**机器学习实验报告**

**实验名称：音乐流行度预测**

**学生姓名：陆文韬**

**学生学号：58122231**

**完成日期：2024/4/4**

**任务描述**

构建线性回归模型，对音乐流行度进行预测。

**数据集简介**

人类与歌曲和音乐有着紧密的联系。音乐可以改善情绪，减轻疼痛和焦虑，为情感表达提供机会。研究表明，音乐可以在许多方面有益于我们的身体和心理健康。最近，多项研究已经开展，以了解歌曲和其受欢迎程度之间的关系。本数据集中，歌曲用多种指标度量对进行描述并记录为表格，任务是预测歌曲流行度。数据集包含18835个样本，每个样本包含13个属性与1个实值标记，属性包括连续数据属性和离散属性，属性信息见下表。要注意数据之间存在多重共线性，这增加了任务的挑战难度。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变量名 | 属性/标记 | 连续/离散 | 类型 | 上下界 | 离散值个数 |
| Song\_name |  |  | string |  |  |
| Song\_popularity | Label | Continuous | int | (0,100) |  |
| Song\_duration\_ms | Attribute | Continuous | int | (12k,1.8m) |  |
| acousticness | Attribute | Continuous | double | (0,1) |  |
| daceability | Attribute | Continuous | double | (0,0.99) |  |
| energy | Attribute | Continuous | double | (0,1) |  |
| instrumentalness | Attribute | Continuous | double | (0,1) |  |
| key | Attribute | Discrete | int |  | 12 |
| liveness | Attribute | Continuous | double | (0.01,0.99) |  |
| loudness | Attribute | Continuous | double | (-38.8,1.58) |  |
| audiomode | Attribute | Discrete | int |  | 2 |
| speechiness | Attribute | Continuous | double | (0,0.94) |  |
| tempo | Attribute | Continuous | double | (0,242) |  |
| timesignature | Attribute | Discrete | int |  | 6 |
| audiovalence | Attribute | Continuous | double | (0,0.98) |  |

**实验要求**

1. 使用Python编程构建线性回归算法，预测音乐流行度。
2. 数据集预处理：
   1. 离散属性连续化
   2. 连续属性归一化normalization
   3. 共线性的检测与处理
3. 使用80%的数据集进行训练，20%的数据集进行测试。需要报告训练误差和测试误差。请注意，需要保证模型能够在测试集上进行准确的预测。
4. 使用可视化技术来展示你的模型和数据集。可以使用Python的Matplotlib库来完成该任务。
5. 报告模型的性能和准确率。可以使用Python的Scikit-Learn库中的mean\_squared\_error函数来计算均方误差（MSE）。
6. 提交代码与报告，其中报告内容包括以下几个部分：
   1. 预处理
   2. 实验设置，包括实验评估方法与性能度量
   3. 实验结果，包括模型性能，可视化结果展示
   4. 结果分析
   5. 代码附录

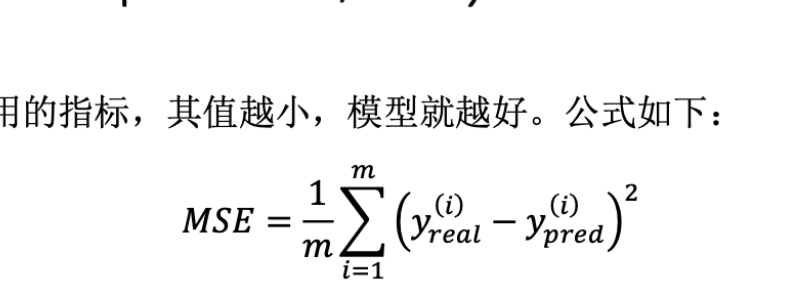
**实验设置**

模型评估

模型评估的方法有很多种，比如留出法、交叉验证法、自助法、留一验证法等方法。在本实验中，程序代码采用了较为简单的留出法。

采用留出法，将完整的数据集分割为互不包含的两部分，训练集占80%，测试集占20%，在模型训练的过程中，模型将训练集进一步分割产生交叉验证集，在训练集上训练得到参数之后在交叉验证集上进行测试得到训练误差，再在测试集上测试模型得到测试误差。交叉验证集有助于我们在模型训练的过程中选择更好的模型，而测试集则更接近于真实情况，测试误差更能反映出模型的泛化能力的高低，体现模型的真正水平。

性能度量

均方误差（Mean Square Error，MSE），均方误差是最常见的用于进行模型性能度量的测试指标，模型测试得到的均方误差越小，模型拟合的效果就越好。

过拟合和欠拟合

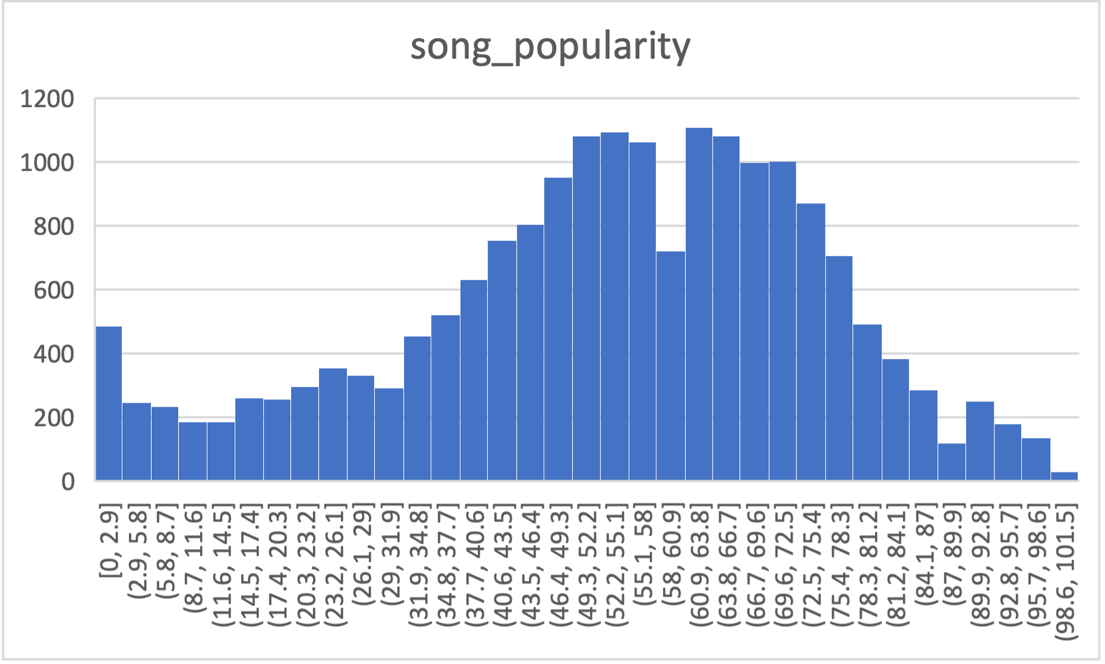
在模型训练的过程中，训练误差在梯度下降的过程中不断下降，但是测试误差却会先下降再上升，当训练误差和测试误差都很高时，说明模型处于欠拟合状态，参数还未下降到合适的范围内。当训练误差很低，但是测试误差很高的时候，说明模型发生了一定程度的过拟合，这个时候模型过分拟合了训练集的数据，但是泛化能力却发生了下降，应对未知样本的能力下降，合适的模型应该有着较低的训练误差和测试误差，并且两者较为接近。

