Assignment – 3

Static Model:

The static model is an offline model trained once. This model serves as a baseline trained model to construct dynamic models incorporating continuously flowing data in real-time. The following steps are followed for data analysis and model training,

Feature Engineering:

Feature Engineering helps to understand the data distribution and convert data into more digestible format to improve the performance of machine learning algorithms.

Data Description - The static_data set has 15 predicting features and one target feature namely "Target Attack" with 268074 records.

Missing values - Checked for NULL values using ".isnull" which reveals missing values in "longest_word". longest_word is" Longest meaningful word over domain length average" and has a range of values, it is decided to delete the rows as '8 rows' account for negligible value quantifying total data count.

Duplicate Values - More than one record that is the same. Duplication can lead you to make incorrect conclusions by leading you to believe that some observations are more common than they really are. The data have 91803 duplicate values among 268066 records which are deleted using ".drop_duplicates()".

Transforming the Variables – Converting the existing features that helps to provide more insight on data and are compatible for most classification models.

- "timestamp" - This feature is in the format of HH-MM-SS. To convert it into numerical data by converting time string into seconds as float64.
- "sld" and "longest_word" - These are mixed type columns. So, applying hash function to columns of data frame to create an identifier for the record and its value differs for even slightly different datapoints making it perfect for generating unique records.

All the data types are now either "int64" or "float64" making them numerical columns suitable for data analysis.

Data Analysis:

View data – ".info()" is used to understand the basic information of data such as columns, dtype and shape of data. Then ".describe()" gives the statistical summaries of all numerical features.

Histogram plot – All the features are plotted over 15 bins (Figure 1) and it is inferred that few columns have highly skewed data while others have normal distribution mentioned below.

- Multimodal Distribution – 'timestamp'.
- Outliers - 'upper', 'lower', 'entropy','labels_max', 'labels_average' ,'len'.

Visualize Outliers – To better visualize the outliers and better analyze each feature box plot is created on numerical data (Figure 2) and see that lower, lables_max and lables_average have too many outliers.

Target Distribution - 147179 of 268074 number of Attacks and it is 54.90% of the training set and 120895 of 268074 of non-Attacks and it is 45.10% of the training set, resulting in data imbalance. As the imbalance is 10%, balancing doesn't make any improvement in model performance in this problem (Figure 3).

Feature-Target Relationships - Visualize the distribution of attributes for data instances for each class.The data is grouped by the class attribute (two groups) then a matrix of histograms is created for the attributes in each group. However, 'timestamp', 'subdomain', 'lower' shows good dependency on target.

- Label '0' - Have normal distribution with a wide range of values over each feature.
- Label '1' - Features have a high number of outliers except for timestamp.

Feature Selection:

Features are separated into predicting and target features. And the dataset is split into 80% train and 20% test data using "train_test_split". In Total, three feature selection techniques are used, and the final feature selection set is obtained by voting method.

Features_selected : It is a self-defined function to extract features based on feature importances.

RandomForestClassifier – Calculates the extent that each feature reduces the impurity, while training a tree. A feature's importance increases depending on how much it reduces impurity.

Parameters - n_estimators=100, random_state=0, class_weight= 'balanced', n_jobs=-1

GradientBoostingClassifier - Feature selection method Recursive Feature Elimination, wrapper-style feature selection which uses filter-based feature selection internally.

Parameters - (n_estimators=100, random_state=1), n_features_to_select=8

Lasso – Feature selection method, Recursive feature elimination with cross-validation to select features is used. RFECV finds the best number of features to keep using Lasso.

Parameters – estimator = Lasso (), cv=3, scoring="r2", min_features_to_select=8

All the three techniques have different ways for assigning optimum feature importance that helps in model performance and importantly captures best features.

Voting method – All the features are voted in each feature selection. Combining the votes with boolean values of selection using NumPy, final features are selected as votes greater than 2.

Model training:

The data split during feature selection is used here. Three different models are used for data training on the scaled data.

Feature scaling – Standard scalar is used to normalize the data, each value in the dataset will have the sample mean value of the feature subtracted. The scaling is applied on data after feature selection.

