Assignment 3

Name: Anas Elbattra

Student Id: 300389368

1 - Static

Data Cleansing and Feature creation

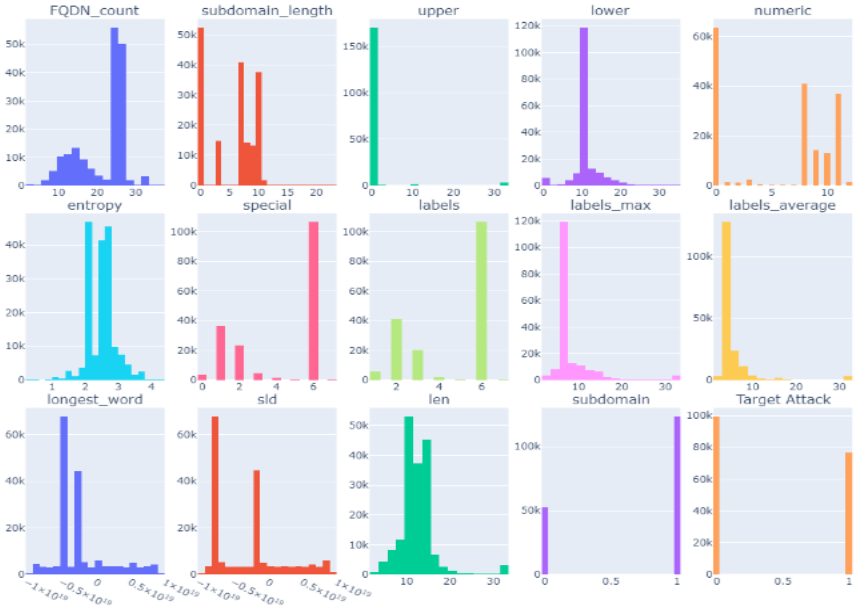
In the initial stage of data cleaning, duplicate entries were identified and eliminated to ensure the uniqueness of each data record. Subsequently, rows containing null values were dropped to uphold data integrity.

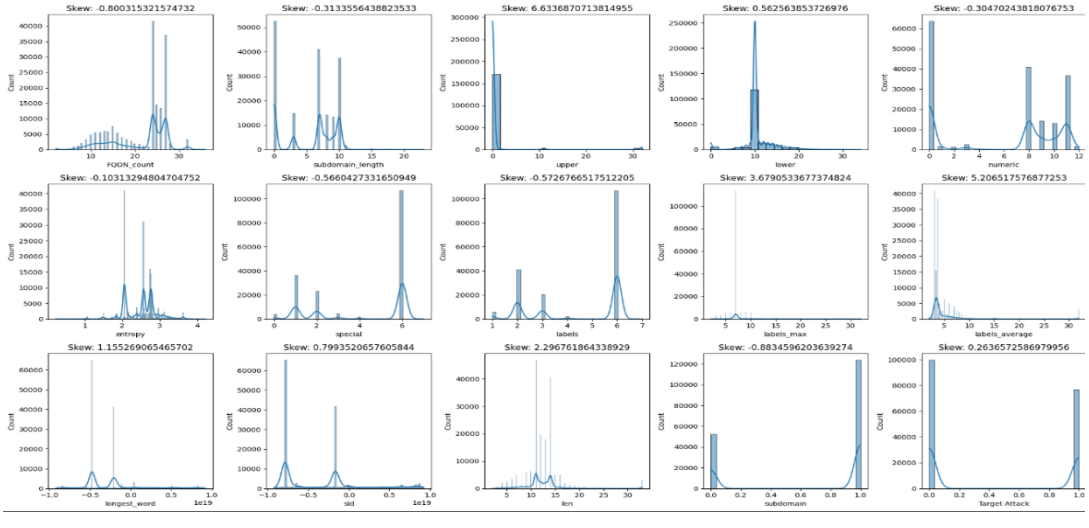
Upon reviewing the dataset, it was observed that the 'longest_word' and 'sld' columns comprised data of different types. To manage this, the hash function was applied to these columns. This action served multiple purposes: anonymizing data, standardizing the representation of variable-length input, improving the efficiency of data comparisons, and creating new features for potential use in machine learning models.

In addition, the 'timestamp' column was dropped from the dataset. This action was taken as the timestamp information was not relevant to our intended analysis. Retaining this column could have introduced problems such as high cardinality and overfitting in future modeling processes.

