Machine Learning and Pattern Classification

Team Report

Assignment 3: Data Classification

Team Aberrant

Ábel Boros (k11944603)
Henry-Willy Lechner (k12007370)
Luiz Henrique Madjarof (k12141871)

Outline

In this assignment, our goal was to run several classification experiments in order to test the efficiency of different classification strategies, models, and approaches. In this section, we will explain our approach to this assignment listing our general step-by-step method.

Our classification was done as follows:

1. Feature selection
    a. Analysis
    b. Dimensionality Reduction
2. Cross-validation
3. Model training
4. Evaluation

There were a few steps for which the approach was of unanimous agreement and others in which we tested different strategies as an attempt to improve the efficiency of classification. The most important step prior to modelling was feature selection. There, we experimented with using inter-feature correlation, PCA and feature-label correlation (under alia) to reduce dimensionality.

Outline

After having set the features that would be used in our model, the data was split into training and testing sets at a rate of 80% to 20%. And cross-validation was done by dividing the dataset into 5 randomised subgroups and using Stratified K-Fold for cross-validation.

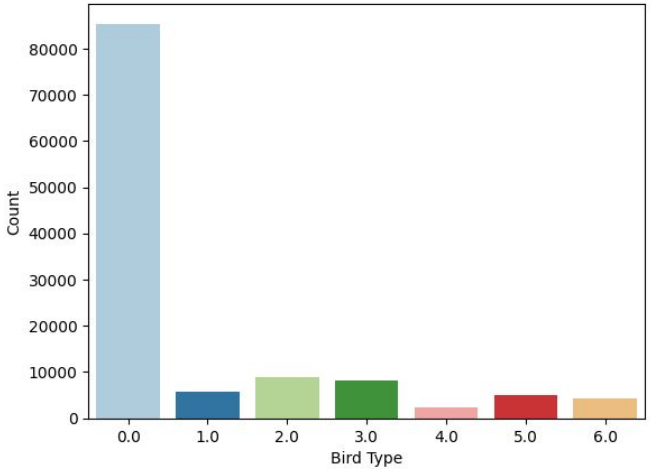
For classification, we focused on the following models:

1. K Nearest Neighbors (KNN)
2. Quadratic Discriminant Analysis (QDA)
3. Naive Bayes
4. Random Forest

In the following sections, we will show the results of our experimentation expliciting what worked best for us and what did not.

1. Label Distribution

Figure 1. Occurrences of Birds



Unbalanced Label Distribution

Before we begin discussing our feature selection strategies, we want to point out the unbalanced distribution of labels in our data.

Due to the enormously higher number of zeros, classification models will tend to show much higher accuracy rates when classifying the data for this label. Therefore we need to address this issue by using the appropriate methods when training our models and when measuring our model performance. We will discuss these methods in more details at later sections.

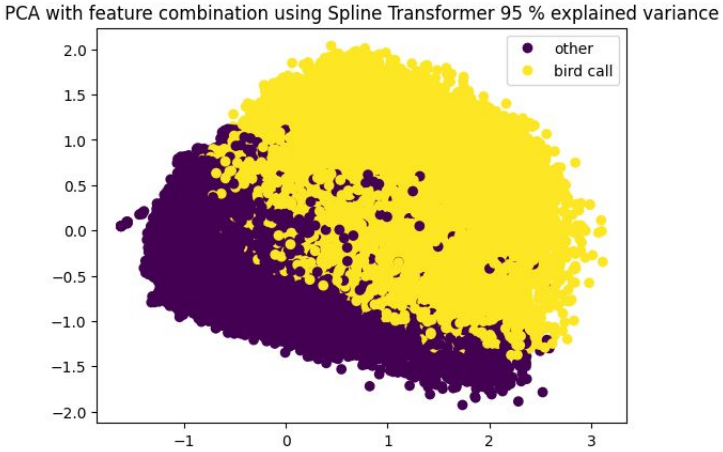


Figure 2

Spline Transformer

If we apply PCA together with Spline Transformation to our label data, we can see how the annotated labels, which are in this case the “0” labels, and every other label from the birds, was clearly separated. However, this may mean that a model would be able to efficiently classify ‘no birds/others’ or zero-label data very well and all other labels inefficiently

Note also that, for this approach, we set the individual annotated labels for every label as “1”, except the “0”-label. Because the goal was to either classify it as an annotated “bird call” or not as an annotated “bird call”.

