Music Genre Classification Using Multiple Classifiers

Machine Learning Project

Lisa Korntheuer, Jan Birkert, Adrian Desiderato, Jan Wangerin, Spyridon Spyropoulos

Technische Hochschule Ulm

January 15, 2025

Table of Contents

Motivation

- Music spans diverse genres, each with different sound characteristics.
- Classifying songs into genres using machine learning is worth trying.
- Few apps focus on genre classification compared to apps like *Shazam*, which identify song using databases.
- Our project uses a Kaggle dataset to test four supervised learning algorithms:
  - Random Forest (RF)
  - Decision Trees (DT)
  - k-Nearest Neighbors (kNN)
  - Artificial Neural Networks (ANN)

Table of Contents

Data Understanding

- GTZAN Dataset: collections of same length song snippets
  - 1000 soundtracks
  - 10 genres, 100 soundtracks per genre
- 29 different features have been extracted using librosa library
  - Fourier analysis
  - mean and variance for every feature (except tempo) $\rightarrow$ 57 features
  - only numerical data
  - no missing values

Data Understanding

- Temporal features:
  - sign changes, loudness
- Spectral features:
  - information about contained frequencies
- Rhythmic features:
  - tempo, information about contained rhythms
- Harmonic features:
  - information about relations between different pitches
- Mel-Frequency Cepstral Coefficients (MFCCs):
  - information about short term energy spectra

Data Understanding

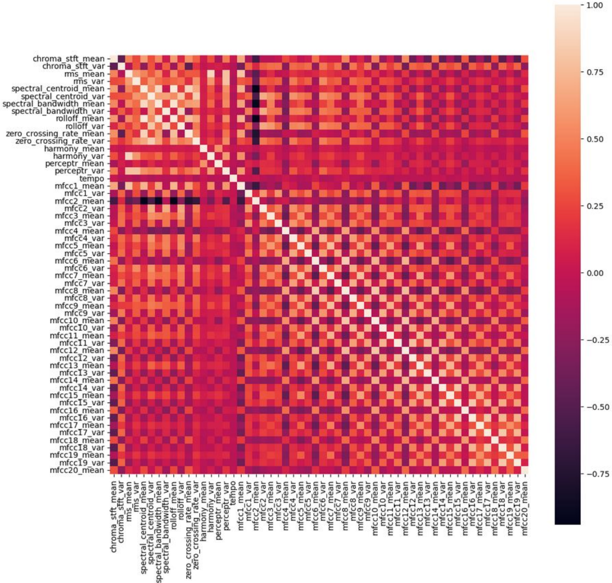


Table of Contents

Data Preparation

- Used LabelEncoder to encode genre labels: 0 - 9
- Used MinMaxScaler to scale feature data to a scale between 0 an 1

Table of Contents

Modeling — Random Forest

- Two experimenting approaches using GridSearchCV:
  - Soft hyperparameter tuning:
    - Number of trees (default = 100)
    - Splitting criterion (default = Gini)
  - Heavy hyperparameter tuning:
    - Number of trees (default = 100)
    - Splitting criterion (default = Gini)
    - Maximum depth (default = None)
    - Minimum samples per leaf (default = 1)
    - Minimum samples per split (default = 2)
    - Maximum features (default = sqrt)

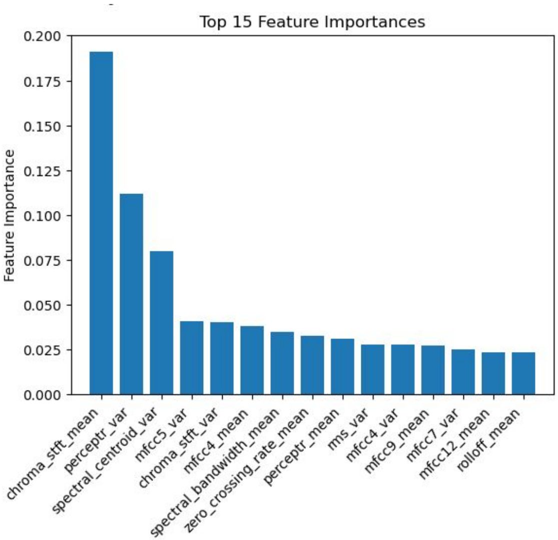
Modeling — Random Forest: Results

- Soft hyperparameter tuning:
  - Training time: 20s
  - Accuracy: 0.760
  - Parameters: Number of trees = 1000, Splitting criterion = Gini
- Heavy hyperparameter tuning:
  - Training time: 18min
  - Accuracy: 0.760
  - Parameters: Exactly the same forest as in the soft hyperparameter tuning

Modeling — Decision Trees

- No pruning: Test Accuracy 51%   Depth: 19
- Post-Pruning: Test Accuracy 49%   Depth: 14
  - Hyperparameter Tuning: regularization parameter α
- **Pre-Pruning: Test Accuracy 53%   Depth: 9**
  - Hyperparameter Tuning: depth, splitting criterion (result "entropy")

Modeling — Decision Trees



Modeling — K-Nearest-Neighbors

Hyperparameter tuning of 3 selected parameters with *GridSearchCV*:

- **n_neighbors** : number of neighbors k
- **weights** : weights assigned to the nearest neighbors
  - 'uniform'
  - 'distance'
- **metric** : method for distance computation
  - 'euclidean'
  - 'manhattan'

Lisa Korntheuer, Jan Birkert, Adrian Desiderato, Jan Wangerin, Spyridon Spyropoulos

Modeling — K-Nearest-Neighbors: Results

- **Optimal parameter set:**
  - $k = 3$ nearest neighbors
  - distance-dependent weights
  - Manhattan distance

  Remark: Distance-dependent weights might lead to overfitting
- **Test accuracy:** 0.74

Modeling — Artificial Neural Network

- Goal: Find the **best combination of hyperparameter values** to train a **Multilayer Perceptron** model
- The **following hyperparameters** were examined by means of a **Randomized Search Cross Validation**:
  - Two hidden layers with random number of neurons $\in [2, 200]$
  - The activation functions `tanh`, `relu` or `logistic`
  - α-values of either 0.0001, 0.001 or 0.05
  - A random learning rate $\in [0.001, 0.01]$
  - A batch size $\in [16, 128]$
  - `constant`, `adaptive` and `invscaling` learning rates were tested
- Cross Validation was run with **10 folds**

Lisa Korntheuer, Jan Birkert, Adrian Desiderato, Jan Wangerin, Spyridon Spyropoulos

Modeling — Artificial Neural Network: Results

- After running the hyperparameter tuning **10 times**, our best result was:
  - Two hidden layers with random number of neurons $\in [2, 200]$
  - The activation functions `tanh`, `relu` or `logistic`
  - α-value of 0.001
  - A random learning rate of ~ 0.0014
  - A batch size of 63
  - An `invscaling` learning rate
- With these parameters, we achieved a **train accuracy of 97.25%** and a **test accuracy of 77.0%**
- **Remarks:**
  - *We also tested with a third hidden layer without improvement*
  - Additionally, we also tried the `lbfgs` activation function which led to way worse results

Table of Contents

Comparison — Results

Model	Accuracy	F1 Score	ROC AUC
MLP	0.770	0.772	0.956
Random Forest	0.760	0.761	0.958
KNN	0.740	0.743	0.913
Decision Tree	0.530	0.532	0.759

Table: Model performance comparison.

Lisa Korntheuer, Jan Birkert, Adrian Desiderato, Jan Wangerin, Spyridon Spyropoulos

Comparison — ROC Curves

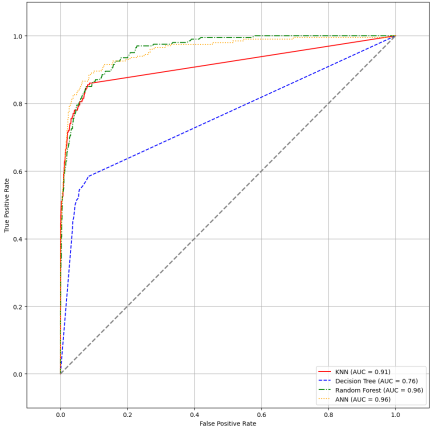


Figure: ROC Curves of the different classifiers

Experimentation — 13 Datapoints

Datapoint	True Label	Random Forest Prediction	KNN Prediction	Decision Tree Prediction	ANN Prediction
0	pop	hiphop	hiphop	hiphop	blues
1	pop	hiphop	hiphop	disco	hiphop
2	pop	hiphop	hiphop	hiphop	hiphop
3	metal	hiphop	hiphop	hiphop	blues
4	metal	hiphop	blues	hiphop	hiphop
5	blues	blues	blues	disco	blues
6	blues	blues	blues	country	hiphop
7	blues	hiphop	reggae	hiphop	reggae
8	classical	jazz	jazz	disco	hiphop
9	classical	jazz	jazz	country	blues
10	classical	jazz	jazz	jazz	hiphop
11	rock	hiphop	hiphop	hiphop	reggae
12	rock	hiphop	blues	hiphop	hiphop
13	rock	hiphop	reggae	hiphop	reggae

Table: Comparison of predictions from Random Forest, KNN, Decision Tree, and ANN against true labels.

Table of Contents

Conclusion

- Each classifier (KNN, Decision Tree, Random Forest, ANN) had different trade-offs
- Decision Trees performed poorly, even with pruning
- Random Forest and ANN showed strong performance
  - ANN with best accuracy ($\sim$ 77%)
  - kNN showed surprisingly good results
- Random Forests, ANNs, and kNNs are suitable for music classification
  - However: $\sim$ 25% of predictions may be incorrect
- Practical example with own tracks showed poor accuracy

References

[1] “Machine learning lectures and notebooks,” available at: https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[2] “Randomforestclassifier,” scikit-learn Documentation. Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[3] “Mlpclassifier,” scikit-learn Documentation. Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier.

[4] “Kneighborsclassifier,” scikit-learn Documentation. Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[5] “Randomizedsearchcv,” scikit-learn Documentation. Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.

[6] “Gridsearchcv,” scikit-learn Documentation. Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.model_selection.GridSearchCV.html.

[7] “Minmaxscaler,” scikit-learn Documentation. Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.preprocessing.MinMaxScaler.html.

[8] “One-vs-rest and one-vs-one for multi-class classification,” available at: https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/.

[9] “Decisiontreeclassifier,” scikit-learn Documentation. Available at: https://scikit-learn.org/dev/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[10] “F1 score in machine learning,” available at: https://encord.com/blog/f1-score-in-machine-learning/#:~:text=Typically%2C%20an%20F1%20score%20%3E%200.9,to%20have%20a%20poor%20performance.

[11] “Minimal studio - download wav,” available at: https://www.minimalstudio.de/download-wav/.

[12] “Pixabay - search wav,” available at: https://pixabay.com/music/search/wav/.

[13] “Librosa features,” available at: https://librosa.org/doc/latest/feature.html#module-librosa.feature.

[14] G. Peeters, “A large set of audio features for sound description (similarity and classification) in the cuidado project,” 2003, available at: http://recherche.ircam.fr/anasyn/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf.