MLPC Report - Task 3: Classification Experiments

Team OBSERVE

Johannes Grafinger **Jonas Gantar** **Leonhard Markus Spanring** **Reinhard Josef Pötscher**

Contributions

As a group a solution approach was discussed for each question, then the questions were answered by individuals or sub-groups. Task 1 was completed by Jonas and Johannes. Task 2 was implemented by Johannes and answered by Reinhard. Task 3 was attempted by Jonas, Johannes and Reinhard independently and then answered by Johannes as he produced the best results. Task 4 was answered be Reinhard. Task 5 was discussed as a group and completed by Leo (including training the classifiers). Task 6 was completed by Johannes and Leo. Collaboration was done over GitHub and project progression was tracked through regular stand-ups.

1.) Labeling Function: For your analysis, you may focus on a subset of the 58 classes provided.

For subtasks b.) and c.), we focused primarily on the following randomly selected subset classes: Airplane, Bird Chirp, Power Drill, Cowbell, Siren, Trumpet, and Rooster Crow.

a.) Assess how accurately the applied labeling functions capture the intended classes. Do the mapped classes correspond well to the free-text annotations? Are the labeled events clearly audible within the indicated time regions?

To evaluate the accuracy of the applied labeling function, the free text annotations were analyzed by applying an algorithm that counted the frequency of semantically meaningful words[1] in each annotation (Table 1). The resulting list of most common words per label clearly showed that the labeling function captured the intended classes with a high accuracy; e.g. all words related to dogs (dog, barking, growling) were clearly captured under a single label. Indicating that the labeled events are clearly audible within the indicated time regions. This was further proven by checking random samples that aligned with the given labels.
This was further confirmed by laying a random selection of labels over a clustered down-projection of the audio features (see section 1.), c.)).

b.) Which audio features appear most useful for distinguishing between the classes of interest?

In order to answer this question, we first of all randomly collected 1000 frames from the dataset (which represents a collection of all frames from all files) for each class-label where it is positive, resulting in a total of around 7000 frames (a bit less, since some frames contain multiple of the classes). For each frame, we additionally include the previous and next two frames as context length by simply concatenating them with the features from the central frame (in temporal order). Each class therefore has 1000 vectors of shape 5 * 942 = 4710, where 942 represents the summed number of dimensions for all features.

For each class, we then estimated the Mutual Information between the label vector and all features dimensions. For each feature, we then calculated the average Mutual Information over all feature dimension corresponding to it. This gives us a measure for how informative the feature generally is for predicting the label correctly, which also captures nonlinear relationships. We then normalized the values per label, resulting in the first plot of Figure 1. However, since

[1] excluded words: 'a', 'an', 'and', 'are', 'as', 'at', 'by', 'from', 'in', 'is', 'is', 'it', 'noise', 'of', 'on', 'sound', 'the', 'then', 'to', 'with'

Table 1: Top 3 Annotation Keywords per Label

Class	Keyword 1	Count	Keyword 2	Count	Keyword 3	Count	Total per Label
Alarm	alarm	359	beeps	110	clock	78	547
Beep/Bleep	beep	251	beeps	231	beeping	193	675
Car	car	741	engine	333	driving	179	1253
Speech	speaking	1166	talking	602	man	608	2376
Hammer	hammer	259	hammering	147	metallic	98	504
Siren	siren	351	police	80	ambulance	61	492
Bell	bell	655	rings	442	ringing	231	1328
Shout	shouting	187	loudly	208	screams	130	525
...	...	...	...	...	...	...	...
Hiccup	hiccups	36	baby	19	high-pitched	17	72
Jackhammer	jackhammer	121	construction	67	loud	46	234
Sewing Machine	sewing	78	machine	78	mechanical	20	176
Cowbell	cowbell	65	cowbells	44	ringing	31	140
Pig Oink	pig	127	grunts	72	snorts	23	222
Chainsaw	chainsaw	232	running	48	cutting	31	311
Washing Machine	washing	176	machine	176	water	23	375
Power Saw	saw	138	cutting	72	circular	43	253

this approach cannot capture effects features only have in combination with one another, we also tried fitting a simple Random Forest Classifier on the data and extracted its feature importances, resulting in the second plot of Figure 1.

