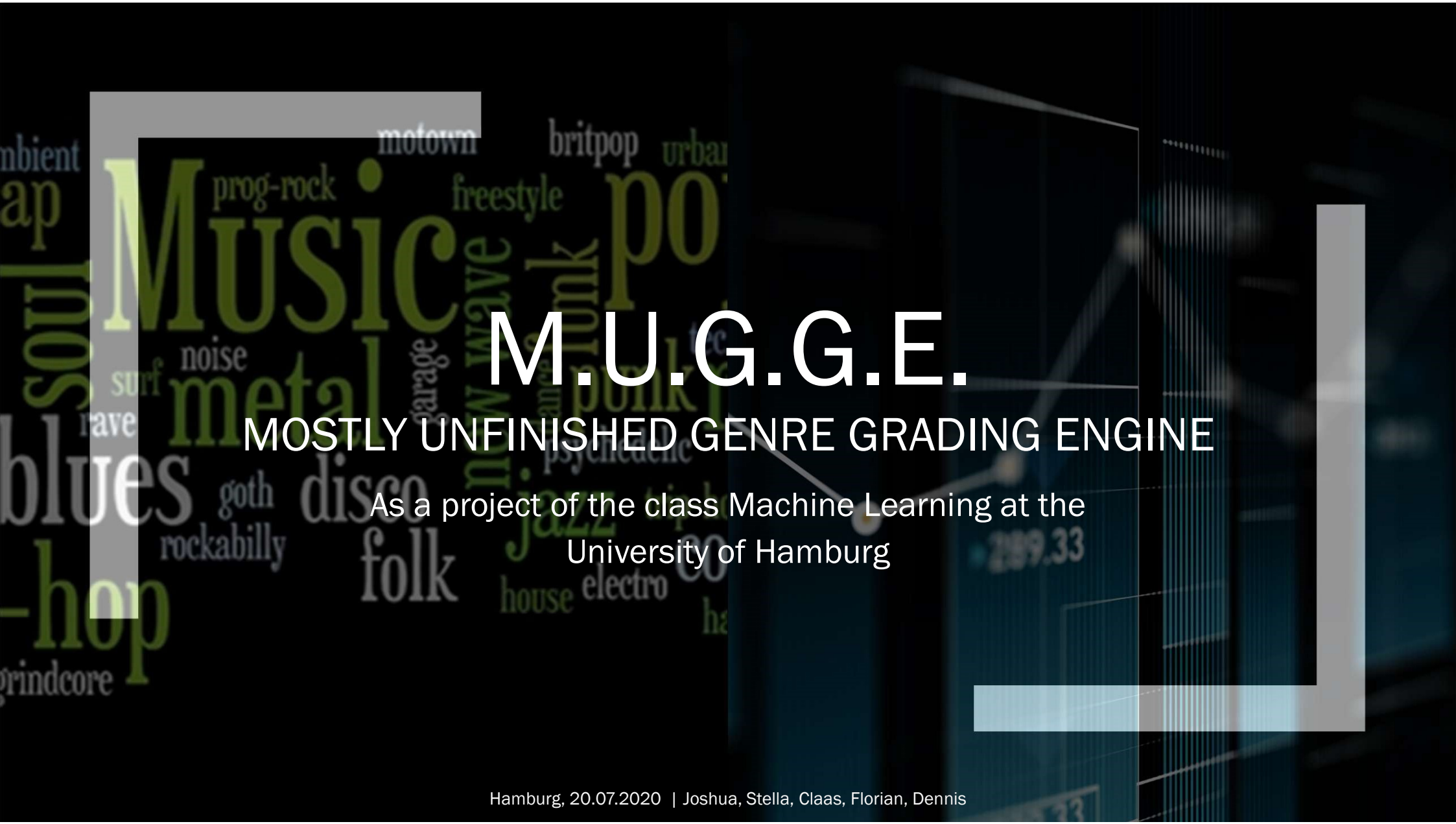


M.U.G.G.E.

MOSTLY UNFINISHED GENRE GRADING ENGINE

As a project of the class Machine Learning at the
University of Hamburg

Hamburg, 20.07.2020 | Joshua, Stella, Claas, Florian, Dennis

Goals

- Build a high performance model for basic learners
- Use a balanced medium size dataset to classify 10 genres
- Investigate different classifiers finding the most efficient one for classifying our genres
- High performance predictions recording new input
- Build a model capable of efficient predictions given new input
- Build a GUI

Music Genres

- Blues
- Classical
- Country
- Disco
- Hip Hop
- Jazz
- Metal
- Pop
- Reggae
- Rock

Dataset

- 1000 songs
- 100 songs for each genre
- Audio duration: 30 seconds
- Source of dataset: Kaggle.com, GTZAN Dataset-Music Genre Classification

Features

- Zero Crossing rate
- Spectral Centroid
- Mel-Frequency Cepstral Coefficients
- Chroma Frequencies
- Chords

Short-Time Fourier Transform

- Music signals are highly non-stationary, i.e., their statistics change over time.
- Meaningless to compute a single FT over an entire song
- The STFT is obtained by computing the Fourier transform for successive frames in a signal:

$$X(m, w) = \sum_n x(n)\omega(n-m)e^{-j\omega n},$$

with time m and frequency ω

- Since we care about the spectral magnitude and not the phase content, we use the spectrogram, which shows the intensity of frequencies over time.
- A Spectogram is simply the squared magnitude of the STFT:

$$S(m, \omega) = |X(m, \omega)|^2$$

Zero Crossing Rate

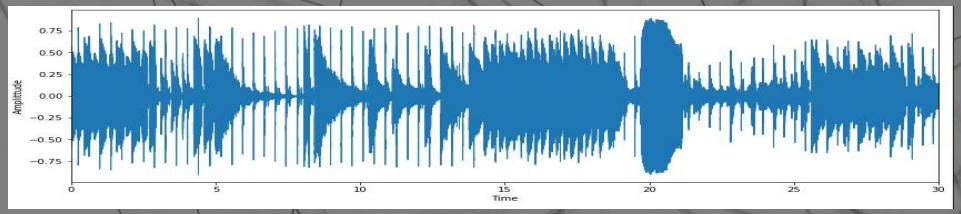
- The number of times the amplitude of the signal passes through a value of zero, within a time interval.
- Measure of smoothness of an audio signal
- Higher values for highly percussive sounds like those in metal and pop.

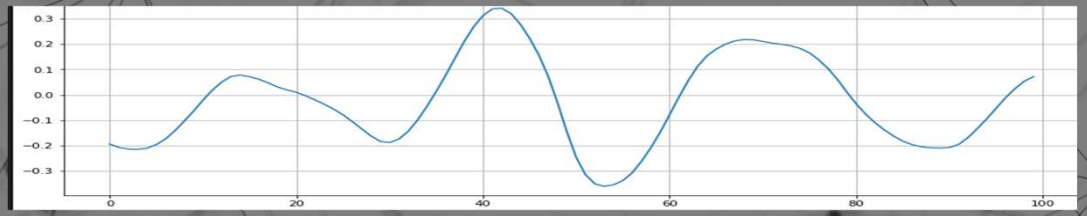
ZCR is defined formally as

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1} 1_{\mathbb{R}_{<0}}(s_t s_{t-1})$$

