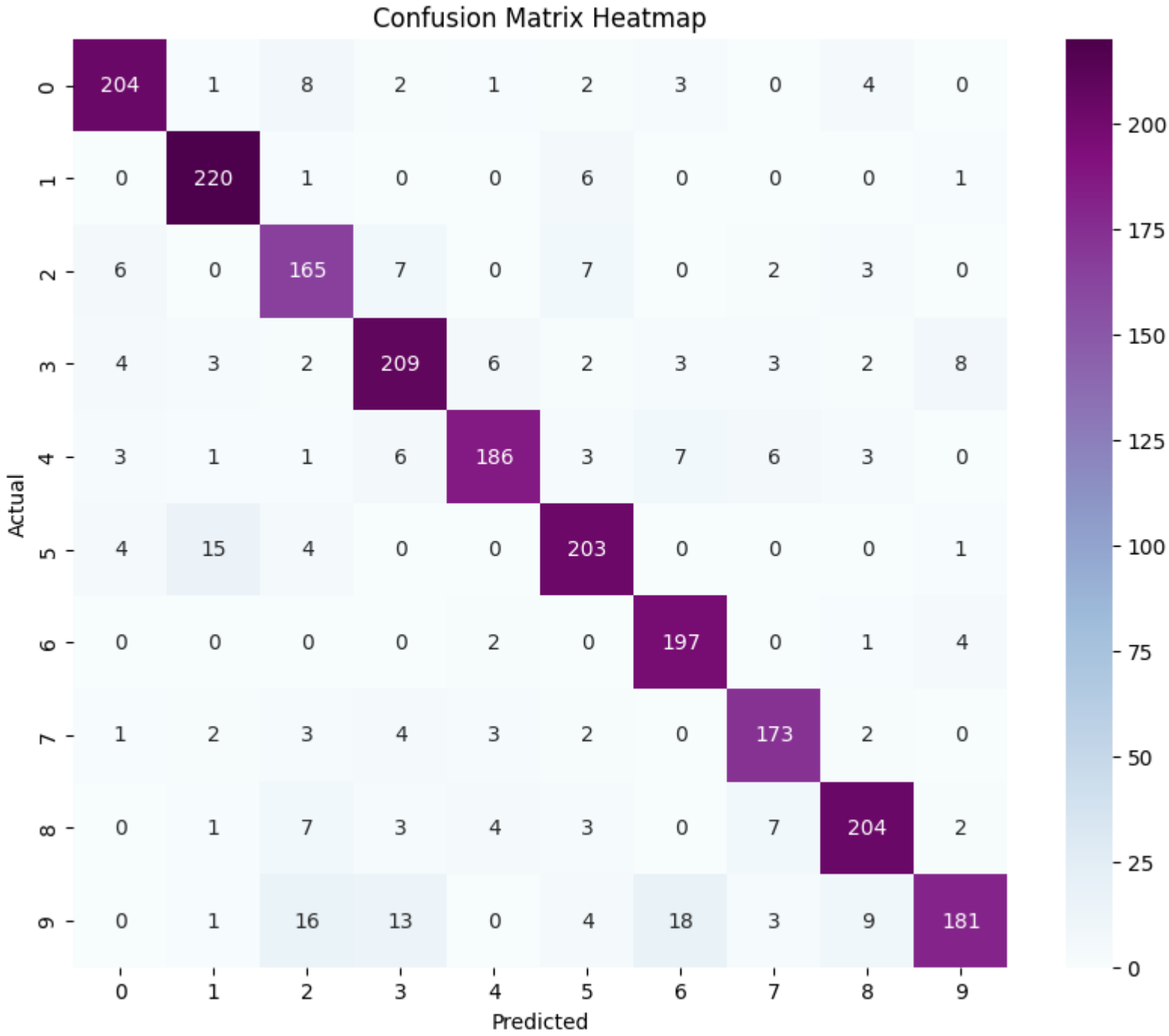
Project 1 - Music Genre Classification

In [1]:
```
## Load necessary Libraries
import numpy as np
import os
import pickle
import random
import operator
import math
import numpy as np
from collections import defaultdict
from python_speech_features import mfcc
import scipy.io.wavfile as wav
import librosa
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.utils import shuffle
```