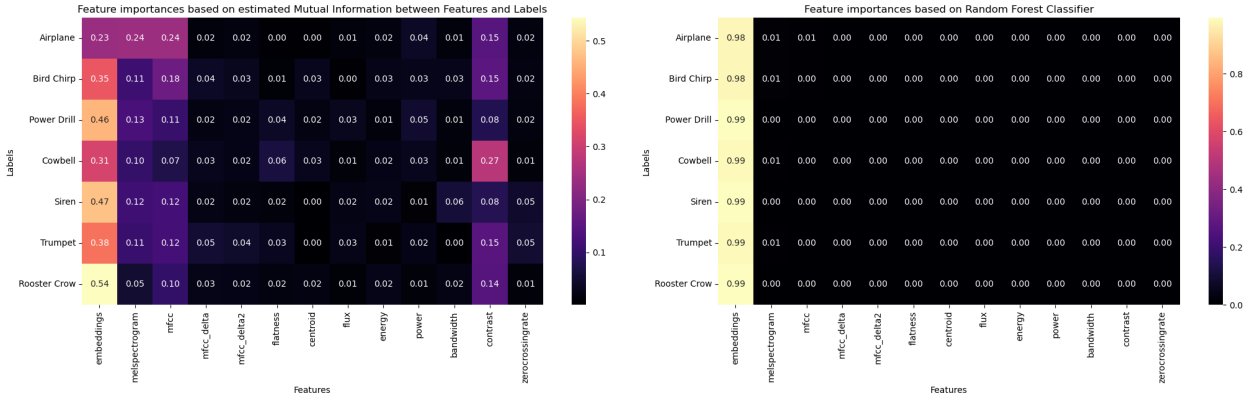

Figure 1: Estimated feature importances as by Mutual Inforamation and Random Forest methods.

The features 'embeddings', 'melspectrogram', 'mfcc' and 'contrast' all seem to be pretty useful according to our Mutual Information graphic, with 'embeddings' being the most important one. However, the fact that the Random Forest almost exclusively relies on the embeddings might suggest heavy correlations between them and the other features. Which does make sense, considering many of the other features are usually used for creating such audio embeddings.

c.) How well do the chosen audio features group according to the discretized class labels? Do samples of the same class form tight clusters?

Using the four features mentioned above, we reduced the high-dimensional data to two dimension using T-SNE and plotted the results in Figure 2. Note that for examples corresponding to more than one of the classes, we randomly chose one for coloring.

Overall, the results look very reasonable. Clusters are identifiable and examples within mostly have the same attributed class. Some overlaps are of course inevitable, considering the relatively high complexity of the dataset.

Figure 2: Downprojected examples using selected features and their class correspondences.

2.) Data Split

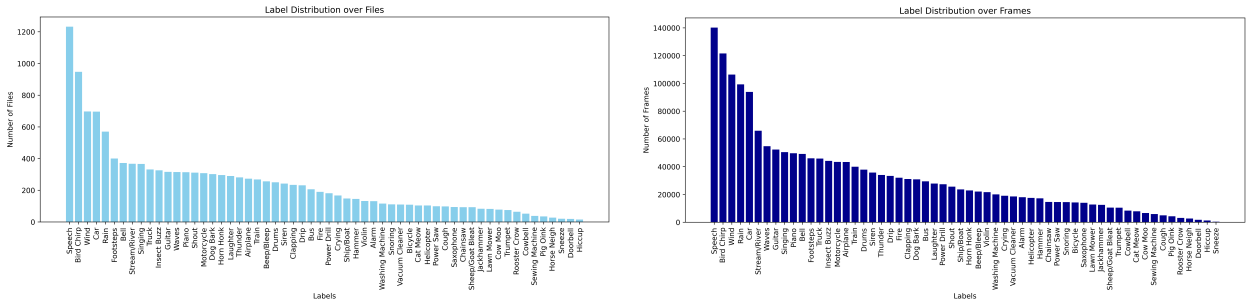

Figure 3: Label distribution

a.) Describe how you split the data for model selection and performance evaluation.

To ensure a robust raining and proper model evaluation the available data set was independently split into three separate sets: Taring Set, Validation Set and Test set.

1. **Training Set:** The Training Set was used to train our classifiers. Training Set size was 60% of our total data.
2. **Validation Set:** The Validation Set was used to estimate the performance of our model during training and to adust hyperparameter, while helping us to detect potential overfitting. Validation Set size was 20% of our total data.
3. **Test Set:** The Test Set was used to evaluate the performance of our final model and estimate how well it would perform on unseen data. This Set was unseen by our classifiers prior to the final evaluation. Test Set size was 20% of our total data.

Using Cross-Validation was considered but ultimately not used as the provided data-set was large enough to have a low risk if underfitting and produced an overall satisfying results without the implementation of Cross-Validation.

