Section 2: Supervised Learning

Introduction

The goal of this section is to build and evaluate machine learning models that can accurately identify different attacks based on the provided ground truth data, which includes the labels of the attacks.

We start by splitting the dataset into training and test subsets in a stratified manner to ensure that the distribution of attack labels is consistent across both subsets. This approach maintains the representativeness of the data and ensures robust model evaluation.

Next, we select three different machine learning methods and train models using their default parameter configurations. We evaluate the performance of each model on both the training and test sets. For each model, we output the confusion matrix and a classification report, which includes metrics such as precision, recall, and F1-score. These results help us identify any signs of overfitting or underfitting.

Following this, we tune the hyperparameters of the selected models using cross-validation techniques to improve model performance. This involves finding the optimal configuration of hyperparameters and comparing the performance of the models before and after tuning. The model with the best performance is identified based on the evaluation metrics.

Finally, we investigate the instances of false positives and false negatives produced by the models. This analysis focuses on understanding the characteristics of the misclassified samples and identifying patterns or features that contribute to these errors. We report our findings, highlighting the most notable cases of misclassification and providing insights into potential areas for model improvement.

Approach to select the best hyperparameters

Initially, validation curves are created by varying the hyperparameters over a wide range to observe the performance of the model. We then identify regions where performance is best, narrowing the intervals and improving computational efficiency and model understanding. Finally, we perform a grid search on the selected parameter intervals, using cross-validation techniques to evaluate model performance on each parameter combination. This combined approach allows us to initially explore a wide range of values to identify promising regions and then perform a detailed search within those regions, optimizing both computational time and model accuracy.

What metrics do we use in the validation score ?

When creating a validation curve to evaluate the performance of a model, the choice of metric to use, between accuracy and F1 score, depends mainly on the distribution of classes in the dataset. Accuracy is a simple measure that represents the proportion of

correct predictions out of the total number of predictions, but it has limitations in the presence of unbalanced classes. The F1 score, on the other hand, is the harmonic mean of accuracy and recall and is useful when you have a dataset with unbalanced classes.

So in our specific case where one class is underrepresented with a size of 15% relative to the others, accuracy is not the best metric, so using the F1 score is more appropriate.

Logistic Regression

Introduction

Logistic regression is a technique used for binary classification problems that can be extended to multiclass classification problems. This statistical model is based on the concept of estimating the probability that a given observation belongs to one of the possible classes, using a logistic function.

The advantages of logistic regression include the possibility of using regularization (such as L1 and L2) to prevent overfitting.In addition, logistic regression is relatively simple to implement and does not require significant computational resources, making it suitable even for large datasets.

However, logistic regression also has some disadvantages. A significant limitation is its assumption of linearity between the independent variables and the logarithm of probabilities, which can reduce performance in the presence of nonlinear relationships.

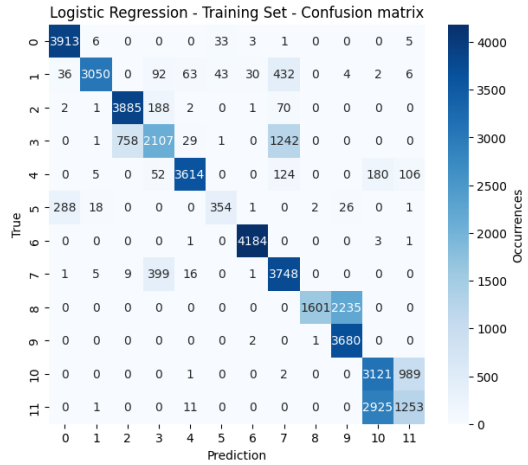
Hyperparameters

Below are the hyperparameters we evaluate for our model:

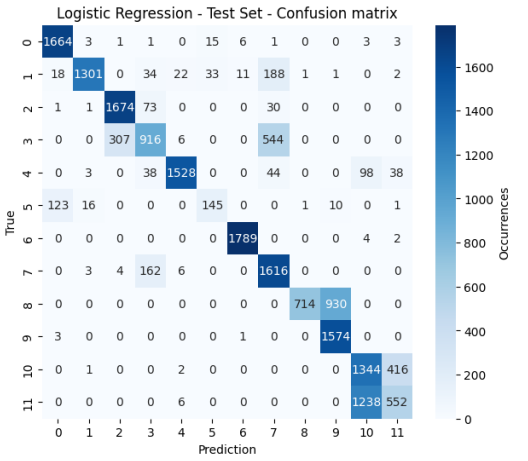
- **solver**: Algorithm to use in the optimization problem.
    - newton-cg
    - lbfgs
    - liblinear
    - sag
    - saga
- **multi_class**: Specifies how to handle multi-class classification.
    - ovr: One-vs-Rest.
    - multinomial: Multinomial loss fit across the entire probability distribution.
- **penalty**: Specifies the norm used in the penalization (regularization).
    - l1: Uses L1 regularization (Lasso), which can set some coefficients to zero, effectively performing feature selection.
    - l2: Uses L2 regularization (Ridge), which tends to distribute the error among all the parameters, reducing the impact of each one.
    - elasticnet: Combines L1 and L2 penalties, allowing for a mix of feature selection and regularization.
    - none: No regularization applied.

- **C**: Inverse of regularization strength, controls the trade-off between achieving a low error on the training data and minimizing the norm of the coefficients. Smaller values specify stronger regularization.

Training with default parameters



	precision	recall	f1-score	support
0	0.92	0.99	0.95	3961
1	0.99	0.81	0.89	3758
2	0.84	0.94	0.88	4149
3	0.74	0.51	0.60	4138
4	0.97	0.89	0.92	4081
5	0.82	0.51	0.63	690
6	0.99	1.00	0.99	4189
7	0.67	0.90	0.77	4179
8	1.00	0.42	0.59	3836
9	0.62	1.00	0.76	3683
10	0.50	0.76	0.60	4113
11	0.53	0.30	0.38	4190
accuracy			0.77	44967
macro avg	0.80	0.75	0.75	44967
weighted avg	0.80	0.77	0.76	44967

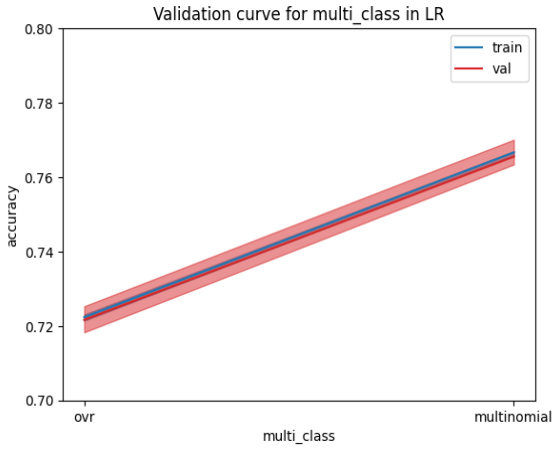


	precision	recall	f1-score	support
0	0.92	0.98	0.95	1697
1	0.98	0.81	0.89	1611
2	0.84	0.94	0.89	1779
3	0.75	0.52	0.61	1773
4	0.97	0.87	0.92	1749
5	0.75	0.49	0.59	296
6	0.99	1.00	0.99	1795
7	0.67	0.90	0.77	1791
8	1.00	0.43	0.61	1644
9	0.63	1.00	0.77	1578
10	0.50	0.76	0.60	1763
11	0.54	0.31	0.39	1796
accuracy			0.77	19272
macro avg	0.79	0.75	0.75	19272
weighted avg	0.80	0.77	0.76	19272

The results obtained show parallel features in both the training set and the test set, indicating that the model seems to generalize well. Using the macro_avg metric, we aim to identify hyperparameters that can further improve the performance of the model.

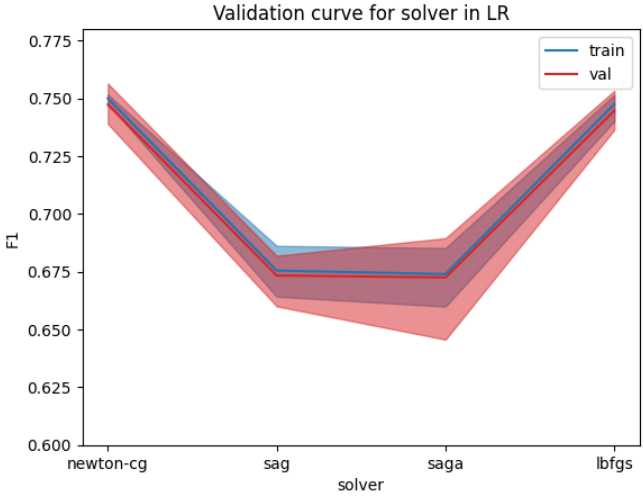
Validation Curve

Multiclass



The graph shows that performance, as measured by the F1 score, is better for the "multinomial" option than "ovr" for both training and validation datasets.
It is also noted that the training and validation curves are parallel and very close, indicating that the model generalizes well, showing no signs of significant overfitting or underfitting.
Finally, the confidence intervals are narrow, indicating relatively low variability in performance between the different folds of the cross-validation.

Solver

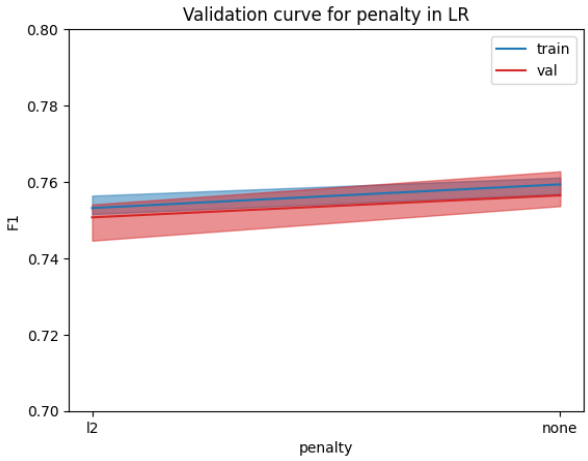


Analyzing the graph, we see that the "newton-cg" solver and the "lbfgs" solver have the highest F1 score values for both the training and validation sets. These two solvers maintain F1 scores close to 0.75, suggesting that they offer superior performance compared to the other solvers considered. The validation curves for "newton-cg" and "lbfgs" are parallel and close to the training curves, indicating that the model generalizes well without showing obvious signs of overfitting or underfitting.

The "sag" and "saga" solvers, on the other hand, show lower performance, with F1 scores dropping to about 0.65-0.68. This drop in performance is evident in both the training and validation data. In addition, the curves for "sag" and "saga" are lower and show greater variability, as indicated by the wider confidence intervals. This suggests that these solvers may not be suitable for the dataset.

In conclusion, we choose "newton-cg" as the solver because it has a slightly higher F1 than "lbfgs," later we will analyze both of them in the random search grid.

