Assignment 3

Description Continuous Evaluation with Kafka

Ahmed Badawy: 300389393

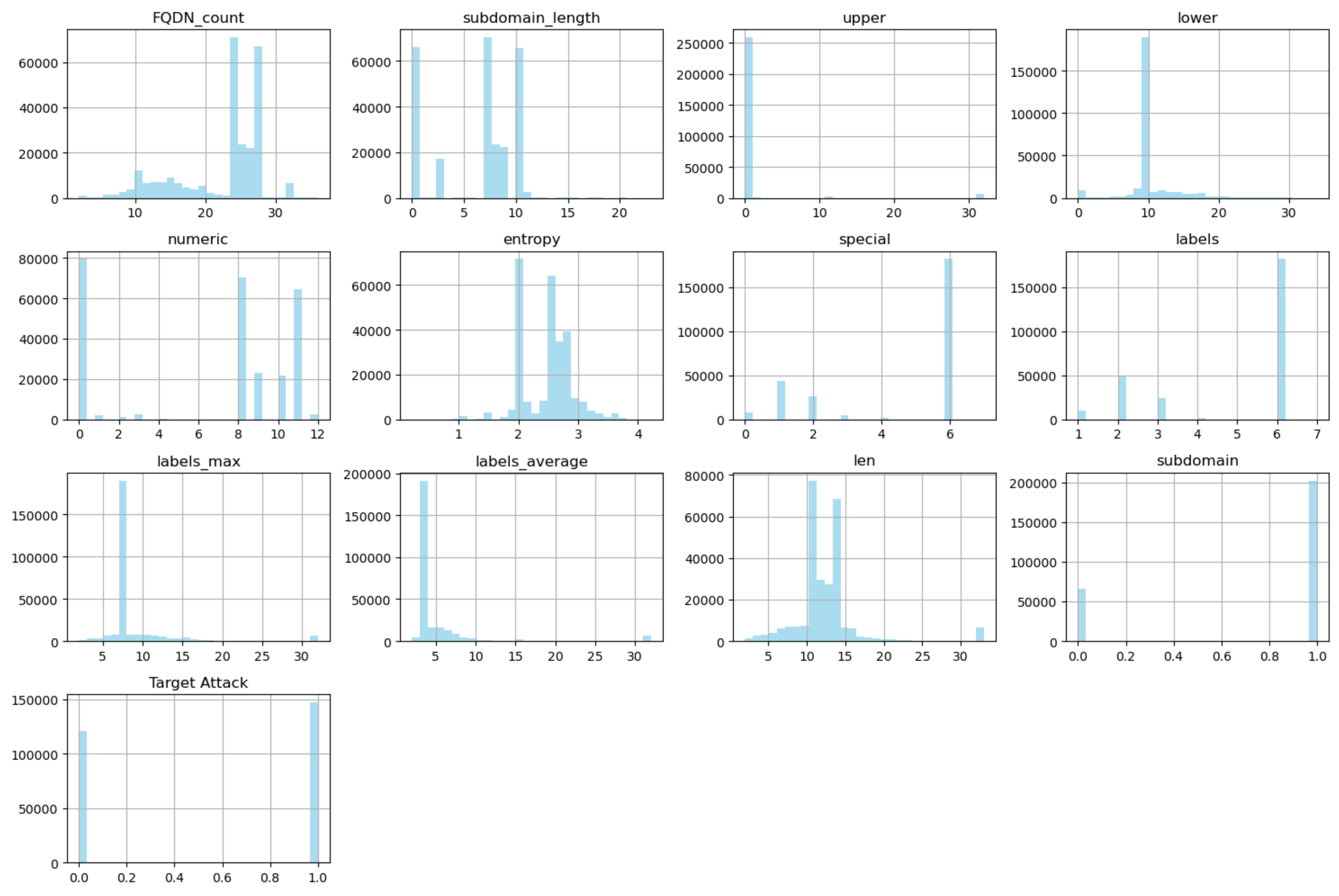
Part I (Static Model):

