Continuous Evaluation with Kafka

Prepared by:

Esmael Alahmady Ebrahim Ezz, 300389885

1 Part 1) Static Model:

1.1 Data analysis:

- Using the static_dataset.csv data, I inspected the data information and found that it contains approximately 268 thousand rows with no missing values in any of the features, as shown in [Figure 1]. This indicates that the data is complete and clean, which is essential for further analysis.
- After checking for outliers in the data using box plots, as shown in [Figure 2], I found that outliers are present in most of the features.
- Upon investigating data skewness, as shown in [Figure 3], I found that four features have significant positive skewness (+2 points), while the other features have positive or negative skewness less than 1 point.
- I checked the target label balancing and found that class 1, which indicates that there is an attack, is slightly more than class 0, which indicates that there is no attack or normal activity, by 5%. However, I believe that this will not affect the model's neutrality. A pie chart illustrating the percentage of each class is shown in [Figure 4].

1.2 Feature engineering and data cleaning:

- While inspecting the feature types and value counts, I found two unusual columns. One column has over 6,000 unique values of mixed types (strings, characters, and integers), all of which are classified as objects. The other column has over 11,000 unique values with the same characteristics, as shown in [Figure 5]. To handle these two features like the rest, I decided to use TargetEncoder, a technique for encoding categorical features into numerical features. It works by replacing each categorical value with a weighted average of the target variable values for that category and the overall target variable mean. The smoothing parameter controls the strength of the regularization.

Next, I dropped the timestamp column, as I will not be using it, and split the data into features and target for the next phases.

1.3 Feature filtering/selection:

- I used three techniques:
    - SequentialFeatureSelector with logistic regression: I looped over all the features to find the best combination that gave the highest accuracy with the model. [Figure 6] shows a line plot of the accuracies for all feature combinations. I selected the seven features with the highest accuracy for further use.
    - SelectKBest with mutual information: I used the mutual_info_classif score function to select the features that were most informative about the target variable. [Figure 7] shows a line plot of the accuracies for different numbers of selected features. I selected the seven features with the highest accuracy.
    - SelectKBest with chi-squared test: I used the chi2 score function to select the features that were most statistically significant in relation to the target variable. [Figure 8] shows a line plot of the accuracies for different numbers of selected features. I selected the nine features with the highest accuracy.

1.4 Model training:

- I decided to use all three sets of features in the modeling phase. I split each dataset into train and test sets with an equal percentage (70% train, 30% test) and stratified the split using the target feature. I then used the StandardScaler to scale the data before feeding it to the models.
- I chose two models: logistic regression and random forest. I chose these two models because they work differently:
    - Logistic regression is a linear model that automatically selects features and is easier to interpret.
    - Random forest is an ensemble model that uses a random subset of features at each split and is more powerful but less interpretable.

- For each train-test split in the three datasets, I used grid search to find the best model parameters for each model, according to the data, by evaluating the test set accuracy, precision, recall, F1 score, and confusion matrix. The results are shown in [Figure 9], [Figure 10], [Figure 11], [Figure 12], [Figure 13], [Figure 14]

1.5 Model evaluation:

- Next, I created a bar plot to visualize all the results, displaying the four metrics for each split of each model [Figure 15]. I ultimately chose the random forest model with the dataset number 3 and 9 features, which gave me the best results, even though these results were very close to those of other models with other datasets.
- Finally, I saved the best model and the scaler to use them in the dynamic part

1.6 Discussion:

- When comparing the performance of the two models on the three datasets, I found that the logistic regression model had good overall accuracy, but there were some false negatives, meaning that the model incorrectly predicted a small number of attacks as normal activity.
- The performance of the random forest model was very similar to the logistic regression model, but the random forest model had significantly fewer false negatives, meaning that it correctly identified more attacks.
- My chosen model that the one predicts the least number of false negatives illustrating this in my confusion matrices

2 Part 2) Dynamic Model:

2.1 Instructions

- After following the instructions from installing docker to installing kafka and dependencies, I finished ingesting the data and ready to get chuck by chunk

2.2 Dynamic evaluation

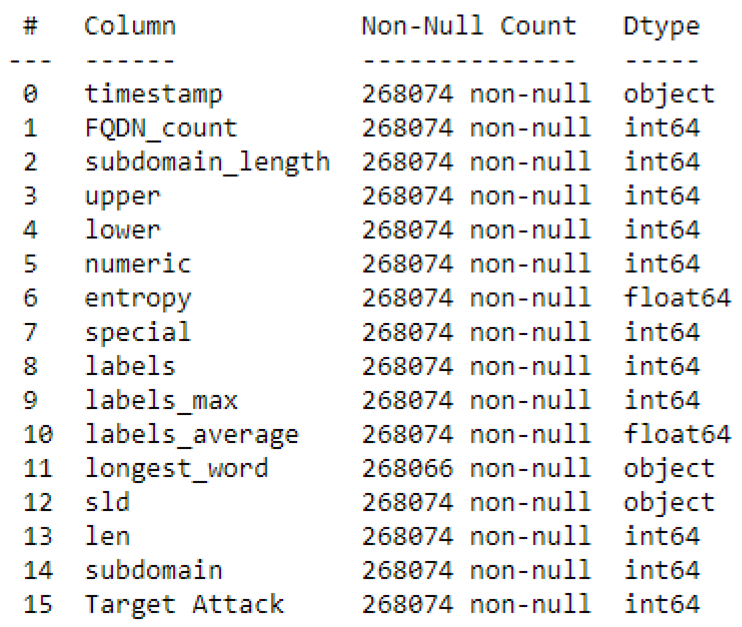
- First, I created a function to get a thousand samples of data at a time, clean it, and put it in a dataframe. I then selected the 9 best features from the previous step and loaded the random forest model. I assigned this model to both the static and dynamic models, and I also loaded the scaler from the previous step.