Statistical analysis of data

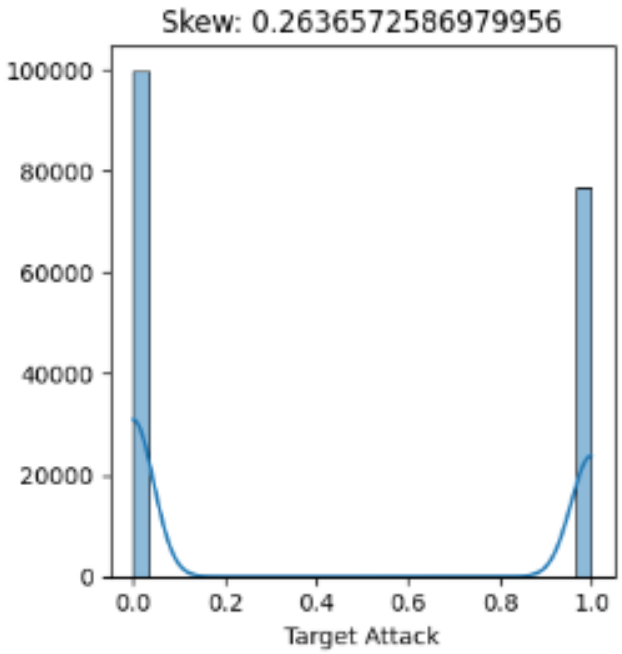
Histograms provide a visual aid for understanding data distribution, identifying patterns, outliers, and anomalies. Skewness quantifies the asymmetry of data distribution, offering insights into potential biases and the deviation direction from the normal distribution.

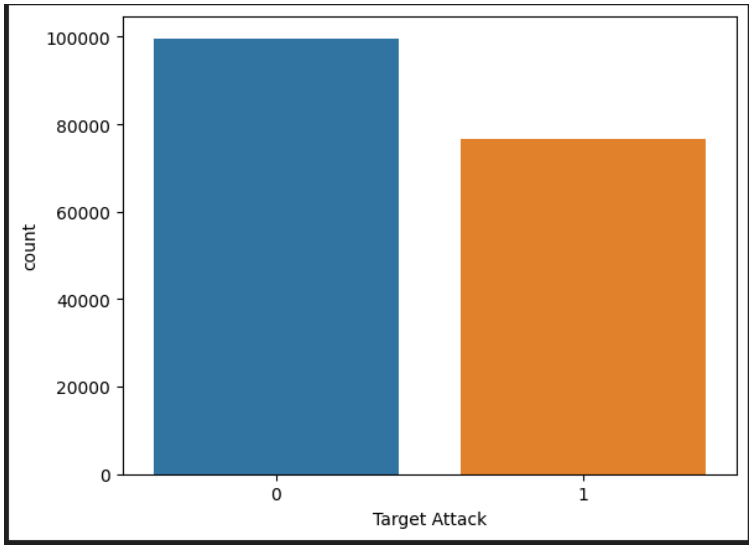




Validate data imbalanced with justifications.

The dataset can be considered balanced because the proportion of instances in the two classes, 0 and 1, are not extremely skewed, with class 1 making up over 43% of the total instances, which is a substantial representation.

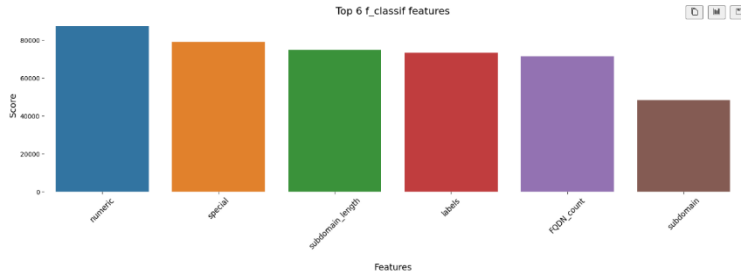




Two methods for Feature Filtering

Our feature selection process leverages two distinct methods: f_classif and mutual_info_classif. We prefer f_classif because it statistically tests the individual effect of each feature, thus eliminating irrelevant ones and reducing the risk of overfitting. On the other hand, mutual_info_classif is used as it measures the dependency between the features and the target, capturing nonlinear relationships, which the former may miss. We limit the selection to the top six features, a decision informed by achieving an F1 score that matches or surpasses our baseline model, thereby ensuring efficiency without compromising the model's performance.

