Machine Learning and Pattern Classification

Team Report

Assignment 4: Challenge

Team Aberrant

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

Outline

In this final stage, we are given completely new data for validation of our classification model. The idea is to put our model to test and see whether it was designed to overfit the training data or whether it is actually a strong classifier.

In this report, we will show the results of our classification model with this new data prior to any changes and calibrations (ex-ante performance) and after it has been calibrated. This report has been structured in the following way:

1. Data Split
2. Ex-ante Classifier Performance
3. Calibrations
4. Ex-post Classifier Performance
5. Discussion and Conclusion

In the final section, we review the performance evolution of each of our tested approached and discuss the role of overfitting in our work for this and the previous assignments.

1. Data Split

Precautions:

An important first step in dealing with the new data was to make sure to avoid using different parts of the same recording file in the different sets.

In the training of our model for the previous assignment, we erroneously loaded our data into one single large dataframe, concatenating all rows together before splitting it into training and test. This led to a very poor first model performance.

To deal with the issue, we implemented a function that splits the data by files. That way, we were able to avoid the overfitting that could be caused by including rows from the same recording in the train and test set.

2. Ex-ante Model: Feature Selection

Feature Selection:

To recap how our original model was designed and planned, in this slide, we go through our initial feature selection strategy, which was two-fold:

1. Eliminate features with inter-feature correlation of over 90%

2. Apply PCA to the remaining features

By looking into the correlation of features with each other, we were able to reduce their number by about a half. We were left with 265 features. PCA as seen in the previous report, showed that we could explain 90% of the feature data variance with 111 feature components from the 265 previously filtered in the first stage.

Finally, we applied GridSearch on the remaining 111 feature components to select the 50 features that were most important for a Random Forest ensemble method. Those were the features that we selected.

2. Ex-ante Model: Random Forest

Result

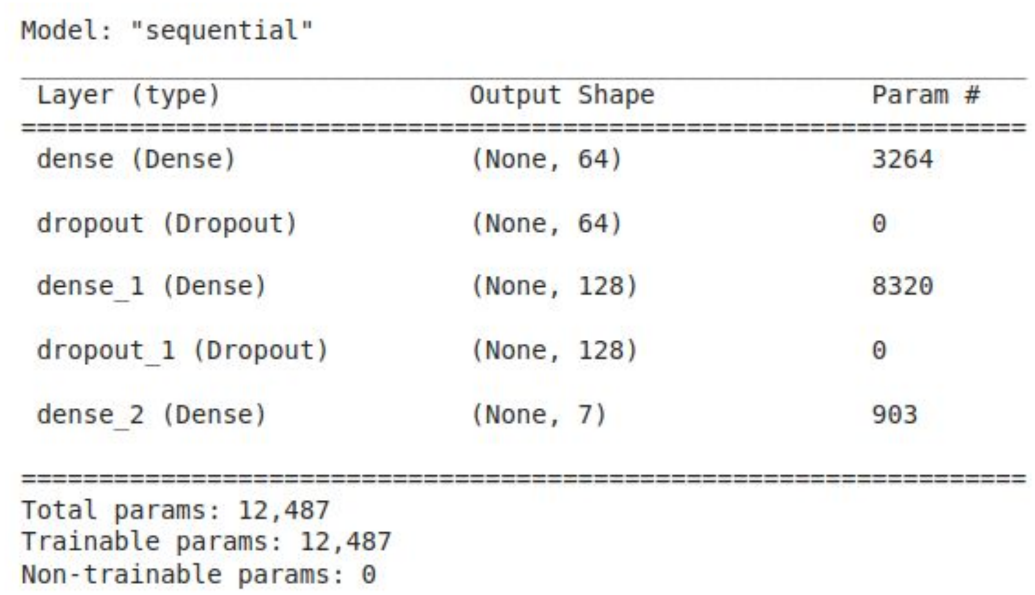
For classification, we relied on a Random Forest model with 100 trees and the previously mentioned 50 features. That was the best performing classifier in our previous assignment.

