AI Cyber Assignment 3

12.11.2022
Karman Singh
300235837

Overview

1. For this assignment, a random forest algorithm was selected for the first part of the problem which involved selecting a model for a static dataset. Given the nature of the conventional problem, various steps were followed to arrive at this decision.
2. This model was saved as a pickle file and imported in the file **'Dynamic_model.py'** to evaluate the same model on dynamic data.
3. The scaler and model were used in the aforementioned file with a threshold/decision training boundary of 80% accuracy, 82% accuracy and 84% with windows of 500 and 1000.
4. Initially, the dynamic and static model are the same with a pickle file of 'finalized_model.sav'. The model evaluates blocks of 1000, preprocesses it in the function **'preprocess'** which returns scaled numpy arrays namely : **new_x and new_y**
5. Preprocessing function handles the data similar to '**Static_model.ipnyb**'
6. The model's performance is evaluated. The prediction of the model is used to evaluate the f1 score and accuracy against the labels for each window of the data.
7. The steps for dynamic model are looped until the data reaches the end point which is defined by **'consumer_data_end'**

Task 1

Task 1 involved the following steps:

- Importing the dataset
- Checking null values in the dataset
- Validating data imbalance
- Statistical Analysis
- Data Cleansing and feature correlation

- Feature filtering with more than 2 methods
- Fitting Models and comparing them
- Exporting the models

Discussion

There were two key stages that involved using metrics.

1. **Feature Selection**

   Upon exploring the data, it was discovered that apart from the 'object' type values, there existed a numerical categorical data which was **'subdomain'**. The categorical data '**longest_word**' presented with a problem of **6220 different values**.

   The traditional approach such as label encoding would have created an ordinal bias in both the columns '**sld**' and **'longest_word'**. Similarly, one-hot-encoding would have created more than 6000 features.

   Initially, I could see that the top 5 features accounted for more than 70% of the value and hence I tried encoding the first 5 features and then naming the rest as 'others' but the accuracy for it was less than I got with feature hashing using the hash function in python.

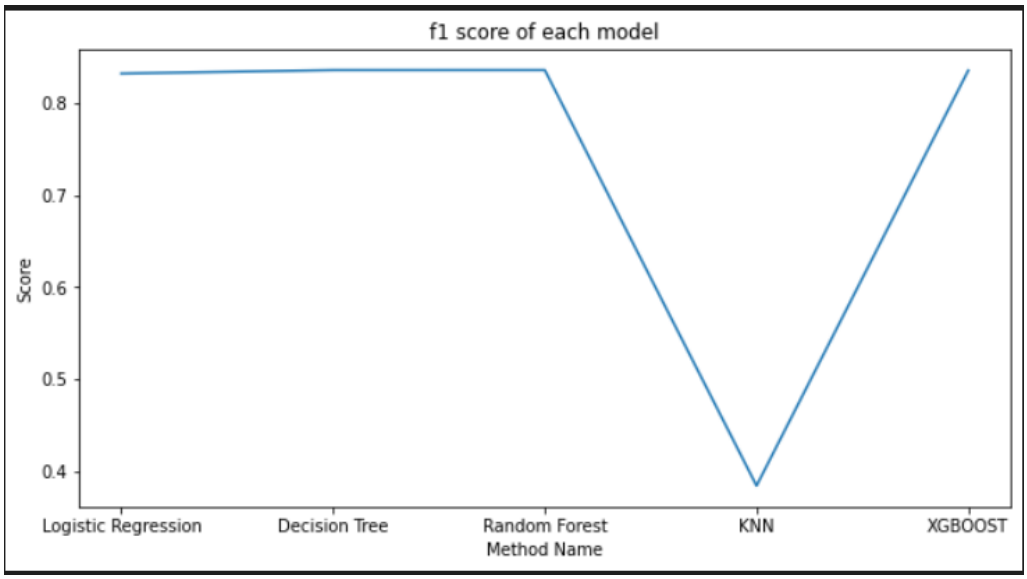
   Hence, for correlation between categorical data and labels, I used **chi square test** which returned a p-value of 0 on the contingency table of each feature.
   For the numerical values and categorical labels, I used **ANOVA** for feature correlation. Decision tree classifier further bolstered the scoring.

   Results of these models were compared and the ones with relatively lower score in each were dropped.

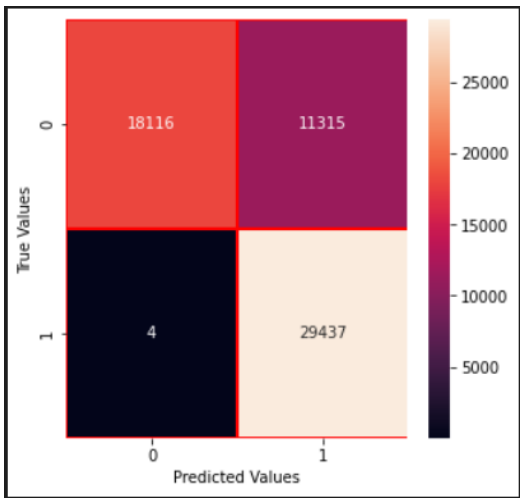
2. **Model Selection**

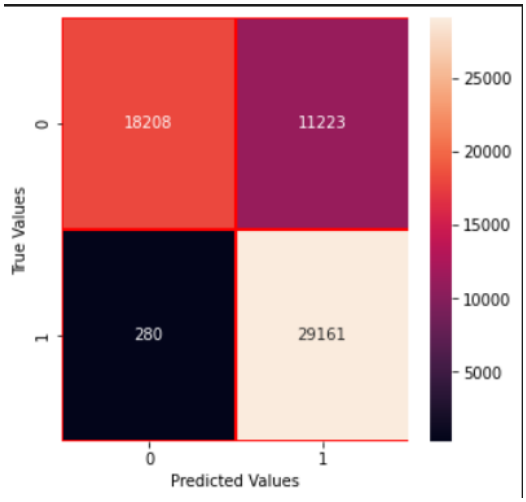
   F1 score was chosen primarily because a **higher f1 score denotes low false positives and low false negatives**. The confusion matrix gave us an insight into the score as well. It could be seen in the jupyter notebook that the tree-based algorithms performed really well on the confusion matrix even that compared to **logistic regression which had a comparable accuracy**.