2. Feature Selection

PCA and Spline Transformation:

If we only apply PCA to the data, there are no significant trends to be observed. Even if we separate our label data into no birds or birds, we cannot see clear patterns.

The spline transformer feature combination method, as seen in the previous slide, allows for more expressive modeling of complex relationships between features by incorporating nonlinear interactions. It can help improve the performance of machine learning models, especially when dealing with high-dimensional data with intricate dependencies.

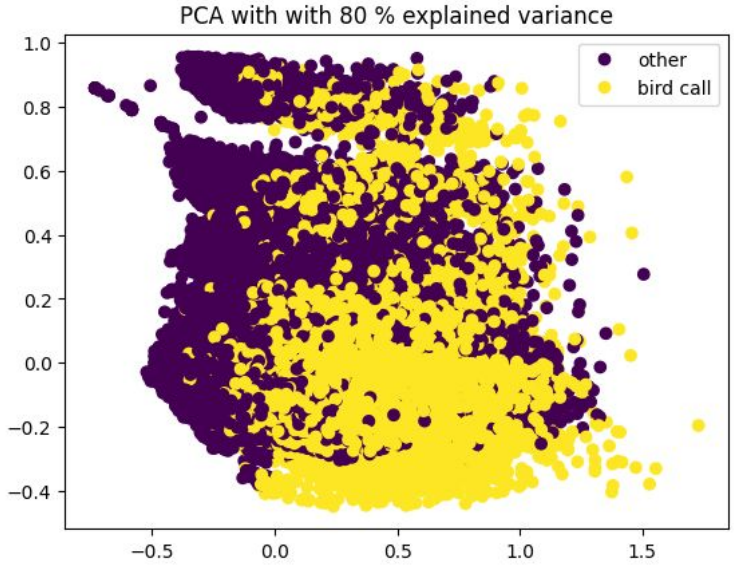


Figure 3.

Drawbacks:

However, although this method may seem very promising, it increases the dimensionality of feature data which for purposes of classification means extremely long data processing time. In our case, the feature combination matrix that resulted from the approach shown in figure 2 was of size (120.000, 848). Its original size was (120.000, 548), meaning we would go from 548 to 848 features. Since our goal was to reduce processing time and avoid overfitting through feature dimensionality reduction, this method was eventually discarded.

2. Feature Selection

Inter-features Correlation:

In order to reduce dimensionality in our feature data, we used a series of approaches. First, we examined inter-feature relationships. We looked for features that highly correlated with other features and set a baseline of 0.9 above which features were disregarded. Meaning that all features with a correlation of over 90% were dropped. We were then left with 265 features, which we analyse and compress.