We first loaded all the audio data with the features and the labels and split it into individual frames.

b.) Are there any potential factors that could cause information leakage across the data splits if they are not carefully designed? If yes, how did you address these risks?

Yes, there are **critical risks of information leakage** in audio classification if splits are not carefully designed:

- **Common Sources of Leakage:**
  Temporal close audio segments: If data is split by audio segments and not by full audio recording/session, temporal close audio segment can end up in different data sets. Because these segments will be closely related or even almost identical this would introduce data leakage to the data splits.

Same recording conditions: If the same noise/audio source appears in both training and test sets, the model may learn source-specific features rather than class-specific ones. This can occur data from the same recording ends up in different data-sets.

Improper data splitting: If data splitting is not done carefully, duplicates of the same sample might be included in multiple Sets causing data leakage.

Leakage though data augmentation: If data augmentation is applied before the data split and thus based on the entire dataset, this can result if information from the Test Set being included in the Training Set, causing leakage.

- **Mitigation Strategies:**

  File Splits: We split the data not over the audio segments but over the full audio files. This ensures that temporally close segments and segments with the same recording condition are grouped into the same data set.

  Datasplit checking: All data samples were indexed prior to data split and checked afterwards to ensure uniqueness among the samples.

  Careful Preprocessing: We avoided any preprocessing (e.g. feature normalization) that used statistics from the full dataset — we used only training data statistics.

To avoid possible problems when splitting into the three data sets, we only split at file boundaries and tried to have an approximately equal distribution in all parts (stratification) (Figure 3). We worked in frames during pre-processing (normalization).

c.) Describe how you obtained unbiased final performance estimates for your models.

Two main aspect were considered during model development to ensure a unbiased final performance estimation:

1. **Leakage minimization:** The prior mentioned data leakage techniques were carefully implemented to ensure a clear separation between the individual data-sets.
2. **Strict Test Set Separation:** The test set was never used during training or validation. Final model evaluation was performed only once on the Test Set after all model development was completed.

3.) Audio Features

a.) Which subset of audio features did you select for your final classifier? Describe the selection process and the criteria you used to make your choice.

After reflecting on the results from section 1.), b.), we have decided to run our experiments on the 'embeddings' feature as well as the other three features mentioned in that section. Even though the results imply heavy correlations and possible redundancies to the other three features - especially considering the possible creation process of these audio embeddings - this still offered a slight increase in performance for tests we ran on a small scale. Performing all tests again using only the 'embeddings' feature would be computationally unfeasible, so we decided to play it safe.

Unfortunately we were not able to include context frames in our experiments as we did earlier, since the computational complexity was already extremely high and this would only have exaggerated the problem.

b.) Did you apply any preprocessing to the audio features? If so, explain which techniques you used and why they were necessary.

Preprocessing is an important step for ensuring consistency in the data and making the lives of models like Support-Vector-Machines much easier. We mainly normalized the data in order to to bring every example to a similar scale magnitude-wise, which generally improves performance and convergence rates.

We tried normalizing the data globally using the global mean and standard deviation from only our training set (to avoid data leakage), normalizing the features frame-wise to have unit mean and variance as well as a combination of these. In the end, doing only the latter proved to be the most performant, so that is what we did.

4.) Evaluation

a.) Which evaluation criterion did you choose to compare hyperparameter settings and algorithms, and why?

In terms of performance metrics we decided to record - per class label - both the Balanced Accuracy as well as the F1-score of our models predictions. As single measures of performance of a model, we then used the macro-averaged Balanced Accuracy / F1-score (over the 58 class labels), like was shown in the tutorial session.

Balanced Accuracy, defined as the average recall per class (always binary in our case) obviously gives a more balanced performance metric, focusing on both the positive and negative classes. This metric was mainly included in order to be able to be able to compare the models to the baseline perfomance (see section 4.), b.)).
The F1-score on the other hand is most likely a much better suited performance metric for this task. Representing the harmonic mean of precision and recall, it has a stronger focus on predictions of the positive class. This makes sense for our task, as one classifier essentially consists of 58 binary classifiers - one per class label - which needs to predict in which frames it is present, i.e. positive. Generally this metric will be much lower than the Balanced Accuracy.

b.) What is the baseline performance? What could be the best possible performance?

In our case, the simple Majority-Class baseline classifier would always and for every label predict the negative class, since no single class-label is active in more than half of the frames in our dataset. This in turn would lead to a F1-score of 0 (since there are no True Positives). As already mentioned, in order to still be able to compare our models to the baseline we also recorded the Balanced Accuracy. Obviously, as the recall for the negative class would be 1 and for the positive class 0, our baseline Balanced Accuracy is 0.5.

