Urban Sound Classification

Author: PAVAN B

Date: 12-09-2024

Abstract

Urban sound classification is a challenging task in the field of machine learning that involves categorizing audio clips into predefined sound classes. This project addresses the Urban Sound Classification problem using a dataset consisting of 8732 labeled urban sound excerpts, categorized into ten distinct classes including car horns, dog barks, and sirens.

The approach involves several key steps: feature extraction, model training, and evaluation. Features are extracted from audio files using Mel-frequency cepstral coefficients (MFCCs), which capture the spectral characteristics of the sound. The dataset is split into training and testing subsets, with features scaled to improve model performance.

A RandomForestClassifier is employed due to its effectiveness in handling complex data and its robustness against overfitting. The model is trained on the extracted features and evaluated using accuracy metrics, confusion matrices, and classification reports. Cross-validation results indicate an average accuracy of 89.12%, with a final test set accuracy of 91.54%.

The project demonstrates the efficacy of machine learning in audio classification and provides insights into model performance and feature importance. Future work could explore advanced models and hyperparameter tuning to further enhance classification accuracy.

Introduction

Urban sound classification is a machine learning problem where we aim to classify sound excerpts into various categories such as car horns, dog barks, and sirens. In this project, we use machine learning models to solve the problem using the "Urban Sound Classification" dataset.

Dataset

The dataset consists of 8732 labeled sound excerpts, each belonging to one of the following 10 urban sound classes:

- Air Conditioner
- Car Horn
- Children Playing
- Dog Bark
- Drilling
- Engine Idling
- Gun Shot
- Jackhammer
- Siren
- Street Music

Each sound excerpt is identified by an ID and labeled with a sound class. We have two datasets: one for training and one for testing, both of which include .wav files.

Data Preprocessing

Feature Extraction:
Audio features are extracted from .wav files using the librosa and soundfile libraries. The primary feature used is the MFCC (Mel-frequency cepstral coefficients), which is a popular feature extraction method in audio processing.
The following function was used to extract MFCC features from audio files:

```
def extract_features(file_name):
    audio, sample_rate = sf.read(file_name)
    if len(audio.shape) > 1:
        audio = audio.mean(axis=1)
    mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
    return mfccs.mean(axis=1)
```

Data Handling:
We processed the train and test datasets, extracted features, and created DataFrames with relevant audio features and labels.
Audio files were processed using the process_audio_directory() function, which loops through each .wav file, extracts the MFCC features, and returns a DataFrame.

Train/Test Split:
The train dataset was split into 70% for training and 30% for testing:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

Feature Scaling: The features were scaled using StandardScaler:

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Model Training and Evaluation
Model: We used a RandomForestClassifier with 100 estimators. Cross-validation was performed using 5-fold cross-validation:

```
model = RandomForestClassifier(n_estimators=100)
cv_scores = cross_val_score(model, X_train_scaled, y_train, cv=5)
```

Cross-validation results:

```
Cross-validation accuracy scores: [0.9028, 0.8896, 0.8817, 0.8844, 0.8974]
Average cross-validation accuracy: 0.8912
```

Test Accuracy:
The model was trained on the entire training set, and predictions were made on the test set:

```
Test Set Accuracy: 0.9154
```

Classification Report: The classification report shows the precision, recall, and F1-score for each class:

Classification Report:

	precision	recall	f1-score	support
air_conditioner	0.97	0.97	0.97	163
car_horn	0.98	0.94	0.96	93
children_playing	0.76	0.92	0.83	163
dog_bark	0.86	0.81	0.84	186
drilling	0.92	0.97	0.95	181
engine_idling	0.96	0.94	0.95	193
gun_shot	1.00	0.81	0.90	74
jackhammer	0.94	0.98	0.96	208
siren	0.94	0.96	0.95	180
street_music	0.93	0.81	0.86	190
accuracy			0.92	1631
macro avg	0.93	0.91	0.92	1631
weighted avg	0.92	0.92	0.92	1631

Confusion Matrix: A confusion matrix was plotted to visualize the performance of the model across all classes.

