In [1]:
```python
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score, KFold
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import make_scorer, mean_squared_error
```

Importing merged data

In [2]:
```python
df = pd.read_excel(r"C:\Users\USER\Desktop\Popularity Datasets\merged_data.xlsx")
```

In [3]:
```python
df.head()
```

Out[3]:

	song_id	duration_ms	key	mode	time_signature	acousticness	danceability	energy	instrumentalness	liveness	loudness	speechiness	valence	tempo	popularity
0	3e9HZxeyfWwjeyPAMmWSSQ	207320	1	1	4	0.22900	0.717	0.653	0.000000	0.1010	-5.634	0.0658	0.412	106.966	83.0
1	5p7ujcrUXASCNwRaWNHR1C	201661	6	1	4	0.29700	0.752	0.488	0.000009	0.0936	-7.050	0.0705	0.533	136.041	85.0
2	2xLMifQCjDGFmkHkpNLD9h	312820	8	1	4	0.00513	0.834	0.730	0.000000	0.1240	-3.714	0.2220	0.446	155.008	60.0
3	3KkXRkHbMCARz0aVfEt68P	158040	2	1	4	0.55600	0.760	0.479	0.000000	0.0703	-5.574	0.0466	0.913	89.911	92.0
4	1rqqCSm0Qe4I9rUvWncaom	190947	5	1	4	0.19300	0.579	0.904	0.000000	0.0640	-2.729	0.0618	0.681	82.014	91.0

Handling missing values

In [4]:
```python
df.isna().sum()
```

Out[4]:
```
song_id               0
duration_ms           0
key                   0
mode                  0
time_signature        0
acousticness          0
danceability          0
energy                0
instrumentalness      0
liveness              0
loudness              0
speechiness           0
valence               0
tempo                 0
popularity          993
dtype: int64
```

