ELG7186[EG] AI FOR CYBER SECURITY [LEC] 20229

Assignment 3 Kafka

NAME

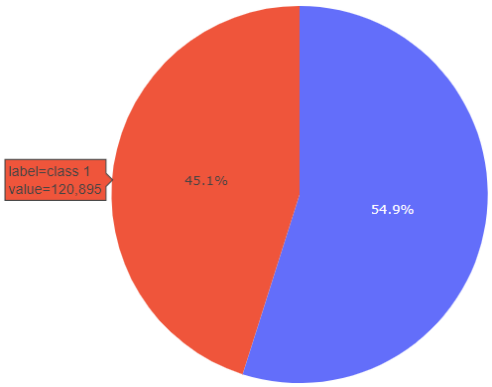
Hosam Mahmoud Ibrahim Mahmoud

uOttawa ID

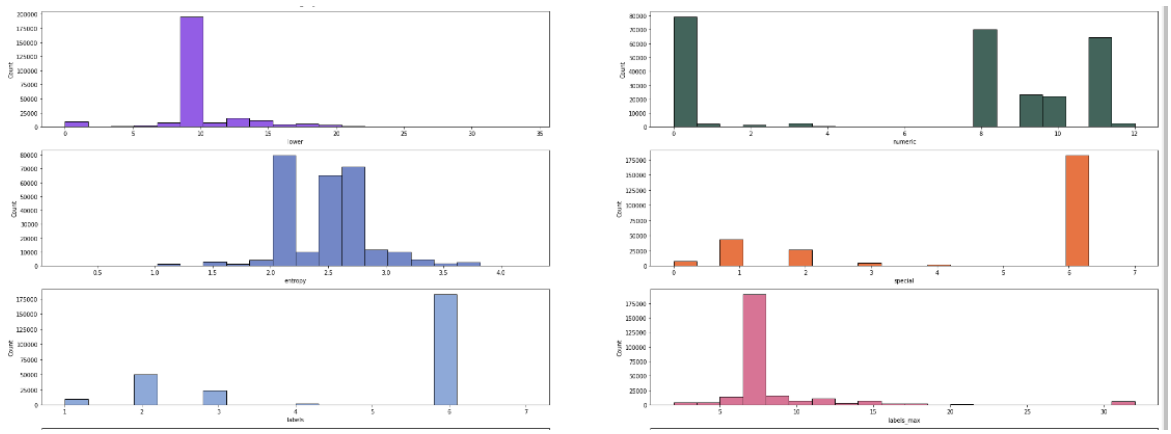
300327269

1. Task 1 Static model

- Please consider taking a look at the **Table of contents**, if these two notebooks would be opening in Google Colab.
- I think that the data is a little bit **imbalanced** because the total records of class 1 are close to class 0.



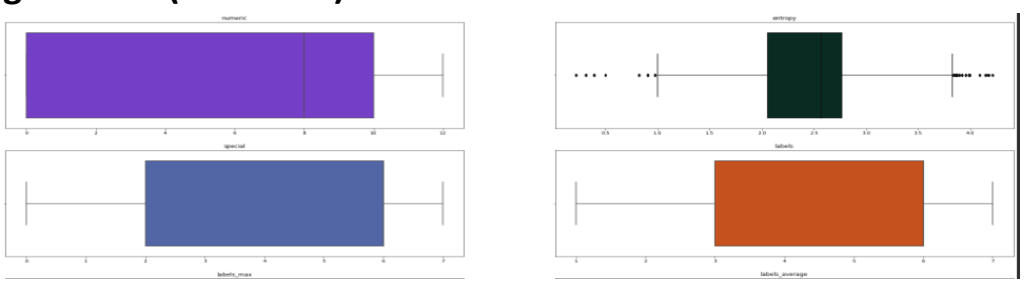
- **Statistical analysis of data**
  - **Data Distribution**



