Project#1 report

111550116 王紹安

GitHub Link: https://github.com/Seanwang0116/AI-Capstone/tree/main/Project%201

1 Introduction

In this project, I collect songs from YouTube and use MFCC for genre classification. In supervised learning, I evaluate the impact of training set size, class weighting, and PCA for dimensionality reduction on model performance. Additionally, I explore unsupervised learning by varying the number of clusters and analyzing changes in the Adjusted Rand Score to assess clustering quality.

2 Dataset

2-1 Data source [1][2][3][4]

These songs come from various playlists published on YouTube.

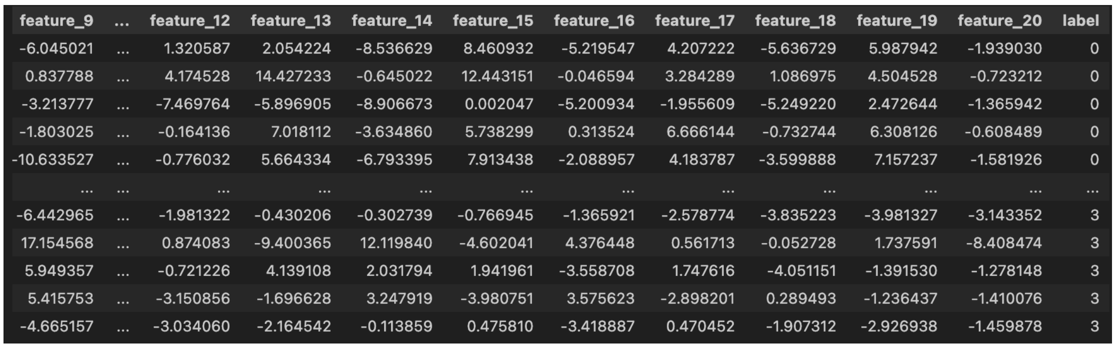
- Jazz: Top 100+ Jazz Classics Playlist | Best Jazz Music of All Time
- Rap: Top 500 Most Viewed Rap Songs Youtube
- Rock: Top 500 Classic Rock songs
- Country: Best Country Mix 500+ songs

2-2 Data preprocess

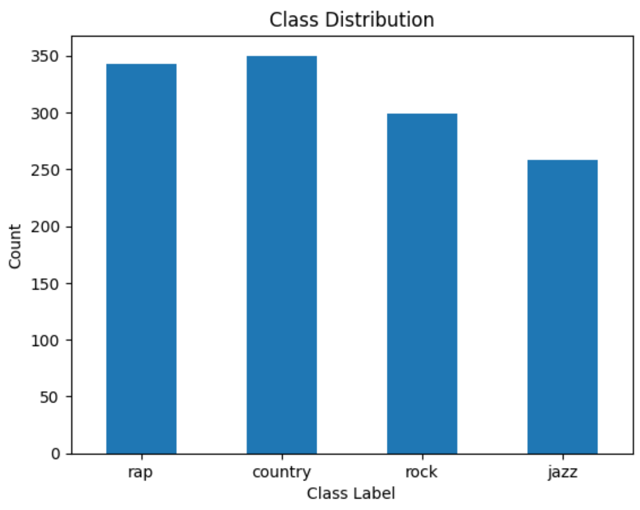
First, I download songs from YouTube by entering command to convert these songs from video to mp3:

yt-dlp -x --audio-format mp3 "*playlist link*"

Then, I use MFCC to extract 20 feature and show the count of each genre with labels: Rap (0), Country (1), Rock (2), and Jazz (3). Below are overview of the dataset and the counts for each genre:



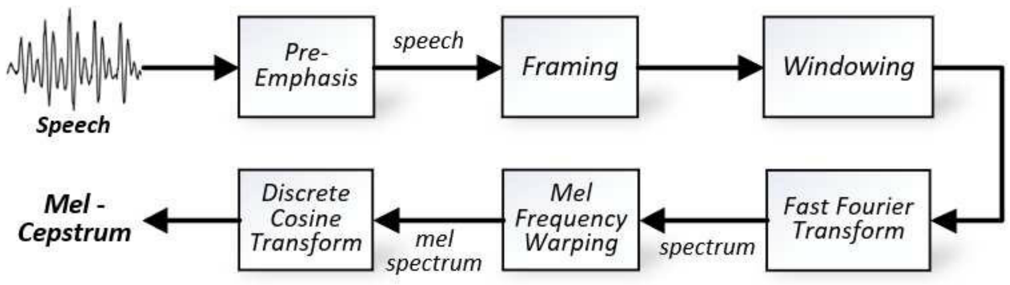
feature_9	...	feature_12	feature_13	feature_14	feature_15	feature_16	feature_17	feature_18	feature_19	feature_20	label
-6.045021	...	1.320587	2.054224	-8.536629	8.460932	-5.219547	4.207222	-5.636729	5.987942	-1.939030	0
0.837788	...	4.174528	14.427233	-0.645022	12.443151	-0.046594	3.284289	1.086975	4.504528	-0.723212	0
-3.213777	...	-7.469764	-5.896905	-8.906673	0.002047	-5.200934	-1.955609	-5.249220	2.472644	-1.365942	0
-1.803025	...	-0.164136	7.018112	-3.634860	5.738299	0.313524	6.666144	-0.732744	6.308126	-0.608489	0
-10.633527	...	-0.776032	5.664334	-6.793395	7.913438	-2.088957	4.183787	-3.599888	7.157237	-1.581926	0
...	...	...	...	...	...	...	...	...	...	...	...
-6.442965	...	-1.981322	-0.430206	-0.302739	-0.766945	-1.365921	-2.578774	-3.835223	-3.981327	-3.143352	3
17.154568	...	0.874083	-9.400365	12.119840	-4.602041	4.376448	0.561713	-0.052728	1.737591	-8.408474	3
5.949357	...	-0.721226	4.139108	2.031794	1.941961	-3.558708	1.747616	-4.051151	-1.391530	-1.278148	3
5.415753	...	-3.150856	-1.696628	3.247919	-3.980751	3.575623	-2.898201	0.289493	-1.236437	-1.410076	3
-4.665157	...	-3.034060	-2.164542	-0.113859	0.475810	-3.418887	0.470452	-1.907312	-2.926938	-1.459878	3



3 Methods

3-1 Mel-frequency cepstral coefficients [5]

Mel Frequency Cepstral Coefficients (MFCC) refer to a set of features to analyze seismic audio echoes and model human voice characteristics. They are essential sound characteristics used in various applications, obtained by taking the discrete Fourier transform of a signal, applying a logarithm, and then taking a Fourier inverse.



3-2 Principal Component Analysis [6]

Principal component analysis, or PCA, is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and thus make analyzing data points much easier and faster for machine learning algorithms without extraneous variables to process.

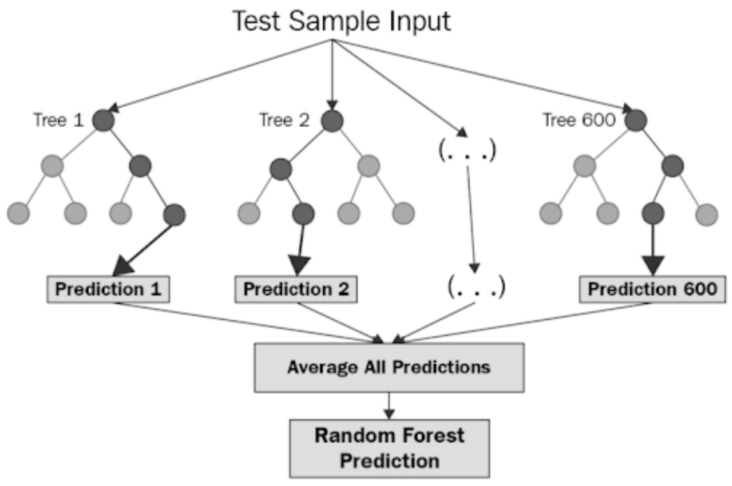
3-3 Supervised Method: Random Forest [7]

Random forest operates by constructing a multitude of decision trees at training time and outputting the class that’s the mode of the classes (classification) or mean prediction (regression) of the individual trees.

A Random Forest is a meta-estimator that combines multiple decision trees with key modifications:

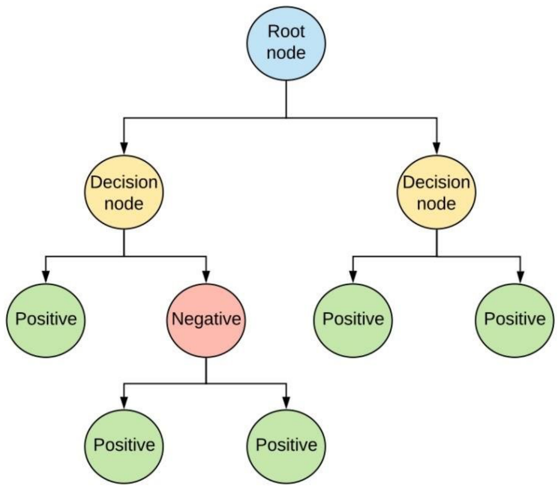
(1) At each node, only a subset of features (defined by a hyper-parameter) is considered for splitting, preventing reliance on any single feature and promoting diversity.

(2) Each tree is trained on a random sample of the data, adding randomness and reducing overfitting.



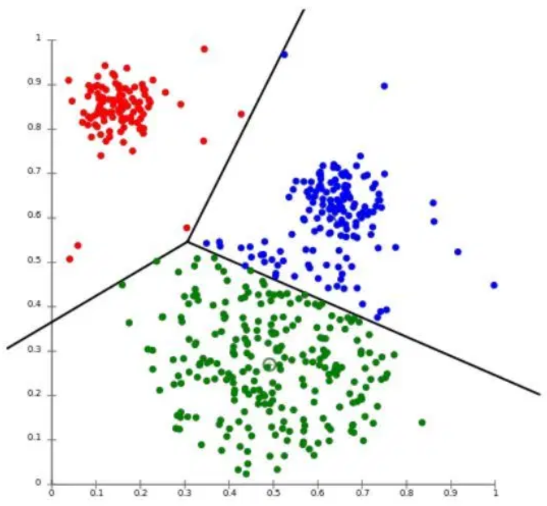
3-4 Supervised Method: Decision Tree [8]

Decision tree is a commonly used methodology for performing classification tasks. It is a tree-based supervised machine learning algorithm that is used to classify or make predictions in a path of how previous questions are answered. Generally, the decision tree algorithm categorizes data into branch-like segments that develop into a tree that contains a root, nodes, and leaves.



3-5 Unsupervised Method: K-Means Clustering [9]

K-means clustering is an unsupervised machine learning approach that uses similarities to divide a dataset into K different clusters. The algorithm aims to minimize the within-cluster variance, ensuring that data points within the same cluster are as similar as possible while data points in different clusters are as dissimilar as possible.



4 Implementation Details

4-1 Evaluation metrics

4-1-1 Supervised Methods

Accuracy	$\frac{TP+TN}{(TP+TN+FP+FN)}$
Precision	$\frac{TP}{(TP+FP)}$
Recall	$\frac{TP}{(TP+FN)}$
F1-score	$2\times\frac{precision\times recall}{(precision+recall)}$

Where TP represents True Positive, TN presents True Negative, FP presents False Positive and FN presents False Negative.

4-1-2 Unsupervised Methods [10]

Adjusted rand score	$ARI=\frac{RI-expected\_RI}{(\max(RI)-expected\_RI)}$

The Rand Index computes a similarity measure between two clustering by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clustering.

4-2 Hyperparameters

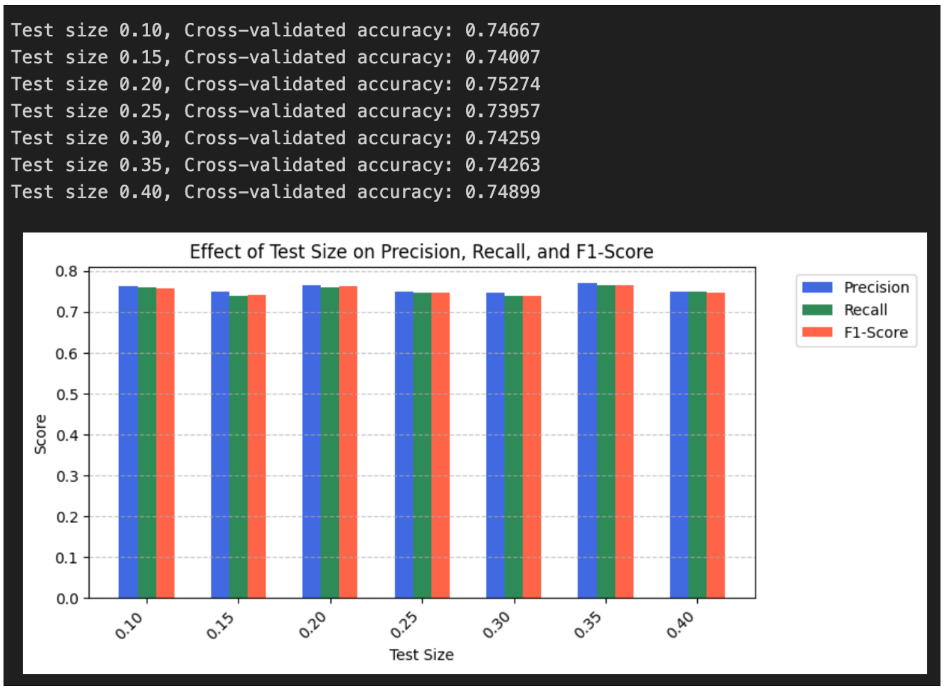
The random seed is set to 777 to make the results reproducible and consistency. The estimator in random forest is 120, the maximum depth in decision tree is 10 and in K means clustering, K is 4. When implying PCA, the dimension is set to 10. Then size of training set in supervised learning ranges from 0.1 to 0.4 and the folds in cross-validation is 5.

