A

SYNOPSIS

of

MINOR PROJECT

on

Music Genre Classification Using Machine Learning

Submitted by

Divyanshi Shrimali (22EGICA008)

Project Guide
Mr. Ritesh K. Jain

Head of Department
Dr. Mayank Patel

Geetanjali Institute of Technical Studies, Dabok , Udaipur (Raj.)
Department of Computer Science and Engineering
October,2023

Problem Statement: Music Genre Classification

With the oceans of music and audio content available in this digital age, efficient methods for organization are imperative. Audio genre classification is as task of automatically labelling audio recordings into distinct genres, sociably shown to be related with their acoustic features in an embodiment.

Brief Description:

It is a machine learning model built for classifying music genres based upon its audio features. This starts with the pre-processing of a dataset for music tracks, after which it extracts pertinent audio features and normalizes data to make them on one scale. The project uses Random Forest model to train data for classification. To maximise the chance of getting a good model and test it using Grid Search, used for hyperparameter tuning as well as validation of models. For model interpretation visualizations like PCA plots are used and Confusion Matrix for checking performance of a trained Models. The Model explainers will give features impacts in predicting target variable. The goal of the project is to provide a solid taxonomy allowing music recommendation algorithms or streaming platforms categorize music with more accuracy.

Objective and Scope:

Objectives:

Collecting Data Idea and Preparation: Collection of a varied dataset that includes more than 1-GB audio read-auto recordings in multiple genres including enough variability between the artists, styles along with production quality.

Feature extraction: These features will include spectral characteristics (such as timbre, brightness or tempo), rhythmic patterns and harmonic content that numerically represent the audio track.

Model Development: Using Machine Learning algorithm, Random Forest and tuning to learn from extracted features with a hope for better predictions on upcoming audio tracks.

Performance evaluation: Assess the performance of model using accuracy, precision, recall, F1 score. The face is robust and can generalize confusion matrix analysis.

Visualization and Interpretation: Use plots like confusion matrices, PCA visualizations to visualize the results which helps in interpretation of classification decisions made by the model thereby giving tips on where algorithms need improving.

Scope:

The goal for this project was a working music genre classifier based on machine learning. This includes collating and pre-processing a varied music track dataset, extracting relevant audio features, and improvements to data quality accompanied by cleaning and standardisation. The project will see implementation of Random Forest for classification along with optimization models by using Grid Search cross-validation. This has several evaluation metrics to assess model performance. They are useful to analyze classification results and feature importances using visualizations such as confusion matrices and PCA plots. The purpose of this project is to deploy a robust model that accurately classifies music genres for use in improving music recommendation systems or online streaming services. Heavy documentation provides transparency and contributes to advancing music genre classification research and applications.

Methodology:

1. Loading and Inspecting Data:

- **Libraries Used:** Pandas for data manipulation and management.
- Data is loaded from a CSV file ('features_3_sec.csv') using pd.read_csv().
- Initial exploration includes displaying the first few and last few rows, checking the dataset's shape, column names, and information summary (data.head(), data.tail(), data.shape, data.columns.values, data.info()).

2. Handling Missing Values:

- **Libraries Used:** NumPy for numerical operations.
- Infinite values (np.inf, -np.inf) are replaced with NaN using data.replace([np.inf, -np.inf], np.nan) and rows containing NaN values are dropped with data.dropna().

3. Feature Extraction and Preparation:

- Non-feature columns ('filename', 'length', 'label') are dropped from the dataset to isolate the features for modeling.
- Features are separated into input variables (X) and target variable (y).

4. **Exploratory Data Analysis (EDA):**

- **Libraries Used:** Matplotlib for data visualization.
- The distribution of genres in the dataset is visualized using plt.figure(figsize=(10, 6)) and data['label'].value_counts(normalize=True).plot.barh() to understand the class balance.

5. **Feature Scaling:**

- **Libraries Used:** Scikit-learn for data preprocessing.
- Standard scaling is applied to normalize the features (X_scaled = scaler.fit_transform(X)) using StandardScaler().

6. **Principal Component Analysis (PCA):**

- **Libraries Used:** Scikit-learn for dimensionality reduction.
- PCA is performed on the scaled features (X_scaled) with PCA(n_components=2) to reduce dimensionality to two principal components (X_pca = pca.fit_transform(X_scaled)).

7. **Model Training and Evaluation (Random Forest Classifier):**

- **Libraries Used:** Scikit-learn for model building, evaluation, and optimization.
- Data is split into training and testing sets (train_test_split), and a Random Forest Classifier is initialized and trained (RandomForestClassifier(random_state=42)).

8. **Model Evaluation:**

- **Libraries Used:** Scikit-learn for model evaluation metrics.
- Predictions are made on the test data (y_pred = model.predict(X_test)), and performance metrics such as accuracy, precision, recall, and F1-score are computed.
- A classification report and confusion matrix are generated to evaluate model performance.

9. **Hyperparameter Tuning:**

- **Libraries Used:** Scikit-learn (GridSearchCV) for hyperparameter optimization.

- Grid Search Cross-Validation is performed to find the best combination of hyperparameters for the Random Forest Classifier.

10. **Best Model Evaluation:**

- Using the best parameters obtained from Grid Search, the model is re-evaluated on the test set to assess its improved performance.
- The evaluation metrics (accuracy, precision, recall, F1-score) for the optimized model are calculated and displayed.

11. **Final Model Assessment:**

- A final classification report and confusion matrix are generated for the optimized model to provide a comprehensive overview of its performance.

This methodology combines data preprocessing, dimensionality reduction, model training, evaluation, and optimization using popular Python libraries such as Pandas, NumPy, Matplotlib, and Scikit-learn, tailored specifically for music genre classification. Each step leverages these libraries to ensure efficient data handling, insightful analysis, and robust model development.

Hardware and Software Requirements:

Hardware Requirements:

1. **CPU:** Multi-core processor for efficient data processing.
2. **RAM:** Minimum 8 GB (16 GB recommended) for handling datasets and computations.
3. **Storage:** SSD preferred for faster data access and sufficient disk space for datasets and models.

Software Requirements:

1. **Operating System:** Windows for compatibility with data science tools.

2. **Python and Libraries:**

   - Python 3.6 or higher.
   - Essential libraries: pandas, NumPy, scikit-learn, matplotlib, seaborn.

3. **Development Environment:** IDEs like Jupyter Notebook for coding and experimentation.
4. **Documentation:** Jupyter Notebooks or Markdown for project documentation and reporting.

Technologies:

- **Python:** Primary programming language for data preprocessing, feature extraction, model development, and evaluation.

- **Libraries:** Utilized libraries such as pandas, NumPy, scikit-learn, matplotlib, seaborn for data manipulation, machine learning tasks, and visualization.

- **Machine Learning Algorithms:** RandomForestClassifier for building and optimizing the classification model.

- **PCA (Principal Component Analysis):** Used for dimensionality reduction and visualization of high-dimensional data.

- **GridSearchCV:** Employed for hyperparameter tuning of the RandomForestClassifier model.

- **Jupyter Notebooks:** Environment for interactive development, code execution, and documentation.

Testing Techniques:

- **Train-Test Split:** Data is split into training (X_train, y_train) and test sets (X_test, y_test) using Sklearn's train_test_split to evaluate model performance on unseen data.

- **Hyperparameter Tuning with GridSearch and Cross-Validation:** The code uses GridSearchCV to perform a systematic search over specified hyperparameter values for the RandomForestClassifier, utilizing 5-fold cross-validation to evaluate model performance and select the optimal hyperparameters.

- **Evaluation Metrics:** Metrics such as accuracy, precision, recall, F1-score, and confusion matrix are used to evaluate model performance. These metrics provide insights into how

well the model predicts different classes and its overall effectiveness.

Project Contribution:

This project contributes to the field of music data analysis by automating the process of music genre classification using machine learning techniques. The key contributions include:

1. **Automated Genre Classification**: The project reduces the need for manual labeling of music tracks by accurately categorizing them into genres, which can be subjective and time-consuming.
2. **Enhanced Music Recommendation Systems**: By providing reliable genre classifications, the project improves the performance of music recommendation systems, offering users better-personalized experiences.
3. **Comprehensive Methodology**: Demonstrates an effective methodology for handling and preprocessing music datasets, including feature extraction, scaling, and dimensionality reduction, which can be applied to similar projects.
4. **Model Optimization**: Utilizes techniques like GridSearchCV for hyperparameter tuning to achieve optimal performance, showcasing best practices in model evaluation and optimization.
5. **Insightful Visualizations**: Offers visual representations of data distributions, feature correlations, and model performance, aiding in better understanding and communication of the results.

Contributors

- **Divyanshi Shrimali:** Lead Developer and Data Science Enthusiast, responsible for data preprocessing, model training, and evaluation.
- **Mr. Ritesh K. Jain:** Provided expertise in machine learning techniques and model optimization.
- **Dr. Mayank Patel:** Provided strategic guidance and facilitated access to essential resources and datasets.

Tools Used

- **Kaggle:** For dataset access and initial data exploration.
- **GitHub:** For version control and project collaboration.