LAB5 : Text corpus creation and binary classification using DNN¶

NAME : ANNAPOORNIMA S

ROLL NO: 225229101

Dataset Creation:

In [1]:
```
import nltk
import pandas as pd
```

```
C:\ProgramData\Anaconda3\anacoda\lib\site-packages\scipy\__init__.py:146: Use
rWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version o
f SciPy (detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from nltk.stem import WordNetLemmatizer
```

In [7]:
```
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\JESSY\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\JESSY\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\JESSY\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[7]:
```
True
```

In [8]:
```
df=pd.read_csv("Motive - Sheet1.csv",encoding='cp1252')
```

In [9]:
```
df.shape
```

Out[9]:
```
(20, 2)
```

In [10]:
```
df.head()
```

Out[10]:

	Statement	Label
0	You can achieve anything if you believe in you...	Motivation
1	Every failure is a stepping stone to success.	Motivation
2	Hard work and dedication lead to great results.	Motivation
3	Success comes to those who never give up.	Motivation
4	Your potential is limitless.	Motivation

In [11]:
```
df.groupby('Label').count()
```

Out[11]:

Label	Statement
Demotivation	10
Motivation	10

Pre-processing:

In [12]:
```
X=df.Statement
y=df.Label
```

In [13]:
```
lemmatizer=WordNetLemmatizer()
```

In [14]:
```
def clean_review(review):

    tokens = review.lower().split()
    filtered_tokens = [lemmatizer.lemmatize(w) for w in tokens if w not in stop
    return " ".join(filtered_tokens)
```

In [15]:
```
temp=X.tolist()
fax=[]
for i in temp:
    fax.append(clean_review(i))
n_X=pd.Series(fax)
```

In [16]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

tfidf = TfidfVectorizer()
vectors = tfidf.fit_transform(n_X)
features_names = tfidf.get_feature_names_out()
text_vect = pd.DataFrame(vectors.todense(), columns=features_names)
text_vect
```

Out[16]:

	abilities	achieve	afraid	anything	believe	bother	capable	challenge	change
0	0.000000	0.550965	0.000000	0.550965	0.626798	0.000000	0.00000	0.00000	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
5	0.000000	0.000000	0.421482	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
6	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.57735	0.57735	0.000000
7	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
8	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
9	0.628951	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000
10	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.00000	0.000000

Dataset Preparation:

In [17]:
```
import tensorflow as tf
temp = tf.Variable(text_vect)
```

In [19]:
```
X_train,X_test,y_train,y_test=train_test_split(text_vect,y,train_size=0.75,test
```

In [20]:
```
print(X_train.shape)
```

```
(15, 62)
```

In [21]:
```
print(y_train.shape)
```

```
(15,)
```

In [22]:
```
print(X_test.shape)
```

```
(5, 62)
```

In [23]:
```
print(y_test.shape)
```

```
(5,)
```

Model Build

In [25]:
```
import tensorflow as tf
from tensorflow.keras import Sequential
from keras.layers import Dense,Activation
```

Model 1

In [26]:
```
model1 = Sequential()
model1.add(Dense(8, activation='relu',input_dim=X_train.shape[1]))
model1.add(Dense(2, activation='sigmoid')) #output layer
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8)                 504

 dense_1 (Dense)             (None, 2)                 18

=================================================================
Total params: 522 (2.04 KB)
Trainable params: 522 (2.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [27]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.fit_transform(y_test)
```

In [28]:
```
model1.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=
history=model1.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,ba
```

```
Epoch 1/100
1/1 - 4s - loss: 0.7268 - accuracy: 0.4167 - val_loss: 0.7126 - val_accurac
y: 0.3333 - 4s/epoch - 4s/step
Epoch 2/100
1/1 - 0s - loss: 0.7247 - accuracy: 0.4167 - val_loss: 0.7126 - val_accurac
y: 0.3333 - 111ms/epoch - 111ms/step
Epoch 3/100
1/1 - 0s - loss: 0.7226 - accuracy: 0.4167 - val_loss: 0.7126 - val_accurac
y: 0.3333 - 71ms/epoch - 71ms/step
Epoch 4/100
1/1 - 0s - loss: 0.7206 - accuracy: 0.4167 - val_loss: 0.7127 - val_accurac
y: 0.3333 - 83ms/epoch - 83ms/step
Epoch 5/100
1/1 - 0s - loss: 0.7185 - accuracy: 0.4167 - val_loss: 0.7127 - val_accurac
y: 0.3333 - 84ms/epoch - 84ms/step
Epoch 6/100
1/1 - 0s - loss: 0.7165 - accuracy: 0.4167 - val_loss: 0.7127 - val_accurac
y: 0.3333 - 94ms/epoch - 94ms/step
Epoch 7/100
```

