```
!pip install matplotlib-venn
```

```
Requirement already satisfied: matplotlib-venn in /usr/local/lib/python3.10/dist-packages (0.11.9)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from matplotlib-venn) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from matplotlib-venn) (1.22.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from matplotlib-venn) (1.10.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (4.40.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (8.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (3.1.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->matplotlib-venn) (2.8.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->matplotlib-v
```

```
!apt-get -qq install -y libfluidsynth1
```

```
E: Package 'libfluidsynth1' has no installation candidate
```

```
# https://pypi.python.org/pypi/libarchive
!apt-get -qq install -y libarchive-dev && pip install -U libarchive
import libarchive
```

```
g previously unselected package libarchive-dev:amd64.
 database ... 123105 files and directories currently installed.)
g to unpack .../libarchive-dev_3.4.0-2ubuntu1.2_amd64.deb ...
g libarchive-dev:amd64 (3.4.0-2ubuntu1.2) ...
up libarchive-dev:amd64 (3.4.0-2ubuntu1.2) ...
ng triggers for man-db (2.9.1-1) ...
ng libarchive
ading libarchive-0.4.7.tar.gz (23 kB)
ing metadata (setup.py) ... done
ng nose (from libarchive)
ading nose-1.3.7-py3-none-any.whl (154 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 154.7/154.7 kB 13.9 MB/s eta 0:00:00
 wheels for collected packages: libarchive
ng wheel for libarchive (setup.py) ... done
d wheel for libarchive: filename=libarchive-0.4.7-py3-none-any.whl size=31629 sha256=47c6eeb5f1408c2f53b883d31109cd25d5932021a7b27e219b9
 in directory: /root/.cache/pip/wheels/3a/94/d0/6cd83c8a80a4236fd4cb2a1fd846ecf72ab1e0ac238c5951c0
ully built libarchive
ng collected packages: nose, libarchive
ully installed libarchive-0.4.7 nose-1.3.7
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
from keras.preprocessing.text import Tokenizer

from keras.models import Sequential
from keras.layers import Embedding, LSTM, GRU, Bidirectional, Dense, Dropout
from keras.callbacks import EarlyStopping
```

```
from keras.utils import pad_sequences
```

```
data = pd.read_csv("urdu-sentiment-corpus-v1.tsv", sep="\t")
```

```
data.head()
```

	Tweet	Class

```
data.info()
data.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Tweet   1000 non-null   object
 1   Class   999 non-null    object
dtypes: object(2)
memory usage: 15.8+ KB
```

	Tweet	Class
count	1000	999
unique	999	3
top	اللہ جانے وے ماہی تیرا پیار کی اے دل دی اوداسی...	N
freq	2	499

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

```
data.isna().values.any()
```

```
True
```

```
data.drop(data[data['Class']=='O'].index,inplace = True)
```

```
from sklearn.preprocessing import LabelEncoder
data['Class']=LabelEncoder().fit_transform(data['Class'])
piece = Tokenizer()
piece.fit_on_texts(data['Tweet'])
seq = piece.texts_to_sequences(data['Tweet'])
```

```
vocablen = len(piece.word_index) + 1
max_len = max(len(s) for s in seq)
x = pad_sequences(seq, maxlen=max_len, padding='post')
y = data['Class']
```

```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.25,random_state=101)
```

```
from tensorflow.keras.layers import Dense, Dropout,Flatten,LSTM,Bidirectional,SimpleRNN
```

```
dummy1 = Sequential()
em = Embedding(vocablen,100,input_length=max_len,trainable = True)
dummy1.add(em)
dummy1.add(SimpleRNN(100,return_sequences=True ,dropout=0.3))
dummy1.add(Dense(512,activation='tanh'))
dummy1.add(Flatten())
dummy1.add(Dense(1,activation='sigmoid'))
dummy1.compile(loss ="binary_crossentropy", optimizer = 'adam', metrics=["accuracy"])
```

```
dummy1.fit(xtrain,ytrain,batch_size=20,epochs = 10)
```

```
Epoch 1/10
37/37 [==============================] - 4s 34ms/step - loss: 0.7156 - accuracy: 0.5156
Epoch 2/10
37/37 [==============================] - 1s 33ms/step - loss: 0.3101 - accuracy: 0.8612
Epoch 3/10
37/37 [==============================] - 1s 33ms/step - loss: 0.0319 - accuracy: 0.9850
Epoch 4/10
37/37 [==============================] - 2s 59ms/step - loss: 0.0224 - accuracy: 0.9878
Epoch 5/10
37/37 [==============================] - 2s 47ms/step - loss: 0.0041 - accuracy: 0.9946
Epoch 6/10
37/37 [==============================] - 1s 38ms/step - loss: -0.0218 - accuracy: 0.9959
Epoch 7/10
```

```
37/37 [==============================] - 2s 48ms/step - loss: -0.0088 - accuracy: 0.9932
Epoch 8/10
37/37 [==============================] - 2s 41ms/step - loss: 0.0077 - accuracy: 0.9864
Epoch 9/10
37/37 [==============================] - 2s 40ms/step - loss: -2.0940e-04 - accuracy: 0.9891
Epoch 10/10
37/37 [==============================] - 1s 36ms/step - loss: 0.0061 - accuracy: 0.9864
<keras.callbacks.History at 0x7ff9c43669b0>
```

```
prediction1 = dummy1.predict(xtest)
```

```
8/8 [==============================] - 0s 10ms/step
```

```
import numpy as np
prediction2 = np.argmax(prediction1,axis=1)
ac = accuracy_score(ytest,prediction2)
pres = precision_score(ytest,prediction2,average = 'macro')
```

```
f1 = f1_score(ytest,prediction2, average = 'macro')
rec = recall_score(ytest,prediction2, average = 'macro')
```

```
dataset = pd.DataFrame({
 'precision':[pres],
 'recall' :[rec],
 'accuracy':[ac],
 'F1_score':[f1]
})
```

```
dataset
```

	precision	recall	accuracy	F1_score
0	0.244898	0.5	0.489796	0.328767

```
#BiLSTM
dummy2 = Sequential()
em = Embedding(vocablen,100,input_length=max_len,trainable = True)
dummy2.add(em)
dummy2.add(Bidirectional(LSTM(100,return_sequences=True)))
dummy2.add(Dropout(0.3))
dummy2.add(Dense(2,activation='relu'))
dummy2.add(Flatten())
dummy2.add(Dense(1,activation='sigmoid'))
dummy2.compile(loss ="binary_crossentropy", optimizer = 'sgd', metrics=["accuracy"])
```

```
dummy2.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 37, 100)           549800

 bidirectional (Bidirectiona  (None, 37, 200)          160800
 l)

 dropout (Dropout)           (None, 37, 200)           0

 dense_2 (Dense)             (None, 37, 2)             402

 flatten_1 (Flatten)         (None, 74)                0

 dense_3 (Dense)             (None, 1)                 75

=================================================================
Total params: 711,077
Trainable params: 711,077
Non-trainable params: 0
_________________________________________________________________
```

```
dummy2.fit(xtrain,ytrain,batch_size=30,epochs = 10)
```

```
Epoch 1/10
25/25 [==============================] - 7s 81ms/step - loss: 0.6935 - accuracy: 0.4707
Epoch 2/10
25/25 [==============================] - 2s 80ms/step - loss: 0.6933 - accuracy: 0.4844
Epoch 3/10
25/25 [==============================] - 3s 114ms/step - loss: 0.6932 - accuracy: 0.5252
Epoch 4/10
25/25 [==============================] - 2s 87ms/step - loss: 0.6929 - accuracy: 0.5197
Epoch 5/10
25/25 [==============================] - 2s 81ms/step - loss: 0.6929 - accuracy: 0.5170
Epoch 6/10
25/25 [==============================] - 2s 73ms/step - loss: 0.6930 - accuracy: 0.5116
Epoch 7/10
25/25 [==============================] - 2s 76ms/step - loss: 0.6929 - accuracy: 0.5129
Epoch 8/10
25/25 [==============================] - 2s 73ms/step - loss: 0.6928 - accuracy: 0.5143
Epoch 9/10
25/25 [==============================] - 2s 96ms/step - loss: 0.6930 - accuracy: 0.5224
Epoch 10/10
25/25 [==============================] - 4s 170ms/step - loss: 0.6929 - accuracy: 0.5143
<keras.callbacks.History at 0x7ff9b3e71300>
```

```
prediction1 = dummy2.predict(xtest)
```

```
8/8 [==============================] - 1s 21ms/step
```

```
prediction2 = np.argmax(prediction1,axis=1)
ac = accuracy_score(ytest,prediction2)
pres = precision_score(ytest,prediction2,average = 'macro')
f1 = f1_score(ytest,prediction2, average = 'macro')
rec = recall_score(ytest,prediction2, average = 'macro')
```

```
dataset2 = pd.DataFrame({
 'precision':[pres],
 'recall' :[rec],
 'accuracy':[ac],
 'F1_score':[f1]
})
```

```
dataset2
```