In [5]:
```python
df = df.dropna()
```

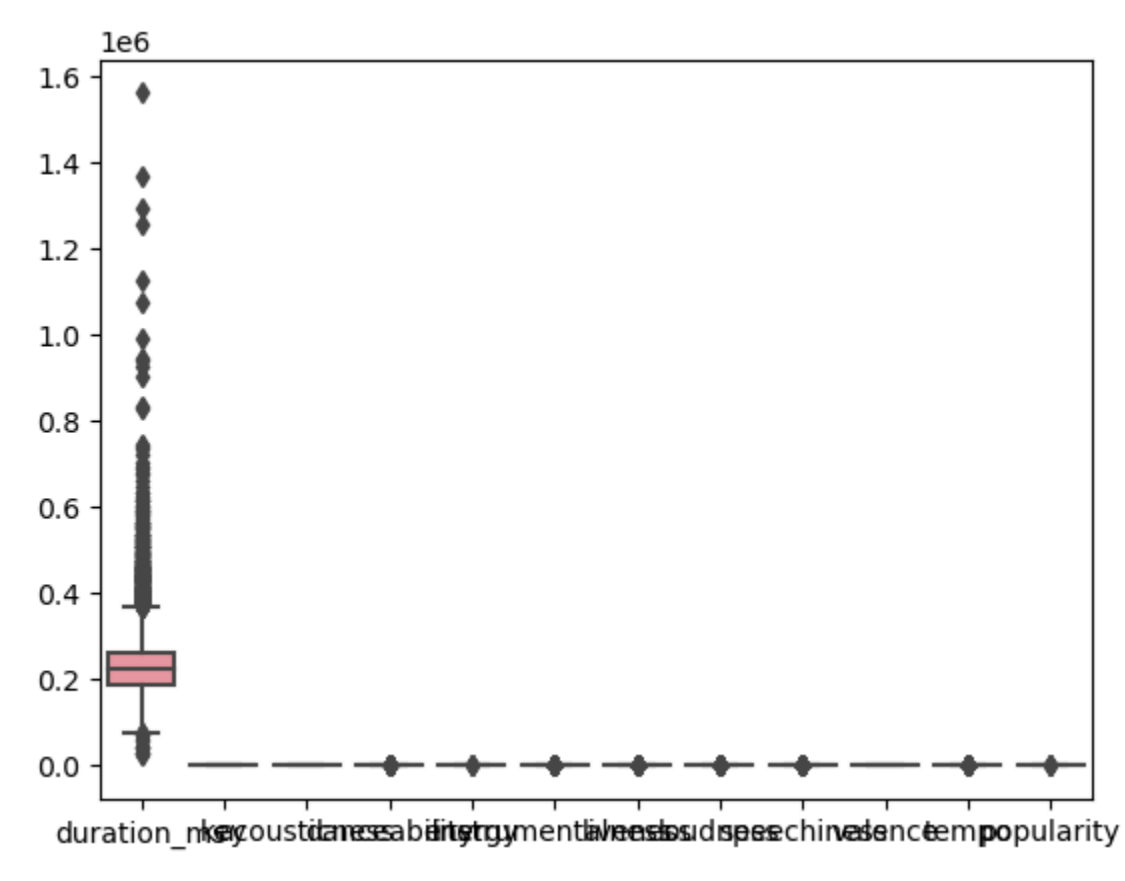
In [6]:
```python
df.isna().sum()
```

Out[6]:
```
song_id             0
duration_ms         0
key                 0
mode                0
time_signature      0
acousticness        0
danceability        0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
valence             0
tempo               0
popularity          0
dtype: int64
```

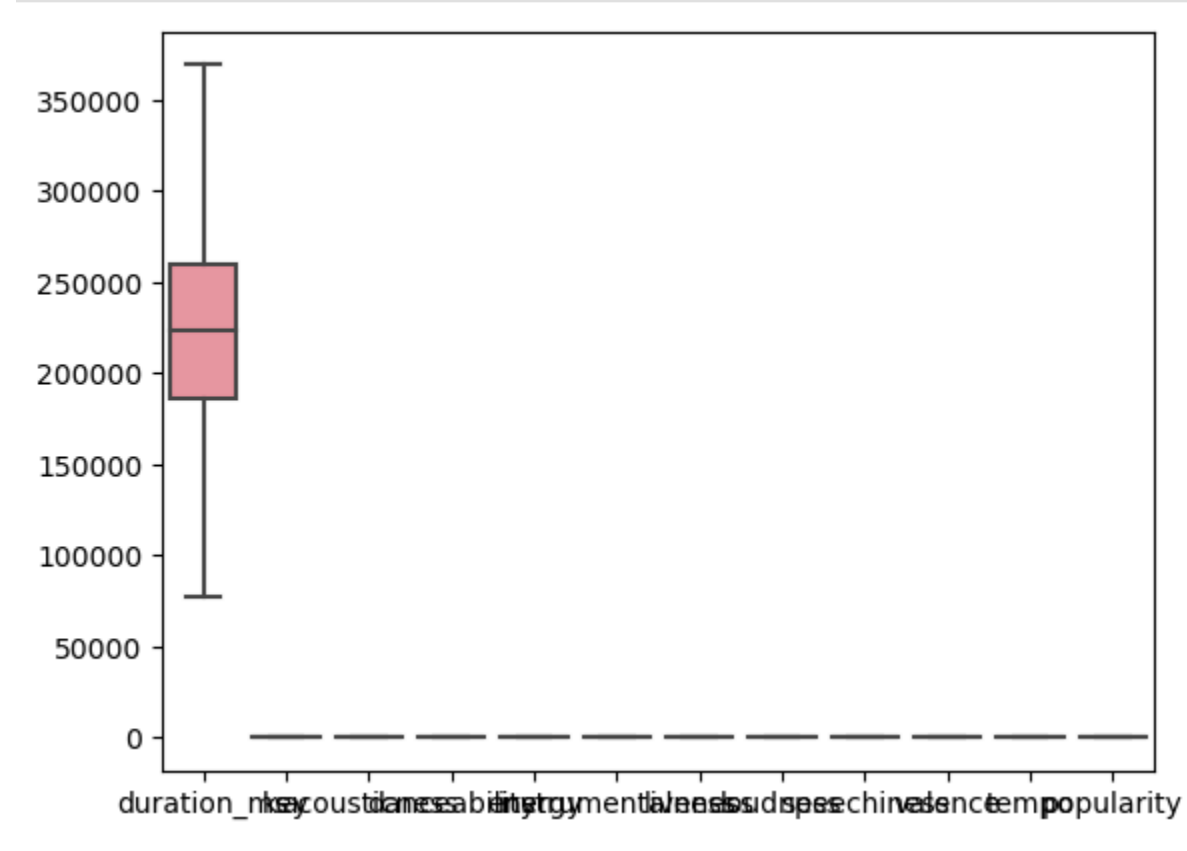
Handling ouliers

In [7]:
```python
sns.boxplot(df[['song_id', 'duration_ms', 'key',
       'acousticness', 'danceability', 'energy', 'instrumentalness',
       'liveness', 'loudness', 'speechiness', 'valence', 'tempo',
       'popularity']])
```

Out[7]: <Axes: >



In [8]:
```python
cols = ['song_id', 'duration_ms', 'key', 'time_signature',
       'acousticness', 'danceability', 'energy', 'instrumentalness',
       'liveness', 'loudness', 'speechiness', 'valence', 'tempo',
       'popularity']
def correct_outliers_iqr(df, threshold=1.5):
    for col in cols:
        if df[col].dtype in ['int64', 'float64']:
            Q1 = df[col].quantile(0.25)
            Q3 = df[col].quantile(0.75)
            IQR = Q3 - Q1
            lower_bound = Q1 - threshold * IQR
            upper_bound = Q3 + threshold * IQR
            df[col] = np.where(df[col] < lower_bound, lower_bound,
                               np.where(df[col] > upper_bound, upper_bound, df[col]))
    return df
df = correct_outliers_iqr(df)
sns.boxplot(df[['song_id', 'duration_ms', 'key',
       'acousticness', 'danceability', 'energy', 'instrumentalness',
       'liveness', 'loudness', 'speechiness', 'valence', 'tempo',
       'popularity']])
plt.show()
```



Exploratory data analysis

Descriptive Statistics

In [9]:
```python
df.describe()
```

Out[9]:

	duration_ms	key	mode	time_signature	acousticness	danceability	energy	instrumentalness	liveness	loudness	speechiness	valence	tempo	popularity
count	19412.000000	19412.000000	19412.000000	19412.0	19412.000000	19412.000000	19412.000000	19412.000000	19412.000000	19412.000000	19412.000000	19412.000000	19412.000000	19412.000000
mean	227183.498139	5.223728	0.727128	4.0	0.265084	0.600175	0.625311	0.000721	0.180434	-8.831543	0.052288	0.607150	120.209436	42.305018
std	56330.318593	3.571198	0.445447	0.0	0.264527	0.150344	0.197321	0.001151	0.126371	3.546715	0.027650	0.236911	27.280311	17.399278
min	76924.875000	0.000000	0.000000	4.0	0.000001	0.193500	0.026500	0.000000	0.013000	-18.950500	0.000000	0.000000	45.608250	0.000000
25%	186489.750000	2.000000	0.000000	4.0	0.038900	0.501000	0.481000	0.000000	0.088400	-11.209000	0.031900	0.425750	99.795750	31.000000
50%	223107.000000	5.000000	1.000000	4.0	0.169000	0.607000	0.640000	0.000010	0.129000	-8.469500	0.040400	0.632000	118.975000	43.000000
75%	259533.000000	8.000000	1.000000	4.0	0.440000	0.706000	0.784000	0.001130	0.249000	-6.048000	0.063000	0.804000	135.920750	55.000000
max	369097.875000	11.000000	1.000000	4.0	0.995000	0.988000	0.998000	0.002825	0.489900	1.693500	0.109650	1.000000	190.108250	91.000000

Correllation

In [10]:
```python
correlation_matrix = df.corr()['popularity']
correlation_matrix
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_6900\1872523458.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will defa
ult to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = df.corr()['popularity']
```

Out[10]:
```
duration_ms         0.028248
key                -0.004338
mode               -0.023353
time_signature           NaN
acousticness       -0.068465
danceability        0.031640
energy              0.046531
instrumentalness   -0.052700
liveness           -0.030577
loudness            0.119268
speechiness         0.048556
valence            -0.090769
tempo               0.028276
popularity          1.000000
Name: popularity, dtype: float64
```

In [11]:
```python
feature_columns = ['acousticness', 'danceability', 'energy','loudness', 'speechiness', 'valence', 'tempo','popularity']
df = df[feature_columns]
```

In [12]:
```python
df.head()
```

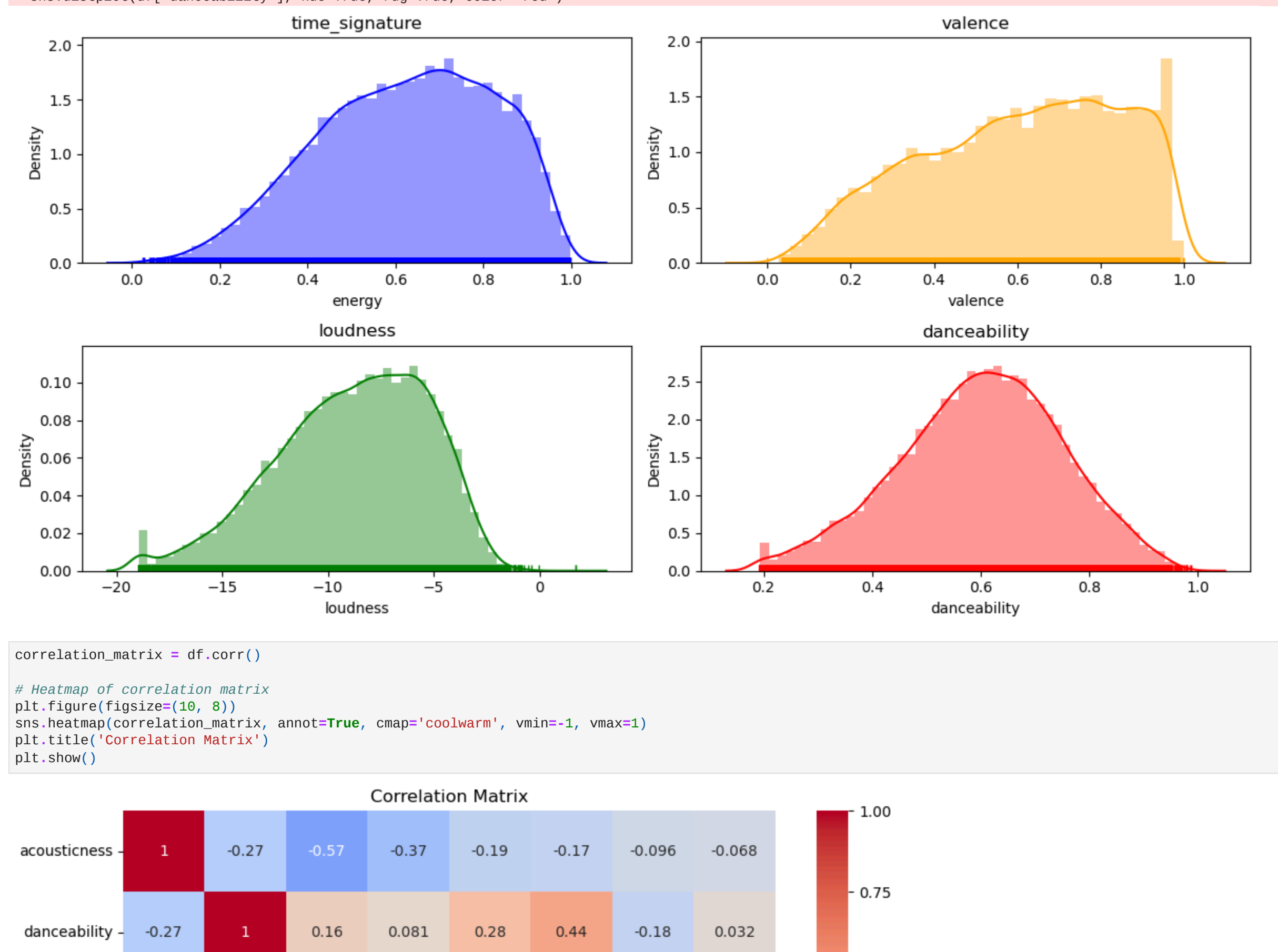
Out[12]:

	acousticness	danceability	energy	loudness	speechiness	valence	tempo	popularity
0	0.22900	0.717	0.653	-5.634	0.06580	0.412	106.966	83.0
1	0.29700	0.752	0.488	-7.050	0.07050	0.533	136.041	85.0
2	0.00513	0.834	0.730	-3.714	0.10965	0.446	155.008	60.0
3	0.55600	0.760	0.479	-5.574	0.04660	0.913	89.911	91.0
4	0.19300	0.579	0.904	-2.729	0.06180	0.681	82.014	91.0

