The Study of Intrusion Detection in Network Security Using K-Nearest-Neighbors and KDE Naïve-Bayes Classification Algorithms

Pisai Daria
Grupa 333AA

1. Introduction

The blooming technological era brings numerous advantages, such as the ongoing discoveries in scientific fields and the breakthroughs that enhance our daily lives. However, with great performance come significant potential threats. Cybersecurity has become a requirement in all industries, as almost everything relies on the online medium.

Network intrusion detection systems are crucial in discovering potential security breaches, as they are responsible for monitoring network traffic. Their purpose comprises of identifying malicious actions and policy violations, helping in the detection of unauthorized accesses.

The aim of this essay is to highlight the importance of Machine Learning algorithms in the domain of cybersecurity. Through the utilization of K-Nearest-Neighbors and Kernel Density Estimation (KDE) Naïve-Bayes classification algorithms, the paper's end goal is the detection of malicious websites and in doing so, to help with the prevention of cybersecurity attacks.

Through the usage of classification techniques, which ensure the assignation of labels to instances based on feature values, it is possible to quickly classify network activity, to identify attack patterns and prevent intrusions.

K-Nearest Neighbors is a non-parametric algorithm that classifies data points based on a majority class among their k-nearest neighbors. It calculates the Euclidian distance between instances and assigns labels based on where the majority is placed. KNN performs well when decision boundaries are complex. However, it is disadvantageous in this case to work with large datasets, as KNN is computationally expensive.

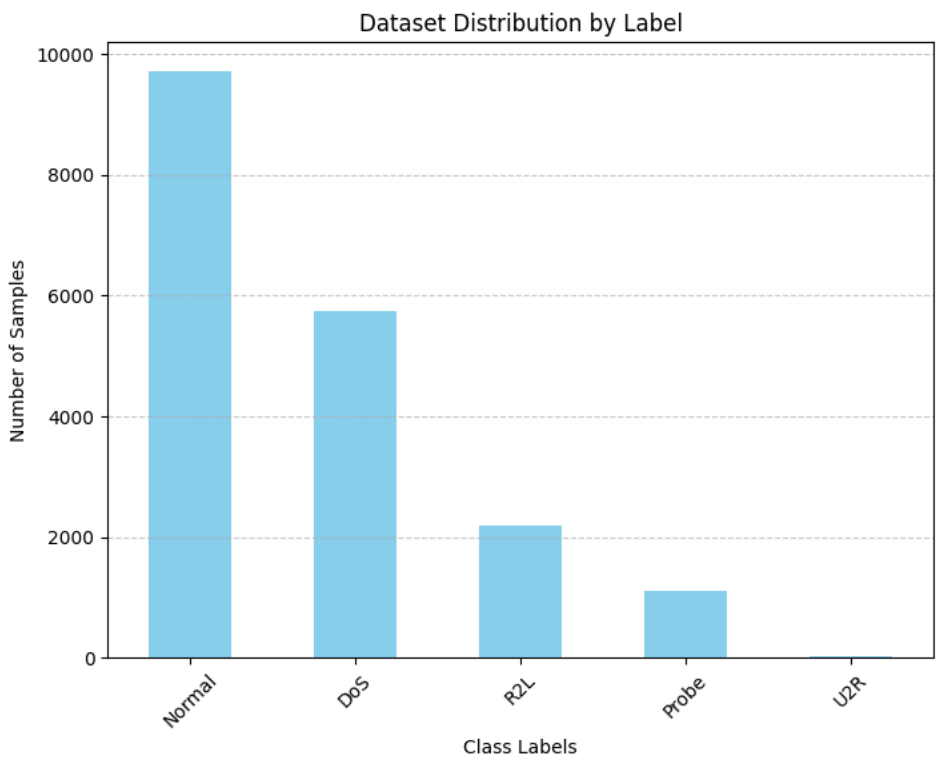
Traditional Naive Bayes assumes feature independence and follows a Gaussian distribution for continuous features. However, network attack data can have complex, non-Gaussian distributions. KDE-based Naive Bayes estimates probability densities using kernel functions, allowing a more flexible distribution fitting. In this study, the kernel that was used is a Gaussian kernel. The difference between the Gaussian Naïve-Bayes and KDE Naïve-Bayes that uses a Gaussian kernel is that the latter is recommended for non-Gaussian distributions, as it is more effective.

