CS181 Spring 2023 Practical I: Classifying Sounds

Adam Mohamed, with help from Evan Jiang and Taj Gulati ♡
adammohamed@college.harvard.edu

May 12, 2023

1 Introduction

This paper explores the use of linear and nonlinear models in the classification of sounds from the UrbanSound8k dataset.

2 Part A: Feature Engineering, Baseline Models

2.1 Approach

For the baseline models, I used sklearn's LogisticRegression class to train a linear classification model on both datasets (Raw Amplitude and Mel Spectogram.) The model takes in as input a sound, and predicts one of the 10 classes/categories to which the sound belongs.

For the 2 different datasets, the inputs have different representations:

- Each point in the Raw Amplitude dataset is made up of 44,100 amplitude values (taken over the span of 2 seconds.)
- The Mel Spectogram dataset is a 2D representation of the corresponding sound's raw amplitude, where each point has shape (128×87) – as described in the handout, the original amplitudes are split up into 87 time windows, with each window containing 'audio-related features'. The resulting representation is richer and contains more information than the Raw Amplitude dataset.

Logistic Regression: our model predicts data by taking a linear combination of a given input's features, weighted by the estimated parameters for each of the 10 classes (calculated when the model is trained). The resulting linear combinations are passed to the softmax function to calculate a vector of probabilities (one for each classes linear combination) that sum to 1. The predicted class is calculated based on these classification probabilities.

The inputs for the model trained on the Raw Amplitude dataset are appropriately shaped for the model to train on $(44{,}100 \times 1)$. Since the inputs from the Mel Spectogram dataset are 2D, we flatten them to allow for faster training.

2.2 Results

See Table 1 in the Appendix for class-specific accuracies. Our Raw Amplitude model performs with overall test accuracy of 17.89%, while Mel Spectogram model performs with overall 36.69%. Train accuracy for both models is very high, and about the same (≈ 97.5.)

2.3 Discussion

There are several possible reasons why the baselines might be performing poorly. On one hand, some of the specific classes constitute a small proportion of the training data, which might not be sufficient for the model to learn that classes distinguishing features well. For example, `Car Horn` and `Gun Shot`, which make up 3.54% and 1.49% of the test data respectively, have particularly low scores (with `Car Horn` scoring 0.) This could also explain the high discrepancy between the test accuracy and train accuracy, which indicates possible overfitting.

On the other hand, it could be the case that our linear Logistic Regression doesn't capture the trends accurately. For example, some of the classes might be related in a more complicated, non-linear way (for example, dogs barking might be correlated to children playing.)

The Mel Spectogram's higher performance is likely due to the greater richness of the feature representation. It contains more data than Raw Amplitude dataset, and more so, the data is not just the amplitude, but captures the different frequencies that make up the sounds.

3 Part B: More Modeling – Random Forest and K-Nearest Neighbours

3.1 First Step

3.1.1 Approach

For our nonlinear models, I used `SKLearn`'s Random Forest Classifier and K-Nearest Neighbours classes.

Random Forest: This is a classification algorithm that uses an ensemble of decision trees to make classification predictions. The ensemble of decision trees are trained on the training data using techniques like bootstrapping aggregation and feature randomness such that they are as uncorrelated with each other as possible, and the RF generates a prediction by taking the majority of the class predictions of the individual decision trees.

KNN: This classification algorithm doesn't need training. For a given input, it uses the training data, generating a prediction based on the majority class of the K-nearest neighbours. Distance function of `SKLearn`'s `KNeighborsClassifier` is the standard Euclidean distance by default.

I initialized instances with default settings for the hyperparameters, at $n = 100$ and $k = 5$. These seemed like good starting points; for RF, $n = 100$ is a balance between overfitting and having a high number of trees for higher accuracy, and for KNN $k = 5$ avoids overfitting (like $k = 1$ would), but doesn't make the predictions too general, which is likely to happen for a higher value of k given all the noise that can be found in the sounds :D

3.1.2 Results

See Table 2 in the Appendix for class-specific accuracy. See below for the overall accuracies of our models so far:

	Overall Test Accuracy (%)		Overall Train Accuracy (%)	
Model	Raw Amplitude	Mel Spectogram	Raw Amplitude	Mel Spectogram
Baseline LRC	17.89	36.69	97.73	97.55
RF	24.62	49.20	100.00	100.00
KNN	17.84	26.90	27.30	59.54

Table 1: Table showing overall accuracy for our baseline and preliminary models

We see that overall, RF outperforms our baseline models and KNN on both the Raw Amplitude and the Mel Spectogram datasets, with a score of 49.2% on the Mel Spectogram model. KNN performs worse than baseline models and RFF.

