Palestinian Accents Recognition System

Joud Hijaz [1], *Christina Saba*[2],*Malak Nassar*[3]

[1] Birzeit University
[2] Birzeit University
[3] Birzeit University

1200342@student.birzeit.edu, 1201255@student.birzeit.edu ,1200757@student.birzeit.edu

10 June 2024

Abstract

This report is an approach for accent recognition in speech from different speakers distributed among different geographical locations (Hebron, Jerusalem, Ramallah Reef, and Nablus)in Palestine to automatically classify Palestinian Arabic accents in audio recordings and determine the origin of that speaker. We used the given recordings where each location had ten speakers with recordings ranging from 30 seconds to 9 minutes. Then the acoustic features are extracted including Mel-frequency cepstral coefficients (MFCCs), zero-crossing rate, energy, chroma, and spectral contrast. Then from the MFCC's features clusters information, K-Means clustering is utilized to capture. These features are then trained using RandomForestClassifier optimized using machine learning techniques, KNN classifier. Then performance is evaluated using the accuracy, visualization using confusion matrices and t-SNE, precision, recall, and F1-score. Then the tested recordings are predicted if that accent is the one or not with an accuracy of 75% for RandomForestClassifier and 60% for KNN. Thus, the trained model is applied to unlabeled test data to predict accents in unseen recordings.

Keywords: Accent Recognition, Palestinian Arabic Accents, RandomForestClassifier, K-means clusters, KNN, MFCC.

1. Introduction

The accents are being studied from the audio files using the feature processing techniques which can contribute to many real-life applications in linguistics. However, the recognition of accents can be complex and have some challenges to deal with since these speech signals have linguistic content, such as speaker characteristics including age, gender, accent, language, emotional state, recording conditions, in addition, each speech pattern from the recordings have different lengths and spoken at different speeds, so padding them was quite negatively affecting the performance and the runtime took longer time, also we were working with many features, we ran into some lack of performance problems with execution time and computational complexity and overfitting. Furthermore, dealing with background noise and environmental factors since these recordings had some noise which interfered with the clarity of speech making it difficult to accurately capture and interpret spoken words. [1]

In order to overcome these challenges and enhance the accuracy and performance of our system we tried to implement padding since these signals are varied, but it was avoided since it affected the performance and gave wrong values for the zero crossing technique and the runtime took too long than expected, instead of padding, features were extracted directly from the speech signals. This approach is more efficient and avoids introducing unnecessary bias into the model. We also enhanced the execution time by integrating joblib's Parallel for feature extraction to speed up the process especially when dealing with a large number of audio files. Moreover, to make sure the feature extraction is standard for all audio files, 'librosa.load()' is applied for them to be converted into a suitable format, and Garbage collection 'gc.collect()' is used to free up memory after processing each set of files and enhance memory utilization and runtime.

2. Background/Related Work

This project's purpose is widely used in various aspects such as the platforms of language learning or educational tools where speakers can originate from different regions to be used for information processing, military information retrieval and other fields. Here are some research papers related to our project which studies different dialects using various classifiers models.

2.1. Palestinian Arabic regional accent recognition

This research is done by Dr. Abouelseoud Hannani and Stephen Taylor. They classified four different Palestinian accents: Jerusalem (JE), Hebron (HE), Nablus (NA) and Ramallah (RA) from 200 different speakers. As they utilized Gaussian Mixture Model - Universal Background Model (GMM-UBM), Gaussian Mixture Model - Support Vector Machines (GMM-SVM) and I-vector framework. Achieving the accuracy of 81.5% with 64 Gaussian components.[2]

2.2. Classification of Spoken English Accents using Deep Learning and Speech Analysis

This research was done in 2022 by computer science students from different universities to study multiple accents of English language including American, British, and Indian accents spoken English from AccentDB using the Convolution Neural Network (CNN). [3]

2.3. Accent Classification

This paper applies machine learning to the problem of accent classification in the hopes of being able to reliably classify accents that are in some data set; they worked on three accents: Cantonese, Hindi, Russian English language speakers. They extracted the features using MFCC and PLP with the Gaussian Mixture Models. By having 3 GMMs - one for each accent. We trained the GMM of each accent class by using the features extracted from the windows of the samples from that accent class as our training data. [4]

