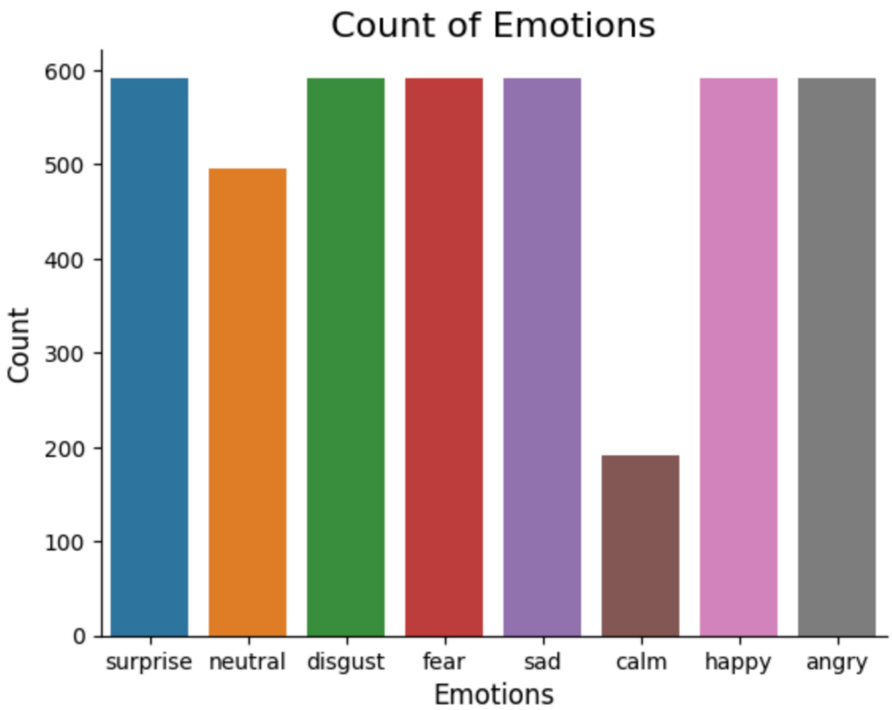
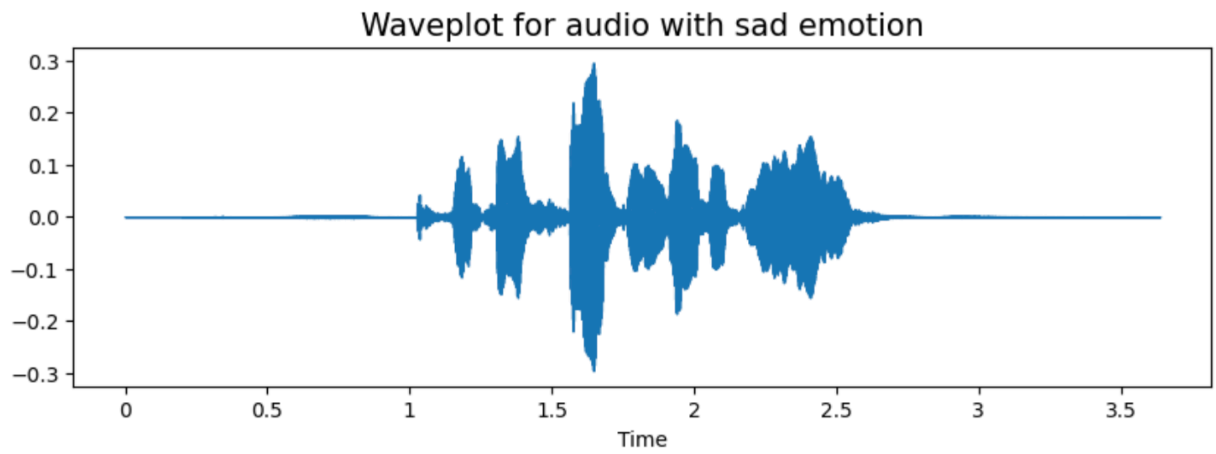
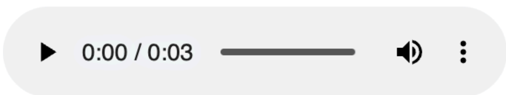
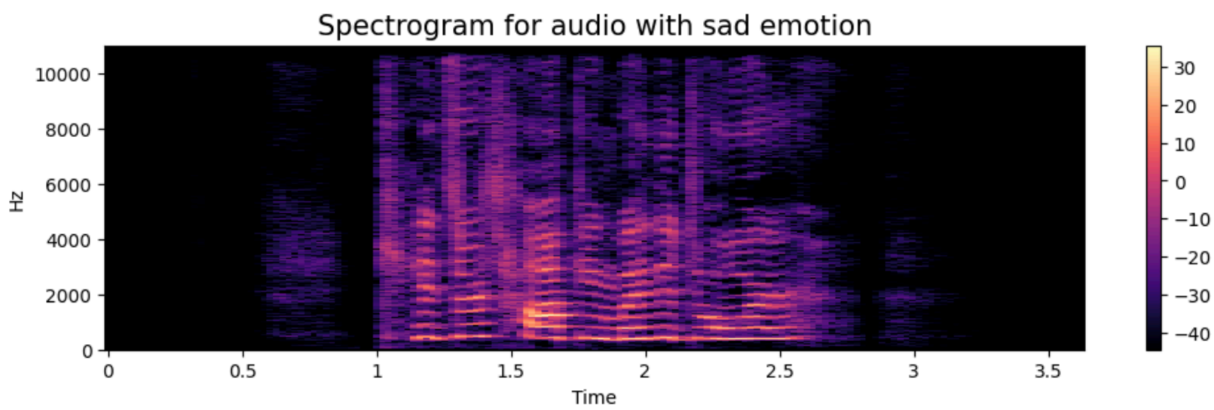
[11]:

```
plt.title('Count of Emotions', size=16)
sns.countplot(x='Emotions', data=data_path)
plt.ylabel('Count', size=12)
plt.xlabel('Emotions', size=12)
sns.despine(top=True, right=True, left=False, bottom=False)
plt.show()
```



