Continuous Evaluation with Kafka

Assignment -3

1st Ameya Chawla*

*University of Ottawa, Ottawa, Canada, achaw017@uottawa.ca, 300281112

1. Data Analysis

Before starting with an analysis of the data, null values(8 instances were dropped) were removed as a prevention for future errors. Refer to cells [5-7] in the Static Model.ipynb.

1.1. Data Imbalance

Dataset had an imbalance of 27000 instances in which the positive class was 55% of the whole dataset. RandomOverSampler was used to randomly create copies of already existing instances of minority class as it is a sensitive issue and using synthetic sampling will create new data which can even lead to false predictions when model is tested on a real-world dataset. Refer to cell [9-11] in the Static Model.ipynb. Data was balanced so that model gives equal priority to each class.

1.2. Statistical Analysis of Data

1.2.1 Skewness Analysis

Statistical Analysis of the data was started by analyzing numerical features by creating their histograms with kernel density plots using seaborn's histogram plot. The Data was highly skewed for many features and used different techniques like Box-Cox power transformation, and logarithmic transformation to handle skewness but the data still remained skewed as shown in Figure [1-3]. Refer to line 13-15 in Static Model.ipynb

1.2.2 Variance Analysis

The variance was calculated using the inbuilt function of pandas and the graph was plotted. Features with the lowest variance were noted for future references as it was also used in feature selection. Subdomain, labels had lowest variance. Refer to cell [16-18] in the Static Model.ipynb

1.2.3 Scatter Plot for features and target attack relation

The scatter plots were created using seaborn's scatter plot where these graphs were not full of observation as they will always produce two regions as y will be always 0 and 1. Refer to cell [17] in the Static Model.ipynb

1.2.4 Scatter Plot of features vs features

All the numerical features were plotted against every other feature for better analysis and some observations were made. The upper feature had the most values 0. FQDN, lower and len are highly correlated features. Refer to cell [18] in the Static Model.ipynb

1.2.5 Categorical Feature Analysis

Categorical features were analysed using a count plot and when plots were created due to high cardinality in the features it was impossible to create plots with unique values. Refer to cell [20] in the Static Model.ipynb

1.2.6 Line Plot for features vs Target variable

Line plots were created for features vs target attack as scatter plots didn't provide much information. Line plots did not show much relation pattern between the features except in the case of the subdomain . Cell[19} in Static Model

2. Feature Engineering and Data Cleaning

2.1 Data Cleaning

Data was already cleaned in the first step before the start of the analysis and all the null value instances were removed.

2.2 Feature Engineering

All the categorical features (timestamp, longest_word, sld) were converted to hash values using Python's inbuilt hash function. The function was not working on the whole column so it was applied instance by instance wise for each of the above-mentioned columns. Hashing converted all those string values into numeric integers.

2.3 Feature Analysis of categorical variables created

All the same, plots were created for numerical features like histogram plots, scatter, and line plots were made, and no notable relation was observed in these graphs. Refer to cells[21-29] in Static Model. ipynb

3. Feature Selection

5 methods were used for the feature selection process. Refer to cells [30 -43] in Static Model.ipynb

3.1 Correlation

The correlation was calculated for each feature using pandas and features with t   lowest correlation were considered for removal.

3.2 Variance Analysis

Features with a variance of less than 0.2 will be considered for removal.

3.3 Skewness Analysis

Features with skewness greater than 1 and less than -1 were considered for removal.

3.4 Random Forest Classifier Selector

Select from the model was used where random forest classifier selects best performing features.

3.5 Recursive Feature Elimination

A recursive feature selector with Logistic Regression as a base estimator was used to select best-performing features.

3.6 Results

The below-mentioned features were dropped on basis of above 5 methods.

'timestamp', 'upper', 'labels_max', 'labels_average', 'entropy', 'subdomain'

4. Model Training

4.1 Data Splitting

The data was split into train: test with the ratio of 67:33 to increase the size of the test for better evaluation of the models. Refer to cell [45] in Static Model.ipynb

4.2 Data Normalization

Data was normalized using MinMaxScaler. Refer to cell [55] in Static Model.ipynb

4.3 Performance Metric

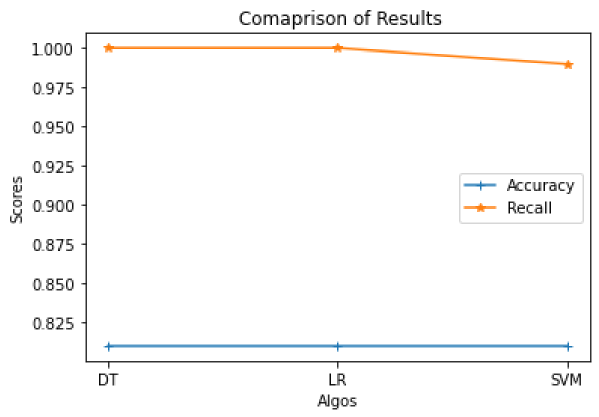
Accuracy and recall for the positive class were taken as appropriate metric as we want our models to classify as many positive instances correctly to measure it recall of t   positive class is best to measure and accuracy used to check on the other class as the recall was close to 1.0 in most cases so less accuracy means only negative class was mislabeled.