Feature Importance

The feature importance plot helps to understand which features contributed the most to the classification model. Here's a plot showing feature importance for the RandomForest model:

plot_feature_importance(feature_importances)

Predictions on New Test Set

The model was used to predict the classes of sound excerpts from the test dataset. The predictions were saved as a CSV file:

Predictions saved to '/content/drive/My Drive/urbansoundclassification/predictions.csv'

Conclusion

- The RandomForestClassifier achieved a test accuracy of 91.54%, showing strong performance in classifying urban sound clips.
- Future improvements could include exploring deep learning approaches like Convolutional Neural Networks (CNNs) for audio data.

Appendix

Code:

```
from google.colab import drive
import pandas as pd

drive.mount('/content/drive')
!ls "/content/drive/My Drive"

#directories
train_labels_df = pd.read_csv('/content/drive/My
Drive/urbansoundclassification/audio datasets2/train.csv')
test_labels_df = pd.read_csv('/content/drive/My
Drive/urbansoundclassification/audio datasets1/test.csv')
import pandas as pd
import soundfile as sf
import librosa
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.model_selection import train_test_split, cross_val_score
import joblib
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Function to extract features from an audio file
def extract_features(file_name):
  try:
    # Load the audio file using soundfile
    audio, sample_rate = sf.read(file_name)

    # Convert to mono if it's stereo
    if len(audio.shape) > 1:
      audio = audio.mean(axis=1)

    # Extract MFCC features using librosa
    mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)

    # Return the mean of the MFCCs
    return mfccs.mean(axis=1)
  except Exception as e:
    print(f"Error processing {file_name}: {e}")
    return None

# Function to process audio directory and extract features
def process_audio_directory(directory):
  features = []
  file_names = []

  if not os.path.exists(directory):
    print(f"Error: Directory '{directory}' not found.")
    return None

  for file_name in os.listdir(directory):
    if file_name.endswith('.wav'):
      file_path = os.path.join(directory, file_name)
      feature_vector = extract_features(file_path)
      if feature_vector is not None:
        features.append(feature_vector)
        file_names.append(file_name)
```

```
if len(features) == 0:
    print(f"No .wav files found in '{directory}' or all files caused errors.")
    return None

  features_df = pd.DataFrame(features)
  features_df['file_name'] = file_names
  return features_df

# Load labels
def load_labels(file_path):
  try:
    return pd.read_csv(file_path)
  except FileNotFoundError:
    print(f"Error: File '{file_path}' not found.")
    return None

# Function to extract ID from file name
def extract_id_from_filename(file_name):
  try:
    return int(file_name.split('.')[0])
  except (ValueError, IndexError):
    return None

# Optional plotting function
def plot_confusion_matrix(conf_matrix, y_labels):
  plt.figure(figsize=(10, 6))
        sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=y_labels, yticklabels=y_labels)
  plt.title('Confusion Matrix')
  plt.xlabel('Predicted')
  plt.ylabel('Actual')
  plt.show()
def plot_feature_importance(importances):
 plt.figure(figsize=(10, 6))
 plt.bar(range(len(importances)), importances)
```

```
  plt.title('Feature Importance')
  plt.xlabel('Feature Index')
  plt.ylabel('Importance')
  plt.show()

def plot_predicted_class_distribution(predictions):
  predictions['predicted_class'].value_counts().plot(kind='bar', figsize=(10, 6))
  plt.title('Predicted Class Distribution on New Test Set')
