Asher Elazary – Assignment 2

Title: Asher Elazary
Student ID: s3931943
Student Name and email (contact info): Asher Elazary – arhelazary@gmail.com
Affiliations: RMIT University.
Date of Report: 24/05/2023

I certify that this is all my own original work. If I took any parts from elsewhere, then they were non-essential parts of the assignment, and they are clearly attributed in my submission. I will show I agree to this honor code by typing "Yes": *Yes*.

Introduction:

In this paper, I will be using machine learning to classify data from the *Turkish Music and Emotion Dataset*. This dataset was used by researchers to compare the performance of various neural networks on the task of 'music emotion recognition'. In this study, the dataset was pre-prepared by participants, who labelled each sample of music either 'happy', 'sad', 'angry', or 'relax', before the data was modelled and then classified by the researcher's neural network. (Bilal Er & Aydilek 2019) My goal was to explore this same dataset's features, and see if I could train my own model using decision tree and random forest classifiers to correctly label each sample. In doing so, I used my domain specific knowledge around music and music theory, and identified some possibly flaws in the paper's research methodology.

Data pre-processing:

The *Turkish Music and Emotion Dataset's* features are generated by a Music Information Retrieval MATLAB library called *MIRtools* (Lartillot 2021), where 50 descriptive features are created by analysing the music's data. As MIRtools is largely a frame-based analysis software (e.g. each music file is decomposed into successive frames with a SDFT or similar), most the features are the mean values of each statistic over the running time of the audio file. Whilst every generated feature is a continuous numeric feature, it is important to note that the features contain a number of different measurements, such as frequency in hertz (0 to Nyquist), amplitude/magnitude (0-1 range), BPM, and slope value (negative to positive range). I note that there no missing values in this dataset, and no spurious numeric values that I can see that aren't natural outliers in the dataset.

To begin data pre-processing, I performed the typical tasks of removing the classifiers from the dataset to prevent data leakage, and transforming the class labels into integers for the model fitting. Importantly, I decided not to treat outliers through removal or imputation as I found that these treatments negatively affected the final performance of the tree-based models. One thing I did want to experiment with, however, was creating a new feature derived from the mean chromagram amplitude features (Chromagram_Mean_1 – Chromagram_Mean_12) that could be possibly indicative of the piece's key, and tonality. The chromagram features 1-12 correspond the binned amplitude values of the chromatic scale on the frequency spectrum. Knowing this, for each sample we can take the arg. max of the chromagram features to identify the piece's probable root. From here, we can then try and match a scale/scale fragment to the piece by summing the indices from the previously identified root, modulo 12.

Using this algorithm, I wanted to see if I could determine whether the mean 'majorness' or 'minorness' of a piece had some relation to the participant's classifier. One caveat to this algorithm is that because the Major and Natural minor scales are inversions of one another, they will always score the same. To mitigate this effect, I defined 'Major' as [0, 2, 4, 7, 9, 11], and 'Minor' as [0, 2, 3, 7, 8, 10] in the scale dictionary, omitting the P4th (index 5). Picking the exact scale degrees took some experimentation, as different diatonic permutations (e.g. [0, 4, 7, 9, 11] vs [0, 3, 7, 8, 10]) had negative effects downstream on the performance of the model. After some tweaking, omitting the P4th from both sets was found to be the most relevant to each model.

Class	most_likely_scale	Minor count	Minor mean	Major count	Major mean
angry	Major	45	0.335263	45	0.392041
	Minor	55	0.391300	55	0.321361
happy	Major	47	0.446160	47	0.536011
	Minor	53	0.508475	53	0.437415
relax	Major	43	0.356942	43	0.514368
	Minor	57	0.443213	57	0.332000
sad	Major	44	0.364973	44	0.471754
	Minor	56	0.447104	56	0.340696

From these figures, we can see that there is a 'Minor' tinge to the majority of samples. Interestingly, there is still a majority 'minor' affect to the 'happy' classifiers, but this may show that my algorithm has not effectively captured the tonality of the pieces, or that the correlation between minor/major vs 'good' (happy/relax) and 'bad' (angry/sad) emotions is not strong enough to imply a statistically significant link. Additionally, I would like to note that many of the compositions in the dataset contain microtonal pitch content, an idiom found frequently in Turkish music.('Arabic, Turkish, Persian - Xenharmonic Wiki' n.d.) Given this, it is quite likely that 12 tone chromagram analysis is not always going to accurately capture musical information in microtonal music – as microtonal pitches are quantised into 12-tone bins– resulting in a loss of information. In light of this, I wanted to see if these engineered features would be relevant to the tree models despite the possible loss of information from chromagram binning.