5 Experiments

5-1 Supervised learning

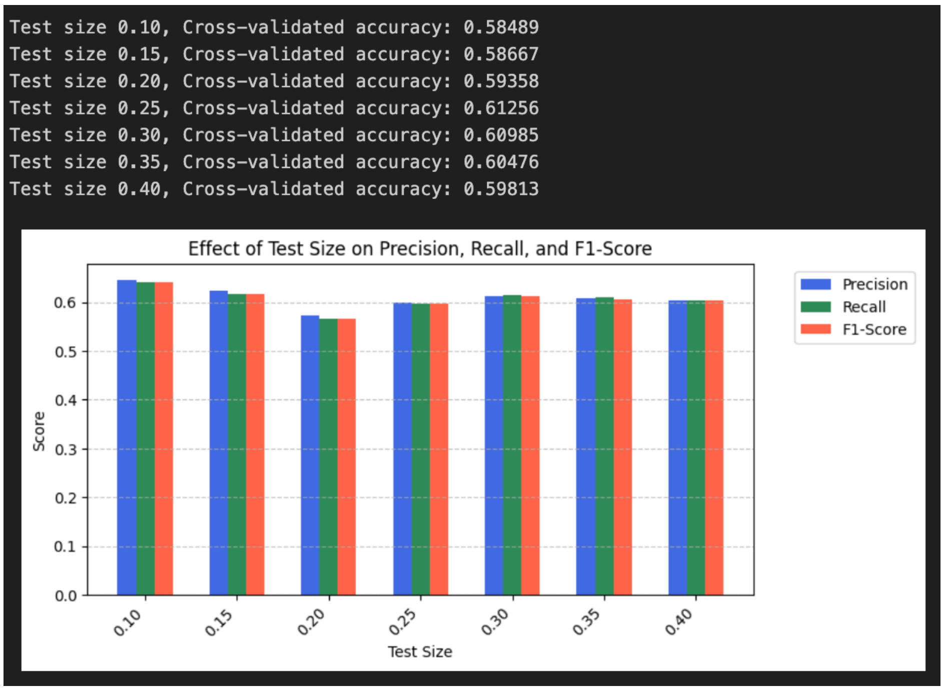
5-1-1 Test Dataset Size

I experiment with different test dataset sizes, ranging from 0.1 to 0.4, to analyze their impact on the model's performance. By evaluating metrics such as precision, recall, and F1-score, I aim to determine how varying the test dataset size influences the overall effectiveness and stability of the model.



```
Test size 0.10, Cross-validated accuracy: 0.74667
Test size 0.15, Cross-validated accuracy: 0.74007
Test size 0.20, Cross-validated accuracy: 0.75274
Test size 0.25, Cross-validated accuracy: 0.73957
Test size 0.30, Cross-validated accuracy: 0.74259
Test size 0.35, Cross-validated accuracy: 0.74263
Test size 0.40, Cross-validated accuracy: 0.74899
```

Figure 1: Cross-validation accuracy, precision, recall, and F1-score across different test dataset sizes using the Random Forest classifier



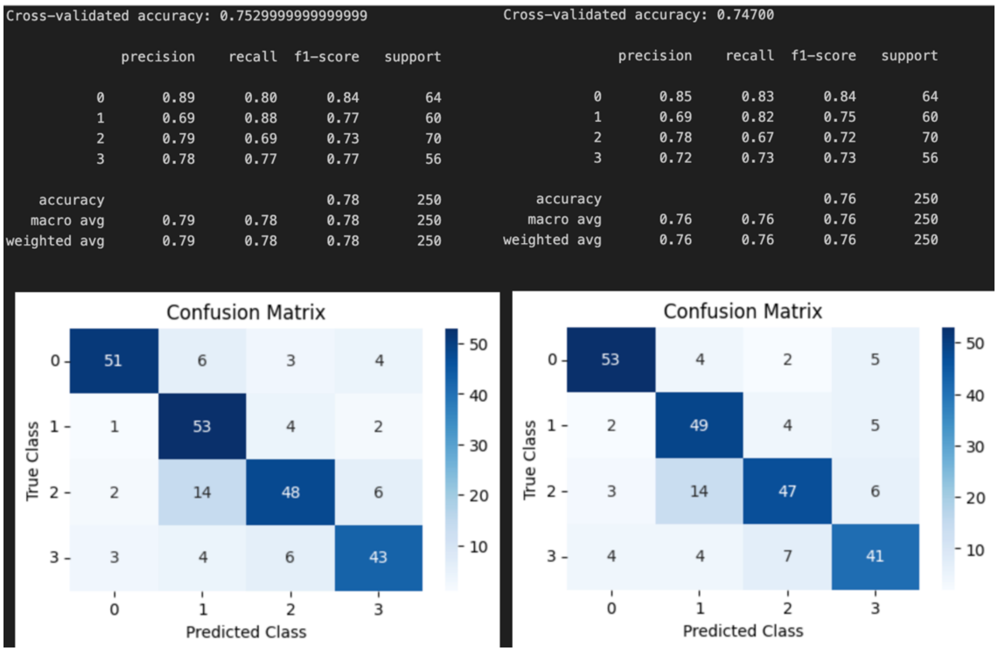
```
Test size 0.10, Cross-validated accuracy: 0.58489
Test size 0.15, Cross-validated accuracy: 0.58667
Test size 0.20, Cross-validated accuracy: 0.59358
Test size 0.25, Cross-validated accuracy: 0.61256
Test size 0.30, Cross-validated accuracy: 0.60985
Test size 0.35, Cross-validated accuracy: 0.60476
Test size 0.40, Cross-validated accuracy: 0.59813
```

Figure 2: Cross-validation accuracy, precision, recall, and F1-score across different test dataset sizes using the Decision Tree classifier

As shown in the results above, I can conclude that the size of the test dataset does not significantly affect the performance of the model. The cross-validation accuracy, precision, recall, and F1-score remain stable across the different test dataset sizes.