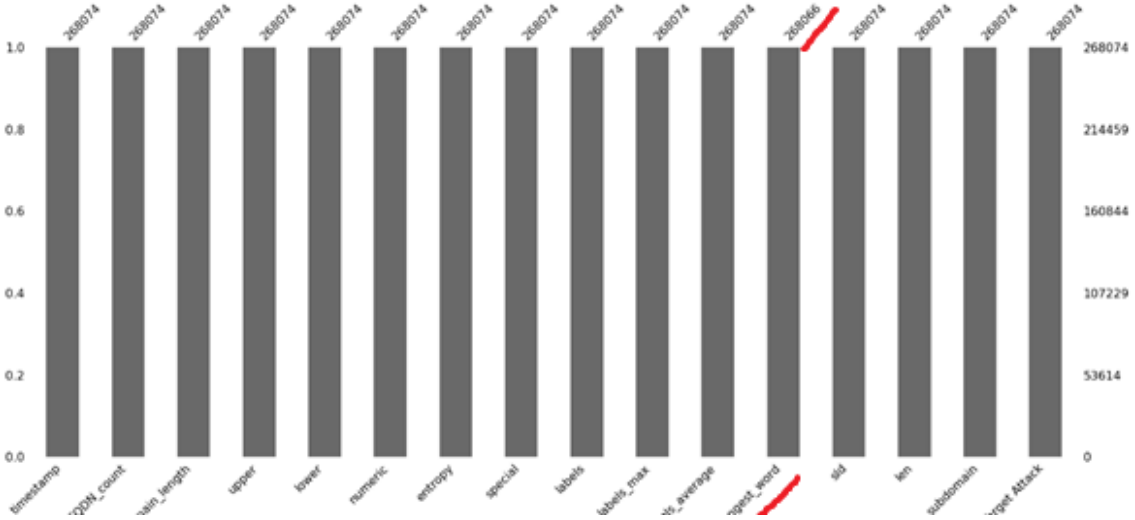
  - **Mean, min, max, count, std, and the 3 quartiles.**

	FQDN_count	subdomain_length	upper	lower	numeric	entropy	special	labels
count	268074.000000	268074.000000	268074.000000	268074.000000	268074.000000	268074.000000	268074.000000	268074.000000
mean	22.286596	6.059021	0.845420	10.410014	6.497586	2.485735	4.533577	4.788823
std	6.001205	3.899505	4.941929	3.207725	4.499866	0.407709	2.187683	1.803256
min	2.000000	0.000000	0.000000	0.000000	0.000000	0.219195	0.000000	1.000000
25%	18.000000	3.000000	0.000000	10.000000	0.000000	2.054029	2.000000	3.000000
50%	24.000000	7.000000	0.000000	10.000000	8.000000	2.570417	6.000000	6.000000
75%	27.000000	10.000000	0.000000	10.000000	10.000000	2.767195	6.000000	6.000000
max	36.000000	23.000000	32.000000	34.000000	12.000000	4.216847	7.000000	7.000000

  - **Detecting outliers (Box Plots)**



- **Data Cleansing and Feature creation**
  - I've dropped the **missing values** which was 8 records

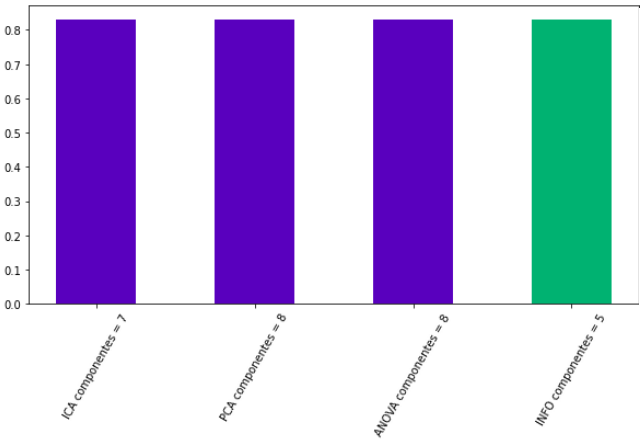


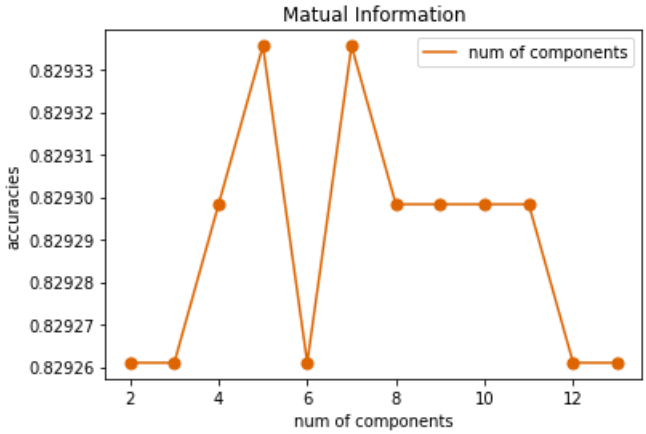
  - since I will not use time series model, I've dropped the **timestamp column**
  - for string columns, **I've used the md5 hashing technique** and converted the hex values to float numbers.

```
print(data['longest_word_hash'][0])
print(data['sld_hash'][0])
```

```
2.660036914772862e+38
1.1781833566796777e+38
```

