

## 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: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of 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]:

	Statement
	Label
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	0.000000	0.000000	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5	0.000000	0.000000	0.421482	0.000000	0.000000	0.000000	0.000000	0.000000	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	0.000000	0.000000	0.000000
8	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
9	0.628951	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
10	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	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 #
<hr/>		
dense (Dense)	(None, 8)	504
dense_1 (Dense)	(None, 2)	18
<hr/>		
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_accuracy: 0.3333 - 4s/epoch - 4s/step
Epoch 2/100
1/1 - 0s - loss: 0.7247 - accuracy: 0.4167 - val_loss: 0.7126 - val_accuracy: 0.3333 - 111ms/epoch - 111ms/step
Epoch 3/100
1/1 - 0s - loss: 0.7226 - accuracy: 0.4167 - val_loss: 0.7126 - val_accuracy: 0.3333 - 71ms/epoch - 71ms/step
Epoch 4/100
1/1 - 0s - loss: 0.7206 - accuracy: 0.4167 - val_loss: 0.7127 - val_accuracy: 0.3333 - 83ms/epoch - 83ms/step
Epoch 5/100
1/1 - 0s - loss: 0.7185 - accuracy: 0.4167 - val_loss: 0.7127 - val_accuracy: 0.3333 - 84ms/epoch - 84ms/step
Epoch 6/100
1/1 - 0s - loss: 0.7165 - accuracy: 0.4167 - val_loss: 0.7127 - val_accuracy: 0.3333 - 94ms/epoch - 94ms/step
Epoch 7/100
1/1 - 0s - loss: 0.7145 - accuracy: 0.4167 - val_loss: 0.7127 - val_accuracy: 0.3333 - 94ms/epoch - 94ms/step
```

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

```
1/1 [=====] - 0s 47ms/step - loss: 0.7391 - accuracy: 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 #
<hr/>		
dense_2 (Dense)	(None, 16)	1008
dense_3 (Dense)	(None, 8)	136
dense_4 (Dense)	(None, 2)	18
<hr/>		
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_accuracy:  
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_accuracy:  
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_accuracy:  
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_accuracy:  
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_accuracy:  
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_accuracy:  
y: 0.0000e+00 - 77ms/epoch - 77ms/step  
Epoch 7/100  
1/1 [=====] - 0s 57ms/step - loss: 0.7475 - accuracy: 0.2000
```

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

```
1/1 [=====] - 0s 57ms/step - loss: 0.7475 - accuracy: 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 #
<hr/>		
dense_5 (Dense)	(None, 32)	2016
dense_6 (Dense)	(None, 16)	528
dense_7 (Dense)	(None, 8)	136
dense_8 (Dense)	(None, 2)	18
<hr/>		
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,batch_size=32))
```

```
Epoch 1/100
1/1 - 2s - loss: 0.7241 - accuracy: 0.3333 - val_loss: 0.7662 - val_accuracy: 0.3333 - 2s/epoch - 2s/step
Epoch 2/100
1/1 - 0s - loss: 0.7179 - accuracy: 0.3333 - val_loss: 0.7629 - val_accuracy: 0.3333 - 88ms/epoch - 88ms/step
Epoch 3/100
1/1 - 0s - loss: 0.7119 - accuracy: 0.4167 - val_loss: 0.7595 - val_accuracy: 0.3333 - 68ms/epoch - 68ms/step