5-1-2 Sample weighting

I experiment with sample weighting, using a test dataset size of 0.2 to evaluate the model's performance, I utilize cross-validation along with a confusion matrix. By analyzing metrics such as precision, recall, and F1-score, I aim to assess how sample weighting impacts the overall effectiveness and stability of the model.



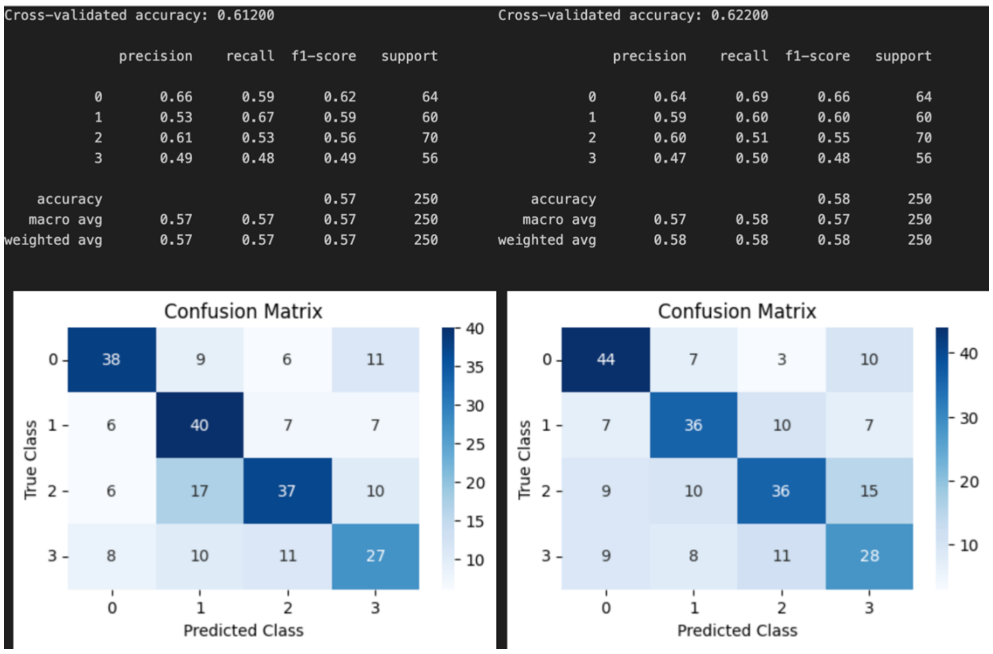
```
Cross-validated accuracy: 0.7529999999999999

              precision    recall  f1-score   support

           0       0.89      0.80      0.84        64
           1       0.69      0.88      0.77        60
           2       0.79      0.69      0.73        70
           3       0.78      0.77      0.77        56

    accuracy                           0.78       250
   macro avg       0.79      0.78      0.78       250
weighted avg       0.79      0.78      0.78       250
```

```
Cross-validated accuracy: 0.74700

              precision    recall  f1-score   support

           0       0.85      0.83      0.84        64
           1       0.69      0.82      0.75        60
           2       0.78      0.67      0.72        70
           3       0.72      0.73      0.73        56

    accuracy                           0.76       250
   macro avg       0.76      0.76      0.76       250
weighted avg       0.76      0.76      0.76       250
```

True Class \ Predicted Class	0	1	2	3
0	51	6	3	4
1	1	53	4	2
2	2	14	48	6
3	3	4	6	43

True Class \ Predicted Class	0	1	2	3
0	53	4	2	5
1	2	49	4	5
2	3	14	47	6
3	4	4	7	41

Figure 3: Classification report and confusion matrix comparing the performance without(left) and with(right) sample weighting using the Random Forest classifier



```
Cross-validated accuracy: 0.61200

              precision    recall  f1-score   support

           0       0.66      0.59      0.62        64
           1       0.53      0.67      0.59        60
           2       0.61      0.53      0.56        70
           3       0.49      0.48      0.49        56

    accuracy                           0.57       250
   macro avg       0.57      0.57      0.57       250
weighted avg       0.57      0.57      0.57       250
```

```
Cross-validated accuracy: 0.62200

              precision    recall  f1-score   support

           0       0.64      0.69      0.66        64
           1       0.59      0.60      0.60        60
           2       0.60      0.51      0.55        70
           3       0.47      0.50      0.48        56

    accuracy                           0.58       250
   macro avg       0.57      0.58      0.57       250
weighted avg       0.58      0.58      0.58       250
```

True Class \ Predicted Class	0	1	2	3
0	38	9	6	11
1	6	40	7	7
2	6	17	37	10
3	8	10	11	27

True Class \ Predicted Class	0	1	2	3
0	44	7	3	10
1	7	36	10	7
2	9	10	36	15
3	9	8	11	28

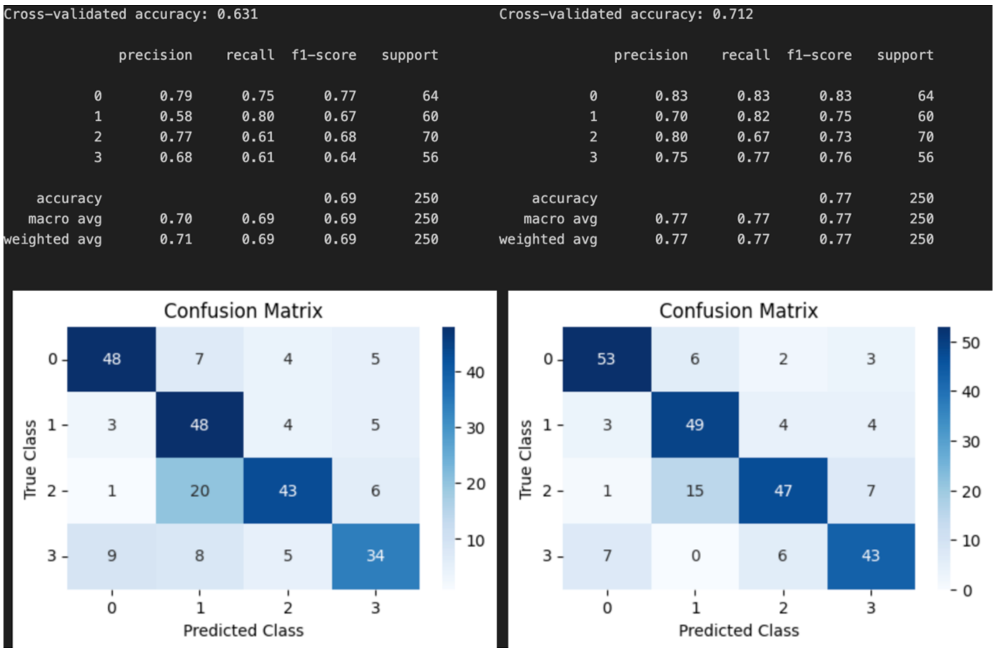
Figure 4: Classification report and confusion matrix comparing the performance without(left) and

