

**University of Ottawa / Université d’Ottawa**

Faculty of Engineering

School of Electrical Engineering and Computer Science

**Assignment**

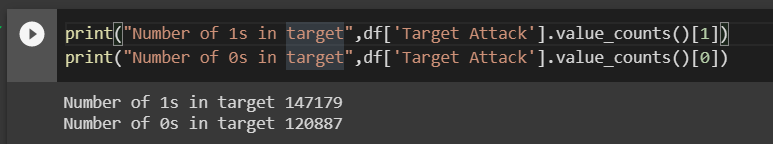
|  |  |  |  |
| --- | --- | --- | --- |
| *Course CSI5137* |  |  | AI for CyberSecurity |
| *Academic year* |  |  | 2022-23 |
| *Semester* |  |  | Fall |
| *Instructor* |  |  | Paula Branco |

|  |  |  |
| --- | --- | --- |
|  | ***Student Name: Siddhant Tiwari Student ID: 300294512*** |  |

The data provided here seems to be DNS Queries. So the following points have been adjusted for this dataset:-

**Validate data imbalanced with justifications**

The data was accepted without the use of oversampling or undersampling, and there was also no requirement for the use of data argumentation or undersampling techniques like SMOTE or subsampling. After applying value\_counts()[1] and [0], we can see that there are almost equal labels



Hence there is no need of managing data imbalances or even using hyperparameters for the same.

**Data Cleansing and Feature creation**

Since both categorical and numerical features were present in the data, only the categorical features required some data cleaning.

Firstly we can drop the time\_stamp column as it is the time\_stamp value where the dns packet was sent to the server. We have to process the data inside the query so time\_stamp is irrelevant.

Here we also removed 8 values from the "Longest word" feature because the isnull().sum() function revealed that they were missing values. Because of the size of the dataset, deleting these 8 values is the right approach as it won’t have a major impact.

After analyzing the feature set, I could establish that the longest\_word column contains either the length of the longest\_word or the longest word itself.

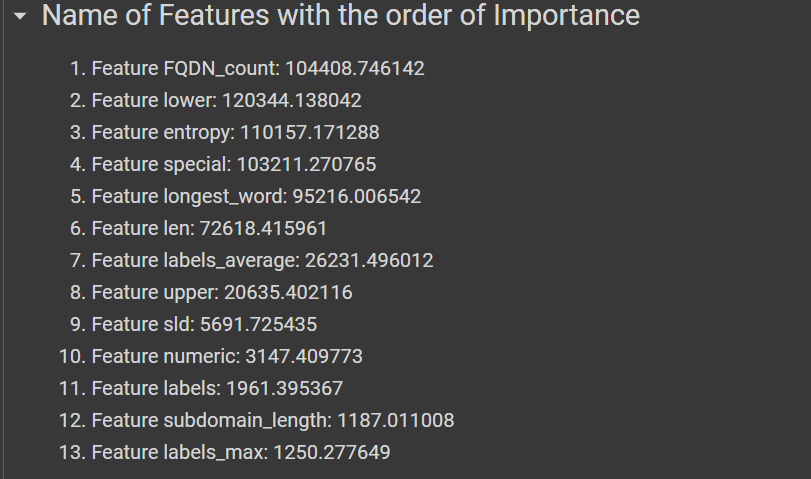
For sld we have used LabelEncoding so that we can categorize each of the subdomain names.

**Feature Filtering with more than 2 methods**

For feature selection, two distinct features selection were used. which are listed below:

1. **Anova F-Test**

The ANOVA F-test is another method of feature selection that I have employed. I have chosen eight features using the ANOVA F-test. Because there is a big difference between the eighth and ninth features, I used features up until the eighth level and discarded features after that.