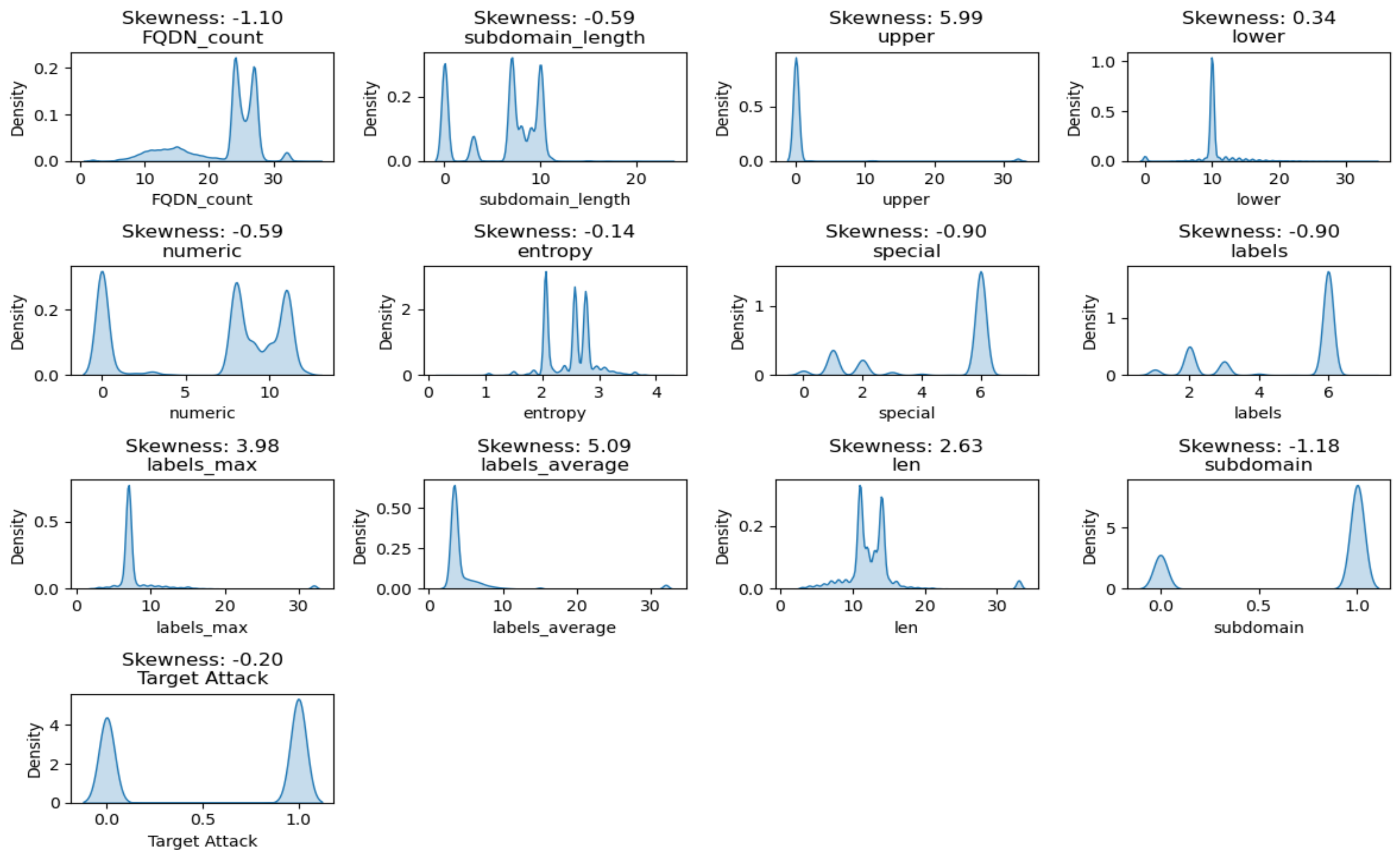
1.1 (Data Analysis):

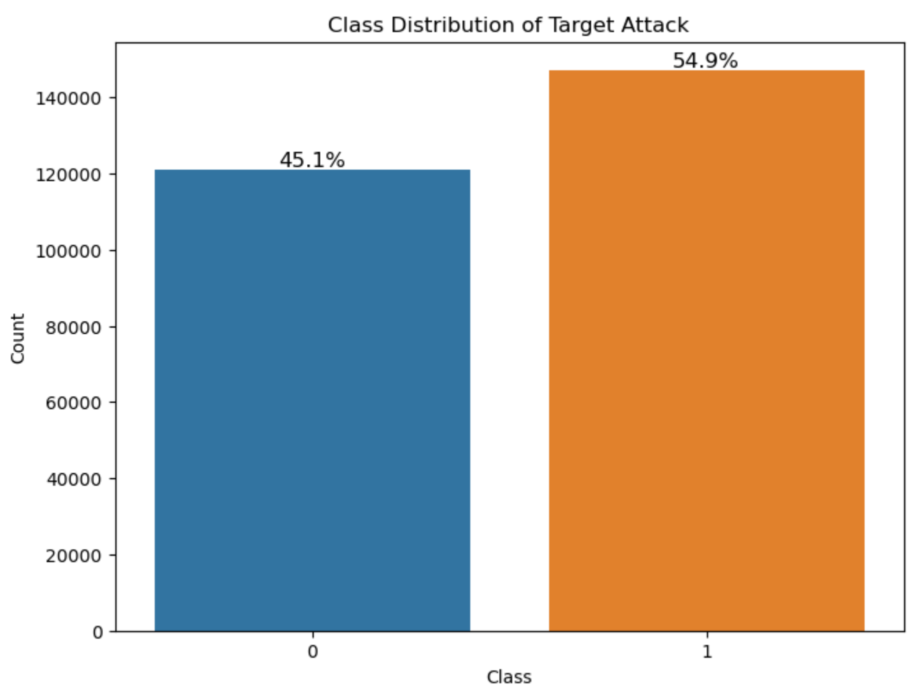
• After reading the static_dataset.csv, I found that this dataset consists of 15 features and their labels, containing 268,074 rows. It also includes two features with categorical values.