  plt.xlabel('Class')
  plt.ylabel('Frequency')
  plt.show()

# Set this flag to True to enable plotting
ENABLE_PLOTTING = True

# Process train dataset
train_audio_directory = '/content/drive/My
Drive/urbansoundclassification/audio datasets2/Train'
train_features_df = process_audio_directory(train_audio_directory)

if train_features_df is not None:
  # Load train labels
  train_labels_df = load_labels('/content/drive/My
Drive/urbansoundclassification/audio datasets2/train.csv')

  if train_labels_df is not None:
    # Extract IDs from file names and convert them to integers
    train_features_df['ID'] =
train_features_df['file_name'].apply(extract_id_from_filename)
    train_features_df = train_features_df.dropna(subset=['ID'])
    train_features_df['ID'] = train_features_df['ID'].astype(int)
    train_labels_df['ID'] = train_labels_df['ID'].astype(int)

    # Merge features with labels
    merged_df = pd.merge(train_features_df, train_labels_df, on='ID')

    # Separate features and labels
    X = merged_df.drop(columns=['file_name', 'ID', 'Class']) # Drop non-numeric
columns
```

```
y = merged_df['Class']

 # Split the data into 70% training and 30% testing
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

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

 # Train RandomForestClassifier model with cross-validation
 model = RandomForestClassifier(n_estimators=100) # Using 100 trees

 # Perform 5-fold cross-validation on the training set
    cv_scores = cross_val_score(model, X_train_scaled, y_train, cv=5)

    # Print cross-validation results
    print(f"Cross-validation accuracy scores: {cv_scores}")
    print(f"Average cross-validation accuracy: {cv_scores.mean():.4f}")

    # Train the final model on the entire training set
    model.fit(X_train_scaled, y_train)

    # Predict on the test set
    y_test_pred = model.predict(X_test_scaled)

    # Calculate test accuracy
    test_accuracy = accuracy_score(y_test, y_test_pred)

    # Print test set accuracy
    print(f"Test Set Accuracy: {test_accuracy:.4f}")

    # Confusion Matrix Plot
    if ENABLE_PLOTTING:
      conf_matrix = confusion_matrix(y_test, y_test_pred)
      plot_confusion_matrix(conf_matrix, np.unique(y))
```

```
# Classification Report
    print("\nClassification Report:\n", classification_report(y_test,
y_test_pred))

    # Feature Importance Plot
    if ENABLE_PLOTTING:
      feature_importances = model.feature_importances_
      plot_feature_importance(feature_importances)

    # Save the trained model and scaler after the final iteration
    joblib.dump(model, '/content/drive/My
Drive/urbansoundclassification/random_forest_model.pkl')
    joblib.dump(scaler, '/content/drive/My
Drive/urbansoundclassification/scaler.pkl')

    # Load the saved model and scaler for testing
    model = joblib.load('/content/drive/My
Drive/urbansoundclassification/random_forest_model.pkl')
    scaler = joblib.load('/content/drive/My
Drive/urbansoundclassification/scaler.pkl')

    # Process the test dataset for new predictions
    test_audio_directory = '/content/drive/My
Drive/urbansoundclassification/audio datasets1/Test'
    test_features_df = process_audio_directory(test_audio_directory)

    if test_features_df is not None:
      # Scale new test features
      X_new_test = test_features_df.drop(columns=['file_name'])
      X_new_test_scaled = scaler.transform(X_new_test)

      # Predict on the new test set
      test_predictions = model.predict(X_new_test_scaled)

      # Create a DataFrame for the predictions
      output_df = pd.DataFrame({
        'file_name': test_features_df['file_name'],
        'predicted_class': test_predictions
      })
```

