SPOKEN DIGIT CLASSIFICATION

P SAIKARTHIK

VU21CSEN0100398

1. Abstract

This report outlines the process of classifying spoken digit audio files into corresponding digit labels using a machine learning model. The features are extracted from audio signals using Mel Frequency Cepstral Coefficients (MFCC), a popular technique in audio processing. The classification is performed using a Random Forest classifier, which achieves reliable accuracy in predicting the correct digit based on audio features.

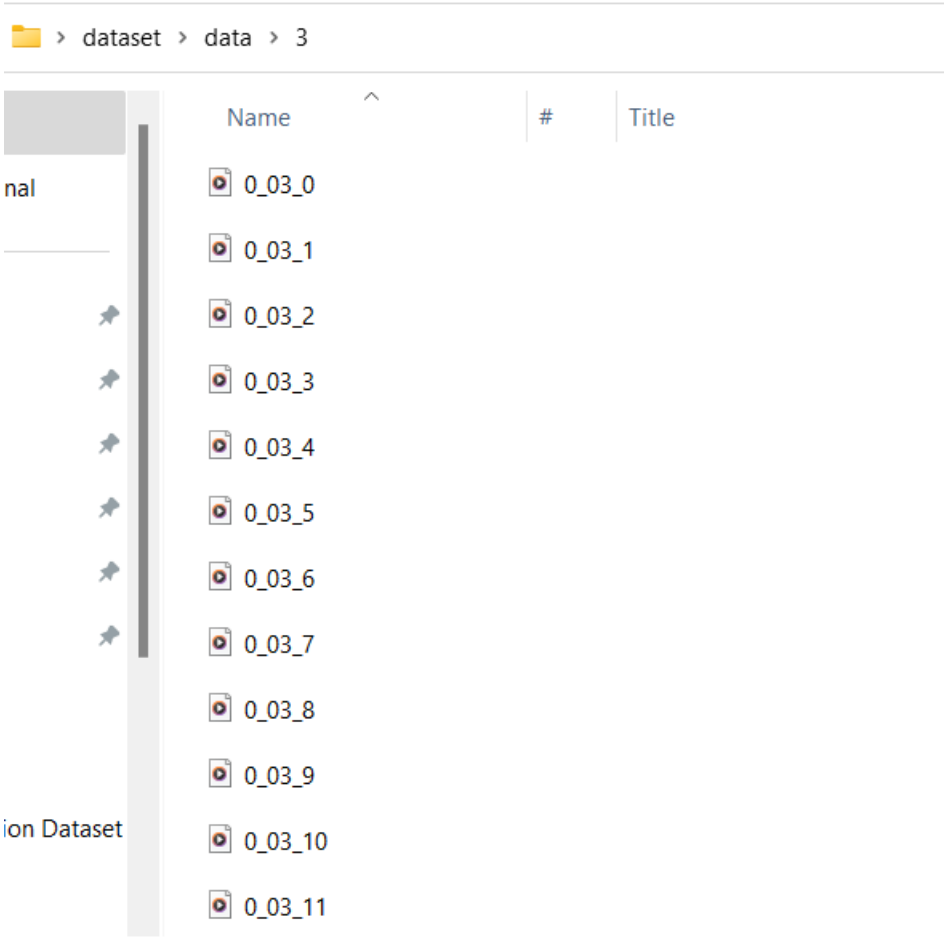
2. Introduction

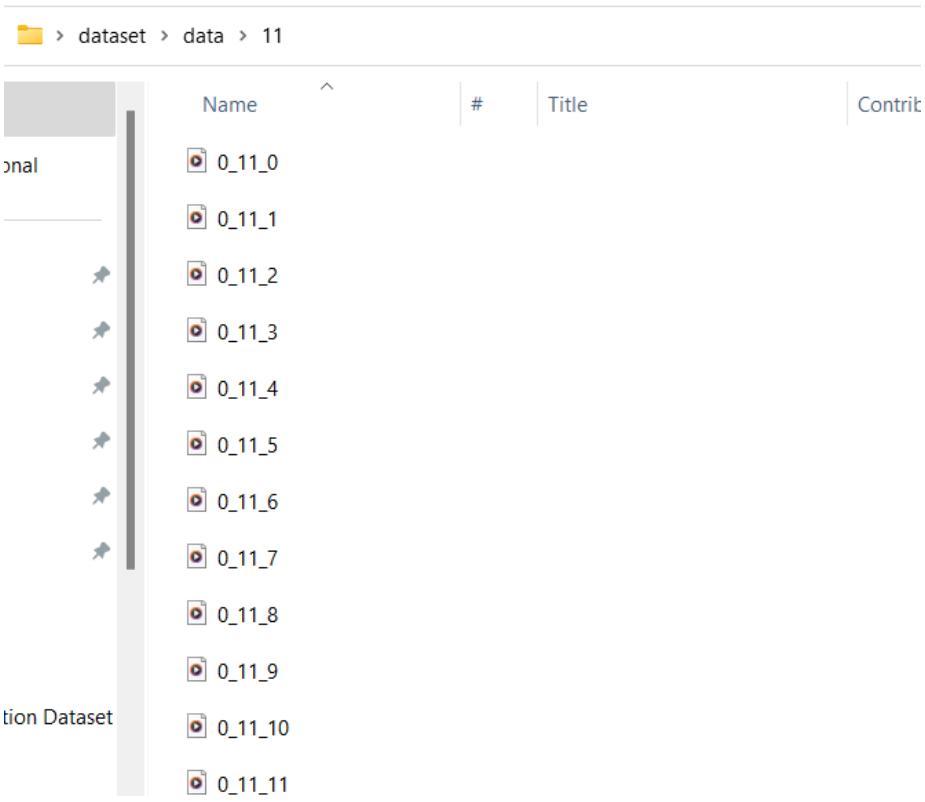
The goal of this project is to classify spoken digits from audio files into their respective digit labels. This task is a classic problem in the field of speech recognition and has applications in voice-activated systems and automated transcription.

3. Data Collection

Dataset Structure: The dataset consists of audio files, where each file represents a spoken digit. The files are organized in directories named after the corresponding digit labels (1-60).

File Handling: The function get_audio_file_paths retrieves all .wav files and their associated labels from the dataset directory.


dataset › data › 3
Name
#
Title
0_03_0
0_03_1
0_03_2
0_03_3
0_03_4
0_03_5
0_03_6
0_03_7
0_03_8
0_03_9
0_03_10
0_03_11


dataset › data › 11
Name
#
Title
Contrit
0_11_0
0_11_1
0_11_2
0_11_3
0_11_4
0_11_5
0_11_6
0_11_7
0_11_8
0_11_9
0_11_10
0_11_11

4. Feature Extraction

MFCC Extraction: The feature_extractor function processes each audio file using librosa to extract MFCC features, specifically using 40 coefficients. These coefficients summarize the spectral properties of the audio, capturing essential information for classification.

Feature Aggregation: The MFCC features for each audio file are averaged across time frames, resulting in a single feature vector per file.

[2]:
```
def feature_extractor(file):
    dat,sir = librosa.load(file)
    mpccs = librosa.feature.mfcc(y=dat,sr=sir,n_mfcc=40)
    mpccs = np.mean(mpccs.T,axis=0)
    return mpccs
```

5. Modeling

Data Preparation: The extracted features are split into training and testing sets using an 80-20 split. The features are then standardized using a StandardScaler to ensure that all features contribute equally to the model.

Classifier: A RandomForestClassifier with 100 estimators is trained on the extracted features. Random Forest is chosen for its robustness and ability to handle high-dimensional data.

Confusion Matrix: The model's performance is evaluated using a confusion matrix, which provides insights into the classification accuracy for each digit.

