Identification of Instruments Through Sound Characteristics

Avery Cameron: 200254563

Raymond Knorr: 200302685

Mason Lane: 200376573

Kegan Lavoy: 200378170

Project Summary:

Using the NSynth dataset, a large, high-quality dataset of annotated musical notes, our aim is to train a model to classify a note played on a violin against a note played on a piano. We extract audio features from .wav files using the LibROSA python package, and use these features to train supervised learning models. We also plan on recording and labelling a smaller set of our own .wav files for violin and piano to compare against the accuracy produced from the NSynth dataset. Algorithms being used to train models include Binary Decision Trees, AdaBoost, K-NN, and Support Vector Machines.

Description:

The classification of musical sounds has widespread potential. The ability for a computer to accurately distinguish different instruments is the underpinning of any automated transcription software. This would involve computers processing a full song and writing out all of the various instrument parts, and would fundamentally change the way that musical property is managed, among other things.

Our project does not aim to tackle such an overarching issue as full-scale musical transcription. However, developing individual classifiers to distinguish notes played on different instruments is vital for any larger applications.

The NSynth dataset is made up of sounds from various sampling libraries, and the sounds are free from additional noise. We are interested to see if models trained on pristine audio samples will still perform well on our own gathered samples.

Data and Algorithms:

Dataset:

We are using the NSynth dataset to train our models. The NSynth dataset consists of 305,979 musical notes, each with a unique pitch, timbre, and envelope. Pitch is “the degree to the height or depth of a tone or of sound” (“Pitch”), it is very closely related to frequency. Timbre refers to the “quality of tone distinctive of a particular singing voice or musical instrument.” (“Timbre”). Lastly, envelope refers to the “attack, sustain, and decay of a sound.” (The Editors of Encyclopaedia Britannica, 2014). The full NSynth dataset consists of individual train, validation, and test datasets. We used all three of these datasets in the development of our models.

Each note is a four second long .wav file taken at a sampling rate of 16kHz. All of the samples were synthetically generated using commercial sampling libraries. The samples are additionally annotated with three additional pieces of information:

- **Source**: the method of sound production. Is one of acoustic, electronic, and synthetic. Our focus is on the acoustic source group.
- **Family**: the high-level family of which the note's instrument is a member. Each instrument is a member of exactly one family. The selection of families is as follows: bass, brass, flute, guitar, keyboard, mallet, organ, reed, string, synth_lead, and vocal. Our objective only involves classification of violin and piano audio files, which are resembled by the string and keyboard families.
- **Qualities:** Sonic qualities of the note. There can be multiple sonic qualities per note. The selection of sonic qualities is as follows: bright, dark, distortion, fast_decay, long_release, multiphonic, nonlinear_env, percussive, reverb, tempo-synced

In addition to the annotations provided in the NSynth dataset, we are also using the LibROSA python library to extract several other audio characteristics from the .wav files. These characteristics are: harmonic, percussive, Chroma Energy Normalized, Mel Frequency Cepstrum Coefficients, Mel Spectrum and Spectral Contrast. The harmonic and percussive components of the sound are extracted and saved separately. Chroma Energy Normalized is the sound separated by pitch and height (chromatic quality and the octave of the pitch) and is the short time energy spread over the twelve chroma bands (Müller, Ewert, 2011). The Mel-Frequency Cepstrum Coefficients (MFCC) is the fast fourier transform (FFT) of the power spectrum, used in audio recognition and modeling (Xu, et al., 2004)(Xu et al., 2003). In addition to MFCC, the Mel spectrum is an intensity plot of a short-time fourier transform (Smith). Spectral Contrast measures the difference between the valleys and peaks of the spectrum in the sub-bands. (Jiang, Lu, Zhang, Tao, & Cai, 2002).

Another dataset that could be used for instrument classification would be the Instrument Recognition in Musical Audio Signals (IRMAS) dataset with approximately 7000 train and 3000 test files which is similar to NSynth but is less annotated and has less examples (MTG - Music Technology Group).