with(right) sample weighting using the Decision Tree classifier

As shown in the results above, I can conclude that after applying sample weighting, the performance of the Random Forest classifier slightly decreased compared to the original model. However, the Decision Tree classifier showed improved performance after applying sample weighting, outperforming the original results.

5-1-3 Principal Component Analysis

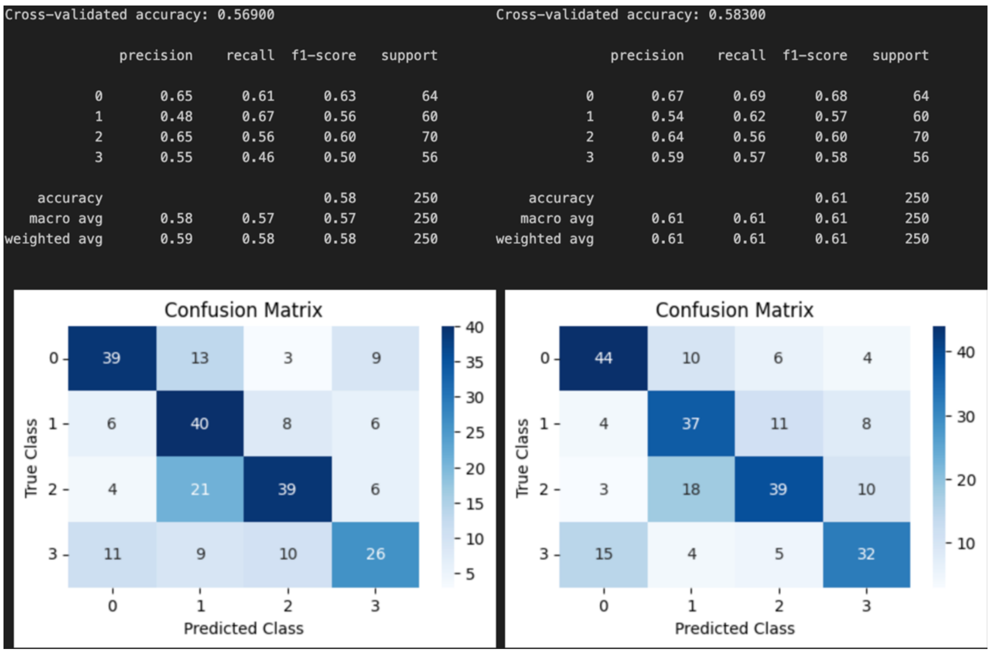
I experiment with PCA dimensionality reduction, reducing the dataset to 5 and 10 dimensions, while keeping other settings same as 5-1-2.



```
Cross-validated accuracy: 0.631

              precision    recall  f1-score   support

           0       0.79      0.75      0.77        64
           1       0.58      0.80      0.67        60
           2       0.77      0.61      0.68        70
           3       0.68      0.61      0.64        56

    accuracy                           0.69       250
   macro avg       0.70      0.69      0.69       250
weighted avg       0.71      0.69      0.69       250
```

```
Cross-validated accuracy: 0.712

              precision    recall  f1-score   support

           0       0.83      0.83      0.83        64
           1       0.70      0.82      0.75        60
           2       0.80      0.67      0.73        70
           3       0.75      0.77      0.76        56

    accuracy                           0.77       250
   macro avg       0.77      0.77      0.77       250
weighted avg       0.77      0.77      0.77       250
```

Figure 5: Classification report and confusion matrix comparing performance with 5 (left) and 10 (right) dimensions using the Random Forest classifier



```
Cross-validated accuracy: 0.56900

              precision    recall  f1-score   support

           0       0.65      0.61      0.63        64
           1       0.48      0.67      0.56        60
           2       0.65      0.56      0.60        70
           3       0.55      0.46      0.50        56

    accuracy                           0.58       250
   macro avg       0.58      0.57      0.57       250
weighted avg       0.59      0.58      0.58       250
```

```
Cross-validated accuracy: 0.58300

              precision    recall  f1-score   support

           0       0.67      0.69      0.68        64
           1       0.54      0.62      0.57        60
           2       0.64      0.56      0.60        70
           3       0.59      0.57      0.58        56

    accuracy                           0.61       250
   macro avg       0.61      0.61      0.61       250
weighted avg       0.61      0.61      0.61       250
```

Figure 6: Classification report and confusion matrix comparing performance with 5 (left) and 10 (right) dimensions using the Decision Tree classifier

As shown in the results above, I can conclude that the performance of the model is significantly improved when reducing the dimensions to 10 using PCA, compared to reducing it to 5. The cross-validation accuracy, precision, recall, and F1-score are all higher with 10 dimensions, indicating better overall performance and stability of the model after applying PCA with 10 components.

5-2 Unsupervised learning

I experiment with different values of k ranging from 2 to 10 to analyze their impact on the performance of the K-Means clustering. By evaluating the Adjusted Rand Score, I aim to determine how varying the number of clusters influences the clustering effectiveness and stability of the model.