2. Methodology

In order to compare the two algorithms, metrics such as accuracy, precision, the F1 score and recall were used, as well as the two confusion matrixes produced by both algorithms.

This study uses the NSL-KDD dataset, which divides the data into two categories: normal network traffic and attacks targeting a network system. The data has been further divided into 5 classes: Normal, DoS (Denial of Service), Probe, R2L (Remote to Local) and U2R (User to Root). These 5 classes represent the five labels that the algorithms used (the y component) and the features are represented by the 41 characteristics for each data instance (the X component). The dataset contains 19000 values, that are split into a training set (60%), into a validation set (20%) and into a testing set (20%).

As observed in the distribution diagram down below, the dataset is highly uneven. Therefore, the data available for U2R should not be used in classification.



In this study, a preprocessing method that was used is data cleaning, followed by label mapping. This implies that the dataset goes through a filtering process where the invalid and missing labels are refined. Through label mapping, the models focus on classifying general attack patterns.

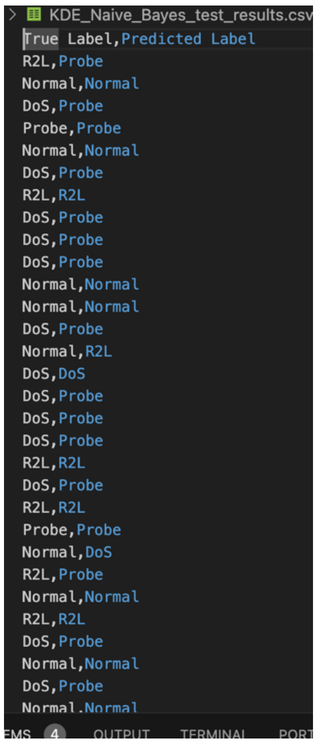
Other preprocessing procedures include feature encoding, which uses categorical features (“protocol_type”, “service”) and transforms them into numeric values and normalization, which ensures that all features contribute equally to distance calculations (in the case of KNN).

The implementation of the code consists of creating the KDE Naïve-Bayes classifier, of training and evaluating each model independently on the dataset.

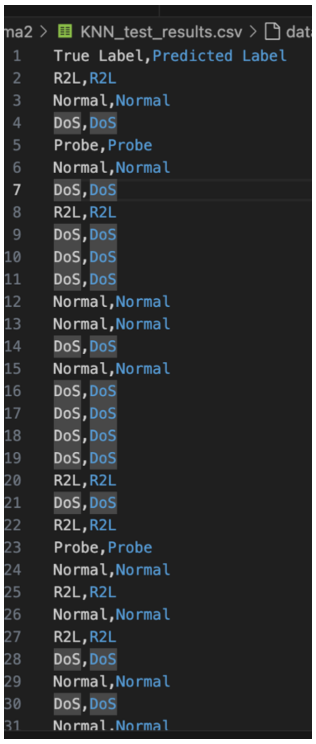
To create the KDE Naïve-Bayes classifier, I used the Principal Component Analysis to reduce the dimensionality of the feature space and then I extracted empty class labels and initialized empty KDE models. For each class, all training samples were selected and each feature was fitted with a Gaussian KDE model. Using the same PCA, the test samples were transformed and the "predict" method would calculate the probability of each feature given the class using KDE. In the end, the class with the highest computed probability would be the predicted label.

The KNN classifier was trained using 5 nearest neighbors (k = 5).