	precision	recall	accuracy	F1_score
0	0.244898	0.5	0.489796	0.328767

```
#GRU
dummy3 = Sequential()
em = Embedding(vocablen,100,input_length=max_len,trainable = True)
dummy3.add(em)
dummy3.add(GRU(100,return_sequences=True))
dummy3.add(Dense(2,activation='relu'))
dummy3.add(Flatten())
dummy3.add(Dense(1,activation='sigmoid'))
dummy3.compile(loss ="binary_crossentropy", optimizer = 'sgd', metrics=["accuracy"])
```

```
dummy3.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 37, 100)           549800

 gru (GRU)                   (None, 37, 100)           60600

 dense_4 (Dense)             (None, 37, 2)             202

 flatten_2 (Flatten)         (None, 74)                0

 dense_5 (Dense)             (None, 1)                 75

=================================================================
Total params: 610,677
Trainable params: 610,677
```

```
Non-trainable params: 0
_________________________________________________________________
```

```
dummy3.fit(xtrain,ytrain,batch_size=20,epochs = 10)
```

```
Epoch 1/10
37/37 [==============================] - 4s 38ms/step - loss: 0.6936 - accuracy: 0.4871
Epoch 2/10
37/37 [==============================] - 1s 38ms/step - loss: 0.6932 - accuracy: 0.5061
Epoch 3/10
37/37 [==============================] - 1s 38ms/step - loss: 0.6931 - accuracy: 0.5156
Epoch 4/10
37/37 [==============================] - 1s 37ms/step - loss: 0.6930 - accuracy: 0.5116
Epoch 5/10
37/37 [==============================] - 2s 43ms/step - loss: 0.6930 - accuracy: 0.5156
Epoch 6/10
37/37 [==============================] - 4s 98ms/step - loss: 0.6930 - accuracy: 0.5156
Epoch 7/10
37/37 [==============================] - 3s 71ms/step - loss: 0.6929 - accuracy: 0.5156
Epoch 8/10
37/37 [==============================] - 2s 50ms/step - loss: 0.6929 - accuracy: 0.5156
Epoch 9/10
37/37 [==============================] - 2s 66ms/step - loss: 0.6929 - accuracy: 0.5156
Epoch 10/10
37/37 [==============================] - 1s 40ms/step - loss: 0.6929 - accuracy: 0.5156
<keras.callbacks.History at 0x7ff9be673a30>
```

```
prediction1 = dummy3.predict(xtest)
```

```
8/8 [==============================] - 1s 10ms/step
```

```
prediction2 = np.argmax(prediction1,axis=1)
ac = accuracy_score(ytest,prediction2)
pres = precision_score(ytest,prediction2,average = 'macro')
f1 = f1_score(ytest,prediction2, average = 'macro')
rec = recall_score(ytest,prediction2, average = 'macro')
```

```
dummy3= pd.DataFrame({
 'precision':[pres],
 'recall' :[rec],
 'accuracy':[ac],
 'F1_score':[f1]
})
```

```
dummy3
```

	precision	recall	accuracy	F1_score
0	0.244898	0.5	0.489796	0.328767

```
#LSTM
dummy4 = Sequential()
em = Embedding(vocablen,100,input_length=max_len,trainable = True)
dummy4.add(em)
dummy4.add(LSTM(100,return_sequences=True))
dummy4.add(Dropout(0.7))
dummy4.add(Dense(2,activation='relu'))
dummy4.add(Flatten())
dummy4.add(Dense(1,activation='sigmoid'))
dummy4.compile(loss ="binary_crossentropy", optimizer = 'sgd', metrics=["accuracy"])
```

```
dummy4.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 37, 100)           549800

 lstm_1 (LSTM)               (None, 37, 100)           80400

 dropout_1 (Dropout)         (None, 37, 100)           0
```

```
 dense_6 (Dense)             (None, 37, 2)             202

 flatten_3 (Flatten)         (None, 74)                0

 dense_7 (Dense)             (None, 1)                 75

=================================================================
Total params: 630,477
Trainable params: 630,477
Non-trainable params: 0
_________________________________________________________________
```

```
dummy4.fit(xtrain,ytrain,batch_size=20,epochs = 10)
```

```
Epoch 1/10
37/37 [==============================] - 4s 38ms/step - loss: 0.6937 - accuracy: 0.4884
Epoch 2/10
37/37 [==============================] - 2s 41ms/step - loss: 0.6930 - accuracy: 0.5075
Epoch 3/10
37/37 [==============================] - 2s 41ms/step - loss: 0.6931 - accuracy: 0.5048
Epoch 4/10
37/37 [==============================] - 2s 60ms/step - loss: 0.6925 - accuracy: 0.5252
Epoch 5/10
37/37 [==============================] - 4s 119ms/step - loss: 0.6923 - accuracy: 0.5374
Epoch 6/10
37/37 [==============================] - 4s 97ms/step - loss: 0.6929 - accuracy: 0.5075
Epoch 7/10
37/37 [==============================] - 3s 67ms/step - loss: 0.6929 - accuracy: 0.5238
Epoch 8/10
37/37 [==============================] - 2s 62ms/step - loss: 0.6932 - accuracy: 0.5129
Epoch 9/10
37/37 [==============================] - 3s 78ms/step - loss: 0.6927 - accuracy: 0.5211
Epoch 10/10
37/37 [==============================] - 2s 41ms/step - loss: 0.6929 - accuracy: 0.5170
<keras.callbacks.History at 0x7ff9bc0b3c70>
```

```
prediction1= dummy4.predict(xtest)
```

```
prediction2 = np.argmax(prediction1,axis=1)
ac = accuracy_score(ytest,prediction2)
pres = precision_score(ytest,prediction2,average = 'macro')
f1 = f1_score(ytest,prediction2, average = 'macro')
rec = recall_score(ytest,prediction2, average = 'macro')
```

```
dummy4 = pd.DataFrame({
 'precision':[pres],
 'recall' :[rec],
 'accuracy':[ac],
 'F1_score':[f1]
})
```

```
dummy4
```

	precision	recall	accuracy	F1_score
0	0.244898	0.5	0.489796	0.328767

✓ 0s completed at 20:38