4.4 Training Procedure

Hyperparameter tuning was done using K-Fold cross-validation to find the best estimator.

4.5 Training Results and Threshold

Decision Tree, Logistic Regression, and Support Vector Machine were trained and results were plotted.



Logistic Regression and Decision Tree had the same recall and accuracy, but the decision tree was selected as decision tree as it had a higher precision score in comparison to logistic regression. The threshold was set to 5% lower values than the accuracies and recall obtained by these models.

The threshold for Accuracy is 0.7695

The Threshold for Recall is 0.95

5. Dynamic Model

5.1 Window Creation

One counter variable named ‘c’ was used for windows creation as when input came from the stream, then it was split and preprocessed and then inserted at the start of the list t_x,t_y. When the length of this storage was a multiple of 1000 if the condition was used to test the first 1000 elements of the list and data was always inserted at the start of the list. The value c was updated after every 1000 instances were tested. Cell 2 in Dynamic Model.ipynb

5.2 Retraining Procedure

When the model’s accuracy and recall both were below the above-mentioned threshold the model was retrained on the whole stream of data received till that window. The whole dataset was stored in t_x,t_y and then these were preprocessed and then the model was refitted and all the previous knowledge of the model was removed. The dynamic model was only retrained twice for windows 3rd and 82nd. Cell 2 in Dynamic Model .ipynb

5.3 Performance Metric Justification

The recall and accuracy were selected as the aim from the start were to create models which can classify as many as positive instances correctly to detect all the target attacks and the best metric to observe this scenario is the recall of the positive class as it tells how many positives correctly classified out of total positives. Accuracy was used as the second metric to keep an eye on other class when the recall is close to 1 so the most reduction in accuracy will be due to misclassified instances of the negative class.

5.4 Plotting The Results

The Results were plotted for the models and the following graphs were obtained shown in the appendix. Cell 3

5.5 Result Analysis

Overall dynamic model got higher accuracy and recall score in comparison to the static model. There was less variance in t   scores of the dynamic model in comparison with static model.The dynamic model was highly adaptable as it was trained on less dataset as it was trained only on 82000 samples as it was last trained on 82000 samples in 82th window. We are getting a high recall score(close to 1) and less accuracy which means model is predicting almost all positive instances correctly and some negative instances incorrectly as we get 0.8 accuracy with recall score of 1 which means the 20% remaining samples are from the negative class. The dynamic model is performing better than static as it was only trained on 82000 vs static trained on 3 times more big dataset. Dynamic model is a little biased towards positive class as the recall score is close to 1.0 and accuracy shows it also has high false positives point 4. But the recall score is good which makes it easy for the model to detect all the attacks. Cell 5

5.6 Advantages/Limitations and Knowledge

1. The dynamic model is highly adaptable in comparison to the static model which is an advantage.

2. The dynamic model performs better than static model which is an advantage.

3. The dynamic model had less variance in results in comparison to static model which is and advantage.

4. The dynamic model is biased towards positive class which static model was too which is a disadvantage.

5. Less computation is required to train dynamic model in comparison to static model which is an advantage.

6. Dynamic model is able to achieve a recall of 1.0 which is good sign it is able to classify all positive instances correctly.