Algorithms & Initial Results:

State-of-the-Art Approaches:

The state-of-the-art approaches used are based partially on research by Patil & Sanjekar (2017) who suggest the use of SVM, and AdaBoost. Additionally, research by Nagawade and Ratnaparkhe (2017) used MFCC to accurately represent unique qualities and features of the sound. Random forests were also implemented as a successful solution for instrument classification by Owers (2016).

Binary Decision Tree:

The first algorithm we used was a Binary Decision Tree. The tree was created and fit to the data using a Sci-kit learn decision tree. Sci-kit learn uses an optimised version of the Classification and Regression Trees (CART) algorithm to construct its decision trees. The CART algorithm “constructs binary trees using the feature and threshold that yield the largest information gain at each node.” (“Decision Trees”). Using such a tree, an error rate of ~11% was obtained for both training and test data.

AdaBoost:

The second algorithm used was AdaBoost which was implemented with Sci-kit learn. Sci-kit learn allows you to define a base estimator which defaults to a DecisionTreeClassifier with a max depth of 1. An estimator is a simple model that can give a classification. Sci-kit learn implements the AdaBoost algorithm to apply heavier weights to specific training estimators with the goal of correcting wrong predictions (“AdaBoost”). Applying the AdaBoost algorithm to the validation and train set using a decision tree classifier with a max depth of 1 and 50 estimators provided a train error of 3.69% and a test error of 8.01%.

KNN:

The third algorithm was a K Nearest Neighbor (KNN) Classifier. The classifier was created using the Sci-kit learn KNeighborsClassifier function. This function uses the minkowski distance to compute the distance between points. After some initial tuning of the hyperparameter K, we found that the best value to use for our dataset was K = 27. The initial error estimations were 17.3% and 23.5% for the train and test set respectively.

SVM:

Next, we used a Support Vector Machine (SVM) from Sci-kit learn. Results initially showed an accuracy of 75%. However, the result applied to every feature which seemed odd. Using more than three features generated predictions accurate to 83%-86% depending on the kernel used, linear or radial basis. Balancing the number of string to keyboard examples diminished accuracy. This led to an accuracy of 50%-63% with many features. Further work on the SVM began to generate promising results. Our current best is 74.7% using chroma_cens, mfcc, mel_spec and spec_contrast together as features. Working to determine the most relevant features may help improve the results shown in Appendix C.

Future Direction:

Our tentative plan is to continue to work with our chosen algorithms and to see if we can optimize the choice of parameters used as input. Most of the annotated characteristics given in the NSynth dataset are going unused, and they could prove valuable to include in our classifiers.

Currently our best performing model is our AdaBoost model at about an 8% error rate, however the Binary Decision Tree comes in at a comparable second at 11%. We plan on developing a Random Forest model to see if we can improve upon the success of the Binary Decision Tree. We also plan to improve the SVM implementation to reduce misclassified results.

Additionally, Avery and Raymond plan on recording violin or piano .wav files in the same format as the NSynth dataset. These will be manually annotated as either originating from a piano or a violin. We hope to have somewhere in the range of 50-100 annotated samples to test on. It is worth mentioning that in order to test on these generated samples, we cannot use NSynth specific annotations in our training of our models as these annotations will not be available in our created samples.

Student Contribution:

Avery: Extracted features using the LibROSA library for use with the classification algorithms. Using LibROSA, extracted, analyzed and plotted mel-frequency cepstrum, chroma feature and spectral properties of sound files. Read articles on feature extraction to determine the set of features to extract. Implemented a simple AdaBoost algorithm using Sci-kit learn. Tested AdaBoost algorithm to determine estimator values with low error for the train and test set (Appendix B).

Ray: Used extracted features from LibROSA and the CART decision tree classifier from Sci-kit learn to create a binary decision tree classifier. The maximum depth of the tree was varied as a hyperparameter, and the max depth of four was found to have the best results. Wrote a majority of the progress report.

Mason: Trained support vector machines using Scikit-learn. After seeing unsatisfactory results, attempted to reorganize the X and Y data to achieve better results while training. Added a plot of our SVM to visualise the results. Using Avery’s feature extraction, developed a method that generates a dataset of the features for each .wav file using an initial json file’s index. The database saves to a .csv file for easy access.