3.1.3 Discussion

The RF had an especially high accuracy compared to the rest of the models, despite also having a large discrepancy between its test and train accuracy (indicating overfitting.) It's relatively high performance on the test data is likely due to its ability to capture complex trends, given that everyday sounds (like the ones in the given classes) tend to be non-linear and complicated. It is also better at handling noise due to its ensembling of multiple, uncorrelated decision trees. This further explains why the baseline model's performance wasn't so good – they are not quite as good at handling noise, and they don't capture the complicated trends that our sound files contain.

On the other hand, the KNN model's poor performance could be due to a poor choice of k – however, it may also not be a suitable model for our data. It performs relatively well for `siren` (see results in appendix) but on other classes does poorly.

3.2 Hyperparameter Tuning and Validation

3.2.1 Approach

To tune and validate my models of interest, I used `SKLearn`'s GridSearchCV, searching over number of estimators for RF and k for KNN. For both, I used cross-validation of 5 to protect against overfitting and to test the models against how they might perform on unseen data.
RF: For Random Forest, I ran GridSearchCV, searching through n in the range between 20 and 200, in increments of 20. The result was 180 for both datasets, and so I ran another GridSearch in smaller increments and on a smaller range, from 160 to 200, with increments of 5.
KNN: Avoiding $k = 1$ due to risk of overfitting to the noise in the data, I searched through each k value between 2 and 10.

3.2.2 Results TODO

RF: RF's grid search yielded $n = 170$ for Mel data, and $n = 195$ for Amp. These provided overall accuracy scores of
KNN:

3.2.3 Discussion TODO

The accuracy scores didn't improve as much as I'd hoped. Due to time constraints, I did not search over max-depth, which I suspect would've helped improve the model (with less over-fitting.) A RandomizedSearchCV might have been more appropriate, considering the time it took to run the GridSearchCV (to run it exhaustively

4 Sources

- Understanding Random Forest
- Understanding the Mel Spectogram

5 Final Write-up and Reflections TO-DO

5.1 Discussion:

Data Pipeline: For all the models, I used both datasets. Mel Spectogram data consistently yielded better results, but rather than sticking to one for the latter models, I was curious to see the difference in accuracy. The Mel Spectogram data was flattened before being used, and this boosted performance

Model Selection:

Model Tuning:

Bias-Variance Trade-off:

Evaluation:

Domain-specific Evaluation:

Design Review:

Execution & Implementation:

6 Appendix

List of Tables

	Test Accuracy (%)		Train Accuracy (%)	
Model	Raw Amp	Mel Spect	Raw Amp	Mel Spect
Overall Accuracy	17.89	36.69	97.73	97.55
Air Conditioner	28.67	28.67	99.29	97.71
Car Horn	0.00	33.33	98.98	98.98
Children Playing	34.45	23.08	98.13	95.98
Dog Bark	12.23	19.21	95.03	94.46
Drilling	1.14	45.83	99.01	98.85
Engine Idling	30.68	40.53	97.64	95.98
Gun Shot	6.67	70.00	97.59	100.00
Jackhammer	4.24	46.19	99.54	99.54
Siren	13.98	70.34	94.16	99.55
Street Music	15.67	23.33	98.14	97.29

Table 2: Baseline model accuracy on Raw Amplitude & Mel Spectogram data

	Test Accuracy (%)		Train Accuracy (%)	
Model	RF Raw Amp	RF Mel Spect	RF Raw Amp	RF Mel Spect
Overall Accuracy	24.62	49.20	100.00	100.00
Air Conditioner	15.33	36.00	100.00	100.00
Car Horn	0.00	25.64	100.00	100.00
Children Playing	54.18	55.85	100.00	100.00
Dog Bark	29.26	44.54	100.00	100.00
Drilling	19.32	59.09	100.00	100.00
Engine Idling	36.74	43.18	100.00	100.00
Gun Shot	0.00	43.33	100.00	100.00
Jackhammer	27.97	47.03	100.00	100.00
Siren	8.05	61.86	100.00	100.00
Street Music	11.00	51.33	100.00	100.00

Table 3: Vanilla RFF Performance on Raw Amplitude & Mel Spectogram data

	Test Accuracy (%)		Train Accuracy (%)	
Model	KNN Raw Amp	KNN Mel Spect	KNN Raw Amp	KNN Mel Spect
Overall Accuracy	17.84	26.90	27.30	59.54
Air Conditioner	27.00	16.67	41.43	78.29
Car Horn	0.00	38.46	8.12	50.25
Children Playing	16.05	24.08	37.07	46.84
Dog Bark	10.92	16.59	30.02	41.11
Drilling	0.00	31.44	0.66	67.22
Engine Idling	14.77	32.95	28.85	87.93
Gun Shot	3.33	53.33	9.64	66.27
Jackhammer	0.00	63.56	0.30	82.37
Siren	80.93	26.27	75.15	49.55
Street Music	2.33	6.00	10.14	21.14

Table 4: Vanilla KNN Performance on Raw Amplitude & Mel Spectogram data