2.4. Identification of the English Accent Spoken in Different Countries by the k-Nearest Neighbor Method

This research has been published in 2020 Classification has been made with 12 features obtained by Mel Frequency Cepstrum Coefficients feature extraction method. k-Nearest Neighbor (kNN) were used in the classification and 87.2% success was achieved. There are 330 sound samples including

English accents spoken in Spain, France, Germany, Italy, England and America. [5]

2.5. CLASSIFICATION AND CONVERSION OF ENGLISH ACCENTS USING MACHINE LEARNING

This paper addresses the critical task of accent recognition in English, focusing on up to three European accents. Using machine and deep learning algorithms like Random Forest (RF) and Logistic Regression, assessing performance metrics including accuracy, precision, recall, and f1-score. They considered 80% of the input dataset to be the training data and the remaining 20% to be the testing data. After training the model with Random Forest and Logistic Regression, accuracy score is 95.8% & 91.6% is obtained.[6]

3. Methodology

3.1. Feature Extraction

3.3.1. Mel-frequency Cepstral Coefficients (MFCC)

The Mel-frequency cepstrum (MFCC) represents the short-term power spectrum of the sound, it is a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.[9]
To get the best performance 13 MFCC coefficients were used, they capture the essential characteristics of the speech signal that will make it distinguishable between accents.
The 0th coefficient represents the average log energy of the signal; it captures the loudness of the audio. [8]
1-5 coefficients also known as lower-order coefficients capture the coarse features of the spectrum, which are usually the slope of the spectral envelope.
However, the coefficients from 6-12 capture the finer details such as the formant structures and spectral nuances, which leads to the distinguishing between different sounds and phonemes.
MFCCs are usually driven from taking the fourier transformation of a signal, mapping the power of the spectrum obtained onto the mel scale via the triangular overlapping windows or cosine overlapping windows, then the logs of each of the powers were taken at the mel frequency, then they were transformed to a discrete cosine transform, in this case the amplitudes are the MFCC. [9]

3.3.2. Energy

It is also known as the Root Mean Square (RMS) energy; this feature gives an insight on the audio's amplitude (voice loudness), and how it changes with time. The labrosa library provided us with a function that can measure the energy in the voice.[10]

3.3.3. Zero-Crossing

zero-crossing is the point where the sign of a mathematical function changes, such that there is no presence of voltage power; it normally occurs twice during each cycle.
zero-crossing in electronics is a device that detects when the voltage crosses zero.
However, the zero-crossing rate is the rate at which the signal changes over time, it gives an insight of the noise occurrence. [11]

3.3.4. Chroma Features

The chroma feature represents the whole spectrum in 12 bins octave (twelve pitch categories). they are also referred to as "pitch class profiles", the tuning part approximates to the equal tempered scale.
Chroma Features also capture the harmonic and melodic characteristics of the voice. so, using it will help improve the accent prediction scheme. [12][13]

3.3.5. Spectral Contrast

It is the power that fluctuates regularly along the frequency axis, it gives an insight on the differences between the peaks and valleys in the spectrum [14]

3.3.6 Mel spectrogram

The mel spectrogram finds the frequency content of an audio signal over time. the frequencies are converted to mel scale. This feature represents the short-term power of the voice over the mel-frequency band [15][16]

3.3.7. Spectral Centroid

Also known as the center of spectral mass, it measures the location of the center mass of the spectrum, it also has a connection with the impression of brightness of a sound. [17]

3.3.8. Spectral Bandwidth

It is the width of the frequency band in the spectrum.the range of wavelengths or frequencies over which the magnitude of all spectral components [18]

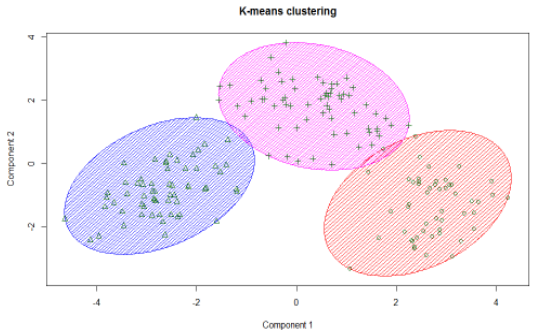
3.3.9. Spectral Rolloff

This feature is a frequency, this is specifies the percentage of the total spectral energy [19]

All these features were combined to give the best representation of each accent, and give the best accuracy and prediction.

