Assignment 2 - Natural Computing

Jasper Pieterse (s1017897)
Daria Mihaila (s1124590)
Myrthe Deen (s1051792)

February 2024

1 Introduction

In this report, we applied the negative selection algorithm to anomaly detection in sequence data. The negative selection algorithm is a works as a one-class classification algorithm. It is trained on unlabelled data sampled from a certain sub-region of the problem domain, and then used to determine whether or not new unseen data-points belong to the same sub-region. It's main conceptual working is detailed in pseudo-code Algorithm 1.

Algorithm 1: Pseudocode for Negative-Selection Algorithm [1]

```
Data: A set S ⊂ U ("self-set"); a set M ⊂ U ("monitor set"); an integer n
Result: For each element m ∈ M, either "self" or "non-self".
D ← empty set;
while |D| < n do
    d ← random detector;
    if d does not match any element of S then
        insert d into D;
for each m ∈ M do
    if m matches any detector d ∈ D then
        output "m is non-self" (an anomaly);
    else
        output "m is self";
```

2 Methods

We made use of the provided negative selection algorithm [2]. To gather the data and post-process, we used Python and Numpy and Matplotlib. We evaluated performance of the algorithm using a Receiver Operating Curve (ROC) visualization and the Area Under the Curve (AUC) metric. We outputted the logarithms of the amount of detectors (i.e. -c and -l enabled in training).

For both experiments, we performed multiple runs with different values of the chunk length n and the value of the r-contiguous counter r. For the second experiment, we preprocessed the sequences by to a set of non-overlapping fixed-length chunks n. When a chunk belonging to a certain system-call was shorter than the desired chunk length, we filled the chunk up with '-' signs. This character is the alphabet given to the negsel algorithm and thus is considered safe and won't affect overall results. The results were obtained for test set 1, but the code is modular to be used with other test sets as well for those who are interested.

We labelled each chunk with an system-call id. At time of inference, all chunks belonging to the same system-call id were evaluated and the mean value over all chunks was taken as the anomaly score for that particular line. We chose the mean over all chunks because it measures the overall average anomaly of a process. An other option would be to take the maximum value over all chunks for that line, however we believe that if a single chunk is anomalous, the whole system call should not be labelled anomalous. The code used can be found here.

3 First Experiment

We first trained the negative selection algorithm on the English language. We ranged the parameter $r = \{1, \ldots, 9\}$ and computed AUC $= 0.79$ for $n = 10$, $r = 4$. The specificity curves for varying r are shown Figure 1. We obtained the overall best discrimination for $n = 10$ and $r = 3$ with AUC $= 0.83$. We used these parameters for our initial guesses for the next experiment from part 3.) for different languages. Our results are consistent with the Figures of the assignment, corresponding to $r = [1, 7, 4]$.

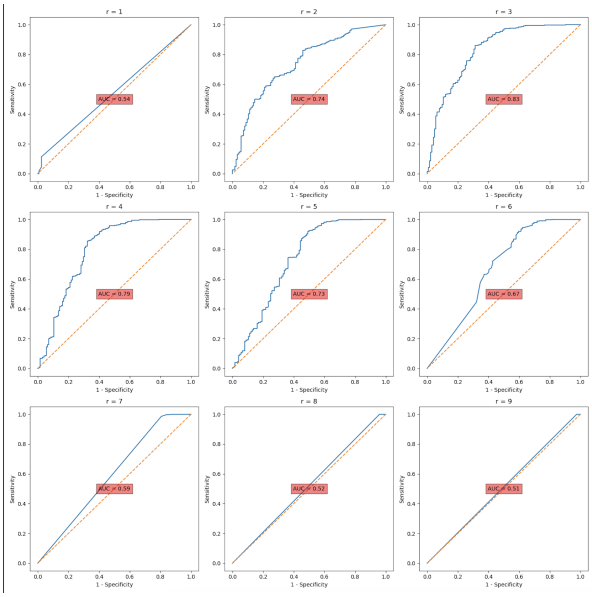

Figure 1: Figure showing 9 subplots, each obtained from the same $n = 10$ but with varying r values from 1 to 10, showing the specificity curve and the value of the AUC in each case.

Table 1 summarises the found AUC values for increasing r, a trend which is further visualised Figure 2, highlighting the trend of this variation.

r	1	2	3	4	5	6	7	8	9
AUC	0.54	0.74	0.83	0.79	0.73	0.67	0.59	0.52	0.51

Table 1: Table showing the variation in AUC with values of r

We can explain the trend of the curve in terms of the optimal length of the strings for learning and testing. Firstly, when thinking about length of words for finding patterns in a language, a single letter (r = 1) is too little to understand the 'stem' of words or form patters and larger strings (r = 9 or 10) is far too large for learning even most single words, which would be shorter than that amount of letters. Secondly, the learning was done with an r = 4, thus, we can expect that the best performance was also found for a similarly valued r. Therefore, an in-between value, r = 3, as found in our study here, will be ideal for training and testing of languages.

Performing the same training but testing on 4 different languages this time, and setting r = 3 for testing (as it previously got the best result), we obtained the curves shown in Figure 3. The model performed best on the Xhosa test set and worst on the Middle-English test set. This results makes sense, because Middle-English uses a structure (i.e. combinations of letters) that are very similar to modern English. Xhosa, on the other hand, uses sequences that never occur in English. Because of this, it is more likely that a detector will match Xhosa then Middle-English, resulting in a better score for Xhosa.

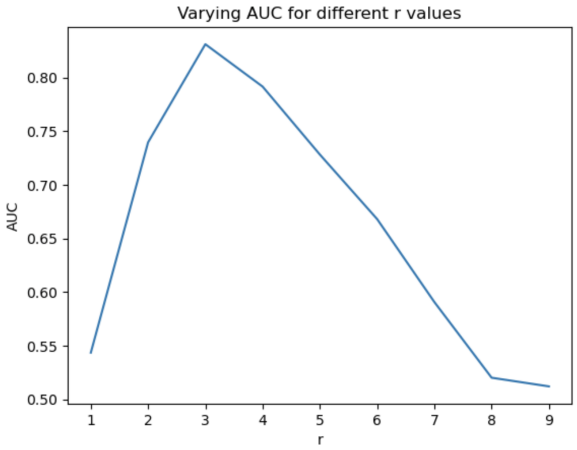

Figure 2: Figure showing the trend in AUC values for increasing r

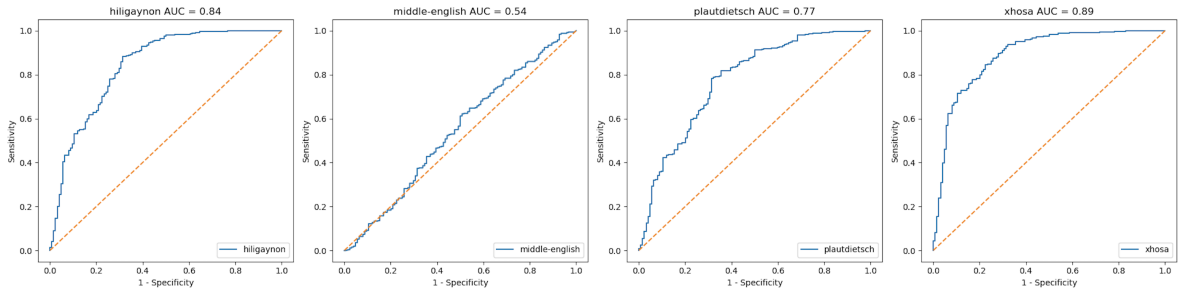

Figure 3: Figure showing the different specificity curves and AUC values as we test our learning on the english.train file on different languages

4 Second Experiment

The negative selection algorithm demonstrates overall strong performance, as depicted in Figure 5. In general, larger values of n and mid-range r values (around $n/2$) appear to yield the best results. The algorithm falters only when $r = 1$, which aligns with expectations. Notably, it exhibits a rapid recovery with increasing r values, as illustrated in Figure 4. From these results, we conclude that the negative selection algorithm can be used effectively for anomaly detection in Unix processes.