- Next, I used the two models to predict the thousand samples. In the first iteration, I got the exact same results from both models. I then set a condition that if the dynamic model's precision fell below 75%, I would retrain it on all of the previous data. To prevent overfitting, I decided that until the data reached 10,000 samples, I would use all of the data for retraining. However, if the data exceeded 10,000 samples, I would only use 5,000 samples for retraining.
- Although recall is generally considered a more important metric than precision, I have chosen to focus on precision in this case. This is because I have already selected the model that yielded the highest recall score and minimized the number of false negatives. Now, I am trying to retrain the model to minimize the number of false positives. I have chosen a precision threshold of 75% because this is the lowest precision value I have observed in my previous experiments. By retraining the model, I aim to improve its overall performance.
- If the condition happened and I retrained the model, I would use the retrained model in the next window, not the current one, because the current window would now be seen data.
- Finally, for 268 windows, each with 1,000 samples, I calculated the static and dynamic precision and displayed a line plot [Figure 16] showing the difference between the two models in each window.

2.3 Discussion

- Comparing the precision of the two models are so difficult because the 2 precisions are so close to each other just there are few windows they are varying from each other most of them are after retraining the dynamic model. This means the pretrained dynamic model fed on a small number of data comparing to the static one but this still be a very good performance and take less much time to train over the new data.

3 Appendix



```
#   Column           Non-Null Count   Dtype
---  ------           --------------   -----
0   timestamp        268074 non-null  object
1   FQDN_count       268074 non-null  int64
2   subdomain_length 268074 non-null  int64
3   upper            268074 non-null  int64
4   lower            268074 non-null  int64
5   numeric          268074 non-null  int64
6   entropy          268074 non-null  float64
7   special          268074 non-null  int64
8   labels           268074 non-null  int64
9   labels_max       268074 non-null  int64
10  labels_average   268074 non-null  float64
11  longest_word     268066 non-null  object
12  sld              268074 non-null  object
13  len              268074 non-null  int64
14  subdomain        268074 non-null  int64
15  Target Attack    268074 non-null  int64
```

Figure 1: Static_data information

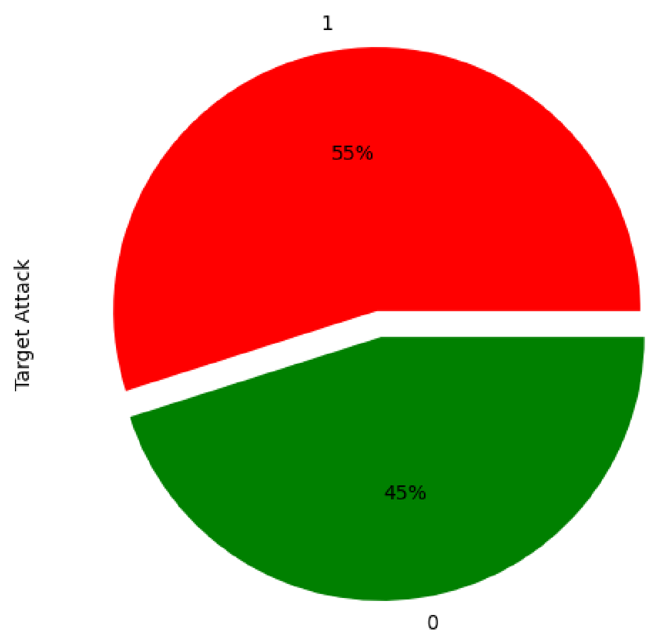


Figure 2: Target feature pie chart

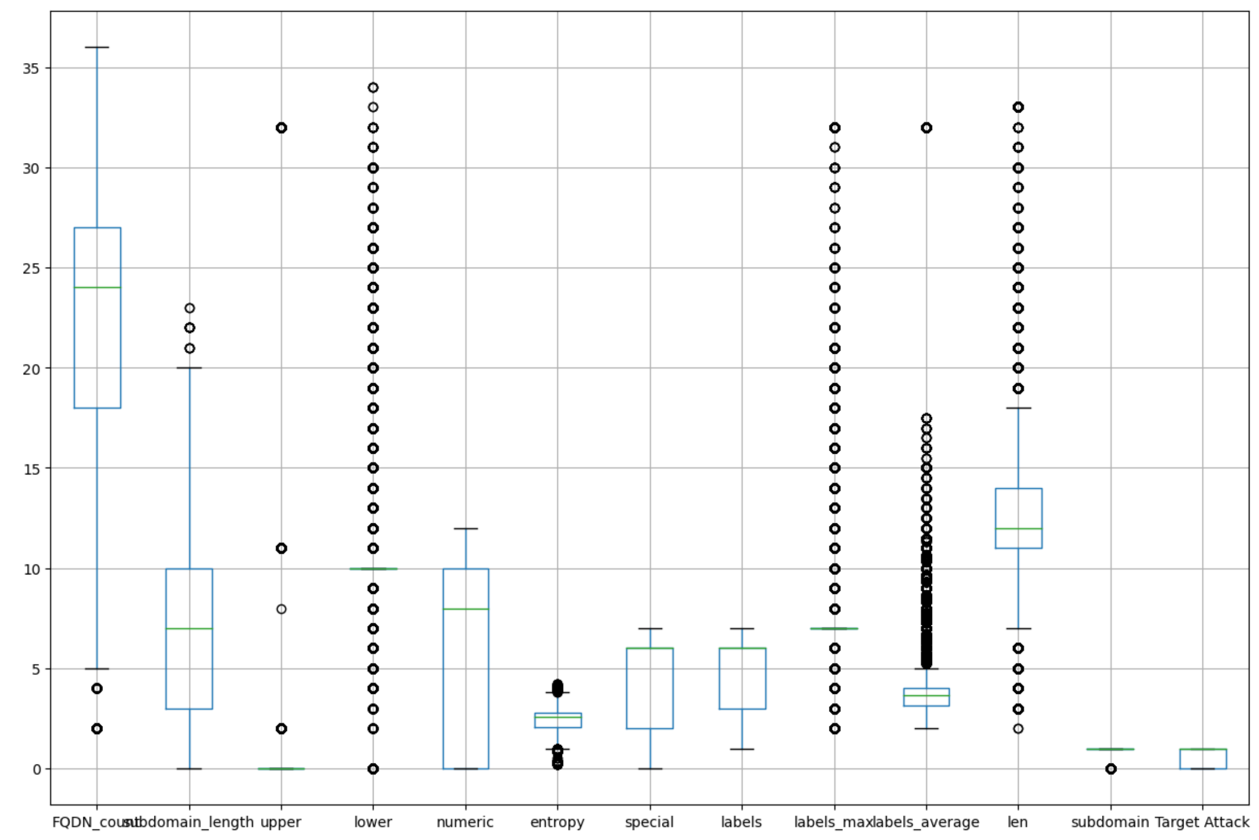


Figure 3: Data feature box plot

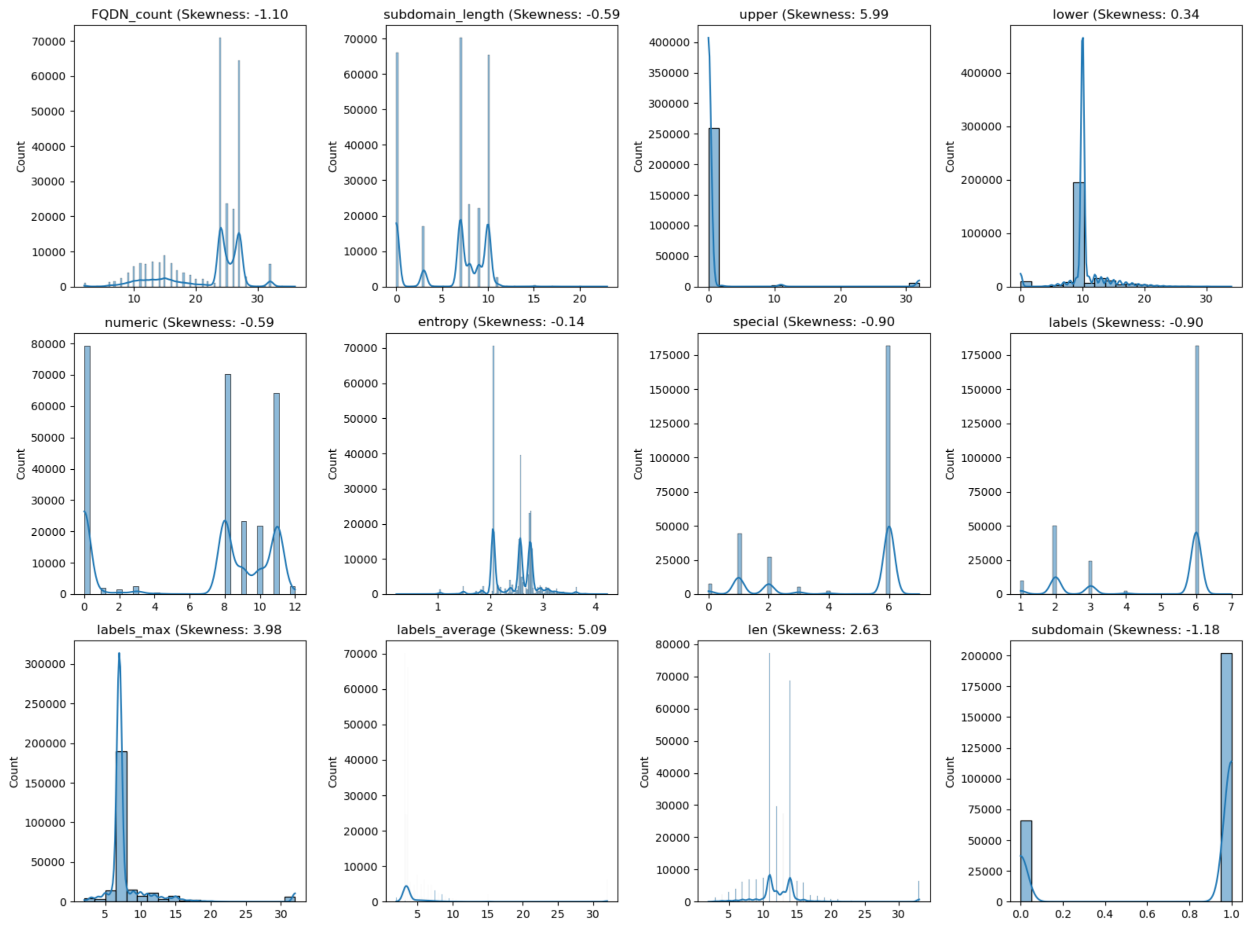
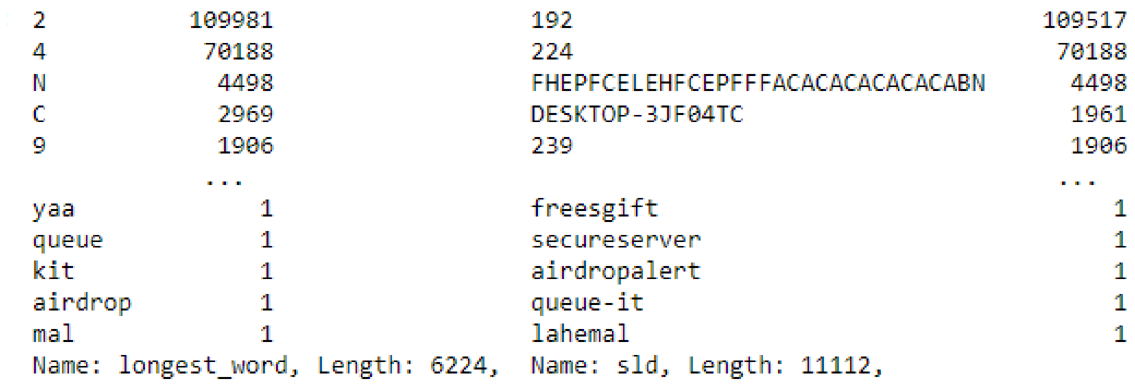