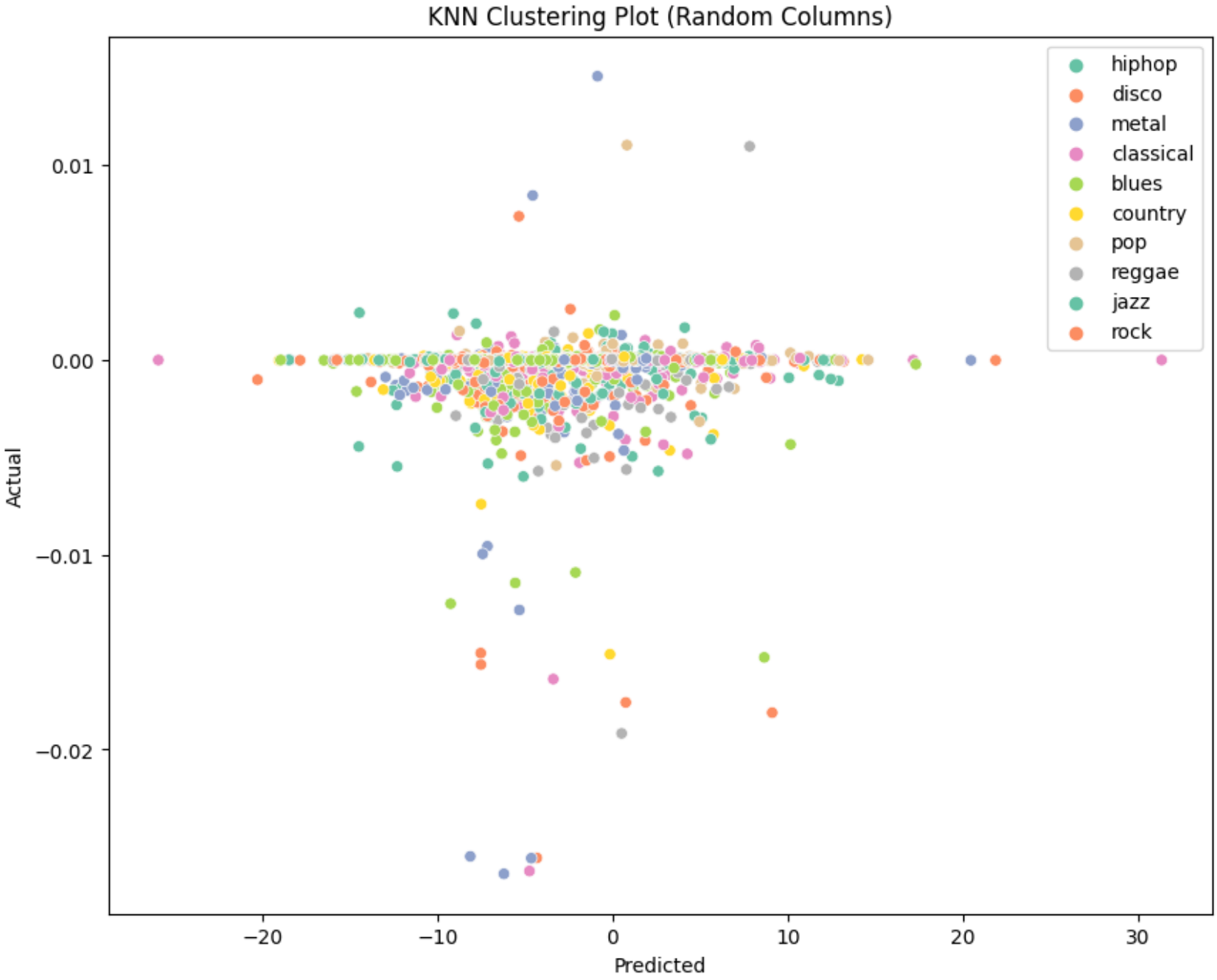
In [2]:
```
## Load first CSV file into a DataFrame
df1 = pd.read_csv('U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/features_3_sec.csv')
```

In [3]:
```
## Load second CSV file into another DataFrame
df2 = pd.read_csv('U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/features_30_sec.csv')
```