3.2. RandomForest with KMeans Clustering

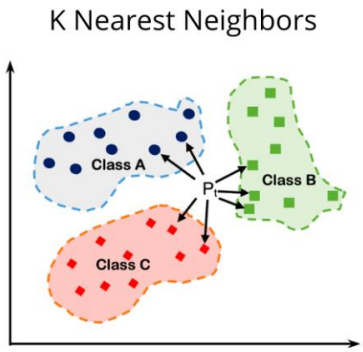
Randomforest is a machine learning algorithm, it combines the output of multiple decision trees to reach a single result.
This type of algorithm handles both classification and regression problems, it also has a great ability in handling complex datasets and mitigating overfitting.[7]
To gain the best accuracy for the RandomForest with KMeans clustering, the model was added to a pipeline along with a standard scaler to scale the features. GridSearchCV was also added to perform hyperparameter tuning, which will find the best combination of parameters for the randomForest classifier.
The Kmeans clustering was used to enhance the feature extraction process, which resulted in improving the classification, it was applied to the Mel-frequency cepstral coefficients (MFCC) features, which made them into a cluster counts, making it more manageable for classification.



1Figure3.2: RandomForest with KMeans Clustering[22]

3.3 K-Nearest Neighbors (KNN)

KNN is a supervised machine learning method that is primarily used for its simplicity and ease of implementation. It does not require any assumptions about the underlying data distribution. It can also handle both numerical and categorical data, making it a flexible choice for various types of datasets in classification [20]. In this project KNN utilized various extracted features e.g., MFCCs, energy, zero-crossing rate to capture the distinct characteristics of different accents. This classifier can be robust to noise due to the fact that it considers multiple neighbors for decision making.



2Figure3.3: K-Nearest Neighbors (KNN)[21]

3.4. Testing Evaluation

When each classifier is being executed and used for training and testing, an evaluation matrix is utilized to study the efficiency of that model.

3.4.1 Accuracy:

The ratio among the correctly predicted outcomes to the total data.

Accuracy= (Correct predictions) / Total predictions

3.4.2 Precision

The ratio of correctly predicted positive observations to the total predicted positives.

Precision= (True Positives + False Positives) / True Positives

3.4.3 Recall

The ratio of correctly predicted positive observations to all observations in actual class.

Recall= (True Positives + False Negatives) / True Positives

3.4.4 F1 score

The balance among precision and recall.

F1 Score= 2×(Precision + Recall) / (Precision×Recall)

4. Experiments and Results

When studying these accents several tests have been done to predict the best outcome for accent recognition. KNN and RandomForest Classifiers have been utilized to train the datasets and test the audio files. The datasets given were two files one for the training and testing, each one contains a separate file for each Palestinian accent and each file of these accents i.e. Hebron has their own audio files from different speakers.

4.1. KNN Classifier Train and Test

Through this classifier when generating the predictions, we tried to achieve the best accuracy which is the proportion among the correct classified instance among all instances and prediction as possible, by calculating the accuracy percentage, and classification report including the precision which is the percentage of true positive prediction among all positive ones, recall which is the percentage of all true positive predictions among all actual positive ones, and, F1 score that is a balance among precision and recall which is the mean of both of them. As seen from the figure below, the overall accuracy is 45% with each calculation of each accent, since for Hebron accent with three incorrectly classified, Jerusalem accent with two incorrectly classified, Nablus accent two incorrect results, and Ramallah reef with two incorrect outcomes. From the graphs below, it is seen that accuracy is 45% as it faces difficulty distinguishing between the classes, also precision is 53% and recall 45% indicate that there are numbers of false positives and false negatives, also F1 score is 0.47 where the model is not maintaining the ability to handle the false positive and neg.

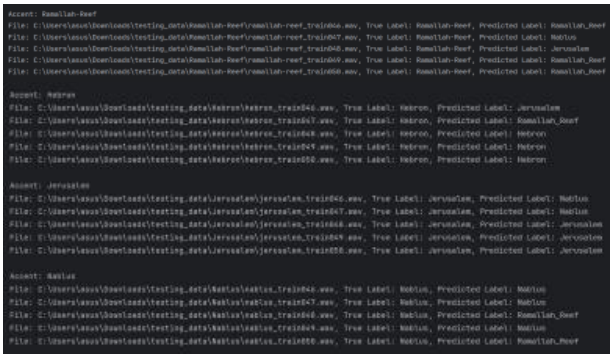
```
Accuracy: 0.45
Precision: 0.53
Recall: 0.45
F1 Score: 0.47

Classification Report for Test Data:
              precision    recall  f1-score   support

      Hebron       1.00      0.60      0.75         5
   Jerusalem       0.60      0.60      0.60         5
      Nablus       0.50      0.60      0.55         5
Ramallah-Reef      0.00      0.00      0.00         5
Ramallah_Reef      0.00      0.00      0.00         0

    accuracy                           0.45        20
   macro avg       0.42      0.36      0.38        20
weighted avg       0.53      0.45      0.47        20
```