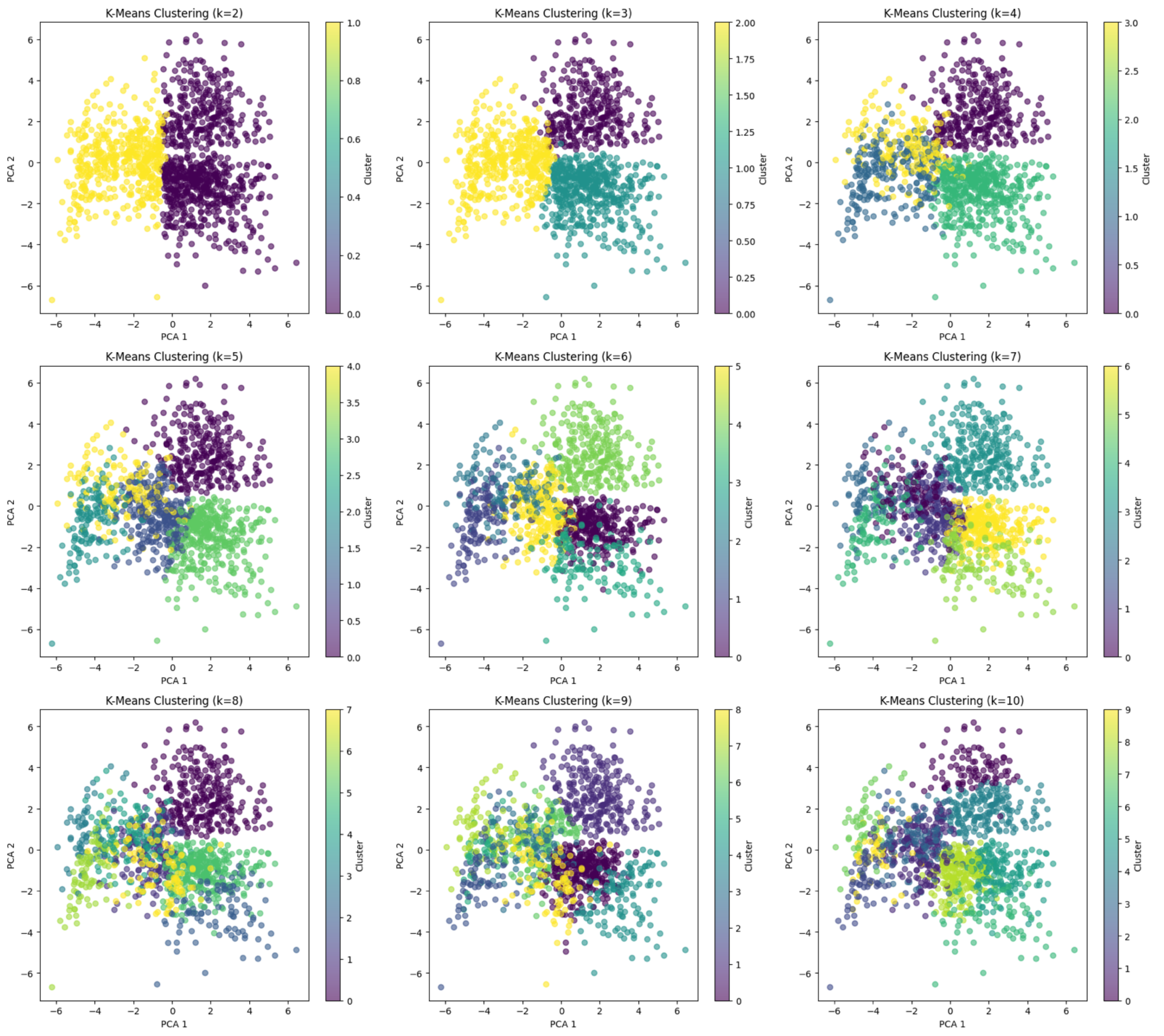


Figure 7: Scatter plot of k ranging from 2 to 10, showing the variation in clustering performance across different numbers of clusters

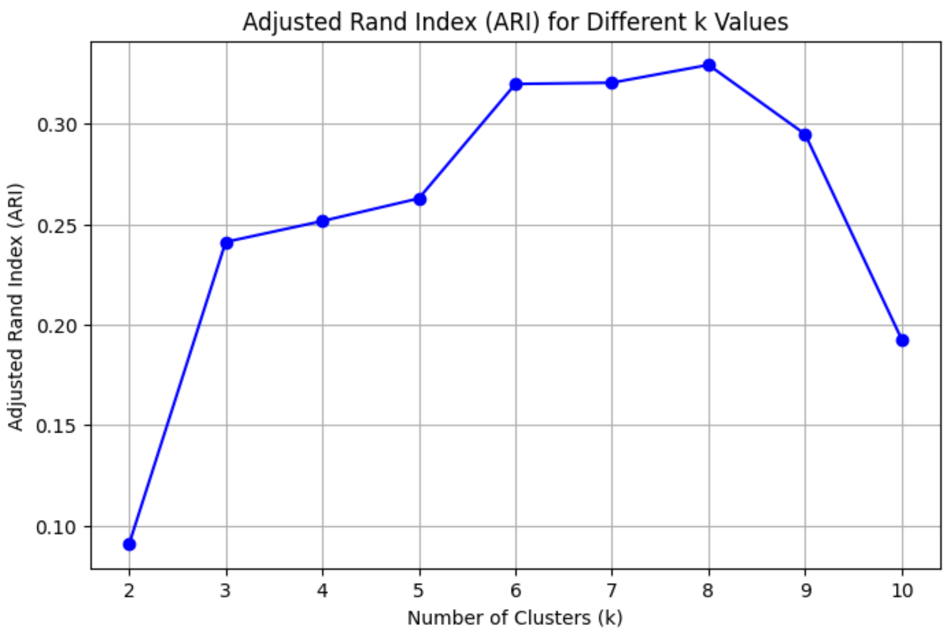


Figure 8: Adjusted Rand Score across different numbers of clusters

As shown in the results above, I can conclude that the performance is best at k = 6, 7, and 8, but it deteriorates for values beyond that.

6 Discussion

6-1 Experiment results

In supervised learning, it is surprising that the size of the test dataset does not significantly impact performance. This could be due to the relatively small size of the overall dataset, which may not reveal noticeable differences. Similarly, class weighting appears to have little effect. However, the PCA experiment was successful, as theory suggests that PCA should not influence model performance.

In unsupervised learning, after experimenting with various values of k, the results indicate that k = 8 yields the best performance for this project.

6-2 Factors that affect performance

As mentioned in the previous section, the overall size of the dataset is too small, making it difficult to train the classifier effectively, especially in supervised learning. The training set ends up being too small due to the test dataset size, which hinders the model's ability to generalize. Additionally, I intentionally created an imbalanced distribution in the dataset (with fewer jazz songs) to observe the effect of class imbalance. However, no significant results were observed. The difficulty remains in the unsupervised learning task as well, where clustering proves challenging.