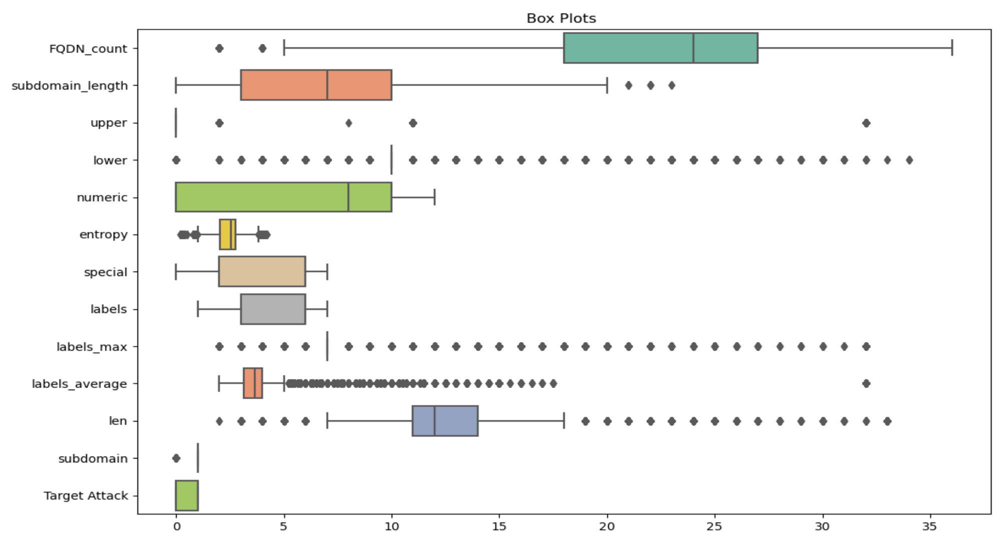
• Plotting histograms to represent the distribution of values within each feature and analyzing skewness for features help identify whether the data is skewed to the left, right, or symmetrically distributed. Identifying outliers is done using box plots.

• I checked for data imbalance and found that class 1 constitutes 54.9%, while class 0 constitutes 45.1%. This difference is minimal and does not significantly affect the model.