In [29]:
```
model1.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 47ms/step - loss: 0.7391 - accurac
y: 0.2000
```

Out[29]:
```
[0.7391146421432495, 0.20000000298023224]
```

Model 2

In [30]:
```
model2 = Sequential()
model2.add(Dense(16, activation='relu',input_dim=X_train.shape[1]))
model2.add(Dense(8, activation='relu'))
model2.add(Dense(2, activation='sigmoid')) #output layer
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 16)                1008

 dense_3 (Dense)             (None, 8)                 136

 dense_4 (Dense)             (None, 2)                 18

=================================================================
Total params: 1162 (4.54 KB)
Trainable params: 1162 (4.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [31]:
```
model2.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=
history=model2.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,ba
```

```
Epoch 1/100
1/1 - 2s - loss: 0.7001 - accuracy: 0.4167 - val_loss: 0.7297 - val_accurac
y: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 2/100
1/1 - 0s - loss: 0.6964 - accuracy: 0.4167 - val_loss: 0.7310 - val_accurac
y: 0.0000e+00 - 93ms/epoch - 93ms/step
Epoch 3/100
1/1 - 0s - loss: 0.6927 - accuracy: 0.4167 - val_loss: 0.7324 - val_accurac
y: 0.0000e+00 - 82ms/epoch - 82ms/step
Epoch 4/100
1/1 - 0s - loss: 0.6892 - accuracy: 0.4167 - val_loss: 0.7337 - val_accurac
y: 0.0000e+00 - 83ms/epoch - 83ms/step
Epoch 5/100
1/1 - 0s - loss: 0.6856 - accuracy: 0.4167 - val_loss: 0.7351 - val_accurac
y: 0.0000e+00 - 67ms/epoch - 67ms/step
Epoch 6/100
1/1 - 0s - loss: 0.6822 - accuracy: 0.5000 - val_loss: 0.7365 - val_accurac
y: 0.0000e+00 - 77ms/epoch - 77ms/step
Epoch 7/100
```

In [32]:
```
model2.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 57ms/step - loss: 0.7475 - accurac
y: 0.2000
```

Out[32]:
```
[0.7475031614303589, 0.20000000298023224]
```

Model 3

In [33]:
```
model3 = Sequential()
model3.add(Dense(32, activation='relu',input_dim=X_train.shape[1]))
model3.add(Dense(16, activation='relu'))
model3.add(Dense(8, activation='relu'))
model3.add(Dense(2, activation='sigmoid')) #output layer
model3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 32)                2016

 dense_6 (Dense)             (None, 16)                528

 dense_7 (Dense)             (None, 8)                 136

 dense_8 (Dense)             (None, 2)                 18

=================================================================
Total params: 2698 (10.54 KB)
Trainable params: 2698 (10.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [34]:
```
model3.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=
history=model3.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,ba
```

```
Epoch 1/100
1/1 - 2s - loss: 0.7241 - accuracy: 0.3333 - val_loss: 0.7662 - val_accurac
y: 0.3333 - 2s/epoch - 2s/step
Epoch 2/100
1/1 - 0s - loss: 0.7179 - accuracy: 0.3333 - val_loss: 0.7629 - val_accurac
y: 0.3333 - 88ms/epoch - 88ms/step
Epoch 3/100
1/1 - 0s - loss: 0.7119 - accuracy: 0.4167 - val_loss: 0.7595 - val_accurac
y: 0.3333 - 68ms/epoch - 68ms/step
Epoch 4/100
1/1 - 0s - loss: 0.7065 - accuracy: 0.5000 - val_loss: 0.7562 - val_accurac
y: 0.0000e+00 - 100ms/epoch - 100ms/step
Epoch 5/100
1/1 - 0s - loss: 0.7017 - accuracy: 0.5000 - val_loss: 0.7532 - val_accurac
y: 0.0000e+00 - 91ms/epoch - 91ms/step
Epoch 6/100
1/1 - 0s - loss: 0.6971 - accuracy: 0.5000 - val_loss: 0.7505 - val_accurac
y: 0.0000e+00 - 118ms/epoch - 118ms/step
Epoch 7/100
```