Steps followed for model training:

- Imported the desired model
- Created model instance.
- The model is fitted on Features (x_train) and labels (y_train) . The data are used to train the model, and the information discovered from the data is stored. l.
- Predict labels for new data

LinearDiscriminantAnalysis – A classifier that uses the Bayes' rule to fit class conditional densities to the data and produces a linear decision boundary model with a Gaussian basis.

MLPClassifier - Multi-layer Perceptron classifier, or MLPClassifier uses an underlying Neural Network to carry out the classification task, unlike Support Vectors or Naive Bayes Classifiers.

Parameters - solver='adam', activation='relu', alpha=0.001, shuffle =True, hidden_layer_sizes=(500,), random_state=0, max_iter=1000

hidden_layer_sizes - Number of layers and the number of nodes in the Neural Network Classifier.
Solver - Algorithm for weight optimization across the nodes.
Activation - The activation function for the hidden layers.
max_iter - It denotes the number of epochs.
LogisticRegression - Logistic regression is a statistical method that describes and estimates the relationship between target binary variables and independent variables.

Model Evaluation:
The fitted classifier is evaluated on the test dataset on various evaluation metrics as mentioned below,

- ***confusion_matrix*** – Matrix of number of observations known to be in group but predicted to be.
- ***classification_report*** - measure the quality of predictions from a classification algorithm
- ***accuracy_score*** - Intuitive metric for performance calculation but fails on skewed class distribution.
- ***f1_score*** - combines precision and recall, especially to give more attention to positives.
- ***precision_score*** *and* ***recall_score*** – True positive and false negative predictions.

	LinearDiscriminantAnalysis	MLPClassifier	LogisticRegression
accuracy	0.818760	0.822248	0.818648
f1-score	0.809840	0.813002	0.809712
precision	0.857017	0.864461	0.856949
recall	0.818760	0.822248	0.818648

As the figure shows, all the three models are performing well while MLPClassifier have slightly better performance. MLPClassifier can be applied on non-linear problems but does great on small datasets with quick predictions after training as well which helps for dynamic models during window selection.
The chosen model is then saved using “pickle.dump()” for later use in the dynamic model along with selected features.

Dynamic model:

Collect data - Consumer loop that continually invokes the poll function to get data one at a time that has been effectively pre-fetched by the consumer in the background. Uses the subscribe method to define which topics should be fetched from before starting the consumer loop. Each record is processed to extract the message value. For every 1000 clean records that are appended in the empty list, it creates a dataframe and invokes the model_training.

All steps are created as self-derived functions to perform tasks are mentioned below,
Preprocessing function - Refines the message.value to string and it is cleaned to remove unnecessary characters. The final string is stripped into a list of records.
create_dataframe - Fetches the list of records in chunk sizes of 1000 and returns a data frame. For early first record of column names is removed from the data frame to use as columns.
cleaning_data - Converts the timestamp feature into float64 type by changing timestamp to seconds.
Hash function on Mixed type columns 'sld' and 'longest_word' by applying hash on each value in the features.

Model-Training:

Step 1 - Load saved model twice as Static and Dynamic model.

Step 2 - clean the data, separating predicting and target features. Perform Feature-selection using selected features from static model.

Step 3 - Standardize features by standardscaler features scaling

Step 4 - Evaluate both models on the final data.

Step 5 - Retrain the models that met the threshold for decision.

Decay - When the sample's f1-score falls below a certain level (in this case, 0.81) and auc score to 0.88, it starts a re-train the task with current 1000 samples of records.

WHEN TO RE-TRAIN MODEL?

Usually, the re-training is considered during significant deviations from streaming observations and original training data.But in case of the dataset here, there is less data drift shown in data analysis.So, in such cases performance decay helps the model to better decide on factors to re-train the model.

F1-score and Auc precision-recall scores provide attention to positives giving more visibility on necessary factors like true positives and false negatives which are priority in cyber-attacks classification problems.

The threshold of F1-score and Auc precision-recall scores is set to 0.82 and 0.88 which are scores of the baseline in static models. Although the final threshold is selected based on trial-and-error method.