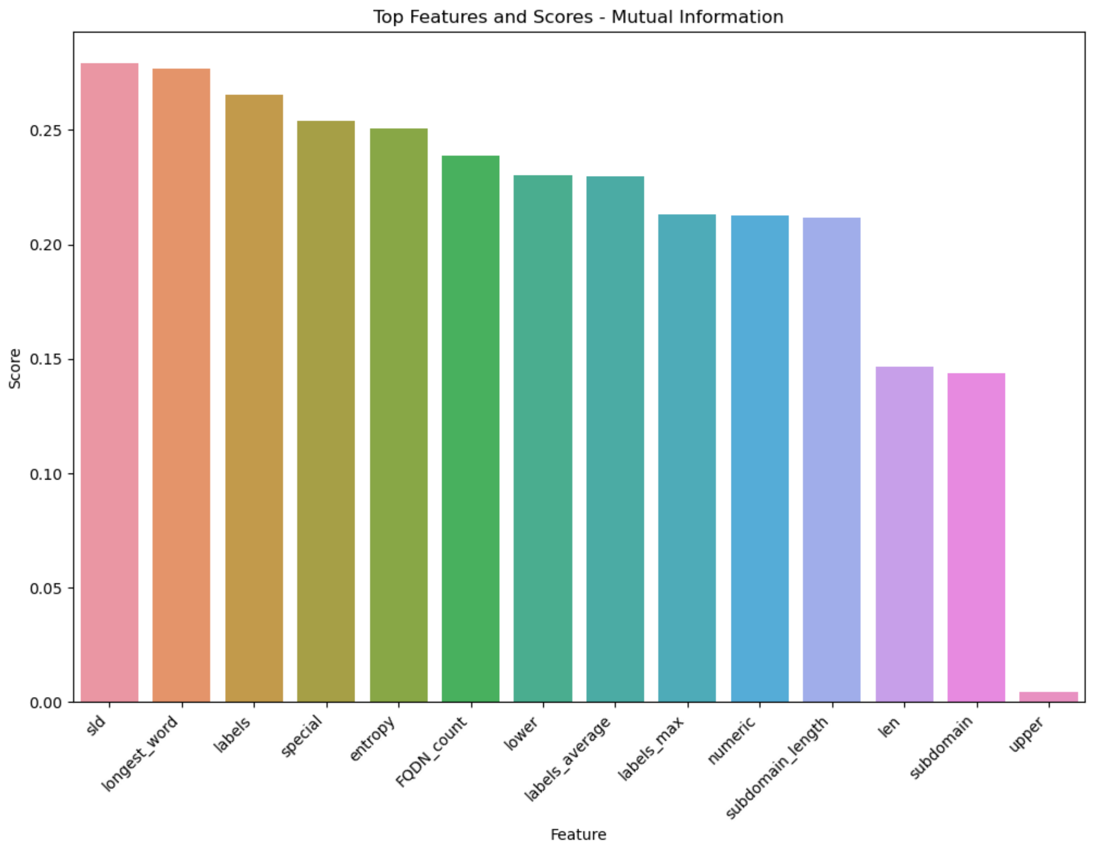


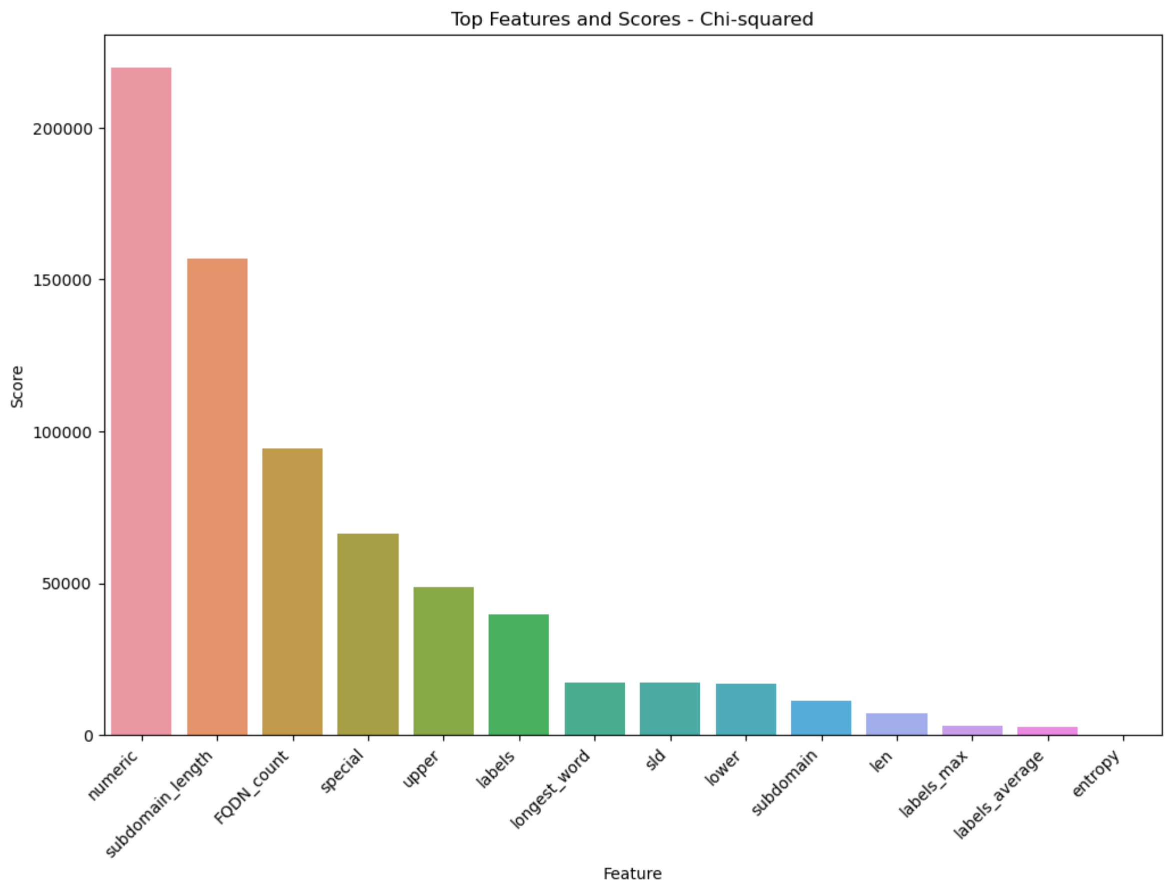
1.2(Data Cleansing and Feature creation):

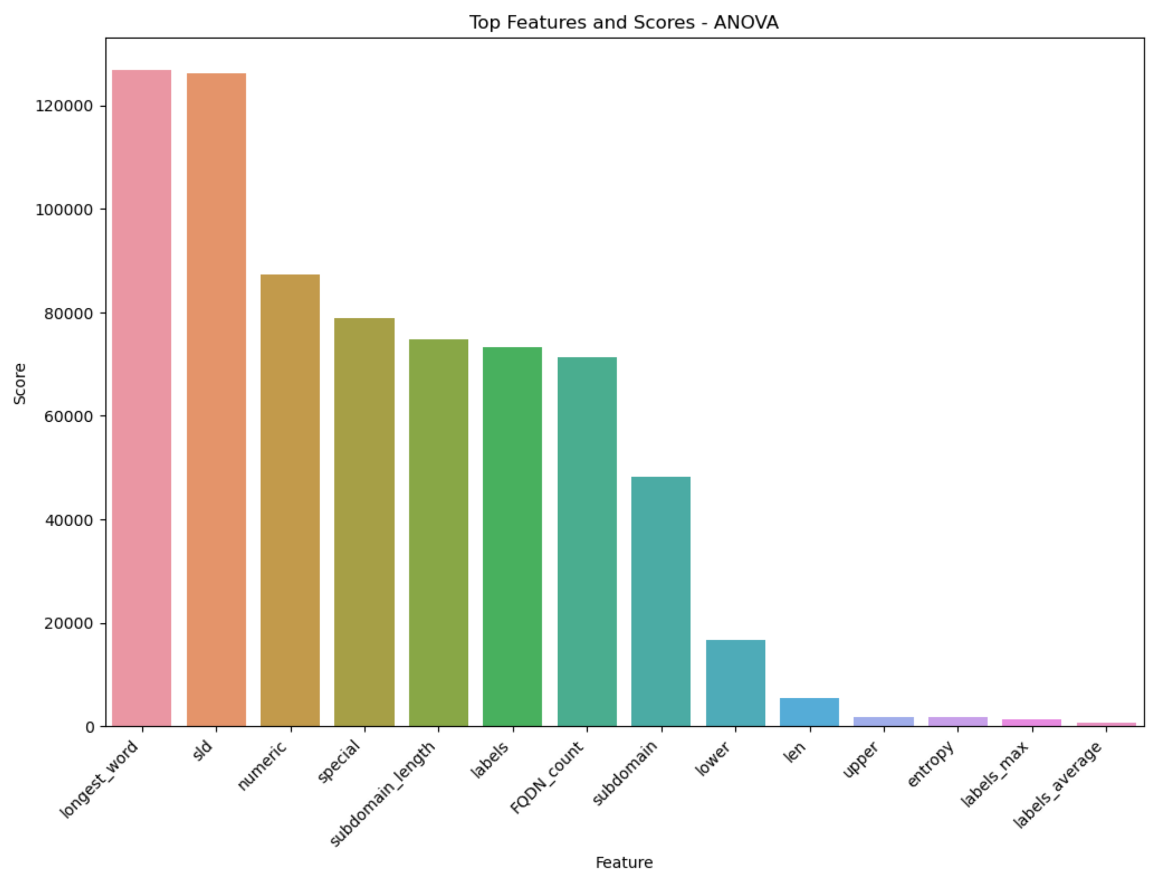
Firstly, I examined the dataset for missing values and promptly removed them. I checked the duplicate rows, identifying 91,803 instances of duplication and I made the decision to eliminate these duplicates to ensure they do not adversely impact the model. Additionally, I identified two categorical columns, namely 'sld' and 'longest_word.'. I employed the Target Encoding technique on these columns. This method assigns a numerical value to each category based on the mean of the target variable for that specific category Furthermore, recognizing that the 'timestamp' column contains numerous unique values and would not be utilized in the model, I opted to remove it from the dataset.

1.3(Feature Filtering/Selection):

I employed three feature selection methods: SelectKBest with Mutual Information, SelectKBest with Chi-squared, and SelectKBest with ANOVA. These methods were trained using all the features in the dataset to identify the most crucial ones. After the analysis, it became evident that the optimal number of features for each method was determined to be 8. Each of these selection methods pinpointed the top 8 features deemed most significant for the models' performance.







1.4 (Data Splitting):

First, I split the dataset to 80% train and 20% test, with 80% of the data allocated to the training set, the model has a substantial amount of data to learn from. Sufficient training data is essential for building a robust and accurate model.

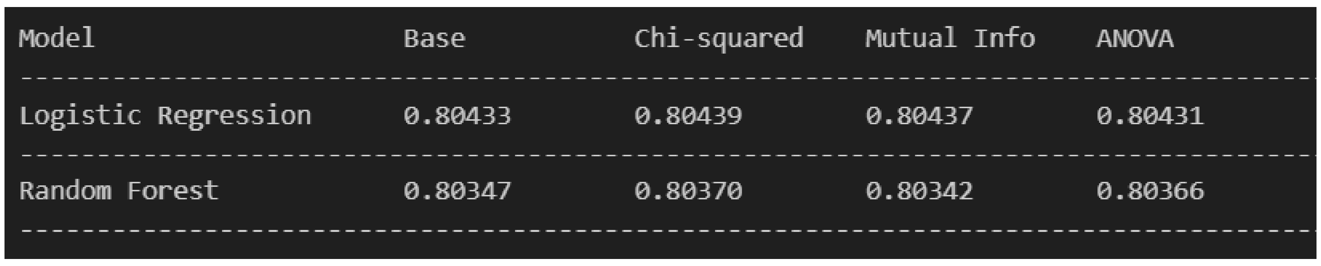
1.5 (performance metric):

I chose the F1-Score as the evaluation metric because the problem involves classifying Cyber Security attacks. In such cases, precision and recall are more relevant than accuracy. The F1-Score provides a balanced assessment, giving equal weight to precision and recall. This is important in Cyber Security, where correctly identifying and classifying attacks are both critical. The F1-Score penalizes the model for missing instances of attacks, making it a suitable metric for imbalanced class distributions.

