Classification of Musical Playing Styles using MIDI Information

Chet N. Gnegy
Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University
chet@ccrma.stanford.edu

***Abstract*—The automation of playing style classification is a crucial step in the analysis of musical information. In this work, "style" is taken to be a more functional description of a specific instrument within a song rather than a description of the genre or articulation in which that instrument is playing. It is shown that by using a supervised learning approach, we can classify the playing style of an instrument as bass, rhythm, lead, or fingerpicking with a very high accuracy. Several machine learning algorithms are tested, the most successful one was the support vector machine with a Gaussian kernel, was found to provide an accuracy of 0.982 using a leave-one-out-cross-validation technique. Additionally, we use a recursive feature search to analyze which subset of a larger feature set is most important for this classification task.**

I. Introduction

In this study, we investigate the use of machine learning as an automatic classifier for playing styles in MIDI tracks. More specifically, our goal is to use a supervised learning approach to identify whether a segment of MIDI data is best described as a bass part, a lead part, a rhythm part, or a fingerstyle part. We have seen the application of machine-learning to style classification before for different definitions of the word "style". Two notable examples are the modeling and replication of the genre of a piece of music[1] and the classification of the articulation of the playing style.[2] Rather than looking at these aspects of music, we will look at the function of a music track in a greater context (i.e. the baseline, or the melody). These classes will be designated `Bass`, `Lead`, `Rhythm`, and `Acoustic`, respectively. As humans, we can easily recognize each class to have the following characteristics: `Bass`, typically a single line of deep tones being in support of the harmonic structure of a song; `Lead`, a melodic line such as a vocal performance or guitar solo; `Rhythm`, typically repeated chords to provide rhythmic and harmonic structure to a melody; and `Acoustic`, characterized by playing multiple individual notes or melody lines, often demonstrated by independent musical lines being performed on the same instrument. It should be mentioned that while the tracks by prominent guitarists such as Chet Atkins or Tommy Emmanuel are intended to be classified as `Acoustic`, the `Acoustic` class is also well suited for piano performances. The reason is that both styles will likely have a melody line supported by harmonic structure and/or a bass line. Even though the class names were picked based on guitar related terminology, the instrument for which these tracks were intended is not of interest in this study. Several examples of data samples are shown in Figure 1.

The training data consists of roughly 75 MIDI tracks spanning many genres. To name a few, we have chosen examples of songs by Bach, The Beatles, Duke Ellington, Eminem, Led Zeppelin, Metallica, Tim McGraw, Slayer, and Willie Nelson. Each song consists of one or more tracks, each hand tagged with one of the four class labels. An analysis of the data is done and a vector of features is collected. The features will be discussed in detail in a following section.

We investigate the use of several different learning algorithms, namely decision trees, K-nearest neighbors, logistic regression, quadratic discriminant analysis, and support vector machines. The most successful of these will be selectively tuned using a feature search.

II. Data Collection

The MIDI songs that were used were either exported from the guitar tablature software, TuxGuitar[1] or obtained from an online guitar tablature database[2] or from an online MIDI fingerstyle database[3]. Each track is tagged with a class in Logic Pro using the Text Tool. To reduce the number of outliers in the data set, each track was scrutinized to make sure that the class label was representative of the entire track. For example,

[1]TuxGuitar: sourceforge.net/projects/tuxguitar/
[2]Guitar Tab Database: http://www.ultimate-guitar.com/
[3]Fingerstyle Guitar File Collection: http://www.acousticfingerstyle.com/

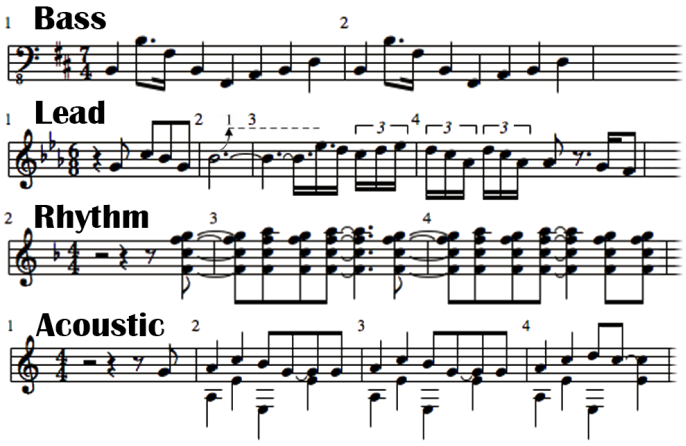

Figure 1: An examples of each class type. The samples from top to bottom are intros to Pink Floyd's *Money*, Led Zeppelin's *Since I've Been Lovin' You*, Fleetwood Mac's *Go Your Own Way*, and John Loudermilk's *Windy and Warm*.

if a `Rhythm` track has a two or three measure stretch in which a brief solo is played, that solo section is omitted.

The tracks are partitioned into sections, typically twelve measures long. This increases the amount of data points provided to our algorithms (at the expense of the assumption that the data is identically and independently distributed) and allows us to maintain short-time characteristics in the performances. In other words, if we did not partition the tracks, any measure to measure variation would be averaged out during feature computation. Each feature is normalized in a preprocessing step such that all examples of any particular feature are within the range [-1, 1]. Even though it is typical to normalize to a mean of zero and a standard deviation of 1, this was shown to produce somewhat higher accuracy rates.

III. Features

Many features were proposed, ranging from simple averages and standard deviations to more complex ones that were often designed for separating two specific classes, most often for distinguishing `Rhythm` from `Lead` or `Acoustic`. To aide in the definitions of our features, we will first discuss terminology. First, readers should be aware of standard MIDI notation for pitch[4]. For example, "Bb3" is a $B\flat$ in octave 3 (MIDI pitch 47), and "C4" is middle C (MIDI pitch 60). There will be some discussion of note groupings, which we will define to be any set of notes that are played simultaneously in a single MIDI track. Note groupings will be designated by $\vec{\gamma}$, where γ_i is the i^{th} note in some set. A note will be designated using a superscript. For example γ_i^j is the j^{th} note in note grouping i. Each note evaluates to its defined MIDI pitch. The median pitch in a note grouping is designated by $\hat{\gamma}$. The number of note groupings in a set is designated by $|\vec{\gamma}|$. The pitch class (often called "chroma") of some note, α_i, will be designated $C(\alpha)$. Pitch class is similar to pitch but is invariant to changes in the octave of the note (i.e. "A3" and "A5" are of the same pitch class, "A").

The features that have been considered for this classification problem are as follows:

Mean Pitch: The average of the MIDI pitch values

Median Pitch: The median of the MIDI pitch values

Lowest Pitch: The minimum of the MIDI pitch values

Highest Pitch: The maximum of the MIDI pitch values

Pitch Standard Deviation: The standard deviation of the MIDI pitch values

Duration: The average of the MIDI note lengths. Rather than use the raw durations, we scale the durations and take the base 2 logarithm such that sixteenth notes, eighth notes, quarter notes, and half notes have durations of -2, -1, 0, and 1, respectively.

Duration Standard Deviation: The standard deviation of the MIDI note lengths. We use the scaled log of the raw duration data as seen in the Duration feature.

Play Count: The average number of note groupings per measure

Note Count: The average number of notes per measure

Repetition: The average number of notes that were played in the previous note grouping

Polyphony: Average number of notes being played simultaneously. This is measured using the overlap of notes and not simply counting the number of 'note on' events occurring at any given time.

Polyphonic Separation: Given a time in which multiple notes are playing, the average difference between consecutive MIDI pitches. Zero for monophonic tracks.

Polyphonic Repetition(n_{thresh}): Given that a note is polyphonic (greater than n_{thresh} notes in a grouping), the average number of pitches that are common between a note grouping and the previous note grouping. Zero for monophonic tracks.

Percent Polyphony(n_{thresh}): The percentage of the sample in which the number of playing notes exceeds n_{thresh}.