[4]:
```
[[array([-6.1790826e+02,  1.0158977e+02,  9.8475780e+00,  2.6403582e+01,
          2.5784245e+01,  1.7553813e+00,  7.3807864e+00, -1.2520157e+01,
          8.8408928e+00, -2.5133634e-01, -5.3353767e+00,  5.4605556e-01,
          9.1450186e+00, -5.2191892e+00, -6.9610351e-03,  5.0607262e+00,
          5.0575286e-01,  5.3159928e+00, -1.2108281e+01,  1.2537379e+00,
         -2.9457762e+00,  9.6537006e-01, -2.3022270e+00,  9.2183232e-01,
         -4.7672043e+00,  2.8122263e+00, -7.9008355e+00,  3.3631225e+00,
         -6.1831099e-01, -1.5131553e+00,  4.6017653e-01,  1.5612110e+00,
         -3.2615309e+00,  3.7520194e+00, -1.9152343e+00, -1.8270775e+00,
         -2.7764413e+00,  2.1276488e+00,  2.8370333e-01, -3.6298856e-01],
        dtype=float32),
  1],
 [array([-6.3650385e+02,  1.0466348e+02,  1.8785160e+01,  3.2966640e+01,
          3.2188751e+01,  5.5603471e+00,  2.7990646e+00, -8.2715473e+00,
          8.1379118e+00, -4.7811623e+00, -3.0265276e+00,  2.4315081e+00,
          6.4764619e+00, -2.4659326e+00,  3.5354555e+00,  6.5043545e+00,
          2.5801427e+00,  5.0594425e+00, -1.0814376e+01, -1.9987603e+00,
         -3.8614321e+00,  4.7192736e+00, -1.9209657e+00,  6.9846380e-01
```

IMPLEMENTATION

[1]:
```
import librosa
import numpy as np
import pandas as pd
```

[2]:
```
def feature_extractor(file):
    dat,sir = librosa.load(file)
    mpccs = librosa.feature.mfcc(y=dat,sr=sir,n_mfcc=40)
    mpccs = np.mean(mpccs.T,axis=0)
    return mpccs
```

[3]:
```
import os
def get_audio_file_paths(base_dir):
    file_paths = []
    for label in range(1, 61):
        folder_path = os.path.join(base_dir, 'data', str(label))
        if os.path.isdir(folder_path):
            for file_name in os.listdir(folder_path):
                if file_name.endswith('.wav'):
                    file_paths.append((os.path.join(folder_path, file_name), label))
    return file_paths

base_dir = ""
audio_file_paths = get_audio_file_paths(base_dir)
audio_file_paths[1000]
```

[3]:
```
('data\\3\\0_03_0.wav', 3)
```

[4]:
```
extracted=[]
for i in audio_file_paths:
    feature = feature_extractor(i[0])
```

[4]:
```
extracted=[]
for i in audio_file_paths:
    feature = feature_extractor(i[0])
    extracted.append([feature,i[1]])
extracted
```

```
          -4.7552905e+00, -1.1814265e+00,  3.5069315e+00,  4.3159443e-01,
          -7.1796101e-01,  1.7011865e+00, -2.7039731e+00,  1.1306893e+00,
          -5.3126459e+00,  3.7995965e+00, -4.7157079e-01, -8.7783432e-01],
         dtype=float32),
  1],
 [array([-6.41805481e+02,  1.01099884e+02,  1.93796291e+01,  3.99196053e+01,
          3.70425377e+01,  1.43610363e+01,  8.96954823e+00, -1.41925449e+01,
         -2.95069790e+00, -4.68774885e-01, -3.03564048e+00,  2.94162321e+00,
          1.74928755e-01,  1.82381773e+00, -4.96266890e+00,  6.43663979e+00,
          2.24818802e+00,  2.36560249e+00,  2.64937282e+00,  1.12577945e-01,
         -6.77204704e+00,  6.80447531e+00, -2.90879178e+00, -1.64264357e+00,
          1.86239135e+00, -4.44232130e+00, -2.48431826e+00, -7.43807197e-01,
         -4.21050215e+00,  3.17727029e-01,  2.11969900e+00, -1.56583440e+00,
         -4.41288680e-01,  2.62102151e+00, -2.19893336e+00,  1.13192058e+00,
          2.16341436e-01,  3.98350763e+00, -1.32747912e+00,  4.79065847e+00],
         dtype=float32),
  1],
 [array([-6.2818634e+02,  1.0793441e+02,  1.4876519e+01,  3.0683203e+01,
```

[5]:
```
df = pd.DataFrame(extracted,columns=['feature','label'])
x=np.array(df['feature'].tolist())
y=np.array(df['label'].tolist())
```