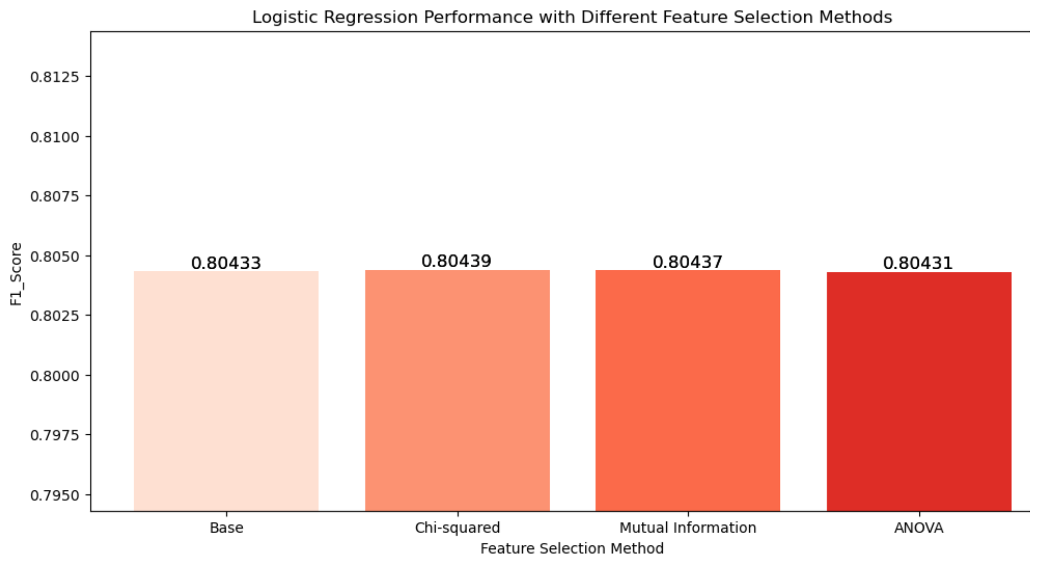
1.6 (Compare and describe the two models):

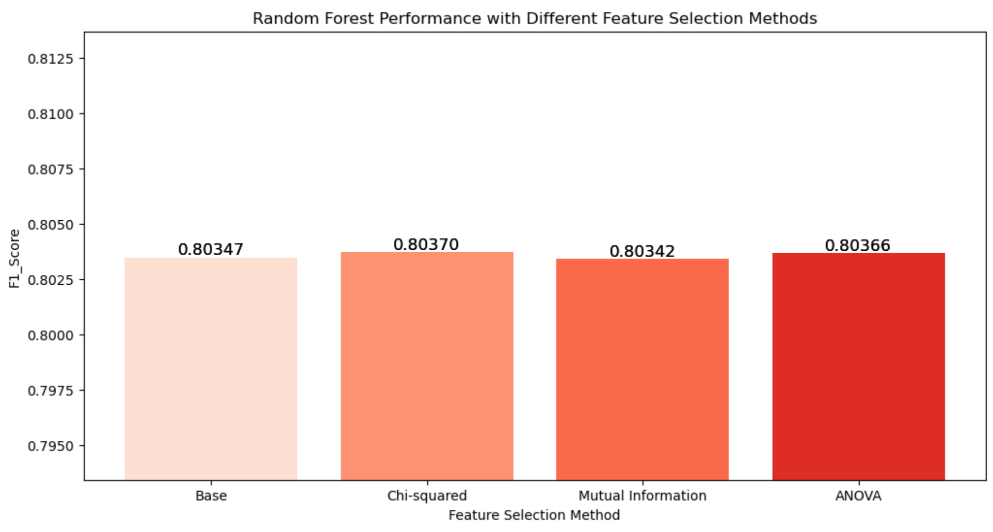
I began by normalizing the data using the Standard Scaler. Subsequently, I chose two widely used classification algorithms, Logistic Regression and Random Forest, for model development. I trained these models both with the normalized data and without feature selection, establishing them as the base models.

Next, I employed three feature selection methods and identified the features they deemed most significant. The selected features were then used to train the Logistic Regression and Random Forest models. The best set of features for Logistic Regression was determined by the Chi-squared method, achieving an F1-Score of 80.439%. Similarly, for Random Forest, the best features were obtained using the Chi-squared method, resulting in an F1-Score of 80.37%. These F1-Scores represent the models' performance on the selected features after feature selection.



```
Model                Base        Chi-squared  Mutual Info  ANOVA
---------------------------------------------------------------------
Logistic Regression  0.80433     0.80439      0.80437      0.80431
---------------------------------------------------------------------
Random Forest        0.80347     0.80370      0.80342      0.80366
---------------------------------------------------------------------
```



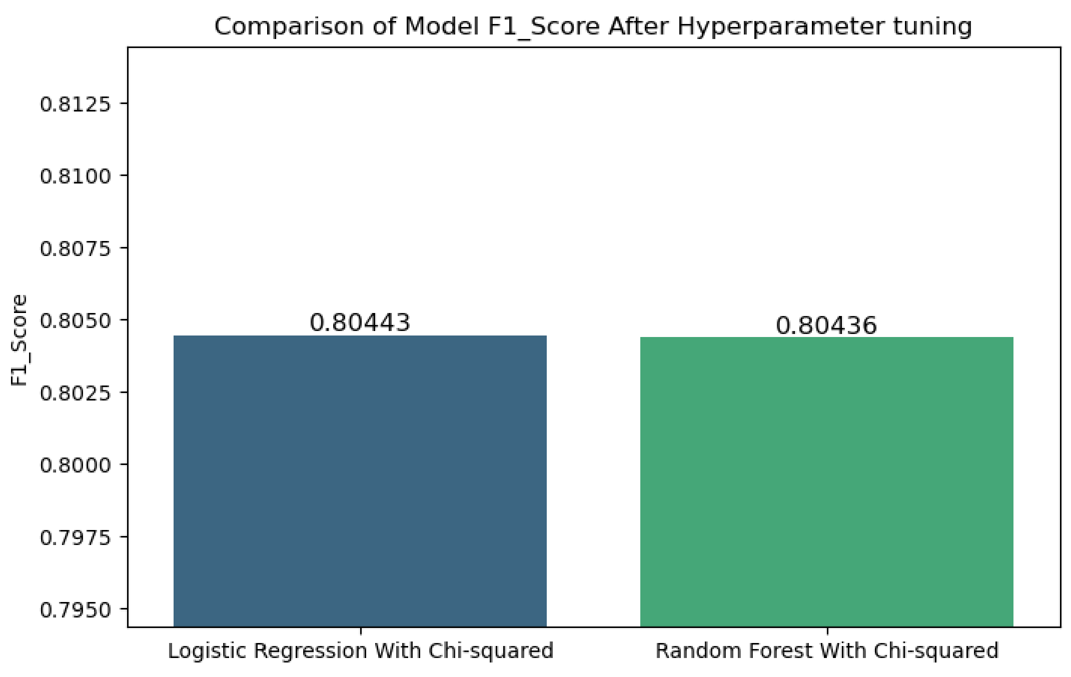


1.7 (Hyperparameter tuning):

I conducted hyperparameter tuning for both models using the features selected through the Chi-squared method. Following this process, the Logistic Regression model achieved an F1-Score of 80.443%, showcasing a slight improvement compared to its performance without hyperparameter tuning. Similarly, the Random Forest model attained an F1-Score of 80.436% after hyperparameter tuning, representing a marginal enhancement compared to its performance without hyperparameter tuning.

1.8 (Plot the models' results):

I created a figure to compare the F1-Scores of the two models, Logistic Regression and Random Forest, after hyperparameter tuning. The plot indicates that the Logistic Regression model achieved a higher F1-Score after hyperparameter tuning compared to the Random Forest model.



1.9(Discuss and analyze the results):

In the specific context of the chosen features and hyperparameter tuning, it is evident that the Logistic Regression model outperformed the Random Forest model in terms of precision and recall balance. The Logistic Regression model exhibited a higher F1-Score, showcasing the effectiveness of feature selection and hyperparameter tuning. Conversely, the Random Forest model, while showing improvement with these adjustments, did not exhibit a significant performance boost compared to the Logistic Regression model. Finally, I created a pipeline that incorporates feature selection, followed by scaling, and then the Logistic Regression classifier. This pipeline was then saved into a pickle file for future use.

Part 2 (Dynamic Model):

2.1(Windows use 1,000 datapoints):

The consumer provided all the data, prompting me to create a function that reads 1000 datapoints and appends them to a list. The function returns this list. Subsequently, I developed another function to take these 1000 datapoints, which are in binary format, and convert them into a data frame resembling the static data.

2.2(Training reevaluation process is clearly described):

Initially, I loaded the static model from the pickle file. Initially, the static model and dynamic model were the same. Following this, I created a function to preprocess the data obtained from the stream. In this function, I applied the same preprocessing steps as those used on the static data. The features were selected from the data based on the feature selection performed by the static model, and subsequently, the data was scaled using the scaler obtained from the saved model. Additionally, I randomly sampled from the static data and applied the same preprocessing to create a sample. This sample would be used in the retrain process.

Finally, I implemented a loop through the stream data, processing it in batches of 1000 rows. I applied the preprocessing, selected features, and scaling to test this data on both the static and dynamic models. If the dynamic F1-Score fell below a specified threshold (0.85), I initiated the retraining of the dynamic model using the sample from the static data and the previous stream data.

2.3(Correct performance metrics are selected):