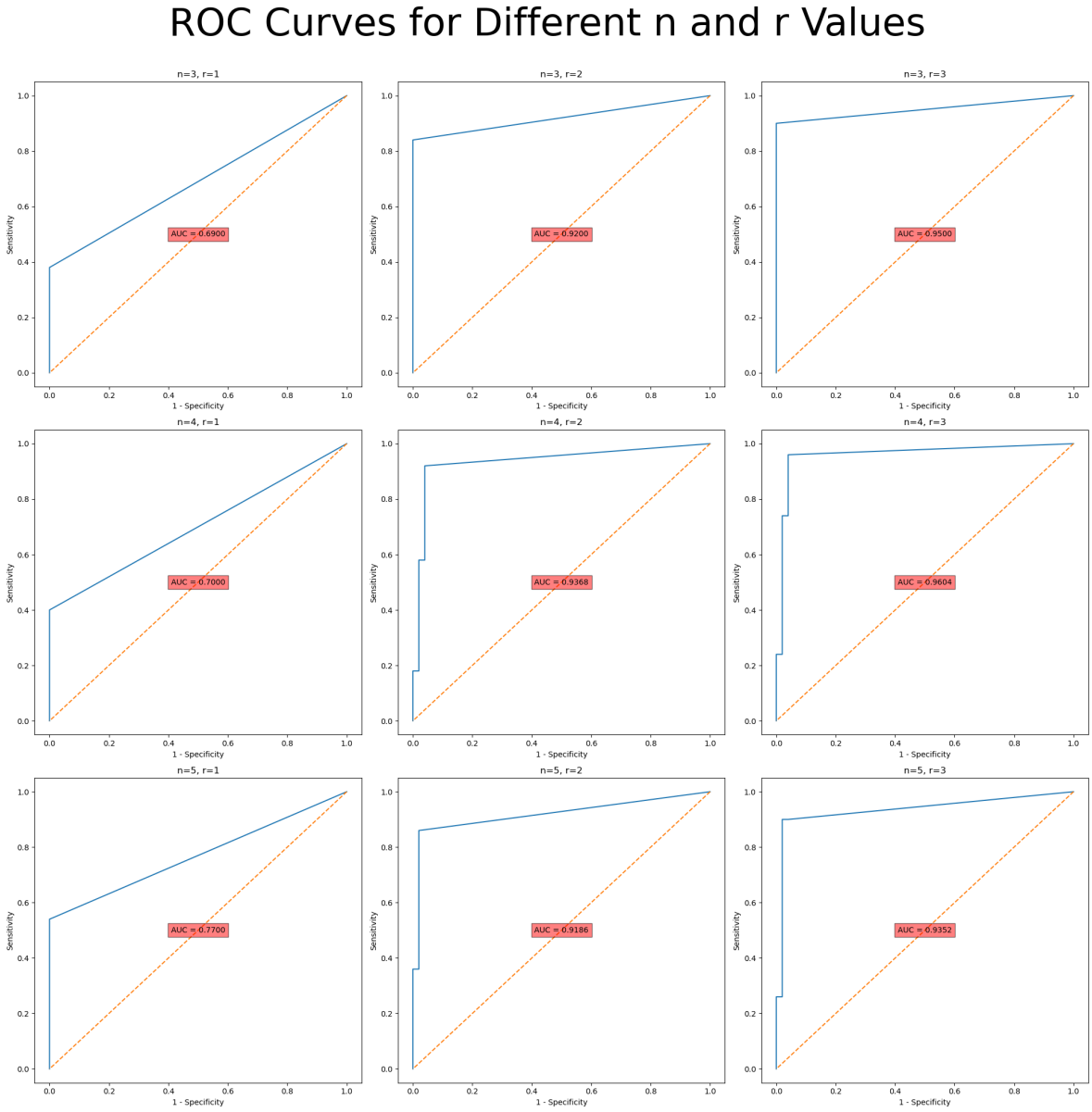

Figure 4: ROC curves and AUC scores resulting from a grid search over $n = [3, 4, 5]$ and $r = [1, 2, 3]$. Notably, the negative selection algorithm exhibits suboptimal performance only at extreme parameter values ($r = 1$). With an increase in this parameter, a rapid improvement in performance is observed. The chunk size n appears to have a less pronounced effect on performance within this parameter range. The optimal values are found at $n = 4$ and $r = 3$.

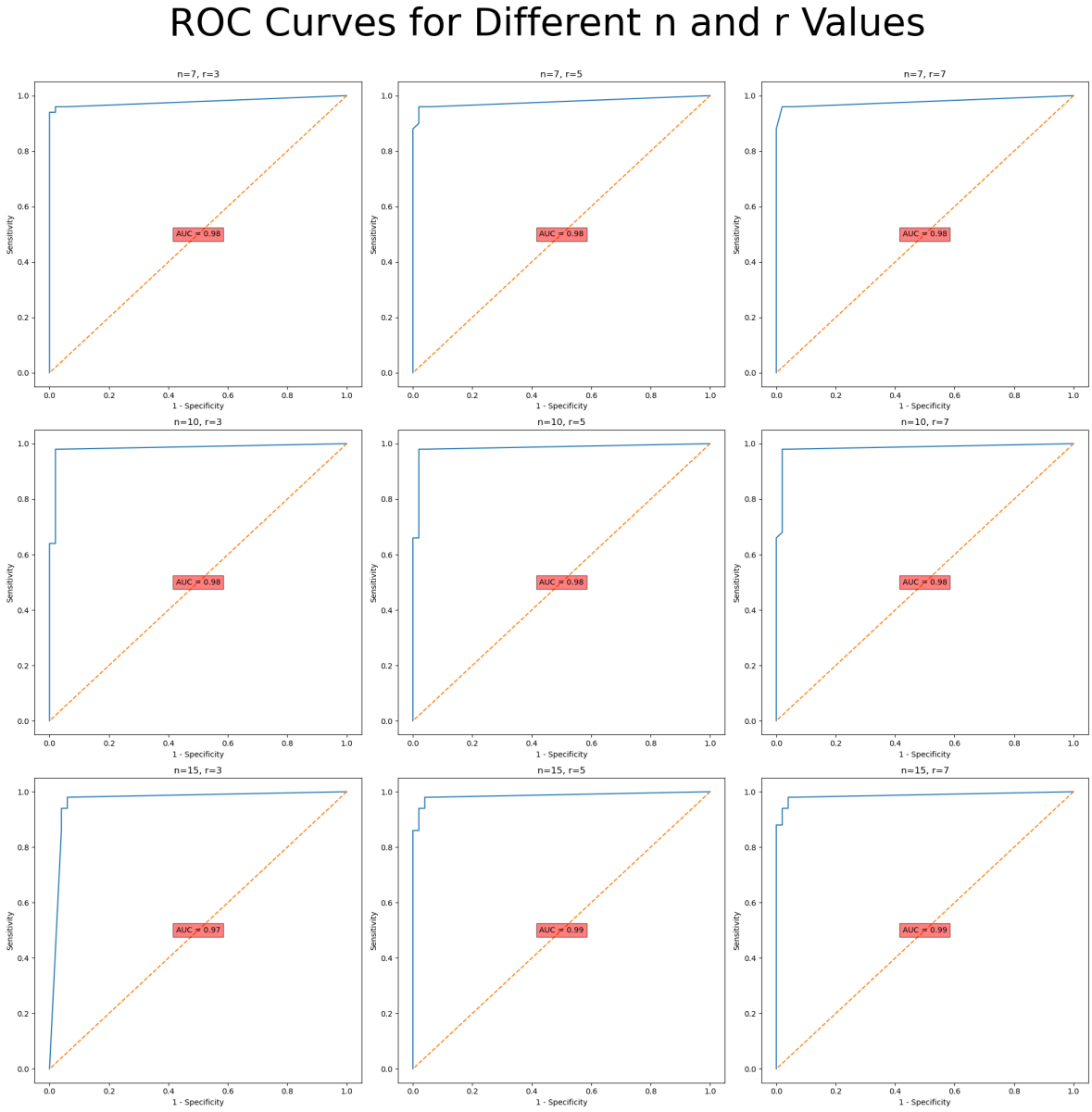

Figure 5: ROC curves resulting from a grid search over $n = [7, 10, 15]$ and $r = [3, 5, 7]$. At intermediate parameter values, the specific configurations of n and r become less critical, with all combinations yielding high AUC scores. The optimal values are identified at $n = 15$ and $r = 5, 7$

References

[1] J. Textor, *Efficient Negative Selection Algorithms*. [Online]. Available: https://tbb.bio.uu.nl/textor/negativeselection.html.

[2] J. Textor, *Negsel: String-based models of thymic selection in the immune system*. [Online]. Available: https://github.com/jtextor/negsel.