Coverage: The probability distribution, $\Gamma(\vec{\gamma})$ for pitch class is computed on a per-measure basis. For each note grouping, we sum the probabilities corresponding to each note. Coverage is the average of this metric for all note groupings in the sample as computed in Equation 1. This represents the amount of tonal diversity in the sample. Repeatedly strumming the same chord will produce a high coverage. Playing a scale will produce a low coverage. Note that even though coverage is probability based, the coverage of a sample will exceed one if multiple octaves of the same pitch class are routinely played. Let N represent the number of notes groupings (each of size Q_n) in $\vec{\gamma}$)

$$\mathscr{C}(\vec{\gamma}) = \frac{1}{N}\sum_{n=0}^{N}\sum_{q=0}^{Q_n} p\{C(\gamma_n^p)|\Gamma(\vec{\gamma})\} \tag{1}$$

Jump Size: The absolute value of the difference between MIDI pitch numbers from one note grouping to the next. Each note grouping is represented as the median of its pitches. Let N represent the number of notes groupings in $\vec{\gamma}$)

$$\mathscr{J}(\vec{\gamma}) = \frac{1}{N}\sum_{n=0}^{N} |\hat{\gamma}_n - \hat{\gamma}_{n-1}| \tag{2}$$

Lyricality: A measure of the "singability" of a sample. Lines that alternate in pitch are given a negative lyricality, and lines that monotonically increase or decrease are given high lyricality. Repeated notes are scored as zeros. This metric scales according to the jump size and the length of the alternating or monotonic segment. The exact computation of a lyricality score for a length L subset of $\vec{\gamma}$ is seen in Equation 3. Each alternating, monotonically increasing, and monotonically decreasing sequence of note groupings in a sample is given a score, $\ell(\vec{\gamma})$. We sum the scores from the monotonic sequences and subtract the scores from the alternating sequences, averaging by the number of note groupings, as seen in Equation 4.

$$\ell(\vec{\gamma}) = L\sum_{n=1}^{L} |\hat{\gamma}_n - \hat{\gamma}_{n-1}| \tag{3}$$

$$\mathscr{L}(\vec{\gamma}) = \frac{1}{|\vec{\gamma}|}\left(\sum_i \ell_i(\vec{\gamma})_{mono} - \sum_j \ell_j(\vec{\gamma})_{alt}\right) \tag{4}$$

[4]MIDI Numbers: http://newt.phys.unsw.edu.au/jw/notes.html

Other features were used, some based on pitch autocorrelations, repetitions, and onset quantization to a quarter note or eighth note grid. We will avoid discussion of these features because they were eliminated during feature selection. The means of feature selection will be discussed in a later selection.

IV. Data Separability

The features in the previous section, especially the latter ones, were designed in response to looking at the distributions in early stages of this study. Using Polyphony, Note Count, and Mean Pitch alone, we can make a rudimentary classification and correctly predict the majority of `Bass` data points, as well as a decent fraction of the `Lead` points. However, discerning the `Acoustic` class from `Rhythm` and even `Lead` with any reasonable accuracy is a hopeless endeavor. Figure 2 shows the distribution of the data with respect to these simple features. The more complex features are quite useful for distinguishing between parts labeled `Rhythm` and `Acoustic`. Figure 3 shows a subset of the feature space in which the `Rhythm` and `Acoustic` classes are fairly separable.

It is worth mentioning that an additional reason that the data is not completely separable is because the class assignments are not completely objective. While there are parts that a human would unquestionably recognize as a lead part, such as a guitar solo, there are other parts with similar characteristics that function as a rhythm part. Consider the thrash/heavy metal genres; during a verse, it is typical for a distorted guitar to repeatedly pick a low, often drop-tuned, open string, occasionally playing a fifth chord or a complex riff. Even though this may be a low polyphony, even technically intricate guitar lick, it still functions as a rhythmic backing track. The classification algorithm is likely to mistake it for lead or even bass if the mean pitch is low enough. In this case, the context of the song matters. Similar ambiguities exist between rhythm and acoustic parts. There is a grey area between finger-picking and strumming, most notably, a hybrid picking approach in which a guitarist will hold a pick between their thumb and index finger and pluck different strings using their other three fingers. Simple piano parts can also walk the line between the `Rhythm` and `Acoustic` labels. For these cases, it would not be uncommon for human musicians to debate the ground truth labeling. We therefore do not expect perfect classification from our algorithm in these cases. Of course, we will look at the errors of our algorithm to make sure that they are indeed examples of a disputable ground truth.