Kegan: Using the feature extraction functions developed by Avery, as well as some data organization functions made by Ray, I applied the KNN classifier from Sci-kit learn on the data. I tried many values of K to determine which value gave the best error results. To do this I used the square root of our number of training samples as the upper limit (197) and 3 as the lower limit. I then tried the values listed in Appendix A and compared their results.

References:

AdaBoost. (n.d.). Retrieved March 13, 2020, from https://scikit-learn.org/stable/modules/ensemble.html#adaboost

Decision Trees. (n.d.). Retrieved March 12, 2020, from https://scikit-learn.org/stable/modules/tree.html

Jiang, D.-N., Lu, L., Zhang, H.-J., Tao, J.-H., & Cai, L.-H. (2002). Music type classification by spectral contrast feature. *Proceedings. IEEE International Conference on Multimedia and Expo*. doi: 10.1109/icme.2002.1035731

Kawwa, N. (2019, April 29). Can We Guess Musical Instruments With Machine Learning? Retrieved March 13, 2020, from https://medium.com/@nadimkawwa/can-we-guess-musical-instruments-with-machine-learning-afc8790590b8

KNeighbors Classifier. (n.d.). Retrieved March 12, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

MTG - Music Technology Group. (n.d.). Retrieved from https://www.upf.edu/web/mtg/irmas

Müller, M., Ewert, S. (2011) "Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features" International Society for Music Information Retrieval Conference (ISMIR) Retrieved March 12, 2020, from https://ismir2011.ismir.net/papers/PS2-8.pdf

Nagawade, M. S., & Ratnaparkhe, V. R. (2017). Musical instrument identification using MFCC. *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. doi: 10.1109/rteict.2017.8256990

Navlani, A. (2019, December 27). (Tutorial) Support Vector Machines (SVM) in Scikit-learn. Retrieved March 9, 2020, from https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

Owers, J. (2016). *An Exploration into the Application of Convolutional Neural Networks for Instrument Classification on Monophonic Note Samples*

(Master’s thesis, University of Edinburgh, Edinburgh, United Kingdom). Retrieved from https://jamesowers.github.io/files/thesis.pdf

Patil, S. D., & Sanjekar, P. S. (2017). Musical instrument identification using SVM, MLP& AdaBoost with formal concept analysis. *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*. doi: 10.1109/icisim.2017.8122157

Pitch. (n.d.). Retrieved March 12, 2020, from https://www.dictionary.com/browse/pitch

Smith, J. O. (2007). Mathematics of the Discrete Fourier Transform (DFT) with Audio Applications. Retrieved March 12, 2020, from https://ccrma.stanford.edu/~jos/st/Spectrograms.html

The Editors of Encyclopaedia Britannica. (2014, November 17). Envelope. Retrieved March 12, 2020, from https://www.britannica.com/science/envelope-sound

Timbre. (n.d.). Retrieved March 12, 2020, from https://www.merriam-webster.com/dictionary/timbre

Xu, M., Duan, L.-Y., Cai, J., Chia, L.-T., Xu, C., & Tian, Q. (2004). HMM-Based Audio Keyword Generation. *Advances in Multimedia Information Processing - PCM 2004 Lecture Notes in Computer Science*, 566–574. doi: 10.1007/978-3-540-30543-9_71

Xu, M., Maddage, N., Xu, C., Kankanhalli, M., & Tian, Q. (2003). Creating audio keywords for event detection in soccer video. *2003 International Conference on Multimedia and Expo. ICME 03. Proceedings (Cat. No.03TH8698)*. doi: 10.1109/icme.2003.1221608

Appendix A: KNN K Value test results

Value of K tried	Training set Error (%)	Test set Error (%)
5	8.5	24.2
7	11	26.5
9	12.8	25.8
11	13.7	26.3
13	14.8	26.1
15	15.2	24
17	15.8	24.4
19	16	25
21	16.6	24.7
23	17	23.9
25	17.2	24.2
27	17.3	23.5
29	17.4	23.9
31	17.7	23.5
33	17.8	24.2
37	18	23.5
41	18.4	24.2
47	18.7	24.5
53	19	24.8
59	19.4	25.5
65	19.8	24.8
73	20	24.8
83	20.4	24.8
93	20.7	25.3
103	20.7	26.3