Another functionality that the code implements is to save the results of the training, validation and testing stages from both algorithms in 6 different VCS files. Therefore, after both models pass all three stages, in each of the VCS file there will be a column containing the True Label and a column containing the Predicted Label.



```
True Label,Predicted Label
R2L,Probe
Normal,Normal
DoS,Probe
Probe,Probe
Normal,Normal
DoS,Probe
R2L,R2L
DoS,Probe
DoS,Probe
DoS,Probe
Normal,Normal
Normal,Normal
DoS,Probe
Normal,R2L
DoS,DoS
DoS,Probe
DoS,Probe
DoS,Probe
R2L,R2L
DoS,Probe
R2L,R2L
Probe,Probe
Normal,DoS
R2L,Probe
Normal,Normal
R2L,R2L
DoS,Probe
Normal,Normal
DoS,Probe
Normal,Normal
```



```
True Label,Predicted Label
R2L,R2L
Normal,Normal
DoS,DoS
Probe,Probe
Normal,Normal
DoS,DoS
R2L,R2L
DoS,DoS
DoS,DoS
DoS,DoS
Normal,Normal
Normal,Normal
DoS,DoS
Normal,Normal
DoS,DoS
DoS,DoS
DoS,DoS
DoS,DoS
R2L,R2L
DoS,DoS
R2L,R2L
Probe,Probe
Normal,Normal
R2L,R2L
Normal,Normal
R2L,R2L
DoS,DoS
Normal,Normal
DoS,DoS
Normal,Normal
```

3. Results

The metrics that were calculated within this study are accuracy, precision, recall and F1-score. Accuracy measures overall correctness, but it does not always reflect balanced classification. Precision is used to indicate how many instances were predicted correctly as belonging to a certain class. Recall measures the ability of the model to correctly identify positive cases, leading to the reduction of false negatives. The F1-score represents the harmonic mean of precision and recall, providing a balanced evaluation.

In the case of a confusion matrix with 5 labels, the True Positives (TP) represent the correctly classified instances of the class that can be observed on the diagonal. False Positives are instances that are wrongly classified as a certain class, False Negatives are instances that were supposed to be in a class but were misclassified and True Negatives are correctly predicted instances that do not belong to the class. Below are the formulae of the 4 indicators for all the confusion matrixes in the paper.

$$TP_C = \; ConfusionMatrix[C, C]$$

$$FP_C = \sum_{i \neq C} ConfusionMatrix[i, C]$$

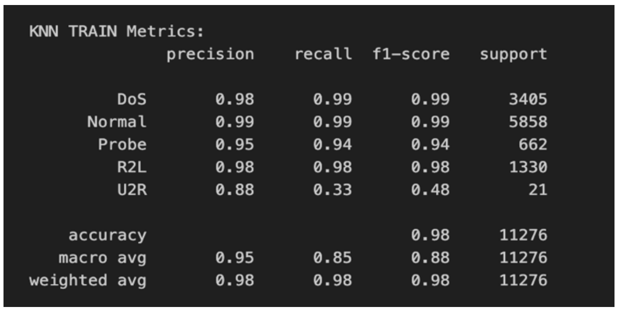
$$FN_C = \sum_{j \neq C} ConfusionMatrix[C, j]$$

$$TN_C = \sum_{i \neq C} \sum_{j \neq C} ConfusionMatrix[i, j]$$

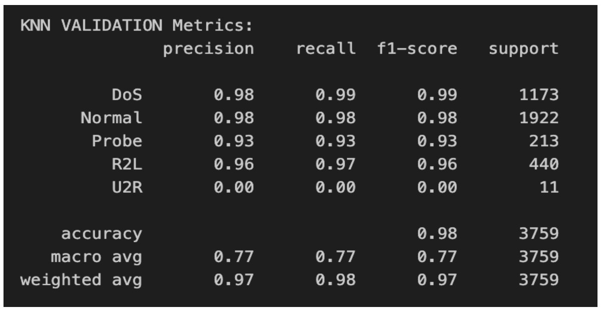
3.1 KNN Performance Analysis

In the train set's case, the overall accuracy was 98%. DoS, Normal Traffic, Probe and R2L had an almost perfect precision recall, which means that they were predicted very well by the model. In the case of U2R, although the precision value is high (0.88), the recall value is low (0.33), which indicates that U2R attacks were misclassified.