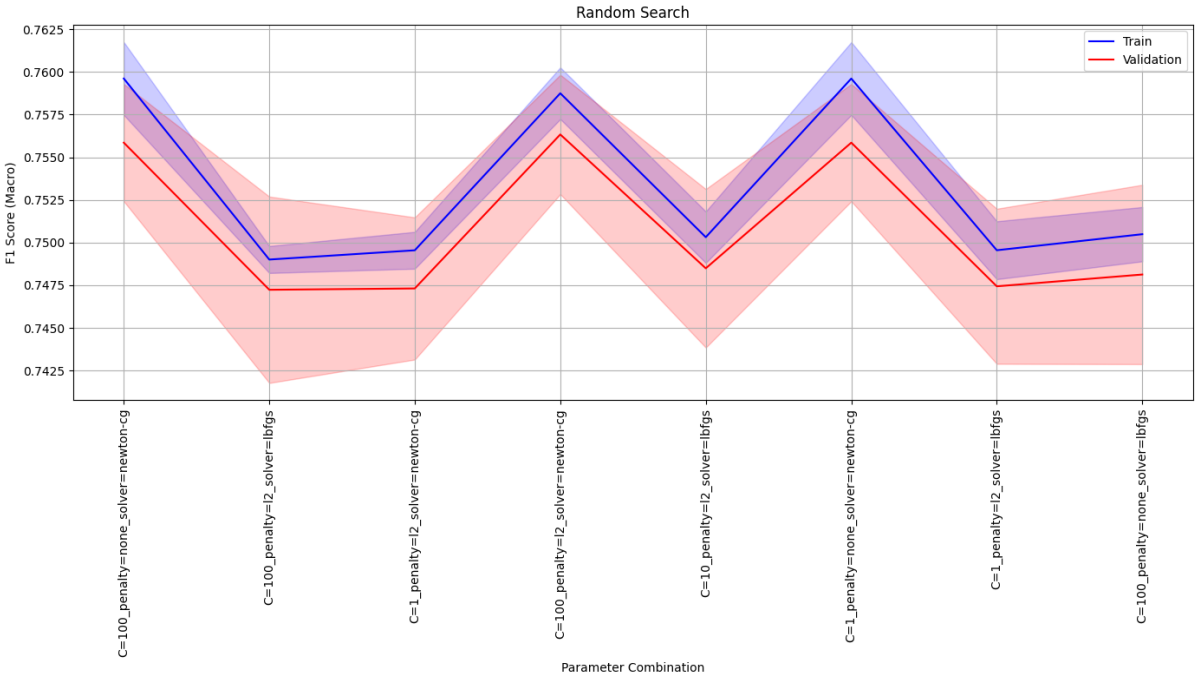
Penality



The graph shows that performance, as measured by the F1 score, is slightly better for the "none" option than for "l2" for both the training and validation datasets, indicating that the absence of regularization may lead to better performance in the context of this specific dataset.

To confirm the results obtained, we will proceed by using a randomized search a random grid search examining a range of values for the parameter C in comparison with the penalty.

Randomized search grid




```
Best F1 Score (Macro): 0.7563350567282467
Parameter Combination: C=100_penalty=l2_solver=newton-cg
```

These parameters were identified as the best in the context of our classification problem, achieving an F1 macro score on the validation set of 76 percent. This represents an improvement over the initial score of 75%, suggesting that the model has gained in accuracy and recall for the different classes.

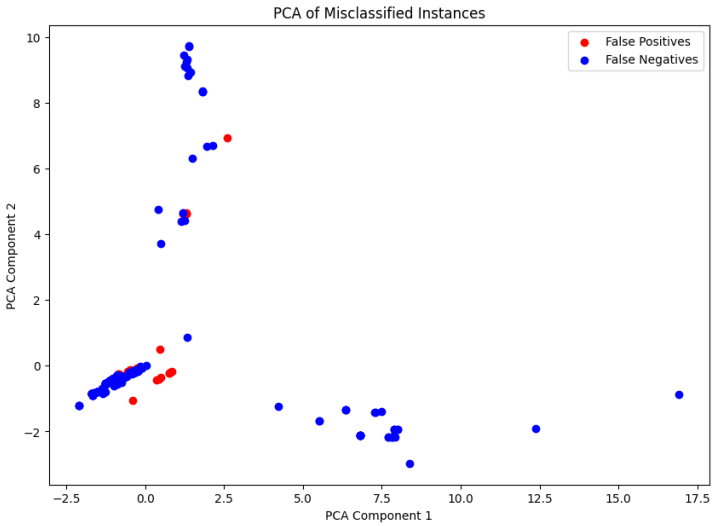
The validation graph, which shows the performance in terms of accuracy for each iteration of the search, shows that the training and validation scores have remained stable, indicating that the generalization of the model has not been compromised.

Analyzing Misclassifications: False Positives and Negatives in Feature Context

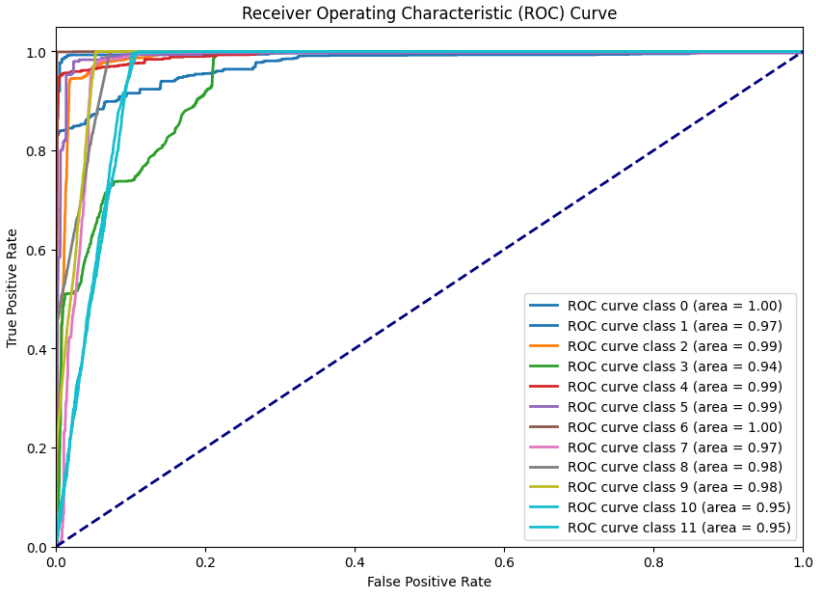
```
Confusion Matrix:
[[1662    6    0    1    0   17    6    1    0    1    3    0]
 [  13 1335    1   34   19   34   11  160    1    1    1    1]
 [   0    3 1668   72    0    0    0   36    0    0    0    0]
 [   0    1  307  908    6    0    0  551    0    0    0    0]
 [   0    3    0   38 1624    0    0   48    0    0   25   11]
 [ 116    5    0    0    0  163    0    0    1   10    0    1]
 [   0    0    0    0    0    0 1787    0    0    0    6    2]
 [   0    3    3  137    9    0    0 1638    1    0    0    0]
 [   0    0    0    0    0    1    0    0  716  927    0    0]
 [   3    0    0    0    0    0    1    0    2 1572    0    0]
 [   0    1    0    0    1    0    0    0    0    0 1118  643]
 [   0    0    0    0    6    0    0    0    0    0 1017  773]]
```

Classes 0, 2, 6 and 9 show a high number of true positives, indicating that the model is particularly effective in classifying these categories. However, class 1 shows numerous false positives and false negatives, with many instances misclassified as class 7. Class 3 shows significant confusion with class 7, with 551 instances misclassified. Class 10 has 643 false positives from class 11 and 1017 false negatives misclassified as class 11. In addition, class 7 shows confusion with several other classes, particularly with classes 3 and 1. Classes 8 and 11 show significant confusion with each other, with 927 instances of class 8 classified as class 9 and 643 instances of class 10 classified as class 11.

These observations suggest that the model is effective for some classes but has separability problems for others.