- **Feature Filtering with more than 2 methods**
  - I've tried a baseline model (LGBMClassifier) **as an experiment** to be able to plot the accuracies over the number of components for each method, it's a very fast model.
  - I've used 4 methods, **Fast ICA**, **PCA**, **ANOVA,** and **Mutual information**, and the best feature selection was from **Mutual information** with **0.8293356 Accuracy (5 features).**



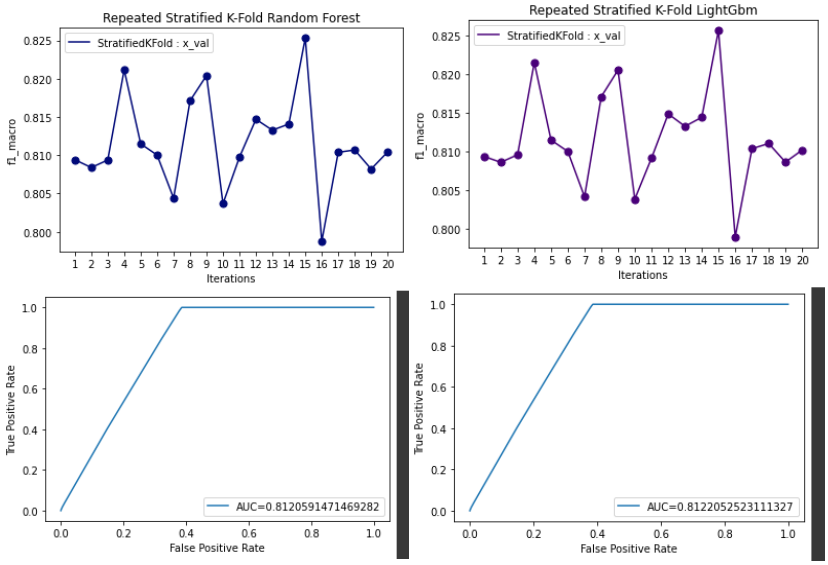


- **Data Splitting and justification**
  - I've splatted the dataset into **Train, Test, and Validation.**
  - I've used the **validation** set to plot the **T-SNE visualization.**

- I've used my **train** set to do **hyperparameter tuning**, and **train my models.**
- And the **test** set to **evaluate** my models (by using cross-validation), and see the final results (confusion matrix, and classification report).
- After the data splitting I've applied normalization technique on the data which is MinMaxScaler.

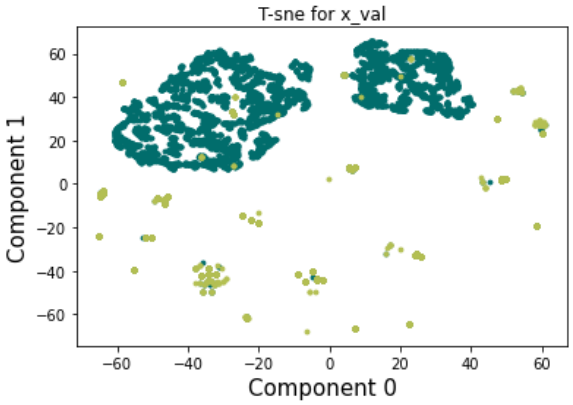
```
normalizer = MinMaxScaler()
x_train = normalizer.fit_transform(x_train)
x_val = normalizer.transform(x_val)
x_test = normalizer.transform(x_test)
```

- **performance metric**
  - I've chosen the **f1_macro metric** because it's a good metric when it comes to **machine learning problems in the security field** and also the data is **a little bit imbalanced.**
  - Also, I've plotted the **ROC Curve**, which gave a good indication about the True positive rate and the false positive rate.
- **Hyperparameter tuning**
  - I've used a **grid search** with **scoring = f1_macro**, and I've used my train set to tune the parameters.
- **My two models**
  - First, I've tried **Random Forest Classifier** which is a bagging algorithm that builds decision trees on data samples and takes their majority vote for classification.
  - Second, I've tried **LightGBM Classifier** which is boosting algorithm and it's a lot faster than Xgboost.
  - For model evaluation I've used **Repeated Stratified KFold** with number of splits = 10, number of repeats = 2, and **scoring = f1_macro.**