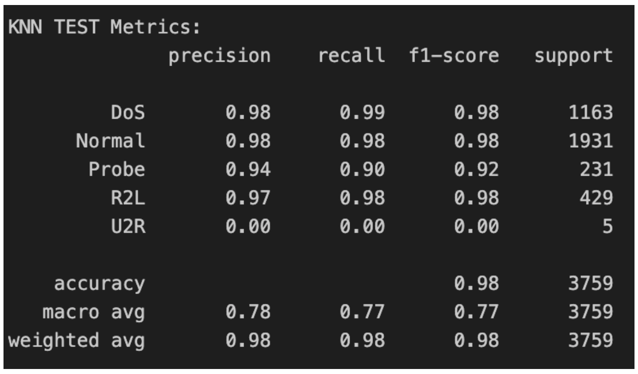
In the case of the validation and test sets, the predictions are almost the same, with a slight drop in precision for Probe and R2L (0.9-0.97). U2R detection is however null, with a 0% recall. The U2R attacks are being misclassified because they represent a very small percentage of the sample.



```
KNN TRAIN Metrics:
              precision    recall  f1-score   support

         DoS       0.98      0.99      0.99      3405
      Normal       0.99      0.99      0.99      5858
       Probe       0.95      0.94      0.94       662
         R2L       0.98      0.98      0.98      1330
         U2R       0.88      0.33      0.48        21

    accuracy                           0.98     11276
   macro avg       0.95      0.85      0.88     11276
weighted avg       0.98      0.98      0.98     11276
```

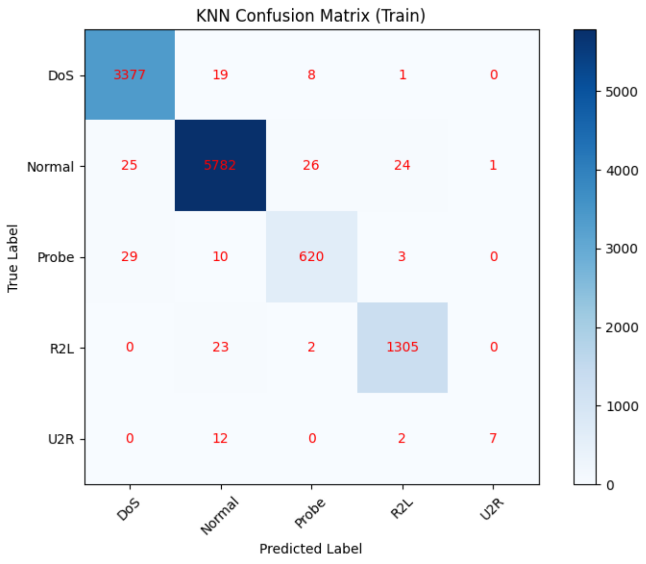


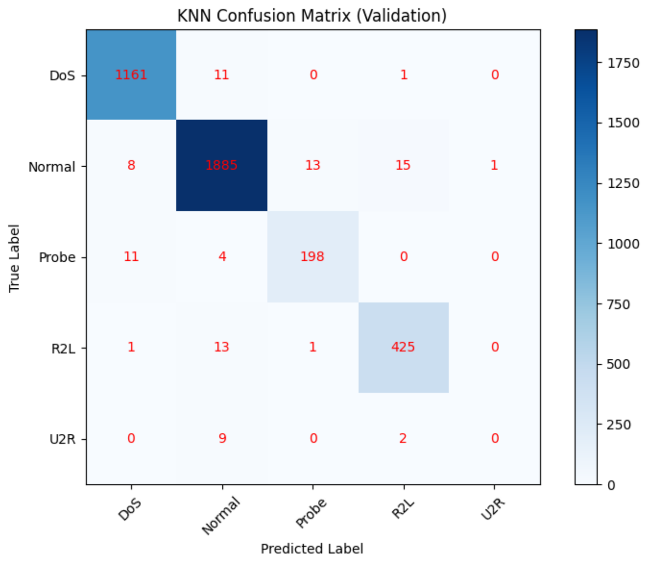
```
KNN VALIDATION Metrics:
              precision    recall  f1-score   support

         DoS       0.98      0.99      0.99      1173
      Normal       0.98      0.98      0.98      1922
       Probe       0.93      0.93      0.93       213
         R2L       0.96      0.97      0.96       440
         U2R       0.00      0.00      0.00        11

    accuracy                           0.98      3759
   macro avg       0.77      0.77      0.77      3759
weighted avg       0.97      0.98      0.97      3759
```

