Building a Classification and Recommendation Model for Music Genres

(A machine learning approach)

2023A2PS0296P	Rudra Narayan Bhatt
2023A2PS0235P	Smita Prajna Prusty
2023A8PS0709P	Arpit Kumar Khatri
2023A2PS0232P	Agrim Gupta
2023A2PS1301P	Pinak Sharma

Topic Number: 25

Group Number: 8

Business Objective:

To develop an accurate music genre classification system that can automatically categorize songs into predefined genres (e.g., Rock, Pop, Blues, etc.). These algorithms can benefit streaming platforms (e.g., Spotify, Apple Music) for personalized recommendations, and playlist generation.

Evaluation Criteria (Business Metric to Measure)

- **Accuracy**: Percentage of correctly classified tracks.
- **F1-score**: To account for class imbalance across genres (which is not considered by accuracy)
- **Confusion Matrix**: For detailed error analysis across genres.
- **Training Time** (technical metric, but also a very important business metric because it directly affects costs, speed to market, and scalability).

Data Science Problem Statement:

Given a dataset of audio tracks with extracted features (spectral, temporal, and MFCCs), build a sufficiently capable machine learning model to classify each track into its correct genre and lastly evaluate performance trade-offs between traditional ML and deep learning approaches (like CNNs)

Data Sets under Consideration

- GTZAN Dataset (widely used for music genre classification, contains 10 genres, 1000 audio tracks, 30seconds of playback time for each).
  Link: GTZAN Music Genre Dataset

Summary of approach (with reasons for each) and findings from related research papers:

Considering that the dataset is a fairly high-dimensional one, we would try to learn a basic naïve bayes model on the dataset and use the performance indicators to measure the naïve bayes model performance. We do this to prove that the feature-space in this dataset (like MFCCs- Mel Frequency Cepstral Coefficients) is fairly correlated and therefore training a model on the data that assumes class-conditional independence could produce suboptimal results. Therefore, this could be a good baseline to begin.[1]

The next model that we could test for better results would be the KNN (K-Nearest Neighbours) algorithm while maintaining the increasing order of complexity. It doesn't require parameter training — it just stores the dataset and classifies new inputs based on distance to neighbours. However, KNN comes with extreme compute overhead, because we have to find the Euclidean distances from every training data point, further sort the distances in increasing order and

take the K-smallest values to arrive at a conclusion using majority-voting and this is to be performed for every point in the testing point, making it a sufficiently interpretable model but with very limited scalability due to the heavy computation.[2]

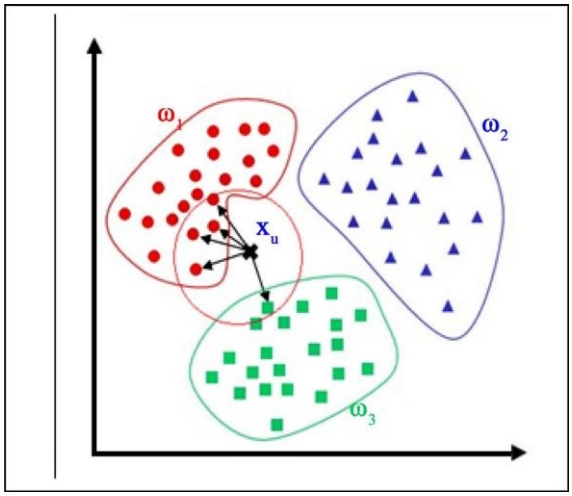

fig 1 (X_u represents the test point, please note that the training dataset is not divided into clusters like in the diagram)

The next model that we would like to evaluate would be the Random Forest technique which is a widely used Ensemble method with all of its base-classifiers being decision trees. Here each tree is built on a random bootstrapped sample (with replacement) from the dataset. We use a collection of trees and majority voting to predict our output instead of a single decision tree because decision trees are meant to overfit training data, hence we use the output from multiple trees each trained to predict the same output but with different data and then select the most likely class by taking a majority vote from the predictions of all the trees. This allows the tree to **approximate very complex decision boundaries**. Instead of a single straight line (like in linear regression or logistic regression), the tree creates a **piecewise-constant function** that can follow curved or irregular shapes.[3][4]

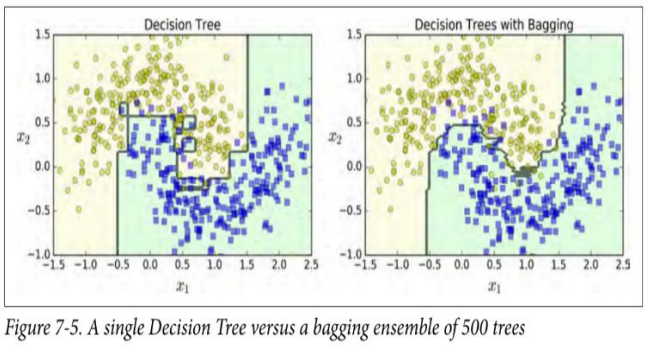

Figure 7-5. A single Decision Tree versus a bagging ensemble of 500 trees

Fig 2(shows the decision boundary of a simple decision tree vs the boundary for a random forest on the same dataset)

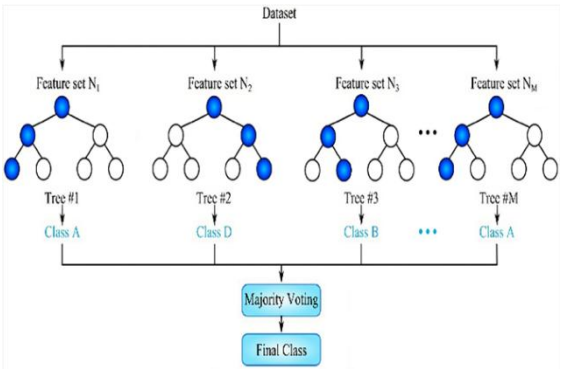

Fig 3 (visualisation of a random forest with its base classifiers)

Finally, the best approach for this problem would be to use a deep learning approach, called Convolutional Neural Networks (CNN). However, in this case the training data would not be a collection of n-dimensional points, rather it would be in the form of spectrograms (i.e., a time vs

frequency representation of sound as an image). This could be a very capable approach because CNNs are excellent at detecting spatial patterns in images. There are certain patterns like repeated rhythms, harmonic lines/structures, timbre signatures which could be very hard to hand engineer into features without noise, while CNNs learn them naturally. A CNN has multiple layers, in our case, the lower layers would detect the simple and local features from the raw data while the higher layers would detect the abstract and genre-specific features and focus on features that best separate the classes. In doing so, CNNs consistently outperform traditional ML algorithms discussed before owing to the avoidance of heavy manual feature engineering.[5] [6]

Key Innovations:

We have tried to improve the traditional ML methods (KNN and Random Forest) by adding additional preprocessing steps to optimise the ML methods. However, this was done purely from an experimental point of view and could eventually result in loss of accuracy, however yield a proven decrease in training/testing time of the model. Like for KNN, we aim to store the data points not just as discrete points in a hyperspace but as clusters, preferably 10 (since there are 10 genres in total). Additionally , when a test point comes , we would assign it to the nearest cluster (using the proximity from the cluster's centroid) and then train the KNN algorithm for the points within that cluster alone .This could reduce the computation significantly as the clusters are only trained once , however the clusters might not be pure , since the final result in K-means clustering is subject to the randomness in the initial choice of centroids.

Next as a part of preprocessing for the random forest method, we would add an extra layer of optimization to train the base classifier decision trees using a technique called as the **Sparrow Search Algorithm** to train the base classifiers faster and with more robustness.

Stretch Objectives:

-We might try and train a **Support Vector Machine (SVM)** algorithm additionally just to test owing to its good generalisation capabilities.

-We also aim at building an additional recommendation engine to recommend music from specific genres using **SVD** (singular value decomposition) or **ALS** (alternating least squares).

Tentative Timeline & Milestones:

Phase 1 (Week 1): Data Exploration & Preprocessing

• Collect and verify GTZAN dataset

• Extract features (MFCCs, spectral, temporal features)

• Clean, normalize, and prepare dataset

• Deliverable: Feature-ready dataset

Phase 2 (Week 2): Baseline Models

• Train & evaluate Naïve Bayes

• Train & evaluate KNN

• Measure performance (Accuracy, F1-score, Confusion Matrix)

• Deliverable: Baseline benchmark report

Phase 3 (Week 3): Ensemble Methods

• Train & tune Random Forest

• Experiment with Sparrow Search optimization for base trees

• Compare results vs. baseline

• Deliverable: Ensemble performance report

Phase 4 (Week 4): Deep Learning (CNN)

• Convert audio → spectrograms

• Build & train CNN model

• Evaluate results vs. ML models

• Deliverable: CNN model performance report

Phase 5 (Week 5): Optimization & Stretch Goals

• Try SVM for generalization

• Add preprocessing improvements (KNN clustering, etc.)

• Start recommendation engine (SVD/ALS) prototype

• Deliverable: Extended experimental results

Phase 6 (Week 6): Final Evaluation & Reporting

• Compare traditional ML vs. Deep Learning trade-offs

• Consolidate results (Accuracy, F1, Confusion Matrix, Training Time)

• Prepare final documentation & presentation

• Deliverable: Final Report + Demo

END