V. Choosing a Learning Algorithm

Now that we have a set of features to test on, it is time to pick a learning algorithm. Some immediate choices come to mind: logistic regression, quadratic discriminant analysis, and support vector machines (SVM). The Scikit-Learn[3] Python module came with a wealth of other options, so the data were tested using decision trees and K-nearest neighbor algorithms as well.

The testing procedure uses a modified version of Leave-one-out-cross-validation (LOOCV) to estimate the accuracy of the learning algorithm. Standard LOOCV removes a single training example from the training set, trains the algorithm on all but that example, and then tries to classify that training example. It does that to each one of the training examples in turn and computes the accuracy as the fraction of correct classifications. Rather than testing on individual data points, we test on all of the data points that came from an individual song. We will refer to this as LOOCV$_{song}$. This is done because we cannot robustly estimate the test error of the algorithm if there are data points from the same track in the training and test set (recall that the tracks are partitioned into measure groupings and that we get a small cluster of points from a single track rather than a single point). This would cause the accuracy estimate to be inflated. By ensuring that no song appears in both the training and test set, we can be confident that the test data are not being overfit. It has the added benefit of being much faster than traditional LOOCV.

Algorithm	Accuracy
Decision Tree	0.818
Logistic Regression	0.920
K-Nearest Neighbors (K = 3)	0.906
K-Nearest Neighbors (K = 10)	0.941
Quadratic Discriminant Analysis	0.792
SVM Linear	0.934
SVM Polynomial Kernel	0.908
SVM RBF Kernel	0.912

Table 1: We have several algorithms that perform reasonably well. It is clear that decision trees and QDA are not good candidates for a learning algorithm.

Table 1 shows the results. It is clear that K-nearest neighbors (K = 10) performed the best of all of the algorithms, but does that make it the most appropriate to use? To learn about how the class labels were being decided, the algorithms were tested on a subset of the feature space (two features at a time) and the decision boundaries were viewed on a 2D plot. Some of these results are shown in Figure 4. Note that we don't expect performance that is representative of training on the whole feature space, we only want to look at the nature of the decision boundaries. As we can see from the figure, the K-nearest neighbors decision boundary is quite jagged. It is expected that even a noise in the test set could cause different classes to be assigned. The decision boundaries for the support vector machines and for logistic regression look much more smooth. The linear SVM had decision boundaries that

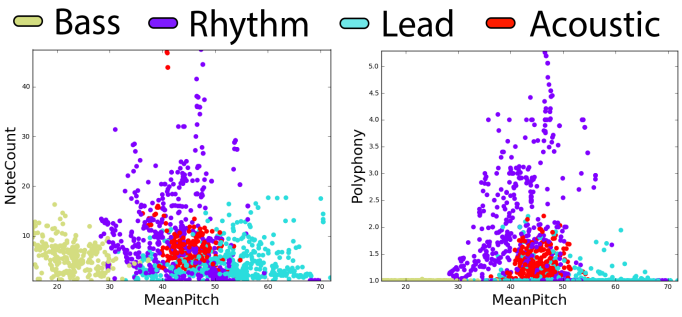

Figure 2: Using Note Count, Mean Pitch, and Polyphony, we can see a high degree of separation for the `Bass` class, but the `Acoustic` class is not separable.

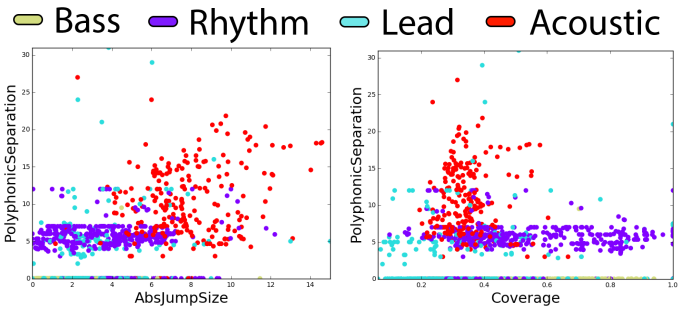

Figure 3: Using Polyphonic Separation, Jump Size, and Coverage, the `Rhythm` and `Acoustic` classes become more separable. Almost none of the `Bass` examples have polyphony and are therefore clustered on the axis.