**实验过程**

数据分析

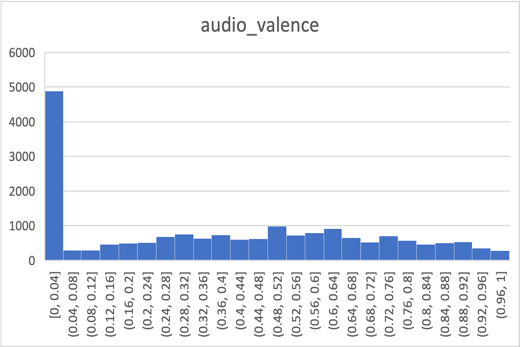
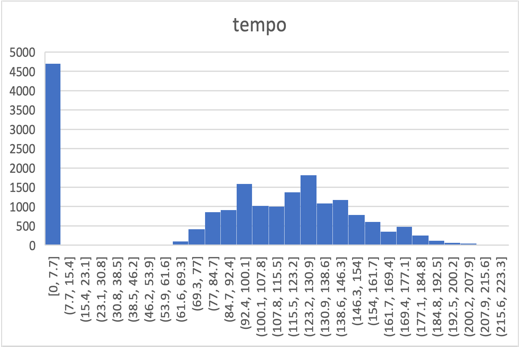
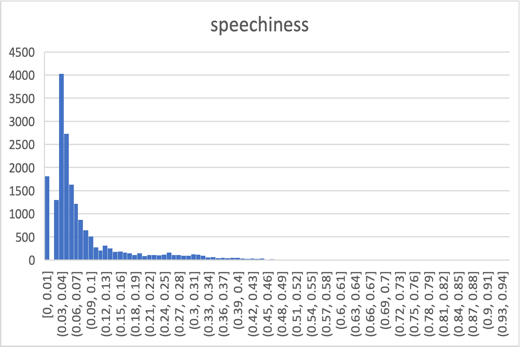
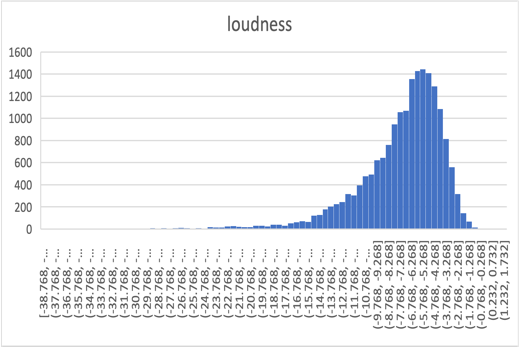
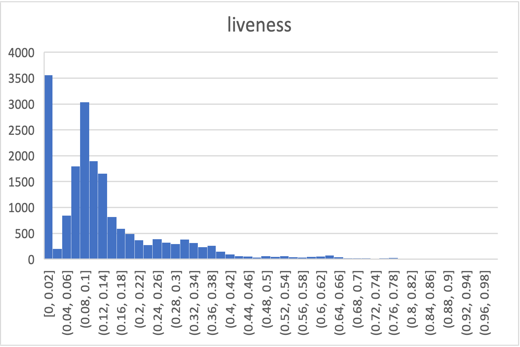
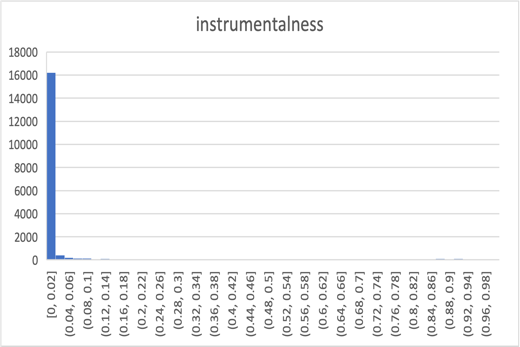
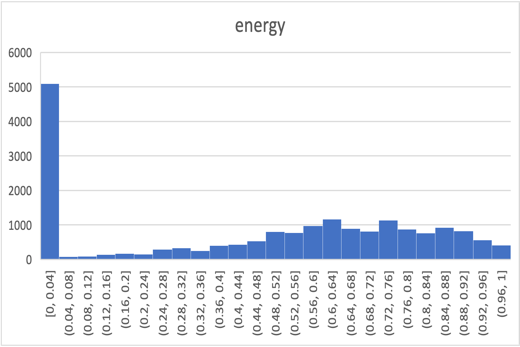
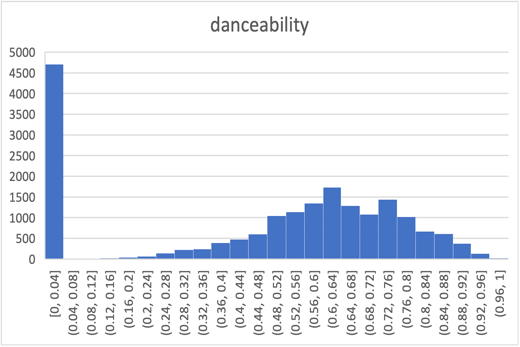
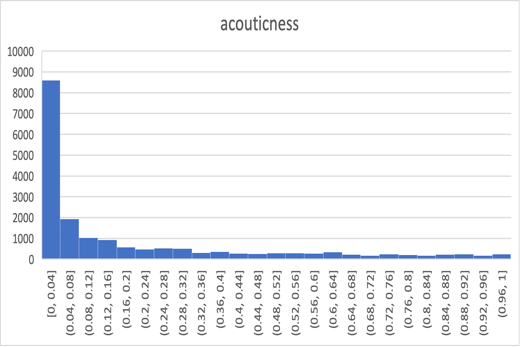
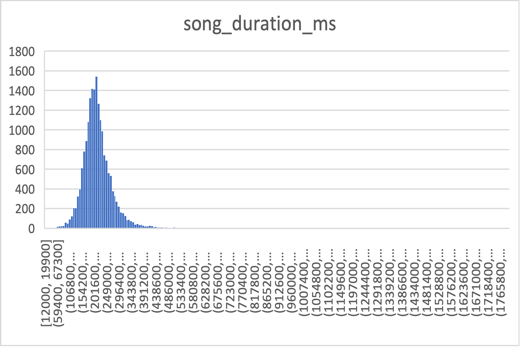
目标属性分析

