sklearn.ensemble library as the baseline model with these settings:

Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
bootstrap	True	max_samples	None	n_ estimators	None
ccp_alpha	0.0	min_impurity_ decrease	0.0	n_jobs	False
class_weight	None	min_impurity_split	None	oob_score	None
criterion	gini	min_samples_leaf	1	random_ state	0
Max_depth	None	min_samples_split	2	verbose	False
Max_features	auto	Random_state	0.0		
max_leaf_nodes	None	min_weight_fraction_le af	100		

Evaluation

The model has an accuracy that is very close to 1 which means that the model predicted all labels correct 100% correct. The model has also 0.99 F1-score which means that the model can differentiate between both classes near 99% correct.

The data here is imbalanced, hence accuracy won't give us a very clear performance about the model so using f1-score will be much important cause it focuses on misclassified records.



```
              precision    recall  f1-score   support

      ATTACK       0.99      0.98      0.99       806
      BENIGN       1.00      1.00      1.00      7504

    accuracy                           1.00      8310
   macro avg       1.00      0.99      0.99      8310
weighted avg       1.00      1.00      1.00      8310
```

2. **Adaptive Solution:**
   Using Kafka data streams, two models will be tested on the new data and one of them would be re-trained on both the new data and the old data to keep track of the changing performance over time, **Random forest** will be used on this experiment for both adaptive and static implementation.
   Kafka Setup: Kafka Consumer was used to start any connection with Kafka Server you must initialize KafkaConsumer session to start getting messeges from the server, and below config parameters controlling it.

parameter	**value**	**Parameters**	**Value**
	'task_1'	security_protocol	"SASL_PLAINTEXT"
bootstrap_servers	"34.130.121.39:9092"	sasl_mechanism	"PLAIN"
sasl_plain_username	"student"	auto_offset_reset	'earliest'
sasl_plain_password	"Uottawa"	enable_auto_commit	False

Methodology:

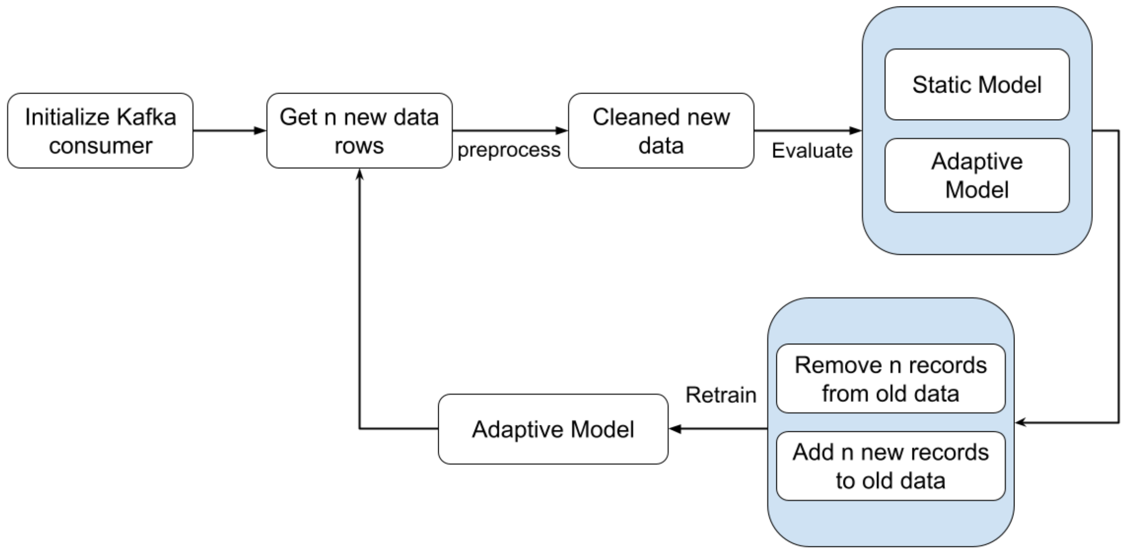


Figure 1-adaptive method

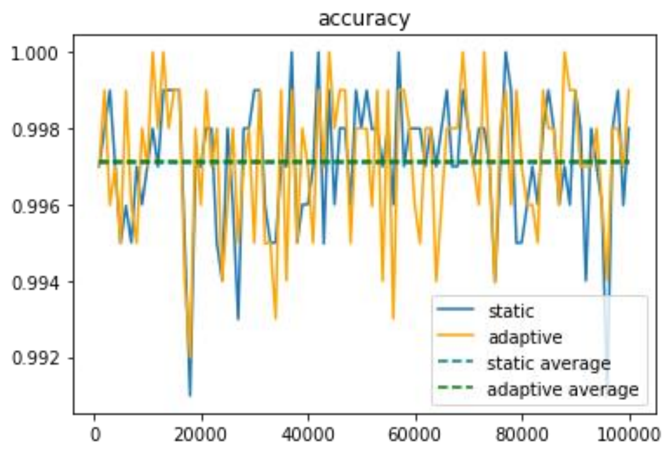
Sliding window settings:

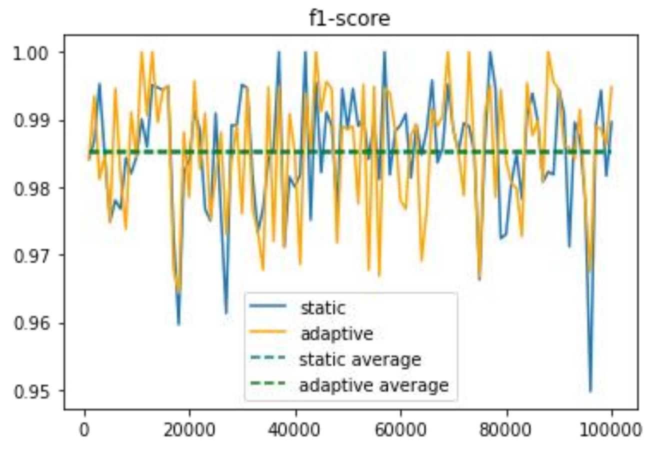
- window size of 1000 packets were used
- 100 iteration ~ 100,000 packets were used
- Original data size = 25000 records

Sliding window steps:

1. every 1000 packets (unseen) were tested using both adaptive and static models (Both Random Forest).
2. After that 1000 records will be dropped from the original data (same size as the window size)
3. Then a combination of original and stream data (stream packets = 1000 + original data = 24000) was used to retrain the adaptive model (Random Forest).
4. Then we will do the same steps until we reach 100,000 packets meaning that we have replaced all the original data then we will stop and to evaluate both static and dynamic models.

Evaluation and results:





```
Average of accuracy for adaptive model :0.9971549989466085
Average of accuracy for static model :0.9971054212501955

Average of f1-score for adaptive model :0.985391552559074
Average of f1-score for static model :0.9851388914980862
```

Accuracy and F1-scores were used to evaluate both models.

We can see that both adaptive and static have nearly the same average scoring on both accuracy and f1-score, but adaptive model has a very slight improvement with 0.00005 on accuracy and 0.0001 on f1-score.

At First, the adaptive model has better performance till we reached the first 25,000 packets, after that there were a lot of fluctuations and the adaptive one didn't seem to have better performance than the static one until the data reached 80,000 packets the adaptive model had a slight increase in performance again over the static one.

The results may be near to each other because the new streaming may be like the original ones so fitting one model of them won't affect much the accuracy or f1-scores.

Task-2 : IOT Botnet Attack Detection problem (Multi-Classification)

1. **Static way:** In which we take the data and perform on it the needed preprocessing to be safely passed to the baseline model and predict the initial model(static) using Pandas, NumPy and Matplotlib.

First, the data consists of **117** columns containing the "Class" column which have "BENIGN" and 10 different types of attacks

The data had few **inf** and **-inf** values hence it was necessary to drop all these records to be passed smoothly to the model, also dropping **Source** column because it didn't give us much importance, Finally, the data is ready to be trained by the baseline model.

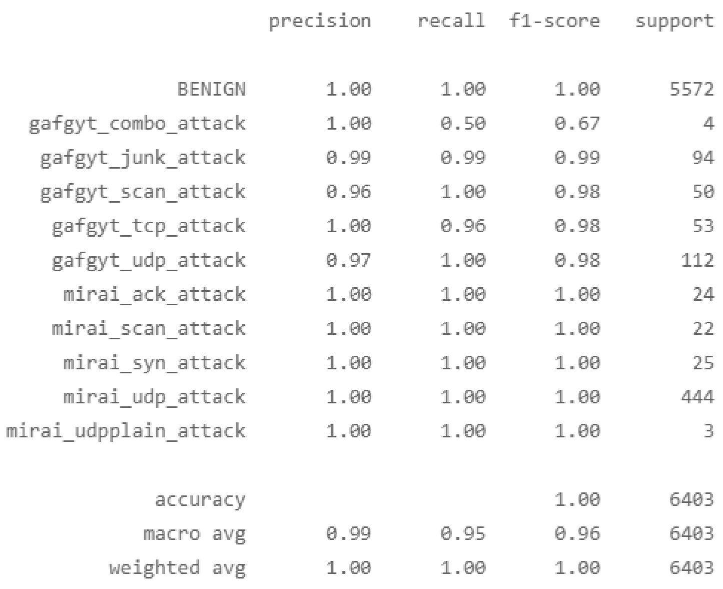
Parameter	**Value**	**Parameter**	**Value**	**Parameter**	**Value**
bootstrap	True	max_samples	None	n_ estimators	None
ccp_alpha	0.0	min_impurity_ decrease	0.0	n_jobs	False
class_weight	None	min_impurity_split	None	oob_score	None
criterion	gini	min_samples_leaf	1	random_	0

				state	
Max_depth	None	min_samples_split	2	verbose	False
Max_features	auto	Random_state	0.0		
max_leaf_nodes	None	min_weight_fraction_leaf	100		

The data was split by using Sklearn function called **train_test_split,** the test data length was 0.33 from the whole dataset, I have used **Random Forest** which was imported from Sklearn.ensemble library as the baseline model with these settings:

Evaluation:

The model has an accuracy that is very close to 1 which means that the model predicted all labels correct almost 100% correct, The model has also 0.96 F1-score which means that the model can differentiate between both classes near 96% correct.



```
                        precision    recall  f1-score   support

                BENIGN       1.00      1.00      1.00      5572
   gafgyt_combo_attack       1.00      0.50      0.67         4
    gafgyt_junk_attack       0.99      0.99      0.99        94
    gafgyt_scan_attack       0.96      1.00      0.98        50
     gafgyt_tcp_attack       1.00      0.96      0.98        53
     gafgyt_udp_attack       0.97      1.00      0.98       112
      mirai_ack_attack       1.00      1.00      1.00        24
     mirai_scan_attack       1.00      1.00      1.00        22
      mirai_syn_attack       1.00      1.00      1.00        25
      mirai_udp_attack       1.00      1.00      1.00       444
 mirai_udpplain_attack       1.00      1.00      1.00         3

              accuracy                           1.00      6403
             macro avg       0.99      0.95      0.96      6403
          weighted avg       1.00      1.00      1.00      6403
```

The data here is imbalanced, hence accuracy won’t give us a very clear performance about the model so using f1-score will be much important cause it focuses on misclassified records.

2. **Adaptive Solution:** Using Kafka data streams, two models will be tested on the new data and one of them would be re-trained on both the new data and the old data to keep track of the changing performance over time, **Random forest** will be used on this experiment for both adaptive and static implementation.
   Kafka Setup: Kafka Consumer was used to start any connection with Kafka Server you must initialize KafkaConsumer session to start getting messeges from the server, and below config parameters controlling it.

parameter	value	Parameters	Value
	‘task_2’	security_protocol	“SASL_PLAINTEXT”
bootstrap_servers	"34.130.121.39:9092"	sasl_mechanism	"PLAIN"
sasl_plain_username	"student"	auto_offset_reset	'earliest'
sasl_plain_password	“Uottawa”	enable_auto_commit	False

Methodology:

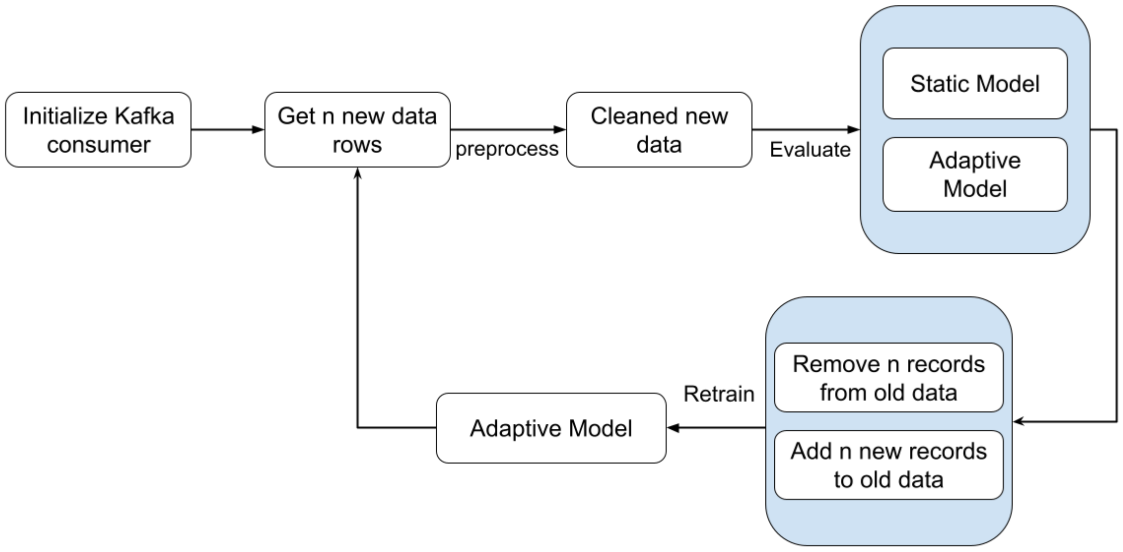


Figure 2-adaptive method

Sliding window settings:

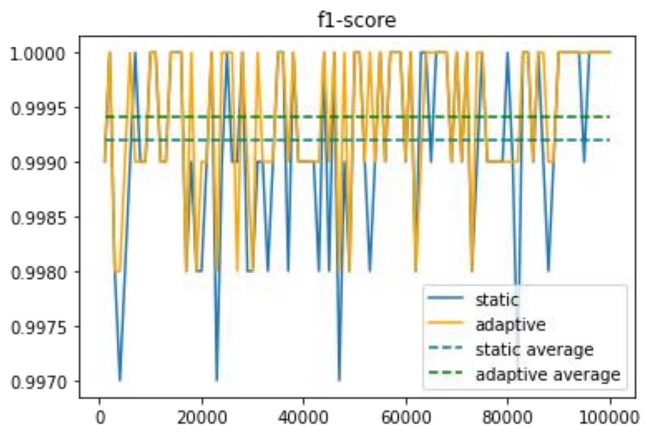
- window size of 1000 packets were used
- 100 iteration ~ 100,000 packets were used
- Original data size = 25000 records

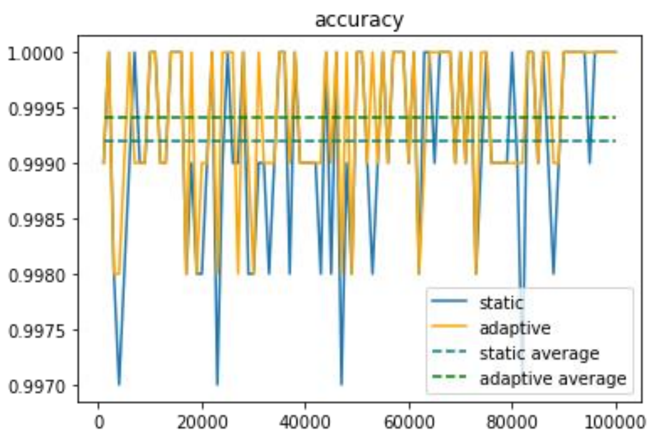
Sliding window steps:

1. every 1000 packets (unseen) were tested using both adaptive and static models (Both Random Forest).
2. After that 1000 records will be dropped from the original data (same size as the window size)
3. Then a combination of original and stream data (stream packets = 1000 + original data = 24000) was used to retrain the adaptive model (Random Forest).

4. Then we will do the same steps until we reach 100,000 packets meaning that we have replaced all the original data then we will stop and to evaluate both static and dynamic models.

Evaluation and results:





Accuracy and F1-scores were used to evaluate both models.

F1-Micro was used as F1-score metric because it aggregates the contributions of all classes to sum the average metric which is preferable when using a multiclass imbalanced problem

We can see that both adaptive and static have nearly the same average scoring on both accuracy and f1-score, but adaptive model has a slight improvement on both accuracy and f1-score

There are a lot of fluctuations on both models, but the adaptive model seems to be more stable than the static one.

The results may be near to each other because the new streaming may be like the original ones so fitting one model of them won't affect much the accuracy or f1-scores.

Knowledge learned:

- How to setup and implement Kafka consumer and deal with json objects using python
- Know more about adaptive learning strategies and used one of them "Sliding window"
- Different between Traditional and Stream Processing
- How to evaluate adaptive learning model