Top feature mutual info



Top feature f_classif

Data Splitting and justification

Data splitting in our project involves dividing the dataset into a 70% training set and a 30% testing set. This division allows the model to learn from most of the data (70%), providing a rich foundation for understanding the underlying patterns. The remaining 30% is reserved for testing, offering a substantial proportion for an unbiased evaluation of the model's performance on unseen data.

Choose and justify the correct performance metric.

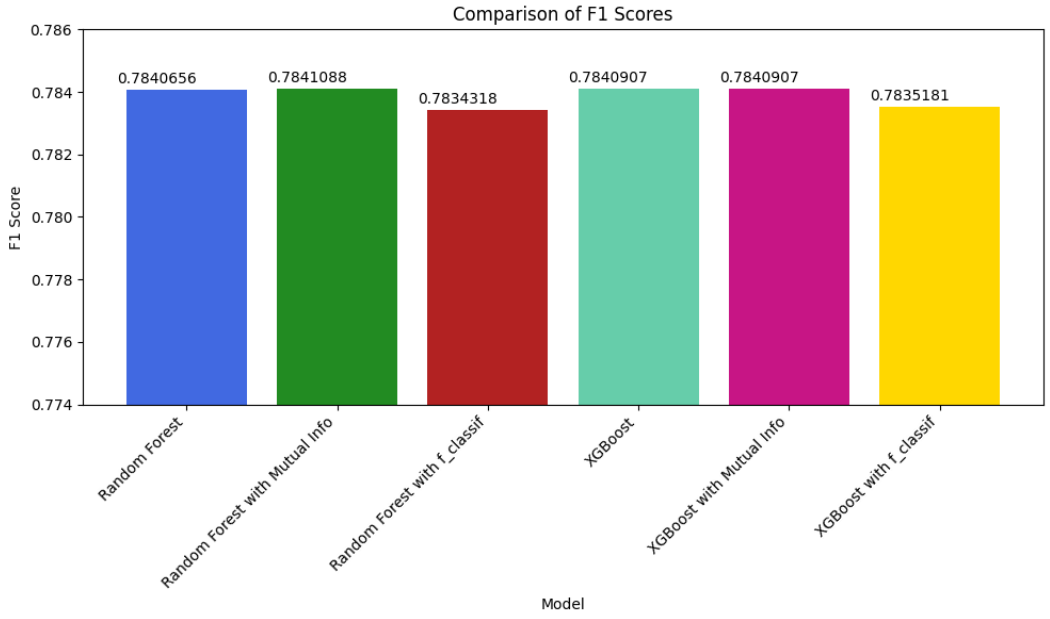
The F1 score is an ideal choice for evaluating this dataset due to its ability to balance precision and recall, particularly valuable in imbalanced datasets like yours where false positives and false negatives hold significant weight. As a harmonic means of precision and recall, it considers both metrics, ensuring high scores only when both are high. In the context of a potential "Target Attack," an F1 score provides a more nuanced understanding of model performance, making it crucial for cases where incorrectly identifying or missing an actual attack could have serious consequences.

Compare and describe the two models you will use

XGBoost and Random Forest are suitable for training this dataset due to several reasons. They can **handle** categorical variables and outliers, making them ideal for complex cybersecurity data. They are robust and can manage high-dimensional spaces and large feature sets. Both models effectively deal with imbalanced datasets, a characteristic of our target column, using ensemble learning techniques. They offer interpretable output, making it easier to understand the most significant predictors of a cyber-attack. Known for their high performance and accuracy, these models can deliver strong predictive results, which is crucial in cybersecurity to minimize false negatives.

Plot the models' results.

This plot depicts the F1 scores for each model, comparing the baseline performance with the performance after applying two different feature selection methods.



Hyperparameter tuning is correct and clear.
Following an exhaustive grid search, we extracted and display the XGBoost model with mutual_info feature selection. The objective is to pinpoint the most effective hyperparameter configuration for this dataset. It's important to highlight that, in our scenario, the F1 score remained constant irrespective of the adjustments made to the hyperparameters.

Discuss and analyze the results.
The research evaluated the performance of two machine learning algorithms: Xgboost and RandomForest. They were assessed using both the full set of features and the top six features identified through feature selection techniques. The F1 scores achieved, ranging from 0.783 to 0.784, indicate comparable and commendable performance by both models.
However, it's essential to weigh the balance between the simplicity of a model and its performance. Using a reduced number of features can lead to a more straightforward and quicker to execute model, but it might compromise the performance if crucial features are overlooked.