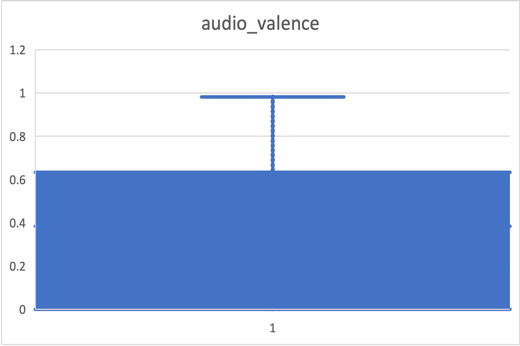
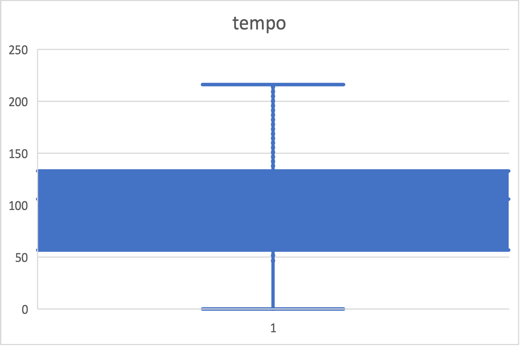
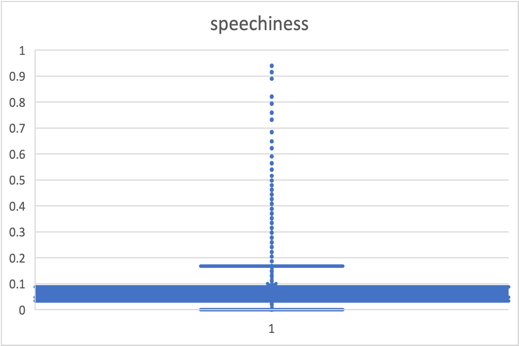
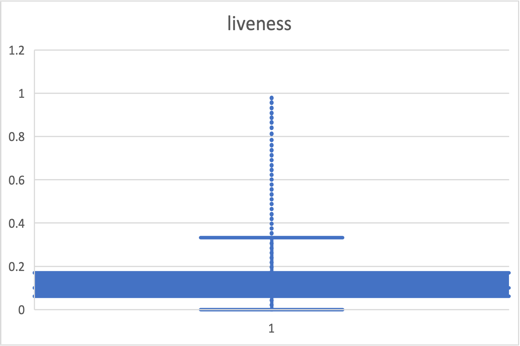
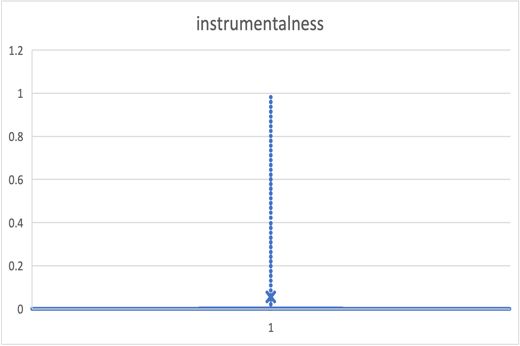
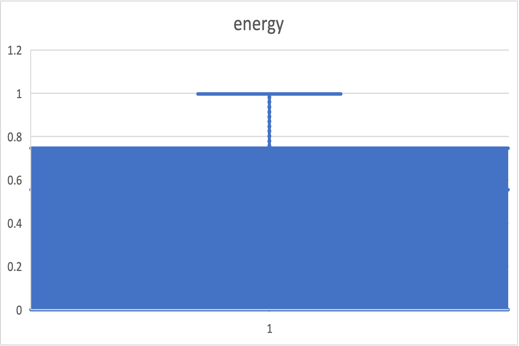
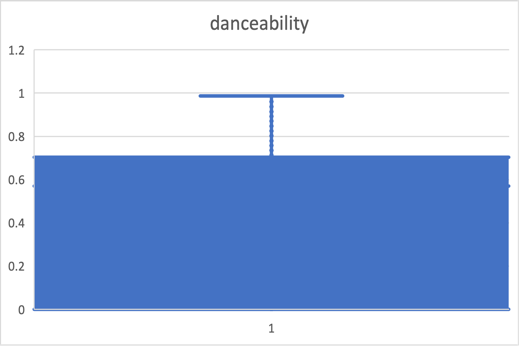
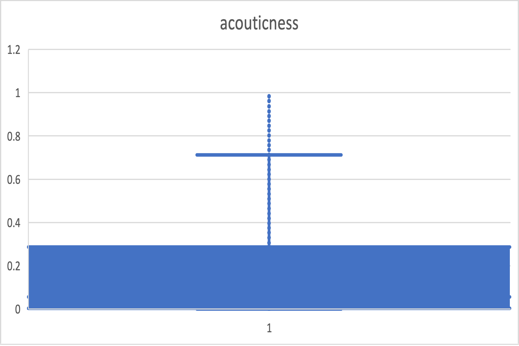
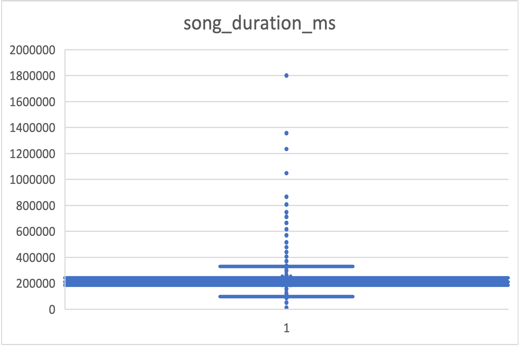
目标属性的分布大致满足正态分布