3Figure4.1.1: classification report for KNN

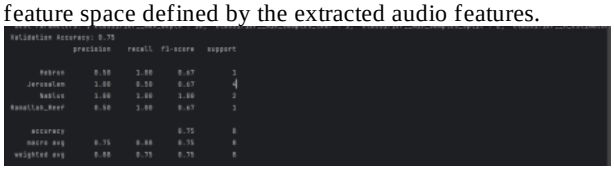


4Figure4.1.2: KNN results

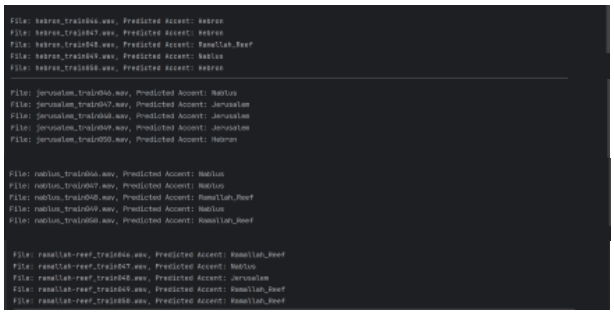
4.2. RandomForest Classifier with K-mean clustering Train and Test

This classifier constructs multiple decision trees during training and outputs the mode for classification or mean prediction for regression of the individual trees. Multiple subsets of the training data are created by random sampling with replacement. Then by averaging multiple trees, the model reduces the risk of overfitting compared to individual decision trees. K-means clusters choose K initial centroids randomly, so the similar features are divided into clusters for each group. By training a separate Random Forest classifier for each cluster, the model can specialize in different subsets of the data and therefore enhance the accuracy and performance as seen where the accuracy was 75% and for each accent the predictions were as follows: Hebron, three correctly specified, Jerusalem, three

correct results, Nablus three correct ones, and Ramallah Reef three correct outcomes as well. The clusters figure shows how the features of each accent are represented by k-means clusters (5 clusters) as each point in the plot represents an audio sample. Points that are closer together in this plot are more similar in the feature space defined by the extracted audio features.



5Figure4.2.1: classification report for Random Forest



6Figure4.2.2: Results for Random Forest

7Figure4.2.3: Random Forest Kmeans clustring

5. Conclusion and future work

This research focused on developing a system to recognize and classify Palestinian Arabic accents from audio recordings using machine learning. By extracting acoustic features such as Mel-frequency cepstral coefficients (MFCCs), zero-crossing rate, energy, chroma, and spectral contrast, we trained models to differentiate accents from various regions in Palestine. Our findings revealed that the RandomForestClassifier combined with KMeans clustering outperformed the K-Nearest Neighbors (KNN) classifier, achieving an accuracy of 75% compared to KNN's 45%. For future works and enhancements to improve our system, we would like to add more train datasets by collecting more audio recordings from diverse speakers to make sure the system has seen enough features, also we would like to apply noise reduction techniques to clean the audio recordings before feature extraction. Moreover, we would like to efficiently implement the Gaussian Mixture model as each Gaussian component in the mixture represents a cluster of similar feature vectors which might improve our accuracy and give better outcomes.

Partners Participation Tasks

We started planning for this project by studying the project's details and discussing the efficient ways to classify these Palestinian accents. Through zoom meetings, each partner was assigned to observe different techniques for accent recognition by reading some related papers and summing up the strengths and limitations of these techniques. Then we all started by preparing the dataset ensuring it is correctly labeled and organized in folders. Each folder should correspond to a different accent, then we loaded these audio files to start data processing and capturing features trying to find the best ways to overcome the challenges found. After the feature extraction and studying the classification models available, we decided to dive into KNN and RandomForestClassifier with k-means clusters as they were the best scenarios to classify each accent efficiently. We started with training the KNN model using the best parameters found then evaluating the model's performance using accuracy, precision, recall, and F1-score. Create a classification report and confusion matrix and get the predicted outcomes for each accent audio .wav file. Then we decided to use RandomForest Classifier with k-means clusters to compare and check if the performance has changed. We edited the feature extraction by adding the enhancing k-means clustering to cluster MFCC features and incorporate cluster counts into the final feature vector with parallel processing. We then tried to visualize these clusters to better understand the results and check the predictions and compare the percentage of the correct ones to the overall recordings. We would also try to refine the clustering of the features, seeing that the clusters are not that well-separated, we would consider refining the feature extraction process to include more discriminative features.