Considering the unavoidable noise, inaccuracies and mislabelings in the data, we suspect the best possible performance one could ever achieve to be around an F1-score of 0.9 up to 0.95.

5.) Experiments

a.) For at least three different classifiers, systematically vary the most important hyperparameters and answer the following questions for each of them:

The chosen classifier are **Random Forest, SVM and Logistic Regression**.

i.) **How does classification performance change with varying hyperparameter values? Visualize the change in performance.**

For tuning the hyperparameters 5% of the training set was used on different combinations of hyperparameters.

Random Forest To find the optimal hyperparameters for Random Forest, different combinations of n_estimators, max_depth and min_samples_split were tested (See table2). Overall, the performance is unsatisfactory and especially bad with low max_depth values (See figure4.

SVM To find the optimal hyperparameters for SVM, different combinations of C and kernel were tested (See table2). More extensive testing was not feasible due to the long training times. SVM performs relatively good and the result of the testing is that low C values on the rbf kernel lead to the best results (See figure6).

Logistic Regression

To find the optimal hyperparameters for Logistic Regression, different combinations of C, penalty and solver were tested (See table2). The performance iscomparable with SVM but the training time is many times faster. The only outlier was the combination of l2 and newton-cholesky which yielded very long training durations (See figure5).

Tested hyperparameters (Random Forest			Tested hyperparameters (SVM)		Tested hyperparameters (Logistic Regression)		
n_estimators	max_depth	min_samples_split	C	kernel	C	penalty	solver
10	5	2	0.1	linear	0.001	l2	lbfgs
30	10	5	1	rbf	0.01	None	newton-cholesky
50	None	10	10	poly	0.1		

Table 2: Tested hyperparameters

ii.) **(To what extent) Does overfitting or underfitting occur, and what does it depend on?**

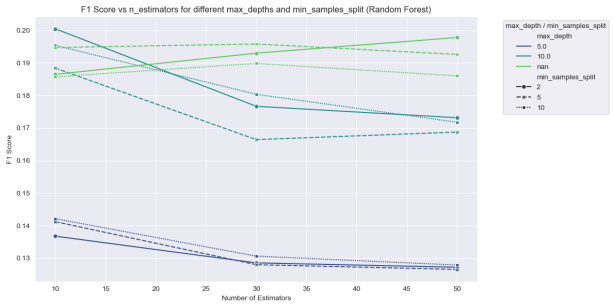

Figure 4: F1 Score for different hyperparameters (Random Forest)

Figure 5: F1 Score for different hyperparameters (Logistic Regression)

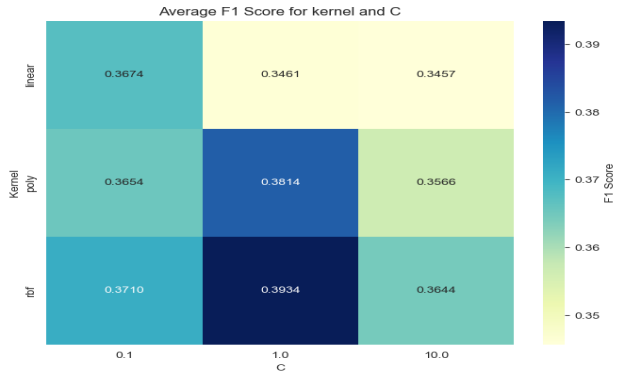

Figure 6: F1 Score for different hyperparameters (SVM)

Random Forest n_estimators: Severe underfitting for n_estimators = 10. Severe overfitting for n_estimators $\geq$ 30 max_depth: Strong underfitting for max_depth = 5. min_samples_split: Dependent of the max_depth, but doesn't cause major changes to the performance.

SVM C and kernel: For rbf and polynomial kernal, minor underfitting can be detected at C < 1(See figure 6). Overfitting occurs for rbf and polynomial kernel at C $\geq$ 1 and for the linear kernel at C > 1.

Logistic Regression No major signs of over- or underfitting were observed

b.) After selecting appropriate hyperparameters, compare the final performance estimate of the three classifiers.

The used hyperparameters and amount of training data used for the final models are dependent on the performance and train times during the parameter tuning:
Random Forest: n_estimators=10; max_depth=10; min_samples_split=5; 30% of the training set was used.
SVM: C=1; kernel=rbf; 25% of the training set was used.
Logistic Regression: C=0.1; penalty=l2; solver=lbfgs; 100% of the training set was used.