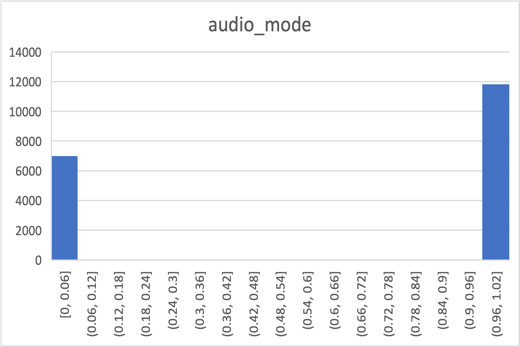
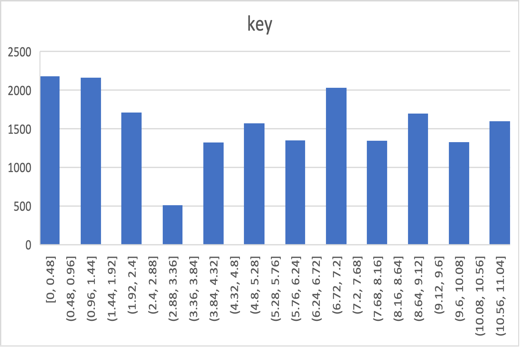
连续数据分析

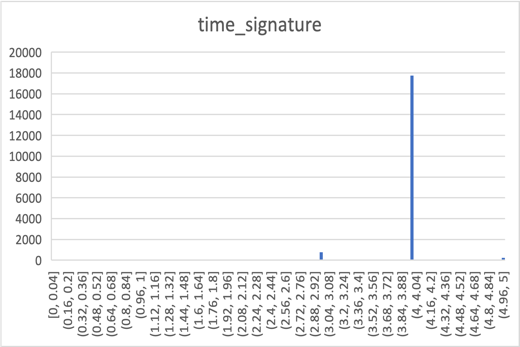
以下的直方图展示了数据集所包含的十个连续属性的分析，图中展示了这些属性的分布情况，其中箱形图还展示了这些连续属性有的还含有一些离散群，这些离散群会影响训练的效果，在程序设计的过程中，如果有必要的话，可以对这些离散群进行一定的处理，使得训练的效果更好。