6. References

[1] "Challenges In Speech Recognition," *FasterCapital*. https://fastercapital.com/topics/challenges-in-speech-recognition.html (accessed Jun. 10, 2024).

[2] A. Hanani, H. Basha, Y. Sharaf, and S. Taylor, "Palestinian Arabic regional accent recognition," Palestinian Arabic Regional Accent Recognition, Oct. 2015, doi: 10.1109/sped.2015.7343088.

[3] (PDF) classification of spoken English accents using Deep Learning and speech analysis. (n.d.-c). https://www.researchgate.net/publication/362705130_Classification_of_Spoken_English_Accents_Using_Deep_Learning_and_Speech_Analysis

[4] P. Watanaprakornkul, C. Eksombatchai, and P. Chien, "Accent Classification." Accessed: Jun. 10, 2024. [Online]. Available: https://cs229.stanford.edu/proj2010/WatanaprakornkulEksombatchaiChien-AccentClassification.pdf

[5] (PDF) identification of the English accent spoken in different ... (n.d.-d). https://www.researchgate.net/publication/348141979_Identification_of_the_English_Accent_Spoken_in_Different_Countries_by_the_k-Nearest_Neighbor_Method

[6] J. Raju, G. Murali, S. Nallamolu, P. Kavyasri, and P. Kumar Ghosh, "CLASSIFICATION AND CONVERSION OF ENGLISH ACCENTS USING MACHINE LEARNING," International Research Journal of Modernization in Engineering Technology and Science, pp. 2582–5208, doi: https://doi.org/10.56726/IRJMETS53622.

[7] S. E. R, "Understand random Forest algorithm with examples (Updated 2024)," Analytics Vidhya, Jun. 06, 2024. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[8] "Learn FLUCOMA." https://learn.flucoma.org/reference/mfcc/

[9] Wikipedia contributors, "Mel-frequency cepstrum," Wikipedia, Feb. 28, 2024. https://en.wikipedia.org/wiki/Mel-frequency_cepstrum

[10] M. Harris, "Understanding RMS in audio," Mix & Master My Song, Feb. 13, 2024. https://mixandmastermysong.com/understanding-rms-in-audio/#:~:text=In%20audio%2C%20RMS%20is%20used,can%20handle%20effectively%20over%20time.

[11] Wikipedia contributors, "Zero crossing," Wikipedia, Aug. 13, 2023. https://en.wikipedia.org/wiki/Zero_crossing

[12] Wikipedia contributors, "Chroma feature," Wikipedia, Feb. 09, 2024. https://en.wikipedia.org/wiki/Chroma_feature#:~:text=Chroma-based%20features%2C%20which%20are,to%20the%20equal-tempered%20scale.

[13] Chroma feature analysis and synthesis. (n.d.-a). https://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn/

[14] So, N.L.T., Edwards, J.A. and Woolley, S.M.N. (2020) *Auditory selectivity for spectral contrast in cortical neurons and behavior, The Journal of neuroscience : the official journal of the Society for Neuroscience*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6989003/#:~:text=Vocalizations%20typically%20have%20spectral%20structure,multiples%20of%20a%20fundamental%20frequency. (Accessed: 10 June 2024).

[15] L. Roberts, "Understanding the MEL Spectrogram - Analytics Vidhya - Medium," Medium, Jan. 17, 2024. [Online]. Available: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53

[16] "Introduction to audio data - Hugging Face Audio Course." https://huggingface.co/learn/audio-course/en/chapter1/audio_data

[17] Wikipedia contributors, "Spectral centroid," Wikipedia, Jan. 03, 2023. https://en.wikipedia.org/wiki/Spectral_centroid#:~:text=The%20spectral%20centroid%20is%20a,called%20center%20of%20spectral%20mass

[18] Wikipedia contributors, "Spectral width," Wikipedia, Nov. 06, 2023. https://en.wikipedia.org/wiki/Spectral_width

[19] "spectral_features." https://musicinformationretrieval.com/spectral_features.html#:~:text=Spectral%20rolloff%20is%20the%20frequency,feature.