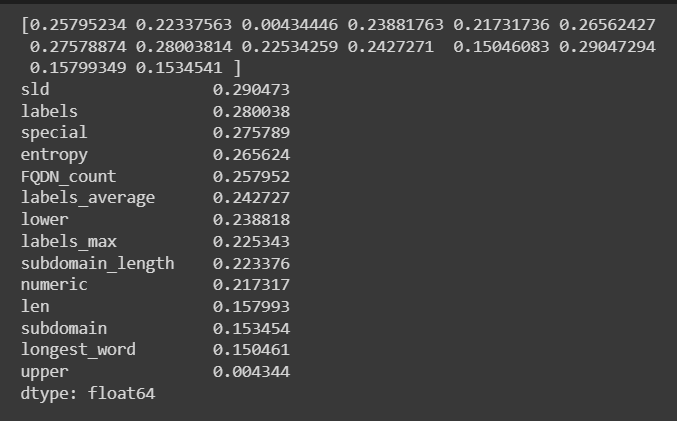


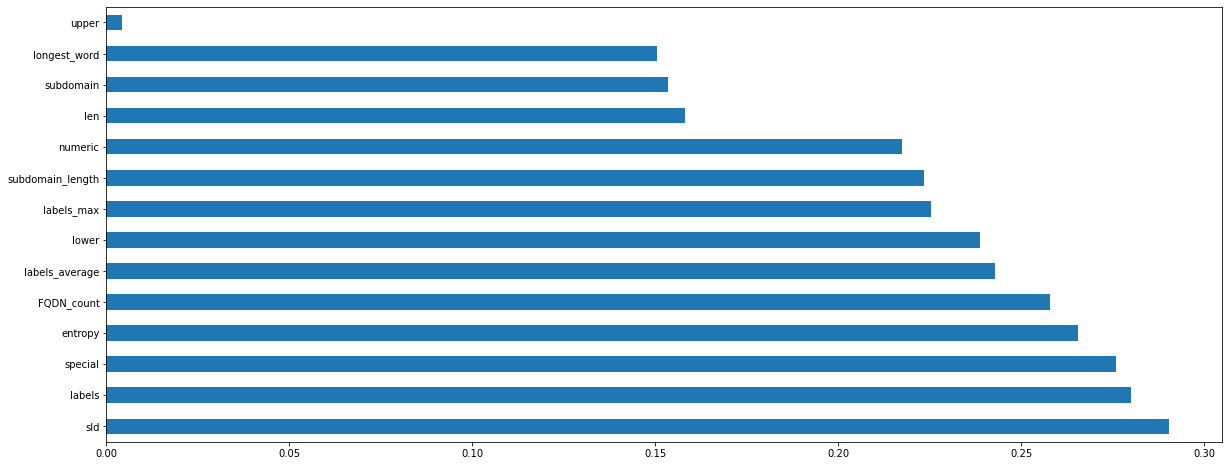
Now this shows us that Features FQDN\_count, lower, entropy, special, longest\_word, len, labels\_average and upper are the important features.

As the importance value drops by around 15,000 we shall ignore features after feature#8.

1. **Mutual Information**

Nine features were chosen using this feature selection technique because the difference between the ninth and tenth features was substantially greater than the difference between the eighth and ninth features. So it makes more sense to stop adding features after 9 Features.





**Statistical analysis of data**

**Data Splitting and justification**

The data is divided more than once, with the ultimate split being 33% for testing and 67% for training. Other ratios, like 4:1 (80% for training and 20% for testing), 9:1 (90% for training and 10% for testing), and 3:1 (75% for training and 25% for testing) are also represented in the data. After evaluating the precision for each ratio, the maximum accuracy is ultimately attained at 33% for testing and 67% for training.

**Choose and justify the correct performance metric**

When calculating Accuracy, Precision, and Recall, I discovered that Accuracy was the greatest performance statistic because Precision (True +ve Rate) was turning out very high and Recall was slightly lower, so I deleted it. Accuracy is the right measurable statistic for me simply because Dynamic model was taken into account.