Data Exploration:

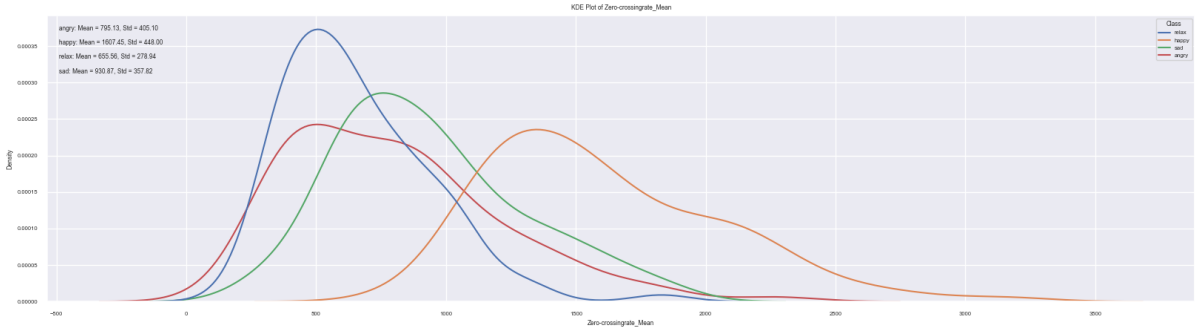
In my data exploration phase, I decided to use 'permutation importance' to perform feature selection on the dataset, adapted from the sk-learn documentation('Feature importances with a forest of trees' n.d.). The permutation importance algorithm assesses the importance of a feature on a chosen model by shuffling the data in that feature column, and comparing the model's performance with the same feature unshuffled. The bigger the effect on the performance by shuffling the feature, the more important the feature is to the model. Considering this, I wrote a function adapted from the scikit-learn documentation that assessed permutation importance for both the Decision Tree and Random Forest classifiers, as permutation importance is assessed per model. I set a semi-arbitrary threshold to capture the top ~10 features from each selection, and then segmented the dataframe by the feature list. Interestingly, my engineered tonality features ('Major' and 'Minor') indicated some importance to the Decision Tree model, but not to the Random Forest model.

	Decision Tree Feature Rank	std
0	Zero-crossingrate_Mean	0.036345564
1	HarmonicChangeDetectionFunction_Std	0.032264532
2	HarmonicChangeDetectionFunction_Period Amp	0.025612497
3	Pulseclarity_Mean	0.022561028
4	Rolloff_Mean	0.018867962
5	HarmonicChangeDetectionFunction_Mean	0.017464249
6	MFCC_Mean_2	0.014456832
7	Major	0.012041595
8	Minor	0.011661904
9	AttackTime_Slope	0.011661904
10	MFCC_Mean_11	0.010440307

	Random Forest feature rank	std
0	Zero-crossingrate_Mean	0.024819347
1	HarmonicChangeDetectionFunction_Std	0.019078784
2	HarmonicChangeDetectionFunction_Period Amp	0.017916473
3	MFCC_Mean_4	0.016401219
4	Eventdensity_Mean	0.015620499
5	Roughness_Slope	0.01356466
6	MFCC_Mean_2	0.012688578
7	EntropyofSpectrum_Mean	0.01183216
8	Spectralkurtosis_Mean	0.011135529
9	MFCC_Mean_1	0.010954451
10	Roughness_Mean	0.010198039

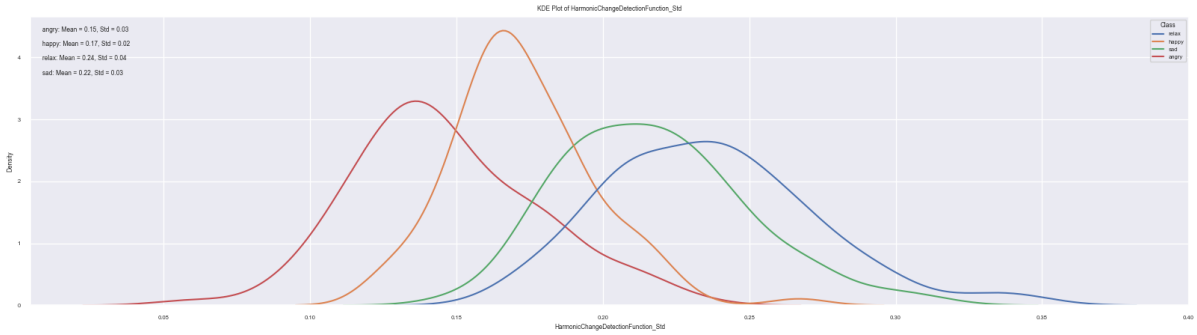
Below is a combined list of the combined top 10 features of each model, segmented by class label and graphed on a KDE plot:

Zero-crossing rate mean:



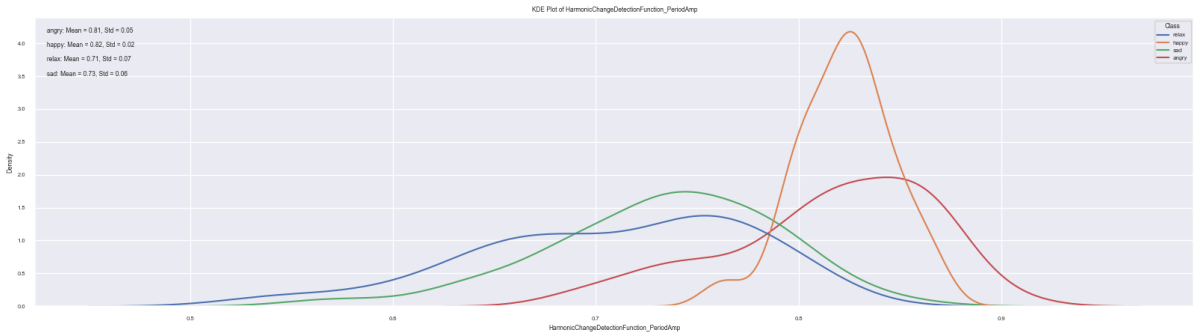
The zero-crossing rate provides some indication of the amount of the level of activity in a signal, as a higher zero crossing rate may indicate less periods of silence (very high pitched material will also have an influence on this as pitch has a relationship with frequency/rate). From this graph, we can see that the 'happy' classifier has the highest mean zero-crossing rate, possibly indicating that pieces labelled as 'happy' may have higher levels of 'activity'.

Harmonic Change Detection Function Std:



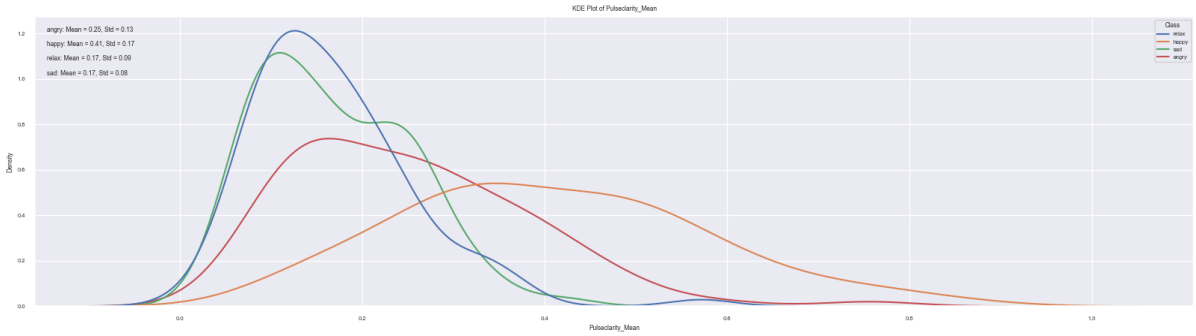
Accoding to the MIRtools manual, the Harmonic Change Detection Function (HCDF) is the flux of the tonal centroid. The tonal centroid being the approximate pitch centre of the piece. The standard deviation of this function is indicative of how much the tonal centre changes over the course of the composition. We can see that 'angry' has the least amount of change, possibly indicating that compositions that are labelled as 'angry' are more likely to have a constant tonal centre of pitch, from composition/orchestration techniques such as drone, heavy percussion, or low tessitura. On the other hand, the higher mean standard deviation of the 'relax' label may indicate a wider range of tonal variations and fluctuations throughout the composition.

Harmonic change detection function period amplitude:



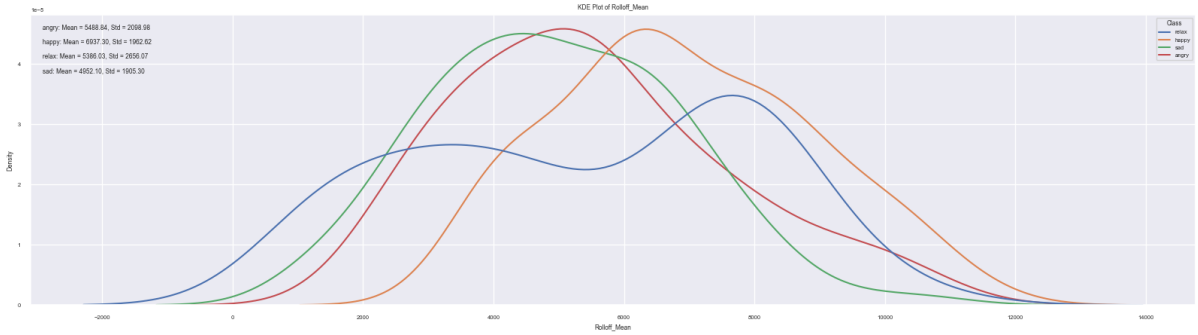
According to the MIRtools manual(Lartillot 2021), the period amp statistic is the 'normalized amplitude of that main periodicity'. In the context of the harmonic Change detection function, this corresponds with the amplitude of the tonal centroid. We can see that the 'angry' and 'happy' labels have the highest tonal centroid amplitude values, indicating that the tonal centroid in these classifiers have a much higher amplitude.

