ARTIFICIAL INTELLIGENCE

How can Machine Learning in Audio Analysis?

Learn how machine learning could be used to analyze
generate predictions for both classification and regres

Suhas Maddali
Jan 10, 2023 6 min read

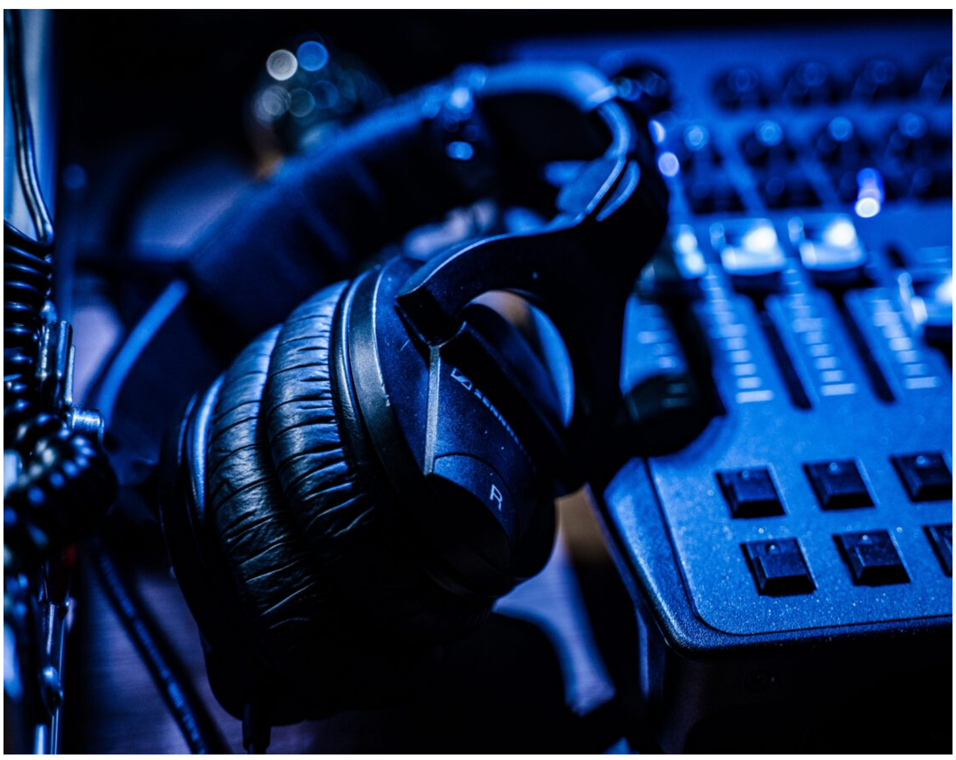

Photo by Jarrod Reed on Unsplash

Machine learning has been gaining rapid tractior
decade. In fact, it is being used in numerous ind
healthcare, **agriculture**, and **manufacturing**. The
potential applications of machine learning being
advancement of technology and computational p
is available in various formats in abundance, it is
use **machine learning** and **data science** to extrac
from data and make predictions using them.

One of the most interesting applications of mach
audio analysis and understanding the quality of

formats respectively. Therefore, using various machine learning and deep learning algorithms ensures that predictions are created and understood with the **audio data**.

