In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df=pd.read_csv('Sheet_1.csv')
```

In [3]:
```
df.head(3)
```

Out[3]:

	response_id	class	response_text	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnai
0	response_1	not_flagged	I try and avoid this sort of conflict	NaN	NaN	NaN	NaN	
1	response_2	flagged	Had a friend open up to me about his mental ad...	NaN	NaN	NaN	NaN	
2	response_3	flagged	I saved a girl from suicide once. She was goin...	NaN	NaN	NaN	NaN	

In [4]:
```
df['text']=df['response_text']
```

In [5]:
```
x=df['text']
x.head(3),x.nunique(),x.shape
```

Out[5]:
```
(0                I try and avoid this sort of conflict
 1    Had a friend open up to me about his mental ad...
 2    I saved a girl from suicide once. She was goin...
 Name: text, dtype: object,
 80,
 (80,))
```

In [6]:
```
df['class'].value_counts()
```

Out[6]:
```
not_flagged    55
flagged        25
Name: class, dtype: int64
```

In [7]:
```
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
df['class']=lb.fit_transform(df['class'])
df['class'].value_counts()
```

Out[7]:
```
1    55
0    25
Name: class, dtype: int64
```

In [8]:
```
y=df['class']
```

CountVectorizer

In [9]:
```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer(lowercase=True,stop_words='english',ngram_range=(1,1), t
x = cv.fit_transform(df['text'])
x
```

Out[9]:
```
<80x505 sparse matrix of type '<class 'numpy.int64'>'
        with 913 stored elements in Compressed Sparse Row format>
```

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
x = cv.fit_transform(df['text'])
x
```

In [10]:
```
from keras.utils import np_utils
npy=np_utils.to_categorical(y)
print(npy[:5])
```

```
[[0. 1.]
 [1. 0.]
 [1. 0.]
 [0. 1.]
 [0. 1.]]
```

ANN

In [11]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,npy,test_size=.25, random_st

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler(with_mean=False)
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)

import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy

model = Sequential()
model.add(Dense(units=16,activation='relu', input_dim=505))
model.add(Dense(units=2,activation='sigmoid'))


model.compile(optimizer= 'Adam', loss='categorical_crossentropy',metrics=['ac


model.summary()

model.fit(x_train,y_train, epochs=10, verbose=1)
```

```
(60, 505)
(20, 505)
(60, 2)
(20, 2)
(60, 505)
(20, 505)
(60, 2)
(20, 2)
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 16)                8096

 dense_1 (Dense)             (None, 2)                 34

=================================================================
Total params: 8,130
Trainable params: 8,130
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10

C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework\inde
xed_slices.py:444: UserWarning: Converting sparse IndexedSlices(IndexedSlic
es(indices=Tensor("gradient_tape/sequential/dense/embedding_lookup_sparse/R
eshape_1:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/sequ
ential/dense/embedding_lookup_sparse/Reshape:0", shape=(None, 16), dtype=fl
oat32), dense_shape=Tensor("gradient_tape/sequential/dense/embedding_lookup
_sparse/Cast:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown sh
ape. This may consume a large amount of memory.
  warnings.warn(
```

```
2/2 [==============================] - 1s 4ms/step - loss: 0.8844 - accurac
y: 0.6667
Epoch 2/10
2/2 [==============================] - 0s 3ms/step - loss: 0.7433 - accurac
y: 0.6833
Epoch 3/10
2/2 [==============================] - 0s 3ms/step - loss: 0.6267 - accurac
y: 0.7333
Epoch 4/10
2/2 [==============================] - 0s 4ms/step - loss: 0.5591 - accurac
y: 0.7667
Epoch 5/10
2/2 [==============================] - 0s 4ms/step - loss: 0.5015 - accurac
y: 0.7667
Epoch 6/10
2/2 [==============================] - 0s 3ms/step - loss: 0.4517 - accurac
y: 0.7833
Epoch 7/10
2/2 [==============================] - 0s 2ms/step - loss: 0.4159 - accurac
y: 0.8000
Epoch 8/10
2/2 [==============================] - 0s 2ms/step - loss: 0.3848 - accurac
y: 0.8167
Epoch 9/10
2/2 [==============================] - 0s 4ms/step - loss: 0.3564 - accurac
y: 0.8167
Epoch 10/10
2/2 [==============================] - 0s 2ms/step - loss: 0.3355 - accurac
y: 0.8167
```

Out[11]: <keras.callbacks.History at 0x1611c8183d0>

In [12]:
```
y_pred=model.predict(x_test)
y_pred
```