Figure 4. Correlation Matrix for the Remaining 265 Features

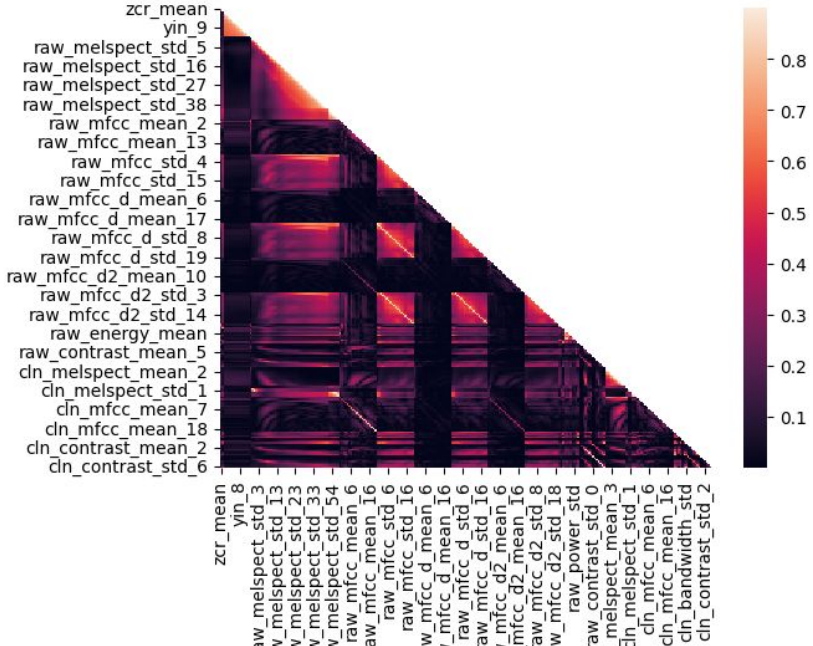
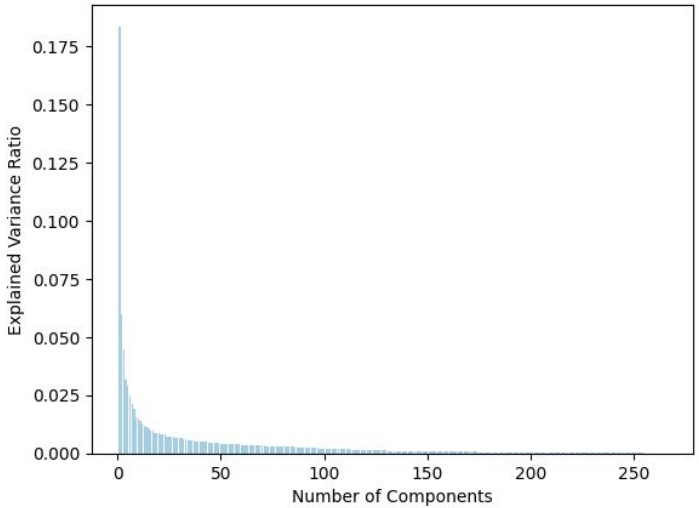


Figure 5. Scree Plot



PCA:

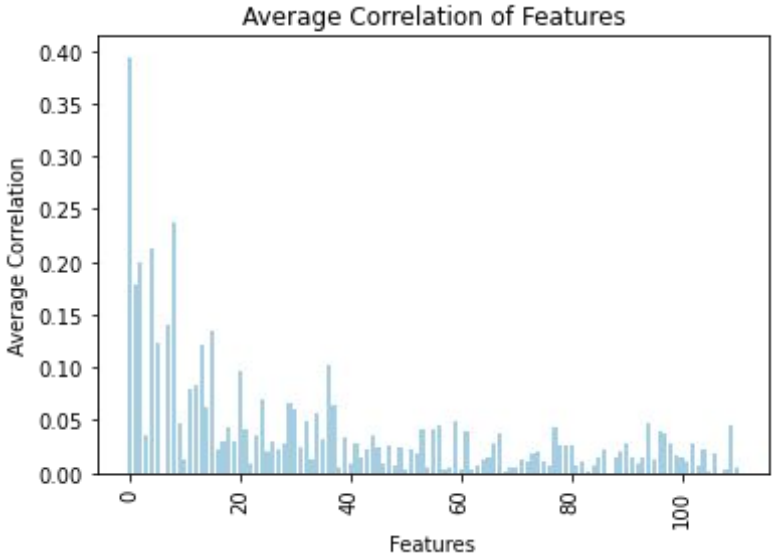
We then went a step further. By applying PCA, we found that with 111 components we could explain 90% of the variance in the remaining feature dataset, such that we could further reduce our number of features for classification.

2. Feature Selection

Feature-Label Correlation:

To further reduce dimensionality in our feature data, we decided to keep looking into correlation. We examined label-feature relationships and eliminated features which had little-to-no correlation with the labels. Our baseline used for that was 0.02. Meaning that all features that correlated with labels at a rate less than 2% were eliminated.

Figure 6.



Feature-Label Correlation Distribution:

In order to graphically visualize the relationship between labels and our features, we created figure 6. It shows for each of the 111 features included in our dataset it's corresponding average correlation value with the labels dataset. As we can see, there are no features that correlate very strongly with labels, yet there is a slight pattern. For the majority of features, correlation does not reach 0.05 or 5%. Such that we can set a baseline for further dimensionality reduction. Since 0.05 is still a bit too high for our feature data, we decided to set our standard at 0.02, as previously mentioned, which gave us a total of 61 features to use in our classification task.