By applying this model to the new dataset in the challeng server, we were able to save 6.156,38 Euros, which was a relatively poor performance. Previously, on the training dataset, our model had shown quite strong results. This is a clear indicator that our model, in the way we had trained it, was slightly overfitting the training data.

3. Calibrations

Multiple Layer Perceptrons (MLP):

We wanted to experiment with Neural Networks and establish a baseline performance before making large changes and calibrations. Multi Layer Perceptrons are the simplest form of a Neural Network. So, we decided to begin there.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                3264

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 128)               8320

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 7)                 903

=================================================================
Total params: 12,487
Trainable params: 12,487
Non-trainable params: 0
```

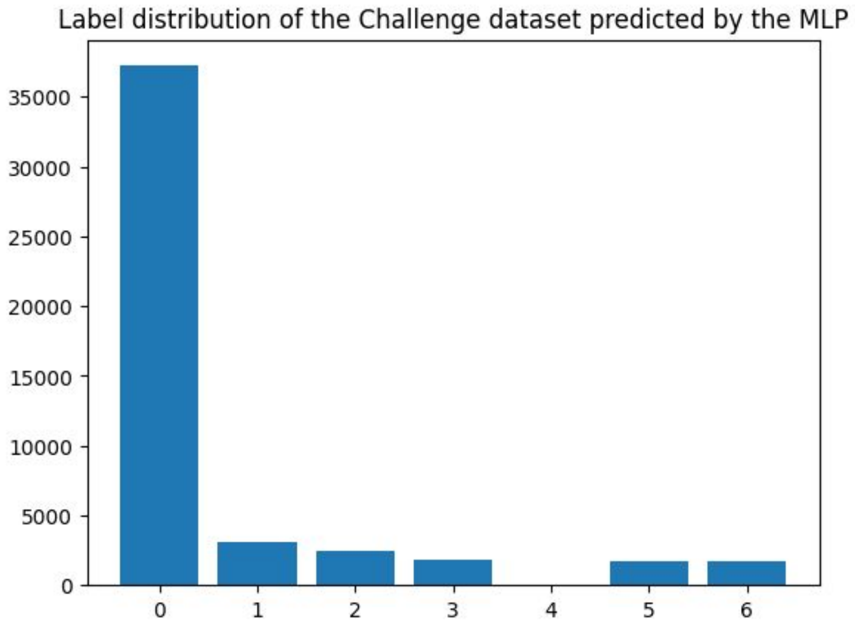
Performance:

Not surprisingly, by using only a Feed Forward Neural Network and MLP, the result was very weak.

In the challenge server, this basic model rendered us only a 3.954,48 € score.

3. Calibrations

Figure 1.

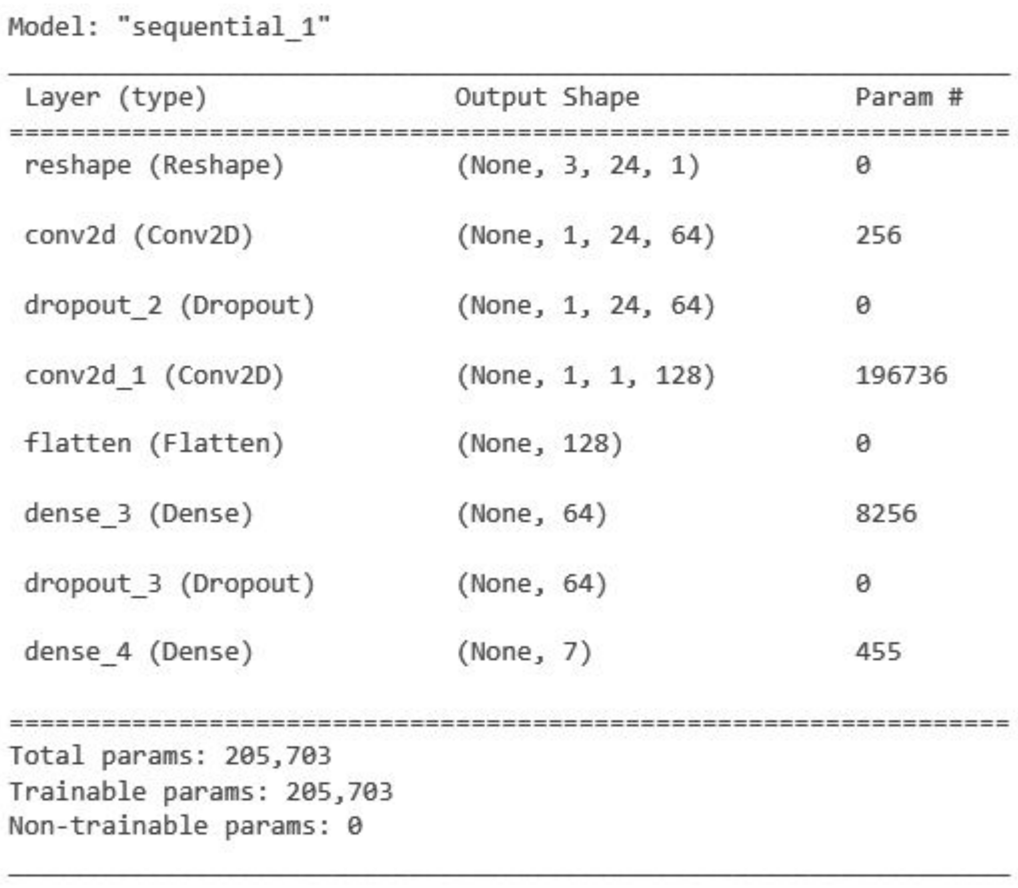


Performance:

Figure 1 shows the label distribution of the predictions of the MLP model.

We see that this model misclassified all eueowl1 observations (class 4). That is a very poor result which we attempt to improve by going one step further and using Convolutional Neural Networks.

3. Calibrations



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 reshape (Reshape)           (None, 3, 24, 1)          0

 conv2d (Conv2D)             (None, 1, 24, 64)         256

 dropout_2 (Dropout)         (None, 1, 24, 64)         0

 conv2d_1 (Conv2D)           (None, 1, 1, 128)         196736

 flatten (Flatten)           (None, 128)               0

 dense_3 (Dense)             (None, 64)                8256

 dropout_3 (Dropout)         (None, 64)                0

 dense_4 (Dense)             (None, 7)                 455

=================================================================
Total params: 205,703
Trainable params: 205,703
Non-trainable params: 0
_________________________________________________________________
```