[20] GeeksforGeeks, "K-Nearest Neighbor(KNN) algorithm," GeeksforGeeks, Jan. 25, 2024. https://www.geeksforgeeks.org/k-nearest-neighbours/

[21] "Fig. 2 Example on KNN classifier," ResearchGate. https://www.researchgate.net/figure/Example-on-KNN-classifier_fig1_331424423

[22] M. Khan, "KMEANS Clustering for Classification - towards Data Science," Medium, Jun. 18, 2018. [Online]. Available: https://towardsdatascience.com/kmeans-clustering-for-classification-74b992405d0a

7. Appendix

7.1 RandomForest with KMeans Clustering

```
import os
import numpy as np
import librosa
import noisereduce as nr
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from joblib import Parallel, delayed
import gc

# Suppress warnings globally
warnings.filterwarnings("ignore", category=RuntimeWarning)
warnings.filterwarnings("ignore", category=UserWarning)

# Enhanced feature extraction function with clustering
def extract_features(file_path, n_mfcc=13, n_clusters=5):
    y, sr = librosa.load(file_path, sr=None)

    # Perform noise reduction
    # y = nr.reduce_noise(y=y, sr=sr)

    # Extract features
    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=n_mfcc)
    zero_crossing_rate = np.mean(librosa.feature.zero_crossing_rate(y))
    energy = np.mean(librosa.feature.rms(y=y))
    chroma = np.mean(librosa.feature.chroma_stft(y=y, sr=sr), axis=1)
    spectral_contrast = np.mean(librosa.feature.spectral_contrast(y=y, sr=sr), axis=1)
    mel_spectrogram = np.mean(librosa.feature.melspectrogram(y=y, sr=sr), axis=1)
    spectral_centroid = np.mean(librosa.feature.spectral_centroid(y=y, sr=sr))
    spectral_bandwidth = np.mean(librosa.feature.spectral_bandwidth(y=y, sr=sr))
    spectral_rolloff = np.mean(librosa.feature.spectral_rolloff(y=y, sr=sr))

    mfccs_mean = np.mean(mfccs, axis=1)

    # Perform KMeans clustering on the MFCC features
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    mfccs_cluster_labels = kmeans.fit_predict(mfccs.T)

    # Count the occurrences of each cluster label
    cluster_counts = np.bincount(mfccs_cluster_labels, minlength=n_clusters)

    # Concatenate all features
    features = np.concatenate((mfccs_mean, chroma, spectral_contrast, mel_spectrogram,
                               [zero_crossing_rate, energy, spectral_centroid, spectral_bandwidth,
spectral_rolloff], cluster_counts))

    return features

# Directories containing the training data
train_data_dirs = {
    'Hebron': r'C:\Users\asus\Downloads\training_data\Hebron',
    'Jerusalem': r'C:\Users\asus\Downloads\training_data\Jerusalem',
    'Nablus': r'C:\Users\asus\Downloads\training_data\Nablus',
    'Ramallah_Reef': r'C:\Users\asus\Downloads\training_data\Ramallah_Reef'
}

# Directory containing the test data
test_data_dir = r'C:\Users\asus\Downloads\testing_data\Ramallah_Reef'

# Function to load and extract features for a given set of data directories
def load_and_extract_features(data_dirs):
    X = []
    y = []
    for accent_label, dir_path in data_dirs.items():
        files = [file for file in os.listdir(dir_path) if file.endswith('.wav')]
        features = Parallel(n_jobs=-1)(delayed(extract_features)(os.path.join(dir_path, file))
for file in files)
        X.extend(features)
        y.extend([accent_label] * len(features))
        gc.collect()  # Garbage collection to free memory
    return np.array(X), np.array(y)

# Load and extract features from the training data
X_train, y_train = load_and_extract_features(train_data_dirs)
```

```
# Perform t-SNE for dimensionality reduction
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X_train)

# Plot the t-SNE visualization of the extracted features
plt.figure(figsize=(10, 7))
sns.scatterplot(x=X_tsne[:, 0], y=X_tsne[:, 1], hue=y_train, palette='viridis', legend='full')
plt.title('t-SNE visualization of extracted features')
plt.xlabel('t-SNE component 1')
plt.ylabel('t-SNE component 2')
plt.legend(title='Accent')
plt.show()

# Split the training data into training and validation sets
X_train_split, X_val, y_train_split, y_val = train_test_split(X_train, y_train, test_size=0.2,
random_state=42)

# Define a pipeline with scaling and RandomForestClassifier
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', RandomForestClassifier(random_state=42))
])

# Define the parameter grid for GridSearchCV
param_grid = {
    'classifier__n_estimators': [100, 200, 300, 400, 500],
    'classifier__max_depth': [10, 20, 30, 40, 50],
    'classifier__min_samples_split': [2, 5, 10, 15, 20],
    'classifier__min_samples_leaf': [1, 2, 4, 6, 8]
}

# Perform GridSearchCV to find the best parameters
grid_search = GridSearchCV(pipeline, param_grid, cv=5, n_jobs=-1, verbose=1)
grid_search.fit(X_train_split, y_train_split)

# Best parameters from GridSearchCV
print(f'Best Parameters: {grid_search.best_params_}')

# Evaluate the model on the validation set
y_val_pred = grid_search.predict(X_val)
val_accuracy = accuracy_score(y_val, y_val_pred)
print(f'Validation Accuracy: {val_accuracy}')
print(classification_report(y_val, y_val_pred, zero_division=1))

# Plot the confusion matrix for the validation set
cm = confusion_matrix(y_val, y_val_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=train_data_dirs.keys(),
yticklabels=train_data_dirs.keys())
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix - Validation Set')
plt.show()

# Function to load and extract features from the test data
def load_and_extract_test_features(test_dir):
    files = [file for file in os.listdir(test_dir) if file.endswith('.wav')]
    features = Parallel(n_jobs=-1)(delayed(extract_features)(os.path.join(test_dir, file)) for
file in files)
    return np.array(features), files

# Load and extract features from the test data
X_test_unlabeled, test_files = load_and_extract_test_features(test_data_dir)

# Predict the accents of the unlabeled test files using the trained model
y_test_unlabeled_pred = grid_search.predict(X_test_unlabeled)

# Display predictions for each test file
for file_path, predicted_label in zip(test_files, y_test_unlabeled_pred):
    print(f"File: {file_path}, Predicted Accent: {predicted_label}")

# Clean up to free memory
gc.collect()
```