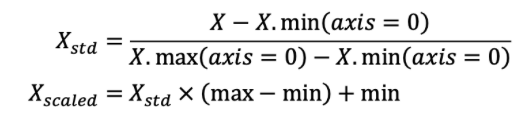
离散数据分析





数据预处理

连续数据归一化

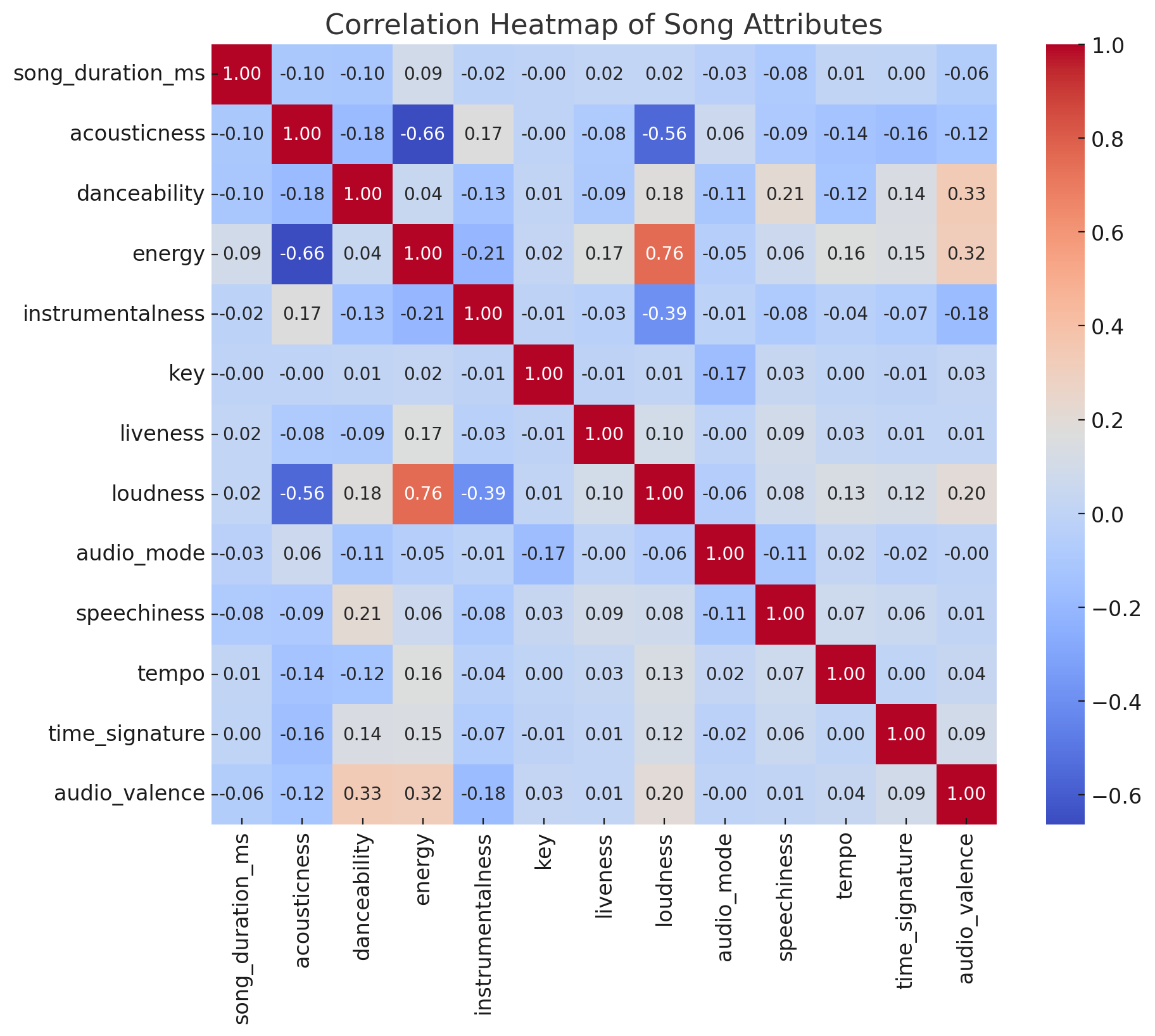
引用sklearn.preprocessor的MinMaxScaler，将连续属性的数据归一化到[0,1]范围内

