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

RF: On the refined second search, GridSearch yielded $n = 195$ for Amp and $n = 170$ for Mel data. These provided overall accuracy scores of 26.35% and 48.57% respectively (see Table 5 in Appendix.)

KNN: The results of our GridSearch are outlined in Table 6 in the appendix. For the Raw Amplitude dataset, it found $k = 2$ to be the best fit with a score of 17.8%, and for Mel Spectogram $k = 3$ with a score of 32.2%.

3.2.3 Discussion

To validate the results, I compared the overall predictive accuracies. This notably takes away from specific-class high performances (such as the KNN model's high performance on `siren`.) I expect the models to produce better results after tuning, because we can control the hyperparameters to produce the best results, and because cross-validation helps account for noise in the data, reducing overfitting across the models and assessing model performance on unseen data.

For RF, the accuracy scores didn't improve as much as I'd hoped, with a slight improvement on the Raw Amplitude models (24.62 vs 26.35), but slightly worse performance on the Mel Spectogram models (49.20 vs 48.57). The expectation was for our hyperparameter tuning to produce a model with better predictive accuracy than our untuned model, but perhaps to achieve this outcome, I would need to tune over more than 1 hyperparameter, and fit the preliminary model over multiple seeds (to account for random variation and noise.)

Our results for KNN show no improvement on the Raw Amplitude set over the preliminary model, but a slight improvement on the Mel Spectogram dataset. The model performs best on classes of sounds that are generally consistent. However, overall, it doesn't fit the data well (neither training nor test) and may not be the appropriate model for this problem.

4 Sources

- Understanding Random Forest
- Understanding the Mel Spectogram

5 Final Write-up and Reflections

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 to 1D before being used–this boosted performance on training the data/making predictions for our various models.

Model Selection: I chose to implement 2 models in Part B: Random Forest and K-Nearest Neighbours. The one I would choose would be Random Forest. Both have their trade-offs–they require more time to train (in the case of RF) or make predictions (KNN), but this also allows them to capture more complex trends in the data. Not using KNN loses the high accuracy it has for very consistent, loud sounds (like `siren`.) However, RF provides more consistent accuracy across the board.

Model Tuning: My model tuning was done exhaustively over a chosen hyperparameter set for both models. For RF, 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.)

Bias-Variance Trade-off: My models had a tendency to overfit, especially RF (with perfect scores on the training data.) This is an expected consequence of the noise in our data (for our Logistic Regression models), as well as the complexity of the RF model. I tried to account for this using 5-fold cross-validation while tuning for hyperparameters, and perhaps would have yielded better results with an improved tuning strategy (as discussed above.)

Evaluation: For evaluating the models, I mainly looked at the overall predictive accuracy of each. This was a simple and easy metric to consider, but it doesn't account for the imbalance within the classes, which was obvious when looking at higher performance on some of the specific classes for our poor-performing models (and vice versa.) Higher consistency in accuracy was also considered, as this helped us work towards more generalizable models–however, the same issue arises where we compromise for higher class-specific performance.

Domain-specific Evaluation: It was important to think about how to choose and train our model given that our sound data is highly complex and highly noisy. Simple logistic regression would be unable to accurately model our data, which we saw in our results, and attempted more complex models hopefully would be able to capture the complexity while avoiding overfitting. Our KNN model's $k = 5$ focuses on looking at the closest sounds to classify an input, and our RF's ensembling with bootstrapping techniques aims to get the right bias-variance balance.

That being said, our models still have a lot of room for improvement, and perhaps a different model that is better at ignoring noise in data is more suitable for this problem (such as CNN.)
Design Review: My tuning could have been improved, as mentioned eralier: I could have checked multiple hyperparams (specifically for RF) and used RandomizedSearchCV for more efficient hyperparameter search, as mentioned earlier.

For evaluation, I could have used standard deviation of predictive accuracy for different seeds to better understand the certainty of the accuracies, and to better evaluate the performance of the models. Other metrics could have also been used, including using `SKLearn`'s `precision_recall_fscore_support` function, and considering the ROC curve and corresponding AUC curve. However, due to space and time constraints, these were not considered in this paper.

I could also have used some way of accounting for class imbalance to provide results that gave the full picture, given that some classes made up a very small proportion of the training dataset.

Execution & Implementation: I would not deploy this model, based on the accuracy scores. There are certainly real-world considerations to consider–with potential uses like accessibility (assistance for people who are deaf/have impaired hearing) or crime prediction (which in itself raises ethical concerns, such as infringement on privacy and freedom), and I think that our model would

have to perform a lot better before it could be deployed in such contexts. With that said, training time is a consideration for more complicated models that would better be suited for sound classification.

6 Appendix

List of Tables

1 Table showing overall accuracy for our baseline and preliminary models 3
2 Baseline model accuracy on Raw Amplitude & Mel Spectogram data 7
3 Vanilla RFF Performance on Raw Amplitude & Mel Spectogram data 7
4 Vanilla KNN Performance on Raw Amplitude & Mel Spectogram data 8
5 Tuning RF - Accuracy Score for Model trained on Best Parameters 8
6 Tuning KNN - Average Accuracy for Different K Values 8

	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

Model	Number of Estimators	Test Accuracy (%)
RF Raw Amplitude	**195**	26.35
RF Mel Spectogram	**170**	48.57

Table 5: Tuning RF - Accuracy Score for Model trained on Best Parameters

k	KNN Raw Ampl (%)	KNN Mel Spect (%)
2	**17.8**	31.7
3	17.7	**32.2**
4	17.5	31.7
5	16.8	31.3
6	16.0	30.1
7	15.8	30.4
8	15.9	29.9
9	15.5	29.3

Table 6: Tuning KNN - Average Accuracy for Different K Values