False negatives are distributed in different regions of the PCA space, indicating that instances misclassified as negative are scattered among different principal components. False positives are fewer in number and show some concentration, suggesting that there are specific regions of the feature space where the model tends to misclassify instances as positive.
The presence of misclassifications in specific areas of the PCA space suggests that there are some combinations of features that the model fails to discriminate effectively.



The plot of ROC curves for each class shows that the logistic regression model has excellent discrimination abilities for most classes. Classes 0 and 6 show an area under the curve (AUC) of 1.00, indicating perfect performance and an ability of the model to distinguish these classes without error. Classes 2, 4, 5, 8 and 9 show AUCs between 0.98 and 0.99, suggesting that the model discriminates these classes very well, with high rates of true positives and low rates of false positives. However, classes 1, 3, 7, 10 and 11, with AUCs between 0.94 and 0.97, show greater challenges, indicating that the model has more difficulty distinguishing these classes than those with higher AUCs.

In conclusion, the ROC curves and associated AUCs indicate that the logistic regression model has excellent discrimination capabilities for most classes, with some exceptions where performance is good but can be improved. Feature analysis and adoption of more advanced models are promising strategies to address the identified challenges.

Random Forest

Introduction

The Random Forest is a powerful machine learning algorithm used for classification and regression. It combines multiple decision trees to enhance robustness and accuracy, adapting well to various types of data without requiring extensive tuning. Thanks to bagging and the random selection of feature subsets, it significantly reduces the risk of overfitting, ensuring better generalization on new data. It can effectively handle missing data, maintaining good performance. Additionally, it provides estimates of variable importance, helping to identify the most influential features.

However, the Random Forest is less interpretable compared to simpler models like logistic regression, which can be an issue when model interpretability is crucial. Furthermore, it is computationally expensive and requires long training times, especially with large datasets like ours. Despite these disadvantages, the Random Forest remains an excellent choice for those seeking a robust and accurate model capable of handling complex data and providing useful insights.

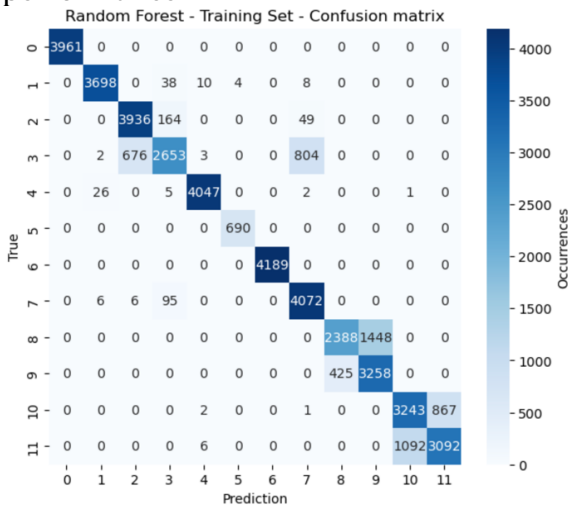
Hyperparameters

Below are the hyperparameters we evaluate for our model:

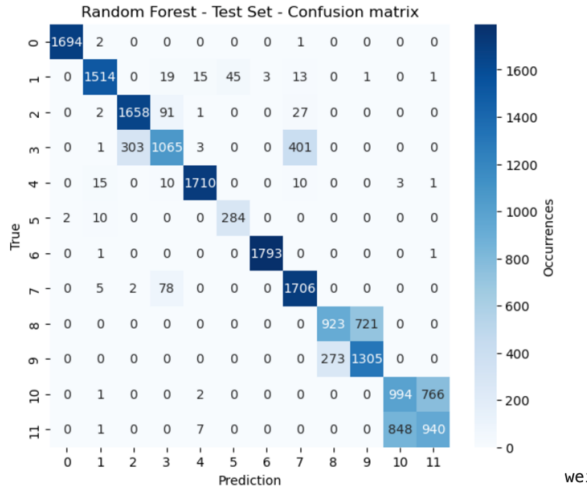
- **n_estimators**: The number of trees in the forest.
    - Positive integer. A higher number of trees generally improves the model's performance (up to a certain point) because it combines more estimates, but it also increases computation time and memory usage.
- **max_depth**: The maximum depth of each tree.
    - Positive integer. A higher value allows the trees to grow deeper, capturing more details of the dataset, but can lead to overfitting. A lower value can prevent overfitting but risks underfitting.
- **bootstrap**: Indicates whether to use bootstrap sampling when building trees.
    - **True**: Trees are built on bootstrapped samples of the dataset (with replacement).
    - **False**: Trees are built on samples without replacement.
- **max_features**: The maximum number of features to consider for the best split.
    - **auto**: Uses all features.
    - **sqrt**: Uses the square root of the total number of features.
- **min_samples_leaf**: The minimum number of samples required to be at a leaf node.
    - Positive integer. Specifying a minimum number of samples per leaf can reduce overfitting, as nodes cannot be too specific to the training data.
- **min_samples_split**: The minimum number of samples required to split an internal node.
    - Positive integer. Setting a higher value for this parameter can prevent the data from fragmenting into too many small segments, thereby reducing overfitting.

Training with default parameters

Training the model without using any particular parameters, we obtained the following performance.



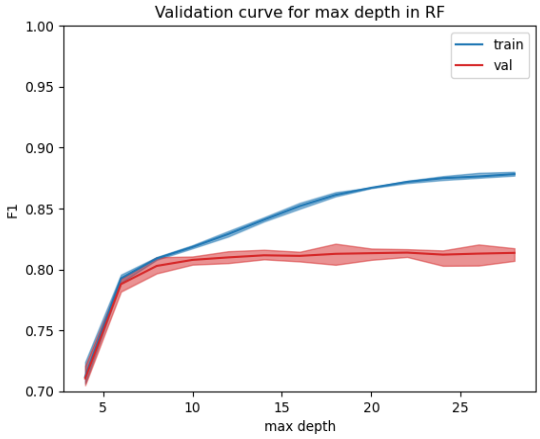
	precision	recall	f1-score	support
0	1.00	1.00	1.00	3961
1	0.99	0.98	0.99	3758
2	0.85	0.95	0.90	4149
3	0.90	0.64	0.75	4138
4	0.99	0.99	0.99	4081
5	0.99	1.00	1.00	690
6	1.00	1.00	1.00	4189
7	0.82	0.97	0.89	4179
8	0.85	0.62	0.72	3836
9	0.69	0.88	0.78	3683
10	0.75	0.79	0.77	4113
11	0.78	0.74	0.76	4190
accuracy			0.87	44967
macro avg	0.89	0.88	0.88	44967
weighted avg	0.88	0.87	0.87	44967