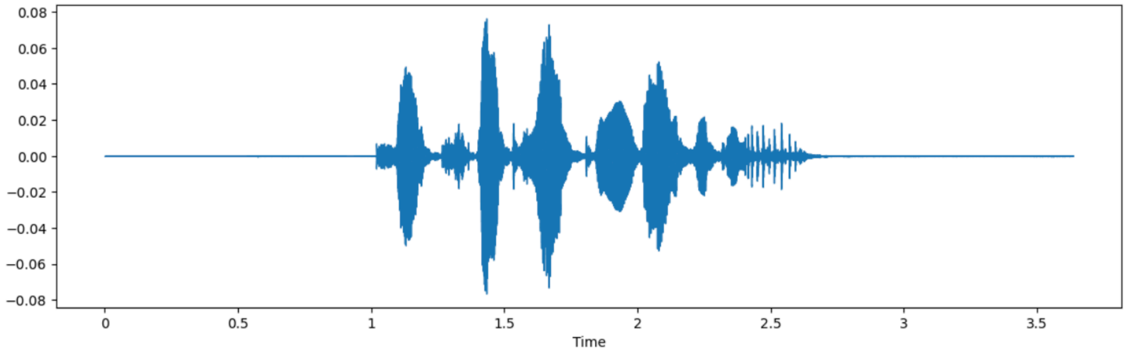



[15]:

```
plt.figure(figsize=(14,4))
librosa.display.waveshow(y=data, sr=sample_rate)
Audio(path)
```

[15…

0:00 / 0:03




[16]:

```
x = noise(data)
plt.figure(figsize=(14,4))
librosa.display.waveshow(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

[16…

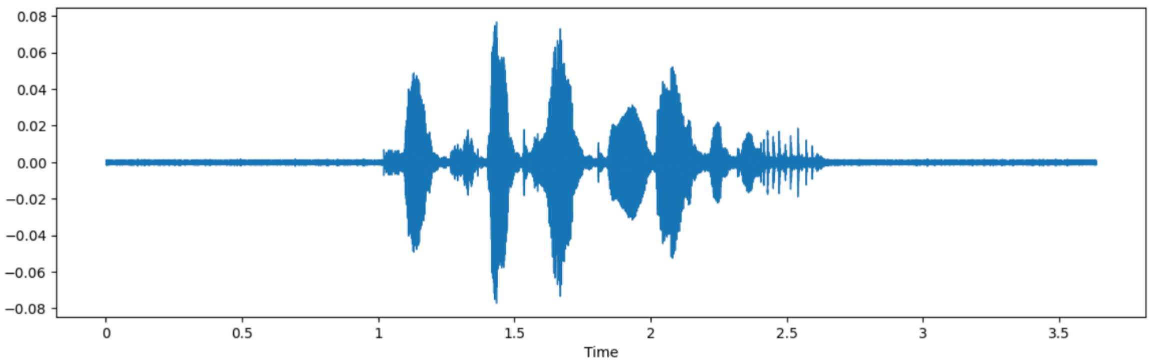
0:00 / 0:03

[16]:

```
x = noise(data)
plt.figure(figsize=(14,4))
librosa.display.waveshow(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

[16…

0:00 / 0:03




+ Code  + Markdown

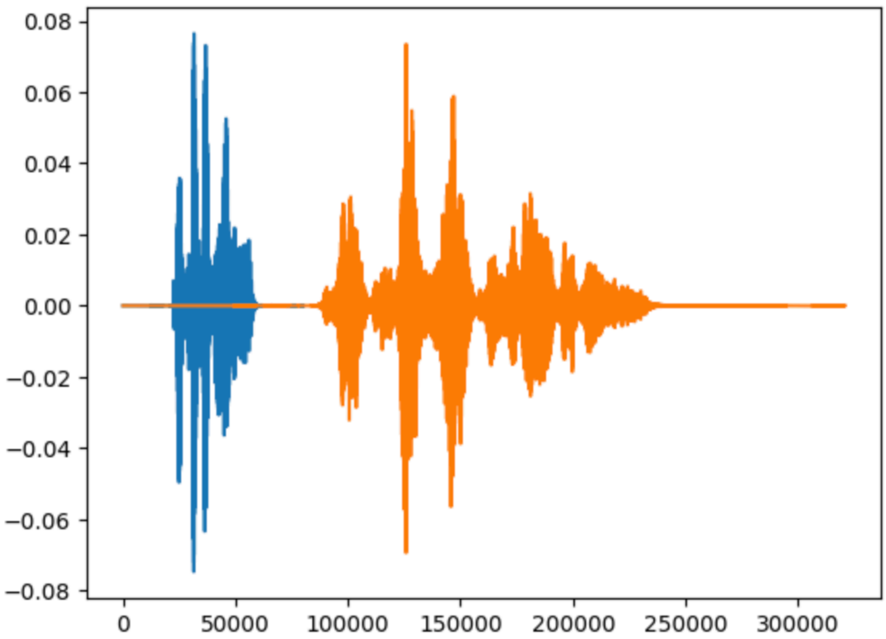
[17]:

```
import numpy as np, librosa

x = data
y = librosa.effects.time_stretch(data, rate=4000/16000)
plt.plot(x) # plotted in blue
plt.plot(y)
```

[17…

```
[<matplotlib.lines.Line2D at 0x7eef06830a60>]
```

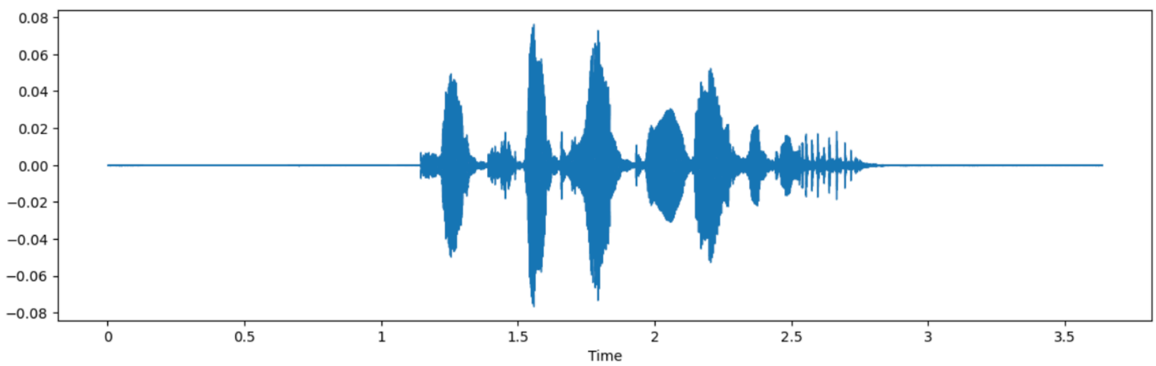



[18]:

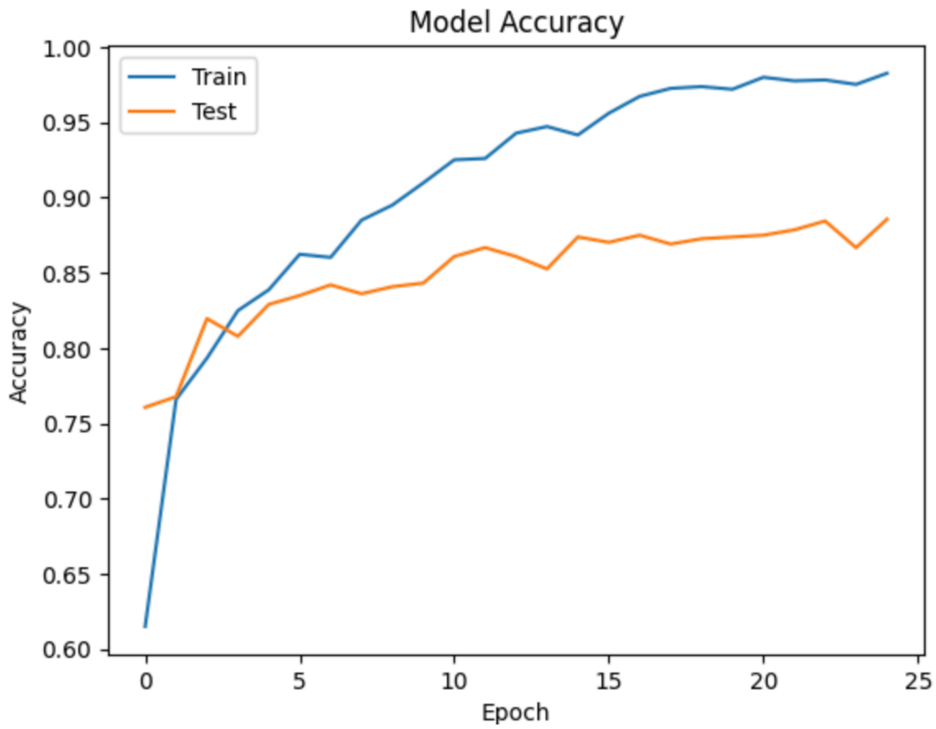
```
x = shift(data)
plt.figure(figsize=(14,4))
librosa.display.waveshow(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

[18…

0:00 / 0:03



```python
import matplotlib.pyplot as plt
# Plot model accuracy over epochs
plt.plot(cnn_results.history['accuracy'])
plt.plot(cnn_results.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```




[41]:

```python
# Calculate pre-training accuracy
score = CNN_model.evaluate(x_testcnn, y_test_lb, verbose=1)
accuracy = 100*score[1]

print("Pre-training accuracy: %.4f%%" % accuracy)
```

```
27/27 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.8786 - loss: 0.5681
Pre-training accuracy: 88.5613%
```

[42]:

```python
# Evaluating the model on the training and testing set
score = CNN_model.evaluate(x_traincnn, y_train_lb, verbose=0)
print("Training Accuracy: ", score[1])

score = CNN_model.evaluate(x_testcnn, y_test_lb, verbose=0)
print("Testing Accuracy: ", score[1])
```

```
Training Accuracy:  0.9973466992378235
Testing Accuracy:  0.885613203048706
```

[50]:

```
#Add labels to confusion matrix
confusion_matrix = pd.DataFrame(confusion_matrix, columns=list(y_labels_encoded.value

print("The rows represents the true values or observations")
print("The columns respresent the model's predictions")
#Print confusion matrix results
confusion_matrix
```

```
The rows represents the true values or observations
The columns respresent the model's predictions
```

[50…

| | angry | calm | disgust | fear | happy | neutral | sad | surprise |
|---|---|---|---|---|---|---|---|---|
| **angry** | 0 | 0 | 2 | 1 | 4 | 1 | 1 | 2 |
| **calm** | 0 | 0 | 2 | 1 | 1 | 5 | 3 | 1 |
| **disgust** | 1 | 1 | 0 | 5 | 1 | 1 | 0 | 3 |
| **fear** | 2 | 0 | 0 | 0 | 1 | 0 | 4 | 2 |
| **happy** | 3 | 0 | 5 | 6 | 0 | 0 | 2 | 0 |
| **neutral** | 0 | 2 | 1 | 0 | 1 | 0 | 1 | 0 |
| **sad** | 0 | 4 | 2 | 5 | 0 | 5 | 0 | 1 |
| **surprise** | 1 | 0 | 3 | 1 | 4 | 5 | 0 | 0 |

[51]:

```
#Plot confusion matrix with results
ax = sns.heatmap(confusion_matrix, annot=True)
```

[53]:

```python
# Select one unit of data from x_testcnn
test_data_unit = x_testcnn[0:1]

# Make predictions
predictions = CNN_model.predict(test_data_unit)

# Get the index of the maximum prediction
predicted_class_index = np.argmax(predictions)

# Map the index to the emotion label using the label encoder
predicted_emotion = lb.inverse_transform([predicted_class_index])[0]

# Print the predicted emotion
print("Predicted Emotion:", predicted_emotion)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
Predicted Emotion: angry
```

```
34/34 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.8100 - loss: 0.5933
Accuracy of our model on test data : 79.25%
```