[12]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score , confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
```

```
[7]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=40)
```

```
[8]: scalar = StandardScaler()
     x_train = scalar.fit_transform(x_train)
     x_test = scalar.transform(x_test)
     x_train
```

```
[8]: array([[ 1.4982452 ,  0.62828267, -0.48200047, ..., -0.9366239 ,
             -0.57168996, -0.6387048 ],
            [-0.8252031 ,  1.4612683 , -0.47313762, ...,  0.64777005,
             -0.6812681 , -0.08610778],
            [ 0.09370439, -1.798307  ,  0.5165468 , ...,  0.57542855,
              0.08015469,  0.39512527],
            ...,
            [-0.07312949,  0.80285054,  0.95932734, ..., -0.5034077 ,
             -0.30151954, -0.9243064 ],
            [-0.6016143 ,  1.1405637 ,  0.56099   , ...,  0.97922134,
             -0.18187448,  0.5610302 ],
            [-1.1525924 ,  1.6839427 ,  0.97054166, ..., -0.04627597,
              0.01872274, -1.1000991 ]], dtype=float32)
```

```
[9]: classifier = RandomForestClassifier(n_estimators=100,random_state=40)
     classifier.fit(x_train,y_train)
```

```
[9]: RandomForestClassifier
     RandomForestClassifier(random_state=40)
```

```
[10]: y_pred = classifier.predict(x_test)
```

```
[13]: print("Accuracy: ",accuracy_score(y_test,y_pred))

      Accuracy:  0.9743333333333334
```

6. Results

Accuracy: The model's accuracy on the test data is a crucial metric indicating how well the model generalizes to unseen audio files.

Confusion Matrix: The confusion matrix and its visualization display the distribution of correct and incorrect predictions, highlighting any digits that may be confused with one another.


```
[9]: classifier = RandomForestClassifier(n_estimators=100,random_state=40)
     classifier.fit(x_train,y_train)
```

```
[9]: RandomForestClassifier
     RandomForestClassifier(random_state=40)
```

```
[10]: y_pred = classifier.predict(x_test)
```

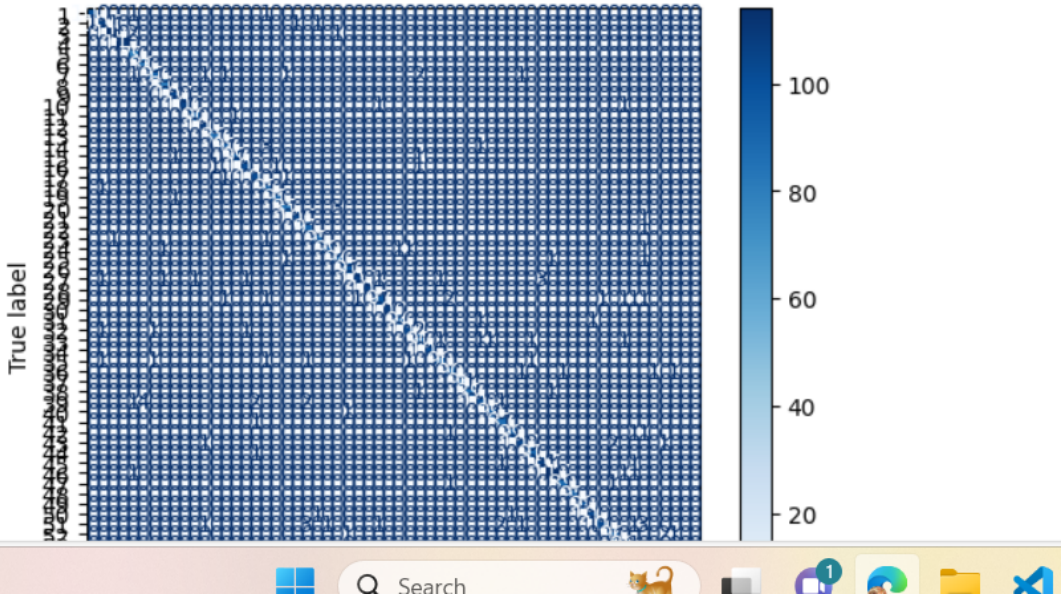
```
[13]: print("Accuracy: ",accuracy_score(y_test,y_pred))

      Accuracy:  0.9743333333333334
```

7. Visualization

The confusion matrix is visualized using matplotlib, providing a clear view of the classifier's performance across all digit classes.