```
1/1 [==============================] - 0s 84ms/step
```

Out[12]:
```
array([[0.5290749 , 0.8302494 ],
       [0.56156003, 0.53598475],
       [0.8187493 , 0.89032173],
       [0.61716294, 0.6754408 ],
       [0.6080903 , 0.73351926],
       [0.16555728, 0.4859988 ],
       [0.33346853, 0.6169858 ],
       [0.24020985, 0.80633867],
       [0.45718488, 0.5602619 ],
       [0.5195393 , 0.57742447],
       [0.41907957, 0.5384567 ],
       [0.40747267, 0.70419014],
       [0.34848258, 0.8408444 ],
       [0.48398453, 0.60092   ],
       [0.40631115, 0.8984616 ],
       [0.18614896, 0.83808017],
       [0.27955285, 0.4629471 ],
       [0.42439848, 0.5979068 ],
       [0.5134166 , 0.8559857 ],
       [0.50435966, 0.53181094]], dtype=float32)
```

```
https://www.geeksforgeeks.org/numpy-argmax-python/
```

In [13]:
```
y_pred=np.argmax(y_pred,axis=1)
y_test=np.argmax(y_test,axis=1)

y_pred,y_test
```

Out[13]:
```
(array([1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       dtype=int64),
 array([0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1],
       dtype=int64))
```

In [14]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_pred))
print(confusion_matrix (y_test,y_pred))
print(classification_report(y_pred,y_test))
```

```
0.55
[[ 0  8]
 [ 1 11]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.92      0.58      0.71        19

    accuracy                           0.55        20
   macro avg       0.46      0.29      0.35        20
weighted avg       0.87      0.55      0.67        20
```

BernoulliNB

In [15]:
```
from sklearn.naive_bayes import BernoulliNB
from sklearn import metrics
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25)
nb= BernoulliNB().fit(x_train,y_train)
print(nb.score(x_train,y_train))
print(nb.score(x_test,y_test))
y_pred_nb= nb.predict(x_test)
print(len(y_pred_nb))
print(y_pred_nb)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

print(metrics.accuracy_score(y_test,y_pred_nb))
print(confusion_matrix (y_test,y_pred_nb))
print(classification_report(y_pred_nb,y_test))
```

```
0.8
0.75
20
[1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1]
0.75
[[ 1  5]
 [ 0 14]]
              precision    recall  f1-score   support

           0       0.17      1.00      0.29         1
           1       1.00      0.74      0.85        19

    accuracy                           0.75        20
   macro avg       0.58      0.87      0.57        20
weighted avg       0.96      0.75      0.82        20
```

MultinomialNB

In [16]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25)
nb= MultinomialNB().fit(x_train,y_train)
print(nb.score(x_train,y_train))
print(nb.score(x_test,y_test))
y_pred_nb= nb.predict(x_test)
print(len(y_pred_nb))
print(y_pred_nb)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

print(metrics.accuracy_score(y_test,y_pred_nb))
print(confusion_matrix (y_test,y_pred_nb))
```

```
1.0
0.6
20
[0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0 0 0 1]
0.6
[[3 4]
 [4 9]]
```

DecisionTreeClassifier

In [17]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
y_pred_dtc=dtc.predict(x_test)
print(f'Predicted_y{y_pred_dtc[:5]} Actual_y{y_test.values[:5]}')
print(confusion_matrix(y_pred_dtc,y_test))
from sklearn.metrics import confusion_matrix,classification_report,accuracy_s
cm=confusion_matrix(y_pred_dtc,y_test)

print(classification_report(y_pred_dtc,y_test))
print(f'model_score- {dtc.score(x_test,y_test)} ')
print(f'accuracy_score- {accuracy_score(y_pred_dtc,y_test)} ')
```

```
Predicted_y[0 1 1 1 1] Actual_y[1 1 1 0 0]
[[4 4]
 [3 9]]
              precision    recall  f1-score   support

           0       0.57      0.50      0.53         8
           1       0.69      0.75      0.72        12

    accuracy                           0.65        20
   macro avg       0.63      0.62      0.63        20
weighted avg       0.64      0.65      0.65        20

model_score- 0.65
accuracy_score- 0.65
```