在数据处理的过程中要注意数据范围的选择，不要将离散数据和目标属性一起进行了归一化处理，这会导致计算得到的模型效果下降，甚至产生严重的偏差。

离散数据连续化

离散那属性的数据的特点是，属性的值离散分布，并且数字的大小关系没有实际上的意义，因此引用sklearn.preprocessor的OneHotEncoder，对离散属性的数据进行独热编码。

共线性检测和处理



用heatmap观察各个属性发现，不同属性之间存在共线性关系。

共线性是指在线性回归模型中，解释变量（自变量）之间存在精确或高度相关的关系，在模型训练的过程中，如果连两个属性之间有线性关系，属性的数值发生变化的时候，因变量发生相应的变化，但如果存在共线性，模型就无法分辨到底是哪个属性的变化造成了因变量的变化，这可能导致模型估计失真或难以准确估计。

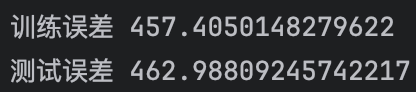
共线性可以通过方差膨胀因子（VIF）来进行检验，VIF值大于10通常被认为是共线性问题的指标。解决共线性问题的常用方法包括手动移除共线性变量、使用逐步回归法、或采用岭回归。此外，主成分回归也是一种处理共线性问题的技术，它通过提取多个变量的主成分来降低变量的维度，从而减少共线性的影响

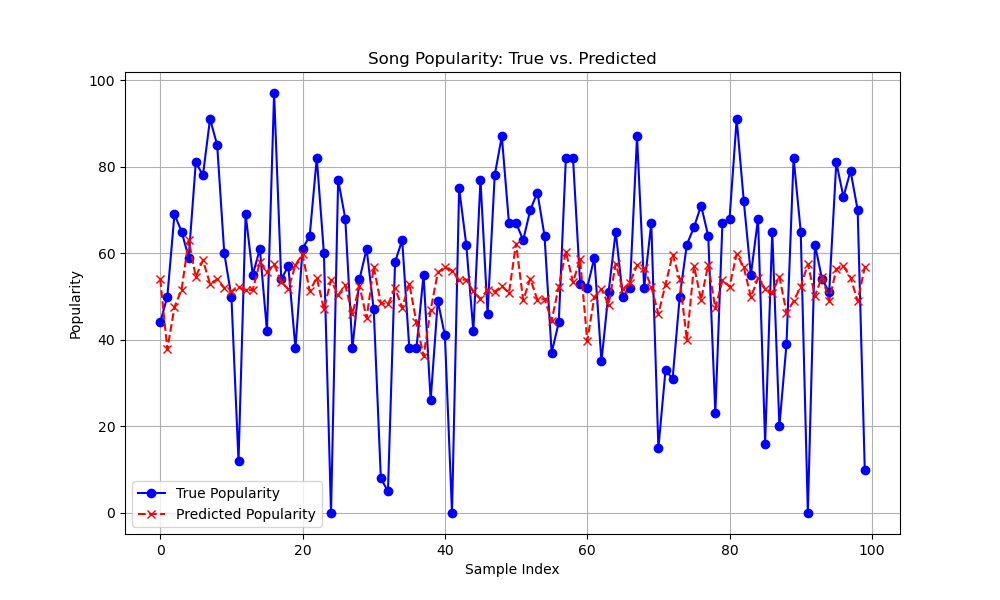
训练与预测

使用80%的数据集进行训练，20%的数据集进行测试

**实验结果**

训练得到的模型在训练集和测试集上的MSE分数



在测试集的前一百个样本中的预测值与真实值的可视化图表

**结果分析**

1. 原始的数据中有较多的离散群，这些离散群对模型的训练会造成不利的影响，在考虑进一步提升训练效果的时候可以考虑对原始数据的离散群进行进一步的处理
2. 有很多属性存在共线性问题，可以通过PCA降维将重复属性去除
3. 训练得到的MSE较大，可能线性模型不太适合这个复杂度的数据集，拟合效果一般
4. 对离散属性进行简单的独热编码可能欠考虑，离散值之间的大小关系可能跟目标属性有关系，在离散属性连续化的过程中，忽略了这一点