113	21	26.5
123	21	26.3
133	21.3	26.5
143	21.4	26
153	21.5	25.8
163	21.8	27.5
173	21.8	27.5
183	21.8	27.3
193	21.9	27
195	21.9	27.5
197	22	27.6

As we can see the best choice from the table is K = 27 because it has the lowest test error / train error combination.

Appendix B: AdaBoost Estimator Test Results

Estimators	Train Error	Test Error	Average Error
1	28.99	29.41	29.2
2	28.99	29.41	29.2
3	14.93	15.85	15.39
4	10.93	13.07	12
5	10.93	13.07	12
6	10.26	12.09	11.175
7	9.83	10.78	10.305
8	9.58	14.54	12.06
9	8.97	12.91	10.94
10	10.01	10.95	10.48
11	10.01	10.95	10.48
12	9.15	10.62	9.885
13	8.42	11.44	9.93
14	7.86	9.8	8.83
15	7.56	10.13	8.845
16	7.37	10.46	8.915
17	6.7	10.95	8.825
18	6.02	10.95	8.485
19	5.71	11.44	8.575
20	5.59	11.6	8.595
21	5.96	9.8	7.88
22	5.53	10.78	8.155
23	5.22	9.8	7.51
24	5.22	10.13	7.675
25	4.67	10.62	7.645
26	4.61	10.78	7.695
27	4.61	10.46	7.535
28	4.55	10.13	7.34
29	4.73	10.62	7.675
30	4.67	9.31	6.99
31	4.24	9.48	6.86
32	4.67	9.8	7.235
33	4.36	9.64	7
34	4.36	9.8	7.08
35	4.24	9.8	7.02
36	4.24	9.48	6.86
37	4.18	9.8	6.99
38	4.24	9.97	7.105
39	4.3	8.33	6.315
40	4.48	8.01	6.245
41	4.55	8.17	6.36
42	4.42	8.01	6.215
43	4.18	9.48	6.83
44	4.05	8.66	6.355
45	4.05	9.15	6.6
46	3.87	8.33	6.1
47	3.75	8.5	6.125
48	4.05	9.8	6.925
49	4.12	9.8	6.96
50	3.69	8.01	5.85
51	4.05	9.97	7.01
52	3.75	9.97	6.86
53	3.93	9.97	6.95
54	3.81	9.8	6.805
55	3.87	10.13	7
56	3.5	9.64	6.57
57	3.75	10.13	6.94
58	3.26	8.99	6.125
59	2.76	10.29	6.525
60	2.7	9.48	6.09
61	2.89	9.8	6.345
62	2.64	9.97	6.305
63	3.07	9.8	6.435
64	3.26	9.8	6.53
65	3.07	9.8	6.435