In [35]:
```
model3.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 67ms/step - loss: 1.0329 - accurac
y: 0.4000
```

Out[35]:
```
[1.0329406261444092, 0.4000000059604645]
```

Model 4

In [36]:
```
model4 = Sequential()
model4.add(Dense(64, activation='relu',input_dim=X_train.shape[1]))
model4.add(Dense(32, activation='relu'))
model4.add(Dense(16, activation='relu'))
model4.add(Dense(8, activation='relu'))
model4.add(Dense(2, activation='sigmoid')) #output layer
model4.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_9 (Dense)             (None, 64)                4032

 dense_10 (Dense)            (None, 32)                2080

 dense_11 (Dense)            (None, 16)                528

 dense_12 (Dense)            (None, 8)                 136

 dense_13 (Dense)            (None, 2)                 18

=================================================================
Total params: 6794 (26.54 KB)
Trainable params: 6794 (26.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [37]:
```
model4.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=
history=model4.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,ba
```

```
Epoch 1/100
1/1 - 2s - loss: 0.7043 - accuracy: 0.5000 - val_loss: 0.7386 - val_accurac
y: 0.3333 - 2s/epoch - 2s/step
Epoch 2/100
1/1 - 0s - loss: 0.6951 - accuracy: 0.5000 - val_loss: 0.7403 - val_accurac
y: 0.3333 - 108ms/epoch - 108ms/step
Epoch 3/100
1/1 - 0s - loss: 0.6862 - accuracy: 0.5000 - val_loss: 0.7417 - val_accurac
y: 0.3333 - 105ms/epoch - 105ms/step
Epoch 4/100
1/1 - 0s - loss: 0.6786 - accuracy: 0.5000 - val_loss: 0.7433 - val_accurac
y: 0.3333 - 76ms/epoch - 76ms/step
Epoch 5/100
1/1 - 0s - loss: 0.6718 - accuracy: 0.5000 - val_loss: 0.7461 - val_accurac
y: 0.3333 - 79ms/epoch - 79ms/step
Epoch 6/100
1/1 - 0s - loss: 0.6648 - accuracy: 0.5000 - val_loss: 0.7498 - val_accurac
y: 0.3333 - 80ms/epoch - 80ms/step
Epoch 7/100
```

In [38]:
```
model4.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 58ms/step - loss: 1.1901 - accurac
y: 0.6000
```

Out[38]:
```
[1.1900800466537476, 0.6000000238418579]
```

Model 5

In [39]:
```
model5 = Sequential()
model5.add(Dense(128, activation='relu',input_dim=X_train.shape[1]))
model5.add(Dense(64, activation='relu'))
model5.add(Dense(32, activation='relu'))
model5.add(Dense(16, activation='relu'))
model5.add(Dense(8, activation='relu'))
model5.add(Dense(2, activation='sigmoid')) #output layer
model5.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_14 (Dense)            (None, 128)               8064