**Hyperparameter tuning is correct and clear**

I then tuned the hyperparameters. I initially tried to tune the hyperparameters manually, but it takes a lot of time and may not be very effective. As a result, I utilised Grid Search CV to fine-tune the hyperparameters for both algorithms, and I saw that the accuracy increased to 82.6%. Grid Search SV is therefore implemented and employed for hyperparameter tuning.

**Compare and describe the two models you will use**

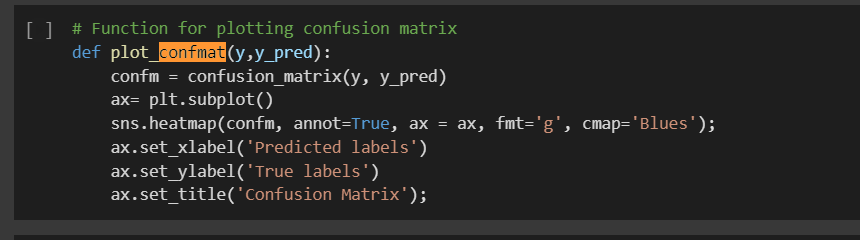
Variety of machine learning models were deployed by me, including gradient boosting, decision trees, random forests, XGBoost and CatBoost. Although all of these gave a similar accuracy, I have chosen the following two models as they were faster to train with gridSearchCV.

1. Random Forest Classifier
2. CatBoost

I initially tested the model with the standard parameter, then I manually adjusted the hyperparameters before using GridSearchCV to achieve better hyperparameter tuning.

**Plot the models’ results**

The models results have been plotted and analyzed using confusion matrix.



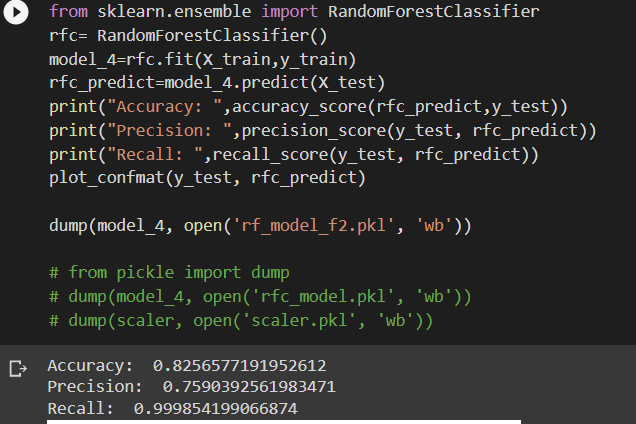
This is the code which is used to calculate the confusion matrix at all instances of model.fit().

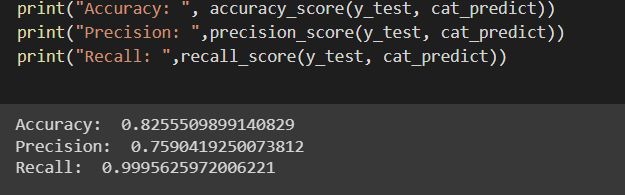
These parameters have then been used to calculate accuracy, precision and recall.

**Discuss and analyze the results**

Initially I looked at recall metric for evaluation of models as recall gives the ratio of total positives out of positives. Recall helps us minimize the attacks which could have been really attacks but were passed on as safe queries.

But as Recall metrics were around 99 for each of the models, retraining them on the dynamic kafka server model would not be possible. So, I have used accuracy as the metric to evaluate results.



****

As we can see that Random Forest gives a higher accuracy by just 0.01% thus both are equally good for the current dataset. I have gone ahead with random forest for the current dataset.

**—--------------------------------**

**Task 2**

Here, the kafka server has been used with the docker to apply the static model to kafka\_dataset.csv().

Here the dataset has been taken in windows of 1000 and then retrained if the accuracy is lesser than 0.83.

The training revaluation has been set to 0.83 as the static dataset itself had an accuracy of 82%.