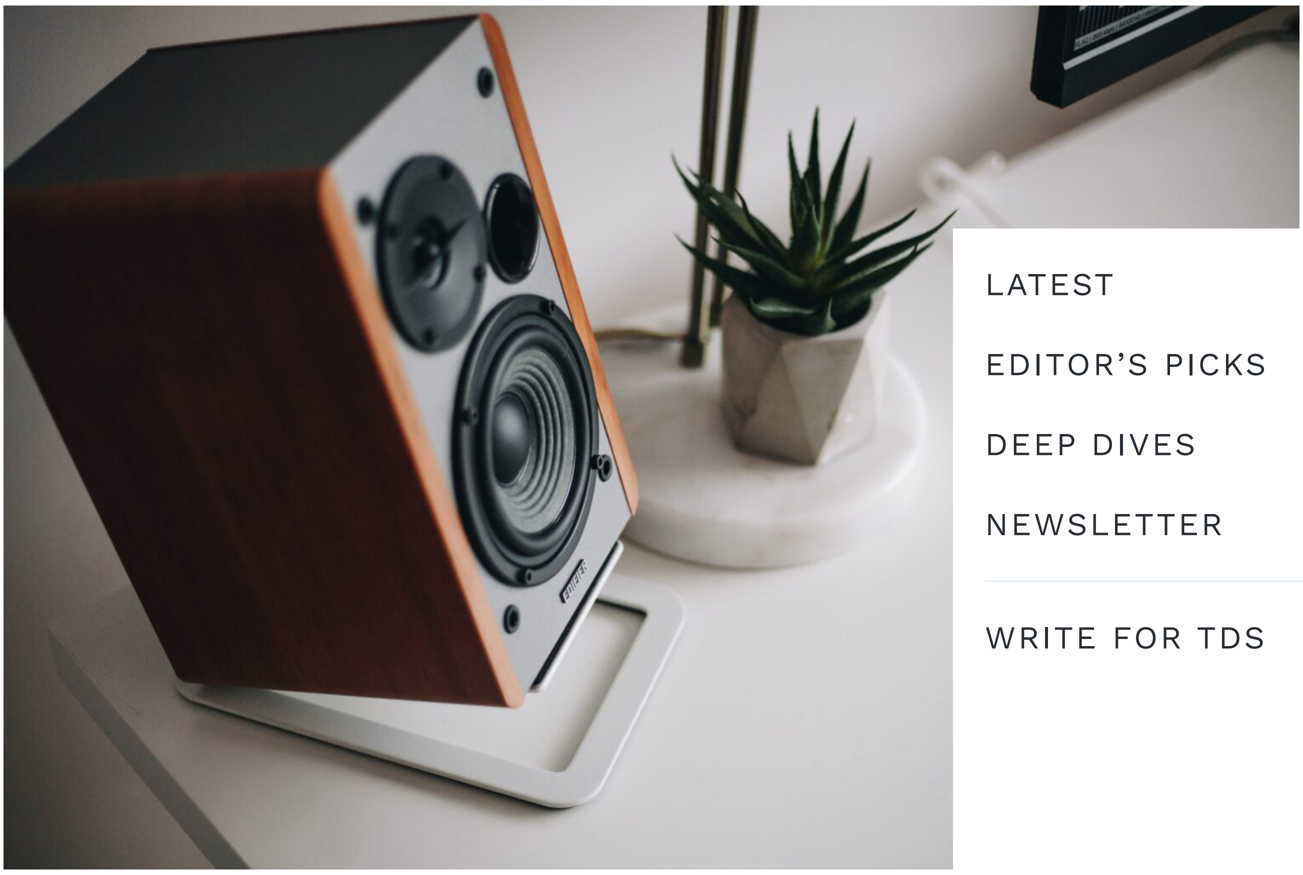

Photo by Med Badr Chemmaoui on Unsplash

Before doing audio analysis, samples of the signa taken and analyzed individually. The rate at whic taking the samples is also known as **the samplir rate**. It would be really handy to convert a time- a frequency domain to get a good logical underst signal along with computing useful components and energy. All these could be given as features learning models which would use them for maki

There is a popular conversion of an audio signal (image) so that it could be given to **convolutiona networks (CNNs)** for prediction. A spectrogram c important characteristics of an audio signal and representations in 2D which can hence be used based networks.

There are plenty of ML models that perform a ve predicting the output labels if they are given an an audio signal which is composed of amplitude different units of frequency, could also be conve and used for robust ML predictions.

In this article, we would be going over how to read an audio file by considering a random example and plotting it to understand its graphical representation. Later, we will perform feature engineering with the image data and perform convolutional operations as the audio is converted to an image. Finally, we will get **sample predictions** for unseen data. Note that this code is used for demonstration and does not take into account specific datasets.

Reading the data

```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.io.wavefile import read

# Read in the audio file
sample_rate, audio_data = read('audio_file.wav')

# Convert the audio data to a numpy array
audio_data = np.array(audio_data)

# Get the length of the audio data
length = audio_data.shape[0] / sample_rate

# Create a time axis
time = np.linspace(0., length, audio_data.shape[0])

# Plot the audio data
plt.plot(time, audio_data)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.show()
```

audio_read.py hosted with ❤ by **GitHub**

We are going to import necessary libraries that a
reading an audio file in the form that is mostly p
'.wav' format. After reading the file, we are gettir
representation as shown in the code cell above.
be plotting the output just to see how it looks w

Feature Engineering

```python
import librosa

# Calculate the short-term Fourier transform (STFT) of the audio c
stft = librosa.stft(audio_data)

# Extract the magnitude and phase of the STFT
magnitude, phase = librosa.magphase(stft)
```

```
# Calculate the mel-scaled spectrogram of the audio data
mel_spec = librosa.feature.melspectrogram(audio_data, sr=sample_rate)

# Extract the Mel-frequency cepstral coefficients (MFCCs) from the mel-scaled spectrogram
mfccs = librosa.feature.mfcc(S=librosa.power_to_db(mel_spec), n_mfcc=20)

# Transpose the MFCCs so that each row represents a time frame and each column represents a
mfccs = mfccs.T
```

audio_feature_engineer.py hosted with ❤ by GitHub view raw

Now that the data is plotted and visualized to se the '.wav' file, we would now be using a popular **'librosa'** that can be used to calculate the short- transform of the audio data. This is to ensure th decomposed into its constituent frequencies an that is widely used in a large number of industri