MODEL IMPLEMENTATION - COMPARISON

Evaluation metrics for each model on each window are visualized by line plots. The three graphs accuracy, f1-score and precision recall score show huge fluctuation of performance throughout the windows in the following ranges, (Figure 4) (Figure 5) (Figure 6)

	Dynamic		Static	
	lowest	highest	lowest	highest
Accuracy	0.79	0.86	0.77	0.80
F1-score	0.785	0.85	0.77	0.83
Precision recall score	0.86	0.90	0.85	0.89

It is clear from the derived table using plots that the Dynamic model has positive results, even though the model performance degrades. Dynamic models performed either equally good or better than static models.

The question of implementing a Dynamic or Static model? I believe the answer hugely depends on data distribution. Here, the provided dataset has many categorical features like upper, lower and len which have high likelihood to be continuous features in real-time. As the static model already had good performance at first, it left less room for model improvement and less data drift made it more difficult. However, considering data distribution, model performance and evaluation plots, there is only a small positive gap between static and Dynamic models.

For this problem approach, Dynamic implementation is a better choice.

Appendix

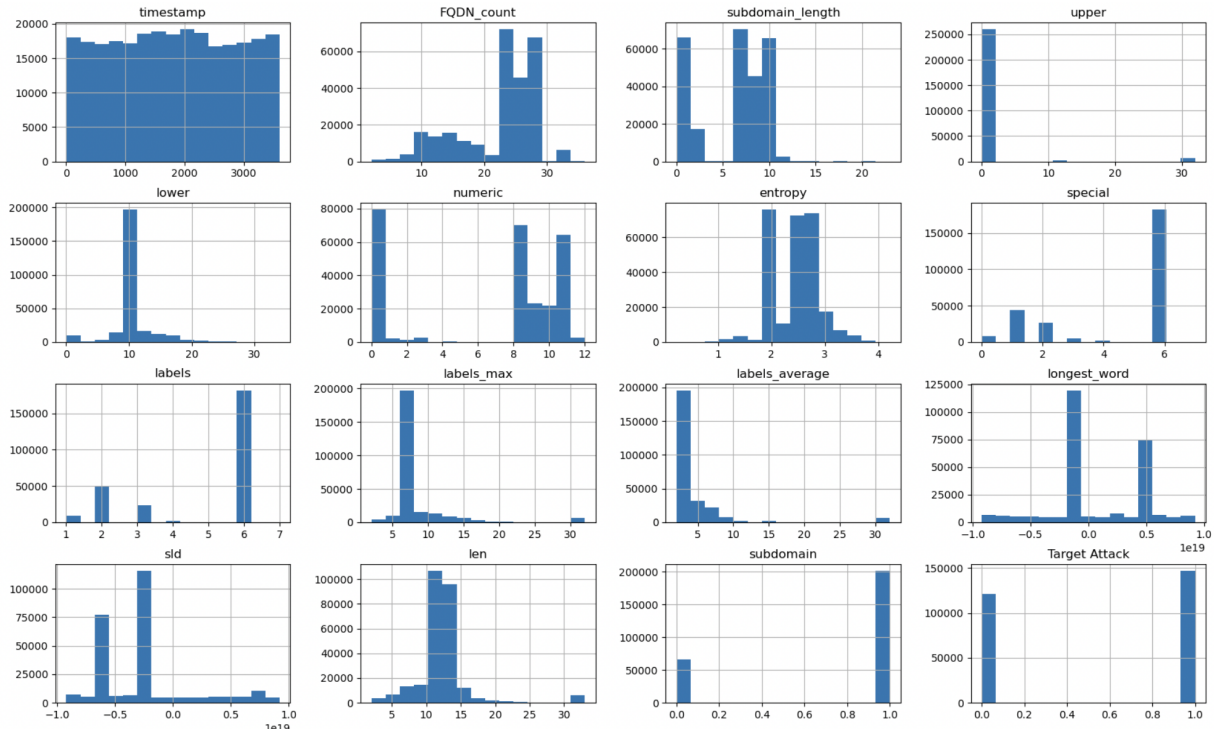

Figure 1: Histogram for all features

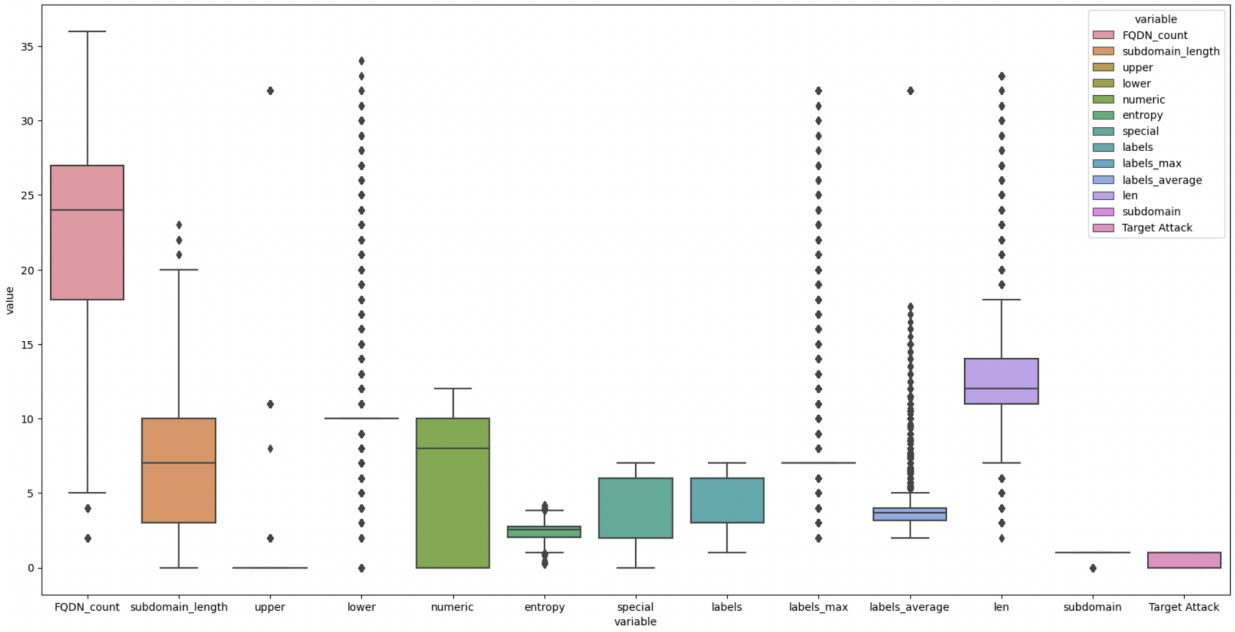