RandomForestClassifier

In [18]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred_rfc=rfc.predict(x_test)
print(f' predicted_y-{y_pred_rfc} actual_y-{y_test.values}')
print(confusion_matrix(y_pred_rfc,y_test))
cm=confusion_matrix(y_pred_rfc,y_test)

print(classification_report(y_pred_rfc,y_test))
print(f'model_score- {rfc.score(x_test,y_test)} ')
print(f'accuracy_score- {accuracy_score(y_pred_rfc,y_test)} ')
```

```
 predicted_y-[1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1] actual_y-[1 1 1 0 0
0 1 1 1 1 0 1 1 0 0 1 1 1 0 1]
[[ 1  0]
 [ 6 13]]
              precision    recall  f1-score   support

           0       0.14      1.00      0.25         1
           1       1.00      0.68      0.81        19

    accuracy                           0.70        20
   macro avg       0.57      0.84      0.53        20
weighted avg       0.96      0.70      0.78        20

model_score- 0.7
accuracy_score- 0.7
```

In [19]:
```
x.shape
```

Out[19]:
```
(80, 505)
```

https://stackoverflow.com/questions/58636087/tensorflow-valueerror-failed-to-convert-a-numpy-array-to-a-tensor-unsupporte (https://stackoverflow.com/questions/58636087/tensorflow-valueerror-failed-to-convert-a-numpy-array-to-a-tensor-unsupporte)

Rnn

https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573 (https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573)

```
https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573
```

2. TF-IDF (Term Frequency-Inverse Document Frequency)

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from nltk.tokenize import RegexpTokenizer
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer(lowercase=True,stop_words='english',ngram_range=(1,1),
tokenizer = token.tokenize)
x = cv.fit_transform(df['text'])
x
```

In [20]:
```
x=x.toarray()
```

https://stackoverflow.com/questions/56634634/convert-2d-array-to-3d-numpy-array (https://stackoverflow.com/questions/56634634/convert-2d-array-to-3d-numpy-array)

In [21]:
```
x
```

Out[21]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

import numpy as np

x = np.asarray(x).astype(np.float32)

In [ ]:
```
```

In [22]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,npy,test_size=.25, random_st
```

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

```
(60, 505)
(20, 505)
(60, 2)
(20, 2)
```

In [24]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler(with_mean=False)
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [25]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(60, 505)
(20, 505)
(60, 2)
(20, 2)
```

In [26]:
```
#data.reshape((data.shape[0], data.shape[1], 1)
```

https://stackoverflow.com/questions/56634634/convert-2d-array-to-3d-numpy-array (https://stackoverflow.com/questions/56634634/convert-2d-array-to-3d-numpy-array)

In [27]:
```
x_train.reshape((x_train.shape[0],x_train.shape[1],1))
```

Out[27]:
```
array([[[0.        ],
        [0.        ],
        [0.        ],
        ...,
        [0.        ],
        [0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ],
        [0.        ],
        ...,
        [0.        ],
        [0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ],
        [0.        ],
        ...,
        [0.        ],
        [0.        ],
        [0.        ]],

       ...,

       [[0.        ],
        [0.        ],
        [0.        ],
        ...,
        [0.        ],
        [0.        ],
        [2.38102737]],

       [[0.        ],
        [0.        ],
        [0.        ],
        ...,
        [0.        ],
        [0.        ],
        [0.        ]],

       [[0.        ],
        [0.        ],
        [0.        ],
        ...,
        [0.        ],
        [0.        ],
        [0.        ]]])
```

In [28]:
```
x_train.shape
```

Out[28]:
```
(60, 505)
```

https://stackoverflow.com/questions/44383080/memory-error-using-cv-fit-transformcorpus-toarray (https://stackoverflow.com/questions/44383080/memory-error-using-cv-fit-transformcorpus-toarray)

from keras.utils import pad_sequences

sequences=pad_sequences(,padding='post')

x_train=pad_sequences(x_train,paddinng='post',maxlen=50)
x_test=pad_sequences(x_test,padding='post',maxlen=50)

