Name : Arul kumar ARK

Roll No.: 225229103

Lab : 08 : Audio corpus creation and binary classification using DNN

In [21]:

```
import numpy as np
import pandas as pd
```

In [22]:

```
from IPython.display import Audio
import IPython.display as ipd
```

In [23]:

```
import warnings
warnings.filterwarnings('ignore')
```

Read the Audio

In [24]:

```
paths = 'Audio/cow1.mp3'
Audio(paths)
```

Out[24]:

0:03 / 0:03

In [25]:

```
import librosa
import librosa.display
```

In [26]:

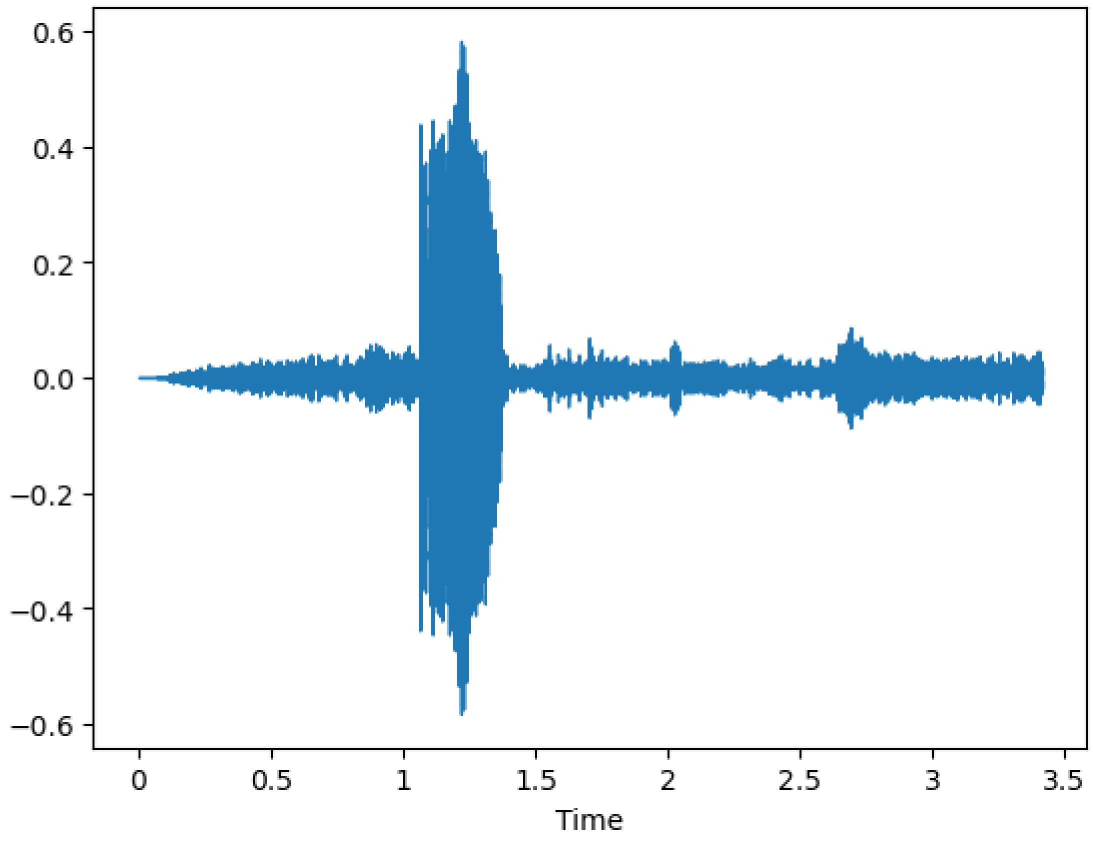
```
import glob
```

In [27]:

```
data, sample_rate = librosa.load(paths)
librosa.display.waveshow(data, sr=sample_rate)
```