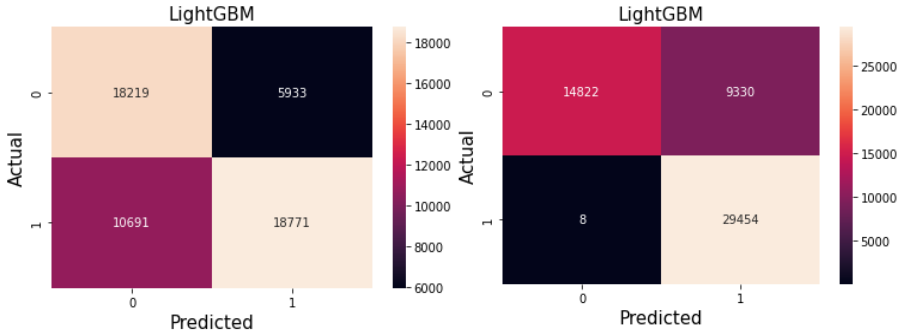


  - They kind of was giving the same results, but the **lightGbm** was a lot faster so picked it as my champion model.

- **Part 1 conclusion**
  - When I plotted **T-SNE** for the validation set, I found that that data is very complicated and it has a lot of overlapped data points.



  - Also, I've tried **ThreshHolding** with **LightGbm,** but it seems that the model can not know the real class for a lot of the data points even when applied **ThreshHolding,** which means that a lot of data (more than 9000 data points) is similar and very overlapped.
  - The one on the left is with **ThreshHolding**, and the right one is the normal one.



	precision	recall	f1-score	support
0	0.63	0.75	0.69	24152
1	0.76	0.64	0.69	29462
accuracy			0.69	53614
macro avg	0.70	0.70	0.69	53614
weighted avg	0.70	0.69	0.69	53614

	precision	recall	f1-score	support
0	1.00	0.61	0.76	24152
1	0.76	1.00	0.86	29462
accuracy			0.83	53614
macro avg	0.88	0.81	0.81	53614
weighted avg	0.87	0.83	0.82	53614

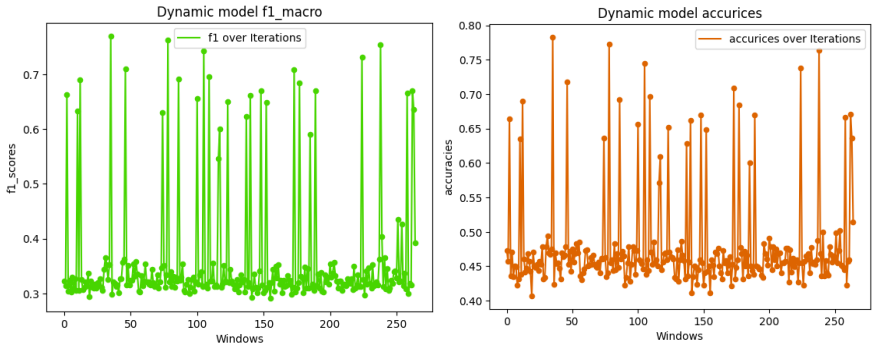
  - I Think advanced techniques like Deep Learning or GAN will give much better results.

2. Task 2 Dynamic model

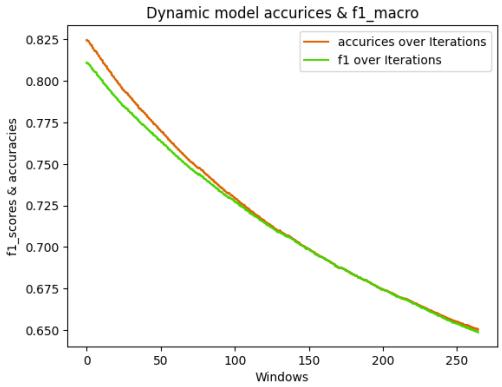
- First, I've **preprocessed the Static data** as it is in the previous notebook.
- After that I loaded my champion model and created two models: **dynamic and static models.**
- After that I created some functions to be able to do these things:
  - Get 1000 new records on each window from Kafka dataset.
  - Covert the bytes to dataframe records to be able to work the streamed data.

