➔ Static and dynamic attacks classification:

- The project is 2 parts:
  - Binary classification between (attack or normal) data.
  - Multi classification between (normal and 10 types of different attacks).
- The (binary and multi) classifications are done with 2 methods:
  1- The static method: means that train the used model only once part of the data, then use it to predict the test data (all data are already exist).
  2- The dynamic method: means that train the model with the already exist data, and use the model to predict another part of the coming data (flow of data from Kafka server). After reading a specific number of the flow of data, replace the old train data with the new once and retrain the model on them. And so on..
  3- Finally, compare between the performances of the model in the both cases.

Hint: I used 4python scripts:

- One script for static and another for dynamic in binary classification
- The same as in multi classification

⇨ Binary classification problem:

1. Algorithms:

I used two models in the static solution. Then I choose the one with best performance and apply it in the streaming solution. Those 2 models are:

1.1. Decision Tree Classifier	1.2. K-Nearest Neighbors classifier
- I used the default parameters of the model, which are: "(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0)" [2] - But, I set the "random_state=0". So, the classifier performs the same in each time. - This classifier will continue expanding its nodes till reach pure leaves or till all leaves have less than min_samples_split s.	- I used the default parameters of the model, which are: "(n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None) "[3] - But, I set "n_jobs=-1" to use all the jobs/processors. - This model by default gives all nodes in each neighbor equal weights. - It is also lazy and non-parametric algorithm. - It classifies each sample based on the majority voting of the K neighbors.

2. Experiments:

2.1. Static solution:

➔ **Preprocessing:**

- Data contains some "null" and "inf" values. I replaced the null with the "mean" and the inf with the "median" of their columns.
- There is a high variance in the data values. So, I scale the data using standard scalar.
- Use "Boruta" model to automatic get the important features of a dataset. In our dataset the most important feature are "66" features.
- Split the data into train and test by 80%, 20%.
- The minority class is ATTACK "imbalanced data". We can apply "somtetomek" as it is a good mechanism to reach balance, but here I leave the data imbalanced.

➔**Metrics used for performance measuring:**

- In the "static solution", to measure model performance, I used:

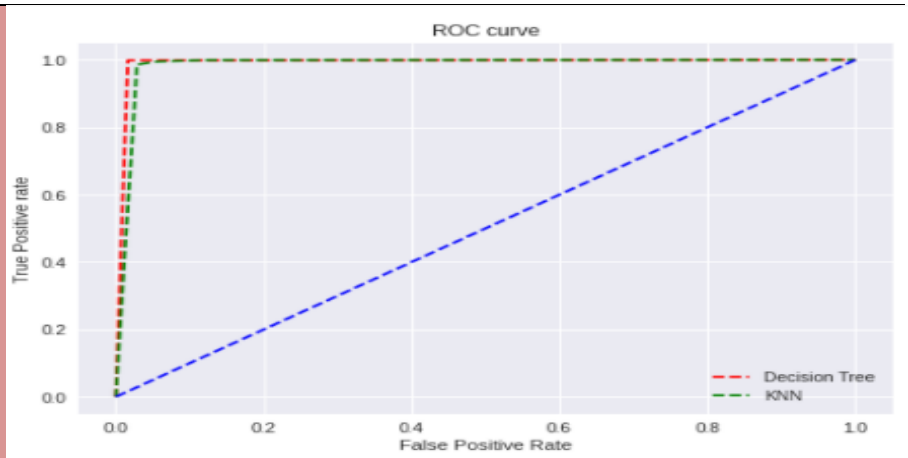
10 folds cross validation	As it good with less biased models like decision tree. Also, it ensures that every sample from dataset appears in the training and testing data. Also, it gives us an expectation about how the model will perform in general. [4]
ROC curve	As it is good to test performance of binary classifier which has imbalanced dataset.
Average macro F1_score	As it is a good metric when the data's classes are imbalanced. Also, it represents the precision and recall in one value.

➔**Models performance results:** From the below table of results it is appear that:

- Decision tree has a higher average cross validation score and higher F1_score.
- Also from ROC curve, decision tree has a slightly better curve than KNN.

➔So, decision tree has a better performance. Then, I will use it in the streaming solution.

	from Decision Tree	from KNN
Average score from cross validation	0. 99727	0.98699
Macro average f1_score	0.99	0.97

ROC curve	

2.2. Dynamic solution:

➔ I used "Forgetting method" as a mechanism to perform adaptive learning. This method works as:

- Evolving: As it retrain the adaptive model in each step/iteration.
- Single classifier: As the adaptive model will not save anything from what it learns before.
- It also makes the model learning on a fixed window of new data.