Appendix

Fig.1 Dynamic Model Recall Score

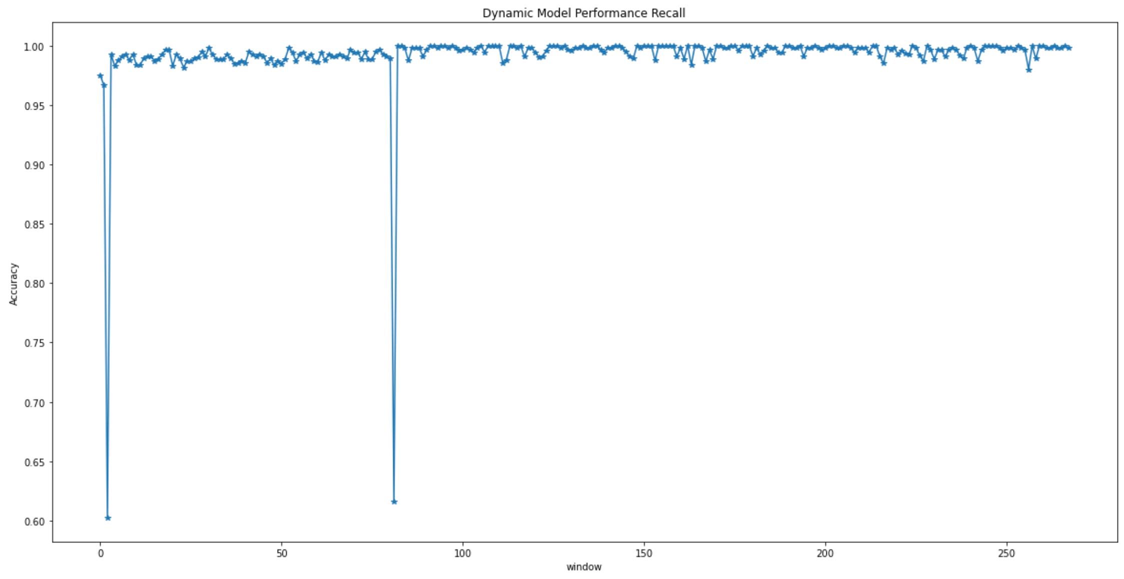


Fig.2 Static Model Recall Score

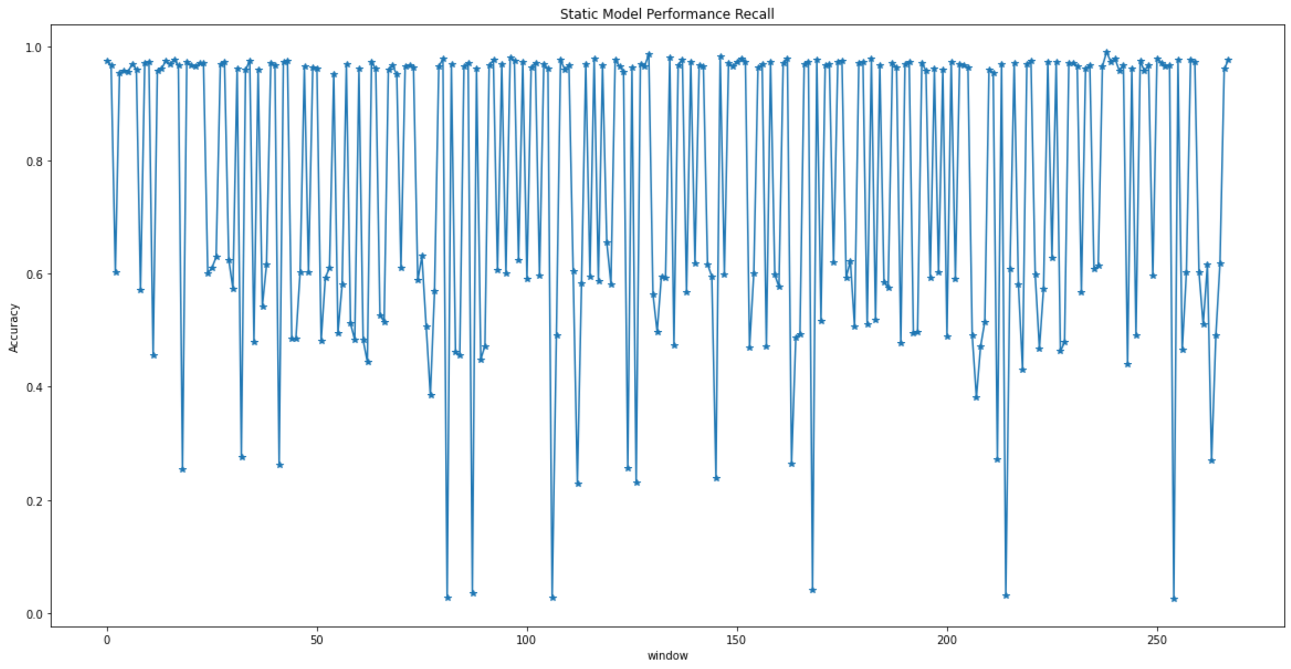


Fig.3 Dynamic Model Accuracy Score

