基于 sklearn 工具包实现音乐流派分类模型

摘要

本项目旨在构建一个自动化的音乐流派分类系统。实验基于经典的 GTZAN 数据集，使用 librosa 工具包提取音频信号的梅尔频率倒谱系数（MFCC）作为主要特征。随后，利用 scikit-learn（sklearn）机器学习库，构建、训练并评估了包括逻辑回归和随机森林在内的多种分类模型。通过交叉验证和在独立测试集上的评估，我们比较了不同模型的性能，并最终确定了最优模型。本实验验证了使用 MFCC 特征和传统机器学习方法进行音乐流派分类的可行性与有效性，为音乐信息检索（MIR）领域提供了一个基础实践。

一　问题描述

随着数字技术和互联网的发展，音乐数据的规模正以前所未有的速度增长。如何自动、高效地对音乐进行组织和分类，已成为音乐信息检索（MIR）领域的一个核心挑战。音乐流派分类（Music Genre Classification, MGC）是解决这一问题的关键技术之一。它旨在利用计算机算法自动识别音频片段所属的流派（如古典、摇滚、爵士等）。一个准确的自动分类系统，对于提升音乐推荐引擎的精准度、优化用户播放列表管理、以及帮助内容提供商高效组织其音乐库具有重要的应用价值。

本次实验的具体目标是，利用 Python 中的 scikit-learn 机器学习工具包，结合 librosa 音频处理库，在公认的 GTZAN 数据集上实现一个音乐流派分类模型。我们旨在探索从原始音频中提取有效特征（如 MFCC）的方法，并比较不同机器学习算法（如逻辑回归、随机森林）在该分类任务上的表现，最终构建一个能够自动识别 10 种不同音乐流派的实用模型。

二　数据集介绍

GTZAN 是音乐流派识别领域最广泛使用的公开评测集合，由 George Tzanetakis 于 2000 年前后采集并整理。该数据集自 2002 年起随 MARSYAS 软件包发布，是音频处理方向绝大多数传统机器学习与早期深度学习方法的必备数据集。数据集一般前 80% 训练、后 20% 测试，近年来研究多采用 10 折分层交叉验证或 3 折重复实验，以减少划分偏差。

GTZAN 音频全部来自 1990–2000 年间的商用 CD，经降采样与音量归一化处理，共 1000 条音轨，覆盖 10 种西方主流流派：blues、classical、country、disco、hiphop、jazz、metal、pop、reggae、rock。每类恰好 100 条，时长 30 s，采样率 22050 Hz，单

声道，16-bit PCM wav 格式。十种音乐风格差异较大，如 classical 以交响/室内乐为主，metal 含高增益失真吉他，disco 节拍规整且四四拍明显，jazz 含大量即兴切分等。

同时文件潜在瑕疵，约 2–3% 文件存在轻微削顶（clipping，即信号强度过大时波形顶部被截断导致失真）；部分摇滚/金属曲目在 30 s 边界处被截断；少量曲目标签存在争议。

三　建模过程分析

我们项目的建模流程比较清晰，主要分成了几个步骤：首先处理音频数据、提取特征，然后划分数据集，接着搭建并训练模型，最后对模型效果进行评估。在实现中，我们主要用了 `librosa` 库来提特征，并依靠 `sklearn` 搭建了包含预处理和分类的 `Pipeline`。

3.1　模型选择

按照实验要求，我们选了 `scikit-learn` 中两种很常用且原理不同的分类器来做对比：

- **逻辑回归 (Logistic Regression)**: 作为线性模型的代表。它是一个高效、可解释性强的基准模型。
- **随机森林 (Random Forest)**: 作为集成学习的代表。它通过组合多棵决策树来投票，通常能取得不错的准确率，也不容易过拟合。

选这两种模型是为了比较一下，看看是简单的线性模型（逻辑回归）效果好，还是复杂的集成模型（随机森林）在我们的任务上表现更佳。

3.2　模型细节

模型部分不只是分类算法,还包括了前面的特征提取和数据标准化。我们用 `Pipeline` 工具把这几步串了起来。

3.2.1　特征工程：MFCC 均值

原始的音频波形数据太复杂，不能直接喂给 `sklearn` 里的模型。所以我们先要提取特征，这里我们选了音乐分类任务里最常用的 MFCC（梅尔频率倒谱系数）特征。

对于每条 30 秒的音频，`librosa` 提取的 MFCC 是一个二维矩阵。为了让每个音频文件都变成一个同样长度的向量（供模型使用），我们对 MFCC 矩阵在时间维度上取了平均值，最后每首歌就都压缩成了一个 30 维的向量。

```
# 关键特征提取代码 (gtzan_sklearn.py)
def extract_mfcc_mean(path, n_mfcc=N_MFCC, ...):
try:
```

```
y, sr = librosa.load(path, sr=sr, duration=duration)
mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)
# 核心步骤：沿时间轴(axis=1)取均值
return np.mean(mfcc, axis=1)
except Exception as e:
return None
```