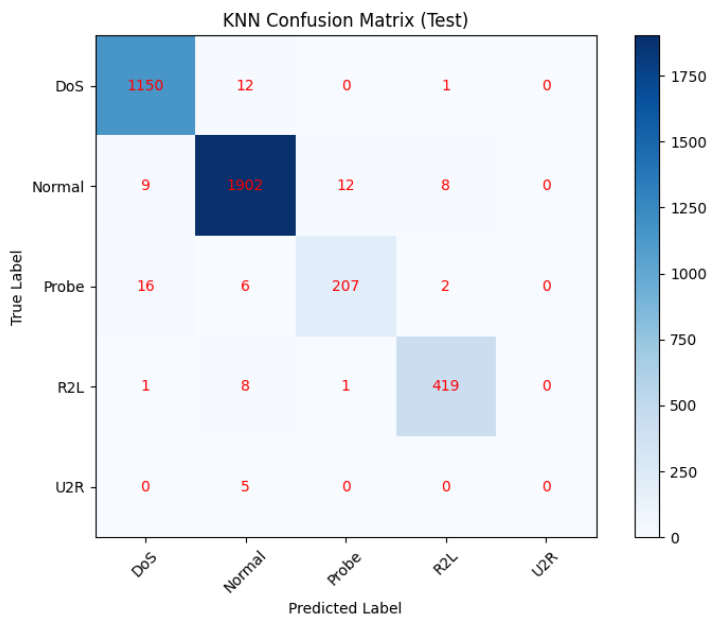


```
KNN TEST Metrics:
              precision    recall  f1-score   support

         DoS       0.98      0.99      0.98      1163
      Normal       0.98      0.98      0.98      1931
       Probe       0.94      0.90      0.92       231
         R2L       0.97      0.98      0.98       429
         U2R       0.00      0.00      0.00         5

    accuracy                           0.98      3759
   macro avg       0.78      0.77      0.77      3759
weighted avg       0.98      0.98      0.98      3759
```

The confusion matrix is also a good classification indicator, and in the case of KNN, as it can be observed in the picture, most samples are correctly classified (strong diagonal presence). Misclassification occurs in minor cases, such as U2R and some of the R2L cases.







True Positive, True Negative, False Positive and False Negative values calculated for one of the labels, Denial of Service:

$$\textbf{TP} = 3377$$

$$\textbf{FP} = 25 + 29 + 0 + 0 = 54$$

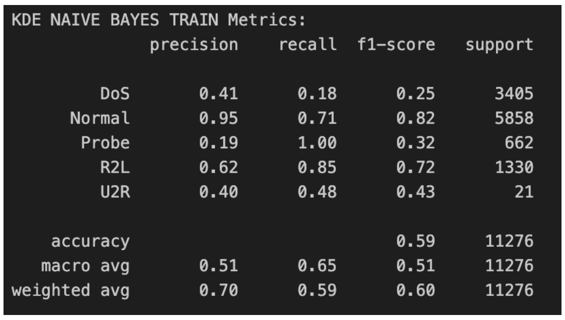
$$\textbf{FN} = 19 + 8 + 1 + 0 = 28$$

$$\textbf{TN} = 5782 + 620 + 1305 + 7 = 7714$$

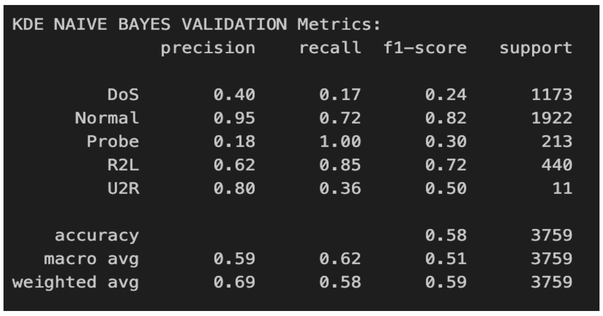
3.2 KDE Naïve-Bayes Performance Analysis

The overall accuracy for the train set is 59%, which indicates a significant difference between the algorithms' performances. Normal Traffic and R2L have an above-average recall score (0.71-0.85), but precision varies therefore, there are some inconsistencies. The Probe class has a perfect recall (1.00), but low precision (0.19), indicating that the Probe is correctly classified, but other attacks are classified as Probe as well. DoS has a recall score equal to 0.18, and U2R is also inconsistent, with a recall score of 0.48 and a precision score of 0.4.