In [29]:

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
from keras.layers import SimpleRNN, Embedding, Flatten
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy

model = Sequential()
model.add(SimpleRNN(units=16,activation='relu', input_shape=(505,1), return_s
model.add(Dense(units=2,activation='sigmoid'))


model.compile(optimizer= 'Adam', loss='binary_crossentropy',metrics=['accurac


model.summary()

model.fit(x_train,y_train, epochs=100, verbose=1)
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 16)                288

 dense_2 (Dense)             (None, 2)                 34

=================================================================
Total params: 322
Trainable params: 322
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
2/2 [==============================] - 1s 37ms/step - loss: 0.7275 - acc
uracy: 0.4833
Epoch 2/100
2/2 [==============================] - 0s 37ms/step - loss: 0.7240 - acc
uracy: 0.6500
```

In [30]:
```
y_pred=model.predict(x_test)
y_pred
```

```
1/1 [==============================] - 0s 118ms/step
```

Out[30]:
```
array([[0.44554797, 0.5082591 ],
       [0.00200743, 0.9985292 ],
       [0.4028237 , 0.5233088 ],
       [0.3143194 , 0.2251133 ],
       [0.12363624, 0.8896469 ],
       [0.07748765, 0.9326416 ],
       [0.27747577, 0.73839784],
       [0.06107333, 0.94758236],
       [0.0086959 , 0.9932025 ],
       [0.2269269 , 0.7896671 ],
       [0.4466747 , 0.22622357],
       [0.49946702, 0.4481244 ],
       [0.38941324, 0.625926  ],
       [0.03892559, 0.96736944],
       [0.11870331, 0.8943035 ],
       [0.00812426, 0.9936687 ],
       [0.01481788, 0.98814094],
       [0.00529223, 0.9959538 ],
       [0.20647737, 0.809792  ],
       [0.00137774, 0.9990069 ]], dtype=float32)
```

In [31]:
```
y_pred=np.argmax(y_pred,axis=1)
y_test=np.argmax(y_test,axis=1)
```

In [32]:
```
y_pred,y_test
```

Out[32]:
```
(array([1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1],
       dtype=int64),
 array([0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1],
       dtype=int64))
```

https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/ (https://www.geeksforgeeks.org/python-word-embedding-using-word2vec/)

In [33]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_pred))
print(confusion_matrix (y_test,y_pred))
print(f'accuracy_score- {accuracy_score(y_pred,y_test)} ')
print(classification_report(y_pred,y_test))
```

```
0.65
[[ 2  6]
 [ 1 11]]
accuracy_score- 0.65
              precision    recall  f1-score   support

           0       0.25      0.67      0.36         3
           1       0.92      0.65      0.76        17

    accuracy                           0.65        20
   macro avg       0.58      0.66      0.56        20
weighted avg       0.82      0.65      0.70        20
```

https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/ (https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/)

LSTM

```
https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
```

In [34]:
```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
from keras.layers import SimpleRNN, Embedding, Flatten,LSTM
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy

model = Sequential()
model.add(LSTM(units=16,activation='relu', input_shape=(505,1), return_sequer
model.add(Dense(units=2,activation='sigmoid'))


model.compile(optimizer= 'Adam', loss='binary_crossentropy',metrics=['accurac


model.summary()

model.fit(x_train,y_train, epochs=100, verbose=1)
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 16)                1152

 dense_3 (Dense)             (None, 2)                 34

=================================================================
Total params: 1,186
Trainable params: 1,186
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
2/2 [==============================] - 1s 92ms/step - loss: 0.6977 - acc
uracy: 0.6667
Epoch 2/100
2/2 [==============================] - 0s 90ms/step - loss: 0.6962 - acc
uracy: 0.7167
```

In [35]:
```
y_pred=model.predict(x_test)
y_pred
```

```
1/1 [==============================] - 0s 142ms/step
```

Out[35]:
```
array([[nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan],
       [nan, nan]], dtype=float32)
```