	precision	recall	f1-score	support
0	1.00	1.00	1.00	1697
1	0.98	0.94	0.96	1611
2	0.84	0.93	0.89	1779
3	0.84	0.60	0.70	1773
4	0.98	0.98	0.98	1749
5	0.86	0.96	0.91	296
6	1.00	1.00	1.00	1795
7	0.79	0.95	0.86	1791
8	0.77	0.56	0.65	1644
9	0.64	0.83	0.72	1578
10	0.54	0.56	0.55	1763
11	0.55	0.52	0.54	1796
accuracy			0.81	19272
macro avg	0.82	0.82	0.81	19272
weighted avg	0.81	0.81	0.81	19272

The results of the Random Forest model show an accuracy of 87% on the training set and 81% on the test set, suggesting a fair ability to generalize. Some classes, however, show a significant drop in performance in the test set compared to the training set, highlighting possible overfitting problems. Classes 10 and 11 have particularly low F1-scores in the test set (0.55 and 0.54), indicating difficulties in their correct classification on new data.

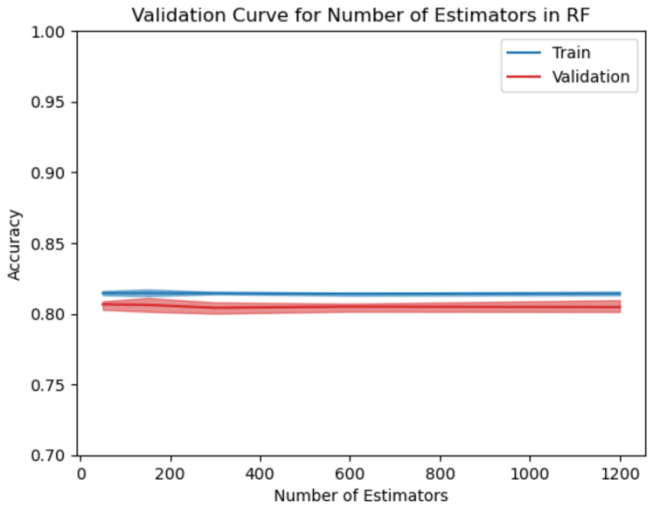
Validation Curve
Max depth



The validation curve for the maximum depth in the Random Forest model shows that training accuracy (blue line) continuously increases as the maximum depth increases. However, validation accuracy (red line) initially increases and peaks around a maximum depth of 10 to 15. Beyond this point, the validation accuracy plateaus and even slightly decreases, suggesting that the model begins to overfit the training data.

Using this analysis, we can set the max_depth to 10 for our subsequent experiments and proceed to explore other parameters.

Number of estimator



We observe that as the number of estimators increases, the training accuracy remains relatively stable around 0.82. This indicates that adding more trees beyond a certain point does not significantly enhance the model's ability to fit the training data.

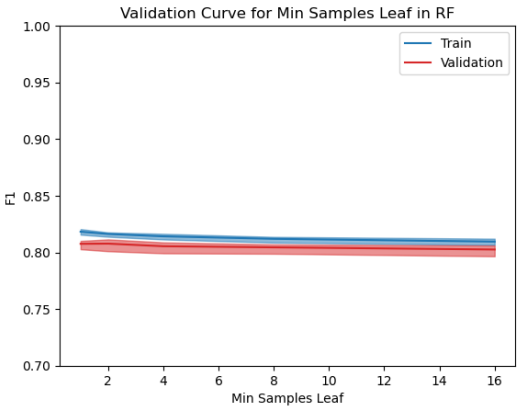
The validation accuracy also remains relatively stable around 0.81 across the different values of n_estimators. This suggests that increasing the number of trees does not substantially impact the model's generalization performance.

Based on the validation curve, there is no significant change in performance with the increase in n_estimators. Therefore, a value in the range of 100-200 estimators is likely

sufficient, as adding more trees does not improve accuracy but increases computational cost.

Using this analysis, we can confidently set the n_estimators to 150 for our subsequent experiments, balancing performance and computational efficiency.

Min Samples

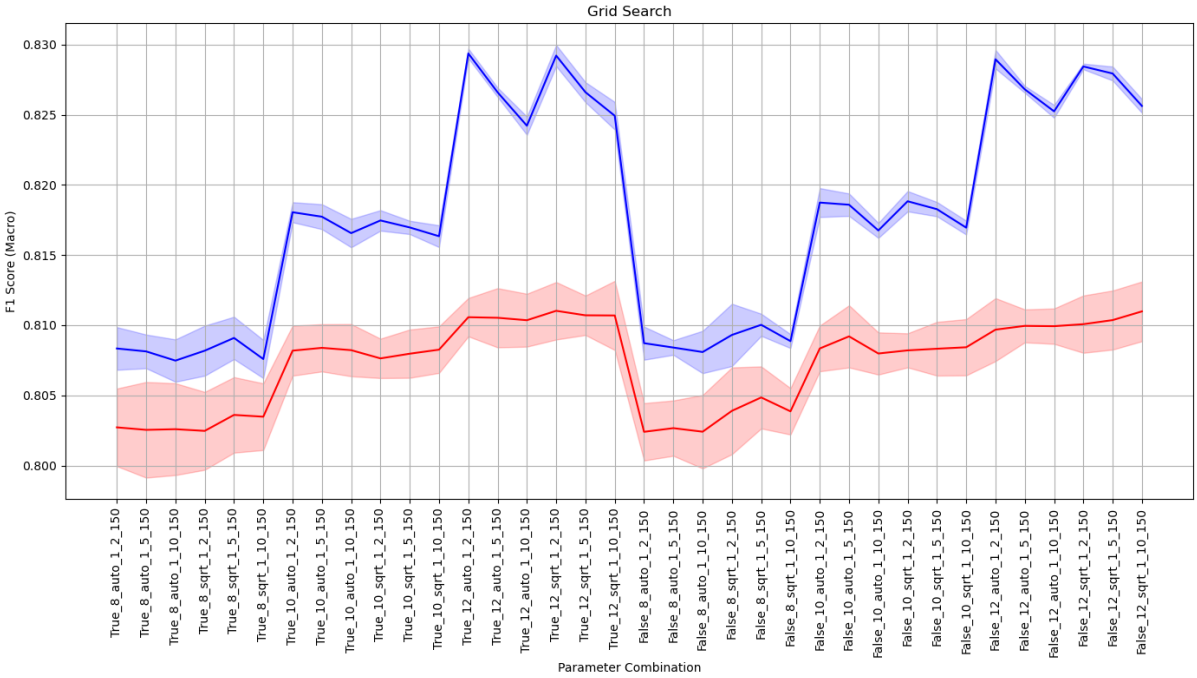


Looking at the figure, we see that both the training curve (blue) and the validation curve (red) tend to decrease slightly as the value of min_samples_leaf increases. Both curves stabilize around an F1 score of about 1-2.