Forgetting pros	Forgetting cons
- It makes the model aware of all new data once it arrives.	- The model forgets everything it was learnt before. - Sometimes the newly coming data is not as valuable as the old data which the model trained with. And that would make model performance decrease through time.

➔Iterate 20 times to reach 100,000 stream record. And in each iteration we have the following:

Read	Preprocessing_1	Test	Remove	Add	Preprocessing_2	Retrain

Read 5000 stream record in a data frame.	I used the same steps as in static solution. But, I used "SelectKBest" which enables me to decide the same number of important features as in the called model. So, it will work correctly.	Test the performance of the 2 models (static & adaptive) on streaming data and save their performance	Remove 5000 record from the head of the static data frame.	Add the new 5000 stream record to the tail of the static data frame.	Apply the same as in preprocessing_1 on the whole data from "Add" step.	Retrain the adaptive model with the data from preprocessing_2

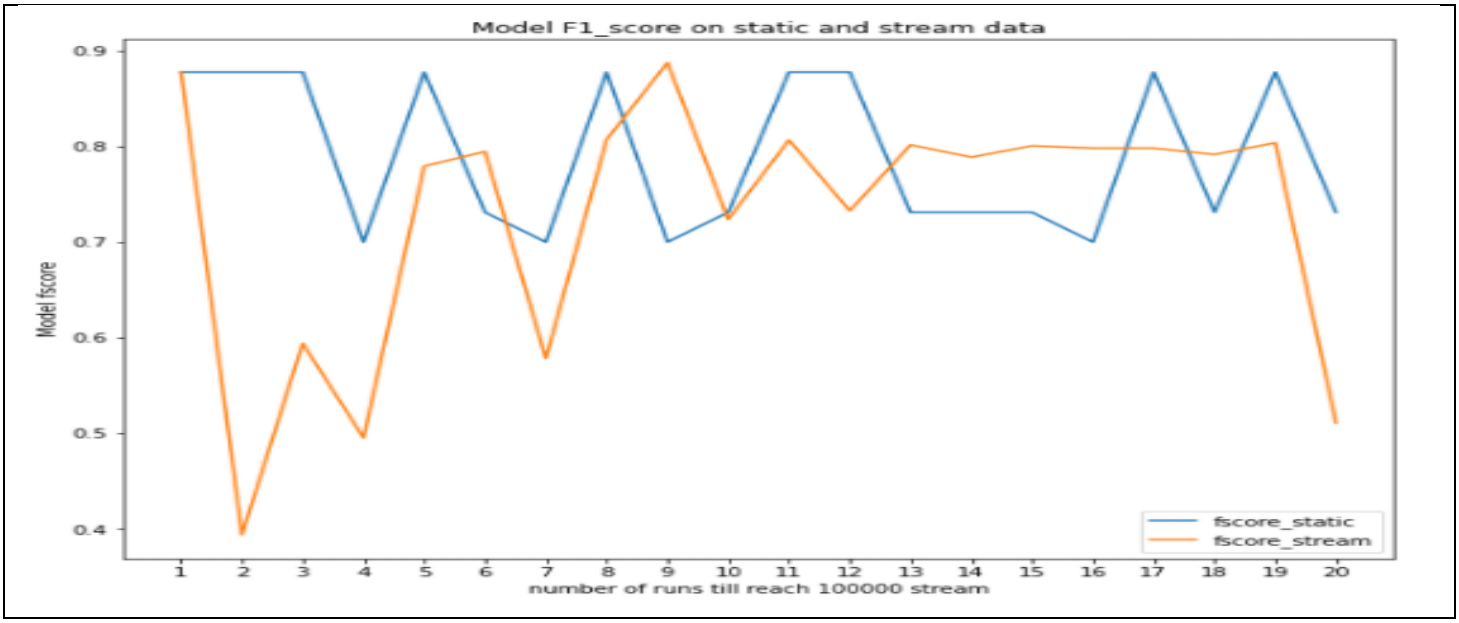
➔**Metrics used for performance measuring:**

In the "dynamic solution", to measure model performance, I used:

Average macro F1_score	Because it represents precision and recall in one number. Also, it is suitable metric in case of imbalanced data.