7.2. K-Nearest Neighbors (KNN)

```
import os
import numpy as np
import librosa
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
classification_report
from sklearn.model_selection import GridSearchCV, cross_val_score
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Function to extract multiple audio features
def extract_features(file_path):
    try:
        y, sr = librosa.load(file_path, sr=None)
        mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
        energy = librosa.feature.rms(y=y)
        zero_crossings = librosa.feature.zero_crossing_rate(y)
        chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
        mel_spectrogram = librosa.feature.melspectrogram(y=y, sr=sr)
        spectral_contrast = librosa.feature.spectral_contrast(y=y, sr=sr)

        # Calculate the mean of the features
        mfccs_mean = np.mean(mfccs.T, axis=0)
        energy_mean = np.mean(energy.T, axis=0)
        zero_crossings_mean = np.mean(zero_crossings.T, axis=0)
        chroma_stft_mean = np.mean(chroma_stft.T, axis=0)
        mel_spectrogram_mean = np.mean(mel_spectrogram.T, axis=0)
        spectral_contrast_mean = np.mean(spectral_contrast.T, axis=0)

        # Concatenate all features into a single feature vector
        feature_vector = np.hstack([mfccs_mean, energy_mean, zero_crossings_mean,
                                    chroma_stft_mean, mel_spectrogram_mean, spectral_contrast_mean])
        return feature_vector
    except Exception as e:
        print(f"Error processing {file_path}: {e}")
        return None

# Load data from a specified directory
def load_data(data_path):
    X, y, file_paths = [], [], []
    for root, dirs, files in os.walk(data_path):
        for file in files:
            if file.endswith('.wav'):
                file_path = os.path.join(root, file)
                label = os.path.basename(root)
                features = extract_features(file_path)
                if features is not None:
                    X.append(features)
                    y.append(label)
                    file_paths.append(file_path)
    return np.array(X), np.array(y), file_paths

# Paths to the datasets
train_data_path = r'C:\Users\asus\Downloads\training_data'
test_data_path = r'C:\Users\asus\Downloads\testing_data'

# Load training data
X_train, y_train, train_file_paths = load_data(train_data_path)

# Load testing data
X_test, y_test, test_file_paths = load_data(test_data_path)

# Normalize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Dimensionality reduction with PCA
pca = PCA(n_components=30)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

# Hyperparameter tuning for KNN
param_grid = {'n_neighbors': np.arange(1, 31), 'metric': ['euclidean', 'manhattan', 'minkowski']}
knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn, param_grid, cv=5)
knn_cv.fit(X_train_pca, y_train)

# Best parameters
best_params = knn_cv.best_params_
best_k = best_params['n_neighbors']
best_metric = best_params['metric']
print(f"\nBest parameters: {best_params}")

# Initialize and train the KNN classifier with the best parameters
knn = KNeighborsClassifier(n_neighbors=best_k, metric=best_metric)
knn.fit(X_train_pca, y_train)

# Cross-validation score
cv_scores = cross_val_score(knn, X_train_pca, y_train, cv=5)
