Urban Sound Classification In-Depth Analysis Report

In this report, we will build and evaluate a classification model to identify the sound source given a brief audio excerpt. We will start by building our space of features from the digitized audio samples, then use these features to train a classifier that can learn patterns from the sound data to correctly identify the class of the sound source.

1. Feature Extraction:

We start building our feature space by computing certain characteristics of the input sound signals. The features we use to build our classifier are the zero-crossing rate, the spectral centroid and roll-off of the signal in addition to 20 mel-frequency cepstral coefficients that describe the signal's power spectrum. Please refer to the Data Exploration and Storytelling report for more details on the description of the extracted features. We compute features for the whole dataset of 5435 samples encompassing the ten classes of sound sources and store them in a pandas dataframe that we can save as a csv file. To facilitate building our classifier in the next step, we merge the labels dataframe loaded from the train.csv file that contains the class labels for each sound sample file with the computed features dataframe using the file ID as a joining key. The resultant merged dataframe can be saved to use in the following classification step.

2. Classification:

Before fitting our classification model, we split the dataset into two independent training and testing subsets. We use a training to testing ratio of 80:20 samples. Next, we encode all ten class labels as categorical target variables. We also normalize all feature values by subtracting the mean and scaling to unit variance.
We start the fitting process by building a random forest classification model that is an ensemble of 100 single decision trees. We find the tree nodes that discriminate the features and split the feature space by minimizing the gini purity index for each node. A random subset of features is used for every iteration to find the tree nodes. We then use the learned ensemble trees along with the test subset to predict class labels for the test samples.

We also use a k-NN classifier with 10 neighbors to classify test samples using the majority vote of the smallest distanced neighbor to the sample in the feature space.

3. Model Performance:

To assess and compare the performance of both classifiers, we will compute the mean classification accuracy score, the precision, recall, F1-scores and Matthew's Correlation Coefficient (MCC) for each independently. It is also important to compare our classifiers' performance to a generic dummy classifier;

we used a stratified dummy classifier that computes a probability for each data point in the training set belonging to the most frequent class as baseline. Results were as shown in Table 1.

Table 1: Classification Results for both classifiers

Classifier	Random Forest Tree (100 estimators)	k-Nearest Neighbor (k=10)	Stratified Dummy Classifier
Mean Accuracy Score	0.91	0.83	0.11
Precision	0.91	0.83	0.11
Recall	0.89	0.83	0.12
F1-Score	0.9	0.82	0.11
MCC	0.89	0.81	0.01

We also ran cross validation tests to better represent the models' ability to generalize. Mean Accuracy scores for 5-fold cross validation tests are shown in Table 2.

Table 2: Cross-validation (k=5) Accuracy Scores for both classifiers

	Random Forest Tree (100 estimators)	k-Nearest Neighbor (k=10)	Stratified Dummy Classifier
Mean Accuracy Score	0.91	0.84	0.1

From Table 1 and Table 2, we can see that the bootstrap aggregation (bagging) process of the 100 individual trees along with randomizing the feature selection process produced superior results to just classifying new data samples according to their 10 nearest data points. Unsurprisingly, both classifier models performed better than our dummy classifier that did not use the patterns or trends present in the data in the classification process.