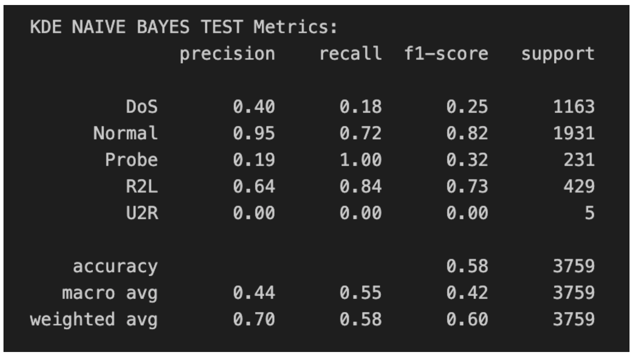
For the validation and test sets the accuracy remains 0.58, which confirms a weak generalization capability. Probe and R2L have a high recall, so they are well-classified, while DoS and U2R are poorly classified.



```
KDE NAIVE BAYES TRAIN Metrics:
              precision    recall  f1-score   support

         DoS       0.41      0.18      0.25      3405
      Normal       0.95      0.71      0.82      5858
       Probe       0.19      1.00      0.32       662
         R2L       0.62      0.85      0.72      1330
         U2R       0.40      0.48      0.43        21

    accuracy                           0.59     11276
   macro avg       0.51      0.65      0.51     11276
weighted avg       0.70      0.59      0.60     11276
```

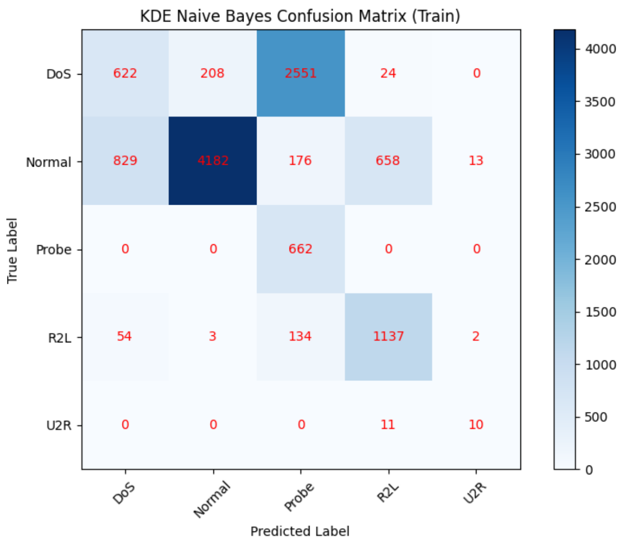


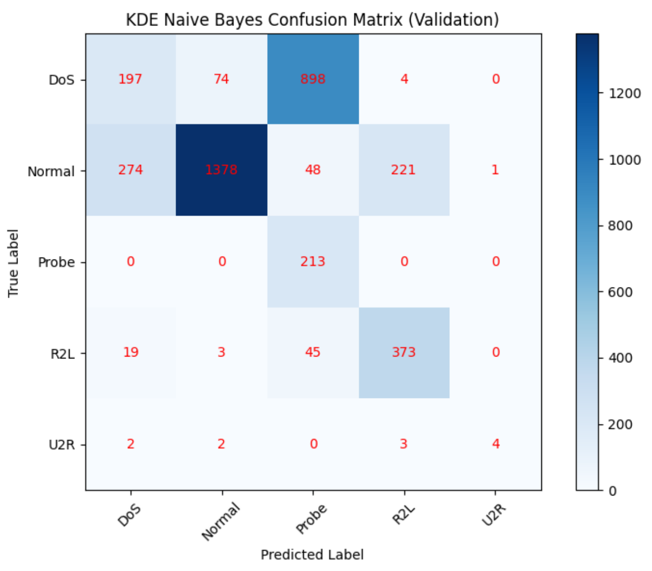
```
KDE NAIVE BAYES VALIDATION Metrics:
              precision    recall  f1-score   support

         DoS       0.40      0.17      0.24      1173
      Normal       0.95      0.72      0.82      1922
       Probe       0.18      1.00      0.30       213
         R2L       0.62      0.85      0.72       440
         U2R       0.80      0.36      0.50        11

    accuracy                           0.58      3759
   macro avg       0.59      0.62      0.51      3759
weighted avg       0.69      0.58      0.59      3759
```