In [36]:
```
y_pred=np.argmax(y_pred,axis=1)
#y_test=np.argmax(y_test,axis=1)
```

In [37]:
```
y_pred,y_test
```

Out[37]:
```
(array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       dtype=int64),
 array([0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1],
       dtype=int64))
```

In [38]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_pred))
print(confusion_matrix (y_test,y_pred))
print(f'accuracy_score- {accuracy_score(y_pred,y_test)} ')
print(classification_report(y_pred,y_test))
```

```
0.4
[[ 8  0]
 [12  0]]
accuracy_score- 0.4
              precision    recall  f1-score   support

           0       1.00      0.40      0.57        20
           1       0.00      0.00      0.00         0

    accuracy                           0.40        20
   macro avg       0.50      0.20      0.29        20
weighted avg       1.00      0.40      0.57        20


C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.
py:1248: UndefinedMetricWarning: Recall and F-score are ill-defined and bei
ng set to 0.0 in labels with no true samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.
py:1248: UndefinedMetricWarning: Recall and F-score are ill-defined and bei
ng set to 0.0 in labels with no true samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.
py:1248: UndefinedMetricWarning: Recall and F-score are ill-defined and bei
ng set to 0.0 in labels with no true samples. Use `zero_division` parameter
to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

GRU

https://keras.io/api/layers/recurrent_layers/gru/
(https://keras.io/api/layers/recurrent_layers/gru/)

In [39]:
```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout
from keras.layers import SimpleRNN, Embedding, Flatten,LSTM,GRU
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy

model = Sequential()
model.add(GRU(units=16,activation='relu', input_shape=(505,1), return_sequenc
model.add(Dense(units=2,activation='sigmoid'))


model.compile(optimizer= 'Adam', loss='binary_crossentropy',metrics=['accurac


model.summary()

model.fit(x_train,y_train, epochs=100, verbose=1)
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 16)                912

 dense_4 (Dense)             (None, 2)                 34

=================================================================
Total params: 946
Trainable params: 946
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
2/2 [==============================] - 1s 105ms/step - loss: 0.6915 - ac
curacy: 0.5833
Epoch 2/100
2/2 [==============================] - 0s 101ms/step - loss: 0.6903 - ac
curacy: 0.7000
```

In [40]:
```
y_pred=model.predict(x_test)
y_pred
```

```
1/1 [==============================] - 0s 156ms/step
```

Out[40]:
```
array([[0.29670092, 0.6961833 ],
       [0.11083812, 0.88630897],
       [0.28601623, 0.70755064],
       [0.58587563, 0.3691424 ],
       [0.16532816, 0.83468646],
       [0.14058456, 0.8578391 ],
       [0.17244563, 0.82796663],
       [0.14961837, 0.849338  ],
       [0.141444  , 0.85702777],
       [0.18862319, 0.81321305],
       [0.5374383 , 0.40817344],
       [0.35564858, 0.6464033 ],
       [0.2116094 , 0.7922293 ],
       [0.14127912, 0.8571783 ],
       [0.15363869, 0.8455734 ],
       [0.21538925, 0.78879493],
       [0.21345772, 0.79054934],
       [0.11205591, 0.8851269 ],
       [0.18560824, 0.81597793],
       [0.07772663, 0.9190981 ]], dtype=float32)
```

In [41]:
```
y_pred=np.argmax(y_pred,axis=1)
#y_test=np.argmax(y_test,axis=1)
```

In [42]:
```
y_pred,y_test
```

Out[42]:
```
(array([1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1],
       dtype=int64),
 array([0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1],
       dtype=int64))
```

In [43]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_pred))
print(confusion_matrix (y_test,y_pred))
print(f'accuracy_score- {accuracy_score(y_pred,y_test)} ')
print(classification_report(y_pred,y_test))
```

```
0.7
[[ 2  6]
 [ 0 12]]
accuracy_score- 0.7
              precision    recall  f1-score   support

           0       0.25      1.00      0.40         2
           1       1.00      0.67      0.80        18

    accuracy                           0.70        20
   macro avg       0.62      0.83      0.60        20
weighted avg       0.93      0.70      0.76        20
```

In [ ]: