Imagine this: You're a data scientist hired by one of the biggest music streaming platforms in the world. Your mission? To revolutionize how users discover new music by accurately predicting the genre of songs based on their audio features. But here's the catch – you'll need to battle it out with your fellow classmates to claim the coveted title of Kaggle Champion!

In this competition, you'll dive deep into the world of audio analysis, utilizing machine learning techniques to extract meaningful features from audio files. From Mel-Frequency Cepstral Coefficients to spectral contrast and beyond, you'll explore the rich tapestry of musical characteristics hidden within each track: from the soulful melodies of jazz to the electrifying beats of hip hop, each genre carries its unique signature, waiting to be decoded by your algorithms.

But it's not just about feature extraction – Imagine the thrill of deciphering the rhythmic patterns, harmonic structures, and tonal textures embedded within audio signals. Your goal is to design and test robust machine learning models capable of accurately classifying music tracks into their respective genres.

As you embark on this journey, remember that the stakes are high, but so are the rewards. Not only will you gain invaluable experience in real-world data science challenges, but you'll also have the chance to showcase your skills and creativity in front of your peers.

So, sharpen your coding skills, ignite your passion for data, and get ready to embark on an unforgettable adventure into the heart of music and machine learning. The stage is set, the competition awaits – may the best data scientist win![1]

Project Specifications

For this project you are asked to **implement logistic regression and gradient descent from scratch**, while utilizing existing libraries for feature extraction, dimensionality reduction, and other machine learning models like SVM, Gaussian Naive Bayes, and Random Forests.

[1] Intro created with ChatGPT

Feature Extraction:

- Extract relevant features from each audio file (choose your own and justify your selection). Common features for audio classification include:
    - Short-Time Fourier Transform (STFT)
    - Mel-Frequency Cepstral Coefficients (MFCC)
    - Chroma feature
    - Spectral Contrast
    - Temporal features (zero-crossing rate, energy)
- Choose a suitable frame size and hop size for feature extraction.
- Store the extracted features along with their corresponding labels (genre) in a dataset.

Dimensionality Reduction:

- If the feature space is high-dimensional, apply Principal Component Analysis (PCA). Choose the number of components/factors based on explained variance or cross-validation.

ML models:

- Train your logistic regression model on the training set. Tune hyperparameters using techniques like grid search or random search.
- Additionally, compare and contrast the LR accuracy with other classical machine learning algorithms. Include:
    - Random Forest
    - Gaussian Naive Bayes
    - Gradient Boosting Machines
    - Support Vector Machines (SVM)

Evaluation:

- Evaluate the trained model on the testing set using metrics such as accuracy, precision, recall, F1-score. Provide confusion matrices as needed
- Cross-validation can also be applied for robust evaluation.

Analysis:

- Analyze misclassified samples to understand where the model is struggling.
- Consider re-evaluating feature selection, feature extraction parameters, or trying more advanced techniques if performance is not satisfactory.

Feature Extraction:

Mel-Frequency Cepstral Coefficients (MFCC): MFCCs are a widely used feature representation for audio signals, especially in speech and music processing. They capture the spectral characteristics of the audio signal by extracting the envelope of its short-term power spectrum. MFCCs are often used for tasks like speech recognition, speaker identification, and music genre classification. For example, you can extract mfccs with the mfcc() function, which contains 13 coefficients (the default value for the nceps parameter mfcc(input, nwin=256, nfft=512, fs=16000, nceps=13), varying this parameter will give you more or less coefficients). Lets say that your song has 4135 frames, you can do an average per coefficient over all the frames (i.e., your classifier will use only 13 features per song).

Chroma Features represent the energy distribution of different pitch classes (e.g., notes in the musical scale) in an audio signal. They are useful for capturing tonal content and harmonic characteristics of music. Chroma features are commonly used in tasks such as music genre classification, chord recognition, and melody extraction.

The Short-Time Fourier Transform (STFT) is a widely used technique in signal processing for analyzing non-stationary signals, such as audio. It represents how the frequency content of a signal changes over time by computing the Fourier transform of short, overlapping segments of the signal. For example, you could use the 1000 first FFT components as features, or you could use the first 200 and the last 200, or any variation of that.

Spectral Contrast measures the difference in energy between peaks and valleys in the spectrum of an audio signal. It provides information about the spectral texture and timbral characteristics of the sound. Spectral contrast features are useful for tasks like audio segmentation, instrument classification, and sound texture analysis.

Zero-Crossing Rate (ZCR) measures the rate at which the audio signal changes its sign (i.e., crosses the zero amplitude level). It reflects the amount of high-frequency content or the presence of abrupt changes in the signal. ZCR is

often used for tasks like speech/music discrimination, audio onset detection, and sound event classification.

To use these features for analysis, you can follow these general steps:

1. Use a suitable library to read .au files and load them into memory.
2. Optionally preprocess the audio signals (e.g., resampling, normalization, noise reduction) to enhance feature extraction.
3. Use the chosen library (e.g., Librosa, Essentia) to extract relevant features from the audio signals. Select features that are most appropriate for your analysis task.
4. Organize the extracted features into a suitable representation (e.g., feature vectors per file) for further analysis.

There are several Python libraries available for processing au files. Some of the popular ones include:

- Librosa is a Python package for music and audio analysis. It provides tools for feature extraction, visualization, and analysis of audio signals. Librosa supports a wide range of audio formats, including .au files. It offers functions for extracting various features such as Mel-Frequency Cepstral Coefficients (MFCC), chroma features, spectral contrast, and many more.
- PyDub is a simple and easy-to-use Python library for audio manipulation. It allows you to work with audio files in various formats, including .au files. While it's primarily focused on audio editing and processing tasks, it can also be used to extract basic features from audio files.
- SciPy is a scientific computing library in Python that provides various functions for signal processing and analysis. It includes modules for reading and writing audio files, as well as functions for basic audio feature extraction such as zero-crossing rate, spectral centroid, and spectral bandwidth.
- Essentia is an open-source library for audio and music analysis. It offers a wide range of algorithms for feature extraction, audio processing, and music information retrieval tasks. Essentia supports multiple audio formats, including .au files, and provides functions for extracting features like MFCC, spectral features, rhythm patterns, and more.

Rubric

Deliverables (10 points)

- Submit 2 separate files in canvas: a pdf report and an archived file of your code and README
- (2 point) Report must be PDF. Report will not be graded otherwise.
- (2 point) Code packaged in archive (tar or zip); no github links, do not include the dataset
    - Include README; at a minimum, describe how to run code and where to specify path to the dataset.
    - (2 points) (File manifest :: describe each file (one sentence, must be more descriptive than “this file contains all code”. Even if one file does hold everything, describe what’s in there.))
    - (2 points) Team names; contributions from each member
    - (2 points) Final Kaggle score, accuracy, and date run

Implementation of Logistic Regression and Gradient Descent (20 points)

- Logistic Regression Implementation (10 points)
    - Correct implementation of logistic regression algorithm from scratch.
    - Handling of multi-class classification tasks.
    - Correct computation of logistic loss function and gradient descent updates.
- Gradient Descent Implementation (10 points)
    - Implementation of gradient descent algorithm from scratch.
    - Correct handling of learning rate and convergence criteria.
    - Proper vectorization for efficient computation (optional).

Feature Extraction and Preprocessing (20 points)

- Feature Extraction (10 points)
    - Effective utilization of existing libraries (e.g., Librosa, Essentia) for feature extraction.
    - Selection of relevant features for audio classification tasks (e.g., MFCC, chroma, spectral features).
- Preprocessing (10 points)
    - Proper preprocessing techniques applied (e.g., normalization, resampling, noise reduction).

        - Clear documentation of preprocessing steps and rationale.

Comparison with other classical classifiers (20 points)

- Implementation and utilization of SVM, Gaussian Naive Bayes, and Random Forests from existing libraries (10 points)
- Proper utilization of libraries for model training, validation, and evaluation (5 points)
- Correct handling of hyperparameter tuning and cross-validation (5 points)

Evaluation and Performance (10 points)

- Performance Metrics (5 points)
    - Calculation and reporting of appropriate evaluation metrics (e.g., accuracy, time to convergence)
    - Clear interpretation of performance results.
    - Clear description of the effects of learning rate on performance
- Model Comparison (5 points)
    - Comparative analysis of different models in terms of performance and computational efficiency.
    - Insightful discussion on the strengths and weaknesses of each model.

Documentation and Code Quality (10 points)

- Documentation (5 points)
    - Clear documentation of code, including inline comments and docstrings.
    - Readable and well-organized code structure.
- Code Quality (5 points)
    - Efficient and optimized code implementation.
    - Proper error handling and defensive programming techniques.

Report quality (10 points)

- Clear and well-structured sections of project methodology, results, and conclusions.
- Effective communication of technical concepts.
- Use of figures/captions/tables and visual aids.
- Demonstrated understanding of the project's technical aspects.