Epoch 4/100
1/1 - 0s - loss: 0.7065 - accuracy: 0.5000 - val_loss: 0.7562 - val_accuracy: 0.0000e+00 - 100ms/epoch - 100ms/step
Epoch 5/100
1/1 - 0s - loss: 0.7017 - accuracy: 0.5000 - val_loss: 0.7532 - val_accuracy: 0.0000e+00 - 91ms/epoch - 91ms/step
Epoch 6/100
1/1 - 0s - loss: 0.6971 - accuracy: 0.5000 - val_loss: 0.7505 - val_accuracy: 0.0000e+00 - 118ms/epoch - 118ms/step
Epoch 7/100
1/1 - 0s - loss: 0.6934 - accuracy: 0.5000 - val_loss: 0.7482 - val_accuracy: 0.0000e+00 - 136ms/epoch - 136ms/step
```

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

```
1/1 [=====] - 0s 67ms/step - loss: 1.0329 - accuracy: 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 #
<hr/>		
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
<hr/>		
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_accuracy: 0.3333 - 2s/epoch - 2s/step
Epoch 2/100
1/1 - 0s - loss: 0.6951 - accuracy: 0.5000 - val_loss: 0.7403 - val_accuracy: 0.3333 - 108ms/epoch - 108ms/step
Epoch 3/100
1/1 - 0s - loss: 0.6862 - accuracy: 0.5000 - val_loss: 0.7417 - val_accuracy: 0.3333 - 105ms/epoch - 105ms/step
Epoch 4/100
1/1 - 0s - loss: 0.6786 - accuracy: 0.5000 - val_loss: 0.7433 - val_accuracy: 0.3333 - 76ms/epoch - 76ms/step
Epoch 5/100
1/1 - 0s - loss: 0.6718 - accuracy: 0.5000 - val_loss: 0.7461 - val_accuracy: 0.3333 - 79ms/epoch - 79ms/step
Epoch 6/100
1/1 - 0s - loss: 0.6648 - accuracy: 0.5000 - val_loss: 0.7498 - val_accuracy: 0.3333 - 80ms/epoch - 80ms/step
Epoch 7/100
1/1 - 0s - loss: 0.6579 - accuracy: 0.5000 - val_loss: 0.7542 - val_accuracy: 0.3333 - 80ms/epoch - 80ms/step
```

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

```
1/1 [=====] - 0s 58ms/step - loss: 1.1901 - accuracy: 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 #
<hr/>		
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
<hr/>		
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\_accuracy: 0.6667 - 3s/epoch - 3s/step  
Epoch 2/100  
1/1 - 0s - loss: 0.6862 - accuracy: 0.5833 - val\_loss: 0.6918 - val\_accuracy: 0.6667 - 97ms/epoch - 97ms/step  
Epoch 3/100  
1/1 - 0s - loss: 0.6793 - accuracy: 0.6667 - val\_loss: 0.6933 - val\_accuracy: 0.6667 - 99ms/epoch - 99ms/step  
Epoch 4/100  
1/1 - 0s - loss: 0.6729 - accuracy: 0.8333 - val\_loss: 0.6942 - val\_accuracy: 0.3333 - 88ms/epoch - 88ms/step  
Epoch 5/100  
1/1 - 0s - loss: 0.6667 - accuracy: 0.9167 - val\_loss: 0.6944 - val\_accuracy: 0.6667 - 73ms/epoch - 73ms/step  
Epoch 6/100  
1/1 - 0s - loss: 0.6602 - accuracy: 0.9167 - val\_loss: 0.6944 - val\_accuracy: 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 - accuracy: 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 #
<hr/>		
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
<hr/>		
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,batch_size=32)
```

```
Epoch 1/100
1/1 - 3s - loss: 0.7044 - accuracy: 0.5000 - val_loss: 0.6873 - val_accuracy: 0.6667 - 3s/epoch - 3s/step
Epoch 2/100
1/1 - 0s - loss: 0.6810 - accuracy: 0.5000 - val_loss: 0.6902 - val_accuracy: 0.6667 - 88ms/epoch - 88ms/step
Epoch 3/100
1/1 - 0s - loss: 0.6660 - accuracy: 0.5000 - val_loss: 0.6912 - val_accuracy: 0.6667 - 84ms/epoch - 84ms/step