Convolutional Neural Networks:

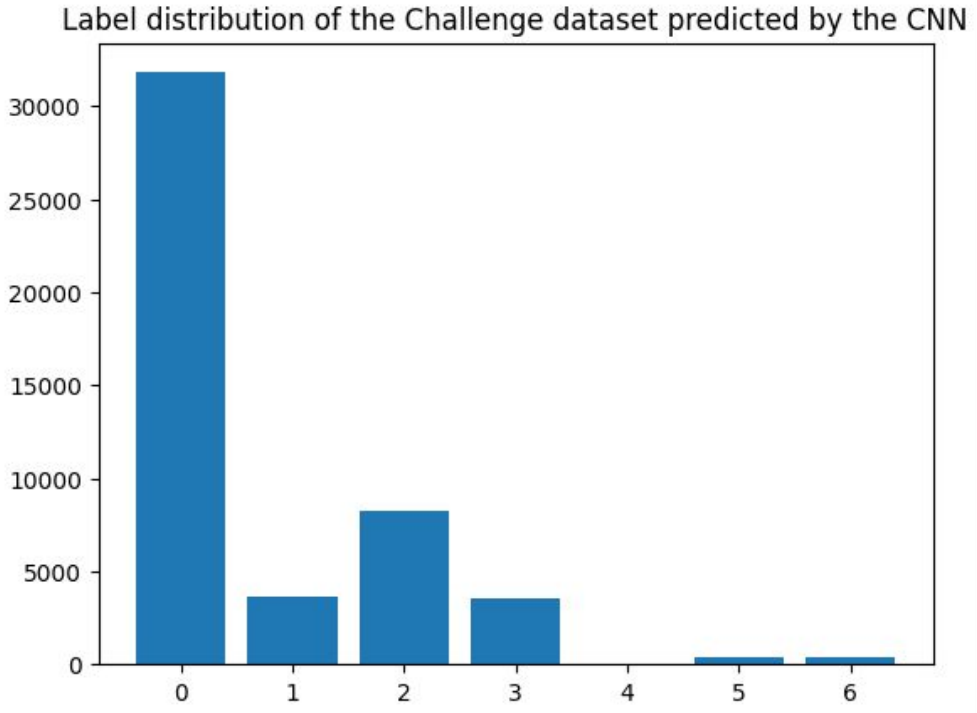
Convolutional Neural Networks or CNN are most typically used in image classification, but can also be applied to other classification tasks. We tried a Convolutional Neural Network as an attempt to assess non-linear relationships in our data.