Pulse clarity mean:



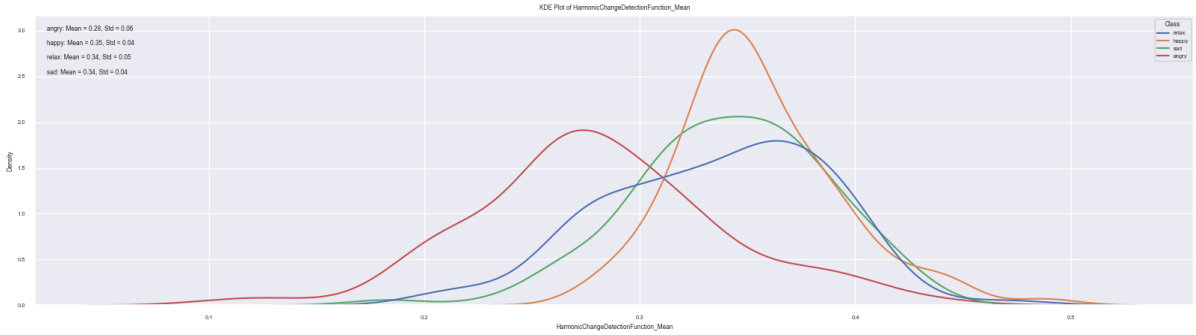
According to the MIRtools manual the pulse clarity 'estimates the rhythmic clarity, indicating the strength of the beats estimated by the mirtempo function.' (Lartillot 2021) From the KDE plot above, we can see that pieces with the 'happy' label have a significantly larger variance in pulse clarity than the other labels, as well as the highest mean value. This indicates that the happy label captures more musical material that is rhythmically 'distinct' in comparison to the other labels. In other words, the beats on average correspond more to the pulse of the composition.

Rolloff mean:



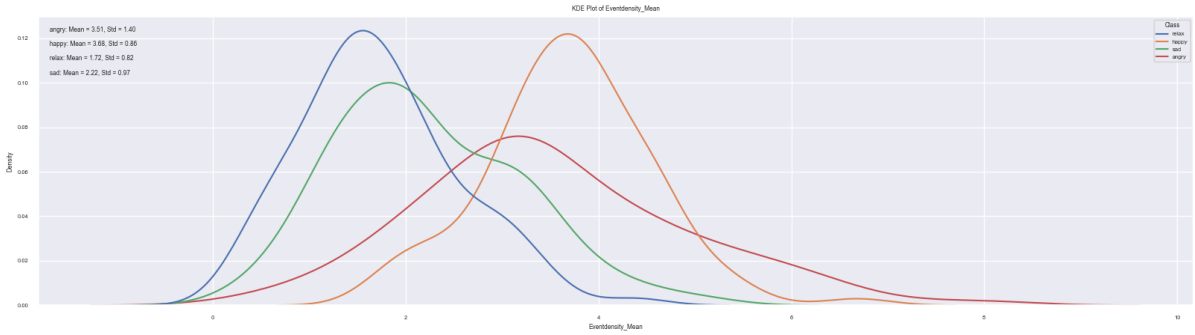
According to the MIRtools manual, rolloff is indicative of the relative high frequency energy of a signal. From the the KDE plot, we can see that the happy has the highest average frequency rolloff, indicating that pieces labelled as 'happy' on average have more high frequency energy. However, the relax class label has a bivariate distribution, with a median distribution around 8000 kHz.

Harmonic Change Detection Function mean:



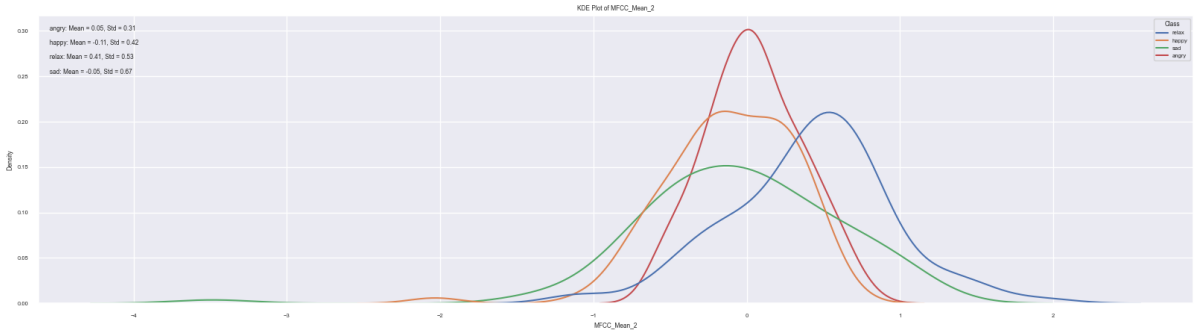
According to the MIRtools manual, the Harmonic Change Detection Function (HCDF) is the 'flux of the tonal centroid'. (Lartillot 2021) A higher mean value of the HCDF suggests a greater average rate of tonal centre changes. This could indicate a more dynamic or fluctuating tonal structure in the composition, where the tonal centre shifts frequently or with larger deviations from the average tonal centroid, and could imply a higher degree of harmonic variation or instability throughout.

Event Density Mean:



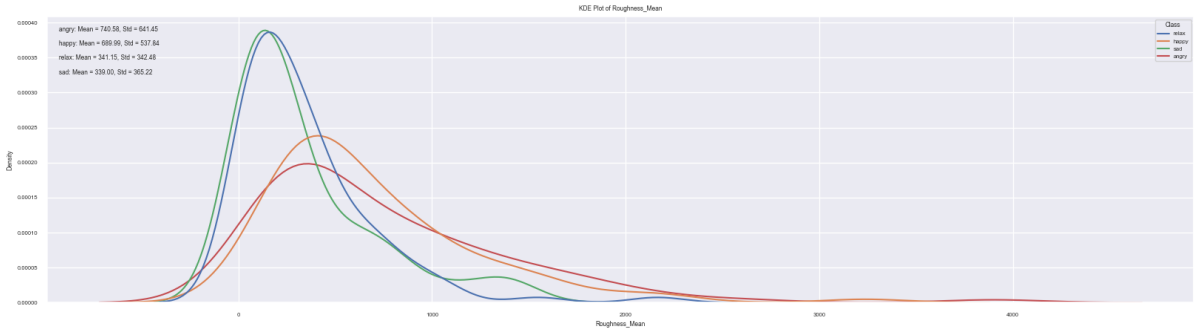
According to the MIRtools manual, the event density feature estimates 'the average frequency of events, i.e., the number of events detected per second.' (Lartillot 2021) Similar to the zero-crossing rate, the event density provides some indication of the amount of the level of activity in a signal. Notably, the 'Happy' classification label has the highest average within both features.

MFCC 2 Mean:



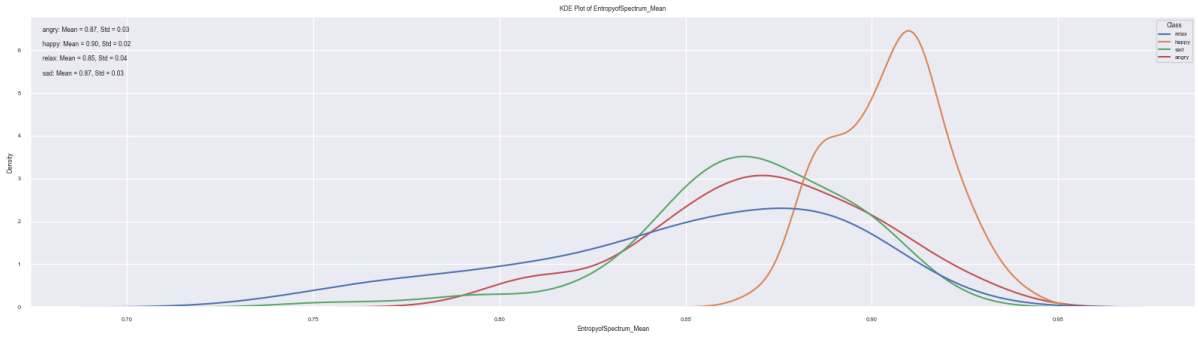
Mel Frequency Cepstral Coefficients (MFCCs) quantify the perceptual importance of different frequency bands to the human ear, using the Mel frequency. In this study, 13 coefficients were chosen. During feature selection with permutation importance, both the decision tree and random forest model indicated the second MFCC coefficient as having some relevancy to the models. Unfortunately, without more information on the exact configuration of the filter banks used, it is difficult to determine what Mel frequency range the second coefficient covers.

Roughness mean:



According to the MIRtools manual, roughness 'estimation of the sensory dissonance, or roughness, related to the beating phenomenon whenever pair of sinusoids are closed in frequency'. (Lartillot 2021) In this KDE plot, we can see that the 'happy', and 'angry' have higher mean sensory dissonance, with more variance than the 'relax' or 'sad' class labels.

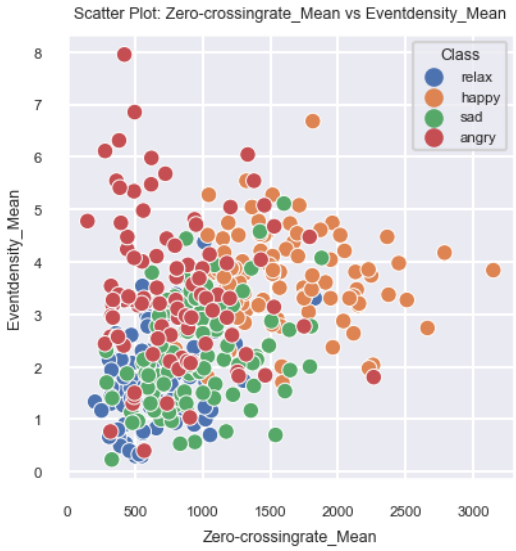
Entropy of Spectrum mean:



According to the MIRtools manual, spectrum entropy 'returns the relative Shannon (1948) entropy of the input', (Lartillot 2021) applied to spectrum. This approximately corresponds to the 'randomness' or noisiness of a signal. In the graph above we can see that the 'happy' class label has the highest mean spectral entropy, whilst the other class labels have similar mean spectrum entropy, with the relax class label having the highest variance.

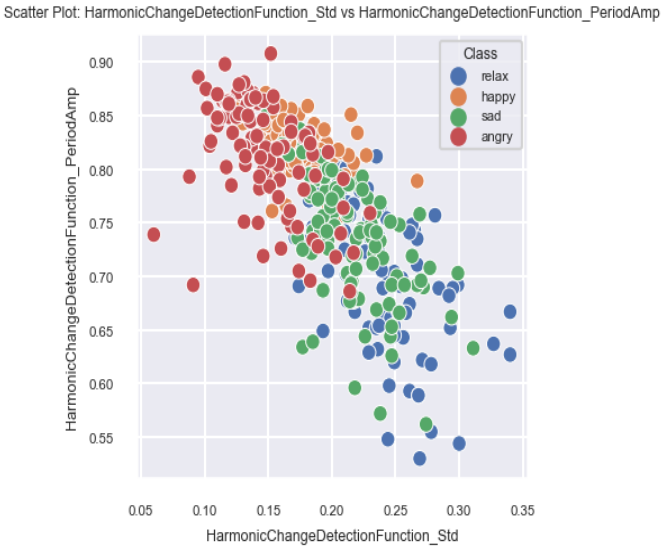
Feature Pairs – Scatter plots:

Having selected relevant features for each model, we can now graph every feature pair on a scatter plot. While assessing every feature pair is beyond the scope of this paper, particularly relevant or novel feature pairs are analysed below. Note that the plots are automatically normalised, and the labelled class names are indicated by colour.



From comparing the zero-crossing rate mean to the event density, we can hypothesise that both the angry and happy class labels have a higher level of musical activity on average than the relax and sad class labels. However, because mean zero-crossing rate has a relationship with a signal's frequency, we can hypothesise that the 'angry' class has a higher-level activity in terms of musical events per second, whilst the 'happy' class is more likely to have higher frequency pitch content.

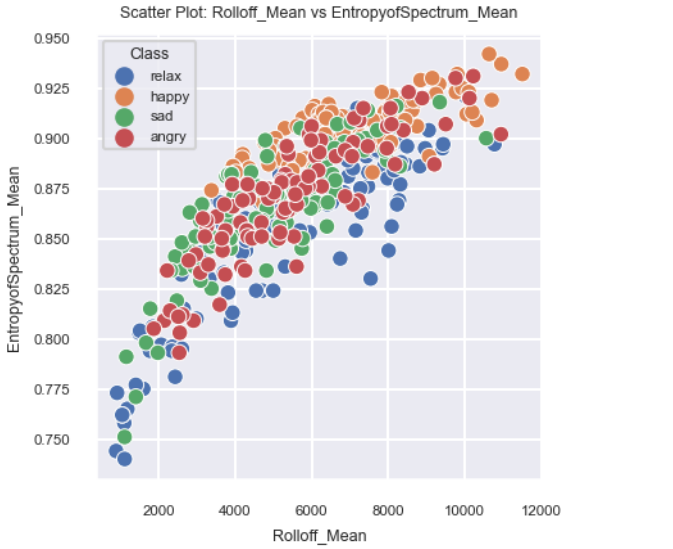
From comparing the zero-crossing rate mean to the entropy of spectrum mean, we can hypothesise that there is a clear relationship between the 'noisiness' of a signal's spectrum and the mean zero-crossing rate; As the mean spectrum entropy increases, so too does the mean zero-crossing rate. We can see a clear segmentation of class label in this distribution, with noisier and higher activity corresponding more often to the 'happy' class label.



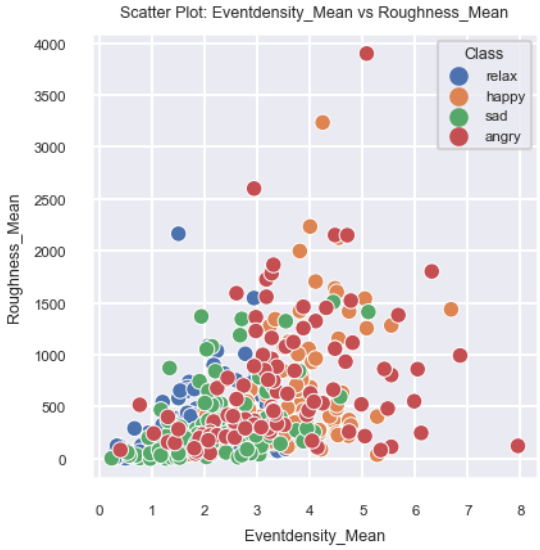
From comparing the HCDF standard deviation with the HCDF period amplitude, we can hypothesise that as the magnitude of the tonal centroid (volume of the average root note) increases, the less likely there is to be a fluctuation of harmonic content. There is an interesting segmentation of class label seen here, as the 'angry' and 'happy' tonal centroids are more constant, and of higher volume, whereas the sad and relax labels are more harmonically variable, but with a quieter volume.

From comparing the HCDF standard deviation with the Roughness slope (MIRtools manual: the roughness slope is the derivative of the sensory dissonance, where positive indicates increasing and negative indicates decreasing (Lartillot 2021)) we can hypothesise that as the sensory dissonance increases over the course of the composition, the more likely the composition is to be harmonically fluctuating.

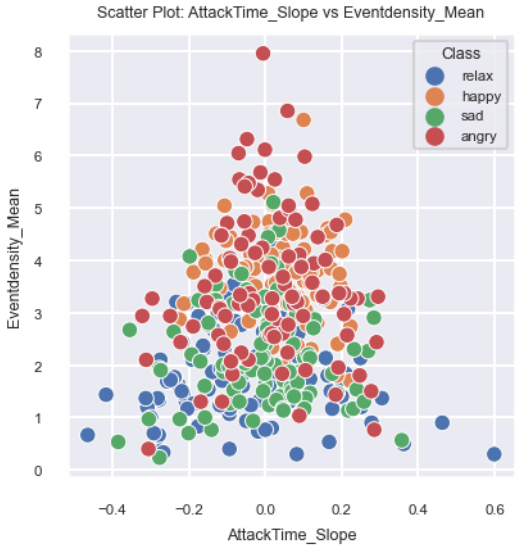
From comparing the HCDF period amplitude to the zero-crossing rate mean, we can hypothesise that the 'happy' class has the highest level of frequency/musical activity as well as chroma-centroid amplitude. In comparison, the 'relax' class has a much lower chroma centroid amplitude as well as a possible lower rate of musical activity.



From comparing the rolloff mean to the entropy of spectrum mean, we can hypothesise that as the noisiness of a signal increases, so does the average high frequency centroid. Interestingly, this feature pair has a similar distribution to the entropy of spectrum mean vs the zero-crossing rate mean. This makes some intuitive sense, as a noisier power spectrum implies a wider distribution of frequency content, and thus a higher average zero-crossing rate.

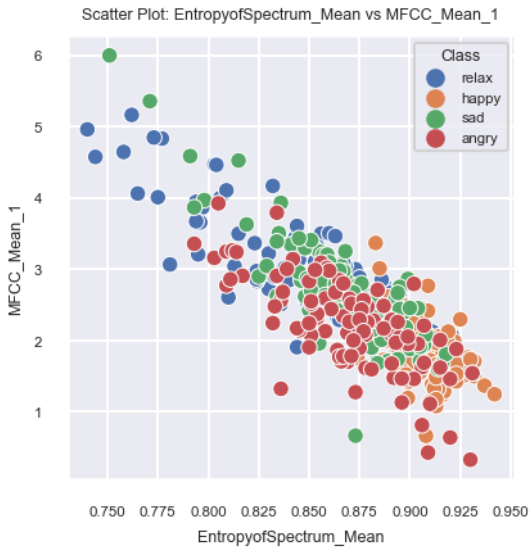


From comparing the event density mean with the roughness mean, we can see hypothesise as the sensory dissonance increases, so too does the average number of musical events per second. The plot also reveals that the samples with the highest sensory dissonance and/or event density are largely outliers from the 'happy' and 'angry' class labels.



From comparing the attack time slope (the derivative of the attack time for each rhythmic onset – positive attack time slope indicates the attack time is increasing over the course of the piece, negative slope indicates the attack time is decreasing) with the event density mean, we can hypothesise that as the event density increases the attack time throughout the piece is more often unchanging. This makes some intuitive sense, because a higher rate of musical events would require musicians to be more precise with their articulation. We can also note that the 'angry' and 'happy' labels have the most constant attack slope (converges to zero on the plot) whereas the 'relax' and 'sad' labels have a considerable amount of variability in attack slope, with a much lower mean event density.

From comparing the HCDF period amplitude to the Major-ness feature, we can hypothesise that compositions with a louder chroma centroid are slightly less likely to have a subjective major tonality. Furthermore, we can note that 'angry' and 'happy' labels are more likely to have a louder chroma centroid, but that 'angry' is less likely to have major key tonality than 'happy'.



From comparing the entropy of spectrum mean to the mean magnitude of the 1st MFCC coefficient, we can hypothesis that the average noisiness of a composition has an inverse relationship with the subjective lowest frequency of the signal; as the entropy of spectrum increases, so too does the power spectrum of the signal, indicating that higher spectrum entropy would create a higher variance across the mean MFCC coefficients.

Data Modelling: Hyperparameter tuning
To choose the optimal parameter values for both the decision tree and random forest model, I decided to apply a randomized search to each model's optimal parameters, by using scikit-learn's RandomizedSearchCV module. Importantly, the random forest model's parameters shares all of the decision tree model's parameters. This is because the Random Forest model is a variation on decision tree model that 'fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.'('1.10. Decision Trees — scikit-learn 1.2.2 documentation' n.d.) I will first justify my choice of chosen parameters and ranges, and then interpret the results of the parameter searches.

Parameters common to both models:

criterion: ['gini', 'entropy', 'log_loss']
I included all three possible sk-learn options for the criterion search, 'gini','entropy' and 'log_loss'. Where Gini is the 'gini impurity score' function and log_loss and entropy are methods of formulating Shannon information gain. I decided to include all three parameters as this was a small search space. Please note that 'log_loss' requires a newer version of scikit-learn.

Max_depth: [1..16, None]
Because there are still 11 relevant features in the data after feature selection, I decided to make the max depth search space relatively large [1..16] In order to prevent either underfitting or overfitting the model. I also included the default 'None' value in the search space to assess performance on the case where the tree nodes continue to expand until they either are pure or run out of samples.

Max_features: [1..num features]
I set the max feature search space to between 1 and the maximum number of features assessed as important for each model (in this case each model has 11 chosen features). I chose a range between 1-11 features that a broad range of different feature subsets could be assessed.

min_samples_split: [2, 8, 12, 16, 20, 24, 28, 32, 36, 40]
With some experimentation, I set the min samples split parameter to be between 2% - 10% of the total number of samples (400). In using this range of values, I allow for the model to explore different thresholds for splitting nodes, where lower values may result in more splits and finer granularity, and higher values may encourage more generalization by reducing the number of splits.

min_samples_leaf: [1..20]
With some experimentation, I set the min samples split parameter to be between 1% - 5% of the total number of samples (400). Similar to min_samples_split, by providing a range of values I allow for exploration of different thresholds in forming leaf nodes; Lower values may result in more specific leaf nodes – potentially capturing more noise – while higher values encourage more generalization by forming leaf nodes with larger sample sizes.

Parameters common to Random Forest:

n_estimators: [10, 60, 110, 160, 210]
This parameter defines the number of decision trees to be included in the random forest. With some experimentation, I chose the search range of 10 to 200, increasing in steps of 50. I decided to include a range of values to allow for exploration of both smaller and larger forest sizes.

Bootstrap: [True, False]
Bootstrapping involves randomly selecting samples with replace from the original dataset to create a new subset. In the context of random forests, a separate decision tree is trained on each bootstrap sample. I chose to allow for the parameter search to attempt Random Forest modelling with bootstrapping on and off.

Results:

As these models are stochastic in nature, I will note that I am interpreting all models with a random_state=0 for all functions (test_train_split, DecisionTreeClassifier, RandomForestClassifier, RandomizedSearchCV). This allows for some between tests. However, in deployment, these models may perform better or worse. The model was trained with a standard test-train-split test size of 0.3. For the randomised search parameters, I found received the best training results from setting the cross validation to 5, and the number of search iterations to 300.

Random Search results – Decision Tree:

Optimal random search space parameters:	criterion': 'entropy' 'min_samples_split': 8 'min_samples_leaf': 4, 'max_features': 8, 'max_depth': 10
Mean best random search space search score:	0.712
Test split score with best estimator parameters:	0.725

Random Search results – Random Forest:

Optimal random search space parameters:	'criterion': 'entropy' 'min_samples_split': 8 'min_samples_leaf': 1, 'max_features': 2, 'max_depth': 15 'n_estimators : 210' 'bootstrap' : True
Mean best random search space search score:	0.767
Test split score with best estimator parameters:	0.756

Discussion:

As we can see from the results, the Random Forest model performed marginally better than the decision tree model in classifying samples from the *Turkish Music and Emotion Dataset*. Interestingly, the random search parameters found a parameter space using the random forest model that used fewer features, but was much more granular in its approach to classification, whereas the decision tree model used a larger subset of features but had slightly more generalised classification parameters. The random search selected the same scoring criterion and minimum samples split for both models. Interestingly when the ideal features selected by the permutation feature selection function were swapped between models, the decision tree model had a mean best search score: *0.717* and a test split score: of *0.638*, whilst the random forest model had a mean best search score of *0.792*, and a test split score of *0.719*. However, the random search that output these scores also selected only 1 feature for the parameter search space, possibly suggesting that the random forest is overfitting when trained on the features selected for the decision tree model.

Conclusion:

In this study, data from the *Turkish Music and Emotion Dataset* was used to train decision tree and random forest models in combination with a random search for parameter hypertuning. The random forest model performed slightly better when trained on spectral and mel frequency cepstrum coefficient features selected for it, whereas the decision tree model performed better when trained on chromagram features, as well as engineered tonal features. Because the random forest performed slightly better, this may suggest that spectral frequency features are more useful than chromagram and 12 tone features when modelling Turkish and microtonal music. Further modelling could be carried out using these techniques, with a knowledge of what features are now most useful, however, a much larger corpus of music data will be required.

Bibliography:

‘1.10. Decision Trees — scikit-learn 1.2.2 documentation’, viewed 25 May 2023, <https://scikit-learn.org/stable/modules/tree.html#tree-mathematical-formulation>.

‘Arabic, Turkish, Persian - Xenharmonic Wiki’, viewed 25 May 2023, <https://en.xen.wiki/w/Arabic,_Turkish,_Persian#Turkish>.

Bilal Er, M & Aydilek, IB 2019, ‘Music Emotion Recognition by Using Chroma Spectrogram and Deep Visual Features’:, *International Journal of Computational Intelligence Systems*, vol. 12, no. 2, p. 1622.

‘Feature importances with a forest of trees’ *scikit-learn*, viewed 24 May 2023, <https://scikit-learn/stable/auto_examples/ensemble/plot_forest_importances.html>.

Lartillot, O 2021, ‘MIRtoolbox 1.1 User’s Manual’,.