   The primary reason, also mentioned in the notebook for the selection of random forest was the number of **false negative** values when compared to logistic regression. Among tree based algorithms, random forest had the highest f1 score.



Using the confusion matrix to make a decision on which algorithm is better. Kindly refer to the confusion matrix of Random Forest (L) and Logistic Regression (R)





Task 2 [Kafka]

Task 2 involved the following steps:

1. Setting up the consumer, producer and docker.
2. Processing each byte string and storing them as a dataframe
3. Dataframe processing for static model
4. Importing the model and calculating the accuracy
5. Setting a threshold based on the variance of the scores
6. Plotting the data to understand the f1 score and accuracy on both models

Discussion

The second task involved importing the pickle file of the saved model and the scaler.

1. **Preprocessing**

   This was a crucial step as it involved first converting the byte strings to readable string format and then converting it into batches of 1,000 data points that would make up the testing dataframe.

   The **preprocess** function is called after every 1000 records and the records are then processed in the same way as that in part one with hashing and feature selection alon with scaling the model with the exact same scaler.

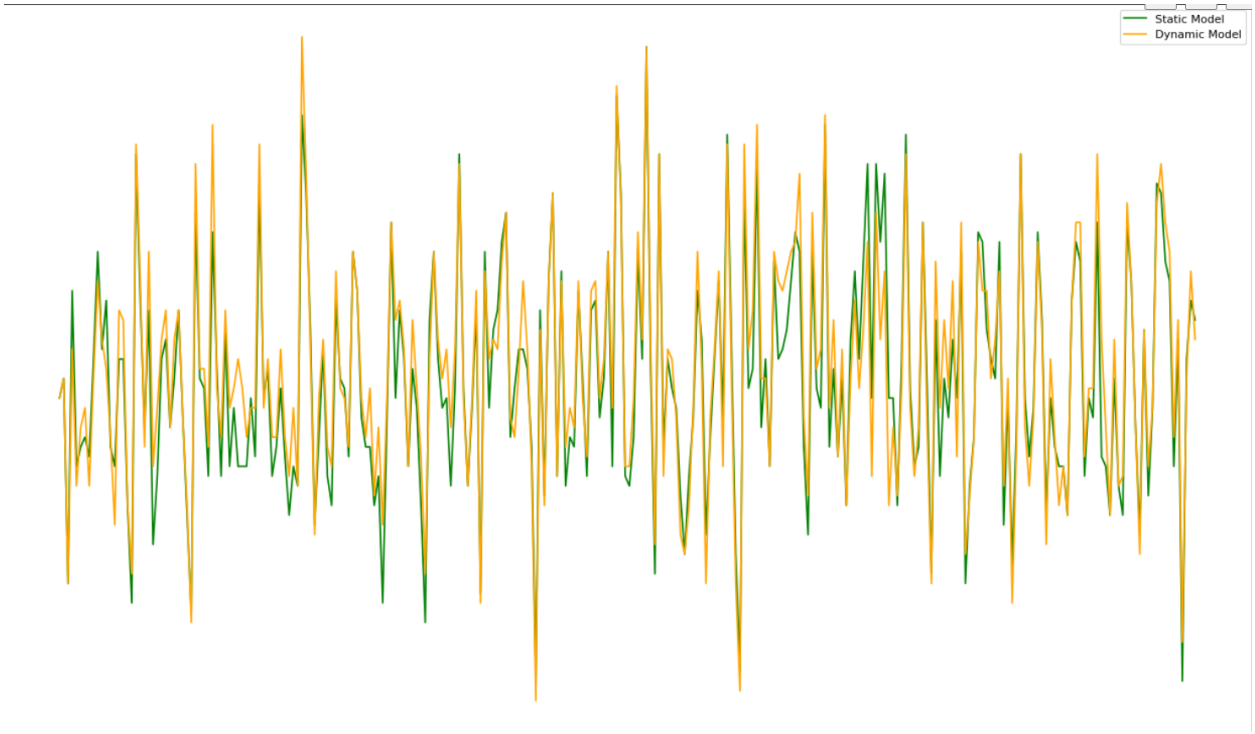
   The metrics used for evaluation were F1 score and accuracy again since those were the basis for our evaluation in the last task. Usage of which had been justified before.

2. **Model Evaluation and training**

   The model deployed for dynamic seemed to have very less difference with higher highs than lower lows. The decision boundary for evaluation with respect to the accuracy was 0.80 or 80%. This was done by measuring the calculated score without retraining initially.

Data Packets Consumed	Static Score
1000	0.810
2000	0.812
3000	0.791
4000	0.821
5000	0.803
...	...
264000	0.816
265000	0.781
266000	0.814
267000	0.820
268000	0.818

However, it can be seen in the .ipnyb file for the number of times that the model was trained that the accuracy did not increase drastically. Please refer to the image below:



This image is a testimonial of the fact that the model initialised for the static dataset was capable of classifying the data just as much as the dynamic model.

I conclude that there is scope for experiment and trying changing the window size, wherein the previous training dataset could have been added to the existing window. However for this experiment, the models performed almost the same.