The final results are as expected Logistic Regression > SVM > Random Forest (See table 3)

Random Forest			SVM			Logistic Regression		
F1 Score	Accuracy	Training Time	F1 Score	Accuracy	Training Time	F1 Score	Accuracy	Training Time
0.3217	0.6174	2409s	0.4565	0.7236	17754s	0.5460	0.7587	6487s

Table 3: Final Model Evaluation

6.) Analysing Predictions: Find two interesting audio files that have not been used for training and qualitatively evaluate your classifier's predictions.

For this task, we use the predictions from our best performing classifier, namely the Logistic Regression model with hyperparameters 'C=0.1, penalty=l2, solver=lbfgs'.
We analyze predictions for the two files: 584022.wav, where the model does a good job and 74790.wav, where the struggles with its classifications.
For the calculation of prediction quality metrics we skipped class-labels which are not present in the file. Otherwise, some of the misclassifications would actually boost the final Balanced Accuracy.

a.) Use the spectrogram and the sequence of predictions to visualize the classifier output.

We simply feed all frames of a single file into the model, which in turn produces the corresponding predictions. These predictions as well as the corresponding melspectrogram are then plotted. We only plot labels which have been predicted to be present in at least one frame within the file, see Figure 7.

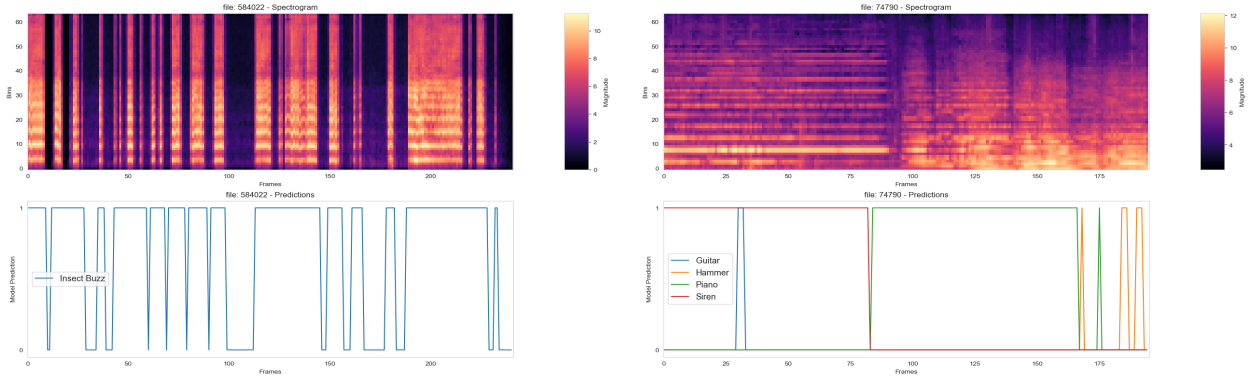

Figure 7: Visualization of predictions from our best model.

b.) Listen to the audios and inspect the corresponding predictions of the classifier. How well does the classifier recognize the classes?

- **584022.wav**: This represents an extremely simple example where the models predictions are really precise and mostly correct. Not only did it correctly predict the (only) class label, but temporally the predictions match very well. Sometimes however, some of the individual buzzes get grouped together, especially when the pauses in between them get really short. The model achieved an F1-score of 0.928 and a Balanced Accuracy of 0.896 for this file.
- **74790.wav**: This represents a more complex example where the models predictions are horribly incorrect. When actually listening to the audio file, one clearly recognizes that it solely consists of string instruments being played as well as some murmuring (disregarding background noises). Confusing the strings in the beginning for a siren still is understandable, everything else is completely wrong. The model achieved an F1-score of 0 and a Balanced Accuracy of 0.25 for this file.

c.) What are particular problematic conditions that cause the classifier to mispredict classes? Can you think of simple postprocessing steps that might help improve the predictions?

From analyzing several predictions of audio recordings we have identified that the models have a hard time predicting classes in recordings containing lots of noise or in which several class labels are active at once. Additionally, predictions for class labels which are underrepresented in the dataset (see Figure 3) also are lacking.

Often the classifier has little peaks and valleys lasting only a few frames in its predictions, as can be seen in the second plot of Figure 7 with the 'Guitar' or 'Piano' labels. Such spurious (mis-)classifications could easily be removed by setting a lower threshold for how many frames a positive or negative prediction has to last, and filtering the output accordingly. This should overall increase the quality of predictions.
Additionally, some resampling methods could be used to improve predictions for underrepresented classes.