Epoch 4/100
1/1 - 0s - loss: 0.6514 - accuracy: 0.5833 - val_loss: 0.6914 - val_accuracy: 0.6667 - 99ms/epoch - 99ms/step
Epoch 5/100
1/1 - 0s - loss: 0.6370 - accuracy: 0.7500 - val_loss: 0.6934 - val_accuracy: 0.6667 - 102ms/epoch - 102ms/step
Epoch 6/100
1/1 - 0s - loss: 0.6204 - accuracy: 0.8333 - val_loss: 0.6954 - val_accuracy: 0.6667 - 99ms/epoch - 99ms/step
Epoch 7/100
1/1 - 0s - loss: 0.6024 - accuracy: 0.8222 - val_loss: 0.6974 - val_accuracy: 0.6667 - 102ms/epoch - 102ms/step
```

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

```
1/1 [=====] - 0s 47ms/step - loss: 3.5033 - accuracy: 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 #
<hr/>		
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
<hr/>		
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_accuracy: 0.6667 - 4s/epoch - 4s/step  
Epoch 2/100  
1/1 - 0s - loss: 0.6845 - accuracy: 0.5000 - val_loss: 0.6873 - val_accuracy: 0.6667 - 131ms/epoch - 131ms/step  
Epoch 3/100  
1/1 - 0s - loss: 0.6752 - accuracy: 0.6667 - val_loss: 0.6843 - val_accuracy: 0.6667 - 84ms/epoch - 84ms/step  
Epoch 4/100  
1/1 - 0s - loss: 0.6631 - accuracy: 0.7500 - val_loss: 0.6810 - val_accuracy: 0.6667 - 67ms/epoch - 67ms/step  
Epoch 5/100  
1/1 - 0s - loss: 0.6490 - accuracy: 0.6667 - val_loss: 0.6811 - val_accuracy: 0.6667 - 82ms/epoch - 82ms/step  
Epoch 6/100  
1/1 - 0s - loss: 0.6330 - accuracy: 1.0000 - val_loss: 0.6797 - val_accuracy: 0.6667 - 84ms/epoch - 84ms/step  
Epoch 7/100  
1/1 [=====] - 0s 60ms/step - loss: 2.8365 - accuracy: 0.6000
```

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

```
1/1 [=====] - 0s 60ms/step - loss: 2.8365 - accuracy: 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 #
<hr/>		
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
<hr/>		
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,batch_size=32)  
Epoch 1/100  
1/1 - 5s - loss: 0.6892 - accuracy: 0.5000 - val_loss: 0.7157 - val_accuracy: 0.3333 - 5s/epoch - 5s/step  
Epoch 2/100  
1/1 - 0s - loss: 0.6495 - accuracy: 0.5000 - val_loss: 0.7636 - val_accuracy: 0.3333 - 124ms/epoch - 124ms/step  
Epoch 3/100  
1/1 - 0s - loss: 0.5988 - accuracy: 0.5000 - val_loss: 0.8011 - val_accuracy: 0.3333 - 117ms/epoch - 117ms/step  
Epoch 4/100  
1/1 - 0s - loss: 0.5312 - accuracy: 0.9167 - val_loss: 0.8814 - val_accuracy: 0.3333 - 116ms/epoch - 116ms/step  
Epoch 5/100  
1/1 - 0s - loss: 0.4658 - accuracy: 1.0000 - val_loss: 1.0724 - val_accuracy: 0.3333 - 118ms/epoch - 118ms/step  
Epoch 6/100  
1/1 - 0s - loss: 0.4079 - accuracy: 0.8333 - val_loss: 1.3474 - val_accuracy: 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 - accuracy: 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_accuracy:  
y: 0.3333 - 2s/epoch - 2s/step  
Epoch 2/100  
1/1 - 0s - loss: 0.7118 - accuracy: 0.2500 - val_loss: 0.6983 - val_accuracy:  
y: 0.3333 - 84ms/epoch - 84ms/step  
Epoch 3/100  
1/1 - 0s - loss: 0.7068 - accuracy: 0.2500 - val_loss: 0.6997 - val_accuracy:  
y: 0.3333 - 100ms/epoch - 100ms/step  
Epoch 4/100  
1/1 - 0s - loss: 0.7018 - accuracy: 0.2500 - val_loss: 0.7012 - val_accuracy:  
y: 0.3333 - 108ms/epoch - 108ms/step  
Epoch 5/100  
1/1 - 0s - loss: 0.6969 - accuracy: 0.3333 - val_loss: 0.7026 - val_accuracy:  
y: 0.3333 - 117ms/epoch - 117ms/step  
Epoch 6/100  
1/1 - 0s - loss: 0.6920 - accuracy: 0.3333 - val_loss: 0.7040 - val_accuracy:  
y: 0.3333 - 123ms/epoch - 123ms/step  
Epoch 7/100  
1/1 - 0s - loss: 0.6871 - accuracy: 0.3333 - val_loss: 0.7054 - val_accuracy:  
y: 0.3333 - 127ms/epoch - 127ms/step  
Epoch 8/100  
1/1 - 0s - loss: 0.6822 - accuracy: 0.3333 - val_loss: 0.7068 - val_accuracy:  
y: 0.3333 - 131ms/epoch - 131ms/step  
Epoch 9/100  
1/1 - 0s - loss: 0.6773 - accuracy: 0.3333 - val_loss: 0.7082 - val_accuracy:  
y: 0.3333 - 135ms/epoch - 135ms/step  
Epoch 10/100  
1/1 - 0s - loss: 0.6724 - accuracy: 0.3333 - val_loss: 0.7096 - val_accuracy:  
y: 0.3333 - 139ms/epoch - 139ms/step  
Epoch 11/100  
1/1 - 0s - loss: 0.6675 - accuracy: 0.3333 - val_loss: 0.7110 - val_accuracy:  
y: 0.3333 - 143ms/epoch - 143ms/step  
Epoch 12/100  
1/1 - 0s - loss: 0.6626 - accuracy: 0.3333 - val_loss: 0.7124 - val_accuracy:  
y: 0.3333 - 147ms/epoch - 147ms/step  
Epoch 13/100  
1/1 - 0s - loss: 0.6577 - accuracy: 0.3333 - val_loss: 0.7138 - val_accuracy:  
y: 0.3333 - 151ms/epoch - 151ms/step  
Epoch 14/100  
1/1 - 0s - loss: 0.6528 - accuracy: 0.3333 - val_loss: 0.7152 - val_accuracy:  
y: 0.3333 - 155ms/epoch - 155ms/step  
Epoch 15/100  
1/1 - 0s - loss: 0.6479 - accuracy: 0.3333 - val_loss: 0.7166 - val_accuracy:  
y: 0.3333 - 159ms/epoch - 159ms/step  
Epoch 16/100  
1/1 - 0s - loss: 0.6430 - accuracy: 0.3333 - val_loss: 0.7180 - val_accuracy:  
y: 0.3333 - 163ms/epoch - 163ms/step  
Epoch 17/100  
1/1 - 0s - loss: 0.6381 - accuracy: 0.3333 - val_loss: 0.7194 - val_accuracy:  
y: 0.3333 - 167ms/epoch - 167ms/step  
Epoch 18/100  
1/1 - 0s - loss: 0.6332 - accuracy: 0.3333 - val_loss: 0.7208 - val_accuracy:  
y: 0.3333 - 171ms/epoch - 171ms/step  
Epoch 19/100  
1/1 - 0s - loss: 0.6283 - accuracy: 0.3333 - val_loss: 0.7222 - val_accuracy:  
y: 0.3333 - 175ms/epoch - 175ms/step  
Epoch 20/100  
1/1 - 0s - loss: 0.6234 - accuracy: 0.3333 - val_loss: 0.7236 - val_accuracy:  
y: 0.3333 - 179ms/epoch - 179ms/step  
Epoch 21/100  
1/1 - 0s - loss: 0.6185 - accuracy: 0.3333 - val_loss: 0.7250 - val_accuracy:  
y: 0.3333 - 183ms/epoch - 183ms/step  
Epoch 22/100  
1/1 - 0s - loss: 0.6136 - accuracy: 0.3333 - val_loss: 0.7264 - val_accuracy:  
y: 0.3333 - 187ms/epoch - 187ms/step  
Epoch 23/100  
1/1 - 0s - loss: 0.6087 - accuracy: 0.3333 - val_loss: 0.7278 - val_accuracy:  
y: 0.3333 - 191ms/epoch - 191ms/step  
Epoch 24/100  
1/1 - 0s - loss: 0.6038 - accuracy: 0.3333 - val_loss: 0.7292 - val_accuracy:  
y: 0.3333 - 195ms/epoch - 195ms/step  
Epoch 25/100  
1/1 - 0s - loss: 0.6089 - accuracy: 0.3333 - val_loss: 0.7306 - val_accuracy:  
y: 0.3333 - 199ms/epoch - 199ms/step  
Epoch 26/100  
1/1 - 0s - loss: 0.6040 - accuracy: 0.3333 - val_loss: 0.7320 - val_accuracy:  
y: 0.3333 - 203ms/epoch - 203ms/step  
Epoch 27/100  
1/1 - 0s - loss: 0.6091 - accuracy: 0.3333 - val_loss: 0.7334 - val_accuracy:  
y: 0.3333 - 207ms/epoch - 207ms/step  
Epoch 28/100  
1/1 - 0s - loss: 0.6042 - accuracy: 0.3333 - val_loss: 0.7348 - val_accuracy:  
y: 0.3333 - 211ms/epoch - 211ms/step  
Epoch 29/100  
1/1 - 0s - loss: 0.6093 - accuracy: 0.3333 - val_loss: 0.7362 - val_accuracy:  
y: 0.3333 - 215ms/epoch - 215ms/step  
Epoch 30/100  
1/1 - 0s - loss: 0.6044 - accuracy: 0.3333 - val_loss: 0.7376 - val_accuracy:  
y: 0.3333 - 219ms/epoch - 219ms/step  
Epoch 31/100  
1/1 - 0s - loss: 0.6095 - accuracy: 0.3333 - val_loss: 0.7390 - val_accuracy:  
y: 0.3333 - 223ms/epoch - 223ms/step  
Epoch 32/100  
1/1 - 0s - loss: 0.6046 - accuracy: 0.3333 - val_loss: 0.7404 - val_accuracy:  
y: 0.3333 - 227ms/epoch - 227ms/step  
Epoch 33/100  
1/1 - 0s - loss: 0.6096 - accuracy: 0.3333 - val_loss: 0.7418 - val_accuracy:  
y: 0.3333 - 231ms/epoch - 231ms/step  
Epoch 34/100  
1/1 - 0s - loss: 0.6047 - accuracy: 0.3333 - val_loss: 0.7432 - val_accuracy:  
y: 0.3333 - 235ms/epoch - 235ms/step  
Epoch 35/100  
1/1 - 0s - loss: 0.6097 - accuracy: 0.3333 - val_loss: 0.7446 - val_accuracy:  
y: 0.3333 - 239ms/epoch - 239ms/step  
Epoch 36/100  
1/1 - 0s - loss: 0.6048 - accuracy: 0.3333 - val_loss: 0.7460 - val_accuracy:  
y: 0.3333 - 243ms/epoch - 243ms/step  
Epoch 37/100  
1/1 - 0s - loss: 0.6098 - accuracy: 0.3333 - val_loss: 0.7474 - val_accuracy:  
y: 0.3333 - 247ms/epoch - 247ms/step  
Epoch 38/100  
1/1 - 0s - loss: 0.6049 - accuracy: 0.3333 - val_loss: 0.7488 - val_accuracy:  
y: 0.3333 - 251ms/epoch - 251ms/step  
Epoch 39/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7502 - val_accuracy:  
y: 0.3333 - 255ms/epoch - 255ms/step  
Epoch 40/100  
1/1 - 0s - loss: 0.6050 - accuracy: 0.3333 - val_loss: 0.7516 - val_accuracy:  
y: 0.3333 - 259ms/epoch - 259ms/step  
Epoch 41/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7530 - val_accuracy:  
y: 0.3333 - 263ms/epoch - 263ms/step  
Epoch 42/100  
1/1 - 0s - loss: 0.6051 - accuracy: 0.3333 - val_loss: 0.7544 - val_accuracy:  
y: 0.3333 - 267ms/epoch - 267ms/step  
Epoch 43/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7558 - val_accuracy:  
y: 0.3333 - 271ms/epoch - 271ms/step  
Epoch 44/100  
1/1 - 0s - loss: 0.6052 - accuracy: 0.3333 - val_loss: 0.7572 - val_accuracy:  
y: 0.3333 - 275ms/epoch - 275ms/step  
Epoch 45/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7586 - val_accuracy:  
y: 0.3333 - 279ms/epoch - 279ms/step  
Epoch 46/100  
1/1 - 0s - loss: 0.6053 - accuracy: 0.3333 - val_loss: 0.7599 - val_accuracy:  
y: 0.3333 - 283ms/epoch - 283ms/step  
Epoch 47/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7613 - val_accuracy:  
y: 0.3333 - 287ms/epoch - 287ms/step  
Epoch 48/100  
1/1 - 0s - loss: 0.6054 - accuracy: 0.3333 - val_loss: 0.7627 - val_accuracy:  
y: 0.3333 - 291ms/epoch - 291ms/step  
Epoch 49/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7641 - val_accuracy:  
y: 0.3333 - 295ms/epoch - 295ms/step  
Epoch 50/100  
1/1 - 0s - loss: 0.6055 - accuracy: 0.3333 - val_loss: 0.7655 - val_accuracy:  
y: 0.3333 - 299ms/epoch - 299ms/step  
Epoch 51/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7669 - val_accuracy:  
y: 0.3333 - 303ms/epoch - 303ms/step  
Epoch 52/100  
1/1 - 0s - loss: 0.6056 - accuracy: 0.3333 - val_loss: 0.7683 - val_accuracy:  
y: 0.3333 - 307ms/epoch - 307ms/step  
Epoch 53/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7697 - val_accuracy:  
y: 0.3333 - 311ms/epoch - 311ms/step  
Epoch 54/100  
1/1 - 0s - loss: 0.6057 - accuracy: 0.3333 - val_loss: 0.7711 - val_accuracy:  
y: 0.3333 - 315ms/epoch - 315ms/step  
Epoch 55/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7725 - val_accuracy:  
y: 0.3333 - 319ms/epoch - 319ms/step  
Epoch 56/100  
1/1 - 0s - loss: 0.6058 - accuracy: 0.3333 - val_loss: 0.7739 - val_accuracy:  
y: 0.3333 - 323ms/epoch - 323ms/step  
Epoch 57/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7753 - val_accuracy:  
y: 0.3333 - 327ms/epoch - 327ms/step  
Epoch 58/100  
1/1 - 0s - loss: 0.6059 - accuracy: 0.3333 - val_loss: 0.7767 - val_accuracy:  
y: 0.3333 - 331ms/epoch - 331ms/step  
Epoch 59/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7781 - val_accuracy:  
y: 0.3333 - 335ms/epoch - 335ms/step  
Epoch 60/100  
1/1 - 0s - loss: 0.6060 - accuracy: 0.3333 - val_loss: 0.7795 - val_accuracy:  
y: 0.3333 - 339ms/epoch - 339ms/step  
Epoch 61/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7809 - val_accuracy:  
y: 0.3333 - 343ms/epoch - 343ms/step  
Epoch 62/100  
1/1 - 0s - loss: 0.6061 - accuracy: 0.3333 - val_loss: 0.7823 - val_accuracy:  
y: 0.3333 - 347ms/epoch - 347ms/step  
Epoch 63/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7837 - val_accuracy:  
y: 0.3333 - 351ms/epoch - 351ms/step  
Epoch 64/100  
1/1 - 0s - loss: 0.6062 - accuracy: 0.3333 - val_loss: 0.7851 - val_accuracy:  
y: 0.3333 - 355ms/epoch - 355ms/step  
Epoch 65/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7865 - val_accuracy:  
y: 0.3333 - 359ms/epoch - 359ms/step  
Epoch 66/100  
1/1 - 0s - loss: 0.6063 - accuracy: 0.3333 - val_loss: 0.7879 - val_accuracy:  
y: 0.3333 - 363ms/epoch - 363ms/step  
Epoch 67/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7893 - val_accuracy:  
y: 0.3333 - 367ms/epoch - 367ms/step  
Epoch 68/100  
1/1 - 0s - loss: 0.6064 - accuracy: 0.3333 - val_loss: 0.7907 - val_accuracy:  
y: 0.3333 - 371ms/epoch - 371ms/step  
Epoch 69/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7921 - val_accuracy:  
y: 0.3333 - 375ms/epoch - 375ms/step  
Epoch 70/100  
1/1 - 0s - loss: 0.6065 - accuracy: 0.3333 - val_loss: 0.7935 - val_accuracy:  
y: 0.3333 - 379ms/epoch - 379ms/step  
Epoch 71/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7949 - val_accuracy:  
y: 0.3333 - 383ms/epoch - 383ms/step  
Epoch 72/100  
1/1 - 0s - loss: 0.6066 - accuracy: 0.3333 - val_loss: 0.7963 - val_accuracy:  
y: 0.3333 - 387ms/epoch - 387ms/step  
Epoch 73/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.7977 - val_accuracy:  
y: 0.3333 - 391ms/epoch - 391ms/step  
Epoch 74/100  
1/1 - 0s - loss: 0.6067 - accuracy: 0.3333 - val_loss: 0.7991 - val_accuracy:  
y: 0.3333 - 395ms/epoch - 395ms/step  
Epoch 75/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8005 - val_accuracy:  
y: 0.3333 - 399ms/epoch - 399ms/step  
Epoch 76/100  
1/1 - 0s - loss: 0.6068 - accuracy: 0.3333 - val_loss: 0.8019 - val_accuracy:  
y: 0.3333 - 403ms/epoch - 403ms/step  
Epoch 77/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8033 - val_accuracy:  
y: 0.3333 - 407ms/epoch - 407ms/step  
Epoch 78/100  
1/1 - 0s - loss: 0.6069 - accuracy: 0.3333 - val_loss: 0.8047 - val_accuracy:  
y: 0.3333 - 411ms/epoch - 411ms/step  
Epoch 79/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8061 - val_accuracy:  
y: 0.3333 - 415ms/epoch - 415ms/step  
Epoch 80/100  
1/1 - 0s - loss: 0.6070 - accuracy: 0.3333 - val_loss: 0.8075 - val_accuracy:  
y: 0.3333 - 419ms/epoch - 419ms/step  
Epoch 81/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8089 - val_accuracy:  
y: 0.3333 - 423ms/epoch - 423ms/step  
Epoch 82/100  
1/1 - 0s - loss: 0.6071 - accuracy: 0.3333 - val_loss: 0.8103 - val_accuracy:  
y: 0.3333 - 427ms/epoch - 427ms/step  
Epoch 83/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8117 - val_accuracy:  
y: 0.3333 - 431ms/epoch - 431ms/step  
Epoch 84/100  
1/1 - 0s - loss: 0.6072 - accuracy: 0.3333 - val_loss: 0.8131 - val_accuracy:  
y: 0.3333 - 435ms/epoch - 435ms/step  
Epoch 85/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8145 - val_accuracy:  
y: 0.3333 - 439ms/epoch - 439ms/step  
Epoch 86/100  
1/1 - 0s - loss: 0.6073 - accuracy: 0.3333 - val_loss: 0.8159 - val_accuracy:  
y: 0.3333 - 443ms/epoch - 443ms/step  
Epoch 87/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8173 - val_accuracy:  
y: 0.3333 - 447ms/epoch - 447ms/step  
Epoch 88/100  
1/1 - 0s - loss: 0.6074 - accuracy: 0.3333 - val_loss: 0.8187 - val_accuracy:  
y: 0.3333 - 451ms/epoch - 451ms/step  
Epoch 89/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8201 - val_accuracy:  
y: 0.3333 - 455ms/epoch - 455ms/step  
Epoch 90/100  
1/1 - 0s - loss: 0.6075 - accuracy: 0.3333 - val_loss: 0.8215 - val_accuracy:  
y: 0.3333 - 459ms/epoch - 459ms/step  
Epoch 91/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8229 - val_accuracy:  
y: 0.3333 - 463ms/epoch - 463ms/step  
Epoch 92/100  
1/1 - 0s - loss: 0.6076 - accuracy: 0.3333 - val_loss: 0.8243 - val_accuracy:  
y: 0.3333 - 467ms/epoch - 467ms/step  
Epoch 93/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8257 - val_accuracy:  
y: 0.3333 - 471ms/epoch - 471ms/step  
Epoch 94/100  
1/1 - 0s - loss: 0.6077 - accuracy: 0.3333 - val_loss: 0.8271 - val_accuracy:  
y: 0.3333 - 475ms/epoch - 475ms/step  
Epoch 95/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8285 - val_accuracy:  
y: 0.3333 - 479ms/epoch - 479ms/step  
Epoch 96/100  
1/1 - 0s - loss: 0.6078 - accuracy: 0.3333 - val_loss: 0.8299 - val_accuracy:  
y: 0.3333 - 483ms/epoch - 483ms/step  
Epoch 97/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8313 - val_accuracy:  
y: 0.3333 - 487ms/epoch - 487ms/step  
Epoch 98/100  
1/1 - 0s - loss: 0.6079 - accuracy: 0.3333 - val_loss: 0.8327 - val_accuracy:  
y: 0.3333 - 491ms/epoch - 491ms/step  
Epoch 99/100  
1/1 - 0s - loss: 0.6099 - accuracy: 0.3333 - val_loss: 0.8341 - val_accuracy:  
y: 0.3333 - 495ms/epoch - 495ms/step
```