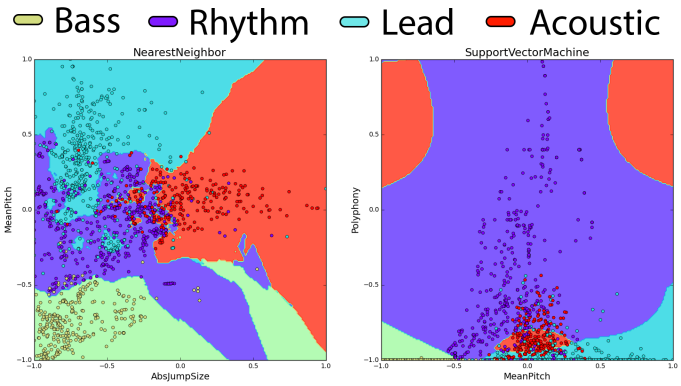

Figure 4: The decision boundaries above were generated by training on the entire dataset and excluding all but two features. This does not give us an estimate of the accuracy of training on the full feature space, but it will give an idea of what decision boundaries look like. The K-nearest neighbors decision boundary is very jagged and is not expected to be very robust to noise. The SVM boundary is much smoother.

were straight lines (as the name suggests) as did the logistic regression algorithm. The SVMs with polynomial and gaussian kernels had curvature and provided a more appropriate-looking fit. A fifth order polynomial kernel and a radial basis function (RBF) kernel were used for the latter two SVMs, respectively. The linear SVM uses a one-vs-the-rest multi-class strategy, and the other SVMs use a one-vs-one strategy.

It was expected from looking at the data that a non-linear decision boundary is well suited to fit the data. For this reason, though we include an analysis of the linear SVM in the feature search section, we ultimately expect to pick an algorithm with smooth, non-linear decision boundaries for the final implementation.

VI. Feature Search

We have seen decent performance across several different algorithms and would now like to get the best performance over each particular algorithm. We do this using a common approach, a recursive feature search. The procedure is as follows:

1) Find LOOCV$_{song}$ accuracy, α using some feature space
2) Backwards search
   a) Remove the i^{th} feature
   b) Find $LOOCV_{song}$ accuracy, α_i
   c) Replace the i^{th} feature
3) Find highest k values of α_i
4) For feature i in top k features, such that $\alpha_i > \alpha$:
   a) Remove the i^{th} feature
   b) Recurse using feature space without feature i for top k features
   c) Replace the i^{th} feature

Of the 30 or so features that were designed for this classification task, approximately 15 remained in the optimal subset for any particular algorithm. Fortunately, we see in Table 2 that the accuracy results have improved quite a lot. The choice of learning algorithm is now quite obvious. The support vector machine using the RBF kernel has an accuracy of 0.982, and the smooth, non-linear decision boundary that was discussed in the previous section. It is not surprising that the SVM with the RBF kernel outperformed the other algorithms, as it is fitting a curve to an infinite dimensional feature space.

Algorithm	Accuracy
Logistic Regression	0.936
Nearest Neighbors (K = 10)	0.968
SVM Linear	0.953
SVM Polynomial Kernel	0.957
SVM RBF Kernel	0.982

Table 2: After an individual tuning using the recursive feature search, we have optimized the performance of five, high performing algorithms.

The optimal feature space for our SVM was found to be Mean Pitch, Median Pitch, Lowest Pitch, Highest Pitch, Pitch Standard Deviation, Duration, Play Count, Note Count, Repetition, Polyphony, Polyphonic Separation, Polyphonic Repetition (n=2), Percent Polyphony (n=1), Coverage, and Jump Size. It was a bit surprising to not see higher accuracy using some of the more complex features. Lyricality, Onset Quantization, and a couple autocorrelation-based features were not especially useful for this classification task.

VII. Misclassifications

The confusion matrix for the SVM using the RBF kernel is shown in Table 3 with the precision and recall statistics. Most of the contents of the matrix lie on the diagonal, indicating correct classification. It is clear that the algorithm doesn't have much difficulty classifying `Bass` examples, but it makes the occasional mistake on other classes. There were generally three types of observed error in the cross validation, we will now look at examples of each.