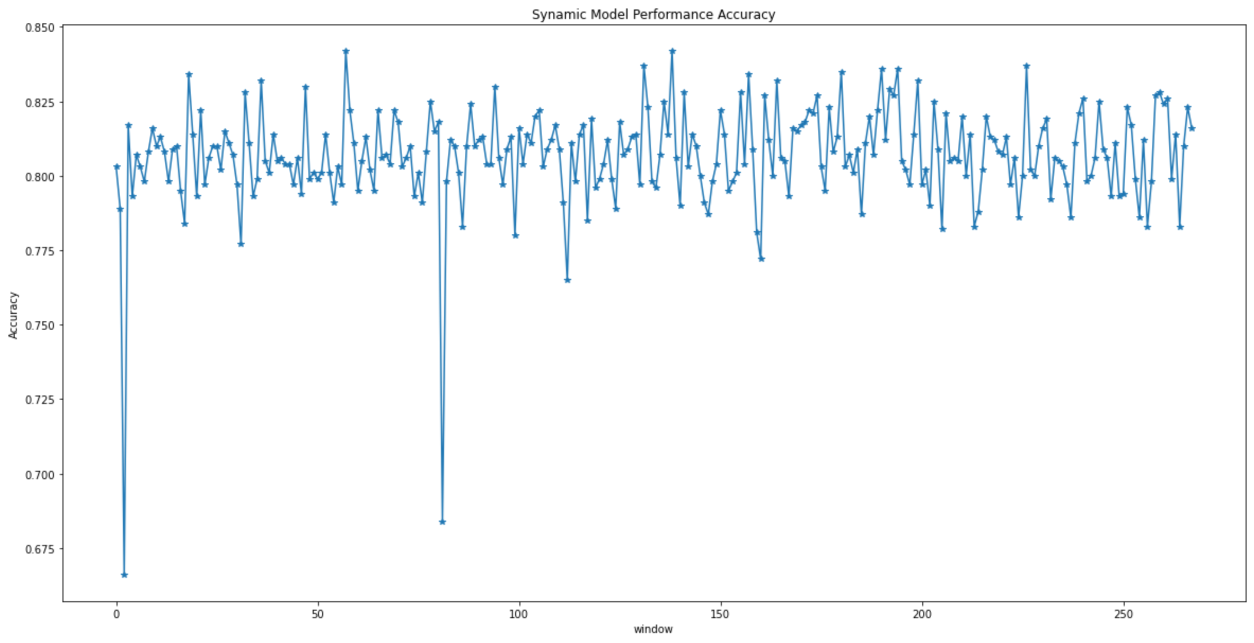
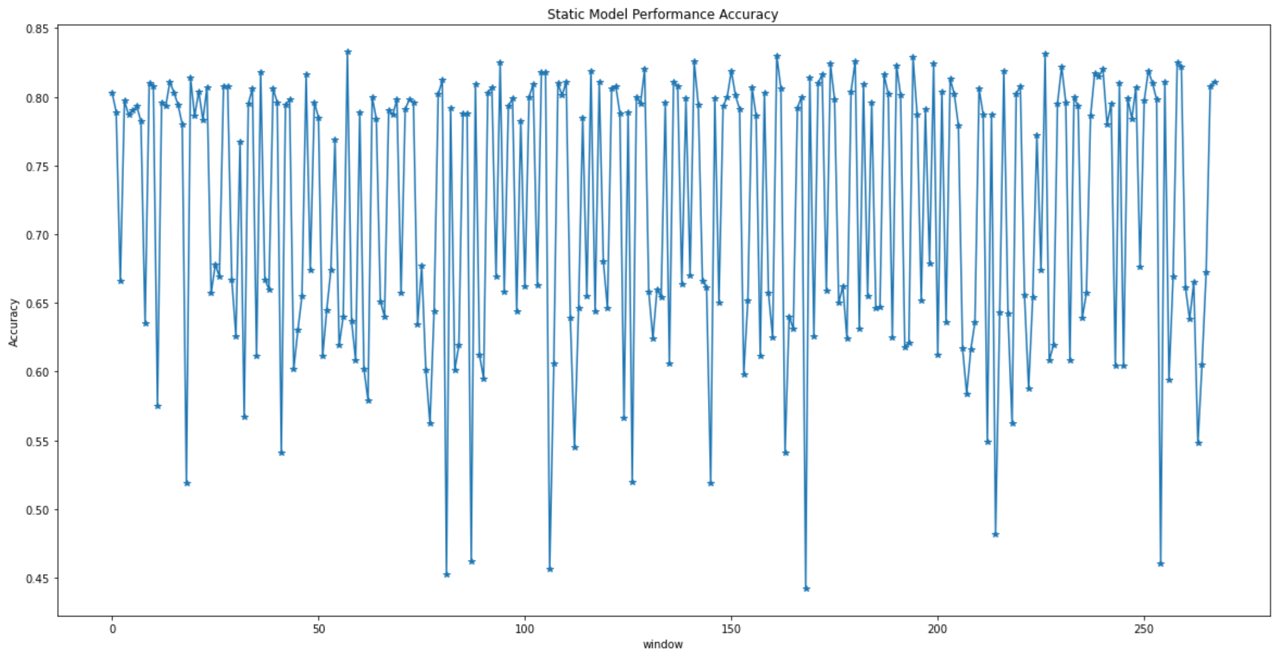


Fig.4 Static Model Accuracy Score


Static Model Performance Accuracy
Accuracy
window