6-3 Further experiments if available

If I have more time to improve this project, I will seek to expand the dataset by collecting more music samples or applying data augmentation techniques to enhance the model’s stability and accuracy, because the current dataset is too small, which limits the model's performance during training. Besides, I would perform more detailed hyperparameter tuning, such as adjusting the number of trees, max depth, and minimum samples required to split a node in the Random Forest model, to improve its generalization on a small dataset. Last, I will clean the raw data further by removing unnecessary noise and apply feature selection methods to improve the model's performance.

6-4 Valuable insight from experiments

Through this project, I learned several important lessons. I realize a small dataset limits the classifier's ability to learn effectively. This taught me the importance of expanding the dataset to improve stability and accuracy. I also realized that class imbalance may not show a significant effect when the dataset is too small, although handling imbalance is still critical for larger datasets. The PCA experiment confirmed that PCA does not directly affect model performance, aligning with theoretical expectations. In unsupervised learning, experimenting with different k values for K-Means revealed that selecting the appropriate number of clusters is crucial. The importance of using diverse evaluation metrics like precision, recall, F1-score, and Average rand score became clear. Overall, this experiment enhanced my understanding of model evaluation, data processing, and algorithm selection, providing valuable experience for future works.

7 Reference

[1]https://www.youtube.com/watch?v=ZZcuSBouhVA&list=PL8F6B0753B2CCA128

[2]https://www.youtube.com/watch?v=RgKAFK5djSk&list=PLLf_1oiSU7VoEM8CUSl5z4SXiz7WAoiN6

[3]https://www.youtube.com/watch?v=fJ9rUzIMcZQ&list=PL0GvsLQil0MmYC96KEs_7dTNsLm1PS6JX

[4]https://www.youtube.com/watch?v=zXDAYlhdkyg&list=PLNLQuN_16YRiejNNdwmdTJuFygMp25zUM

[5]https://www.researchgate.net/figure/MFCC-feature-extraction-steps_fig2_354711774

[6]https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[7]https://builtin.com/data-science/random-forest-python

[8]https://www.researchgate.net/publication/370770672_Classification_of_Adult_Income_Using_Decision_Tree

[9] https://www.bombaysoftwares.com/blog/introduction-to-k-means-clustering

[10]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html

Appendix

preprocess

March 5, 2025

[ ]:
```
import os
import numpy as np
import librosa
import pandas as pd
```

[ ]:
```
n = 20
hop = 512
features = []
labels = []
```

[ ]:
```
folder_path = "./rap"
mp3_files = [f for f in os.listdir(folder_path) if f.endswith(".mp3")]

for mp3_file in mp3_files:
    mp3_file_path = os.path.join(folder_path, mp3_file)

    y, sr = librosa.load(mp3_file_path, sr=None, mono=True, dtype=np.float32)
    #      &
    y = librosa.effects.trim(y)[0]
    y = librosa.util.normalize(y)
    #   MFCC (n_mfcc, time_frames)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n, hop_length=hop)
    #   MFCC      (n_mfcc,)
    mfcc_mean = np.mean(mfcc, axis=1)
    #
    features.append(mfcc_mean)
    labels.append(0)

print(f"   {len(features)}   ")
```

[ ]:
```
folder_path = "./country"
mp3_files = [f for f in os.listdir(folder_path) if f.endswith(".mp3")]

for mp3_file in mp3_files:
    mp3_file_path = os.path.join(folder_path, mp3_file)

    y, sr = librosa.load(mp3_file_path, sr=None, mono=True, dtype=np.float32)
    y = librosa.effects.trim(y)[0]
    y = librosa.util.normalize(y)

    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n, hop_length=hop)

    mfcc_mean = np.mean(mfcc, axis=1)

    features.append(mfcc_mean)
    labels.append(1)

print(f"   {len(features)}   ")
```

[ ]:
```
folder_path = "./rock"
mp3_files = [f for f in os.listdir(folder_path) if f.endswith(".mp3")]

for mp3_file in mp3_files:
    mp3_file_path = os.path.join(folder_path, mp3_file)

    y, sr = librosa.load(mp3_file_path, sr=None, mono=True, dtype=np.float32)

    y = librosa.effects.trim(y)[0]
    y = librosa.util.normalize(y)

    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n, hop_length=hop)

    mfcc_mean = np.mean(mfcc, axis=1)

    features.append(mfcc_mean)
    labels.append(2)

print(f"   {len(features)}   ")
```

[ ]:
```
folder_path = "./jazz"
mp3_files = [f for f in os.listdir(folder_path) if f.endswith(".mp3")]

for mp3_file in mp3_files:
    mp3_file_path = os.path.join(folder_path, mp3_file)

    y, sr = librosa.load(mp3_file_path, sr=None, mono=True, dtype=np.float32)

    y = librosa.effects.trim(y)[0]
    y = librosa.util.normalize(y)

    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n, hop_length=hop)

    mfcc_mean = np.mean(mfcc, axis=1)
    features.append(mfcc_mean)
    labels.append(3)

print(f"    {len(features)}    ")
```

[ ]:
```
features = np.array(features)
labels = np.array(labels)

song_ids = np.arange(1, features.shape[0] + 1)

df = pd.DataFrame(features, columns=[f"feature_{i+1}" for i in range(features.
  ↪shape[1])])
df.insert(0, "song_id", song_ids)
df["label"] = labels
df.to_csv("dataset.csv", index=False, encoding="utf-8")
```

[ ]:
```
dataset = pd.read_csv('dataset.csv')
dataset
```

main

March 5, 2025

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

random_seed = 777

#
PCA_or_not = False
PCA_n = 5
```

[ ]:
```
df = pd.read_csv('dataset.csv')
df.head()
```

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

[ ]:
```
features = df.drop(columns=["song_id", "label"])
labels = df["label"]

def plot(conf_matrix):
    plt.figure(figsize=(5, 3))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted Class')
    plt.yticks(rotation=0)
    plt.ylabel('True Class')
    plt.show()

def plot_size(results):
    x_labels = [f"{size:.2f}" for size in results['test_size']]
    x = np.arange(len(x_labels))

    width = 0.2
    fig, ax = plt.subplots(figsize=(8, 4))