Training the Models

```
# Convert the audio data to a spectrogram
X = compute_spectrogram(X)

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_si

# Build machine learning model
# In this case, let's use a random forest classifier with 100 tree
model = RandomForestClassifier(n_estimators = 100)

# Train model on training data
model.fit(X_train, y_train)

# Evaluate model on test data
accuracy = model.score(X_test, y_test)
print("Test accuracy:", accuracy)
```

training_audio.py hosted with ❤ by GitHub

Now that we have used 'librosa' to get the frequ components, we would be using machine learnir make predictions. It is to be noted that it is a cl problem and hence, we go ahead with using a **ra classifier**. However, feel free to use any other m model that suits your needs and the business.

We are now going to be using the same code but tasks where the output is **continuous** instead of is the coding cell about how training could be d performance monitored with the help of a rando regressor.

```
# Convert the audio data to a spectrogram
X = compute_spectrogram(X)

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

# Build machine learning model
# In this case, let's use a random forest regressor with 100 trees
model = RandomForestRegressor(n_estimators = 100)

# Train model on training data
model.fit(X_train, y_train)

# Evaluate model on test data
accuracy = model.score(X_test, y_test)
print("Test accuracy:", accuracy)
```

training_regression.py hosted with ❤ by GitHub

Hyperparameter Tuning

It is important to determine the right hyperparar model (Random Forest) before it could be deplo There are a lot of hyperparameters to **search** for to deep neural networks. Since we are using ran our baseline models, we should be able to get tl hyperparameters in a minimal search space. Let perform hyperparameter tuning on the general d

```
# Define hyperparameters to tune
param_grid = {'max_depth': [2, 4, 6, 8], 'min_samples_split': [2,

# Create a grid search object with 5-fold cross-validation
grid_search = GridSearchCV(model, param_grid, cv=5)

# Fit the grid search object to the training data
grid_search.fit(X_train, y_train)

# Print the best hyperparameters and score
print("Best hyperparameters:", grid_search.best_params_)
print("Best cross-validation score:", grid_search.best_score_)

# Evaluate the model on the test data
best_model = grid_search.best_estimator_
accuracy = best_model.score(X_test, y_test)
print("Test accuracy:", accuracy)
```

hyperaparameter_tuning_audio.py hosted with ❤ by GitHub

In the code cell, we specify the **number of estim maximum depth** of the tree that we search for t results on the test set. We finally monitor the score and see how changes in the hyperparameters can lead to better performance by the model.

Model Deployment

```python
import pickle

# Save the model to a file
with open('model.pkl', 'wb') as f:
    pickle.dump(model, f)

# Load the model from a file
with open('model.pkl', 'rb') as f:
    model = pickle.load(f)

# Use the model to make predictions on new data
predictions = model.predict(new_data)
```

audio_deployment.py hosted with ❤ by **GitHub**

LATEST
EDITOR'S PICKS
DEEP DIVES
NEWSLETTER
WRITE FOR TDS

Now that we have performed hyperparameter tu
most accurate predictions, it is time to save the
models that provided the best results. Therefore
using the **pickle** library in python so that we wou
freedom to save the machine learning model tha
used for serving.

After saving the model, we would again load it w
building a **production-ready** code and use it to r
on incoming batches or streams of data. It is to
set of steps that were used to perform featuriza
training data must also be performed on the tes
there is no skew in the data.

Constant Monitoring

We know that the model is performing a good jo
where it is receiving incoming data from users, it
important yet neglected step which is to **monito**
quality of the models. There are often scenarios
models might not be performing as they were du
This can be because there is a difference betwe
and serving data. For example, there can be situ
concept drift or **data drift** that can have a signifi
the performance of the inference model that is

production.

Constant monitoring ensures that steps are taken to check the predictive models and understand their behavior with changing data. If the predictions are the **least accurate** and it is leading to a loss of revenue for the business, steps should be taken to again train the models with this deviant data so that there is no unexpected change in the behavior of the models respectively.

Conclusion

After going through this article, you might have g about how to perform machine learning for **audi** understand the overall workflow. We have seen s reading the data, feature engineering, training th hyperparameter tuning, model deployment along monitoring. Applying each of these steps and en are no mistakes when developing a pipeline wou robust machine learning production system.

Below are the ways where you could contact me my work.

GitHub: *suhasmaddali (Suhas Maddali) (github.c*

_YouTube: https://www.youtube.com/channel/UCymdyoyJBC_i7QVfbrIs-4Q_

LinkedIn: *(1) Suhas Maddali, Northeastern Univers | LinkedIn*

Medium: *Suhas Maddali – Medium*

• • •

WRITTEN BY

Suhas Maddali

See all from Suhas Maddali