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

```
1/1 [=====] - 0s 62ms/step - loss: 0.8233 - accuracy: 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_accuracy: 0.3333 - 2s/epoch - 2s/step
Epoch 2/100
1/1 - 0s - loss: 1.0978 - accuracy: 0.1667 - val_loss: 1.1582 - val_accuracy: 0.3333 - 112ms/epoch - 112ms/step
Epoch 3/100
1/1 - 0s - loss: 1.0890 - accuracy: 0.2500 - val_loss: 1.1545 - val_accuracy: 0.3333 - 84ms/epoch - 84ms/step
Epoch 4/100
1/1 - 0s - loss: 1.0803 - accuracy: 0.3333 - val_loss: 1.1509 - val_accuracy: 0.3333 - 101ms/epoch - 101ms/step
Epoch 5/100
1/1 - 0s - loss: 1.0718 - accuracy: 0.3333 - val_loss: 1.1474 - val_accuracy: 0.3333 - 83ms/epoch - 83ms/step
Epoch 6/100
1/1 - 0s - loss: 1.0633 - accuracy: 0.3333 - val_loss: 1.1439 - val_accuracy: 0.3333 - 101ms/epoch - 101ms/step
Epoch 7/100
1/1 - 0s - loss: 1.0550 - accuracy: 0.3333 - val_loss: 1.1405 - val_accuracy: 0.3333 - 83ms/epoch - 83ms/step
```

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

```
1/1 [=====] - 0s 53ms/step - loss: 1.1729 - accuracy: 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 #
<hr/>		
dense_48 (Dense)	(None, 32)	2016
dense_49 (Dense)	(None, 4)	132
<hr/>		
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,)

Epoch 1/100
1/1 - 2s - loss: 1.3713 - accuracy: 0.5000 - val_loss: 1.3709 - val_accuracy: 0.3333 - 2s/epoch - 2s/step
Epoch 2/100
1/1 - 0s - loss: 1.3626 - accuracy: 0.5000 - val_loss: 1.3667 - val_accuracy: 0.3333 - 63ms/epoch - 63ms/step
Epoch 3/100
1/1 - 0s - loss: 1.3539 - accuracy: 0.5000 - val_loss: 1.3624 - val_accuracy: 0.3333 - 63ms/epoch - 63ms/step
Epoch 4/100
1/1 - 0s - loss: 1.3452 - accuracy: 0.5000 - val_loss: 1.3581 - val_accuracy: 0.3333 - 78ms/epoch - 78ms/step
Epoch 5/100
1/1 - 0s - loss: 1.3365 - accuracy: 0.5000 - val_loss: 1.3538 - val_accuracy: 0.3333 - 109ms/epoch - 109ms/step
Epoch 6/100
1/1 - 0s - loss: 1.3279 - accuracy: 0.5000 - val_loss: 1.3496 - val_accuracy: 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 - accuracy: 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,batch_size=32)  
Epoch 1/100  
1/1 - 2s - loss: 1.6387 - accuracy: 0.1667 - val_loss: 1.6957 - val_accuracy: 0.0000e+00 - 2s/epoch - 2s/step  
Epoch 2/100  
1/1 - 0s - loss: 1.6291 - accuracy: 0.2500 - val_loss: 1.6907 - val_accuracy: 0.0000e+00 - 121ms/epoch - 121ms/step  
Epoch 3/100  
1/1 - 0s - loss: 1.6196 - accuracy: 0.2500 - val_loss: 1.6856 - val_accuracy: 0.0000e+00 - 103ms/epoch - 103ms/step  
Epoch 4/100  
1/1 - 0s - loss: 1.6101 - accuracy: 0.2500 - val_loss: 1.6806 - val_accuracy: 0.0000e+00 - 125ms/epoch - 125ms/step  
Epoch 5/100  
1/1 - 0s - loss: 1.6006 - accuracy: 0.2500 - val_loss: 1.6757 - val_accuracy: 0.0000e+00 - 110ms/epoch - 110ms/step  
Epoch 6/100  
1/1 - 0s - loss: 1.5912 - accuracy: 0.2500 - val_loss: 1.6708 - val_accuracy: 0.0000e+00 - 177ms/epoch - 177ms/step  
Epoch 7/100  
1/1 [=====] - 0s 109ms/step - loss: 1.1554 - accuracy: 0.4000
```

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

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

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

```
In [ ]:
```