3. Cross-validation

Findings:

Our cross-validation strategy was of unanimous agreement. We decided to do a Stratified K-Fold cross-validation due to the imbalance in our label distribution. Our main concern was that by choosing another cross-validation strategy, such as K-Fold, we might end up with no samples of a underrepresented label in the validation set. With stratified K-Fold, the similar label distribution that is seen in the entire dataset is also applied to training and validation. Meaning that throughout all 5 folds the same proportion of observations from each label is used, therefore, avoiding any generalization problems.

Whole dataset

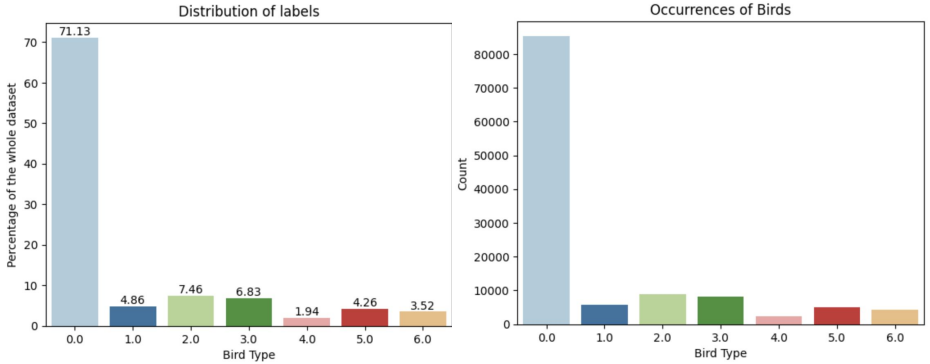


Figure 7.

One fold

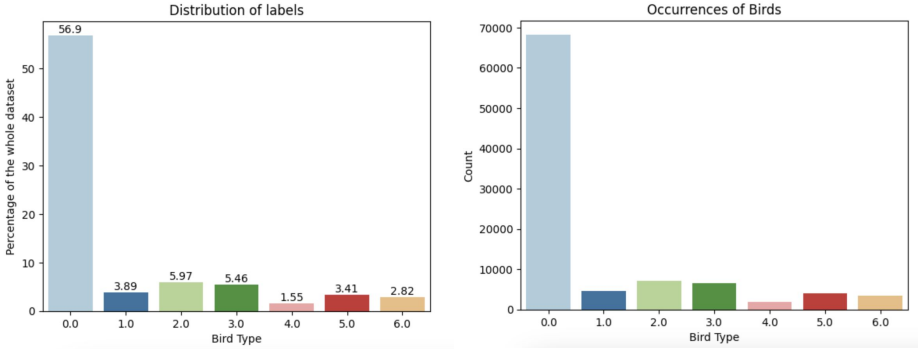


Figure 8.

3. Baseline Performance And What We Can Expect

For the baseline performance, we used the majority class baseline, which predicts the most occurring class in the training data.

Since we have a heavily imbalanced dataset, with 85 358 occurrences of the label “0”, and only 34 642 occurrences of non zero in our annotated label data.
Comparing Accuracy - which we did not seem suitable for evaluation - a classifier, which only predicts “0” would have a accuracy of 71.1 %, predicting only individual labels would yield:

- a 4.86 % Accuracy for “comcuc”
- a 7.46 % Accuracy for “cowpig1”
- a 6.68 % Accuracy for “eucdov”
- a 1.93 % Accuracy for “eueowl1”
- a 4.25 % Accuracy for “grswoo”
- a 3.51 % Accuracy for “tawowl1”

The **baseline performance** we can expect if we predict on test data, if we only predict “0” is

- a weighted average **f1-score** of 59 %
- a weighted average **precision** of 71 %
- a weighted average **recall** of 51 %

4. Classification

As mentioned in the beginning we needed to address the label imbalances during the training method and also during the measuring methods so our workflow is the following: We scaled our features with **StandardScaler** in order to prepare our data for **PCA**. As previously mentioned, we used PCA with 111 components, which explained over 90% of the feature data variance. In order to select the best parameters we used **GridSearchCV** with **Stratified-K-Fold** cross validation.

