机器学习实验报告

实验名称： 音乐流行度预测

学生姓名： 唐梓烨

学生学号： 58122310

完成日期： 2024/4/9

目录

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

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。

目标

根据歌曲指标预测歌曲流行度。

实验内容

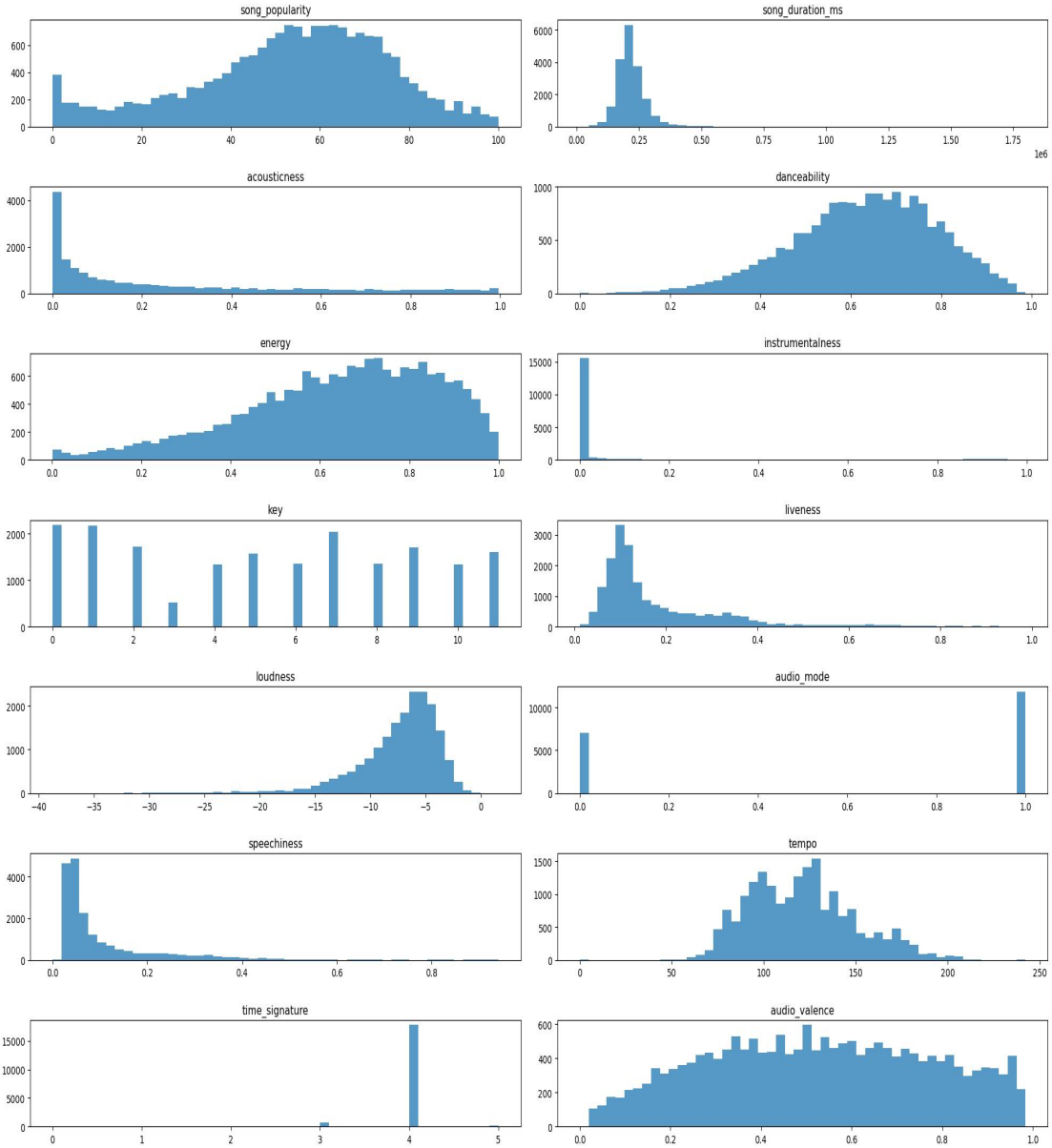
1. 数据预处理

(1) 无关数据剔除

在数据集中，歌曲名称与实验无关，将其剔除。

(2) 观察数据

将数据的属性值和标签值进行可视化，初步观察数据的分布情况。



可以发现一些属性大体成正态分布，歌曲流行度也呈正态分布，但在低流行分数区域歌曲数量增多。

(3) 离散属性连续化

数据集中还存在离散属性，需要将其连续化。

对离散属性连续化可以使用 one-hot 编码。audio_mode 属性只由 0 和 1 勾成，故不用处理。key 和 time_signature 属性由于其是有序离散属性，我认为其中的顺序关系可直接用于线性模型的训练，并不需要进行连续化。（尝试将这两个属性进行独热编码，训练出来的模型性能并无显著提升，故在此不进行连续化）