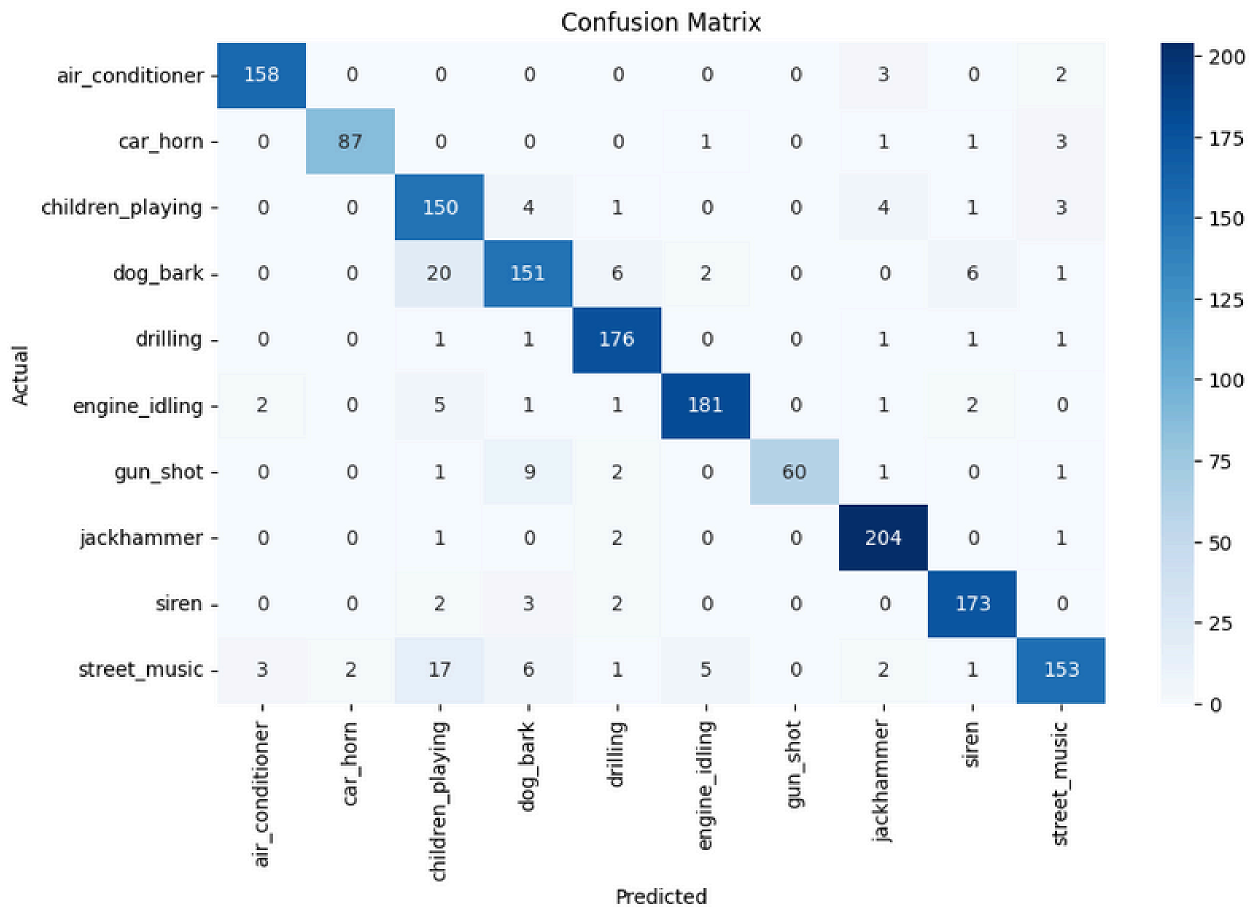
```
 # Save predictions to a CSV file
     output_csv_path = '/content/drive/My Drive/urbansoundclassification/predictions.csv'
     output_df.to_csv(output_csv_path, index=False)
     print(f"Predictions saved to '{output_csv_path}'")

     # Optional: Plot predicted class distribution
     if ENABLE_PLOTTING:
       plot_predicted_class_distribution(output_df)

else:
  print("Error: No valid features extracted from the training dataset.")
```