Architecture:

The figure to the left shows the layers of our Convolutional Network

3. Calibrations

Figure 2.



Performance:

Unfortunately, we did not get good performance for this model either. It presented only a very slight improvement in performance compared to the baseline set by the previous approach. Hence, we did not move forward with this approach. In the challenge server, this model was able to save only 3.996,47 €.

Figure 2 shows the label distribution of the predictions of the CNN model without the use of the under-sampling function. As we can see, the model overfitted heavily. We had different versions, in which the label distribution looked more balanced, which we achieved by applying regularizers and increasing the dropout, however this was still not enough.

3. Calibrations

Random Forest Classifier with Balanced Class Weights:

Experimenting with new models led us back to our original one: the Random Forest classifier. We experimented with it, now using new feature selections and new parameters.

1. Class weights: Instead of training the model with unbalanced data, we used the inherent "class_weight = 'balanced'" option.
2. Feature Selection: Similarly to what we had done before, we took the features that had an inter-feature correlation of under 90%. Then, we selected the 22 features there were most important for our model.

After these changes, we obtained a slightly better performance score in the challenge server. We were able to save 7.415,46 Euros with this setup. Nonetheless, this was still not an ideal performance. So as a final approach, we decided to experiment with Gradient Boosted methods using this same setup.

3. Calibrations

Gradient Boosting Classifier:

The Gradient Boosting classifier is a functional-gradient-based algorithm that aims at minimizing the loss function of negative gradients. Our main concern when using a very restricted number of feature parameters is generalization, which can produce a highly biased classifier. This method targets bias error, which was one of the reasons why we chose to work with it. It is constructed out of a combination of several learning models. It is quite popular for its speed and accuracy.

Using the same setup as in the previous Random Forest Classifier, the Gradient Boosting classifier delivered the best result we had seen so far. In the challenge server, it saves 8.234,98 €.

Another concern was the influence of the unbalance in label data on performance. As said, we had issues with misclassification, especially, with zeros being labelled as birds. So in order to improve the performance of this classifier, we go through a series of label balancing methods.

3. Calibrations

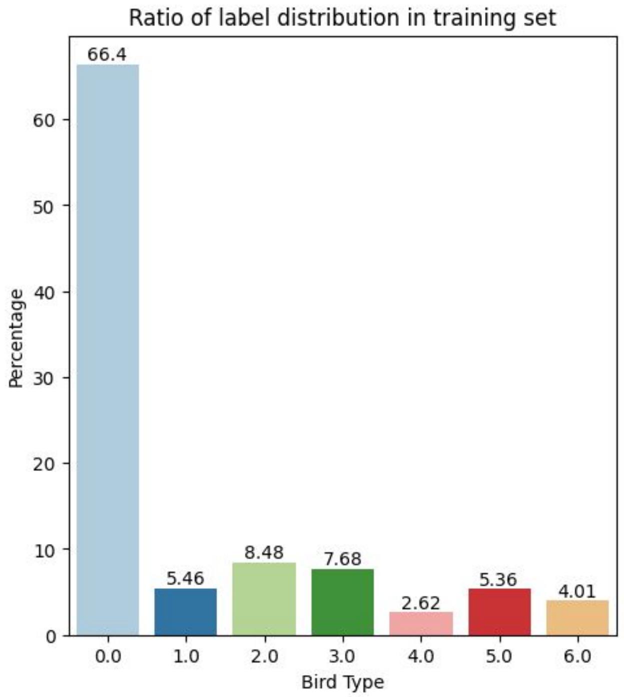
Similarly to what we had already encountered in the previous assignments, labels in this dataset were very unevenly distributed with a concentration of zero-labeled data of over 70%. In order to prevent what we experienced in the previous assignment with zero labels being accurately classified while all others were underrepresented in comparison, we decided this time to reduce the number of zeros.