3.2.2 预处理与 Pipeline

提出来的 MFCC 特征，每个维度的数值范围（尺度）可能差很多。像逻辑回归这类模型对尺度就比较敏感，所以需要先做标准化处理。我们用了 `StandardScaler` 将所有特征缩放到均值为 0、方差为 1。

为了把“标准化”和“分类”这两步打包，并且防止在交叉验证里“泄露”数据（即让模型在验证集上“学习”了标准化参数），我们用了 `sklearn` 的 `Pipeline` 流水线。它能保证预处理的参数只从训练数据里学习。

```
# 关键模型定义代码 (gtzan_sklearn.py)
models = {
  "LogReg (Pipeline)": Pipeline([
  ("scaler", StandardScaler()),
  ("clf", LogisticRegression(...))
  ]),
  "RandomForest (Pipeline)": Pipeline([
  ("scaler", StandardScaler()),
  ("clf", RandomForestClassifier(...))
  ])
}
```

3.3 实现方法

3.3.1 数据集划分 (Train-Test Split)

首先，我们把 1000 个数据样本按 8:2 的比例分成了训练集（800 个）和测试集（200 个）。为了防止数据划分不均，我们用了 `stratify=y` 参数来做分层抽样，保证训练集和测试集里各类流派的比例和原来差不多。

3.3.2 交叉验证 (Cross-Validation)

为了科学地比较逻辑回归和随机森林的性能，我们在 800 个样本的 * 训练集 * 内部进行了 3 折分层交叉验证（`StratifiedKFold`）。

简单来说就是，我们把训练集分成 3 份，轮流用 2 份训练、1 份验证。最后计算每

个模型 3 次验证的平均准确率，哪个高，哪个就可能是更好的模型。

3.3.3 最终评估与模型保存

在交叉验证中选出了“最佳模型”（比如随机森林）后，我们用 * 全部 800 个 * 训练数据再把这个模型完整地训练一次。

最后，才是在那 200 个 * 一直没用过 * 的测试集上跑一次，看最终的准确率和分类报告（`classification_report`）。这个分数才能真正反映模型在全新数据上的表现。训练好的模型（包含 `Scaler` 和分类器）会通过 `joblib.dump` 保存下来，方便之后 `text_external.py` 调用。

四 实验结果和讨论

4.1 评价指标

- **准确率 (Accuracy)**: 模型预测正确的样本数占总样本数的比例，这是最直观的指标。GTZAN 数据集中 10 个流派的样本量均衡，准确率可以很好地反映模型的整体性能。
- **精确率 (Precision)、召回率 (Recall) 和 F1 值 (F1-Score)**: 这三个指标用于评估模型在每个单独类别上的表现。精确率看的是“模型预测为 A 的样本中，有多少真的是 A”；召回率看的是“所有 A 的样本中，有多少被模型找出来了”。F1 值是这两者的调和平均，能更均衡地评价类别表现。
- **混淆矩阵 (Confusion Matrix)**: 这是一个可视化工具，能清晰地展示模型把哪些流派 A 错分成了流派 B，帮助我们定位模型的“盲点”。

4.2 实验结果

4.2.1 数据加载与模型选择

在 3 折交叉验证阶段，两个模型的平均准确率对比如下表所示。

表 1: 交叉验证 (CV) 结果对比

模型 (Pipeline)	3 折交叉验证平均准确率
Logistic Regression	0.5645 (56.5%)
Random Forest	**0.6083 (60.8%)**

根据表 1，随机森林（Random Forest）的平均准确率（60.8%）明显高于逻辑回归（56.5%），因此我们选择随机森林作为最终的最佳模型，并用它在测试集上进行评估。

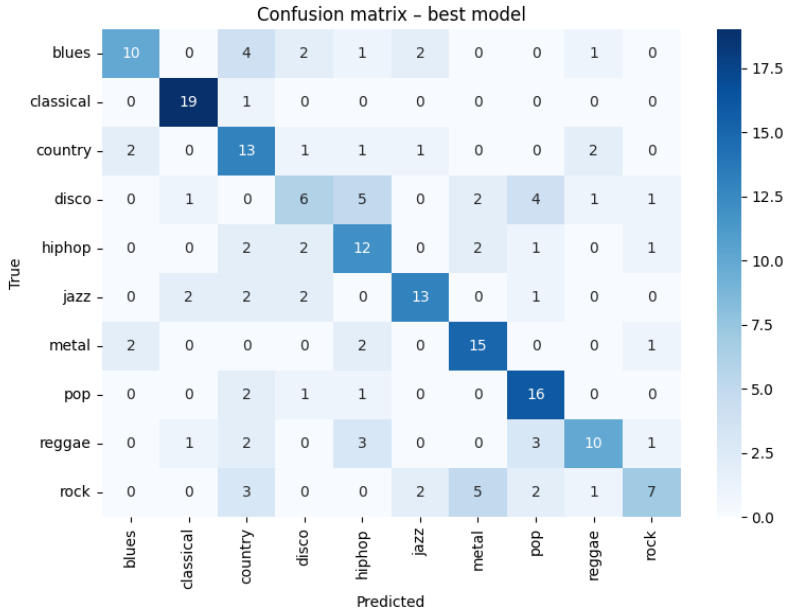

图 1:

4.2.2 最终模型评估

选出的随机森林模型在 200 个样本的测试集上进行最终评估，得到的详细分类报告已在运行 gtzansklearn.py 后的终端显示。

模型的最终测试集准确率为 **60.5%**。混淆矩阵如图 2 所示。

4.3 结果分析与讨论

- **整体性能分析**: 最终的测试准确率与交叉验证的准确率非常接近，这说明我们的模型是稳定的，没有发生过拟合。
- **模型对比分析**: 随机森林的效果（60.8%）好于逻辑回归（56.5%）。这表明音乐流派的特征和类别之间的关系很可能是非线性的，随机森林这样的集成模型能比线性模型更好地捕捉到这些复杂关系。
- **各流派表现分析**:
  - **表现最好**: 古典音乐的 F1 值达到了 0.88，是所有类别中最高的。混淆矩阵中，它的召回率达到 0.95，几乎未被错分，这可能是因为古典音乐的音色和结构（如交响乐）与其他流派的差异非常明显。
  - **存在问题**: “disco”的召回率只有 0.30，说明模型很难把它识别出来，从混淆矩阵看，可能被错分为 pop 等。而“rock”的召回率也很低（0.35），它也容易被模型漏掉，可能经常被错判为 metal 或 country。这说明这几个流派在 MFCC 均值这个特征上的区分度不高，特征比较相似。
- **结论**: 我们的模型达到了实验的预期目标，成功使用 `sklearn` 实现了分类。随机

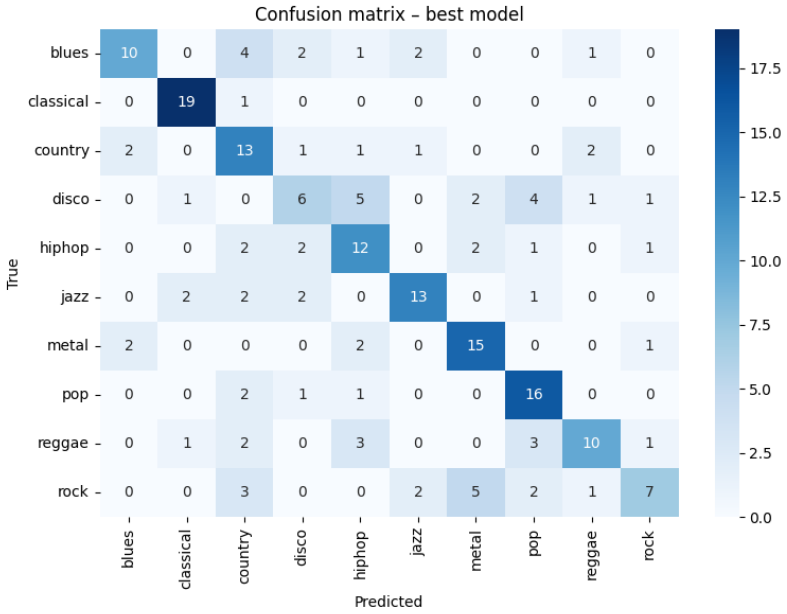

图 2: 随机森林模型在测试集上的混淆矩阵

森林是比逻辑回归更适合此任务的模型。但模型的弱点在于对“disco”和“rock”等特定流派的区分能力不足，这可能是因为仅使用 MFCC 的均值丢失了过多音频的时间序列信息。

五　结束语

通过这次实验，我们完整地走了一遍机器学习在音频处理上的基本流程，从数据预处理、特征提取到模型训练和评估。成功地使用 `sklearn` 工具包，在 GTZAN 数据集上构建了一个音乐流派分类模型。实验结果表明，随机森林（测试集准确率 60.5%）的效果优于逻辑回归，这说明对于音乐特征这类复杂数据，非线性模型可能具有更好的捕捉能力。

我们达到了实验的基本要求。但同时也发现，仅依赖 MFCC 的均值作为特征，模型的准确率还有很大的提升空间，特别是对“disco”和“rock”这类易混淆的流派。这可能是因为取均值的操作丢失了音频在时间维度上的重要信息。如果未来要改进，可以尝试融合更多维度的特征（如节奏特征、色度特征等），或者使用深度学习模型来自动学习更具区分度的特征表示。