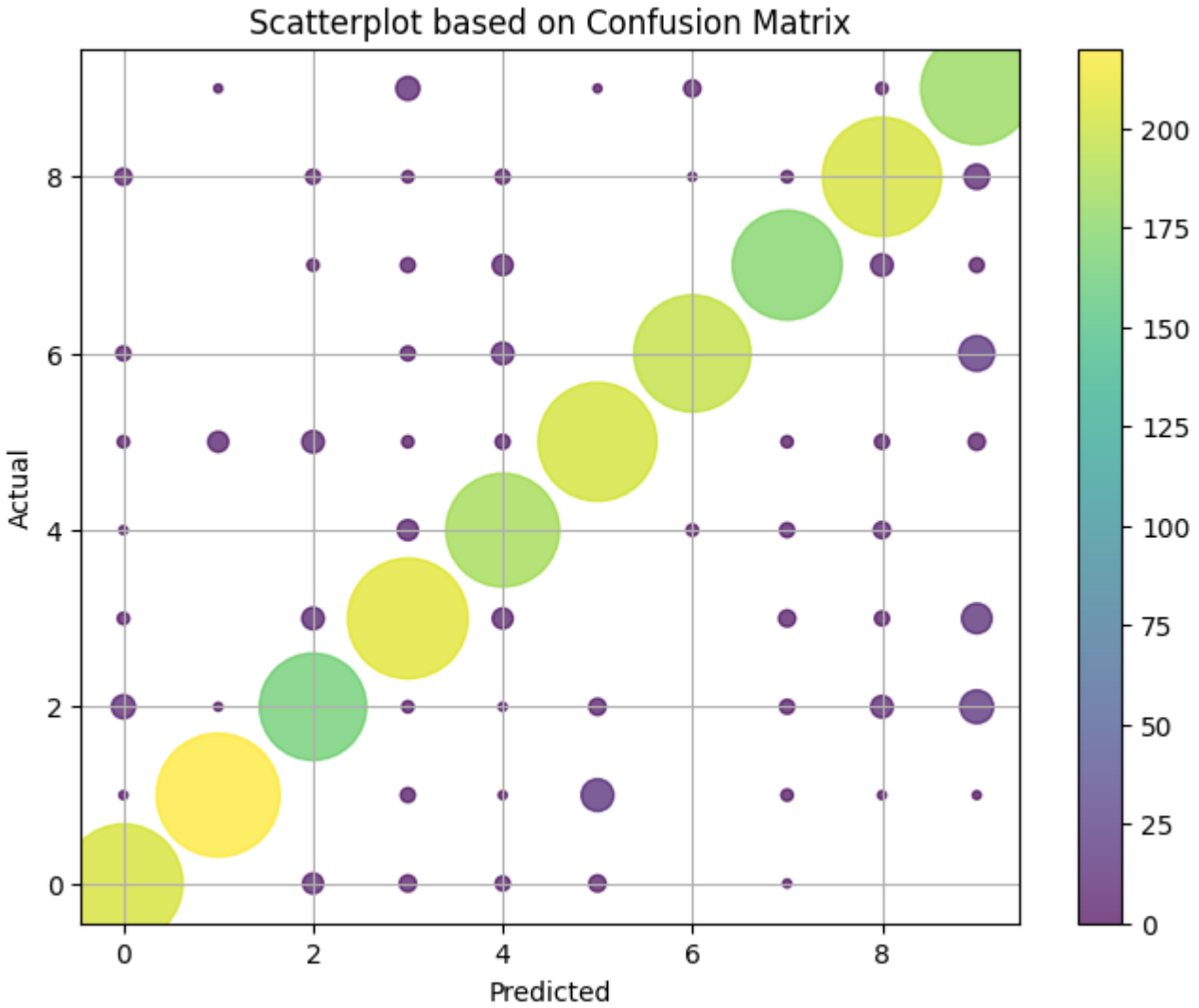
In [4]:
```
## Merge both DataFrames
merged_df = pd.concat([df1, df2], ignore_index=True)

## Remove 'filename' column
merged_df = merged_df.drop(columns=['filename'])

## Split merged DataFrame into x and y, and drop label column
X = merged_df.drop(columns=['label'])
y = merged_df['label']

## Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

## Train Random Forest classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

## Make predictions on test data
y_pred = clf.predict(X_test)

## Calculate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

## Create confusion heatmap matrix
plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='BuPu')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()
```