Annotation Disagreement:

Our first approach was then to identify disagreement in the annotation of zero labels so that we could reduce that concentration. Whenever agreement among annotators was not unanimous, we dropped the row.

By doing so, we were able to reduce the percentage of zero-labeled data to 66.40% as seen in figure 3.

Figure 3. Label Distribution After First Elimination of Excessive Zeros



3. Calibrations

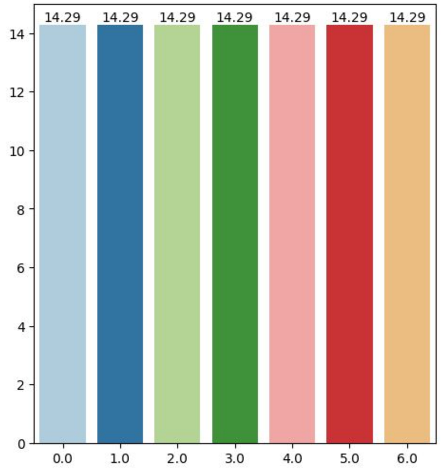
However, our new outcome was not as satisfying as we had wished for. There was still a very high concentration of zero labels, which did not significantly improve our classification results.

Undersampling Of The Majority Class:

Consequently, we decided to try out more radical label balancing approaches. Our most successful strategy was the application of a Random Undersampler to evenly distribute the labels in the validation set. The training set was left untouched with the same class distributions seen in the previous slide.

We are aware that it reduces the size of our validation dataset quite dramatically and also discards not only zero-labeled data in the process of balancing. However, it helped to tackle misclassification.

Figure 4. Label Distribution After Undersampler



4. Ex-post Classifier

Gradient Boosting Classifier With Balanced Data:

After balancing the data as described in the previous slides, we test again our Gradient Boosting Classifier with the set of 22 most important features and it delivers us our best performance.

In the challenge server, this model saves 8.829,73 Euros which is not a large improvement, but still a quite strong performance when compared to the models we tested before.

5. Conclusion

In the table to the right, we see the evolution in the performance of your different classifiers.

Our attempt at using Neural Networks did not deliver as promising results as we had expected.

We see that our initial Random Forest was relatively strong, only its feature selection was not ideal.

Gradient Boosting worked quite well, slightly better than Random Forest.

Model	**Challenge Server Score**
Random Forest (50 features)	6.156,38 €
MLP Neural Network (50 features)	3.954,48 €
Convolutional Neural Network (50 features)	3.996,47 €
Random Forest (22 features)	7.415,46 €
Gradient Boosting (22 features)	8.234,98 €
Gradient Boosting (22 features/balanced data)	8.829,73 €

5. Conclusion

One major concern, when developing a classifier is overfitting it to the train and test set. We try to lapidate it choosing the perfect set of parameters and balancing out the class distribution of the data and unintentionally may end up with a model that cannot be put into practice to classify real-world data. The tradeoff between oversimplifying (generalising) and overcomplicating (overfitting) was the largest challenge in this set of assignments.

There are ways to test models for overfitting. The most efficient perhaps would be to divide data into three: train, test and validation. Then to train the model using only the first two sets, so that after we are sure to have found the perfect model and setup, we can put it to test on data that is unknown.

We did not do this in the previous assignment, such that we were not really sure as to what we would discover about our model in this step of the process. This final challenge served to prove whether our classifiers were fit to use or overfitted.

Our very initial model, which we here call 'ex-ante' did not perform very well. Its performance was not completely catastrophic. However, having seen how well it was classifying data in the previous dataset, we were surprise to learn that, in a real-world scenario, it would not have performed as well as it did in our training dataset.