➔**Models performance results:** From the below figure, it is appearing that:

- Based on the distribution of classes came from Kafka server, the performance of the static and adaptive models changes.
- Based on values of F1_score for static model in the graph, it has a better performance in 10 from the all 20 iterations than adaptive model.
- And the adaptive model performs better in 9 iterations than static one.
- In the first iteration they both have the same F1_score value. And that is normal as in the beginning the both models learns the same.
- The two models have relatively good f1_score from iteration 7 to iteration 19. The score in range(>0.7 to 0.9)
- As I used forgetting method, then the adaptive model is retrained in each step and not when change happened. So, it is normal that the adaptive model to have a lower performance in some iteration.
- Those 20 iterations took about 27.6 minutes to finish.



⇨ Multi classification problem:

1) Algorithms:

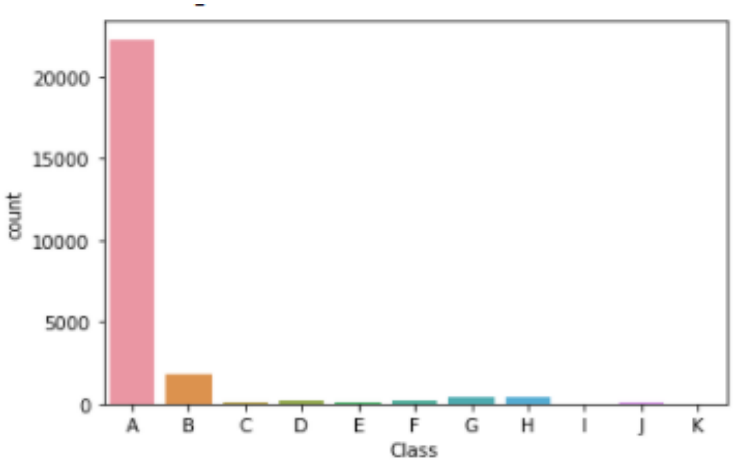
I used the same two models as in "Binary class problem, static solution".

2) Experiments:

2.1. Static solution:

➔ Preprocessing:

- The data has no missing or" inf "values.
- "Source" column contains string values. I encoded them by using "get_dummies" method.
- Scale and select features just as in "Binary class problem". But, the selected features here are "95" features.
- There is 11different class in the data. One is "Benign" and the rest 10 are attack types. And I've interest in attack classes, but the attack classes are very small compared to benign class "the data is unbalanced".



A, B, C, D, E, F, G, H, I, J, K ➔ represent the 11 classes in the data.

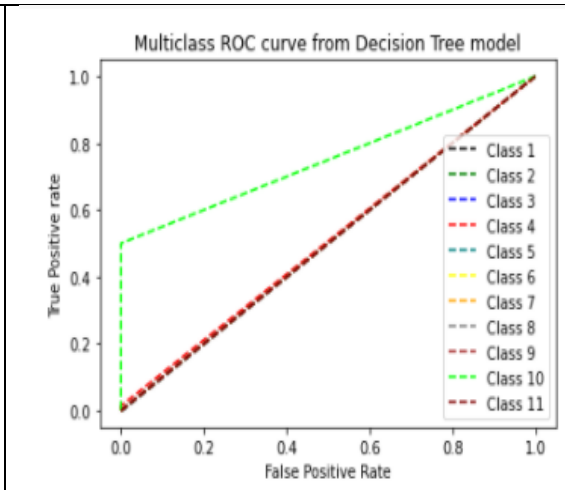
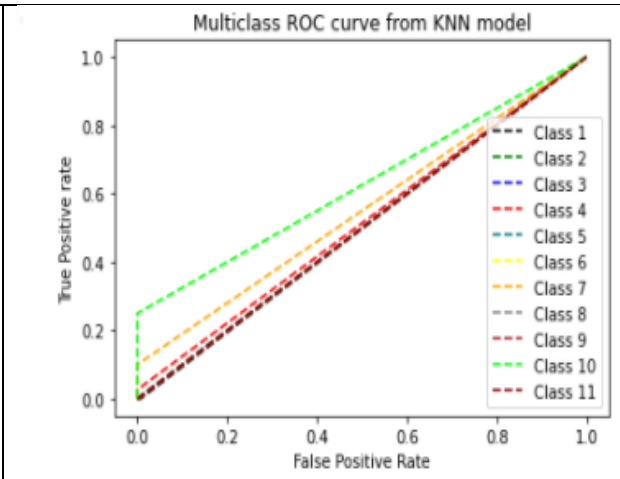
➔**Metrics used for performance measuring:**

- I used the same metrics as in “static solution in Binary class problem”.

➔**Models performance results:** From the below table of results it is appear that:

- Decision tree has the same average cross from cross validation as KNN
- But decision Tree has ahigherF1_score than KNN.
- Although, in the ROC of KNN has better performance for some classes. But, in ROC of Decision Tree class 2 has the best curve of the whole classes.
- Therefore, decision tree has a better performance. So, Use it in the streaming solution.

	from Decision Tree	from KNN
Average score (cross validation)	0. 99917	0.99917
Macro average f1_score	0.94	0.87

ROC curve		

I. Dynamic solution:

➔ I used the same mechanism as in "Binary classification, dynamic solution" which is "Forgetting method" mechanism to perform adaptive learning.

➔I used the same streaming data size (5000 record per iteration).

➔Metrics used for performance measuring:

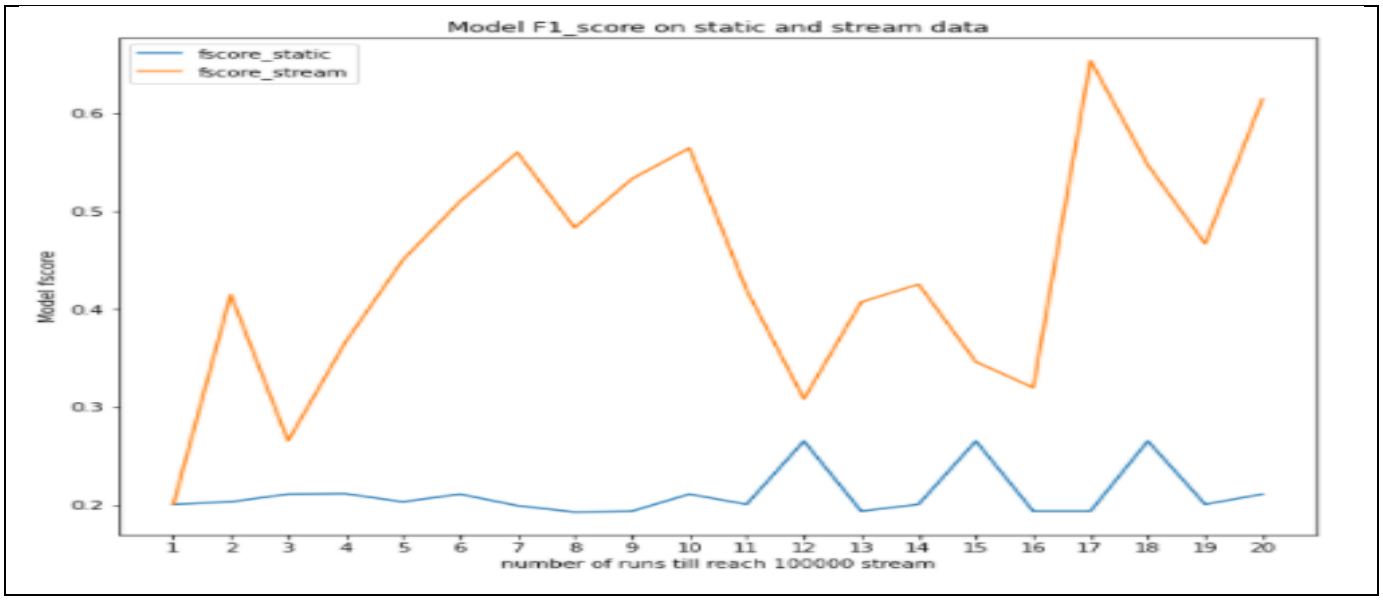
I used average macro F1_score as in "Binary classification, dynamic solution"

➔Models performance results:

- Based on the data coming from Kafka server, classes which have the highest distribution are (BENIGN and mirai_udp_attack) classes. And (mirai_udpplain_attack) has the lowest distribution.

- From the below figure, it is appearing that:

  - The adaptive model has a better performance than static one in the all 20 iteration as it has a higher F1_score.
  - That is because the static data contains very large range of benign samples. And when we remove 5000 record from it, those records have a big portion of benign class. And the new 5000 stream data have a relatively good number of attacks samples. So, the attacks samples increase. As a result the adaptive models learn much better and can classify attacks more efficiently.
  - Those 20 iterations took about 15.16 minutes to finish.



References:

1- Navlani, A., 2018. [online] Available at: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

2- scikit-learn. 2021. *sklearn.tree.DecisionTreeClassifier*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

3- scikit-learn. 2021. *sklearn.neighbors.KNeighborsClassifier*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

4- Brownlee, J., 2018. *A Gentle Introduction to k-fold Cross-Validation*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/k-fold-cross-validation/