**代码附录**

import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.decomposition import PCA  
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
from sklearn.model\_selection import train\_test\_split  
  
# 加载数据集  
df = pd.read\_csv("song\_data.csv")  
  
# df是原始的DataFrame  
# 离散特征列  
discrete\_columns = ['key', 'audio\_mode', 'time\_signature']  
  
# 连续特征列  
continuous\_columns = [col for col in df.columns if col not in discrete\_columns + ['song\_name', 'song\_popularity']]  
  
# One-Hot编码离散特征  
  
# 初始化OneHotEncoder  
encoder = OneHotEncoder(sparse=False)  
  
# 对离散特征列进行独热编码  
encoded\_data = encoder.fit\_transform(df[discrete\_columns])  
  
# 将独热编码的数据转换为DataFrame  
encoded\_df = pd.DataFrame(encoded\_data, columns=encoder.get\_feature\_names\_out())  
  
# 选择连续特征进行归一化  
scaler = MinMaxScaler()  
df[continuous\_columns] = scaler.fit\_transform(df[continuous\_columns])  
  
"""  
vif = pd.DataFrame()  
vif["variables"] = features\_after\_encoding  
vif["VIF"] = [variance\_inflation\_factor(df\_merged.values, i) for i in range(df\_merged.values.shape[1])]  
print("vif: ",vif)  
"""  
  
# X为输入属性，y为输出属性  
X = df[continuous\_columns].values  
y = df['song\_popularity'].values  
  
# 使用PCA进行特征降维  
pca = PCA(n\_components=8)  
X\_pca = pca.fit\_transform(X)  
  
# 输出每个主成分的贡献率  
print("输出每个主成分的贡献率： ", pca.explained\_variance\_ratio\_)  
  
# 输出降维后的属性矩阵  
print("降维后的属性矩阵： ", X\_pca.shape)  
  
# 合并离散特征和连续特征  
X\_pca = pd.DataFrame(X\_pca)  
X\_pca.columns = X\_pca.columns.astype(str)  
df\_merged = pd.concat([X\_pca, encoded\_df], axis=1)  
  
# 划分数据集  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(df\_merged, y, test\_size=0.2, random\_state=42)  
  
# 构建线性回归模型  
lr\_model = LinearRegression()  
lr\_model.fit(X\_train, y\_train)  
  
# 预测训练集和测试集的结果  
y\_train\_predict = lr\_model.predict(X\_train)  
y\_test\_predict = lr\_model.predict(X\_test)  
  
# 计算训练误差和测试误差  
train\_error = mean\_squared\_error(y\_train, y\_train\_predict)  
test\_error = mean\_squared\_error(y\_test, y\_test\_predict)  
  
# 输出训练误差和测试误差  
print("训练误差", train\_error)  
print("测试误差", test\_error)  
  
# 将真实值和预测值转换为DataFrame  
y\_test\_df = pd.DataFrame(y\_test, columns=['true\_target'])  
y\_test\_predict\_df = pd.DataFrame(y\_test\_predict, columns=['predicted\_target'])  
  
# 合并两个DataFrame  
merged\_df = pd.concat([y\_test\_df, y\_test\_predict\_df], axis=1)  
  
# 导出到CSV文件  
merged\_df.to\_csv('y\_test\_and\_predict.csv', index=False)  
  
# 取测试集的真实值和预测值的前100个样本进行可视化  
plt.figure(figsize=(10, 6))  
plt.plot(y\_test[:100], label='True Popularity', color='blue', marker='o')  
plt.plot(y\_test\_predict[:100], label='Predicted Popularity', color='red', linestyle='--', marker='x')  
plt.title('Song Popularity: True vs. Predicted')  
plt.xlabel('Sample Index')  
plt.ylabel('Popularity')  
plt.legend()  
plt.grid(True)  
plt.show()  
plt.savefig("result.png")