print(f"\nCross-validation accuracy: {np.mean(cv_scores):.2f} (+/- {np.std(cv_scores):.2f})")

# Make predictions
y_pred_train = knn.predict(X_train_pca)
y_pred_test = knn.predict(X_test_pca)

# Calculate performance metrics for training data
accuracy_train = accuracy_score(y_train, y_pred_train)
precision_train = precision_score(y_train, y_pred_train, average='weighted', zero_division=0)
recall_train = recall_score(y_train, y_pred_train, average='weighted', zero_division=0)
f1_train = f1_score(y_train, y_pred_train, average='weighted', zero_division=0)

# Calculate performance metrics for testing data
accuracy_test = accuracy_score(y_test, y_pred_test)
precision_test = precision_score(y_test, y_pred_test, average='weighted', zero_division=0)
recall_test = recall_score(y_test, y_pred_test, average='weighted', zero_division=0)
f1_test = f1_score(y_test, y_pred_test, average='weighted', zero_division=0)

# Print the results
print("\nTraining Data Performance:")
print(f"Accuracy: {accuracy_train:.2f}")
print(f"Precision: {precision_train:.2f}")
```

```
print(f"Recall: {recall_train:.2f}")
print(f"F1 Score: {f1_train:.2f}")

print("\nTesting Data Performance:")
print(f"Accuracy: {accuracy_test:.2f}")
print(f"Precision: {precision_test:.2f}")
print(f"Recall: {recall_test:.2f}")
print(f"F1 Score: {f1_test:.2f}")

# Ensure the labels in the classification report match the unique labels in the testing set
unique_labels = np.unique(np.concatenate((y_train, y_test)))

print("\nClassification Report for Test Data:")
print(classification_report(y_test, y_pred_test, labels=unique_labels, zero_division=0))

# Detailed results for each audio file in the test set
test_results = pd.DataFrame({
    'File Path': test_file_paths,
    'True Label': y_test,
    'Predicted Label': y_pred_test
})

print("\nDetailed Results for Each Test Audio File:")
print(test_results)

# Print predictions for each accent in the testing data
print("\nPredictions for Each Accent in the Testing Data:")
accents = np.unique(y_test)
for accent in accents:
    print(f"\nAccent: {accent}")
    accent_files = test_results[test_results['True Label'] == accent]
    for _, row in accent_files.iterrows():
        print(f"File: {row['File Path']}, True Label: {row['True Label']}, Predicted Label: {row['Predicted Label']}")

# Apply KMeans clustering
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(X_train_pca)

# Perform t-SNE for visualization
tsne = TSNE(n_components=2, random_state=42)
X_train_tsne = tsne.fit_transform(X_train_pca)

# Create a DataFrame for plotting
df_tsne = pd.DataFrame({'X': X_train_tsne[:, 0], 'Y': X_train_tsne[:, 1], 'Cluster': clusters, 'Label': y_train})

# Plot the t-SNE visualization with clusters
plt.figure(figsize=(10, 7))
sns.scatterplot(x='X', y='Y', hue='Cluster', palette='viridis', data=df_tsne, legend='full')
for label in np.unique(y_train):
    x = df_tsne[df_tsne['Label'] == label]['X'].mean()
    y = df_tsne[df_tsne['Label'] == label]['Y'].mean()
    plt.text(x, y, label, fontsize=12, ha='center')
plt.title('t-SNE visualization of training data with KMeans clusters')
plt.xlabel('t-SNE component 1')
plt.ylabel('t-SNE component 2')
plt.legend(title='Cluster')
plt.show()
```