The optimal value of min_samples_leaf to choose is 1, since F1 scores for training and validation are slightly better than higher values and the model maintains a good balance between complexity and generalization ability.

We will now proceed to explore other parameters using grid search to further fine-tune the model.

Grid Search



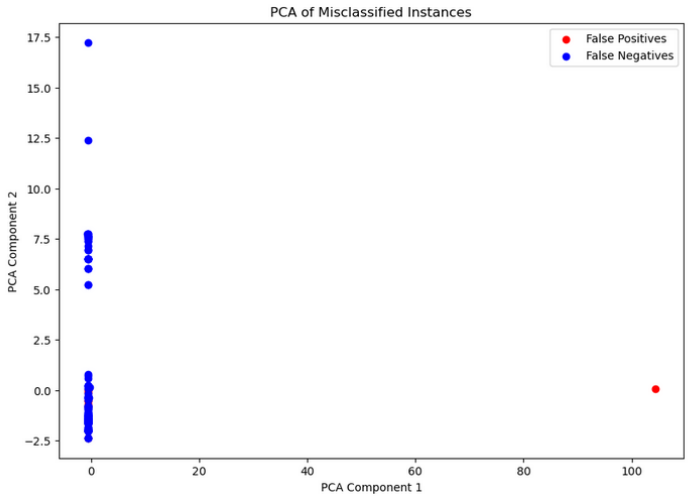
```
Best F1 Score (Macro): 0.8081864094324255
Parameter Combination: True_10_auto_1_2_150
```

There is no improvement in model performance before and after tuning, as evidenced by the fact that the Macro Avg F1-Score remains essentially unchanged around 0.81. However, the level of overfitting has decreased, as indicated by the reduction in training set accuracy from 87% to 82%, signaling that the model is less overtrained on the training set.

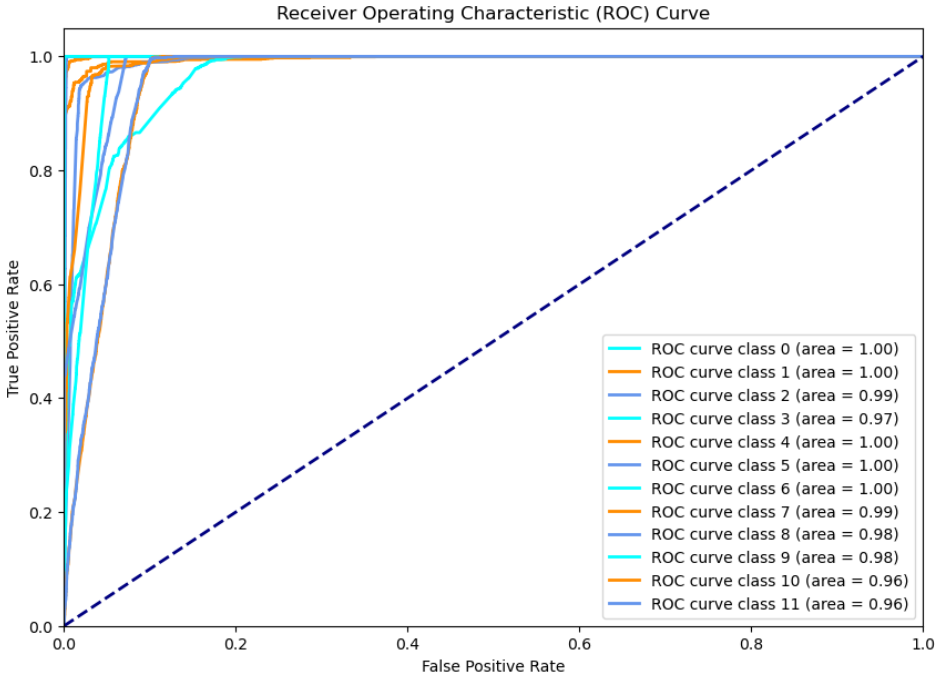
Analyzing Misclassifications: False Positives and Negatives in Feature Context

```
Confusion Matrix:
[[1694    2    0    0    0    0    0    1    0    0    0    0]
 [   0 1442    0   24   20   47    3   73    0    1    0    1]
 [   0    2 1672   76    1    0    0   28    0    0    0    0]
 [   0    1  306 1011    7    0    0  448    0    0    0    0]
 [   0   13    0    8 1704    0    0   10    0    0   11    3]
 [   3    1    0    0    1  291    0    0    0    0    0    0]
 [   0    0    0    0    0    0 1794    0    0    0    0    1]
 [   0    2    2   75    3    0    0 1709    0    0    0    0]
 [   0    0    0    0    0    0    0    0  721  923    0    0]
 [   0    1    0    0    0    0    0    0    7 1570    0    0]
 [   0    1    0    0    4    0    0    0    0    0  950  808]
 [   0    0    0    0    7    0    0    0    0    0  746 1043]]
```

The confusion matrix shows that the model has good overall accuracy, with most instances classified correctly. However, there are significant misclassifications between some pairs of classes, such as between classes 3 and 8, and between classes 6 and 8.



What the matrix illustrates is further evidenced by PCA, where false positives and false negatives are distinguished, indicating potential areas of improvement for the model. The reduction in overfitting, observed in the decrease in training set accuracy, suggests that the model is now less prone to learning noise specific to the training dataset.



The ROC curves show that the model has excellent ability to discriminate between classes, with most classes achieving an area under the curve (AUC) very close to 1.0. Classes 0, 1, 4, 5 and 6 have a perfect AUC of 1.0, indicating error-free classification. Classes 2, 7, 8 and 9 have an AUC between 0.98 and 0.99, demonstrating very high but not perfect discrimination ability. Classes 3, 10 and 11 have AUCs of 0.97 and 0.96, suggesting a slight imperfection in discrimination, although they still remain high performers. These results indicate that while the model is generally very effective, there is room for improvement for classes with slightly lower AUCs, especially to ensure perfect separability in all cases.

Support vector machines

Introduction

The Support Vector Machine (SVM) is a machine learning model capable of handling high-dimensional data and demonstrates robustness in preventing overfitting. SVMs work by finding the hyperplane that maximizes the margin between different classes, contributing to improved model generalization. Additionally, their ability to find an optimal hyperplane in high-dimensional spaces makes them ideal for complex problems.

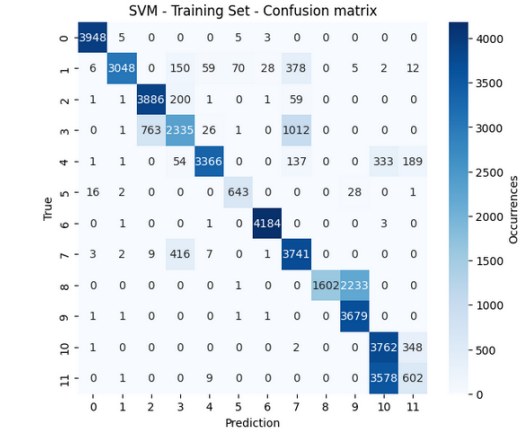
Thanks to kernels, SVMs can handle non-linear relationships in the data by transforming them into a high-dimensional space where a linear hyperplane can be used to separate the classes. The goal is to find the hyperplane that separates the classes with the widest possible margin, thereby reducing the risk of misclassification.

Despite their versatility, effectiveness in high-dimensional spaces, and robustness, training an SVM is computationally expensive and slow. Furthermore, selecting the hyperparameters is time-consuming. Additionally, SVMs can sometimes be sensitive to noisy data.

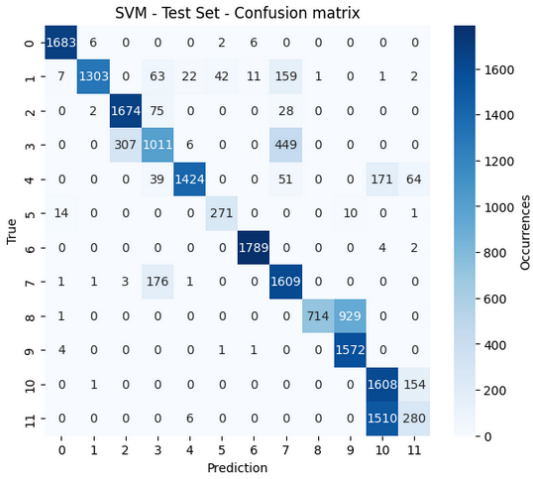
Hyperparameters

- Kernel: Determines the kernel function used to transform the data. The explored options include:
    - Linear
    - Polynomial
    - Radial Basis Function
- Sigmoid C (Regularization Parameter): Controls the trade-off between maximizing the margin of the hyperplane and minimizing the classification error on the training data. A high value of C aims to classify all training points correctly, while a low value allows for a wider margin.
- Gamma (γ): Specific to RBF, polynomial, and sigmoid kernels. It influences the shape of the decision boundary.

Training with default parameters



	precision	recall	f1-score	support
0	0.99	1.00	0.99	3961
1	1.00	0.81	0.89	3758
2	0.83	0.94	0.88	4149
3	0.74	0.56	0.64	4138
4	0.97	0.82	0.89	4081
5	0.89	0.93	0.91	690
6	0.99	1.00	1.00	4189
7	0.70	0.90	0.79	4179
8	1.00	0.42	0.59	3836
9	0.62	1.00	0.76	3683
10	0.49	0.91	0.64	4113
11	0.52	0.14	0.23	4190
accuracy			0.77	44967
macro avg	0.81	0.79	0.77	44967
weighted avg	0.80	0.77	0.76	44967



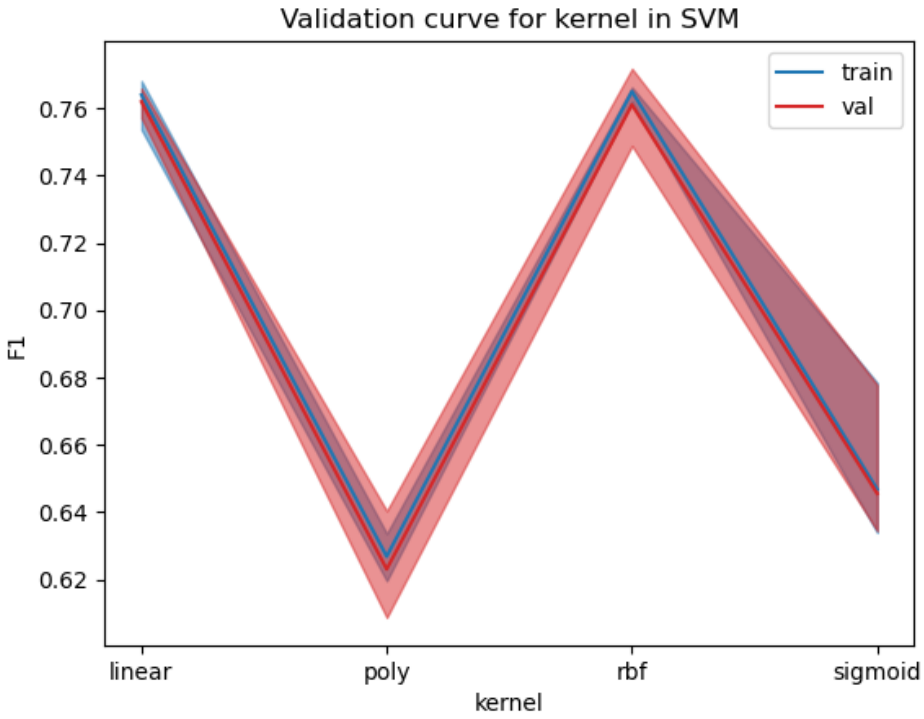
	precision	recall	f1-score	support
0	0.98	0.99	0.99	1697
1	0.99	0.81	0.89	1611
2	0.84	0.94	0.89	1779
3	0.74	0.57	0.64	1773
4	0.98	0.81	0.89	1749
5	0.86	0.92	0.89	296
6	0.99	1.00	0.99	1795
7	0.70	0.90	0.79	1791
8	1.00	0.43	0.61	1644
9	0.63	1.00	0.77	1578
10	0.49	0.91	0.64	1763
11	0.56	0.16	0.24	1796
accuracy			0.78	19272
macro avg	0.81	0.79	0.77	19272
weighted avg	0.81	0.78	0.76	19272

Analysis of the SVM model on the test set shows significant missclassifications. Class 3 is often confused with classes 2 and 7, class 7 with class 3, and class 8 with class 9. Class 11 presents the greatest difficulties, with many examples misclassified as class 10. Accuracy ranges from 0.49 (class 10) to 1.00 (class 8), recall from 0.16 (class 11) to 1.00 (class 6), and the F1-score is particularly low for class 11 (0.24).

In the training set, the missclassification patterns are similar, confirming the difficulties already observed in the test set. Suboptimal performance is mainly due to dataset imbalance and feature overlap between some classes. Optimizing hyperparameters and improving feature quality could help in reducing these missclassifications.

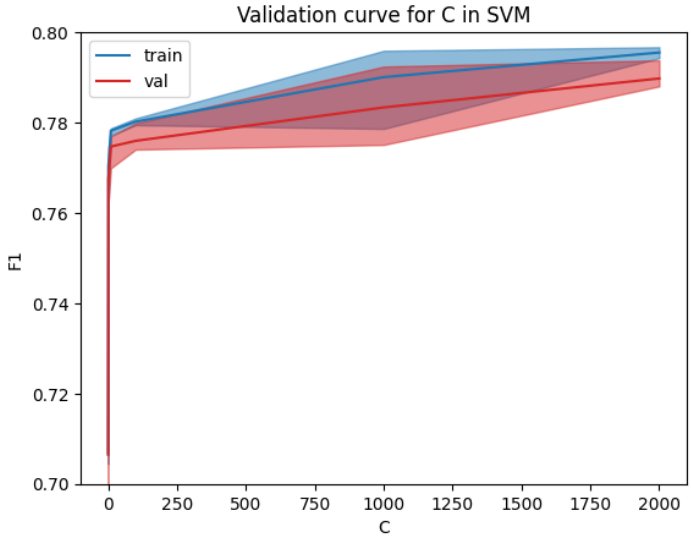
Validation Curve

Kernel



The linear kernel shows similar accuracy as the RBF kernel. By performing various experiments, we found that the linear kernel requires significantly more training time, making it difficult to tune. Therefore, the choice of the RBF kernel is motivated by practical considerations related to computational efficiency and model performance, offering greater flexibility in modeling complex relationships in the data and better ability to generalize to unseen data.

C

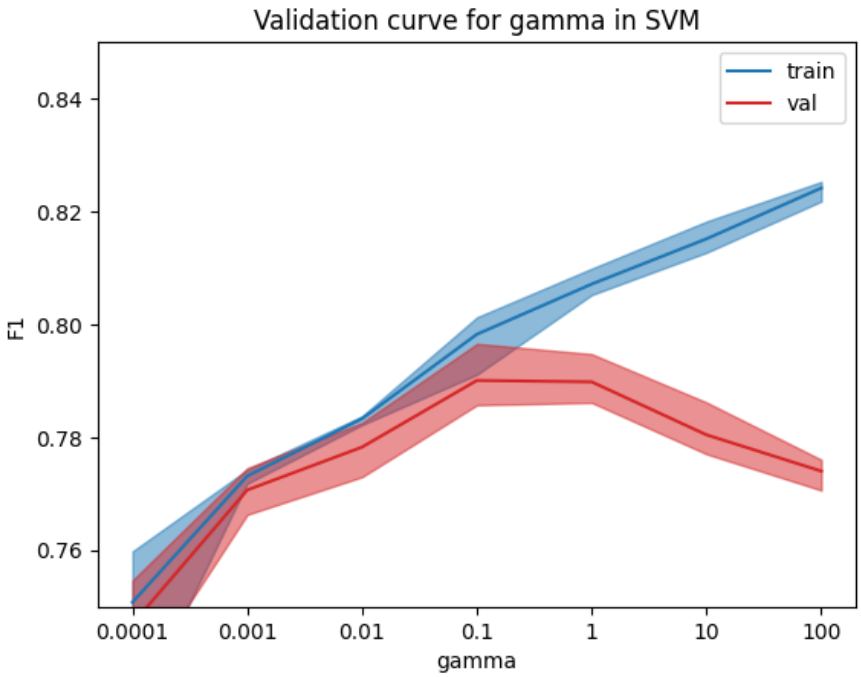


Both curves show a significant initial improvement in F1 score as C increases, which then tends to decrease the slope of this increase for large C. With very low values of C, performance on the validation set is poor, indicating that the model is not sufficiently complex to capture the features of the data. As C increases, the model becomes more complex, improving performance on both sets. However, at very high values of C, there

is a slight tendency toward overfitting, where the score on the training set is slightly higher than that on the validation set.

Nevertheless, at C=1000, the model seems to balance the trade-off between bias and variance well, with an F1 score close to the maximum for both the training and validation sets, indicating good generalization. This value of C avoids both underfitting, associated with very low C values, and the risk of overfitting, associated with very high C values, not printed here because they require large computational resources.

Gamma




```
Average F1 Score (Training): 0.7994878524579546
Average F1 Score (Test): 0.7958686795979264
Parameter Combination: kernel=rbf, C=1000, gamma=0.1 (Used in this run)
```

Performance evaluation of tuning

Although grid search or random search techniques are valid for hyperparameter optimization, their computational requirements were found to be excessive for this specific model. We have already achieved a significant increase in performance, with F1 scores going from 77% to 80% on the training set and from 77% to 79.6% on the test set.

To further optimize the model without incurring significant computational overhead, we could pursue a more focused approach, exploring a narrow range of gamma parameters between 0.1 and 1.0, where a maximum might reside, evaluating them over a range of C values.

Analyzing Misclassifications: False Positives and Negatives in Feature Context