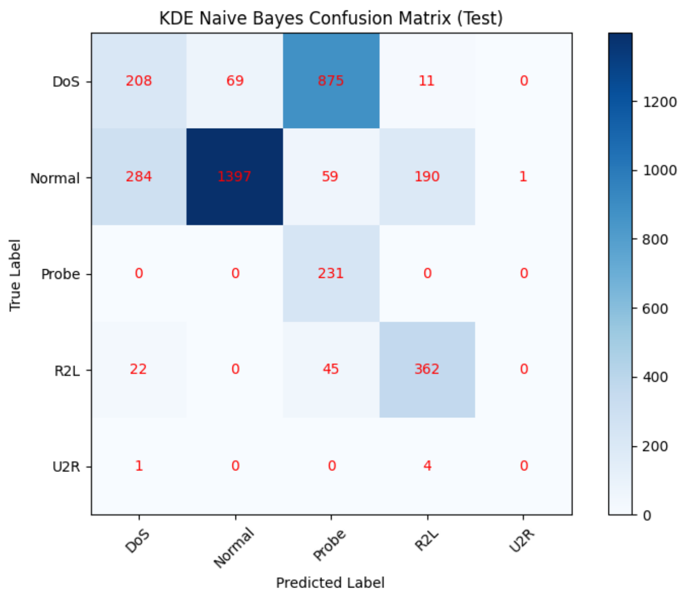


```
KDE NAIVE BAYES TEST Metrics:
              precision    recall  f1-score   support

         DoS       0.40      0.18      0.25      1163
      Normal       0.95      0.72      0.82      1931
       Probe       0.19      1.00      0.32       231
         R2L       0.64      0.84      0.73       429
         U2R       0.00      0.00      0.00         5

    accuracy                           0.58      3759
   macro avg       0.44      0.55      0.42      3759
weighted avg       0.70      0.58      0.60      3759
```

The confusion matrix shows significant misclassification in the case of the KDE Naïve-Bayes classification algorithm. It displays that there is an error in the recognition between the DoS and the Normal Traffic classes, and as for the Probe class, although it has a high recall, it also has high false positives. The R2L class is detected well, but it still suffers misclassification and U2R is very rarely categorized correctly.







True Positive, True Negative, False Positive and False Negative values calculated for one of the labels, Denial of Service:

TP(DoS) = 208

FP(DoS) = 284 + 0 + 22 + 1 = 307

FN(DoS) = 69 + 875 + 11 + 0 = 955

TN(DoS) = 2,289

4. Conclusions

The study has presented the comparison of two classification algorithms, K-Nearest Neighbors and Kernel Density Estimation Naïve-Bayes. This comparison has been made in regard to network intrusion detection through the usage of the NSL-KDD dataset.

The experimental results have demonstrated that for the sampled dataset and with the 5 chosen labels, KNN outperforms KDE Naïve-Bayes through the achievement of higher accuracy and through more consistent precision and recall metrics across the training, validation and testing stages.

Both models have difficulty to correctly identify U2R attacks, which was predicted at the beginning because of the very small sample that could be found in the dataset.

The KNN model surpasses the KDE Naïve-Bayes model in terms of classification, while also presenting better compatibility with larger datasets.

Although the KDE Naïve-Bayes model offers a more flexible probability estimation for non-Gaussian distributions, in this particular study, it displays weak accuracy in differentiating DoS and U2R attacks.

These findings suggest that data distribution can highly influence how the model perceives the dataset and when a significant class imbalance is present, the inaccurate results may not be only because of the model.