Confusion Matrix Heatmap
Actual
Predicted
0 1 2 3 4 5 6 7 8 9
204 1 8 2 1 2 3 0 4 0
0 220 1 0 0 6 0 0 0 1
6 0 165 7 0 7 0 2 3 0
4 3 2 209 6 2 3 3 2 8
3 1 1 6 186 3 7 6 3 0
4 15 4 0 0 203 0 0 0 1
0 0 0 0 2 0 197 0 1 4
1 2 3 4 3 2 0 173 2 0
0 1 7 3 4 3 0 7 204 2
0 1 16 13 0 4 18 3 9 181
200 175 150 125 100 75 50 25 0

In [5]:
```
## Train KNN Classifier
mknn = KNeighborsClassifier(n_neighbors=5)
mknn.fit(X_train, y_train)

## Make predictions on test data
mknn_predictions = mknn.predict(X_test)

## Randomly choose two columns
column_indices = random.sample(range(X_test.shape[1]), 2)
x_column_index, y_column_index = column_indices

## Visualize KNN clustering from random columns chosen
plt.figure(figsize=(10, 8))
sns.scatterplot(x=X_test.iloc[:, x_column_index], y=X_test.iloc[:, y_column_index], hue=mknn_predictions, pale
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('KNN Clustering Plot (Random Columns)')
plt.show()
```


KNN Clustering Plot (Random Columns)
hiphop
disco
metal
classical
blues
country
pop
reggae
jazz
rock
0.01
0.00
−0.01
−0.02
Actual
−20
−10
0
10
20
30
Predicted

In [6]:
```
## Flatten confusion matrix to create scatterplot
conf_matrix_flat = conf_matrix.flatten()

## Create x and y coordinates for scatterplot based on confusion matrix
x = np.repeat(np.arange(conf_matrix.shape[0]), conf_matrix.shape[1])
y = np.tile(np.arange(conf_matrix.shape[1]), conf_matrix.shape[0])

## Plot scatterplot
plt.figure(figsize=(8, 6))
plt.scatter(x, y, s=conf_matrix_flat*10, c=conf_matrix_flat, cmap='viridis', alpha=0.7)
plt.colorbar()
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Scatterplot based on Confusion Matrix')
plt.grid(True)
plt.show()
```