```
•[18]: cm = confusion_matrix(y_test, y_pred, labels=classifier.classes_)
       disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classifier.classes_)
       cm_df = pd.DataFrame(cm, index=classifier.classes_, columns=classifier.classes_)
       print(cm)
       disp.plot(cmap=plt.cm.Blues)
       plt.show()
```

```
[[101   0   0 ...   0   0   0]
 [  1 106   1 ...   0   0   0]
 [  0   0 114 ...   0   0   0]
 ...
 [  0   0   0 ...  89   0   1]
 [  0   0   0 ...   0  90   0]
 [  0   0   0 ...   0   0 103]]
```



[19]:
```
print("Confusion Matrix:")
print(cm_df)
```

```
Confusion Matrix:
     1    2    3    4   5   6   7   8    9    10  ...   51  52  53   54  55
1   101    0    0    0   1   0   0   0    0    0  ...    0   0   0    0   0
2     1  106    1    0   0   0   0   0    0    0  ...    0   0   0    0   0
3     0    0  114    0   2   0   0   0    0    0  ...    0   0   0    0   0
4     0    0    0  111   0   0   0   0    0    0  ...    0   0   0    0   0
5     0    0    0    0  79   0   0   0    0    0  ...    0   0   0    0   0
6     0    0    0    0   0  86   0   0    0    0  ...    0   0   0    0   0
7     0    0    0    0   1   0  92   0    0    0  ...    0   0   0    0   0
8     0    0    0    0   0   0   0  81    0    0  ...    0   0   0    0   0
9     0    0    0    0   0   0   0   0  104    0  ...    0   0   0    0   0
10    0    0    0    0   0   0   0   0    0  106  ...    0   0   1    0   0
11    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   0
12    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   0
13    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   0
14    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   0
15    0    0    0    0   0   0   0   0    1    0  ...    0   0   0    0   0
16    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   0
17    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   0
18    0    1    0    0   0   0   0   0    0    0  ...    0   0   0    0   0
19    0    0    0    0   0   0   0   0    1    0  ...    0   0   0    0   0
20    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   0
21    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   1
22    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   1
23    0    0    1    0   0   0   0   0    0    0  ...    0   0   0    0   0
24    0    0    0    0   0   0   0   1    0    0  ...    0   0   0    0   1
25    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   1
26    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   0
27    0    1    0    0   0   0   0   1    0    0  ...    0   0   0    0   0
28    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   0
29    0    0    0    0   0   0   0   0    0    0  ...    1   0   1    1   1
30    0    0    0    0   0   0   0   0    0    0  ...    0   0   0    0   0
```

8. Conclusion

The Random Forest classifier, trained on MFCC features extracted from audio files, demonstrates strong performance in the task of spoken digit classification. The model can be further refined by exploring additional feature extraction methods or advanced classification algorithms.

39	0	0	0	0	0
40	0	0	0	0	0
41	0	0	0	0	0
42	0	0	0	0	0
43	0	1	0	0	0
44	0	0	0	0	0
45	0	0	0	0	0
46	0	0	0	0	0
47	0	0	0	0	0
48	0	0	0	0	0
49	0	0	0	0	0
50	0	0	0	0	0
51	0	0	0	0	0
52	0	2	1	0	0
53	0	0	0	0	0
54	0	0	0	0	0
55	0	0	0	0	0
56	96	2	0	0	0
57	0	86	0	0	0
58	0	0	89	0	1
59	1	0	0	90	0
60	0	0	0	0	103

[60 rows x 60 columns]

]:

🌐 Sources

librosa.github.io - Feature extraction

scikit-learn.org - RandomForestClassifier

towardsdatascience.com - Audio Classification

kaggle.com - Audio Data Preparation

matplotlib.org - Confusion Matrix Visualization