Out[27]:

```
<librosa.display.AdaptiveWaveplot at 0x1df154242d0>
```



In [28]:

```
data
```

Out[28]:

```
array([0.        , 0.        , 0.        , ..., 0.00351309, 0.01000754,
       0.01899757], dtype=float32)
```

In [29]:

```
stftt = librosa.feature.chroma_stft(y=data, sr=sample_rate)
```

In [30]:

```
stftt.shape
```

Out[30]:

```
(12, 148)
```

Make Metadata

In [31]:

```
dataset = pd.read_csv(r'cow_how.csv')
```

In [32]:

```
dataset.sample(6)
```

Out[32]:

	filename	label	class
10	audio\how1.mp3	1	How
2	audio\cow8.mp3	0	Cow
6	audio\cow4.mp3	0	Cow
18	audio\how9.mp3	1	How
4	audio\cow6.mp3	0	Cow
11	audio\how2.mp3	1	How

Features Extractor

In [33]:

```
from tqdm import tqdm
```

In [34]:

```
df=[]
```

In [35]:

```
def features_extractor(file):
    audio, sample_rate = librosa.load(file_name)
    stftt_features = librosa.feature.chroma_stft(y=audio, sr=sample_rate)
    stftt_scaled_features = np.mean(stftt_features.T, axis=0)
    return stftt_scaled_features
```

In [36]:

```
for index_num, row in tqdm(dataset.iterrows()):
    file_name = row[0]
    final_class_labels = row[1]
    data = features_extractor(file_name)
    df.append([data, final_class_labels])
```

```
20it [00:00, 22.52it/s]
```

In [37]:

```
#main
extracted_df = pd.DataFrame(df, columns=['feature','class'])
```

Train and Test

In [38]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
```

In [39]:

```
X = np.array(extracted_df['feature'].tolist())
y = np.array(extracted_df['class'])
```

In [40]:

```
X,y
```

Out[40]:

```
(array([[0.532438  , 0.4610948 , 0.42987955, 0.46089712, 0.47214538,
         0.6074419 , 0.6476855 , 0.5700796 , 0.5691059 , 0.57608545,
         0.49810052, 0.5037453 ],
        [0.29863465, 0.30551946, 0.41972452, 0.45416725, 0.47523117,
         0.591336  , 0.42310604, 0.34804145, 0.3662879 , 0.40263608,
         0.53428924, 0.40486145],
        [0.50849074, 0.5523569 , 0.5826741 , 0.58775944, 0.49763504,
         0.5274342 , 0.5794037 , 0.5698282 , 0.56291354, 0.5574038 ,
         0.47757378, 0.49834406],
        [0.38663936, 0.39490664, 0.48261943, 0.554509  , 0.5765289 ,
         0.5758111 , 0.51372266, 0.56730425, 0.5301389 , 0.4713307 ,
         0.46951917, 0.41431326],
        [0.32363352, 0.22415118, 0.23891369, 0.2472624 , 0.2877818 ,
         0.5055353 , 0.52030814, 0.31256944, 0.3691179 , 0.509971  ,
         0.5328319 , 0.49773958],
        [0.30538115, 0.385899  , 0.37250605, 0.39885387, 0.47371554,
         0.60399866, 0.67706925, 0.47763428, 0.40000203, 0.3410914 ,
```

In [41]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=
```

In [42]:

```
print("Train : ")
X_train.shape,y_train.shape
```

```
Train :
```

Out[42]:

```
((15, 12), (15,))
```

In [43]:

```
print("Test : ")
X_test.shape,y_test.shape
```

```
Test :
```

Out[43]:

```
((5, 12), (5,))
```

Model

In [44]:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.optimizers import Adam, RMSprop
from tensorflow.keras.utils import to_categorical
```

In [45]:

```
import time
```

In [46]:

```
model = Sequential()
```

In [47]:

```
model.add(Dense(128, activation='relu', input_shape=(12,)))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='mean_squared_error', metrics=['accuracy'], optimizer='adam')
```

In [48]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 128)               1664      
                                                                 
 dense_1 (Dense)             (None, 64)                8256      
                                                                 
 dense_2 (Dense)             (None, 32)                2080      
                                                                 
 dense_3 (Dense)             (None, 16)                528       
                                                                 
 dense_4 (Dense)             (None, 8)                 136       
                                                                 
 dense_5 (Dense)             (None, 1)                 9         
                                                                 
=================================================================
Total params: 12673 (49.50 KB)
Trainable params: 12673 (49.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [49]:

```
start_time = time.time()
history = model.fit(X_train, y_train, batch_size=132, epochs=50 , verbose=2, validation_
end_time = time.time()
```

```
1/1 - 2s - loss: 0.2522 - accuracy: 0.3333 - val_loss: 0.2501 - val_ac
curacy: 0.3333 - 2s/epoch - 2s/step
Epoch 2/50
1/1 - 0s - loss: 0.2502 - accuracy: 0.5000 - val_loss: 0.2515 - val_ac
curacy: 0.3333 - 55ms/epoch - 55ms/step
Epoch 3/50
1/1 - 0s - loss: 0.2491 - accuracy: 0.5833 - val_loss: 0.2528 - val_ac
curacy: 0.3333 - 66ms/epoch - 66ms/step
Epoch 4/50
1/1 - 0s - loss: 0.2481 - accuracy: 0.5833 - val_loss: 0.2525 - val_ac
curacy: 0.3333 - 77ms/epoch - 77ms/step
Epoch 5/50
1/1 - 0s - loss: 0.2476 - accuracy: 0.5833 - val_loss: 0.2538 - val_ac
curacy: 0.3333 - 64ms/epoch - 64ms/step
Epoch 6/50
1/1 - 0s - loss: 0.2468 - accuracy: 0.5833 - val_loss: 0.2556 - val_ac
curacy: 0.3333 - 55ms/epoch - 55ms/step
Epoch 7/50
1/1 - 0s - loss: 0.2458 - accuracy: 0.5833 - val_loss: 0.2578 - val_ac
curacy: 0.3333 - 44ms/epoch - 44ms/step
```

In [50]:

```
al = model.evaluate(X_test, y_test, verbose=0)
print("Test Loss:",  al[0])
print("Test Accuracy:", al[1])
training_time = end_time - start_time
print("Training Time: {:.2f} seconds".format(training_time))
```

```
Test Loss: 0.22703103721141815
Test Accuracy: 0.4000000059604645
Training Time: 5.80 seconds
```

Different Models

In [51]:

```
def models(dense):
    model = Sequential()
    model.add(Dense(128, activation='relu', input_shape=(12,)))
    model.add(Dense(dense, activation='relu'))
    model.add(Dense(dense, activation='relu'))
    model.add(Dense(dense, activation='relu'))
    model.add(Dense(8, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='mean_squared_error', metrics=['accuracy'], optimizer='adam')
    print()
    model.summary()
    start_time = time.time()
    history = model.fit(X_train, y_train, batch_size=132, epochs=50 , verbose=2, validat
    end_time = time.time()
    al1 = model.evaluate(X_test, y_test, verbose=0)
    print()
    print("Test Loss:",  al1[0])
    print("Test Accuracy:", al1[1])
    training_time1 = end_time - start_time
    print("Training Time: {:.2f} seconds".format(training_time1))
```

Dense : 8

In [52]:

```
models(8)
```

```
curacy: 0.3333 - 54ms/epoch - 54ms/step
Epoch 23/50
1/1 - 0s - loss: 0.2459 - accuracy: 0.5833 - val_loss: 0.2617 - val_ac
curacy: 0.3333 - 45ms/epoch - 45ms/step
Epoch 24/50
1/1 - 0s - loss: 0.2458 - accuracy: 0.5833 - val_loss: 0.2624 - val_ac
curacy: 0.3333 - 67ms/epoch - 67ms/step
Epoch 25/50
1/1 - 0s - loss: 0.2456 - accuracy: 0.5833 - val_loss: 0.2631 - val_ac
curacy: 0.3333 - 54ms/epoch - 54ms/step
Epoch 26/50
1/1 - 0s - loss: 0.2454 - accuracy: 0.5833 - val_loss: 0.2638 - val_ac
curacy: 0.3333 - 71ms/epoch - 71ms/step
Epoch 27/50
1/1 - 0s - loss: 0.2453 - accuracy: 0.5833 - val_loss: 0.2645 - val_ac
curacy: 0.3333 - 64ms/epoch - 64ms/step
Epoch 28/50
1/1 - 0s - loss: 0.2451 - accuracy: 0.5833 - val_loss: 0.2652 - val_ac
curacy: 0.3333 - 57ms/epoch - 57ms/step
Epoch 29/50
```

Dense : 16

In [53]:

```
models(16)
```

```
curacy: 0.3333 - 47ms/epoch - 47ms/step
Epoch 23/50
1/1 - 0s - loss: 0.2402 - accuracy: 0.5833 - val_loss: 0.2858 - val_ac
curacy: 0.3333 - 55ms/epoch - 55ms/step
Epoch 24/50
1/1 - 0s - loss: 0.2398 - accuracy: 0.5833 - val_loss: 0.2858 - val_ac
curacy: 0.3333 - 55ms/epoch - 55ms/step
Epoch 25/50
1/1 - 0s - loss: 0.2395 - accuracy: 0.5833 - val_loss: 0.2854 - val_ac
curacy: 0.3333 - 64ms/epoch - 64ms/step
Epoch 26/50
1/1 - 0s - loss: 0.2390 - accuracy: 0.5833 - val_loss: 0.2847 - val_ac
curacy: 0.3333 - 67ms/epoch - 67ms/step
Epoch 27/50
1/1 - 0s - loss: 0.2387 - accuracy: 0.5833 - val_loss: 0.2843 - val_ac
curacy: 0.3333 - 63ms/epoch - 63ms/step
Epoch 28/50
1/1 - 0s - loss: 0.2382 - accuracy: 0.5833 - val_loss: 0.2845 - val_ac
curacy: 0.3333 - 58ms/epoch - 58ms/step
Epoch 29/50
```

Dense : 32

In [54]:

```
models(32)
```

```
curacy: 0.3333 - 76ms/epoch - 76ms/step
Epoch 23/50
1/1 - 0s - loss: 0.2385 - accuracy: 0.5833 - val_loss: 0.2767 - val_ac
curacy: 0.3333 - 52ms/epoch - 52ms/step
Epoch 24/50
1/1 - 0s - loss: 0.2380 - accuracy: 0.5833 - val_loss: 0.2774 - val_ac
curacy: 0.3333 - 92ms/epoch - 92ms/step
Epoch 25/50
1/1 - 0s - loss: 0.2374 - accuracy: 0.5833 - val_loss: 0.2778 - val_ac
curacy: 0.3333 - 71ms/epoch - 71ms/step
Epoch 26/50
1/1 - 0s - loss: 0.2368 - accuracy: 0.5833 - val_loss: 0.2779 - val_ac
curacy: 0.3333 - 68ms/epoch - 68ms/step
Epoch 27/50
1/1 - 0s - loss: 0.2361 - accuracy: 0.5833 - val_loss: 0.2778 - val_ac
curacy: 0.3333 - 65ms/epoch - 65ms/step
Epoch 28/50
1/1 - 0s - loss: 0.2354 - accuracy: 0.5833 - val_loss: 0.2776 - val_ac
curacy: 0.3333 - 73ms/epoch - 73ms/step
Epoch 29/50
```

Dense : 64

In [55]:

```
models(64)
```

```
curacy: 0.3333 - 69ms/epoch - 69ms/step
Epoch 23/50
1/1 - 0s - loss: 0.2402 - accuracy: 0.5833 - val_loss: 0.2873 - val_ac
curacy: 0.3333 - 69ms/epoch - 69ms/step
Epoch 24/50
1/1 - 0s - loss: 0.2397 - accuracy: 0.5833 - val_loss: 0.2860 - val_ac
curacy: 0.3333 - 54ms/epoch - 54ms/step
Epoch 25/50
1/1 - 0s - loss: 0.2393 - accuracy: 0.5833 - val_loss: 0.2844 - val_ac
curacy: 0.3333 - 59ms/epoch - 59ms/step
Epoch 26/50
1/1 - 0s - loss: 0.2387 - accuracy: 0.5833 - val_loss: 0.2827 - val_ac
curacy: 0.3333 - 74ms/epoch - 74ms/step
Epoch 27/50
1/1 - 0s - loss: 0.2382 - accuracy: 0.5833 - val_loss: 0.2809 - val_ac
curacy: 0.3333 - 69ms/epoch - 69ms/step
Epoch 28/50
1/1 - 0s - loss: 0.2376 - accuracy: 0.5833 - val_loss: 0.2795 - val_ac
curacy: 0.3333 - 65ms/epoch - 65ms/step
Epoch 29/50
```

Layers

In [56]:

```
def layer(n):
    model=Sequential()
    model.add(Dense(128, activation='tanh', input_shape=(12,)))
    for i in range(0,n):
        model.add(Dense(32, activation='tanh'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='mean_squared_error',metrics=['accuracy'],optimizer='adam')
    print()
    model.summary()
    start_time = time.time()
    history = model.fit(X_train, y_train, batch_size=132, epochs=50 , verbose=2, validat
    end_time = time.time()
    al1 = model.evaluate(X_test, y_test, verbose=0)
    print()
    print("Test Loss:",  al1[0])
    print("Test Accuracy:", al1[1])
    training_time1 = end_time - start_time
    print("Training Time: {:.2f} seconds".format(training_time1))
```

Layer : 2

In [57]:

```
layer(2)
```

```
Epoch 24/50
1/1 - 0s - loss: 0.1874 - accuracy: 0.7500 - val_loss: 0.2313 - val_ac
curacy: 0.6667 - 63ms/epoch - 63ms/step
Epoch 25/50
1/1 - 0s - loss: 0.1840 - accuracy: 0.7500 - val_loss: 0.2299 - val_ac
curacy: 0.6667 - 57ms/epoch - 57ms/step
Epoch 26/50
1/1 - 0s - loss: 0.1806 - accuracy: 0.8333 - val_loss: 0.2271 - val_ac
curacy: 0.6667 - 52ms/epoch - 52ms/step
Epoch 27/50
1/1 - 0s - loss: 0.1772 - accuracy: 0.8333 - val_loss: 0.2234 - val_ac
curacy: 0.6667 - 54ms/epoch - 54ms/step
Epoch 28/50
1/1 - 0s - loss: 0.1737 - accuracy: 0.8333 - val_loss: 0.2200 - val_ac
curacy: 0.6667 - 63ms/epoch - 63ms/step
Epoch 29/50
1/1 - 0s - loss: 0.1702 - accuracy: 0.8333 - val_loss: 0.2178 - val_ac
curacy: 0.6667 - 62ms/epoch - 62ms/step
Epoch 30/50
1/1 - 0s - loss: 0.1667 - accuracy: 0.8333 - val_loss: 0.2164 - val_ac
```

Layer : 3

In [58]:

```
layer(3)
```

```
1/1 - 0s - loss: 0.1901 - accuracy: 0.8333 - val_loss: 0.2411 - val_ac
curacy: 0.6667 - 50ms/epoch - 50ms/step
Epoch 24/50
1/1 - 0s - loss: 0.1859 - accuracy: 0.8333 - val_loss: 0.2412 - val_ac
curacy: 0.6667 - 56ms/epoch - 56ms/step
Epoch 25/50
1/1 - 0s - loss: 0.1815 - accuracy: 0.8333 - val_loss: 0.2409 - val_ac
curacy: 0.3333 - 60ms/epoch - 60ms/step
Epoch 26/50
1/1 - 0s - loss: 0.1771 - accuracy: 0.8333 - val_loss: 0.2372 - val_ac
curacy: 0.6667 - 57ms/epoch - 57ms/step
Epoch 27/50
1/1 - 0s - loss: 0.1725 - accuracy: 0.8333 - val_loss: 0.2317 - val_ac
curacy: 0.6667 - 62ms/epoch - 62ms/step
Epoch 28/50
1/1 - 0s - loss: 0.1678 - accuracy: 0.8333 - val_loss: 0.2280 - val_ac
curacy: 0.6667 - 54ms/epoch - 54ms/step
Epoch 29/50
1/1 - 0s - loss: 0.1631 - accuracy: 0.8333 - val_loss: 0.2269 - val_ac
curacy: 0.6667 - 49ms/epoch - 49ms/step
```

Layer : 4

In [59]:

```
layer(4)
```

```
1/1 - 0s - loss: 0.1721 - accuracy: 0.8333 - val_loss: 0.2172 - val_ac
curacy: 0.6667 - 41ms/epoch - 41ms/step
Epoch 23/50
1/1 - 0s - loss: 0.1665 - accuracy: 0.8333 - val_loss: 0.2172 - val_ac
curacy: 0.6667 - 46ms/epoch - 46ms/step
Epoch 24/50
1/1 - 0s - loss: 0.1612 - accuracy: 0.8333 - val_loss: 0.2138 - val_ac
curacy: 0.6667 - 61ms/epoch - 61ms/step
Epoch 25/50
1/1 - 0s - loss: 0.1561 - accuracy: 0.8333 - val_loss: 0.2071 - val_ac
curacy: 0.6667 - 49ms/epoch - 49ms/step
Epoch 26/50
1/1 - 0s - loss: 0.1508 - accuracy: 0.8333 - val_loss: 0.2037 - val_ac
curacy: 0.6667 - 55ms/epoch - 55ms/step
Epoch 27/50
1/1 - 0s - loss: 0.1462 - accuracy: 0.8333 - val_loss: 0.2013 - val_ac
curacy: 0.6667 - 49ms/epoch - 49ms/step
Epoch 28/50
1/1 - 0s - loss: 0.1411 - accuracy: 0.8333 - val_loss: 0.1988 - val_ac
curacy: 0.6667 - 45ms/epoch - 45ms/step
```

Layer : 5

In [60]:

```
layer(5)
```

```
Epoch 22/50
1/1 - 0s - loss: 0.1498 - accuracy: 0.7500 - val_loss: 0.2110 - val_ac
curacy: 0.6667 - 55ms/epoch - 55ms/step
Epoch 23/50
1/1 - 0s - loss: 0.1458 - accuracy: 0.7500 - val_loss: 0.2080 - val_ac
curacy: 0.6667 - 70ms/epoch - 70ms/step
Epoch 24/50
1/1 - 0s - loss: 0.1412 - accuracy: 0.8333 - val_loss: 0.2056 - val_ac
curacy: 0.6667 - 55ms/epoch - 55ms/step
Epoch 25/50
1/1 - 0s - loss: 0.1373 - accuracy: 0.8333 - val_loss: 0.2084 - val_ac
curacy: 0.6667 - 57ms/epoch - 57ms/step
Epoch 26/50
1/1 - 0s - loss: 0.1326 - accuracy: 0.8333 - val_loss: 0.2133 - val_ac
curacy: 0.6667 - 64ms/epoch - 64ms/step
Epoch 27/50
1/1 - 0s - loss: 0.1286 - accuracy: 0.8333 - val_loss: 0.2094 - val_ac
curacy: 0.6667 - 73ms/epoch - 73ms/step
Epoch 28/50
1/1 - 0s - loss: 0.1236 - accuracy: 0.8333 - val_loss: 0.2051 - val_ac
```

In [61]:

```
import matplotlib.pyplot as plt
```

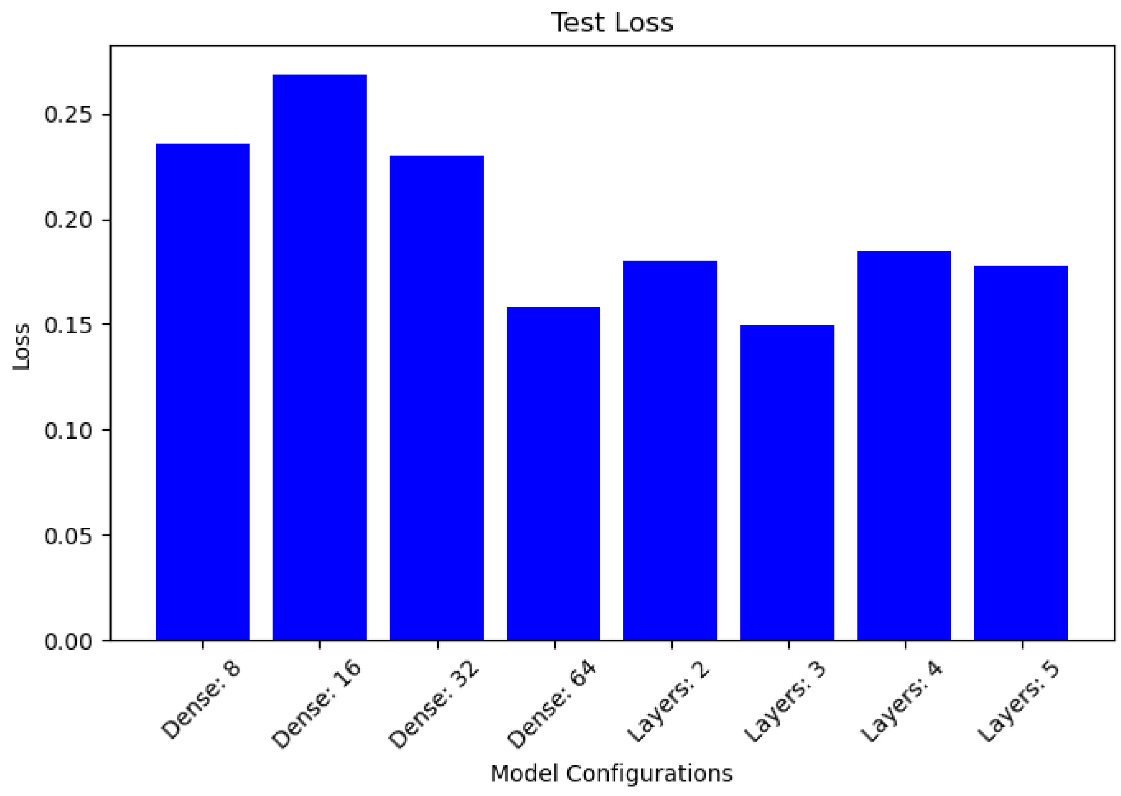
In [62]:

```
models = [
    ("Dense: 8", 0.23565156757831573, 0.6000000238418579),
    ("Dense: 16", 0.2690039575099945, 0.4000000059604645),
    ("Dense: 32", 0.23060576617717743, 0.4000000059604645),
    ("Dense: 64", 0.1585158407688141, 0.800000011920929),
    ("Layers: 2", 0.1802176535129547, 0.800000011920929),
    ("Layers: 3", 0.14923930168151855, 0.800000011920929),
    ("Layers: 4", 0.18485543131828308, 0.800000011920929),
    ("Layers: 5", 0.17763516306877136, 0.800000011920929)]

labels = [model[0] for model in models]
test_loss = [model[1] for model in models]
test_accuracy = [model[2] for model in models]
```

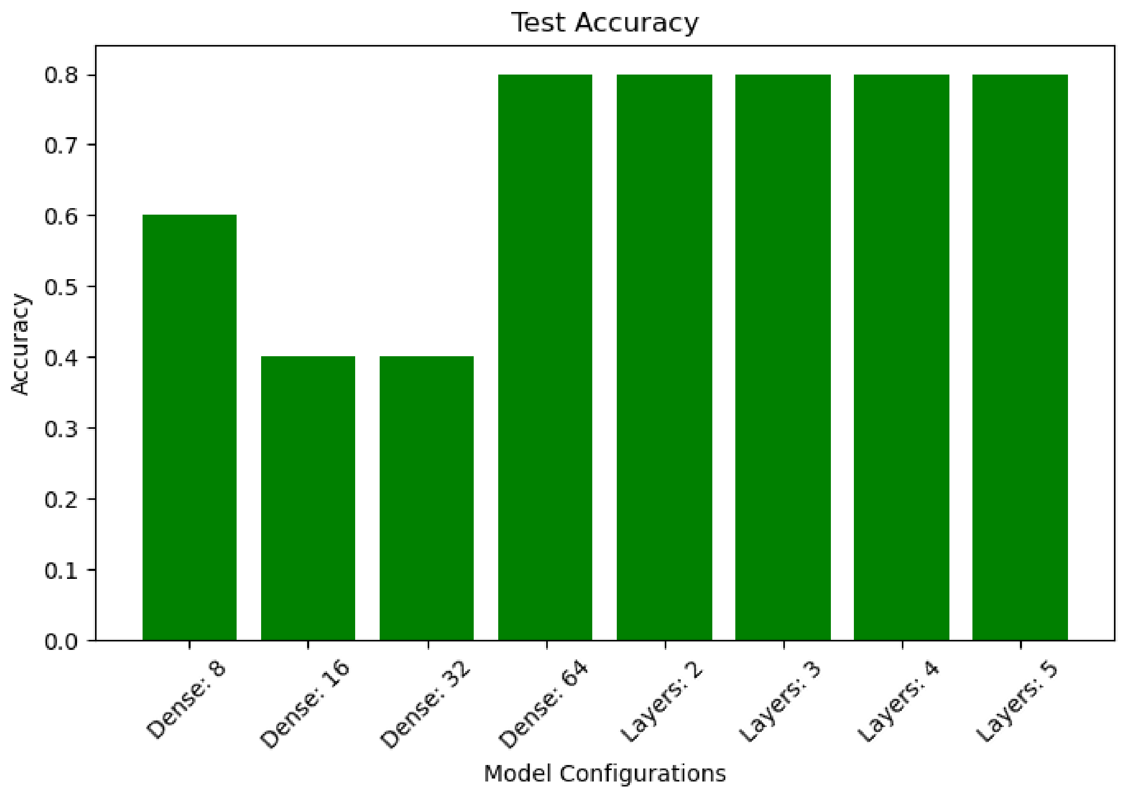
In [63]:

```
plt.figure(figsize=(7,5))
plt.bar(labels, test_loss, color='blue')
plt.title('Test Loss')
plt.xlabel('Model Configurations')
plt.ylabel('Loss')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [64]:

```
plt.figure(figsize=(7, 5))
plt.bar(labels, test_accuracy, color='green')
plt.title('Test Accuracy')
plt.xlabel('Model Configurations')
plt.ylabel('Accuracy')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [65]:

```
best_loss_index = test_loss.index(min(test_loss))
print(f"Best Model (Test Accuracy): {labels[best_accuracy_index]}")
```

```
Best Model (Test Accuracy): Dense: 64
```

In [66]:

```
best_accuracy_index = test_accuracy.index(max(test_accuracy))
print(f"Best Model (Test Loss): {labels[best_loss_index]}")
```

```
Best Model (Test Loss): Layers: 3
```