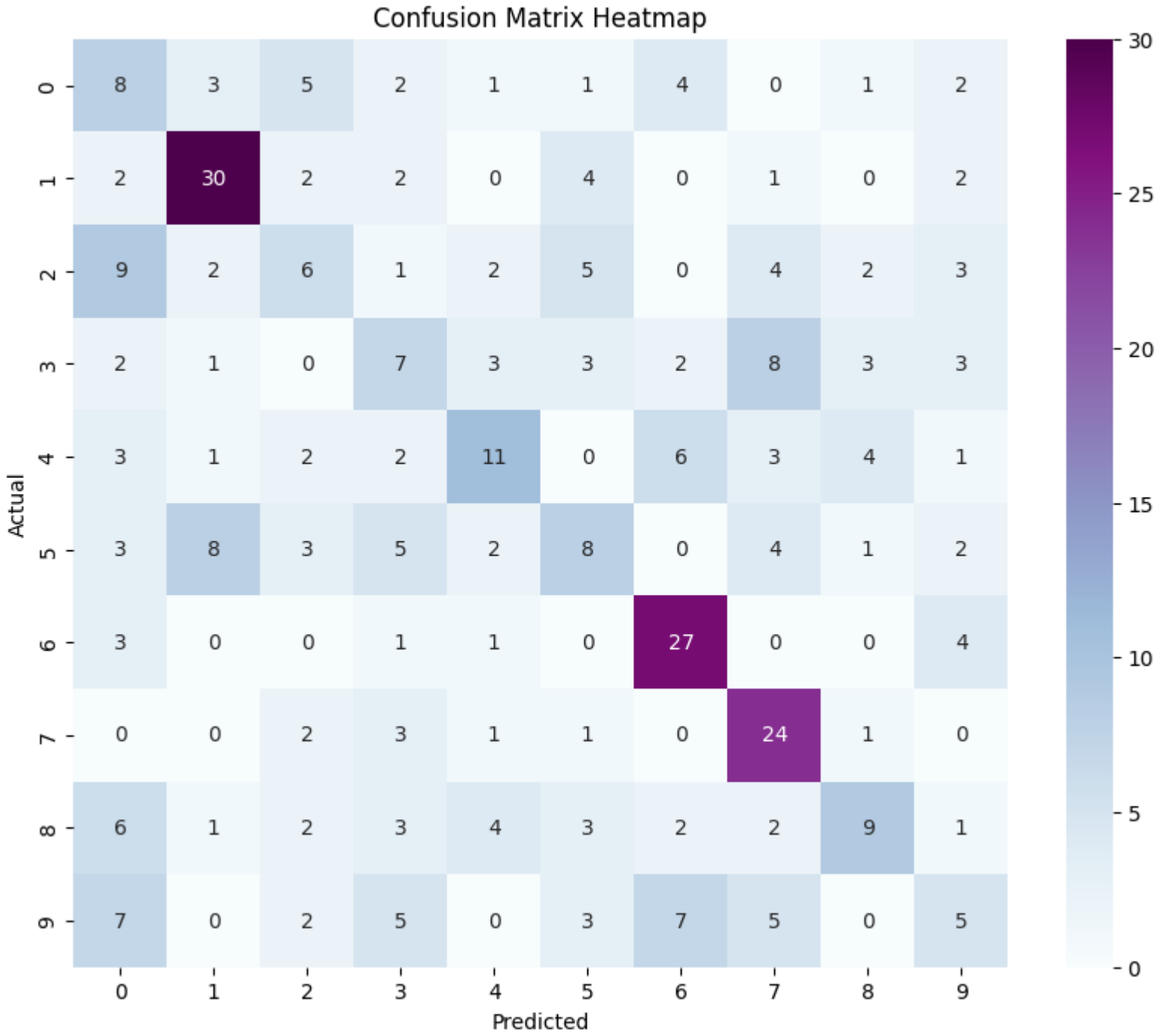

Scatterplot based on Confusion Matrix
Actual
Predicted
0
2
4
6
8
200
175
150
125
100
75
50
25
0

In [7]:
```
## Define function to extract features from an audio file
def extract_features(file_path):
    y, sr = librosa.load(file_path)
    mfcc_feat = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
    covariance = np.cov(mfcc_feat)
    mean_matrix = np.mean(mfcc_feat, axis=1)
    return mean_matrix, covariance

## Define function to load audio data and extract features
def load_and_extract_features(directory, split_ratio=0.66):
    dataset = []

    for folder in os.listdir(directory):
        folder_path = os.path.join(directory, folder)
        if os.path.isdir(folder_path):
            for file in os.listdir(folder_path):
                file_path = os.path.join(folder_path, file)
                if file.endswith(".wav"):
                    try:
                        (rate, sig) = wav.read(file_path)
                    except Exception as e: ## Shows any .wav file with error
                        print(f"Error reading {file_path}: {e}")
                        continue

                    ## Extract MFCC features
                    mfcc_feat = mfcc(sig, rate, winlen=0.020, appendEnergy=False)

                    ## Ensure all feature vectors have same dimensions
                    max_dim = 13
                    if mfcc_feat.shape[0] < max_dim:
                        ## Pad with zeros to make dimensions consistent
                        padding = np.zeros((max_dim - mfcc_feat.shape[0], mfcc_feat.shape[1]))
                        mfcc_feat = np.vstack((mfcc_feat, padding))
                    elif mfcc_feat.shape[0] > max_dim:
                        ## Truncate to make dimensions consistent
                        mfcc_feat = mfcc_feat[:max_dim, :]

                    covariance = np.cov(np.transpose(mfcc_feat))
                    mean_matrix = mfcc_feat.mean(0)
                    feature = (mean_matrix, covariance, folder)
                    dataset.append(feature)

    ## Shuffle dataset
```

```
    random.shuffle(dataset)

    ## Split dataset into training and testing sets
    split_index = int(split_ratio * len(dataset))
    train_set = dataset[:split_index]
    test_set = dataset[split_index:]

    X_train = np.array([data[0] for data in train_set])
    y_train = np.array([data[2] for data in train_set])
    X_test = np.array([data[0] for data in test_set])
    y_test = np.array([data[2] for data in test_set])

    return X_train, X_test, y_train, y_test

## Define function to train and evaluate a model
def train_and_evaluate_model(X_train, X_test, y_train, y_test):
    ## Standardize features
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    ## Train a Random Forest classifier
    clf = RandomForestClassifier(random_state=42)
    clf.fit(X_train, y_train)

    ## Evaluate model
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    conf_matrix = confusion_matrix(y_test, y_pred)

    return accuracy, conf_matrix

## Main part of script
if __name__ == "__main__":
    ## Set directory path where .wav files are located
    directory = "U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_original"

    ## Load audio data and extract features
    X_train, X_test, y_train, y_test = load_and_extract_features(directory, split_ratio=0.66)

    ## Train and evaluate model
    accuracy, conf_matrix = train_and_evaluate_model(X_train, X_test, y_train, y_test)
```