Thereafter, as previously mentioned, we tested the following two approaches in order to find the best classifier:

1. We used the feature set of 111 dimensions
2. We eliminated 50 features which had less than 2% correlation with the label data.

To compare model efficiency for each set of features, we used K Nearest Neighbors as baseline. As we will see in the next few slides, by further reducing our feature dimensionality to 61 features, we lost a bit of information. As a consequence, our classifier began to misclassify labels more often.

Having set the 111 features as the new baseline, we tested different classification models. As we will see, Random Forest performed best, after the first training we obtained the feature importances so to give it a final refinement, we ranked the best performing features in the Random Forest classifier. From that ranking, we isolated the top 50 best features. With the new feature set we retrained the RandomForest which lead us to the final results for this model.

4.1. K Nearest Neighbors

K Nearest Neighbor is used for pattern recognition. KNN is simple and intuitive, this makes using the model very easy. It does not make any assumptions about the underlying data distribution, therefore making it suitable for a lot of applications. However the model is rather computationally expensive.By using **GridSearchCV,** we found that the ideal set of parameters were the 3 nearest neighbors, which we use hereafter for all KNN classifiers. Here we compare both methods that were tested.

Table 1. Prior to eliminating features with little-to-no correlation with labels (111 components)

	0. other	1. comcuc	2. cowpig1	3. eucdov	4. eueowl1	5. grswoo	6. tawowl1
f1-score	0.898	0.462	0.528	0.480	0.384	0.348	0.600
precision	0.85	0.508	0.574	0.570	0.534	0.588	0.720
recall	0.948	0.426	0.494	0.414	0.302	0.250	0.510

Table 2. After eliminating features with little-to-no correlation with labels (61 features)

	0. other	1. comcuc	2. cowpig1	3. eucdov	4. eueowl1	5. grswoo	6. tawowl1
f1-score	0.892	0.362	0.474	0.408	0.258	0.450	0.386
precision	0.854	0.384	0.500	0.494	0.420	0.556	0.452
recall	0.934	0.338	0.452	0.350	0.188	0.376	0.336

Tables 1 and 2 show the results from our KNN classifier (k = 3) using both feature datasets for comparison. In table 1, we train our model using the 111 feature components mentioned in previous slides and in table 2, we have the model trained using the remaining 61 features after the previously presented analysis. As expected, for both methods, we observe a higher overall precision in the classification of zeros. This is a pattern that should be observed in every model since zeros are overrepresented in the label data. The classification of other labels is not as effective, especially for labels which are slightly more underrepresented than others. However, label classification using the 111 components seems to be more efficient.

4.1. K Nearest Neighbors

We use confusion matrices to analyse the level of misclassification for each label. From the previous slide, we had the impression that our first approach which was based on the 111 feature components from the PCA dimensionality reduction was more efficient in predicting non-zero labels. This impression was confirmed by our confusion matrices, which show that not only the classification of non-zero labels was more accurate with the 111 components, but also the prediction of zeros. As we see in figures 9 and 10, for every single label, there was less misclassification in the 111 component PCA approach.

Method 1: 111 Feature Components

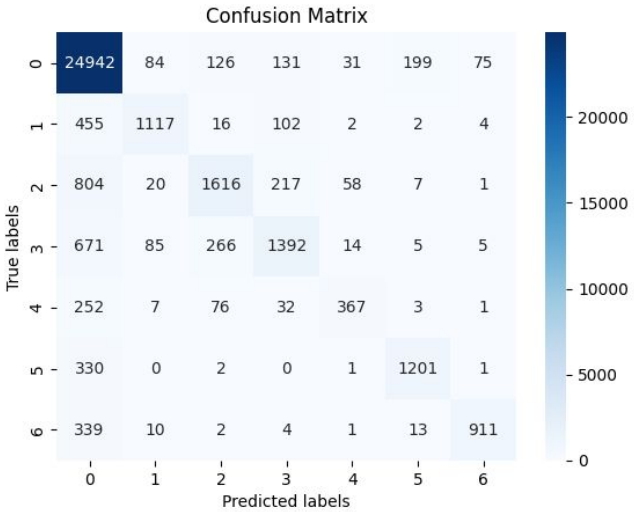


Figure 9.

Method 2: 61 Features

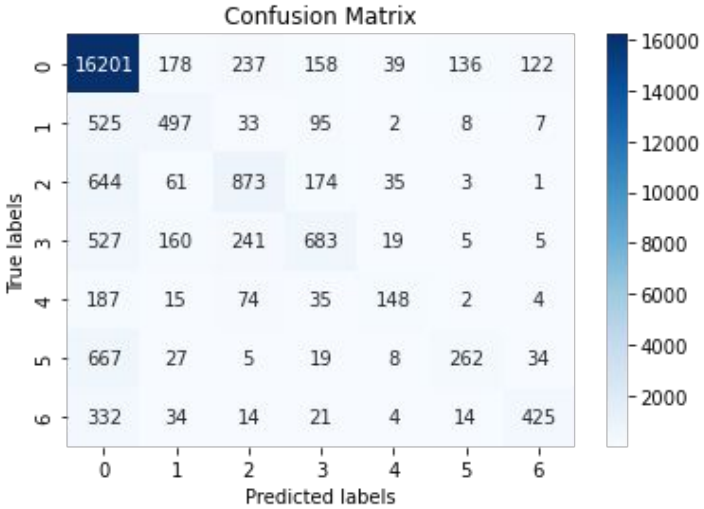


Figure 10.

4.2. Quadratic Discriminant Analysis (QDA)

The Quadratic Discriminant Analysis is a generative models that assumes that each class in the data follows a Gaussian distribution. If we consider that, following the law of large numbers, as sample size grows, it's mean gets closer to the population average and data distribution tends towards a Gaussian distribution, then this method could be very suitable for classification. However, since our label data is unbalanced, this approach might be useful for the overrepresented zero data, but it is not suitable for smaller samples.

Classification Report

Table 3 clearly confirms what was previously argued. We see that classification is fairly efficient for zeros and labels which are slightly more represented. Nonetheless, it is not a good alternative for predicting all labels in our data, since it is not very useful for underrepresented labels.

Table 3. Classification Report Using QDA (111 Feature Components)

	0. other	1. comcuc	2. cowpig1	3. eucdov	4. eueowl1	5. grswoo	6. tawowl1
f1-score	0.870	0.568	0.726	0.736	0.734	0.713	0.750
precision	0.962	0.414	0.642	0.696	0.542	0.594	0.674
recall	0.794	0.906	0.842	0.784	0.734	0.884	0.848

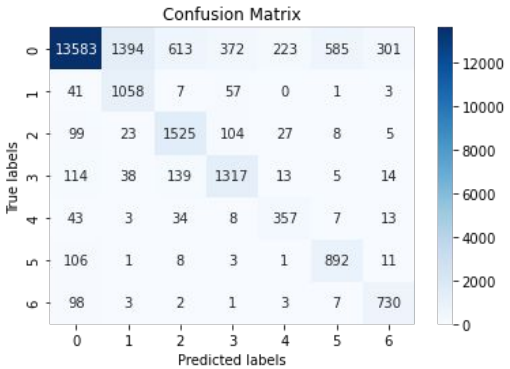


Figure 11.

Confusion Matrix:

Figure 11 shows the confusion matrix for this classifier. We see that it also presents difficulties predicting with misclassification of labels. Zeros are majoritively correctly classified, yet 'comcuc' is more often classified as 'no bird' or zero than as it's own label. This pattern of often classifying labels as zeros is seen in this model across all labels.

4.3. Naive Bayes

Naive Bayes classifier are simple, efficient and effective - given that the independence assumption holds. They also can handle high-dimensional datasets quite well. Intuitively, this classifier is not the best option for us, because the “naive” assumption of the feature independence between each other might not hold.

Classification Report

We have a high precision for the “0”-label, this is kind of expected, also we can see, that the model overall performed poorly in classifying the “bird-calls”, which speaks for our Intuition.