Figure 4: Data statistical analysis



```
2          109981         192                                  109517
4           70188         224                                   70188
N            4498         FHEPFCELEHFCEPFFFACACACACACACABN       4498
C            2969         DESKTOP-3JF04TC                        1961
9            1906         239                                    1906
              ...                                                 ...
yaa             1         freesgift                                 1
queue           1         secureserver                              1
kit             1         airdropalert                              1
airdrop         1         queue-it                                  1
mal             1         lahemal                                   1
Name: longest_word, Length: 6224,  Name: sld, Length: 11112,
```

Figure 5: Object features unique values

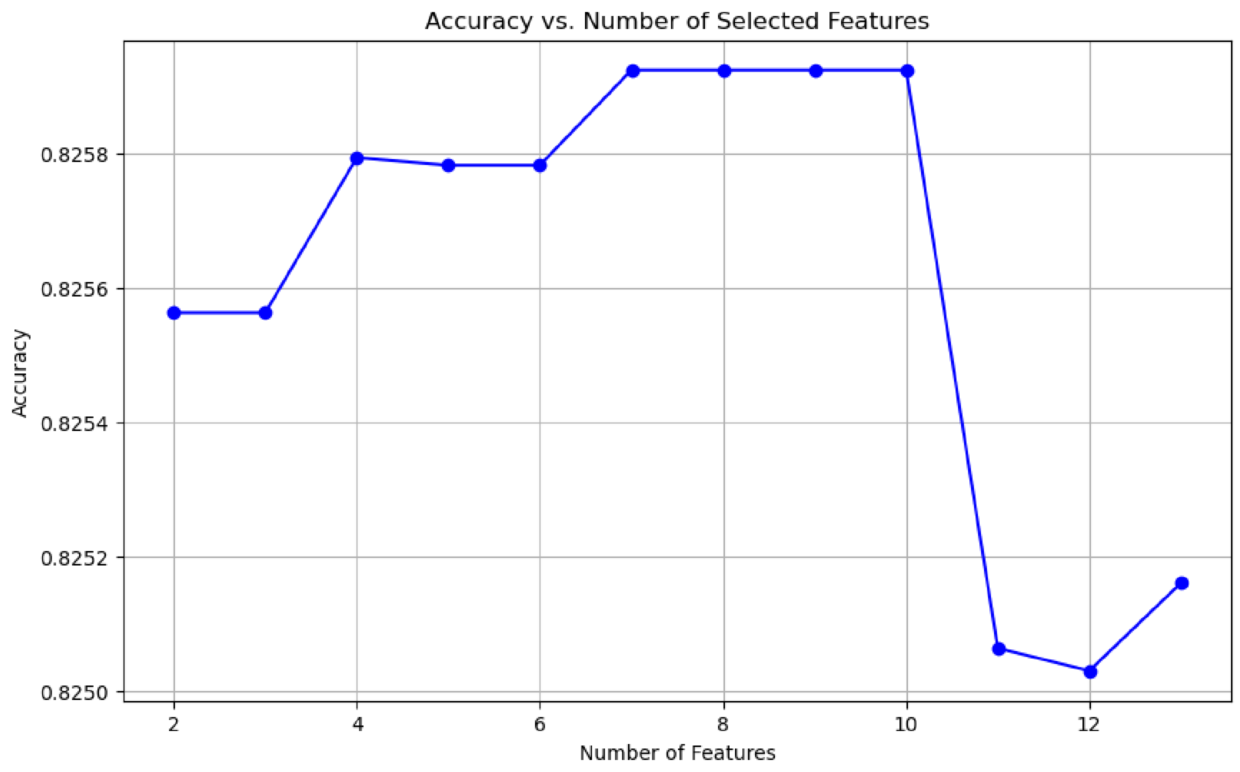


Figure 6: Accuracy values for different numbers of features in sequential feature selector