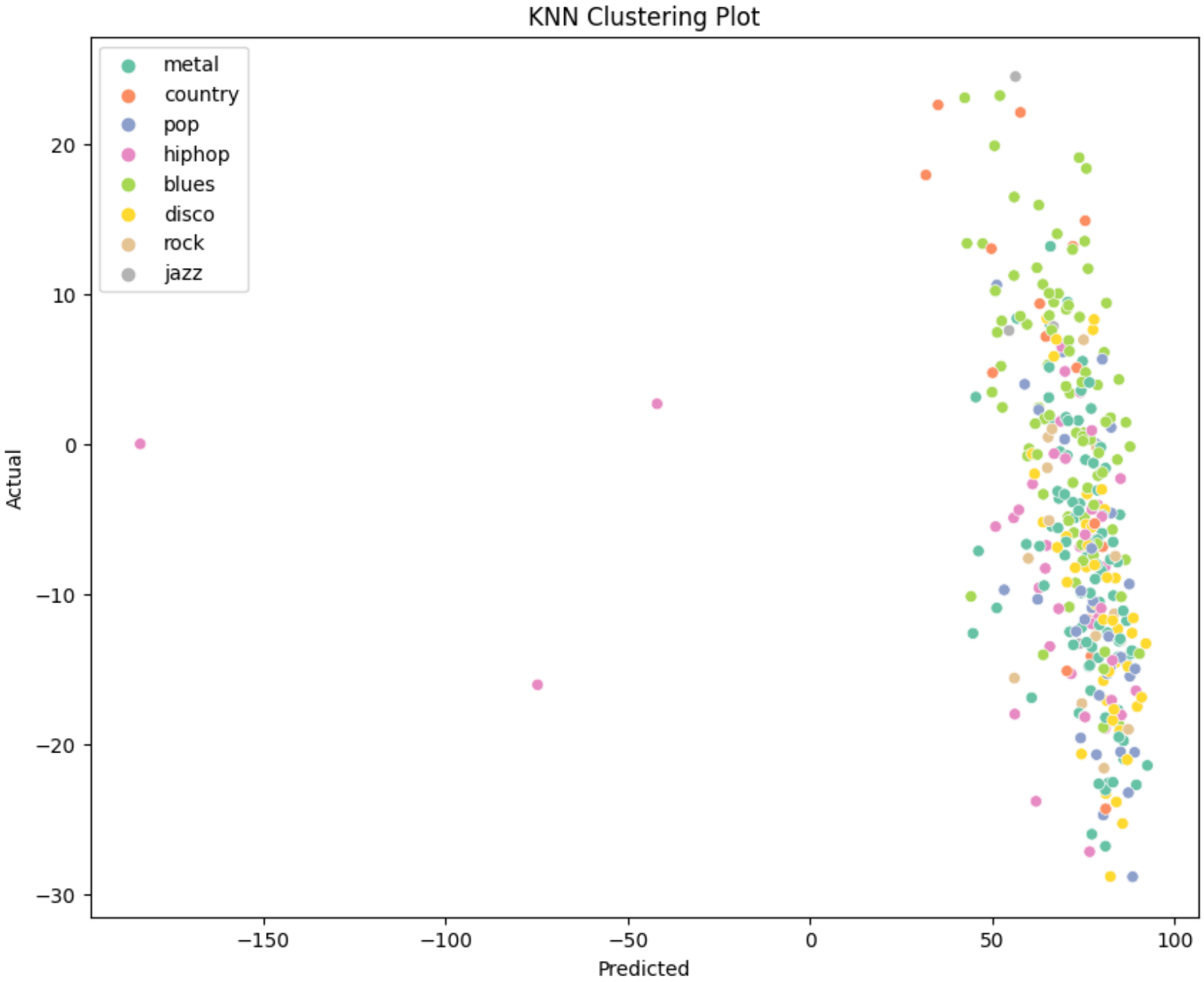
```
## Print accuracy and visualize confusion matrix
print(f"Accuracy: {accuracy:.2f}")

plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='BuPu')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix Heatmap')
plt.show()
```

```
Error reading U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_original\jazz\jazz.00
054.wav: File format b'\xcb\x15\x1e\x16' not understood. Only 'RIFF' and 'RIFX' supported.
Accuracy: 0.40
```