66	2.83	9.8	6.315
67	2.95	9.64	6.295
68	2.83	9.64	6.235
69	3.07	9.31	6.19
70	3.01	9.8	6.405
71	2.89	9.48	6.185
72	2.7	9.97	6.335
73	2.95	9.8	6.375
74	2.7	10.13	6.415
75	2.33	9.97	6.15
76	2.21	9.97	6.09
77	2.33	9.97	6.15
78	2.7	9.97	6.335
79	2.7	9.48	6.09
80	2.52	9.48	6
81	2.15	9.48	5.815
82	2.33	9.48	5.905
83	2.52	9.48	6
84	2.46	9.15	5.805
85	2.46	9.64	6.05
86	2.58	8.99	5.785
87	2.58	9.31	5.945
88	2.27	8.99	5.63
89	2.21	9.48	5.845
90	1.9	9.31	5.605
91	2.03	9.31	5.67
92	1.97	9.31	5.64
93	2.09	9.31	5.7
94	2.03	9.8	5.915
95	2.15	9.15	5.65
96	1.9	9.97	5.935
97	2.09	9.64	5.865
98	2.15	9.31	5.73
99	2.09	9.64	5.865
100	1.84	9.97	5.905
101	2.03	8.99	5.51
102	2.09	9.31	5.7
103	2.03	8.99	5.51
104	2.03	9.31	5.67
105	1.9	9.48	5.69
106	2.21	9.8	6.005
107	1.97	9.31	5.64
108	1.84	9.31	5.575
109	1.84	9.31	5.575
110	1.78	9.15	5.465
111	1.66	9.31	5.485
112	1.66	9.31	5.485
113	1.6	9.31	5.455
114	1.6	9.15	5.375
115	1.54	9.31	5.425
116	1.6	9.15	5.375
117	1.54	9.15	5.345
118	1.6	9.15	5.375
119	1.35	9.31	5.33
120	1.23	9.31	5.27
121	1.35	9.31	5.33
122	1.29	9.31	5.3
123	1.29	9.31	5.3
124	1.29	9.64	5.465
125	1.23	9.64	5.435
126	1.17	9.48	5.325
127	1.29	9.31	5.3
128	1.17	10.13	5.65
129	1.29	9.8	5.545
130	1.17	10.29	5.73
131	1.17	10.29	5.73
132	1.23	10.13	5.68
133	1.11	10.29	5.7

134	1.23	10.13	5.68
135	1.11	9.48	5.295
136	1.29	9.64	5.465
137	1.04	9.64	5.34
138	1.47	9.64	5.555
139	1.04	9.48	5.26
140	1.29	9.48	5.385
141	1.23	9.64	5.435
142	1.23	9.31	5.27
143	1.29	9.64	5.465
144	1.11	9.64	5.375
145	1.17	9.64	5.405
146	1.11	9.64	5.375
147	1.11	9.64	5.375
148	1.11	9.64	5.375
149	1.04	9.64	5.34
150	0.8	10.13	5.465

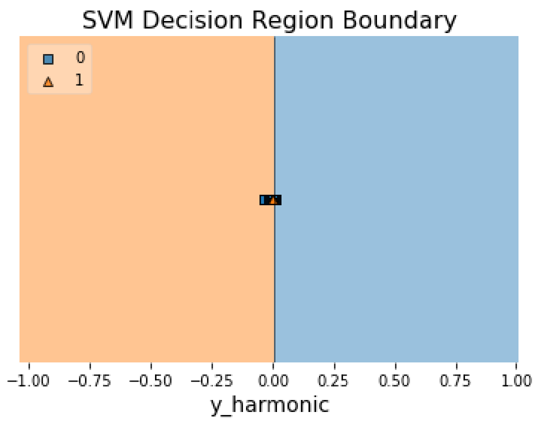
Appendix C: Support Vector Machine Test Results

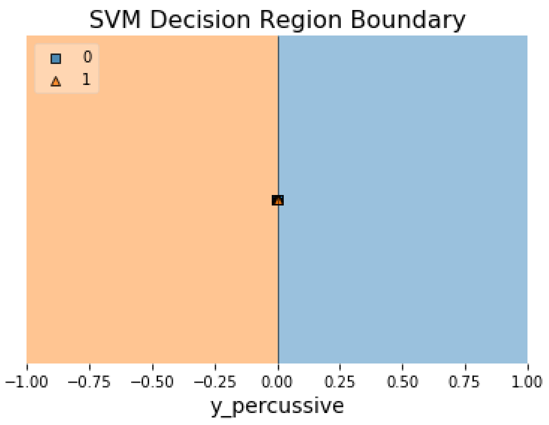
	Feature(s)	Accuracy	precision	recall	f1-score
SVM (linear)	y_harmonic	50%	0: 0 1: 0.5	0: 0 1: 1	0: 0 1: 0.67
SVM (linear)	y_percussive	50%	0: 0 1: 0.5	0: 0 1: 1	0: 0 1: 0.67
SVM (linear)	chroma_cens	66.8%	0: 0.74 1: 0.63	0: 0.52 1: 0.81	0: 0.61 1: 0.71
SVM (linear)	mfcc	47.2%	0: 0.47 1: 0.48	0: 0.41 1: 0.54	0: 0.44 1: 0.50
SVM (linear)	mel_spec	70.8%	0: 0.65 1: 0.86	0: 0.92 1: 0.49	0: 0.76 1: 0.63
SVM (linear)	spec_contrast	46.4%	0: 0.45 1: 0.47	0: 0.34 1: 0.58	0: 0.39 1: 0.52
SVM (linear)	chroma_cens, mfcc, mel_spec, spec_contrast	70.4%	0: 0.71 1: 0.70	0: 0.68 1: 0.73	0: 0.70 1: 0.71
SVM (linear)	chroma_cens, mel_spec	69.9%	0: 0.67 1: 0.74	0: 0.78 1: 0.61	0: 0.72 1: 0.67
SVM (RBF) C=100 g=1	y_harmonic	50%	0: 0 1: 0.5	0: 0 1: 1	0: 0 1: 0.67
SVM (RBF) C=100 g=1	y_percussive	50%	0: 0 1: 0.5	0: 0 1: 1	0: 0 1: 0.67
SVM (RBF) C=100 g=1	chroma_cens	66.5%	0: 0.81 1: 0.61	0: 0.43 1: 0.90	0: 0.56 1: 0.73
SVM (RBF) C=100 g=1	mfcc	41.7%	0: 0.39 1: 0.43	0: 0.34 1: 0.48	0: 0.34 1: 0.48
SVM (RBF) C=100 g=1	mel_spec	67.3%	0: 0.63 1: 0.75	0: 0.82 1: 0.52	0: 0.72 1: 0.62

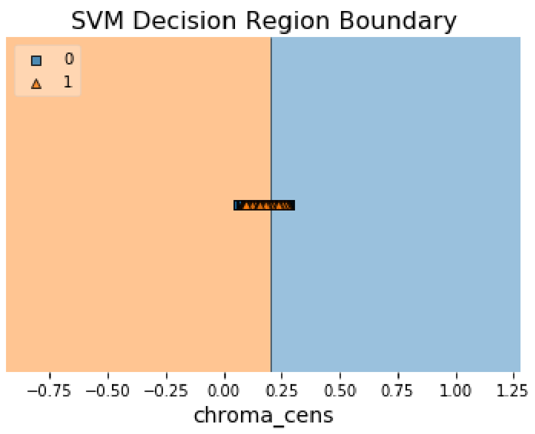
SVM (RBF) C=100 g=1	spec_contrast	59.3%	0: 0.67 1: 0.56	0: 0.37 1: 0.82	0: 0.47 1: 0.67
SVM (RBF) C=100 g=1	chroma_cens,mfcc, mel_spec, spec_contrast	74.7%	0: 0.79 1: 0.72	0: 0.68 1: 0.82	0: 0.73 1: 0.76
SVM (RBF) C=100 g=1	Chroma_cens, mel_spec	67.5%	0: 0.64 1: 0.74	0: 0.80 1: 0.55	0: 0.71 1: 0.63

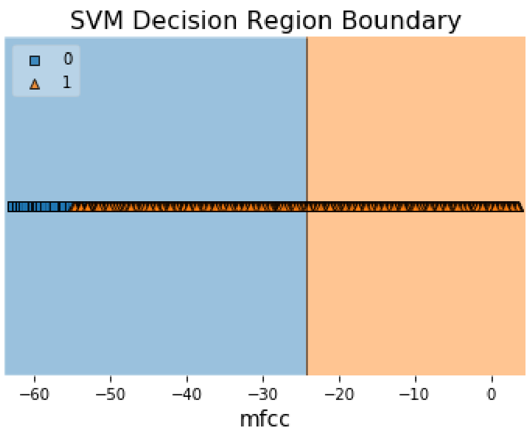
SVM fitted with train dataset, predictions made with input from test dataset

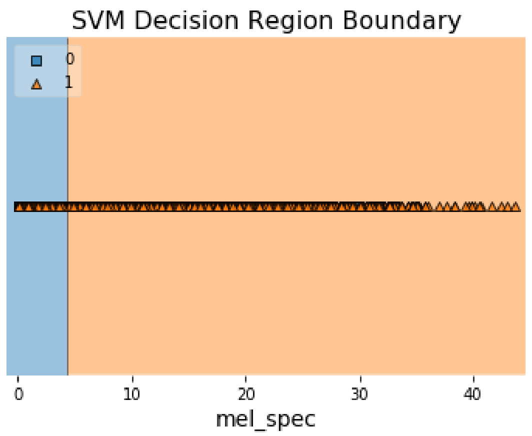
With Linear Kernel

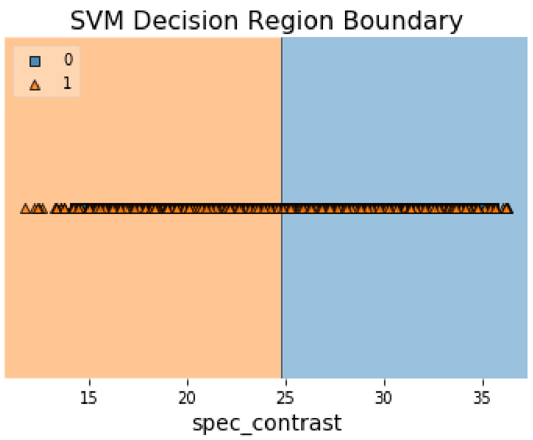


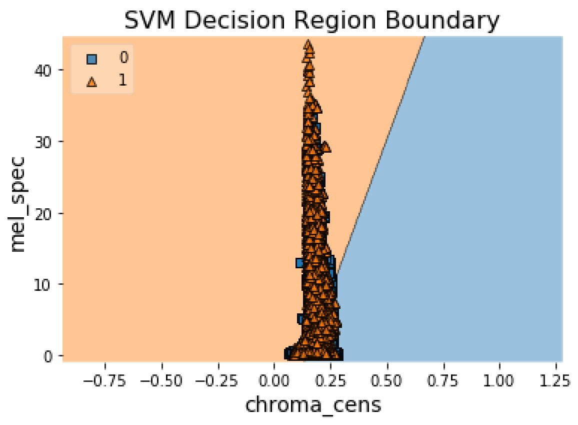




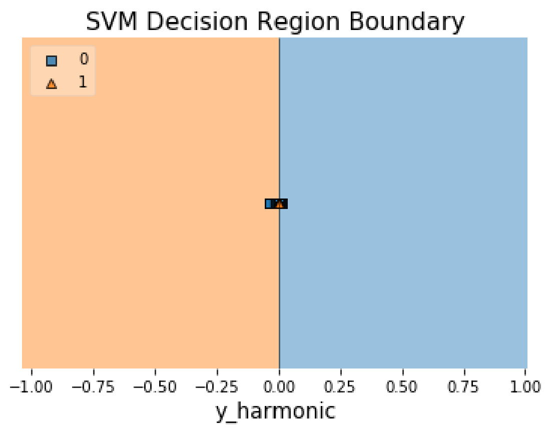


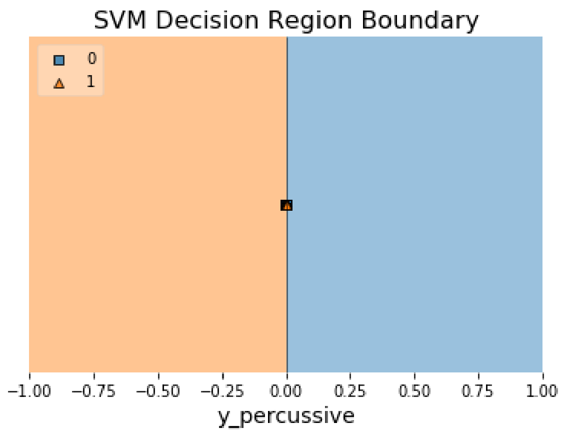


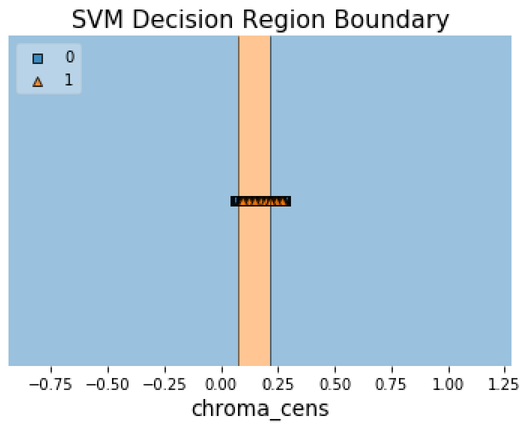


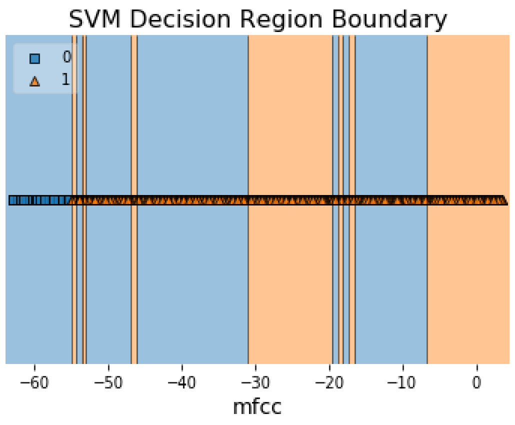


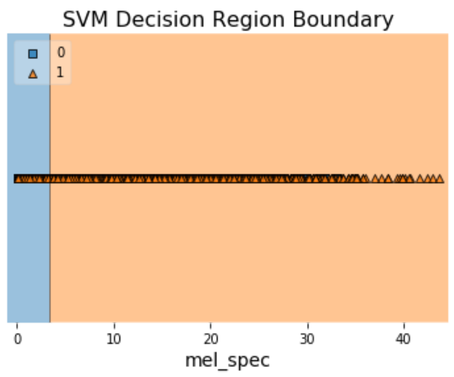
With Radial Basis Function Kernel