Table 4. Classification Report Using Naive Bayes (111 Feature Components)

	0. other	1. comcuc	2. cowpig1	3. eucdov	4. eueowl1	5. grswoo	6. tawowl1
f1-score	0.6	0.334	0.212	0.358	0.104	0.470	0.442
precision	0.904	0.476	0.122	0.412	0.374	0.392	0.580
recall	0.452	0.256	0.780	0.316	0.060	0.586	0.360

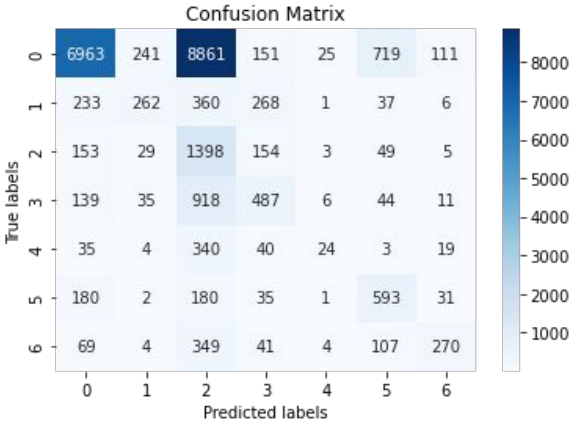


Figure 12

Confusion Matrix:

Figure 12 shows the confusion matrix for the naive Bayes. Nn the diagonal we see the correctly predicted labels. This model interestingly classifies the “2”-label as “0” very often, this may be to the fundamental assumptions of the model.

4.4. Random Forest: 111 Features

Random Forest is an ensemble learning method that combines multiple decision trees to classify. We chose this method, because it is known for its robustness and we personally like this classifier. Also Random Forest models can handle large and high-dimensional datasets really well, as well as it is very robust to overfitting.

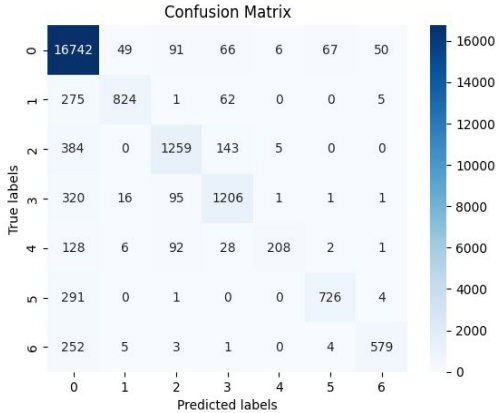
Classification Report:

From all of the models, the Random forest performed the best. In every other model, we had issues regarding the classification of non-zero labels, here these issue are much smaller.

Table 5. Classification Report Using Random Forest (111 Feature Components)

	0. other	1. comcuc	2. cowpig1	3. eucdov	4. eueowl1	5. grswoo	6. tawowl1
f1-score	0.946	0.802	0.75	0.762	0.596	0.796	0.786
Precision	0.912	0.904	0.822	0.780	0.950	0.912	0.912
Recall	0.980	0.722	0.690	0.746	0.434	0.708	0.692

Figure 13.



Confusion Matrix:

Figure 13 shows the Confusion matrix for the Random Forest with 111 features. Along the diagonal we see the accurately predicted labels. We see, as in all other models, a high accuracy in the classification of zero-labelled data. However, we also see much lower misclassifications across all labels. Compared to QDA or Naive Bayes, for instance, where each had a problematic label which was being inaccurately classified, here that issue appears to be handled. This model also outperforms KNN in label prediction accuracy. That is the reason why we decided to set this model as our baseline.

4.4. Random Forest: Top 50 Features

Based on the GridSearch we found that setting **n_estimators** parameter to 100 yields the best results. After the training we checked the feature importances in the model and we retrained the model with the top 50 important features and we ended up getting better results than before.