(4) 数据归一化

由于本数据集的连续属性的上下界已给出，因此使用最大最小值进行归一化。调用 sklearn.preprocessing 中的 MinMaxScaler 进行归一化。具体公式为：

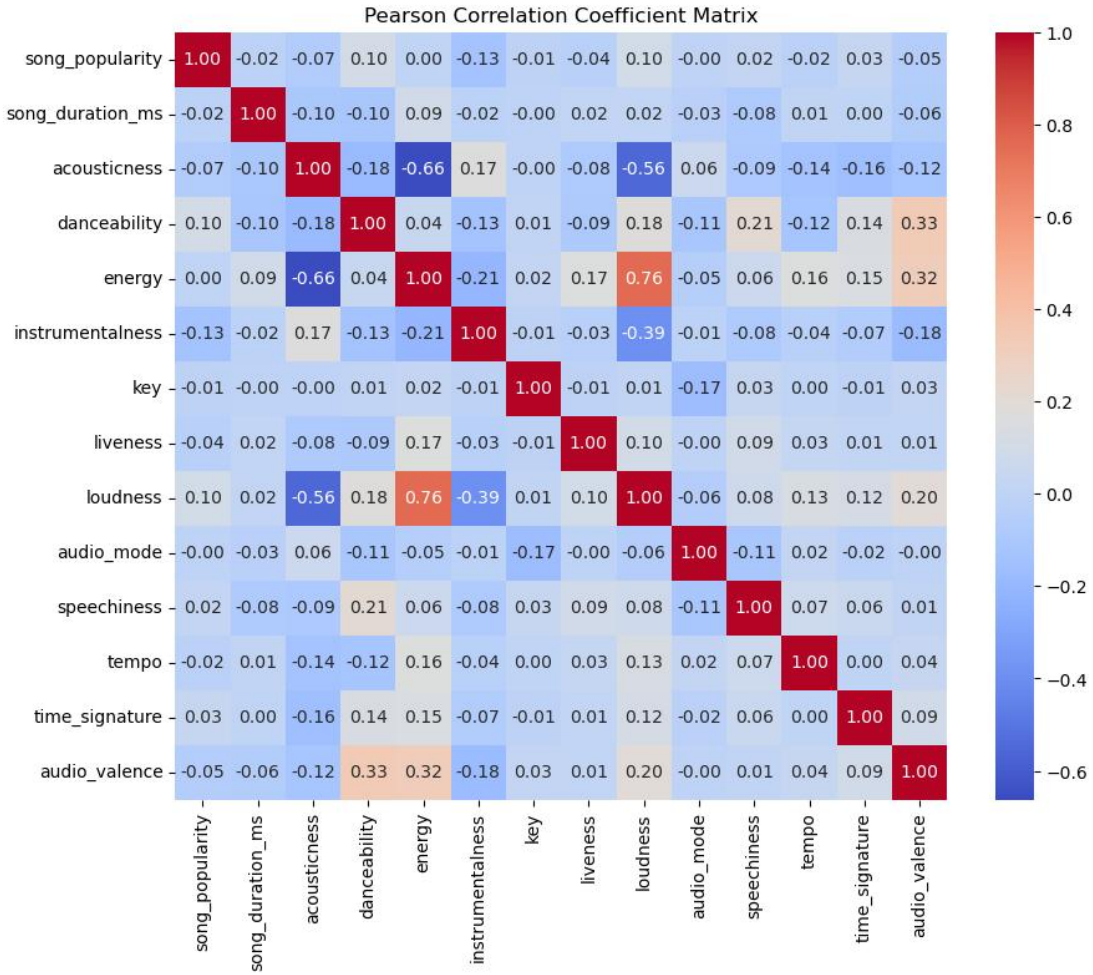
$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

(5) 共线性的检测

属性之间可能存在共线性，需要进行处理。

计算属性两两之间 Pearson 相关系数：Pearson 相关系数衡量的是两个变量间线性关系的强度和方向。其值范围从-1 到 1，其中 1 表示完全正相关，-1 表示完全负相关，0 表示没有线性关系。

相关系数热力图：



从图中可以看出['energy','acousticness','loudness']属性之间可能存在较强的的线性关系。

进一步计算属性间的 VIF 分析共线性：

$$\mathrm{VIF} = \frac{1}{1 - \mathrm{R}^2}$$

其中R^2是决定系数衡量的是在回归模型中，因变量的变异中有多少百分比可以通过自变量的变异来解释。它是回归模型拟合优度的一个度量。这里的R^2是指在考虑多重共线性时，选定一个变量作为因变量，其他变量作为自变量进行线性回归分析得到的R^2值。如果某个变量与其他变量高度线性相关，那么这个变量可以被其他变量很好地预测，其R^2将接近 1，导致 VIF 值很高，表明存在多重共线性问题。

计算方法：

```python
# 进行VIF计算，判断属性间是否存在共线性
from sklearn.linear_model import LinearRegression

def calculate_vif(df):
    vif_df = pd.DataFrame(columns=['Variable', 'VIF'])
    for i, column in enumerate(df.columns):
        X = df.drop(columns=[column])
        y = df[column]

        model = LinearRegression()
        model.fit(X, y)

        r_squared = model.score(X, y)
        vif = 1 / (1 - r_squared)

        vif_df.loc[i] = [column, vif]
    return vif_df
```

在本次实验中，计算属性之间的 VIF 得到

Variable	VIF
song_duration_ms	1.045011
acousticness	1.978771
danceability	1.430254
energy	3.726267
instrumentalness	1.249064
key	1.032045
liveness	1.052267
loudness	2.936453
audio_mode	1.058172
speechiness	1.102085
tempo	1.064834
time_signature	1.045487
audio_valence	1.393416

VIF = 1：没有共线性。

1 < VIF < 5：通常认为共线性是中等的，可以接受。

VIF >= 5：可能存在问题的共线性，需要进一步检查。

VIF >= 10：表明高度共线性，这可能会影响回归模型的准确性和稳定性。
可以发现本次实验数据集的属性之间并不存在明显共线性问题。

(6) 共线性的处理

经过上述共线性分析后，可认为属性之间并不存在明显共线性问题。因此在后续处理中，我采用了三种方法得到最终用于训练的数据。

i. 直接使用前面经过剔除无关数据、归一化完成的数据进行训练

ii. 剔除['energy','acousticness','loudness']属性后进行训练，因为他们互相之间的相关系数较高（>0.5）

iii. 对数据从进行 PCA 主成分分析后进行训练，以实现数据降维，减轻计算代价

2. 实验设置

(1) 实验评估方法：单次留出法。

(2) 性能度量：MSE

使用 Python 的 Scikit-Learn 库中的 mean_squared_error 函数来计算测试集上均方误差作为模型评估方法。

(3) 训练数据选择：剔除歌曲名称数据后，使用共线性的处理部分的三种方法预处理的数据分别进行训练。

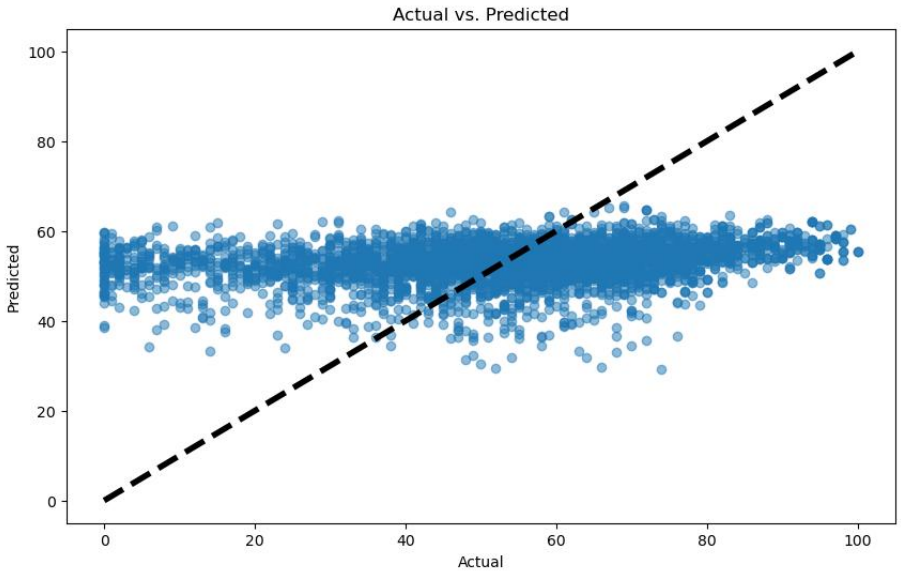
(4) 数据集的划分：使用 80%的数据集进行训练，20%的数据集进行测试。

实验结果

1. 直接使用归一化后的数据训练的结果

```
Mean Squared Error of test: 445.50357212712674
Mean Squared Error of train: 460.86694157228726
       Actual Value  Predicted Value
5451             57        54.936647
7258             41        55.698305
...
5848             40        50.846724
10302            77        50.397594
```

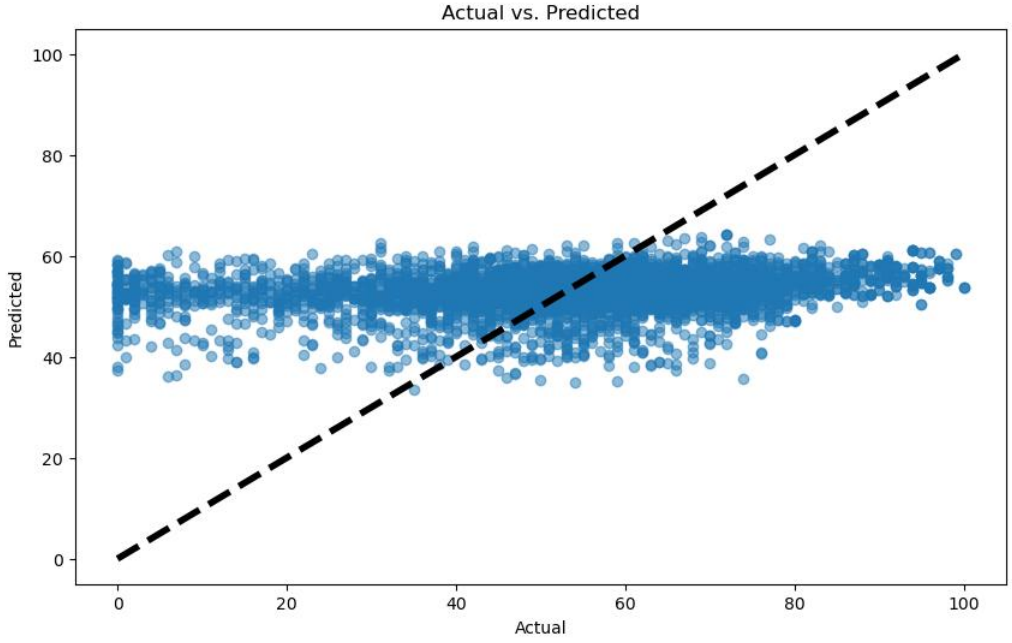
真实值与预测值的散点图：



2. 使用剔除了相关系数较高的属性数据训练的结果

```
Mean Squared Error of test: 450.53620332165593
Mean Squared Error of train: 463.76323136718935
       Actual Value  Predicted Value
5451             57        55.615182
7258             41        54.578930
...
5848             40        48.438721
10302            77        51.311228
```

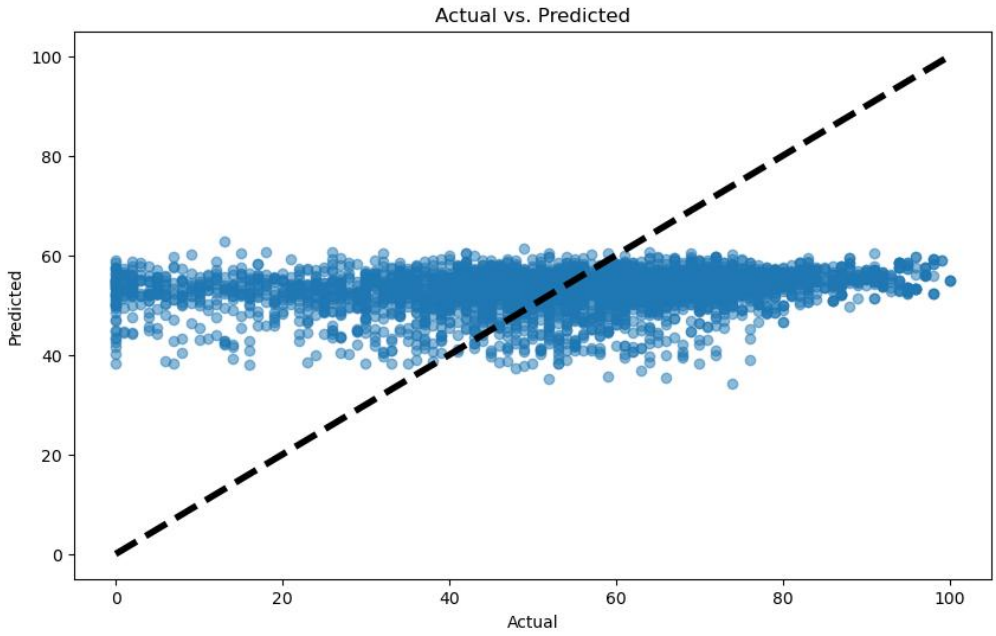
真实值与预测值的散点图：



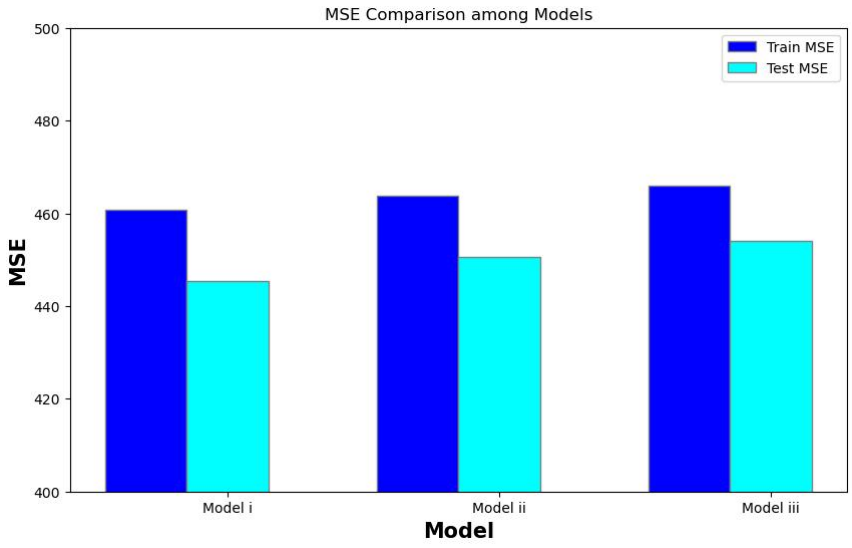
3. 使用 PCA 降维数据，保留前十个最重要的特征向量训练的结果

```
Mean Squared Error of test: 453.97803317172486
Mean Squared Error of train: 465.98514521395896
       Actual Value  Predicted Value
5451             57        54.086065
7258             41        51.189807
4703             64        51.182796
9285             62        55.572427
9354             63        51.896113
...             ...              ...
9718             11        53.996612
12017            56        46.269616
16531            76        55.983599
5848             40        49.122383
10302            77        50.322329
```

真实值与预测值的散点图：



三种方法的 MSE 值对比图：



结果分析

● 本数据集中的离散属性 key，time_signature 存在有序关系，例如过高或过低的 key 会导致音乐流行指数下降，因此我认为这两个离散属性不需要进行独热编码，只需将其归一化即可。

经尝试，在方法 i 下，将[key，time_signature]属性使用独热编码和不使用独热编码训练的模型的 MSE 对比如下：

使用独热编码：

```
Mean Squared Error of test: 444.2834935380992
Mean Squared Error of train: 459.62516829408065
```

不使用独热编码：

```
Mean Squared Error of test: 445.50357212712674
Mean Squared Error of train: 460.86694157228726
```

无显著区别。

● 在共线性检测中，计算每个属性的 VIF 值得到最大的为 3.726267，并不超过 5，认为其实本数据集并没有明显共线性问题。

● 在本次实验中，使用多种方法处理数据集后，使用线性回归模型进行训练，得到的模型预测误差 MSE 在 450 左右。这是一个较大的误差，但尝试了多种数据预处理的方式都没法很好的改善模型性能，推断线性模型并不适合本次实验。

在观察预测值-真实值散点图时也可以发现大部分预测值集中在 50 附近，并没有出现预测值与真实值的正相关关系，这也说明模型没有很好的利用到属性值来进行预测。

代码附录

```
# 读取数据
import pandas as pd


# 读取 CSV 文件
df = pd.read_csv('song_data.csv')


print(df.head())
```

```

```

```
# 数据预处理
# 去除无关列
df = df.drop(columns='song_name')
```

```

```

```
# 可视化样本的每个属性和标签的分布
import matplotlib.pyplot as plt
plt.figure(figsize=(20, 15))

# 为每个属性绘制直方图
for i, column in enumerate(df.columns, 1):
    plt.subplot(5, 3, i)
    plt.hist(df[column], bins=50, alpha=0.75)
    plt.title(f'{column}')

plt.tight_layout()
plt.show()
```

```
# 归一化
from sklearn.preprocessing import MinMaxScaler

# 实例化 MinMaxScaler
scaler = MinMaxScaler()

# 归一化数据
df_scaled = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

# 将数据的标签列去除，只保留属性
df_scaled = df_scaled.drop('song_popularity', axis=1)

# 显示归一化后的数据
print(df_scaled.head())
```

```
# 进行 VIF 计算，判断属性间是否存在共线性
from sklearn.linear_model import LinearRegression

def calculate_vif(df):
    vif_df = pd.DataFrame(columns=['Variable', 'VIF'])
    for i, column in enumerate(df.columns):
        X = df.drop(columns=[column])
        y = df[column]

        model = LinearRegression()
        model.fit(X, y)

        r_squared = model.score(X, y)
        vif = 1 / (1 - r_squared)

        vif_df.loc[i] = [column, vif]
```

```
    return vif_df


print(calculate_vif(df_scaled))
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

correlation_matrix = df.corr()

# 输出相关系数矩阵，用于判断属性两两之间是否存在线性关系
print(correlation_matrix)

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Pearson Correlation Coefficient Matrix')
plt.show()
```

```
# 进行 PCA 主成分分析，降维数据，降低共线性
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# 标准化数据，将数据转换为均值为 0，方法为 1 的数据
scaler = StandardScaler()
df_normal = scaler.fit_transform(df_scaled)

# 初始化 PCA，降维到 10 个主成分（保留前 10 个最重要的特征向量）
pca = PCA(n_components=10)
# 对数据进行 PCA 降维
df_pca = pca.fit_transform(df_normal)

df_pca = pd.DataFrame(data=df_pca,columns = [f'col{i+1}' for i in range(10)])

# 打印降维后的数据
print(df_pca)

# 打印主成分解释的方差比例
print("Variance Ratio of Principal Components:", pca.explained_variance_ratio_)
```

```
# 进行模型训练和评估
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

```
# 将标签列添加到 PCA 处理后的 DataFrame 中
df_scaled['song_popularity'] = df['song_popularity']
print(df_scaled.head())
# 划分数据集
# X 为属性集，y 为标签
X = df_scaled.drop('song_popularity', axis=1)
y = df_scaled['song_popularity']

# 使用 train_test_split 分割数据集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=41)

# 初始化线性回归模型
lr = LinearRegression()

# 训练模型
lr.fit(X_train, y_train)

# 预测测试集和训练集
y_test_pred = lr.predict(X_test)
y_train_pred = lr.predict(X_train)

# 计算均方误差（MSE）
mse_test = mean_squared_error(y_test, y_test_pred)
mse_train = mean_squared_error(y_train, y_train_pred)
print(f'Mean Squared Error of test: {mse_test}\nMean Squared Error of train: {mse_train}')

# 可视化模型预测值和真实值
results_df = pd.DataFrame({
    'Actual Value': y_test,
    'Predicted Value': y_test_pred
})

print(results_df)
```

```
import matplotlib.pyplot as plt

#  y_test 是测试集的真实值，y_test_pred 是模型的预测值
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_test_pred, alpha=0.5)  # 绘制实际值与预测值的散点图
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)  # 完美预测的基线
plt.xlabel('Actual')
plt.ylabel('Predicted')
```

```
plt.title('Actual vs. Predicted')
plt.show()
```