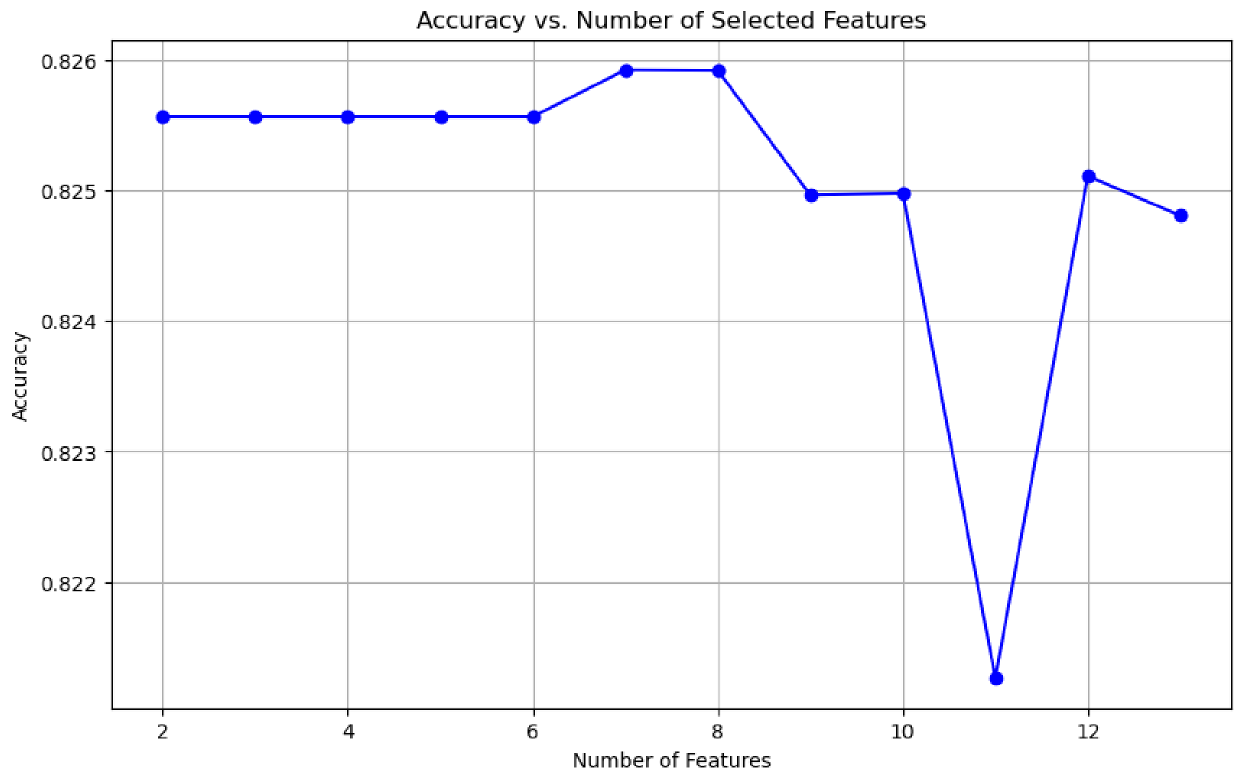


Figure 7: Accuracy values for different number of features in select best k with the mutual_info_classif score function

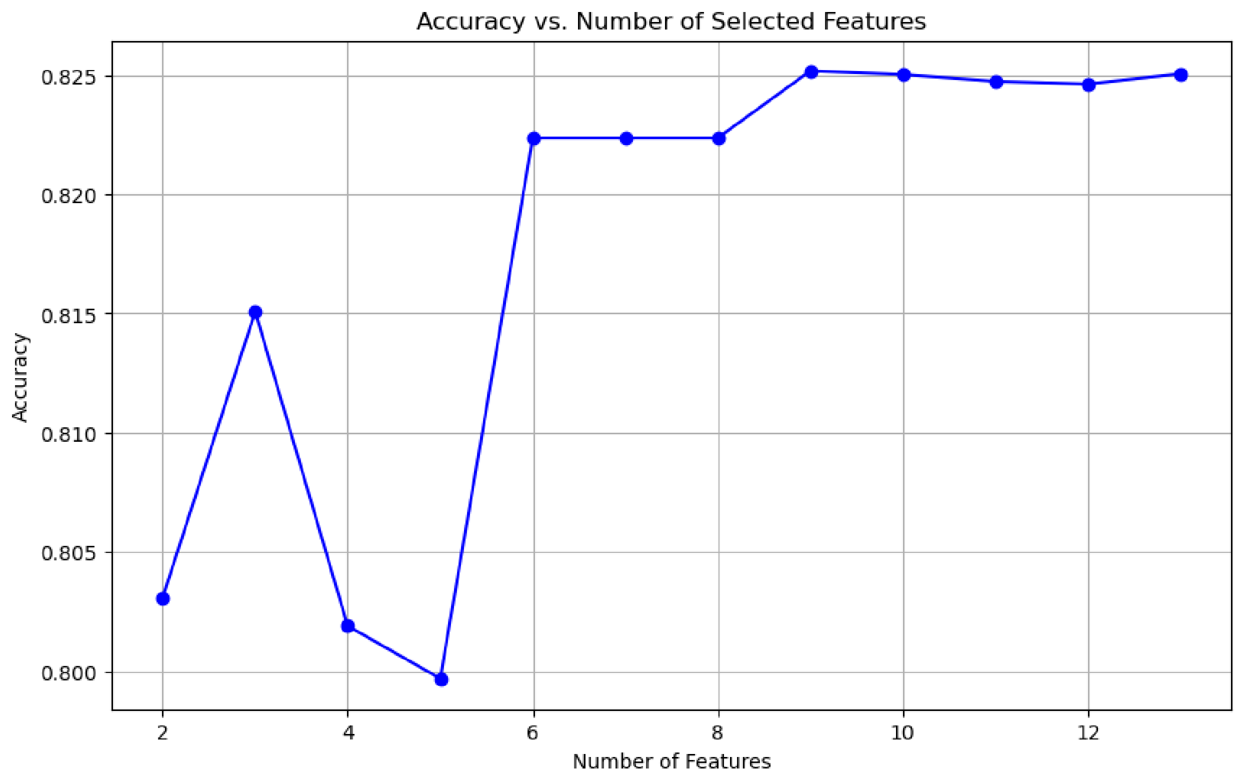


Figure 8: Accuracy values for different number of features in select best k with the chi2 score function