2- Dynamic

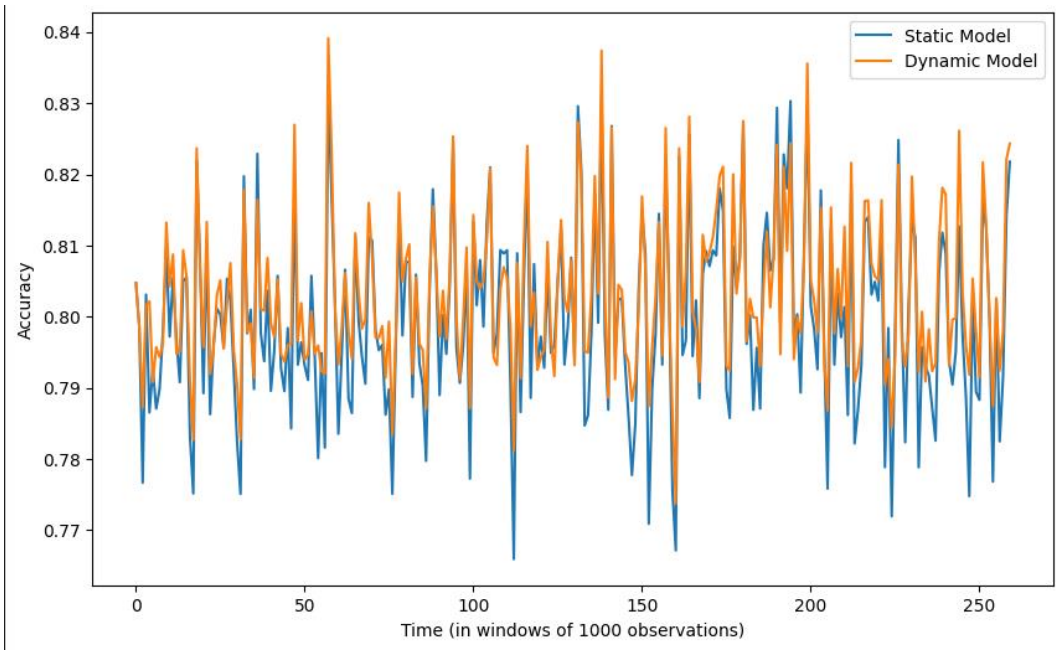
Windows use 1,000 datapoints
The consumer data initially arrives in an unstructured format, requiring cleaning. The first row, serving as the header, is disregarded. Subsequently, a function is created to extract 1000 rows from the consumer data, which are then appended to the data frame. The same preprocessing steps are applied to the static data.

Correct performance metrics are selected and justified.
I continue to utilize the F1 score as the metric for evaluation since dealing with cyber-attacks necessitates a balance between precision and recall.

Training reevaluation process is clearly described.
The model saved in the static file follows a pipeline that begins with scaling and then proceeds to the model. I take a DataFrame containing 1000 rows and input it into the pipeline, applying the same preprocessing steps. Initially, the dynamic model matches the static model. To enhance the score, I set a threshold of 0.79. If the dynamic model score falls below this threshold, a retraining process is initiated by updating the weights in the XGBoost model, followed by reevaluation.

Static and Dynamic models are evaluated.
I assessed the static model before initiating retraining, and it performed quite well. I refrain from updating the dynamic model until I ascertain whether retraining is necessary. Generally, the F1 score for both the static and dynamic models is closely aligned.

Plot the results obtained for both models



Analyze the results obtained for both models.

The static model, which does not incorporate new data, exhibits variable accuracy between 0.77 and 0.82 across different data batches, reflecting batch diversity. The accuracy of the dynamic model, which is updated with new data, is similar to the static model before retraining, suggesting retraining may not drastically change performance. However, there is some improvement in accuracy for the dynamic model after retraining, indicating that retraining does provide benefits.

Advantages/limitations and knowledge learned is clear and correct.

The dynamic model, capable of retraining with new data, generally shows improved accuracy, illustrating the benefits of integrating real-time data. However, retraining doesn't always significantly enhance performance, and the model's effectiveness varies across different data batches. From this analysis, we learn the importance of regular model retraining with streaming data and the need for models robust enough to handle data diversity.