		Correct Class						
		R	L	B	A		Prec.	Recall
Guess	R	322	4	1	8	R	0.961	0.964
	L	2	200	0	0	L	0.990	0.981
	B	0	0	319	0	B	1.000	0.997
	A	10	0	0	181	A	0.947	0.958

Table 3: The confusion matrix for the SVM, and associated precision and recall statistics

The first, and largest sources of error are between the `Acoustic` and `Rhythm` classes. This not too surprising; as it was discussed above, these two classes are not easily separable and share many similar properties. An example of this is shown in Figure 5. The sample is most likely played with a pick with the exception of the higher notes that have the same onset as the bass notes, which are are probably played with the ring, pink, or middle fingers (an example of hybrid picking). A quick listen to this sample in its original context should be convincing enough that the most appropriate label is `Rhythm`, but out of context one could reasonably debate that `Acoustic` is a good label.

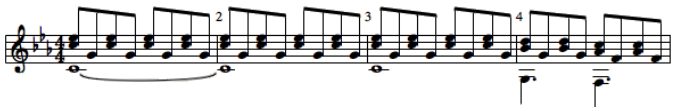

Figure 5: No Doubt's *Don't Speak* is classified as `Acoustic`, but the correct class is `Rhythm`. Taken out of context (perhaps neglecting that No Doubt is a punk band), it would not be surprising for a human to mistake this as a fingerpicked part.

There are also a few instances of `Lead` being mistaken for `Rhythm`, one of which is shown in Figure 6. For a lead part, this example exhibits quite a bit of polyphony. It is no surprise that the algorithm was fooled. The other lead parts that were misclassified were for the exact same reason.

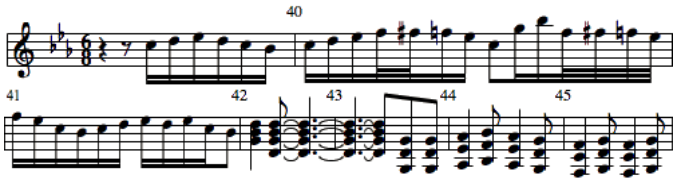

Figure 6: An excerpt from a guitar solo in Led Zeppelin's *Since I've Been Lovin' You*. The high polyphony near the end of the sample causes this lead part to be classified as `Rhythm`.

Finally, repeated riffs with low polyphony that function as rhythm parts are often labeled as `Lead`. Figure 7 shows an example of this, Slayer's *Angel of Death*. The full sample is the shown riff repeated a couple of times. This isn't an easily singable melody that we would typically label as `Lead`. Metallica's *Enter Sandman* riff is actually classified as `Acoustic` by the algorithm.

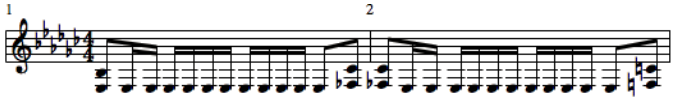

Figure 7: The introduction to Slayer's *Angel of Death* is classified as `Lead` rather than `Rhythm` even though its function in the song is to support the vocal line.

VIII. Conclusions

Using a support vector machine with a Gaussian kernel, we can successfully classify musical data into categories of playing style. An accuracy of 0.982 is shown, but we will still bother to mention that for some examples, the ground truth labeling is not completely objective. Some musicians may argue that some of the mislabeled samples are actually correct (and for that matter that some of the correctly labeled samples should be labeled otherwise).

IX. Future Work

This study is intended to be the first part of an algorithmic composition application that does composition in a similar style to some given set of MIDI files. Once the computer has labeled the input tracks according to playing style, it can analyze the tracks appropriately, extracting rhythm, chords, and tonality information from the rhythm and bass tracks, melodic and harmonic information from the leads, and a bit of everything from tracks with the `Acoustic` label.

Acknowledgment

Thanks to Roger Dannenberg, Anders Oland, Ryan Rifkin, and Julius Smith who provided helpful advice throughout the course of this project.

References

[1] Shlomo Dubnov, Gerard Assayag, Olivier Lartillot, and Gill Bejerano. 2003. Using Machine-Learning Methods for Musical Style Modeling. Computer 36, 10 (October 2003), 73-80.

[2] Dannenberg, Thom, and Watson, A Machine Learning Approach to Musical Style Recognition, in 1997 International Computer Music Conference, International Computer Music Association (September 1997), pp. 344-347.

[3] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol 12, pp. 2825–2830, 2011.