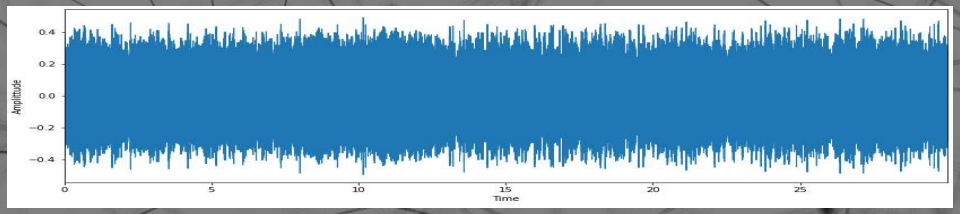
where s: signal of length T

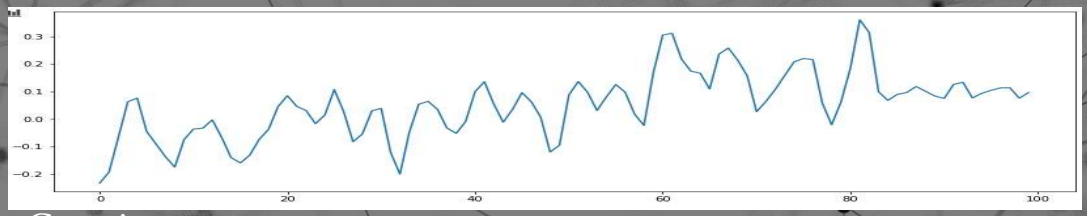
Blues Zero Crossings: 7



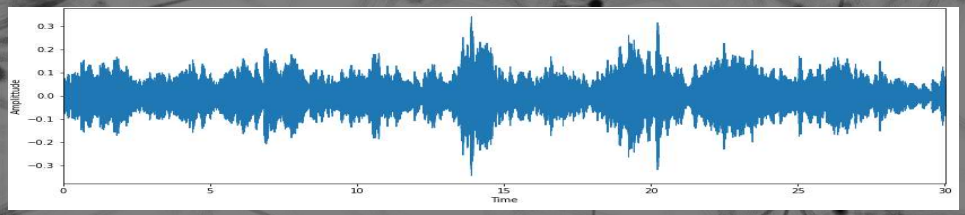


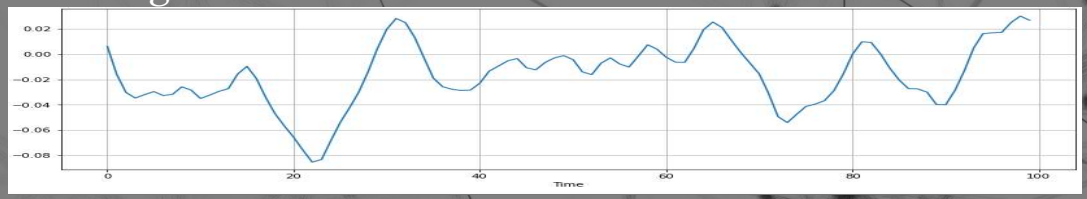
Metal Zero Crossings: 19



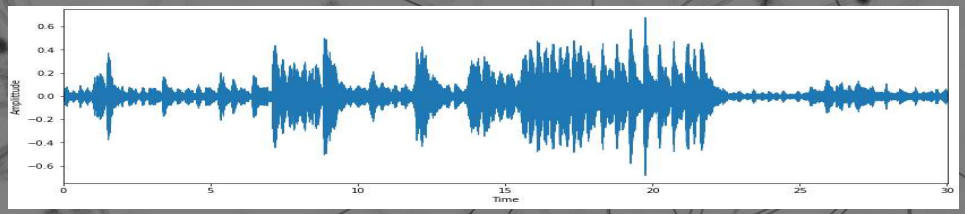


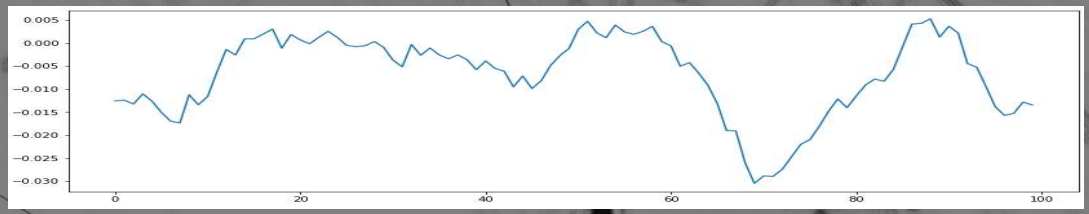
Classical Zero Crossings: 10



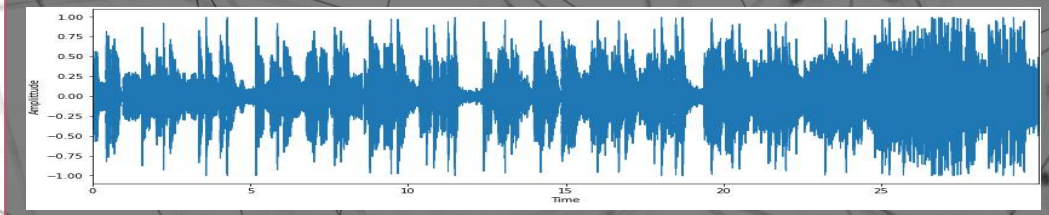


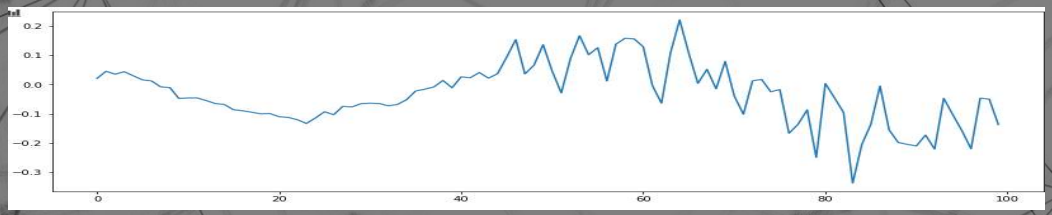
Jazz Zero Crossings: 12



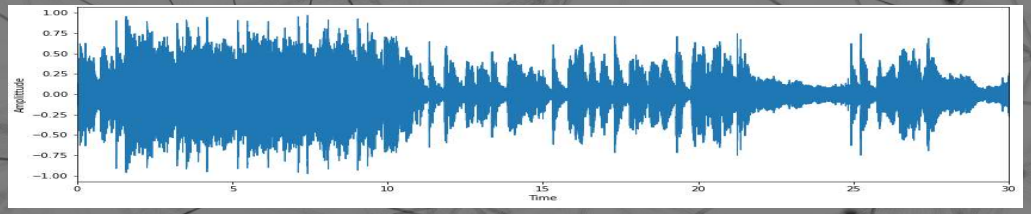


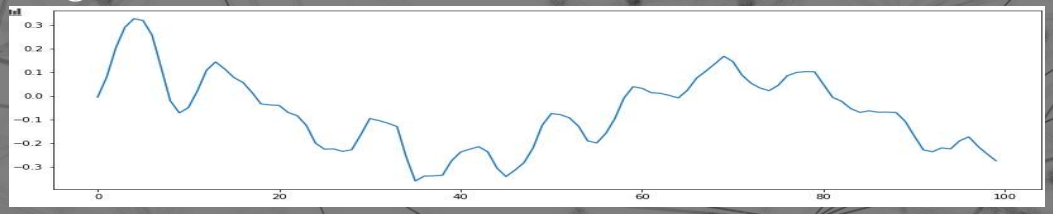
Pop Zero Crossings: 15



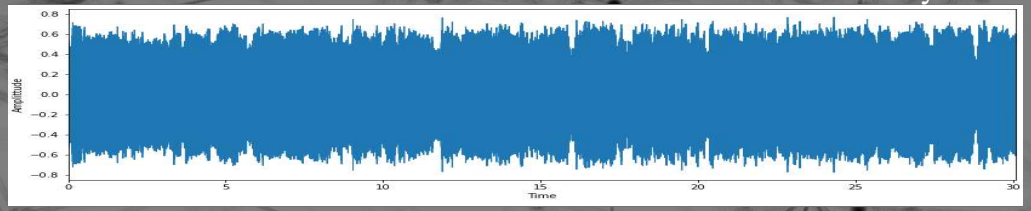


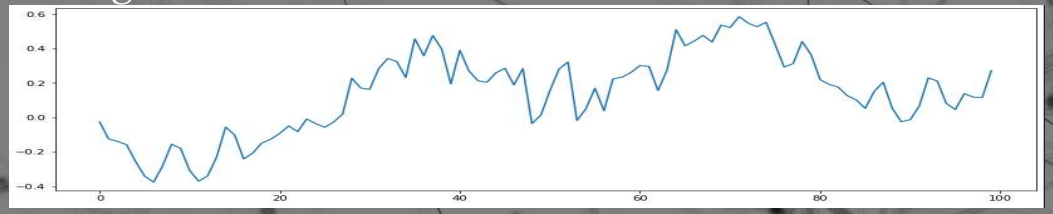
Rock Zero Crossings :8



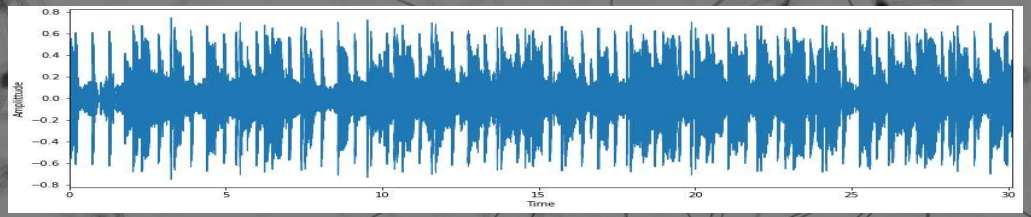


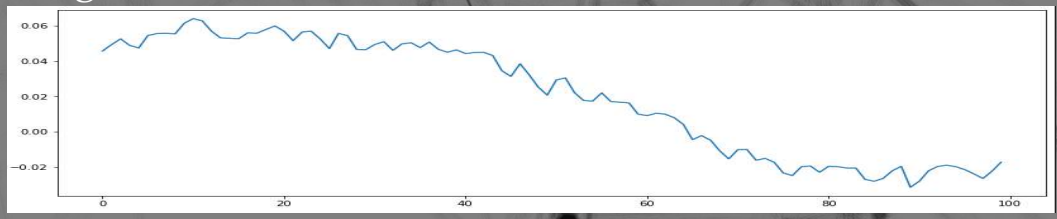
Country Zero Crossings :7



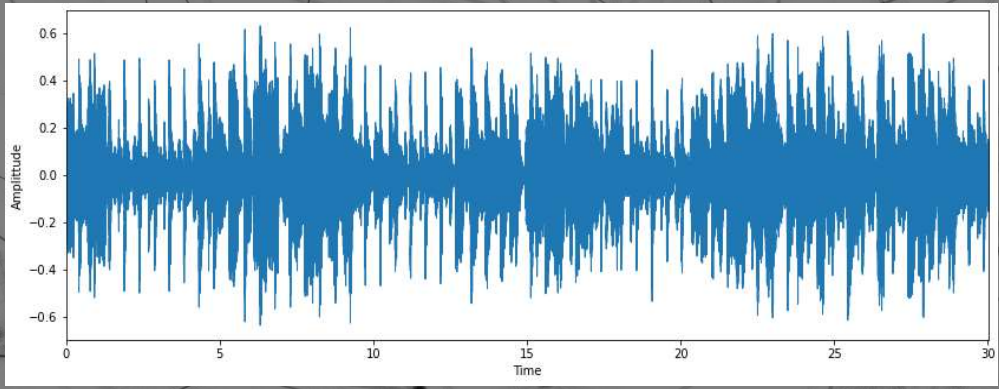


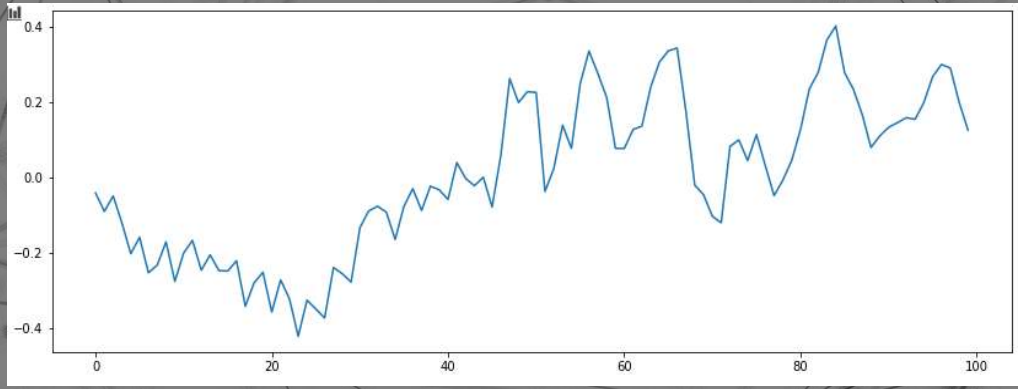
Disco Zero Crossings : 1



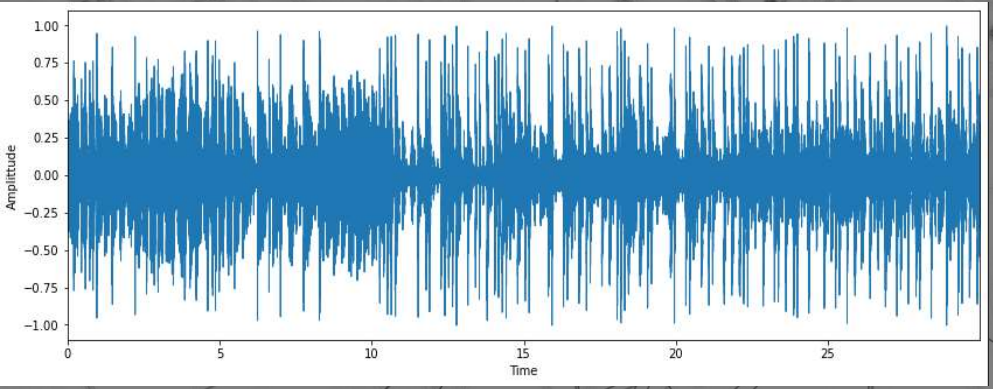


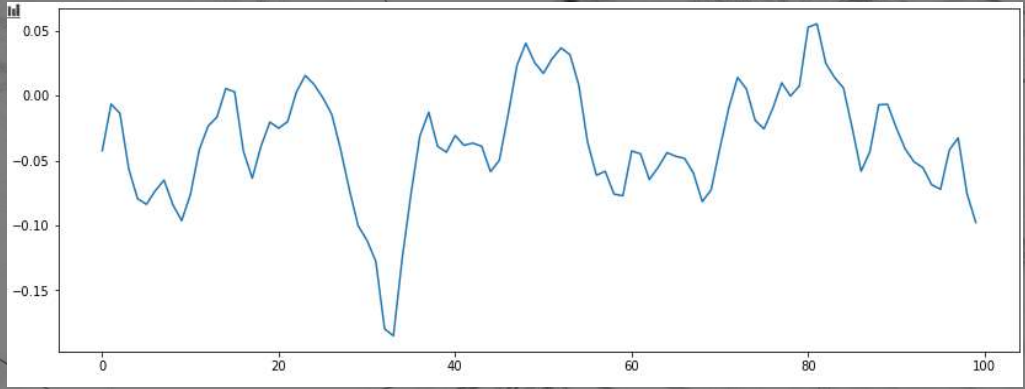
Reggae Zero Crossings:11





Hip Hop Zero Crossings: 12





Spectral Centroid

- Measure used to characterise a spectrum
- Indicates where the „centre of the mass“ for a sound is located
- Calculated as the weighted mean of the frequencies present in the sound

$$\text{Centroid} = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)}$$

- x(n) represents the magnitude, f(n) the mean frequency of every interval

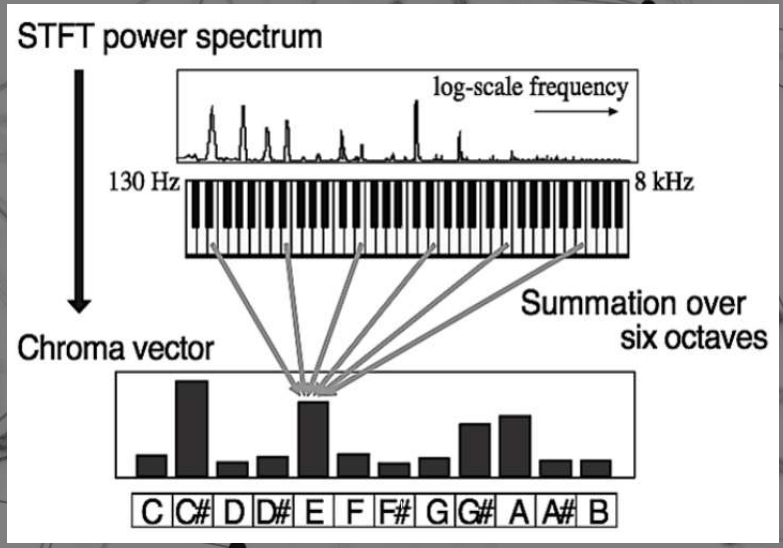
MFCC

- Small set of features (20 in our case)
- Describe the overall shape of a spectral envelope and models the charasteristics of the human voice
- Human perception of speech/sound is linear until 1000 Hz Range(100-3200 Hz) and logarithmic from threre
- Conversion from normal frequency f to mel frequency m:

$$m = 2595 \log_{10}((f/700)+1)$$

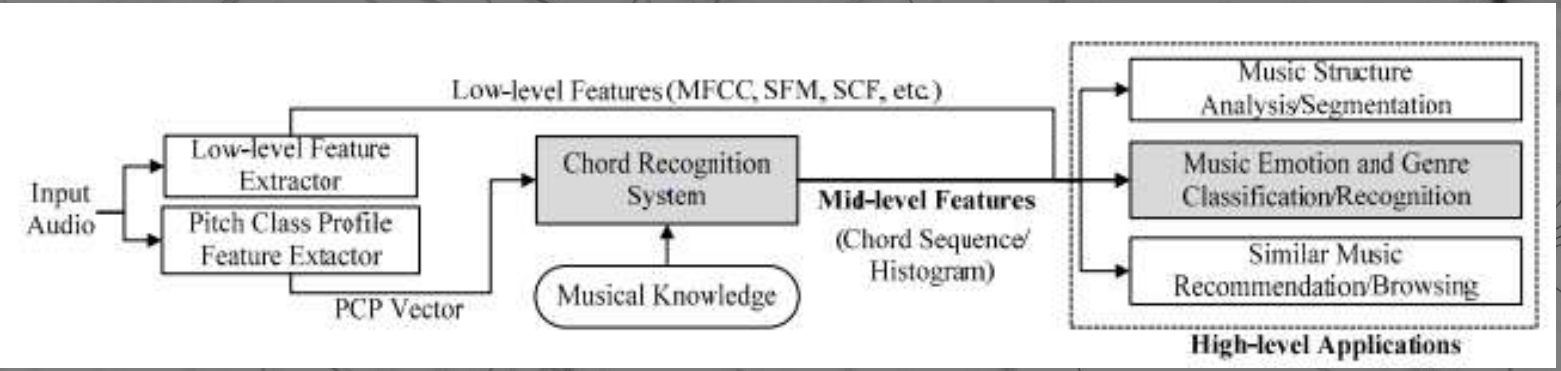
Chroma

- Project the whole spectrum onto the 12 distinct semitones (chromas) of the musical octave



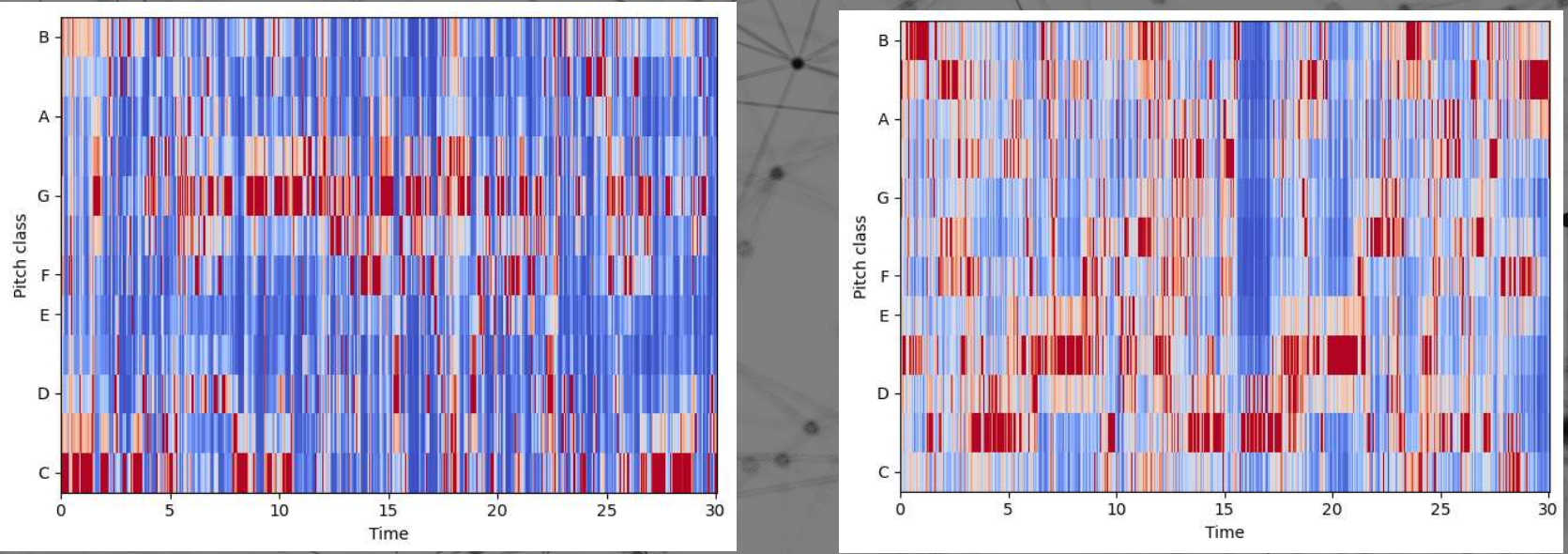
Mellina, Alessio & Sentinelli, Alexandro & Marfia, Gustavo & Roccetti, Marco. (2012). AREEB: Automatic REfrain extraction for ThumBnail. 2012 IEEE Consumer Communications and Networking Conference, CCNC'2012. 10.1109/CCNC.2012.6181003.

Chord Feature



- Taking the Chromagramm and extract the Pitch Class Profile
- Calculate the Transition Matrix
- Train the Neural Network

Chord Feature



Chromagrams for blues (left) and metal (right)

Our Classifiers

- Logistic Regression:
  - “One-versus-rest“ scheme, uses a quasi-Newton optimizer with a maximum of 100 iterations
- Random Forests:
  - Averages over multiple Decision Trees, which are only trained on subsets of the available data
- Multi-Layer Perceptron:
  - Supervised Neural Network with tunable size of hidden layers
  - Uses Backpropagation, and stochastic gradient descent or a quasi-Newton optimizer
- Support Vector Machine:
  - Parameters: c.f. Appendix

Model Selection: MLP

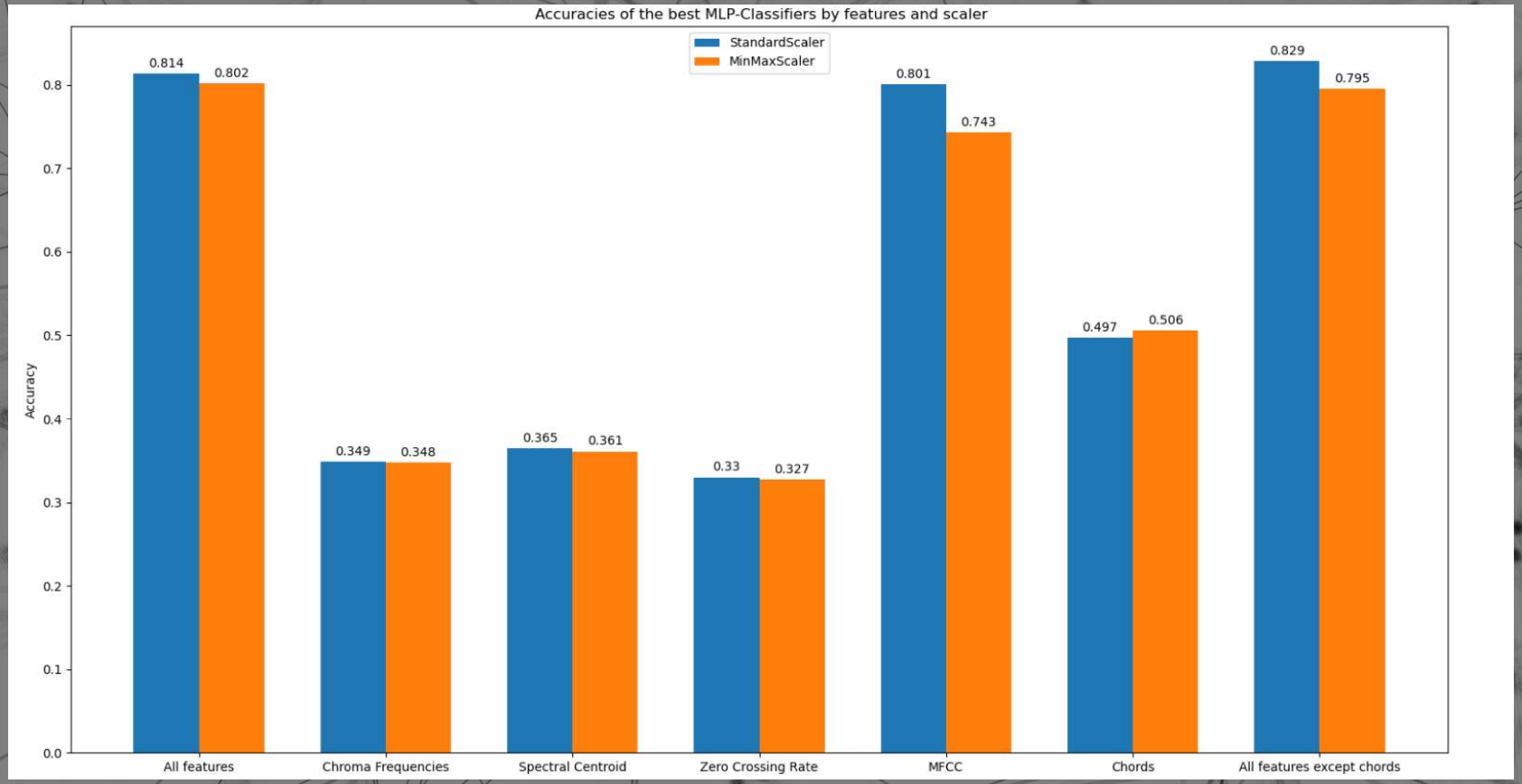
- Which Parameters form the model with the highest accuracy? Investigated options included:
  - hidden_layer_sizes: (100) , (100,50) or (50,50,50)
  - activation: ‘relu‘ or ‘logistic‘
  - solver: ‘lbfgs’ (quasi-Newton optimizer) or ’adam’ (based on stochastic gradient descent)
  - alpha: 0.0001 or 0.01
  - learning_rate: ‘constant‘, ‘invscaling’ or ‘adaptive’
  - max_iter: 250 or 500

→ Find the best parameters with a cross-validated grid-search over this parameter grid

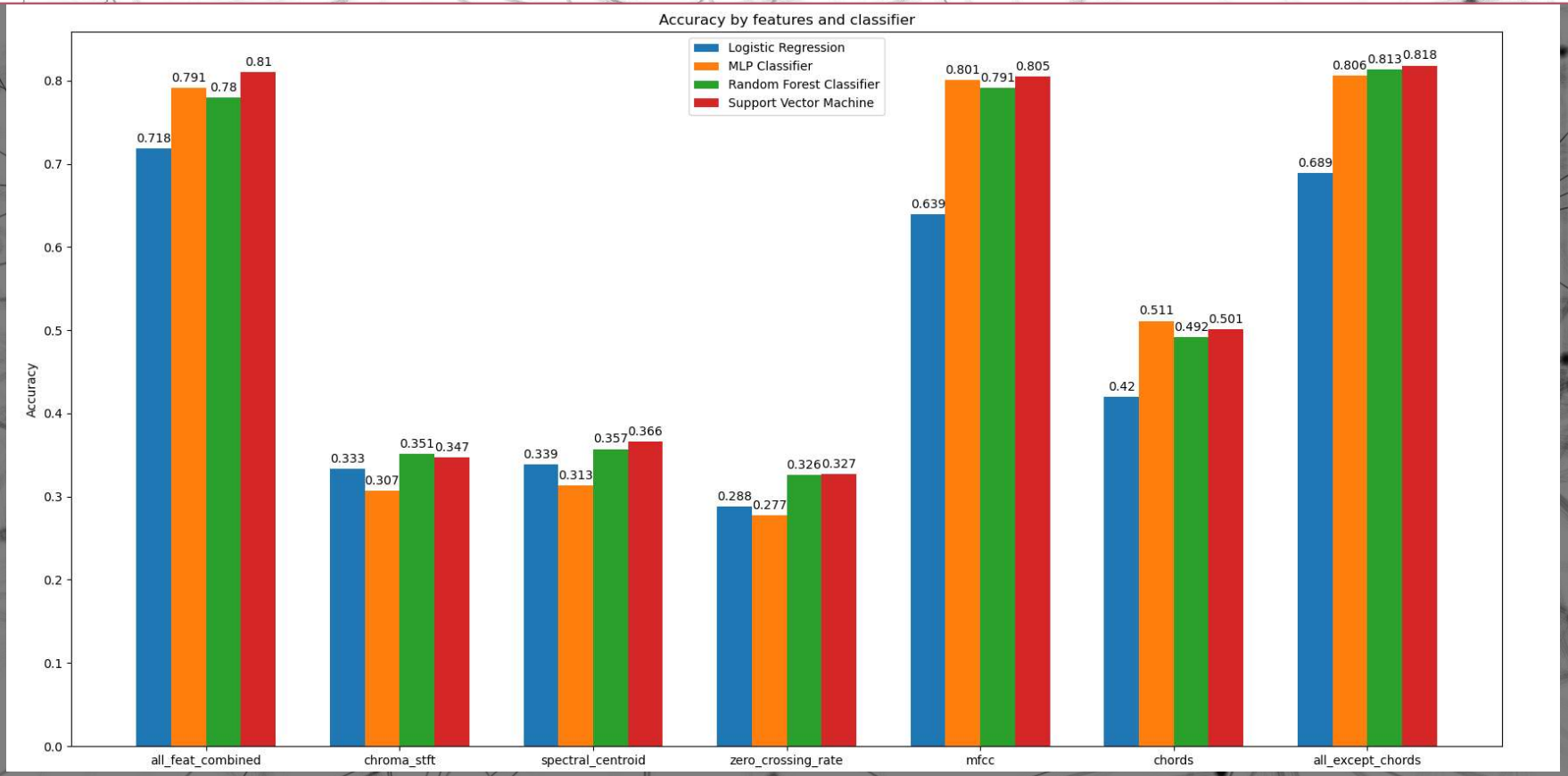
Parameters with highest accuracy

Best Parameters	Scaler	Hidden Layers	Activation function	Solver	Alpha	Learning rate	Maximum iterations	Accuracy
All features	Standard	(100,50)	relu	adam	0.0001	adaptive	250	0.814
Chroma F.	Standard	(100,50)	relu	adam	0.01	adaptive	500	0.348
Spectral C.	Standard	(100,50)	relu	adam	0.0001	adaptive	250	0.371
Zero C. R.	Standard	(50, 50, 50)	logistic	adam	0.01	adaptive	500	0.334
MFCC	Standard	(100,50)	relu	adam	0.0001	adaptive	250	0.81
Chords	MinMax	(100,50)	logistic	adam	0.0001	adaptive	250	0.501
All features except chords	Standard	(100,50)	relu	adam	0.0001	adaptive	250	0.827

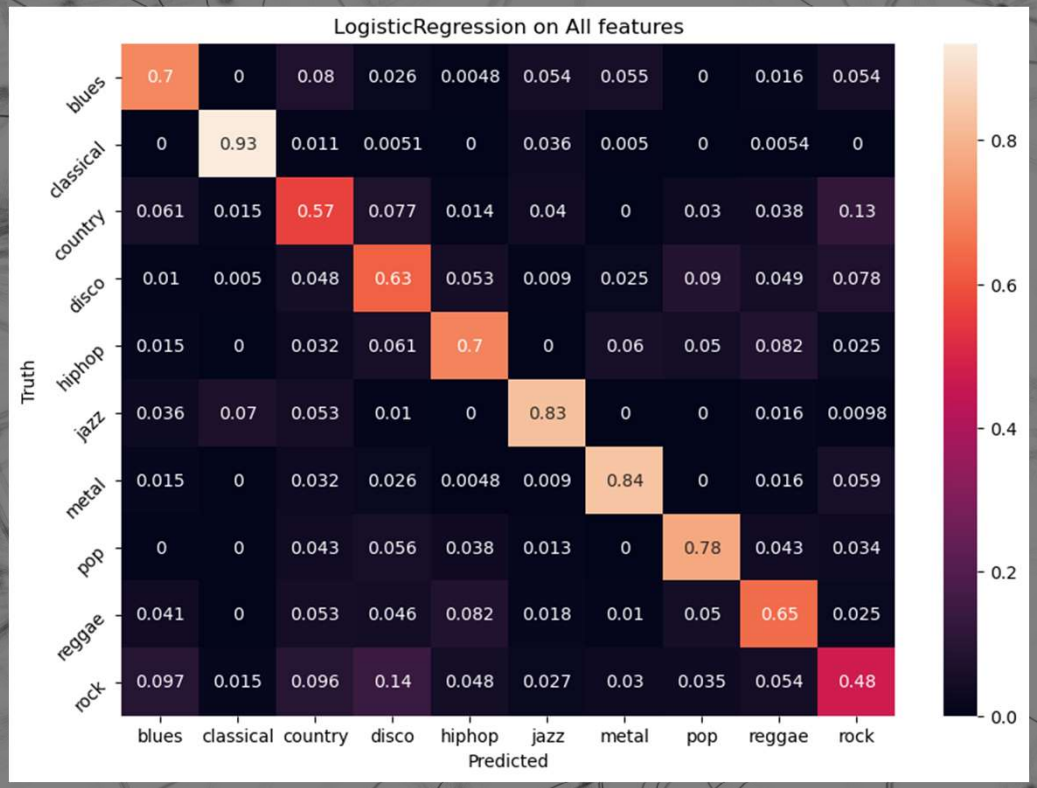
Parameters with highest accuracy

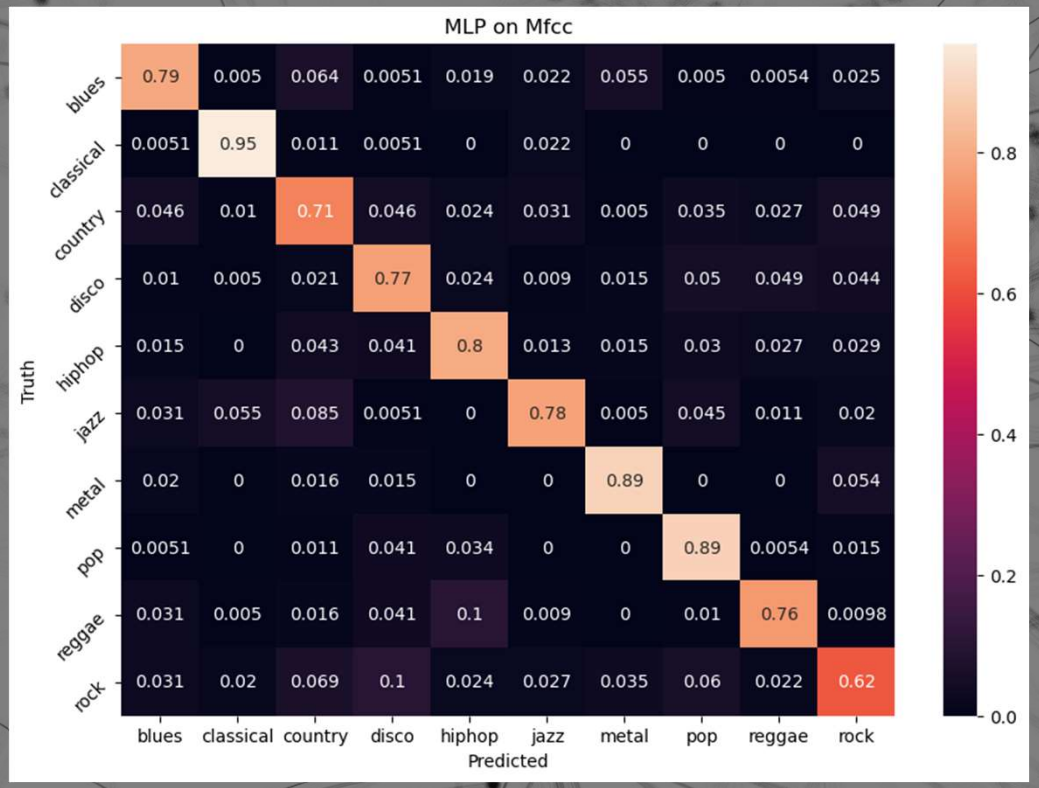


Evaluation: Accuracies

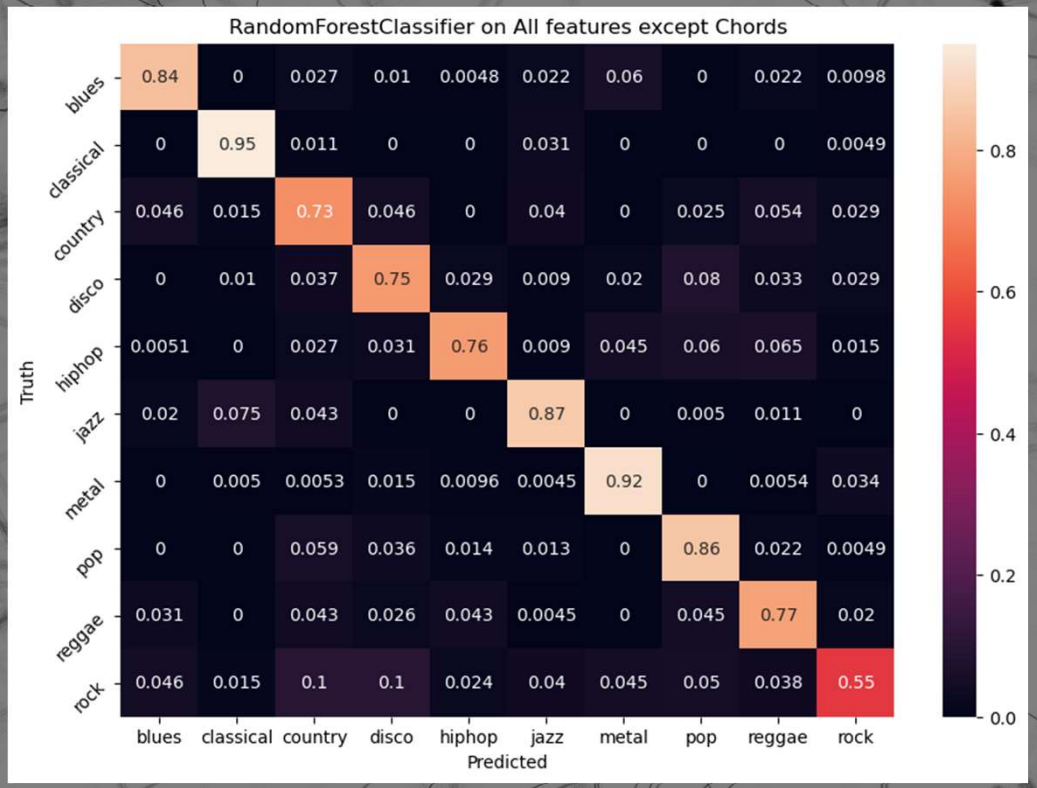


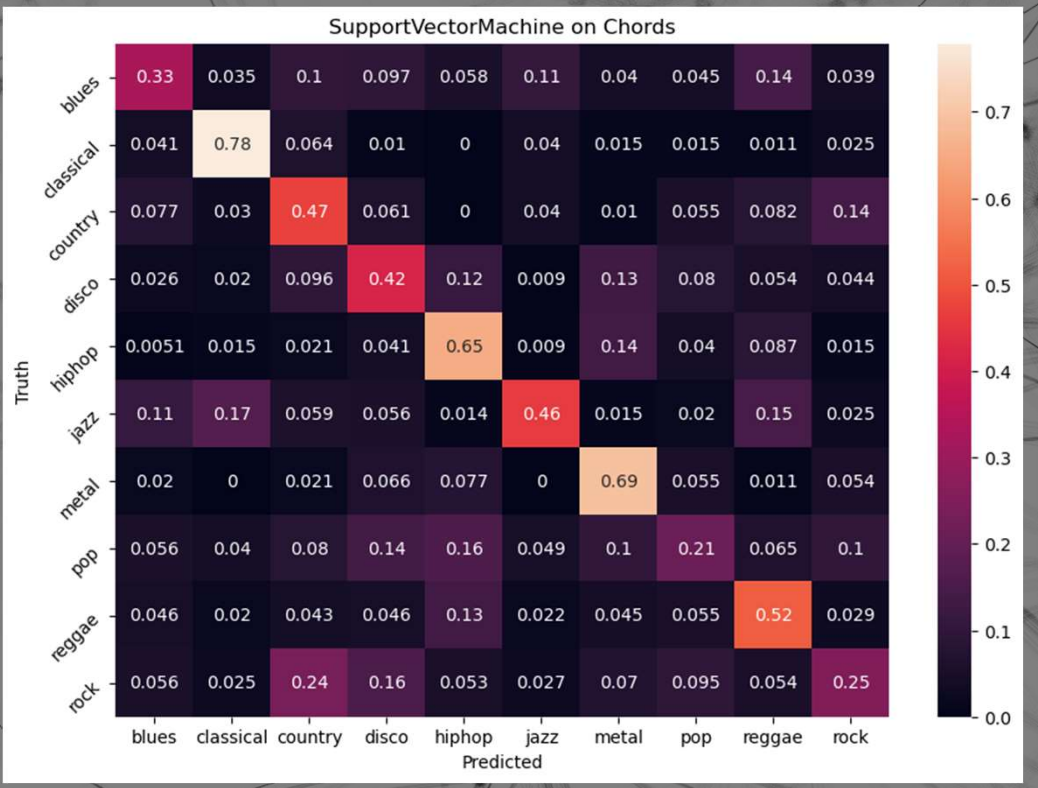
Confusion Matrizes





Confusion Matrizes





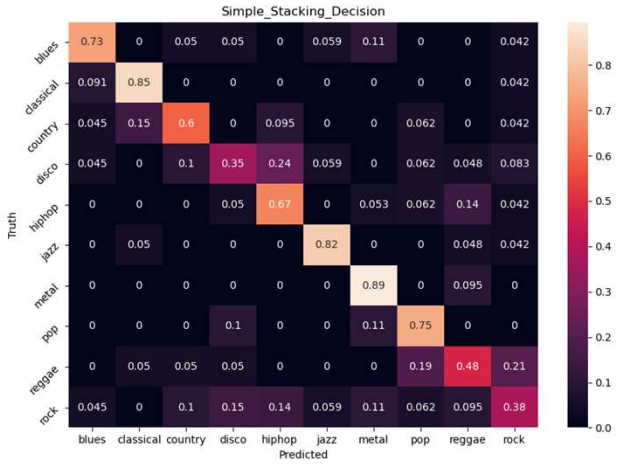
Ensemble Methods

- Reasons why ensemble learning might be better:
  - Training data might not provide sufficient information for choosing a single best learner
  - Search process of the learning algorithm for the best Hypthesis might be imperfect
  - Hypothesis space might not contain the true target function → ensemble could approximate

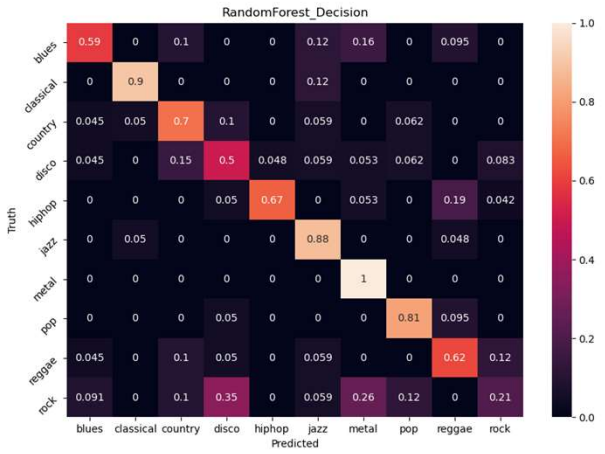
Ensemble Methods

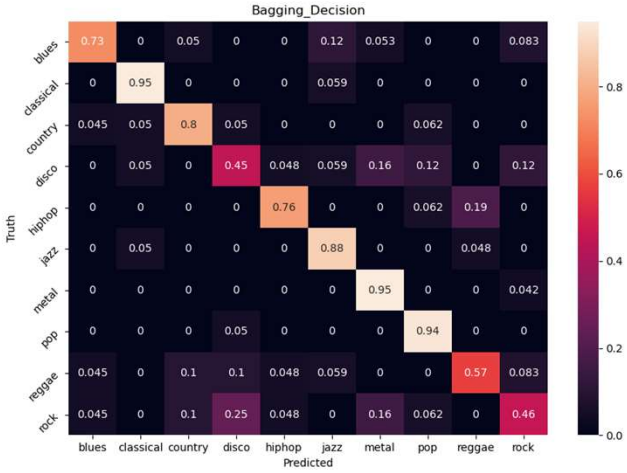
- Bagging:
  - Serveral samples upto the size of the actual data set are drawn from the data set, where certain elements can be drawn more than once
- Boosting:
  - Weights the dataset after each round of training and increases the weight of each incorrectly classified element
- Stacking:
  - Simply takes the predictions of an ensemble of base learners as an input for another machine learning method

Ensemble Methods



Accuracy: 73,5%





Accuracy: 64%

Accuracy: 67%

Input-Pipeline

Job:

- input costum data to be classified
- arrange data to be accessible for the feature- and model boxes

Approach:

- team needs 1-D numpy array of audio-samples, due to Librosa-specification
- therefore, develop routine to read/record audio-files and convert these into required numpy arrays
- libraries: pydub (requires: FFMPEG), sounddevice

Input-Pipeline

Read & Convert Files:

- input: File
- master function reads and delegates tasks to appropriate functions
- decided for WAV and MP3
- use dictionary to make code easily extendable

```python
def prepro(input_file):
    format_type = input_file[-4:]  # find formatextension
    # dictionary instead of if-else to avoid restructuring in case of additional formats
    formats = {
        ".wav": wav_to_array_file,
        ".mp3": mp3_to_array
    }
    # Get the NumPy array of an audiofile by calling function from formats dictionary
    try:
        myarray = formats.get(format_type)(input_file)
        return myarray
    except TypeError:
        print("Invalid format")
```

Input-Pipeline

Convert file to array:

- input: file
- only take 30 seconds of audio , needed for the classifyer
- each channel has an independent audio signal, so taking one channel for samples should suffice
- return: array

```python
def wav_to_array_file(input_file, duration=30, offset=0, normalized=False):
    """Convert an at least 70 seconds long WAV-file to a numpy array"""
    a = pydub.AudioSegment.from_wav(
        input_file)  # get array from AudioSegment object
    # slice = a[40000:70000]  # only take 30 sec. excerpt
    slice = a[1000 * offset:1000 * (offset + duration)]  # only take 30 sec. excerpt
    y = np.array(slice.get_array_of_samples())  # cast as NumPy array

    if a.channels == 2:
        y = y.reshape((-1, 2))
        y_new = y[:, 0]

    else:
        y_new = y

    return np.array(y_new, dtype=float)
```

Input-Pipeline

Record:

- record for 30 seconds to WAV
- technical part outsourced to Sounddevice
- convert WAV to array of samples
- return: array

```
def record(duration):
    """ Record as WAV then convert to NumPy array to return."""
    fs = 22050  # Sample rate
    seconds = duration  # Duration of recording
    myrecording = sd.rec(int(seconds * fs), samplerate=fs, channels=1)
    print(f"Recording for {seconds} sec.:")
    sd.wait()  # Wait until recording is finished
    print("Recording terminated")
    # print(myrecording)
    write('output.wav', fs, myrecording)  # Save as WAV file
    my_new = wav_to_array_rec(os.getcwd()+'\\output.wav')  # get NumPy array"
    #return myrecording
    return my_new
```

Outlook:

- Possible next/further steps:
  - Feature based: extract more features (eg. spectral-rollof), compare standard deviation with variance
  - Data based: Use 1million songs data set
  - Model based: scale with sklearn.Normalizer, cross-validated grid-search for all Classifiers
  - Train a Neural Network with only the images of the Spectograms
  - Evaluate effects of different input formats
  - GUI?

References 1/2

- https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification/data ; free available dataset
- https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8
- Valerio Velardo - The Sound of AI (YouTube Channel)
- https://towardsdatascience.com/musical-genre-classification-with-convolutional-neural-networks-ff04f9601a74
- MSc Computer Science, University of Birmingham,‘‘Machine Learning for Music Genre Classification‘‘, Author: Bryn Lansdown, Supervisor: Dr. Shan He
- https://www.youtube.com/watch?v=t0mNFAFdz_Q&t=267s Caio Miyashiro at Pydata Berlin 2019 about Chord Recognition
- Cheng, Heng-Tze & Yang, yi-hsuan & Lin, Yu-Ching & Liao, I-Bin & Chen, Homer. (2008). Automatic chord recognition for music classification and retrieval. 1505 - 1508. 10.1109/ICME.2008.4607732.

References 2/2

- Mellina, Alessio & Sentinelli, Alexandro & Marfia, Gustavo & Roccetti, Marco. (2012). AREEB: Automatic REfrain extraction for ThumBnail. 2012 IEEE Consumer Communications and Networking Conference, CCNC'2012. 10.1109/CCNC.2012.6181003.
- https://musicinformationretrieval.com/stft.html
- Trevor Hastie, Robert Tibshirani, Jerome Friedman, "The Elements of Statistical Learning - Data Mining, Inference and Prediction", Second Edition, Springer Series in Statistics, Springer, 2008
- https://scikit-learn.org/stable/modules/ensemble.html
- https://realpython.com/playing-and-recording-sound-python/
- https://github.com/jiaaro/pydub/
- https://ffmpeg.org/

Task Distribution

All team members: Find literature, pappers and investigate different approaches od the problem from multiple, open to the public sources

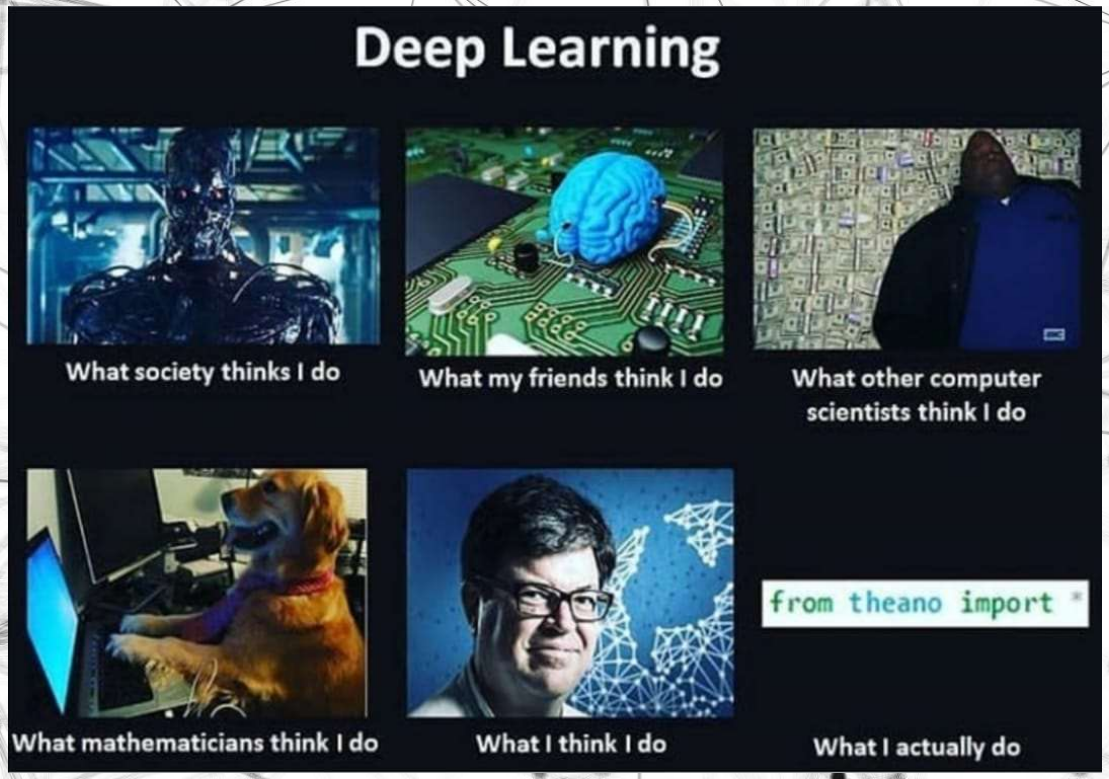
Dennis: Project Management, merge and control the Project Plan, part of feature extraction especially chord feature

Stella: Find dataset, Explore features needed to be extracted (coll. with Dennis, Joshua), Feature extraction from dataset and store in a csv file (coll. with Joshua), Import confusion matrices(coll. with Florian)

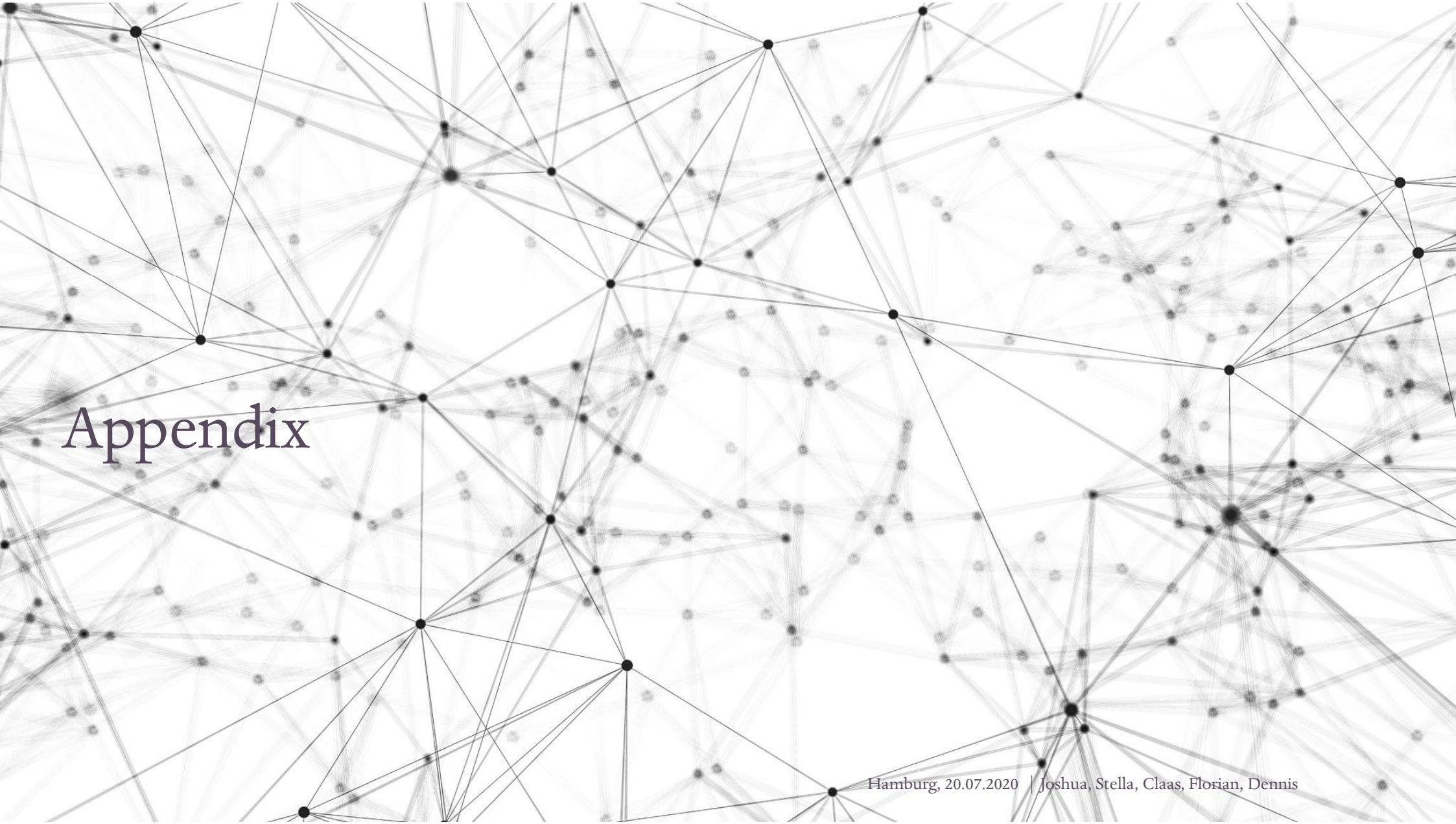
Joshua: Feature Extraction, Model Selection, Model Evaluation

Florian: Coding Manager, designed overall structure of the code and supported people during implementation, implementation of ensemble learners

Claas: design, implemenation and testing of input-box


Deep Learning
What society thinks I do
What my friends think I do
What other computer scientists think I do
What mathematicians think I do
What I think I do
from theano import *
What I actually do


VIELEN DANK FÜR EURE AUFMERKSAMKEIT
NOCH FRAGEN?
makeameme.org



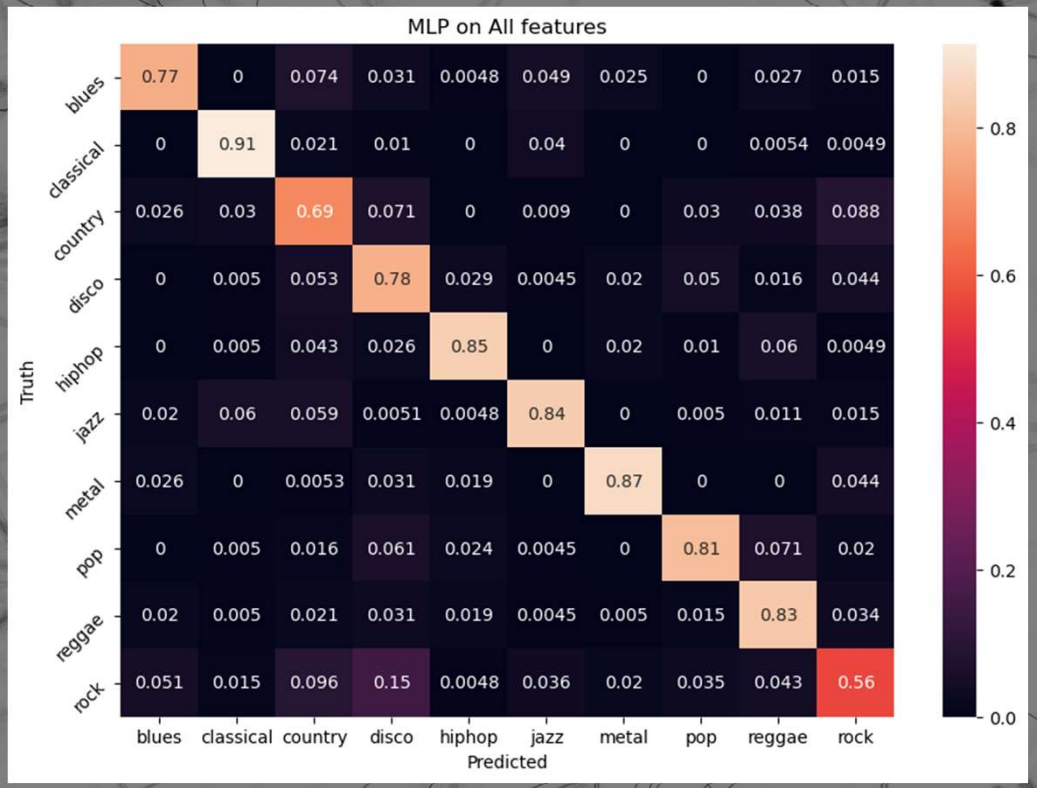
Appendix

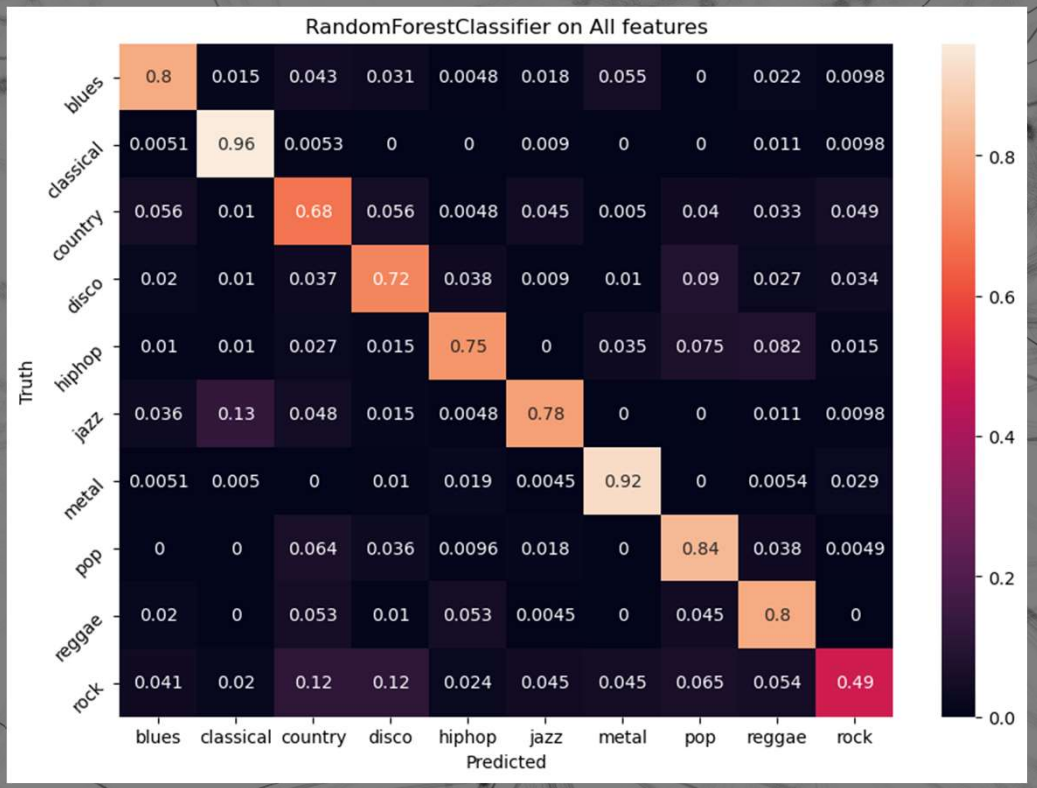
Hamburg, 20.07.2020 | Joshua, Stella, Claas, Florian, Dennis

Model Selection: SVM

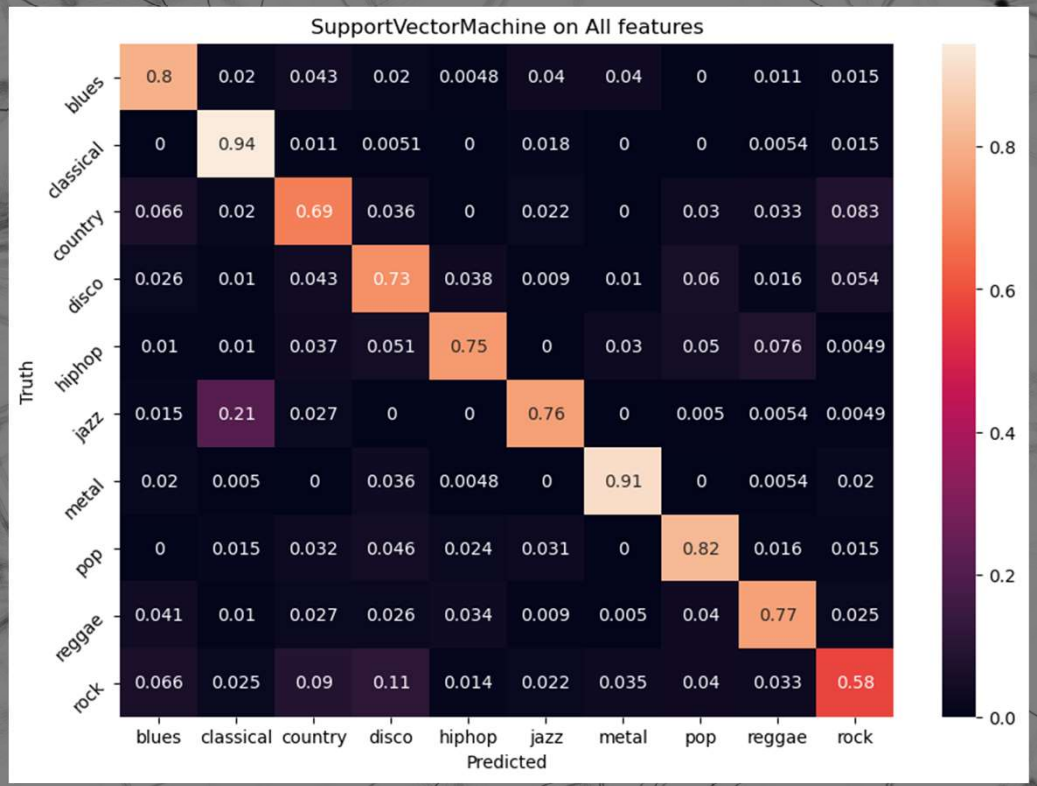
Best Parameters	Scaler	Kernel	C	Gamma	Accuracy
All features	MinMax	'poly'	1.0	'scale'	0.811
Chroma Frequencies	MinMax	'rbf'	1.0	'scale'	0.347
Spectral Centroid	Standard	'rbf'	1.0	'auto'	0.358
Zero Crossing Rate	Standard	'rbf'	1.0	'scale'	0.320
MFCC	Standard	'rbf'	1.0	'auto'	0.795
Chords	MinMax	'poly'	1.0	'scale'	0.524
All features except chords	Standard	'rbf'	1.0	'scale'	0.811

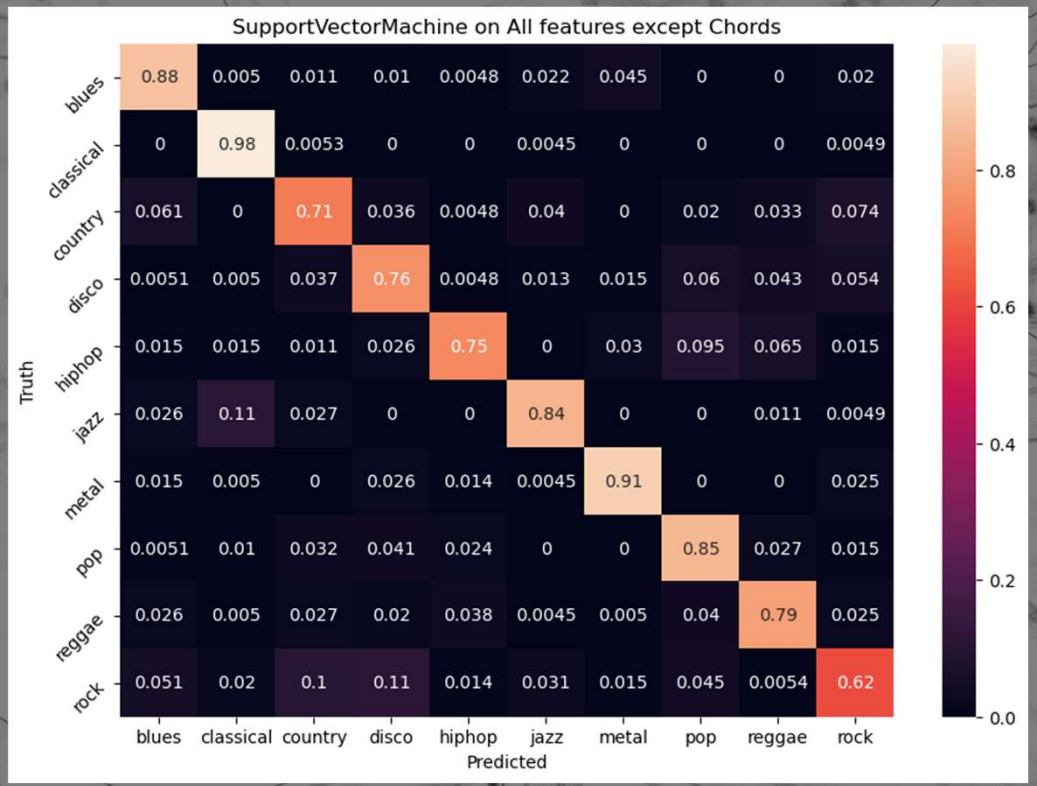
More Confusion Matrizes





More Confusion Matrizes





Code

- You can find the code at:

https://github.com/MrGandalv/MUGGE