modelbuild.py

```python
import librosa
from matplotlib import pyplot as plt
import numpy as np
from scipy.signal import butter, lfilter
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from joblib import dump
from joblib import load
import seaborn as sns

def butter_bandpass(lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a

def bandpass_filter(data, lowcut, highcut, fs, order=5):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = lfilter(b, a, data)
    return y

# Example function to handle non-finite values
def handle_non_finite(y):
    if not np.all(np.isfinite(y)):
        y = np.nan_to_num(y)  # Replace NaN with 0 and Inf with large finite numbers
    return y

# Load and Preprocess Audio Files
audio_paths = ['sounds/1 car_1.WAV','sounds/1 car_2.WAV', 'sounds/1 car_3.WAV', 'sounds/1 car_4.WAV', 'sounds/1 car_5.WAV',
               'sounds/2 car_1.WAV','sounds/2 car_2.WAV', 'sounds/2 car_3.WAV', 'sounds/2 car_4.WAV', 'sounds/2 car_5.WAV',
               'sounds/3 car_1.WAV','sounds/3 car_2.WAV', 'sounds/3 car_3.WAV', 'sounds/3 car_4.WAV', 'sounds/3 car_5.WAV',
               'sounds/4 car_1.WAV','sounds/4 car_2.WAV', 'sounds/4 car_3.WAV', 'sounds/4 car_4.WAV', 'sounds/4 car_5.WAV',
               'sounds/5 car_1.WAV','sounds/5 car_2.WAV', 'sounds/5 car_3.WAV', 'sounds/5 car_4.WAV', 'sounds/5 car_5.WAV', ]  # List of your audio files
labels = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5]  # Corresponding labels indicating the number of cars

features = []
for audio_path, label in zip(audio_paths, labels):
    y, sr = librosa.load(audio_path, sr=None)
    filtered_signal = bandpass_filter(y, 20.0, 2000.0, sr, order=6)
    filtered_signal = handle_non_finite(filtered_signal)  # Handle non-finite values
    mfcc = librosa.feature.mfcc(y=filtered_signal, sr=sr, n_mfcc=13)
    mfcc_scaled = np.mean(mfcc.T, axis=0)
    features.append(mfcc_scaled)

# Prepare Data and Train the Model
# Convert the list of features to a NumPy array
```

```
X = np.array(features)
y = np.array(labels)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Save the model
dump(model, 'car_count_model.joblib')

model = load('car_count_model.joblib')

# Predict on the test set and evaluate
y_pred = model.predict(X_test)

# Calculate and print accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")

# Print a classification report
print(classification_report(y_test, y_pred))

# Generate and display a confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='g', cmap='Blues', xticklabels=[1, 2, 3, 4, 5],
yticklabels=[1, 2, 3, 4, 5])
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()


# List of test audio files
test_audio_paths = ['test_sounds/1 car_t.WAV','test_sounds/2 car_t.WAV',
                     'test_sounds/3 car_t.WAV',
                    'test_sounds/4 car_t.WAV',
                    'test_sounds/5 car_t.WAV']
test_labels = [1, 2, 3, 4, 5]  # Actual number of cars for evaluation, if available

predictions = []
# Process each test audio file
for audio_path in test_audio_paths:
    y_test, sr_test = librosa.load(audio_path, sr=None)
    filtered_signal_test = bandpass_filter(y_test, 20.0, 2000.0, sr_test, order=6)
    mfcc_test = librosa.feature.mfcc(y=filtered_signal_test, sr=sr_test, n_mfcc=13)
    mfcc_scaled_test = np.mean(mfcc_test.T, axis=0).reshape(1, -1)  # Reshape for prediction
    prediction = model.predict(mfcc_scaled_test)
    predictions.append(prediction[0])

# Output predictions for the test dataset
for pred, actual in zip(predictions, test_labels):
    print(f"Predicted: {pred}, Actual: {actual}")
```

```
104
105 # Calculate the Mean Absolute Error (MAE) for the test dataset
106 mae_test = mean_absolute_error(test_labels, predictions)
107 print(f"Mean Absolute Error (MAE) on new test set: {mae_test:.2f}")
108
109 # Generate and display a confusion matrix for the new test data
110 test_conf_matrix = confusion_matrix(test_labels, predictions)
111 plt.figure(figsize=(10, 7))
112 sns.heatmap(test_conf_matrix, annot=True, fmt='g', cmap='Blues', xticklabels=[1, 2, 3, 4, 5],
    yticklabels=[1, 2, 3, 4, 5])
113 plt.xlabel('Predicted labels')
114 plt.ylabel('True labels')
115 plt.title('Confusion Matrix for Test Audio Paths')
116 plt.show()
117
118
119 # List of test audio files (differnt posistion)
120 test_audio_paths = ['test_sounds/1 car_noise.WAV','test_sounds/2 car_noise.WAV',
121                     'test_sounds/3 car_noise.WAV',
122                    'test_sounds/4 car_noise.WAV',
123                    'test_sounds/5 car_noise.WAV']
124 test_labels = [1, 2, 3, 4, 5]  # Actual number of cars for evaluation, if available
125
126 predictions = []
127 # Process each test audio file
128 for audio_path in test_audio_paths:
129     y_test, sr_test = librosa.load(audio_path, sr=None)
130     filtered_signal_test = bandpass_filter(y_test, 20.0, 2000.0, sr_test, order=6)
131     mfcc_test = librosa.feature.mfcc(y=filtered_signal_test, sr=sr_test, n_mfcc=13)
132     mfcc_scaled_test = np.mean(mfcc_test.T, axis=0).reshape(1, -1)  # Reshape for prediction
133     prediction = model.predict(mfcc_scaled_test)
134     predictions.append(prediction[0])
135
136 # Output predictions for the test dataset
137 for pred, actual in zip(predictions, test_labels):
138     print(f"Predicted: {pred}, Actual: {actual}")
139
140 # Calculate the Mean Absolute Error (MAE) for the test dataset
141 mae_test = mean_absolute_error(test_labels, predictions)
142 print(f"Mean Absolute Error (MAE) on new test set: {mae_test:.2f}")
143
144 # Generate and display a confusion matrix for the new test data
145 test_conf_matrix = confusion_matrix(test_labels, predictions)
146 plt.figure(figsize=(10, 7))
147 sns.heatmap(test_conf_matrix, annot=True, fmt='g', cmap='Blues', xticklabels=[1, 2, 3, 4, 5],
    yticklabels=[1, 2, 3, 4, 5])
148 plt.xlabel('Predicted labels')
149 plt.ylabel('True labels')
150 plt.title('Confusion Matrix for Test Audio Paths')
151 plt.show()
```