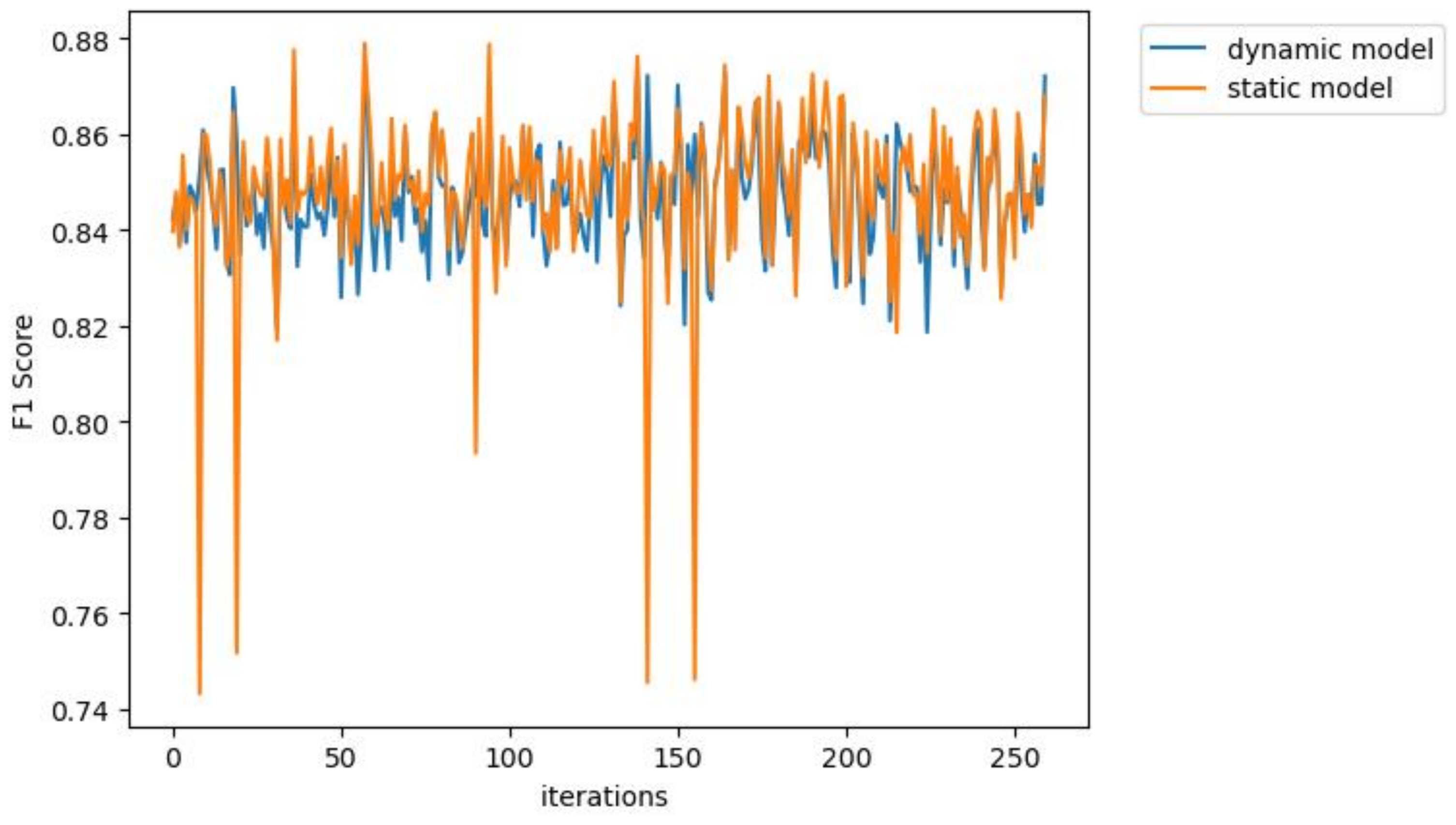
I continue to use the F1-Score because the nature of the problem revolves around classifying Cyber Security attacks. In such cases, precision and recall are more relevant metrics than accuracy. The F1-Score offers a balanced assessment by giving equal weight to both precision and recall. In the realm of Cyber Security, where accurately identifying and classifying attacks are equally crucial, the F1-Score serves as a suitable metric for evaluating model performance.

2.4(Static and Dynamic models are evaluated):

I stored the static and dynamic F1-Scores for each batch in a list to facilitate visualization of their performance. In the event that the dynamic F1-Score dropped below a predefined threshold (0.85), I initiated a retraining process and stored the updated F1-Score after retraining. This approach allowed for continuous monitoring and adaptation of the dynamic model's performance during the streaming process.

2.5(Plot the results obtained for both models):

This plot illustrates the F1-Scores for both the static and dynamic models across each batch.



2.6(Analyze the results obtained for both models):

The plot indicates that the static model remains relatively stable for the majority of the streaming data, with occasional fluctuations, albeit minor. In contrast, the dynamic model, when falling below the defined threshold, undergoes retraining, resulting in a fluctuating F1-Score range. The F1-Score for the static model typically ranges from 75% to 87%, whereas the dynamic model exhibits a slightly higher range, varying between 82% and 87%. This suggests that the dynamic model's adaptability through retraining allows it to achieve comparable or improved performance compared to the static model in certain instances.

2.7(Advantages/limitations and knowledge learned):

Advantages: Advantages of the dynamic model include its effectiveness in real-time scenarios, where it continuously adapts to new data. This adaptability is particularly notable when the F1-Score falls below the specified threshold, triggering an automatic retraining process. This inherent capability enhances the stability of the dynamic model, allowing it to consistently perform well with evolving data, a characteristic that is especially beneficial in dynamic and changing environments.

Limitations: Despite the dynamic model's ability to retrain when falling below the threshold, there are instances where the F1-Score may not necessarily improve after retraining. Additionally, in real-time scenarios with a large volume of streaming data, continuous retraining on all the previous data can impose significant memory and resource requirements. This could potentially lead to scalability challenges and increased computational overhead. Careful consideration is needed to balance the benefits of real-time adaptability with the associated computational costs.

Knowledge Learned: Through this simulation, I gained insights into the deployment of machine learning models in real-time scenarios. I learned strategies for ensuring model stability, particularly the use of dynamic models that adapt to new data by retraining when necessary. This experience provided valuable knowledge on managing and monitoring models in dynamic environments, emphasizing the importance of maintaining stability over time.