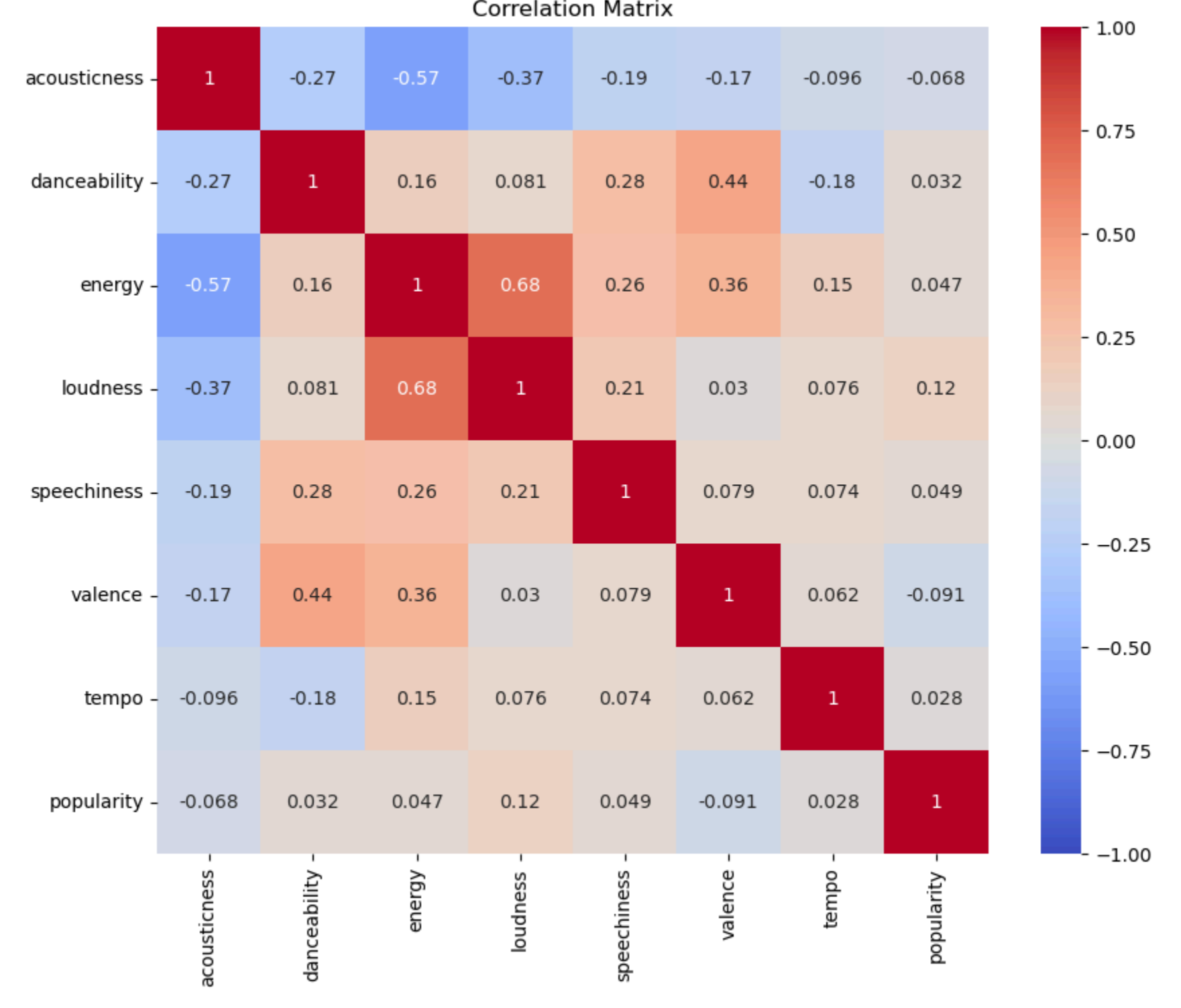
Distribution of the variable selected

In [13]:
```python
plt.figure(figsize=(12, 6))

plt.subplot(2, 2, 1)
sns.distplot(df['energy'],kde=True, rug=True, color='blue')
plt.title('time_signature')

plt.subplot(2, 2, 2)
sns.distplot(df['valence'], kde=True, rug=True, color='orange')
plt.title('valence')

plt.subplot(2, 2, 3)
sns.distplot(df['loudness'], kde=True, rug=True, color='green')
plt.title('loudness')

plt.subplot(2, 2, 4)
sns.distplot(df['danceability'], kde=True, rug=True, color='red')
plt.title('danceability')

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

```
C:\Users\USER\AppData\Local\Temp\ipykernel_6900\1590896074.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['energy'],kde=True, rug=True, color='blue')
C:\Users\USER\AppData\Local\Temp\ipykernel_6900\1590896074.py:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['valence'], kde=True, rug=True, color='orange')
C:\Users\USER\AppData\Local\Temp\ipykernel_6900\1590896074.py:12: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['loudness'], kde=True, rug=True, color='green')
C:\Users\USER\AppData\Local\Temp\ipykernel_6900\1590896074.py:16: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['danceability'], kde=True, rug=True, color='red')
```



In [14]:
```python
correlation_matrix = df.corr()

# Heatmap of correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix')
plt.show()
```



Modelling

In [15]:
```python
feature_columns = ['acousticness', 'danceability', 'energy','loudness', 'speechiness', 'valence', 'tempo']
target_column = 'popularity'

X = df[feature_columns]
y = df[target_column]
```

In [16]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [17]:
```python
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [18]:
```python
lr_model = LinearRegression()
lr_model.fit(X_train_scaled, y_train)

y_pred = lr_model.predict(X_test_scaled)

mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)
print()

rmse_scorer = make_scorer(mean_squared_error, squared=False)

kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(lr_model, X_scaled, y, cv=kf, scoring=rmse_scorer)

print("RMSE for each fold:", cv_scores)
print("Average RMSE:", np.mean(cv_scores))
```

```
Mean Absolute Error (MAE): 13.927757789742147

RMSE for each fold: [17.32793648 16.99006541 17.10479106 17.15523788 17.1038702 ]
Average RMSE: 17.13638020732878
```

In [20]:
```python
model = RandomForestRegressor()
model.fit(X_train_scaled, y_train)


y_pred = model.predict(X_test_scaled)

mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)
print()

rmse_scorer = make_scorer(mean_squared_error, squared=False)

kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(model, X_scaled, y, cv=kf, scoring=rmse_scorer)

print("RMSE for each fold:", cv_scores)
print("Average RMSE:", np.mean(cv_scores))
```

```
Mean Absolute Error (MAE): 14.078977594643316

RMSE for each fold: [17.52859453 17.20709655 17.29496053 17.35123154 17.19264653]
Average RMSE: 17.314905937080802
```

In [21]:
```python
model = DecisionTreeRegressor()
model.fit(X_train_scaled, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test_scaled)

# Evaluate the model's performance
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)
print()

# Define the RMSE scorer
rmse_scorer = make_scorer(mean_squared_error, squared=False)

# Perform k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(model, X_scaled, y, cv=kf, scoring=rmse_scorer)

# Calculate and print the RMSE for each fold and the average RMSE
print("RMSE for each fold:", cv_scores)
print("Average RMSE:", np.mean(cv_scores))
```

```
Mean Absolute Error (MAE): 19.66804017512233

RMSE for each fold: [24.5976377  24.23672845 24.60242341 24.53545207 24.12288756]
Average RMSE: 24.419025837140627
```

In [ ]:
```python
import pickle
filename = "C:\\Users\\USER\\Desktop\\Demo\\savedmodel.pkl"
pickle.dump(lr_model, open(filename, 'wb'))
```