Figure 2: Box plot for outlier detection

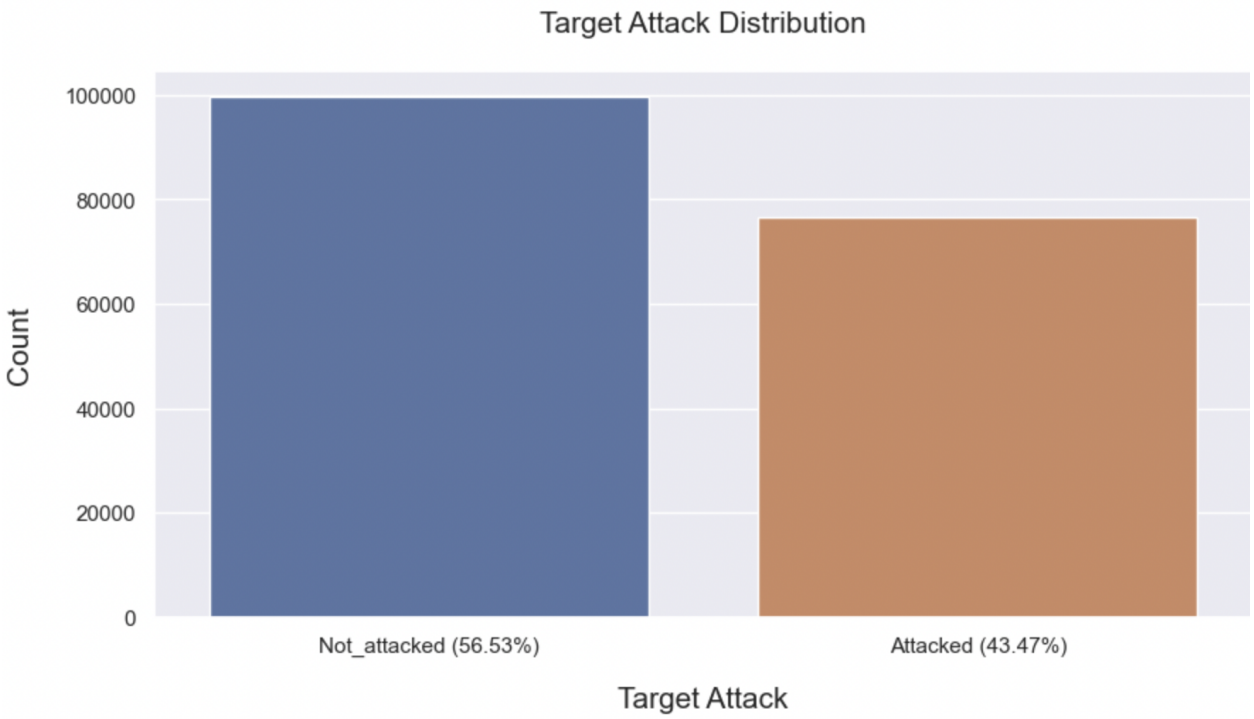

Figure 3: Target feature “Target Attacks” distribution

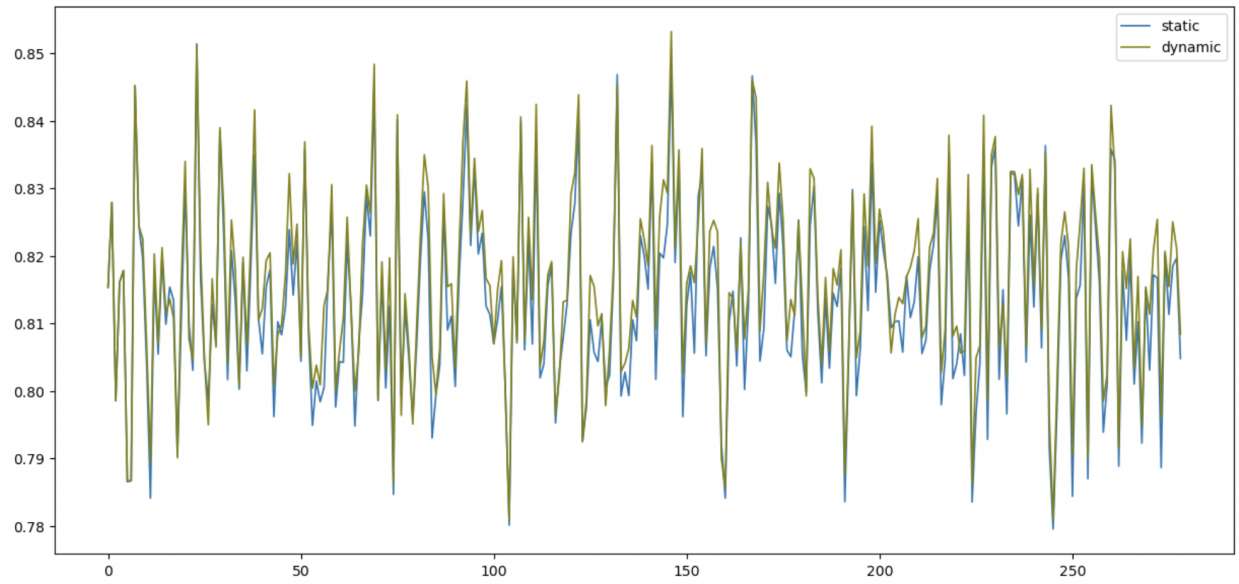

Figure 4: Plot of F1-scores for each model in each windows

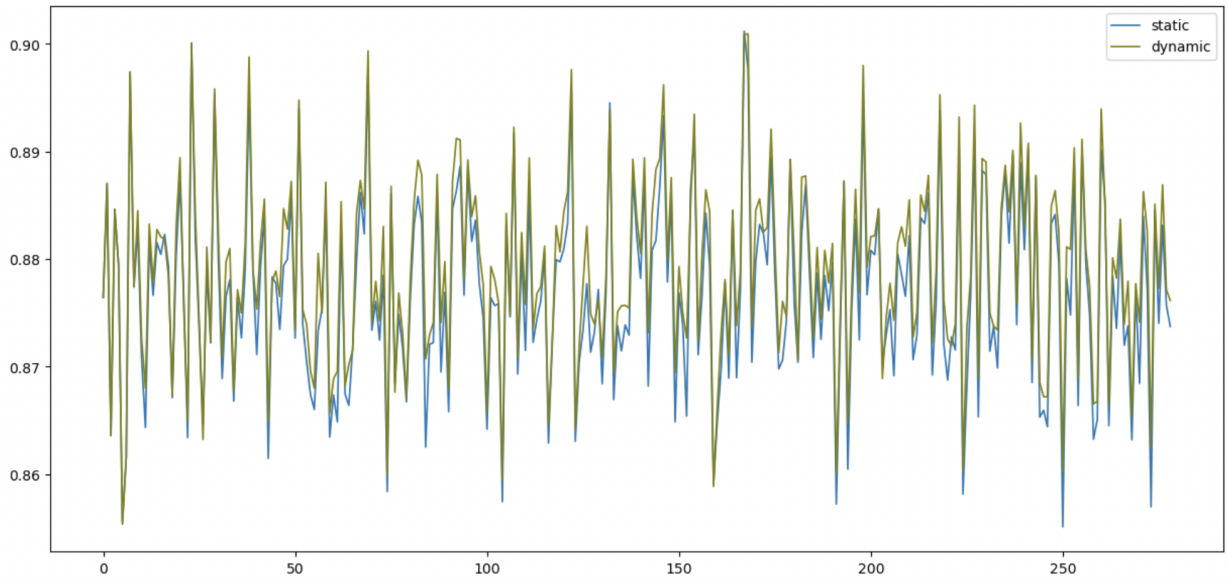

Figure 5: Plot of accuracy for each model in each windows

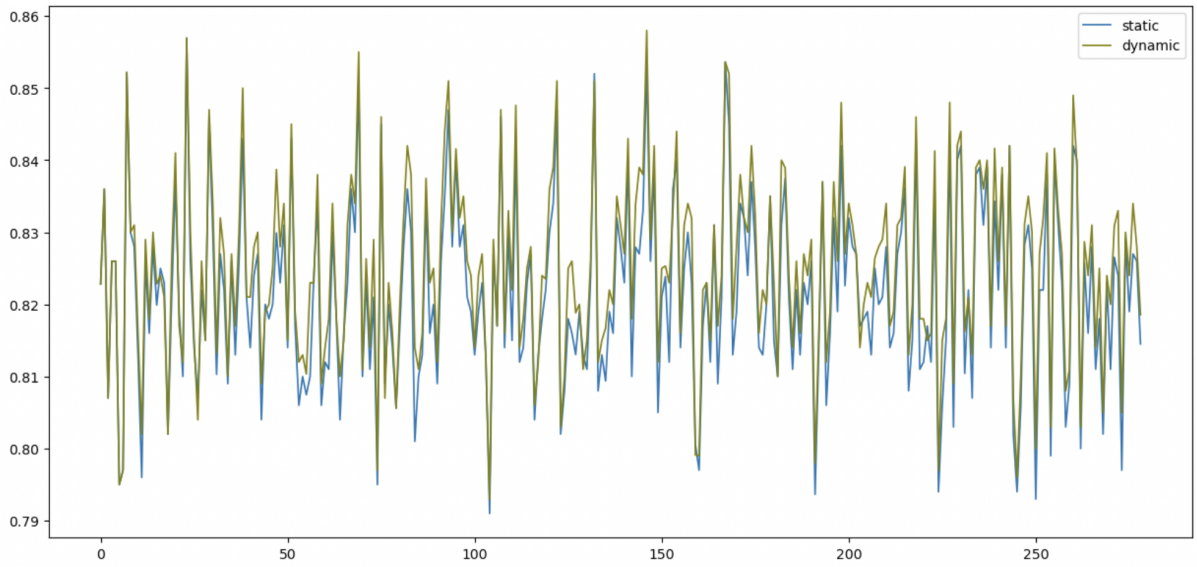

Figure 6: Plot of Precision-recall-scores for each model in each windows