Output

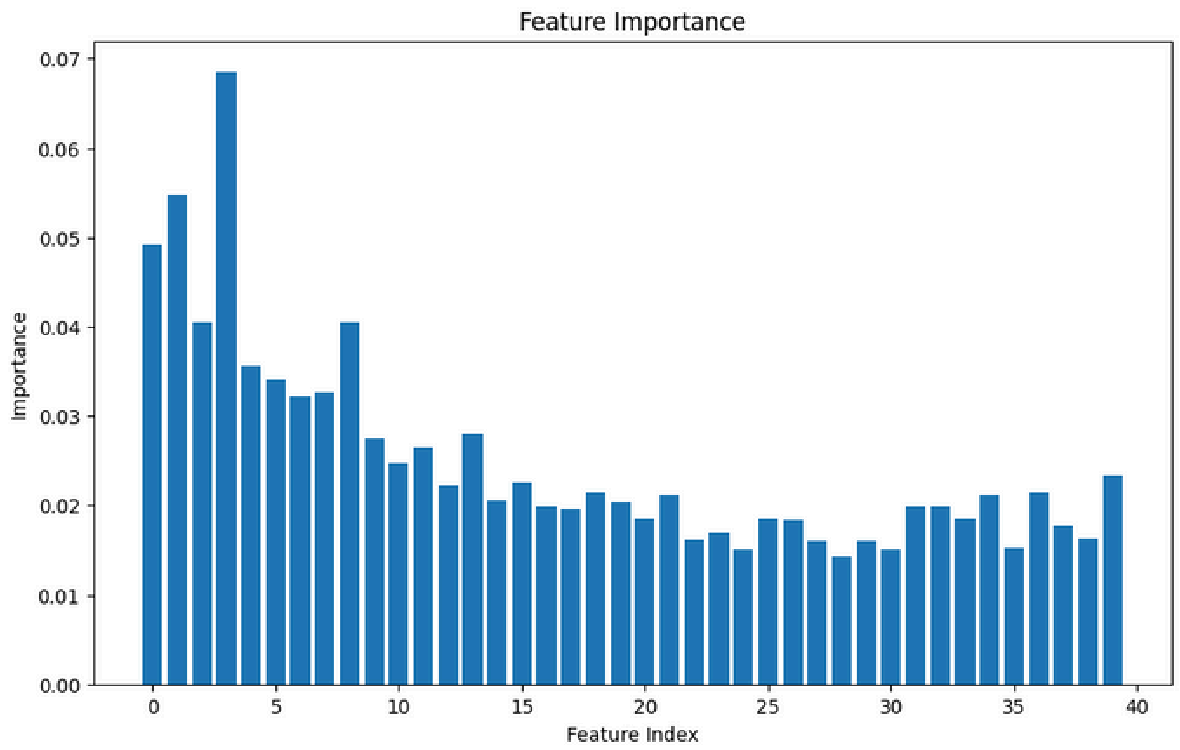
```
Cross-validation accuracy scores: [0.90275953 0.88961892 0.88173456 0.88436268 0.89736842]
Average cross-validation accuracy: 0.8912
Test Set Accuracy: 0.9154
```



```
Classification Report:
                   precision    recall  f1-score   support

 air_conditioner       0.97      0.97      0.97       163
        car_horn       0.98      0.94      0.96        93
children_playing       0.76      0.92      0.83       163
        dog_bark       0.86      0.81      0.84       186
        drilling       0.92      0.97      0.95       181
   engine_idling       0.96      0.94      0.95       193
        gun_shot       1.00      0.81      0.90        74
      jackhammer       0.94      0.98      0.96       208
           siren       0.94      0.96      0.95       180
    street_music       0.93      0.81      0.86       190

        accuracy                           0.92      1631
       macro avg       0.93      0.91      0.92      1631
    weighted avg       0.92      0.92      0.92      1631
```




```
Predictions saved to '/content/drive/My Drive/urbansoundclassification/predictions.csv'
```

prediction.csv - https://drive.google.com/file/d/1-DlVX8vcxZ9BVuAqzdSJrFNZdLtCPAz7/view?usp=drive_link

Dataset Overview

https://drive.google.com/drive/folders/1FfbU5CMjfQwVRz_vFv8L6F8RuhvevQKs?usp=sharing