 dense_15 (Dense)            (None, 64)                8256

 dense_16 (Dense)            (None, 32)                2080

 dense_17 (Dense)            (None, 16)                528

 dense_18 (Dense)            (None, 8)                 136

 dense_19 (Dense)            (None, 2)                 18

=================================================================
Total params: 19082 (74.54 KB)
Trainable params: 19082 (74.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [40]:
```
model5.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=
history=model5.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,ba
```

```
Epoch 1/100
1/1 - 3s - loss: 0.6954 - accuracy: 0.3333 - val_loss: 0.6890 - val_accurac
y: 0.6667 - 3s/epoch - 3s/step
Epoch 2/100
1/1 - 0s - loss: 0.6862 - accuracy: 0.5833 - val_loss: 0.6918 - val_accurac
y: 0.6667 - 97ms/epoch - 97ms/step
Epoch 3/100
1/1 - 0s - loss: 0.6793 - accuracy: 0.6667 - val_loss: 0.6933 - val_accurac
y: 0.6667 - 99ms/epoch - 99ms/step
Epoch 4/100
1/1 - 0s - loss: 0.6729 - accuracy: 0.8333 - val_loss: 0.6942 - val_accurac
y: 0.3333 - 88ms/epoch - 88ms/step
Epoch 5/100
1/1 - 0s - loss: 0.6667 - accuracy: 0.9167 - val_loss: 0.6944 - val_accurac
y: 0.6667 - 73ms/epoch - 73ms/step
Epoch 6/100
1/1 - 0s - loss: 0.6602 - accuracy: 0.9167 - val_loss: 0.6944 - val_accurac
y: 0.6667 - 78ms/epoch - 78ms/step
Epoch 7/100
```

In [41]:
```
model5.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 78ms/step - loss: 1.2698 - accurac
y: 0.4000
```

Out[41]:
```
[1.269778847694397, 0.4000000059604645]
```

Model 6

In [42]:
```
model6 = Sequential()
model6.add(Dense(256, activation='relu',input_dim=X_train.shape[1]))
model6.add(Dense(128, activation='relu'))
model6.add(Dense(64, activation='relu'))
model6.add(Dense(32, activation='relu'))
model6.add(Dense(16, activation='relu'))
model6.add(Dense(6, activation='relu'))
model6.add(Dense(2, activation='sigmoid')) #output layer
model6.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_20 (Dense)            (None, 256)               16128

 dense_21 (Dense)            (None, 128)               32896

 dense_22 (Dense)            (None, 64)                8256

 dense_23 (Dense)            (None, 32)                2080

 dense_24 (Dense)            (None, 16)                528

 dense_25 (Dense)            (None, 6)                 102

 dense_26 (Dense)            (None, 2)                 14

=================================================================
Total params: 60004 (234.39 KB)
Trainable params: 60004 (234.39 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [43]:
```
model6.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=
history=model6.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,ba
```

```
Epoch 1/100
1/1 - 3s - loss: 0.7044 - accuracy: 0.5000 - val_loss: 0.6873 - val_accurac
y: 0.6667 - 3s/epoch - 3s/step
Epoch 2/100
1/1 - 0s - loss: 0.6810 - accuracy: 0.5000 - val_loss: 0.6902 - val_accurac
y: 0.6667 - 88ms/epoch - 88ms/step
Epoch 3/100
1/1 - 0s - loss: 0.6660 - accuracy: 0.5000 - val_loss: 0.6912 - val_accurac
y: 0.6667 - 84ms/epoch - 84ms/step
Epoch 4/100
1/1 - 0s - loss: 0.6514 - accuracy: 0.5833 - val_loss: 0.6914 - val_accurac
y: 0.6667 - 99ms/epoch - 99ms/step
Epoch 5/100
1/1 - 0s - loss: 0.6370 - accuracy: 0.7500 - val_loss: 0.6934 - val_accurac
y: 0.6667 - 102ms/epoch - 102ms/step
Epoch 6/100
1/1 - 0s - loss: 0.6204 - accuracy: 0.8333 - val_loss: 0.6954 - val_accurac
y: 0.6667 - 99ms/epoch - 99ms/step
Epoch 7/100
```

In [44]:
```
model6.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 47ms/step - loss: 3.5033 - accurac
y: 0.4000
```

Out[44]:
```
[3.5032782554626465, 0.4000000059604645]
```

Model 7

In [45]:
```
model7 = Sequential()
model7.add(Dense(512, activation='relu',input_dim=X_train.shape[1]))
model7.add(Dense(256, activation='relu'))
model7.add(Dense(128, activation='relu'))
model7.add(Dense(64, activation='relu'))
model7.add(Dense(32, activation='relu'))
model7.add(Dense(16, activation='relu'))
model7.add(Dense(8, activation='relu'))
model7.add(Dense(2, activation='sigmoid'))
model7.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_27 (Dense)            (None, 512)               32256