    ax.bar(x - width, results['precision'], width=width, label="Precision",␣
 ↪color='royalblue')
    ax.bar(x, results['recall'], width=width, label="Recall", color='seagreen')
    ax.bar(x + width, results['f1-score'], width=width, label="F1-Score",␣
 ↪color='tomato')
    ax.set_xticks(x)
    ax.set_xticklabels(x_labels, rotation=45, ha="right")
    ax.set_xlabel("Test Size")
    ax.set_ylabel("Score")
    ax.set_title("Effect of Test Size on Precision, Recall, and F1-Score")
    ax.legend(loc="upper left", bbox_to_anchor=(1.05, 1))
    ax.grid(axis="y", linestyle="--", alpha=0.7)

    plt.show()
```

```
[ ]: if PCA_or_not:
         scaler = StandardScaler()
         scaled_features = scaler.fit_transform(features)
         pca = PCA(n_components=PCA_n)
         features = pca.fit_transform(scaled_features)
         df = pd.DataFrame(data=features, columns=[f'PC{i+1}' for i in range(PCA_n)])

     df.head()
```

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(features, labels,␣
      ↪test_size=0.2, random_state=random_seed)
     clf = RandomForestClassifier(n_estimators=120, random_state=random_seed)

     cv_accuracy = cross_val_score(clf, X_train, y_train, cv=5, scoring='accuracy')
     print(f'Cross-validated accuracy: {np.mean(cv_accuracy)}')

     clf.fit(X_train, y_train)
     y_pred = clf.predict(X_test)

     print(f'\n{classification_report(y_test, y_pred)}')

     conf_matrix = confusion_matrix(y_test, y_pred)
     plot(conf_matrix)
```

```
[ ]: clf_weighted = RandomForestClassifier(n_estimators=120,␣
      ↪class_weight="balanced", random_state=random_seed)

     cv_accuracy = cross_val_score(clf_weighted, X_train, y_train, cv=5,␣
      ↪scoring='accuracy')
     print(f'Cross-validated accuracy: {np.mean(cv_accuracy):.5f}')
```

```
clf_weighted.fit(X_train, y_train)
y_pred_weighted = clf_weighted.predict(X_test)

print(f'\n{classification_report(y_test, y_pred_weighted)}')

conf_matrix = confusion_matrix(y_test, y_pred_weighted)
plot(conf_matrix)
```

[ ]:
```
test_sizes = np.arange(0.1, 0.45, 0.05)
results = {'precision': [], 'recall': [], 'f1-score': [], 'test_size': []}

for test_size in test_sizes:
    X_train, X_test, y_train, y_test = train_test_split(features, labels,␣
 ↪test_size=test_size, random_state=random_seed)

    clf = RandomForestClassifier(n_estimators=120, random_state=random_seed)
    cv_accuracy = cross_val_score(clf_weighted, X_train, y_train, cv=5,␣
 ↪scoring='accuracy')
    print(f'Test size {test_size:.2f}, Cross-validated accuracy: {np.
 ↪mean(cv_accuracy):.5f}')

    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)

    report = classification_report(y_test, y_pred, output_dict=True)

    results['test_size'].append(test_size)
    results['precision'].append(report['weighted avg']['precision'])
    results['recall'].append(report['weighted avg']['recall'])
    results['f1-score'].append(report['weighted avg']['f1-score'])

plot_size(results)
```

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(features, labels,␣
 ↪test_size=0.2, random_state=random_seed)
clf = DecisionTreeClassifier(max_depth=10, random_state=random_seed)

cv_accuracy = cross_val_score(clf, X_train, y_train, cv=5, scoring='accuracy')
print(f'Cross-validated accuracy: {np.mean(cv_accuracy):.5f}')

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

print(f'\n{classification_report(y_test, y_pred)}')
```

```
conf_matrix = confusion_matrix(y_test, y_pred)
plot(conf_matrix)
```

[ ]:
```
clf_weighted = DecisionTreeClassifier(max_depth=10, class_weight="balanced",␣
 ↪random_state=random_seed)

cv_accuracy = cross_val_score(clf_weighted, X_train, y_train, cv=5,␣
 ↪scoring='accuracy')
print(f'Cross-validated accuracy: {np.mean(cv_accuracy):.5f}')

clf_weighted.fit(X_train, y_train)
y_pred_weighted = clf_weighted.predict(X_test)

print(f'\n{classification_report(y_test, y_pred_weighted)}')

conf_matrix = confusion_matrix(y_test, y_pred_weighted)
plot(conf_matrix)
```

[ ]:
```
test_sizes = np.arange(0.1, 0.45, 0.05)
results = {'precision': [], 'recall': [], 'f1-score': [], 'test_size': []}

for test_size in test_sizes:
    X_train, X_test, y_train, y_test = train_test_split(features, labels,␣
 ↪test_size=test_size, random_state=random_seed)

    clf = DecisionTreeClassifier(max_depth=10, random_state=random_seed)
    cv_accuracy = cross_val_score(clf_weighted, X_train, y_train, cv=5,␣
 ↪scoring='accuracy')
    print(f'Test size {test_size:.2f}, Cross-validated accuracy: {np.
 ↪mean(cv_accuracy):.5f}')

    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)

    report = classification_report(y_test, y_pred, output_dict=True)

    results['test_size'].append(test_size)
    results['precision'].append(report['weighted avg']['precision'])
    results['recall'].append(report['weighted avg']['recall'])
    results['f1-score'].append(report['weighted avg']['f1-score'])

plot_size(results)
```

[ ]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import adjusted_rand_score
```

[ ]:
```
X = df
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[ ]:
```
fig, axes = plt.subplots(3, 3, figsize=(18, 16))
axes = axes.flatten()

ari_scores = []

for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=random_seed, n_init=10)
    clusters = kmeans.fit_predict(X_scaled)

    df["cluster"] = clusters

    pca = PCA(n_components=2)
    X_pca = pca.fit_transform(X_scaled)

    scatter = axes[k-2].scatter(X_pca[:, 0], X_pca[:, 1], c=clusters,␣
 ↪cmap="viridis", alpha=0.6)
    axes[k-2].set_title(f"K-Means Clustering (k={k})")
    axes[k-2].set_xlabel("PCA 1")
    axes[k-2].set_ylabel("PCA 2")

    fig.colorbar(scatter, ax=axes[k-2], label="Cluster")

    ari = adjusted_rand_score(df["label"], clusters)
    ari_scores.append(ari)

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

[ ]:
```
plt.figure(figsize=(8, 5))
plt.plot(range(2, 11), ari_scores, marker='o', linestyle='-', color='b')
plt.title("Adjusted Rand Index (ARI) for Different k Values")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Adjusted Rand Index (ARI)")
plt.grid(True)
plt.show()
```