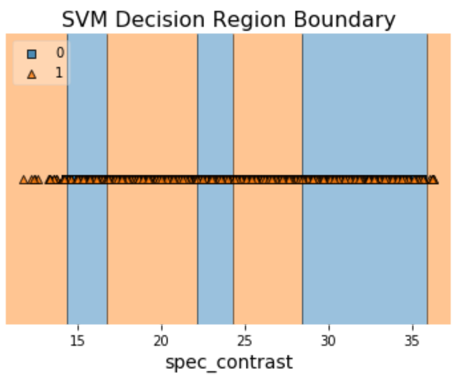

SVM Decision Region Boundary
y_harmonic

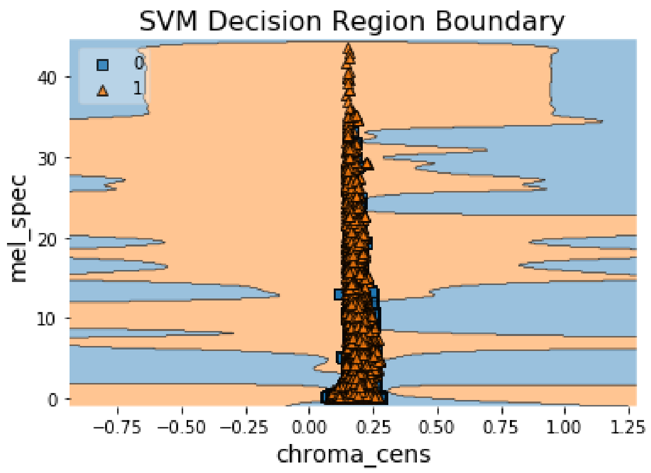

SVM Decision Region Boundary
y_percussive


SVM Decision Region Boundary
chroma_cens


SVM Decision Region Boundary
mfcc


SVM Decision Region Boundary
mel_spec


SVM Decision Region Boundary
spec_contrast


SVM Decision Region Boundary
mel_spec
chroma_cens