 dense_28 (Dense)            (None, 256)               131328

 dense_29 (Dense)            (None, 128)               32896

 dense_30 (Dense)            (None, 64)                8256

 dense_31 (Dense)            (None, 32)                2080

 dense_32 (Dense)            (None, 16)                528

 dense_33 (Dense)            (None, 8)                 136

 dense_34 (Dense)            (None, 2)                 18

=================================================================
Total params: 207498 (810.54 KB)
Trainable params: 207498 (810.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [46]:
```
model7.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=
history=model7.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,ba
```

```
Epoch 1/100
1/1 - 4s - loss: 0.6928 - accuracy: 0.5000 - val_loss: 0.6894 - val_accurac
y: 0.6667 - 4s/epoch - 4s/step
Epoch 2/100
1/1 - 0s - loss: 0.6845 - accuracy: 0.5000 - val_loss: 0.6873 - val_accurac
y: 0.6667 - 131ms/epoch - 131ms/step
Epoch 3/100
1/1 - 0s - loss: 0.6752 - accuracy: 0.6667 - val_loss: 0.6843 - val_accurac
y: 0.6667 - 84ms/epoch - 84ms/step
Epoch 4/100
1/1 - 0s - loss: 0.6631 - accuracy: 0.7500 - val_loss: 0.6810 - val_accurac
y: 0.6667 - 67ms/epoch - 67ms/step
Epoch 5/100
1/1 - 0s - loss: 0.6490 - accuracy: 0.6667 - val_loss: 0.6811 - val_accurac
y: 0.6667 - 82ms/epoch - 82ms/step
Epoch 6/100
1/1 - 0s - loss: 0.6330 - accuracy: 1.0000 - val_loss: 0.6797 - val_accurac
y: 0.6667 - 84ms/epoch - 84ms/step
Epoch 7/100
```

In [47]:
```
model7.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 60ms/step - loss: 2.8365 - accurac
y: 0.6000
```

Out[47]:
```
[2.836496591567993, 0.6000000238418579]
```

Model 8

In [48]:
```
model8 = Sequential()
model8.add(Dense(1028, activation='relu',input_dim=X_train.shape[1]))
model8.add(Dense(512, activation='relu'))
model8.add(Dense(356, activation='relu'))
model8.add(Dense(128, activation='relu'))
model8.add(Dense(64, activation='relu'))
model8.add(Dense(32, activation='relu'))
model8.add(Dense(16, activation='relu'))
model8.add(Dense(8, activation='relu'))
model8.add(Dense(2, activation='sigmoid')) #output layer
model8.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_35 (Dense)            (None, 1028)              64764

 dense_36 (Dense)            (None, 512)               526848

 dense_37 (Dense)            (None, 356)               182628

 dense_38 (Dense)            (None, 128)               45696

 dense_39 (Dense)            (None, 64)                8256

 dense_40 (Dense)            (None, 32)                2080

 dense_41 (Dense)            (None, 16)                528

 dense_42 (Dense)            (None, 8)                 136

 dense_43 (Dense)            (None, 2)                 18

=================================================================
Total params: 830954 (3.17 MB)
Trainable params: 830954 (3.17 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [49]:
```
model8.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=
history=model8.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,ba
```

```
Epoch 1/100
1/1 - 5s - loss: 0.6892 - accuracy: 0.5000 - val_loss: 0.7157 - val_accurac
y: 0.3333 - 5s/epoch - 5s/step
Epoch 2/100
1/1 - 0s - loss: 0.6495 - accuracy: 0.5000 - val_loss: 0.7636 - val_accurac
y: 0.3333 - 124ms/epoch - 124ms/step
Epoch 3/100
1/1 - 0s - loss: 0.5988 - accuracy: 0.5000 - val_loss: 0.8011 - val_accurac
y: 0.3333 - 117ms/epoch - 117ms/step
Epoch 4/100
1/1 - 0s - loss: 0.5312 - accuracy: 0.9167 - val_loss: 0.8814 - val_accurac
y: 0.3333 - 116ms/epoch - 116ms/step
Epoch 5/100
1/1 - 0s - loss: 0.4658 - accuracy: 1.0000 - val_loss: 1.0724 - val_accurac
y: 0.3333 - 118ms/epoch - 118ms/step
Epoch 6/100
1/1 - 0s - loss: 0.4079 - accuracy: 0.8333 - val_loss: 1.3474 - val_accurac
y: 0.3333 - 125ms/epoch - 125ms/step
Epoch 7/100
```

In [50]:
```
model8.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 85ms/step - loss: 7.3142 - accurac
y: 0.6000
```

Out[50]:
```
[7.314168453216553, 0.6000000238418579]
```

Outer Layer

Model 1

In [51]:
```
model = Sequential()
model.add(Dense(32, activation='relu',input_dim=X_train.shape[1]))
model.add(Dense(2, activation='sigmoid')) #output layer
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_44 (Dense)            (None, 32)                2016

 dense_45 (Dense)            (None, 2)                 66

=================================================================
Total params: 2082 (8.13 KB)
Trainable params: 2082 (8.13 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [52]:
```
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=[
history=model.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,bat
```

```
Epoch 1/100
1/1 - 2s - loss: 0.7168 - accuracy: 0.2500 - val_loss: 0.6968 - val_accurac
y: 0.3333 - 2s/epoch - 2s/step
Epoch 2/100
1/1 - 0s - loss: 0.7118 - accuracy: 0.2500 - val_loss: 0.6983 - val_accurac
y: 0.3333 - 84ms/epoch - 84ms/step
Epoch 3/100
1/1 - 0s - loss: 0.7068 - accuracy: 0.2500 - val_loss: 0.6997 - val_accurac
y: 0.3333 - 100ms/epoch - 100ms/step
Epoch 4/100
1/1 - 0s - loss: 0.7018 - accuracy: 0.2500 - val_loss: 0.7012 - val_accurac
y: 0.3333 - 108ms/epoch - 108ms/step
Epoch 5/100
1/1 - 0s - loss: 0.6969 - accuracy: 0.3333 - val_loss: 0.7026 - val_accurac
y: 0.3333 - 117ms/epoch - 117ms/step
Epoch 6/100
1/1 - 0s - loss: 0.6920 - accuracy: 0.3333 - val_loss: 0.7040 - val_accurac
y: 0.3333 - 123ms/epoch - 123ms/step
Epoch 7/100
```

In [53]:
```
model.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 62ms/step - loss: 0.8233 - accurac
y: 0.2000
```

Out[53]:
```
[0.8233494758605957, 0.20000000298023224]
```

Model 2

In [54]:
```
model0 = Sequential()
model0.add(Dense(32, activation='relu',input_dim=X_train.shape[1]))
model0.add(Dense(3, activation='sigmoid')) #output layer
model0.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_46 (Dense)            (None, 32)                2016

 dense_47 (Dense)            (None, 3)                 99

=================================================================
Total params: 2115 (8.26 KB)
Trainable params: 2115 (8.26 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [55]:
```
model0.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=
history=model0.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,ba
```

```
Epoch 1/100
1/1 - 2s - loss: 1.1067 - accuracy: 0.1667 - val_loss: 1.1619 - val_accurac
y: 0.3333 - 2s/epoch - 2s/step
Epoch 2/100
1/1 - 0s - loss: 1.0978 - accuracy: 0.1667 - val_loss: 1.1582 - val_accurac
y: 0.3333 - 112ms/epoch - 112ms/step
Epoch 3/100
1/1 - 0s - loss: 1.0890 - accuracy: 0.2500 - val_loss: 1.1545 - val_accurac
y: 0.3333 - 84ms/epoch - 84ms/step
Epoch 4/100
1/1 - 0s - loss: 1.0803 - accuracy: 0.3333 - val_loss: 1.1509 - val_accurac
y: 0.3333 - 101ms/epoch - 101ms/step
Epoch 5/100
1/1 - 0s - loss: 1.0718 - accuracy: 0.3333 - val_loss: 1.1474 - val_accurac
y: 0.3333 - 83ms/epoch - 83ms/step
Epoch 6/100
1/1 - 0s - loss: 1.0633 - accuracy: 0.3333 - val_loss: 1.1439 - val_accurac
y: 0.3333 - 101ms/epoch - 101ms/step
Epoch 7/100
```

In [56]:
```
model0.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 53ms/step - loss: 1.1729 - accurac
y: 0.2000
```

Out[56]:
```
[1.1728932857513428, 0.20000000298023224]
```

Model 3

In [57]:
```
model01 = Sequential()
model01.add(Dense(32, activation='relu',input_dim=X_train.shape[1]))
model01.add(Dense(4, activation='sigmoid')) #output layer
model01.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_48 (Dense)            (None, 32)                2016

 dense_49 (Dense)            (None, 4)                 132

=================================================================
Total params: 2148 (8.39 KB)
Trainable params: 2148 (8.39 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [58]:
```
model01.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics
history=model01.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,b
```

```
Epoch 1/100
1/1 - 2s - loss: 1.3713 - accuracy: 0.5000 - val_loss: 1.3709 - val_accurac
y: 0.3333 - 2s/epoch - 2s/step
Epoch 2/100
1/1 - 0s - loss: 1.3626 - accuracy: 0.5000 - val_loss: 1.3667 - val_accurac
y: 0.3333 - 63ms/epoch - 63ms/step
Epoch 3/100
1/1 - 0s - loss: 1.3539 - accuracy: 0.5000 - val_loss: 1.3624 - val_accurac
y: 0.3333 - 63ms/epoch - 63ms/step
Epoch 4/100
1/1 - 0s - loss: 1.3452 - accuracy: 0.5000 - val_loss: 1.3581 - val_accurac
y: 0.3333 - 78ms/epoch - 78ms/step
Epoch 5/100
1/1 - 0s - loss: 1.3365 - accuracy: 0.5000 - val_loss: 1.3538 - val_accurac
y: 0.3333 - 109ms/epoch - 109ms/step
Epoch 6/100
1/1 - 0s - loss: 1.3279 - accuracy: 0.5000 - val_loss: 1.3496 - val_accurac
y: 0.3333 - 78ms/epoch - 78ms/step
Epoch 7/100
```

In [59]:
```
model01.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 70ms/step - loss: 1.0599 - accurac
y: 0.6000
```

Out[59]:
```
[1.0598688125610352, 0.6000000238418579]
```

Model 4

In [60]:
```
model02 = Sequential()
model02.add(Dense(32, activation='relu',input_dim=X_train.shape[1]))
model02.add(Dense(5, activation='sigmoid')) #output layer
model02.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_50 (Dense)            (None, 32)                2016

 dense_51 (Dense)            (None, 5)                 165

=================================================================
Total params: 2181 (8.52 KB)
Trainable params: 2181 (8.52 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [61]:
```
model02.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics
history=model02.fit(X_train,y_train,epochs=100,verbose=2,validation_split=0.2,b
```

```
Epoch 1/100
1/1 - 2s - loss: 1.6387 - accuracy: 0.1667 - val_loss: 1.6957 - val_accurac
y: 0.0000e+00 - 2s/epoch - 2s/step
Epoch 2/100
1/1 - 0s - loss: 1.6291 - accuracy: 0.2500 - val_loss: 1.6907 - val_accurac
y: 0.0000e+00 - 121ms/epoch - 121ms/step
Epoch 3/100
1/1 - 0s - loss: 1.6196 - accuracy: 0.2500 - val_loss: 1.6856 - val_accurac
y: 0.0000e+00 - 103ms/epoch - 103ms/step
Epoch 4/100
1/1 - 0s - loss: 1.6101 - accuracy: 0.2500 - val_loss: 1.6806 - val_accurac
y: 0.0000e+00 - 125ms/epoch - 125ms/step
Epoch 5/100
1/1 - 0s - loss: 1.6006 - accuracy: 0.2500 - val_loss: 1.6757 - val_accurac
y: 0.0000e+00 - 110ms/epoch - 110ms/step
Epoch 6/100
1/1 - 0s - loss: 1.5912 - accuracy: 0.2500 - val_loss: 1.6708 - val_accurac
y: 0.0000e+00 - 177ms/epoch - 177ms/step
Epoch 7/100
```

In [62]:
```
model02.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 0s 109ms/step - loss: 1.1554 - accurac
y: 0.4000
```

Out[62]:
```
[1.1553707122802734, 0.4000000059604645]
```

In [ ]:
```

```