- Preprocessing these new 1000 records (the same preprocessing that I've applied on the static dataset)

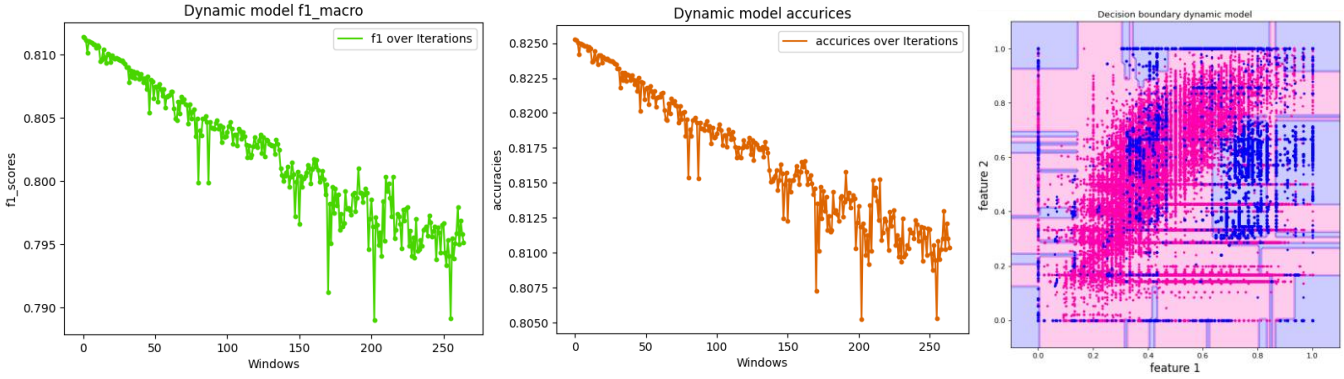
- I've also chosen the **f1_macro** as my metric because I want to see if my recall and precision change over windows when retraining the model or not.
- **After that I've done 4 main things:**
  - **1- Dynamic model predicts 1000 records only for each window (without retraining).**



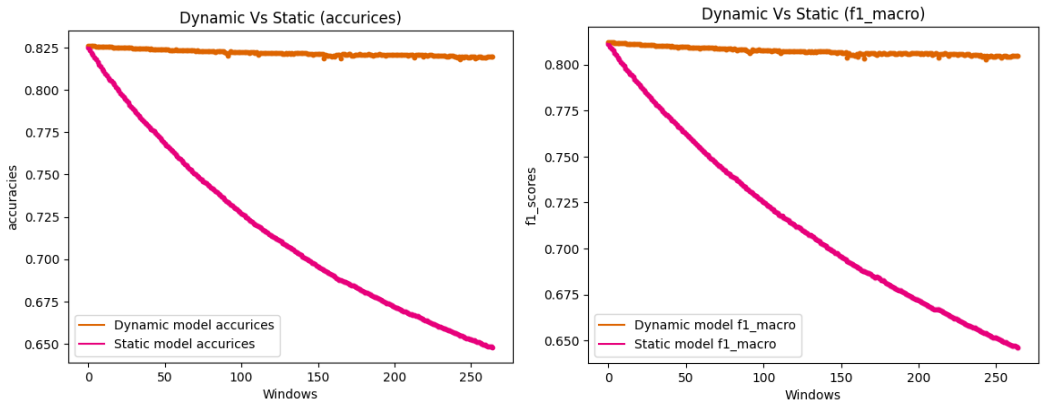
  - **2- Dynamic model predicts 1000 records appended to the static dataset for each window (without retraining).**



  - **3- Retraining Dynamic model on the two first features of the static and Kafka dataset (to be able to plot Decision boundary) (with retraining).**
  - Note* I've provided a GIF file with my submissions to be able to see the animated decision boundary over the windows.



- **4- Retraining the Dynamic model on the original features from the preprocessed dataset of static and Kafka (with retraining).**



- **Part 2 conclusion**
  - for the first part, the graphs tell us that every 1000 records are different from others and every 1000 records have its own distribution, **which explains the hesitating in the accuracies.**
  - For the second part we have found that the loaded model that was trained on approximately (200,000 records) will gradually give bad accuracies over windows because the model is trying to predict a large amount of data that is not trained on.
  - For the third part I've used the first two features (that have the highest amount of information) to Retrain my dynamic model to be able to see the decision boundary results, and I've found from the decision boundary plots, that when the data increased the data points got overlapped more and more.
  - For the fourth part, we have found that the static model gives bad results over windows because it's not retrained on the new data that appended unlike the dynamic model.