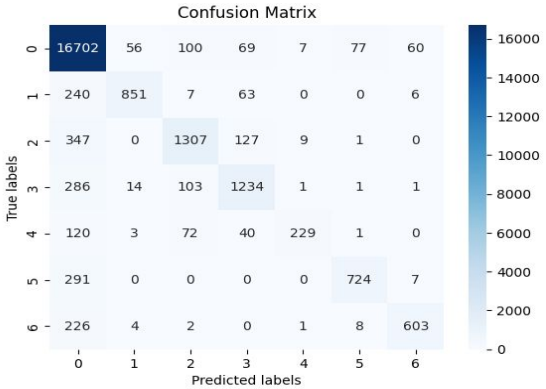
Classification Report

By selecting the best 50 features based on the feature importances we managed to improve our performances a bit while saving capacity. In table 6, we see that precision in the classification increases across all labels.

Table 6. Classification Report Using Random Forest (50 Best Features)

	0. other	1. comcuc	2. cowpig1	3. eucdov	4. eueowl1	5. grswoo	6. tawowl1
F1-score	0.950	0.818	0.764	0.770	0.628	0.786	0.792
Precision	0.920	0.904	0.818	0.788	0.932	0.896	0.896
Recall	0.980	0.748	0.720	0.748	0.474	0.700	0.710

Figure 14.



Confusion Matrix:

By isolating the top 50 features, we see, as previously mentioned, a slight improvement in performance. Misclassification falls across all labels, for some, such as label 2 (“cowpig1”), 48 labels are accurately classified additionally, for others, such as label 4 (“eueowl1”), the improvement is of only 11 more correctly classified labels. Nonetheless, there is a clear improvement.

4. Classification: Overfitting

We used **Stratified K-Fold** for cross-validation which helps to prevent overfitting by ensuring that the labels are proportionally represented during training and validation in model evaluation. We also specifically chose the weighted f1-scores as an evaluation metric, in order to judge our models for overfitting. During training, the f1-score is used to track the performance, and thus overfitting can be detected and appropriate measures can be taken. Low f1-scores mean poor performance which may indicate overfitting.

We are unsure whether lower f1-scores in some of our models are explained by problems of overfitting. Since our strategy was the same across all models, whenever there was a specific label in a model that showed low f1-scores, we attributed it mainly to issues with the model or to sample size (meaning label representation).

Our main focus was our **Random Forest** classifier which was our top performer. There we see consistently high f1-scores across all labels. It suggests that we prevented the classifiers from excessive overfitting. Based on the Confusion Matrices, we could perhaps also judge the overfitting, since we observe the individual labels predicted by the model. Therefore, we can look for any patterns that point towards overfitting. We do not see a unanimous distribution of heavy outliers across models that we can call a pattern.

Although some models, such as Naive Bayes and QDA struggled with a specific label, that was not a pattern. So we would argue, it may be attributed rather to an issue such as a condition of feature independence that does not hold and may cause a model to predict the same label more than the other classifiers. Such issues decrease the performance of the classifier significantly which can be seen in the classification report.

5. Conclusion

1. The main challenges of this task were not the classification and modelling themselves. It was the label imbalance and the high feature dimensionality.
2. To tackle these issues, we first reduced feature dimensionality as far as we could in the way we agreed would lead to the lowest lost of information.
3. We first dropped from the feature data the features that had high correlation among themselves. After that, we scaled our features with **StandardScaler** in order to prepare our data for **PCA**. Then while preserving 90% of feature data variance, we reduced feature dimensionality with **PCA** to only 111 components.
4. We attempted to further reduce dimensionality by looking into feature-label relationships, but this approach was proven not to be efficient. As a note, we apply **StandardScaler** in order to avoid label data imbalance to affect our later modelling.
5. To train different models using these 111 components and our label data, we used **Stratified K-Fold** as our cross-validation strategy. We chose **Stratified K-Fold** specifically as to avoid that label data imbalance distorted the label data distribution across our 5 folds. We confirmed that data was correctly distributed across all folds by comparing the label distribution in each one with the distribution of the entire dataset.
6. Our conclusion after modelling and testing was that **Random Forest** was the most suitable model for our task and feature selection strategy. All other models we tested generated less efficient results.
7. Finally, in order to give our **Random Forest** model a last push, we isolated the 50 most important features for our model and used them to retrain it. This drove classification results up to more satisfying levels.
8. Overfitting was handled in model evaluation. Through detailed evaluation processes, we were able to check for classification imbalances from model overfitting.