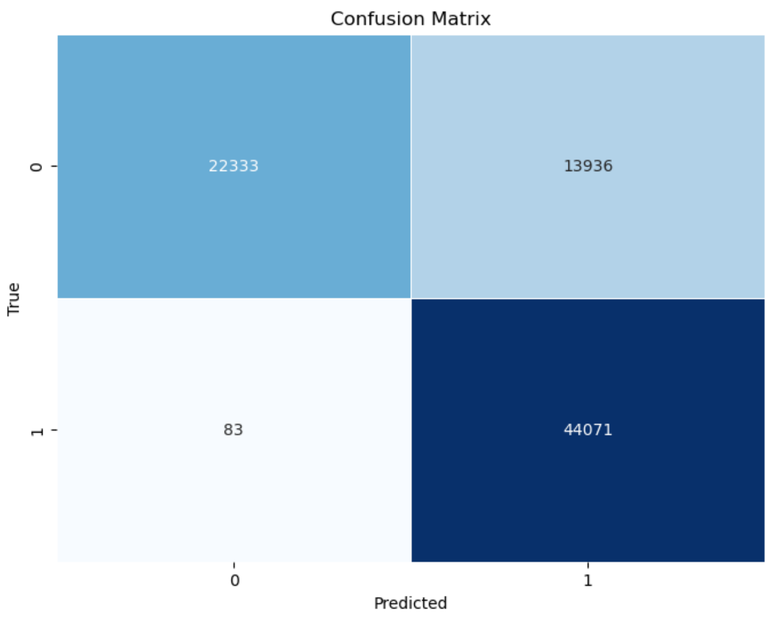


Figure 9: LR model with first data subset

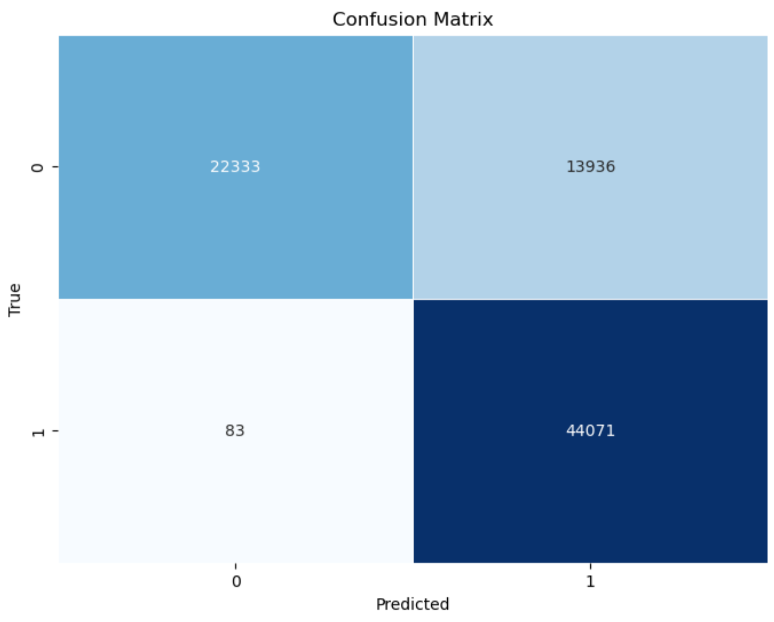


Figure 10: LR model with second data subset

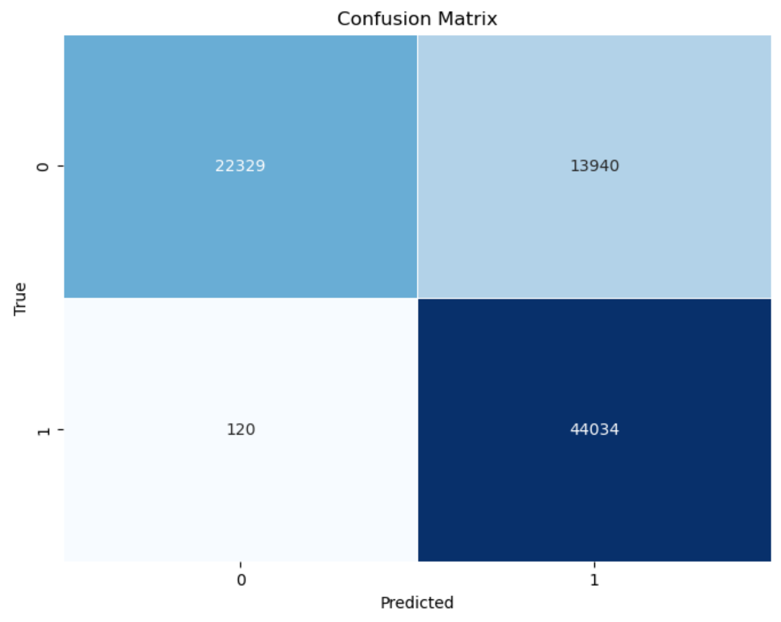


Figure 11: LR model with third data subset

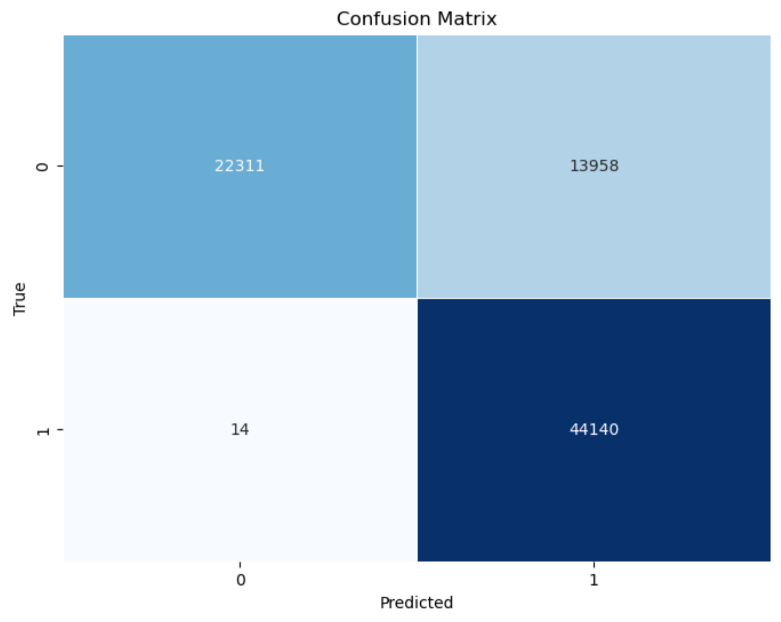


Figure 12: RF model with first data subset

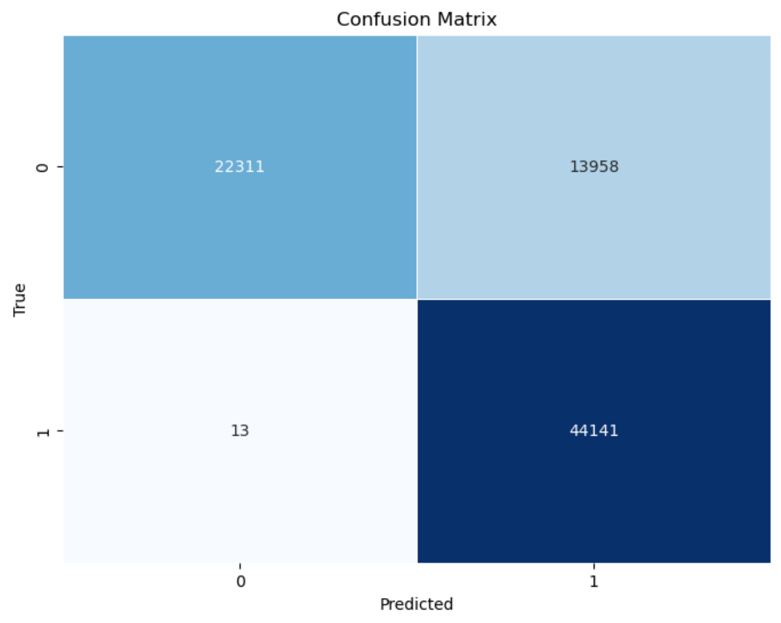


Figure 13: RF model with second data subset

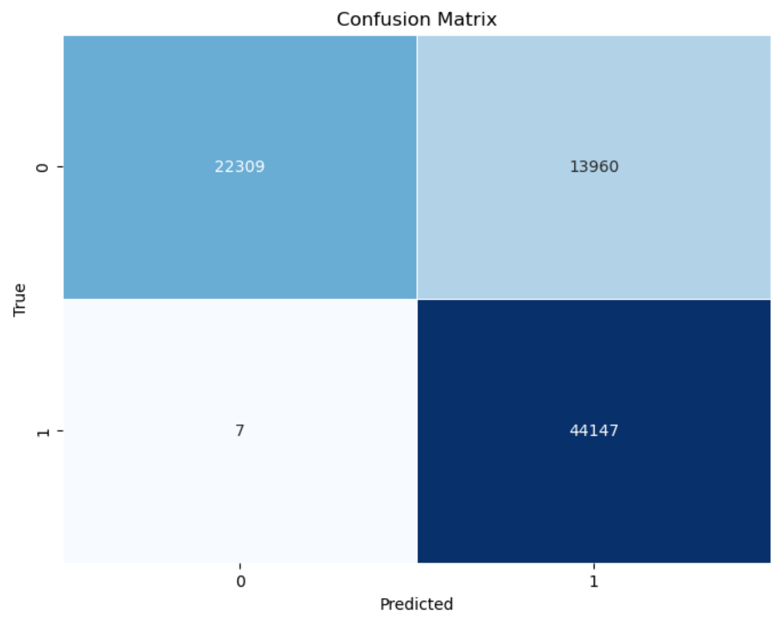


Figure 9: RF model with third data subset

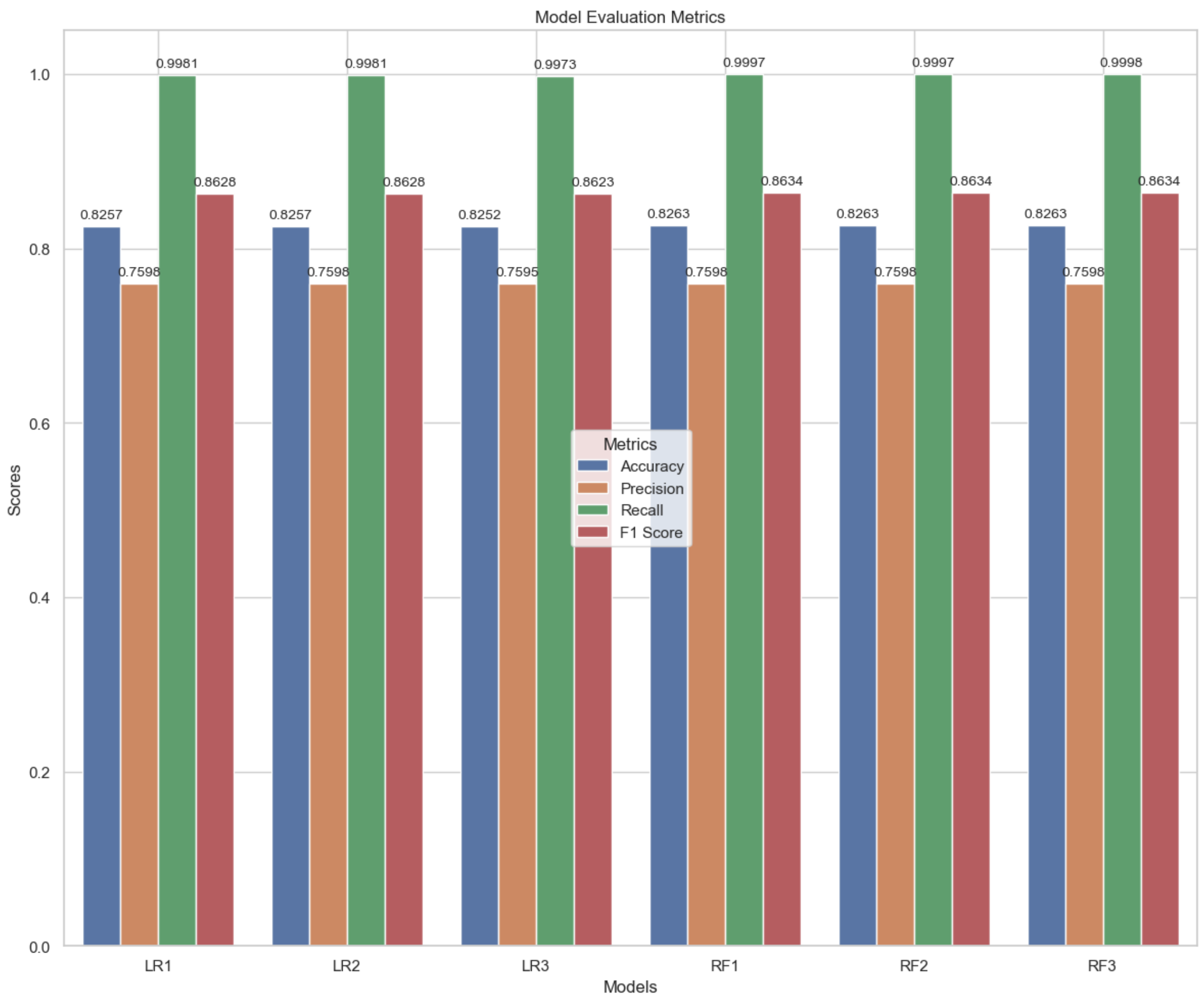


Figure 15: Bar plot for different metrics over three diffirent subsets for the two models

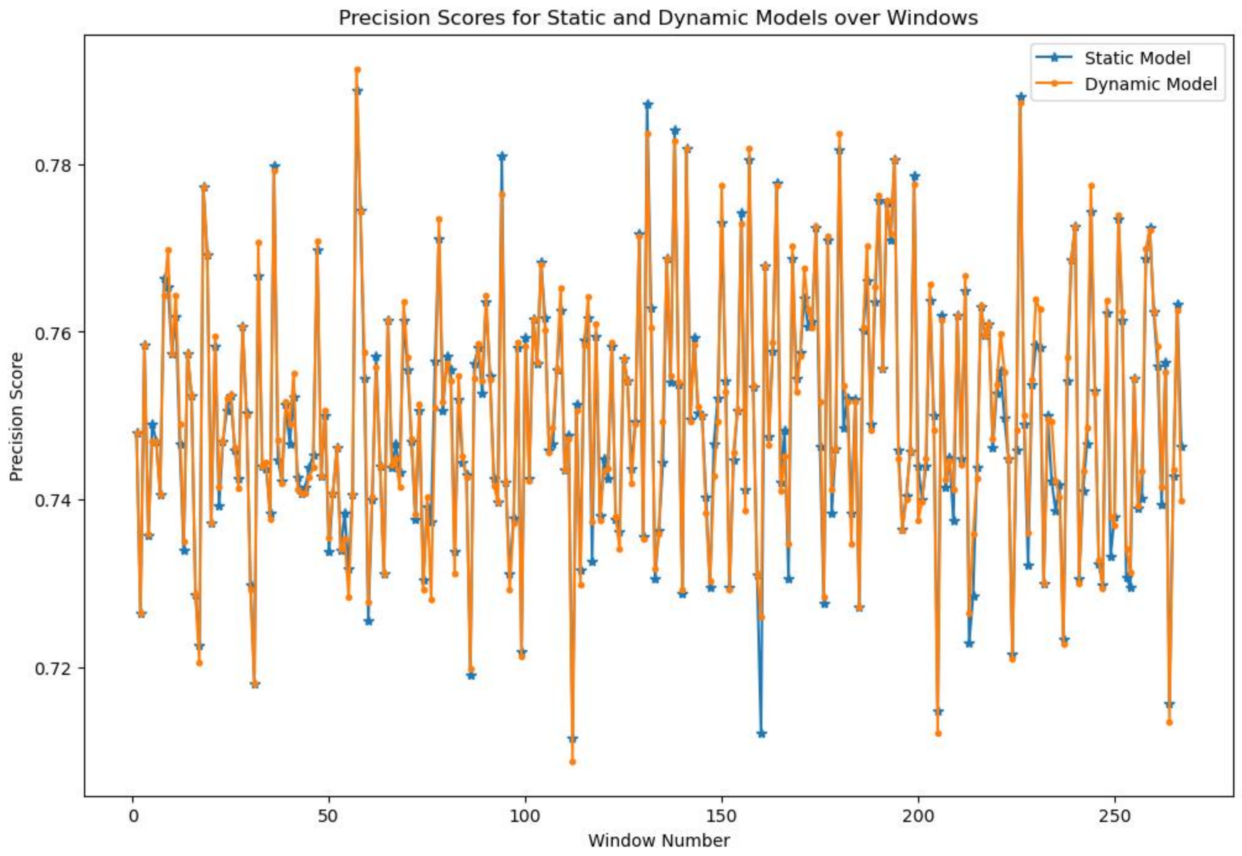


Figure 16: Line plot for comparing precision for the static and dynamic models