Confusion Matrix Heatmap
Actual
Predicted
0 1 2 3 4 5 6 7 8 9
8 3 5 2 1 1 4 0 1 2
2 30 2 2 0 4 0 1 0 2
9 2 6 1 2 5 0 4 2 3
2 1 0 7 3 3 2 8 3 3
3 1 2 2 11 0 6 3 4 1
3 8 3 5 2 8 0 4 1 2
3 0 0 1 1 0 27 0 0 4
0 0 2 3 1 1 0 24 1 0
6 1 2 3 4 3 2 2 9 1
7 0 2 5 0 3 7 5 0 5
30 25 20 15 10 5 0

In [8]:
```
## Define function to train and evaluate a KNN classifier
def train_and_evaluate_knn(X_train, X_test, y_train, y_test, n_neighbors=5):
    ## Standardize features using Scaler
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    ## Train KNN classifier
    knn = KNeighborsClassifier(n_neighbors=n_neighbors)
    knn.fit(X_train, y_train)

    ## Evaluate model
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    return knn, accuracy

## Main part of script
if __name__ == "__main__":
    ## Set directory path where .wav files are located
    directory = "U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_original"

    ## Load audio data and extract features
    X_train, X_test, y_train, y_test = load_and_extract_features(directory, split_ratio=0.66)

    ## Train and evaluate model
    knn, accuracy = train_and_evaluate_knn(X_train, X_test, y_train, y_test)

    # Print accuracy
    print(f"KNN Accuracy: {accuracy:.2f}")

    # Visualize KNN clustering
    knn_predictions = knn.predict(X_test)
    plt.figure(figsize=(10, 8))
    sns.scatterplot(x=X_test[:, 0], y=X_test[:, 1], hue=knn_predictions, palette="Set2")
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('KNN Clustering Plot')
    plt.show()
```

```
Error reading U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_original\jazz\jazz.00
054.wav: File format b'\xcb\x15\x1e\x16' not understood. Only 'RIFF' and 'RIFX' supported.
KNN Accuracy: 0.31
```


KNN Clustering Plot
metal
country
pop
hiphop
blues
disco
rock
jazz
Actual
Predicted
20
10
0
−10
−20
−30
−150
−100
−50
0
50
100

In [9]:
```
## Add code to classify a random audio sample from the genres_original folder
results = defaultdict(int)
i = 1
for folder in os.listdir(directory):
    results[i] = folder
    i += 1

## List all .wav files in the genres_original folder
all_wav_files = []
for folder in os.listdir(directory):
    folder_path = os.path.join(directory, folder)
    if os.path.isdir(folder_path):
        for file in os.listdir(folder_path):
            if file.endswith(".wav"):
                all_wav_files.append(os.path.join(folder_path, file))

## Choose 10 random .wav file for classification
for _ in range(10):
    random_audio_file = random.choice(all_wav_files)

       ## extract genre name from file path
    genre_name = os.path.basename(os.path.dirname(random_audio_file))

    ## print result
    print("Randomly selected audio file:", random_audio_file)
    print("Classified genre:", genre_name)
```

```
Randomly selected audio file: U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_origi
nal\metal\metal.00009.wav
Classified genre: metal
Randomly selected audio file: U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_origi
nal\country\country.00025.wav
Classified genre: country
Randomly selected audio file: U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_origi
nal\disco\disco.00015.wav
Classified genre: disco
Randomly selected audio file: U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_origi
nal\metal\metal.00094.wav
Classified genre: metal
Randomly selected audio file: U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_origi
nal\blues\blues.00067.wav
Classified genre: blues
Randomly selected audio file: U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_origi
nal\disco\disco.00006.wav
Classified genre: disco
Randomly selected audio file: U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_origi
nal\hiphop\hiphop.00051.wav
Classified genre: hiphop
Randomly selected audio file: U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_origi
nal\pop\pop.00003.wav
Classified genre: pop
Randomly selected audio file: U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_origi
nal\classical\classical.00055.wav
Classified genre: classical
Randomly selected audio file: U:/School Homework/Fall 2023/DSC680-Applied Data Science/Project1/genres_origi
nal\country\country.00074.wav
Classified genre: country
```