In [134]:

```
import pandas as pd
import csv
import re
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, GRU, Dense, Bidirectional,Dropout, S
impleRNN
from sklearn.metrics import accuracy_score, precision_score, recall_score,f1_score, class
ification_report
import nltk
from nltk.corpus import stopwords
import spacy
```

In [42]:

```
pip install spacy
```

```
Collecting spacy
  Downloading spacy-3.6.0-cp39-cp39-win_amd64.whl (12.3 MB)
Collecting langcodes<4.0.0,>=3.2.0
  Downloading langcodes-3.3.0-py3-none-any.whl (181 kB)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in c:\users\mannahil miftah\anaconda3\
lib\site-packages (from spacy) (4.62.3)
Collecting thinc<8.2.0,>=8.1.8
  Downloading thinc-8.1.10-cp39-cp39-win_amd64.whl (1.5 MB)
Collecting pathy>=0.10.0
  Downloading pathy-0.10.2-py3-none-any.whl (48 kB)
Requirement already satisfied: setuptools in c:\users\mannahil miftah\anaconda3\lib\site-
packages (from spacy) (58.0.4)
Requirement already satisfied: jinja2 in c:\users\mannahil miftah\anaconda3\lib\site-pack
ages (from spacy) (3.1.2)
Collecting typer<0.10.0,>=0.3.0
  Downloading typer-0.9.0-py3-none-any.whl (45 kB)
Collecting murmurhash<1.1.0,>=0.28.0
  Downloading murmurhash-1.0.9-cp39-cp39-win_amd64.whl (18 kB)
Collecting preshed<3.1.0,>=3.0.2
  Downloading preshed-3.0.8-cp39-cp39-win_amd64.whl (96 kB)
Collecting srsly<3.0.0,>=2.4.3
  Downloading srsly-2.4.6-cp39-cp39-win_amd64.whl (482 kB)
Collecting pydantic!=1.8,!=1.8.1,<1.11.0,>=1.7.4
  Downloading pydantic-1.10.11-cp39-cp39-win_amd64.whl (2.2 MB)
Collecting spacy-loggers<2.0.0,>=1.0.0
  Downloading spacy_loggers-1.0.4-py3-none-any.whl (11 kB)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in c:\users\mannahil miftah\anacon
da3\lib\site-packages (from spacy) (2.26.0)
Collecting cymem<2.1.0,>=2.0.2
  Downloading cymem-2.0.7-cp39-cp39-win_amd64.whl (30 kB)
Requirement already satisfied: numpy>=1.15.0 in c:\users\mannahil miftah\anaconda3\lib\si
te-packages (from spacy) (1.20.3)
Collecting wasabi<1.2.0,>=0.9.1
  Downloading wasabi-1.1.2-py3-none-any.whl (27 kB)
Collecting smart-open<7.0.0,>=5.2.1
  Downloading smart_open-6.3.0-py3-none-any.whl (56 kB)
Collecting spacy-legacy<3.1.0,>=3.0.11
  Downloading spacy_legacy-3.0.12-py2.py3-none-any.whl (29 kB)
Requirement already satisfied: packaging>=20.0 in c:\users\mannahil miftah\anaconda3\lib\
site-packages (from spacy) (21.0)
Collecting catalogue<2.1.0,>=2.0.6
  Downloading catalogue-2.0.8-py3-none-any.whl (17 kB)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\mannahil miftah\anaconda3\lib
\site-packages (from packaging>=20.0->spacy) (3.0.4)
Collecting typing-extensions>=4.2.0
  Downloading typing_extensions-4.7.1-py3-none-any.whl (33 kB)
```

```
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\mannahil miftah\anac
onda3\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\mannahil miftah\anaconda3\lib\sit
e-packages (from requests<3.0.0,>=2.13.0->spacy) (3.2)Note: you may need to restart the k
ernel to use updated packages.
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\mannahil miftah\anaconda
3\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mannahil miftah\anaconda3\l
ib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (2021.10.8)
Collecting blis<0.8.0,>=0.7.8
  Downloading blis-0.7.9-cp39-cp39-win_amd64.whl (7.0 MB)
Collecting confection<1.0.0,>=0.0.1
  Downloading confection-0.1.0-py3-none-any.whl (34 kB)
Requirement already satisfied: colorama in c:\users\mannahil miftah\anaconda3\lib\site-pa
ckages (from tqdm<5.0.0,>=4.38.0->spacy) (0.4.4)
Requirement already satisfied: click<9.0.0,>=7.1.1 in c:\users\mannahil miftah\anaconda3\
lib\site-packages (from typer<0.10.0,>=0.3.0->spacy) (8.0.3)
Collecting colorama
  Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\mannahil miftah\anaconda3\lib\
site-packages (from jinja2->spacy) (2.1.2)
Installing collected packages: typing-extensions, colorama, catalogue, srsly, pydantic, m
urmurhash, cymem, wasabi, typer, smart-open, preshed, confection, blis, thinc, spacy-logg
ers, spacy-legacy, pathy, langcodes, spacy
  Attempting uninstall: typing-extensions
    Found existing installation: typing-extensions 3.10.0.2
    Uninstalling typing-extensions-3.10.0.2:
      Successfully uninstalled typing-extensions-3.10.0.2
  Attempting uninstall: colorama
    Found existing installation: colorama 0.4.4
    Uninstalling colorama-0.4.4:
      Successfully uninstalled colorama-0.4.4
Successfully installed blis-0.7.9 catalogue-2.0.8 colorama-0.4.6 confection-0.1.0 cymem-2
.0.7 langcodes-3.3.0 murmurhash-1.0.9 pathy-0.10.2 preshed-3.0.8 pydantic-1.10.11 smart-o
pen-6.3.0 spacy-3.6.0 spacy-legacy-3.0.12 spacy-loggers-1.0.4 srsly-2.4.6 thinc-8.1.10 ty
per-0.9.0 typing-extensions-4.7.1 wasabi-1.1.2
```

In [110]:

```
data = pd.read_table(r'C:\Users\Mannahil Miftah\Downloads\urdu-sentiment-corpus-v1.tsv')
data
```

Out[110]:

	Tweet	Class
0	... میں ن□ ایٹم بم بنایا ھ□ □□□□او بھائی ایٹم بمب	P
1	...چند□ س□ انقلاب اور عمران خان وزیر اعظم ن□یں بن	N
2	ٹویٹر کا خیال کیس□ آیا ؟	O
3	...سرچ انجن گوگل ک□ نائب صدر ن□ فضا میں ، 130,000	P
4	ابھی تک اسکی ل□ریں کبھی کبھی آ جاتی □یں یار :أْ	P
...	...	...
995	اُس آدمی ن□ اِس سالار کو کافی معقول ٹپ دی □□ □	P
996	... چچا غالب کی روح س□ معذرت ک□ ساتھ□م ن□ مانا ک□	P
997	...وا□ جناب وا□! اچھی ر□ی□ جناب خود کو فرشت□ سمجو	P
998	...اسلام آباد :پی ا□ ٹی کا دھرنا ختم، صفائی ک□ کا	P
999	...دنیا ن□ کس کا را□ وفا میں دیا □□ ساتھتم بھی چل	P

1000 rows × 2 columns

In [111]:

```
# saving the data into lists; Text and Labels
```

```
d = r"C:\Users\Mannahil Miftah\Downloads\urdu-sentiment-corpus-v1.tsv"

Text = []
Label = []

with open(d, 'r', encoding = 'utf-8') as file:
    data_csv = csv.reader(file, delimiter = '\t')
    next(data_csv)

    for row in data_csv:
        # any row which doesnot contain either text, label or both will be skipped
        if len(row) >= 2:
            Text.append(row[0].strip())
            Label.append(row[1])
```

In [112]:

```
# removing unnecessary charactes from the text

for i in range(len(Text)):
    Text[i] = re.sub(r'[^\w\s]', '', Text[i])
```

In [113]:

```
# as we have to do binary classification (P and N), so removing the text with label O

text = []
label = []
for i in range(len(Text)):
    if Label[i] == 'P' or Label[i] == 'N':
        text.append(Text[i])
        label.append(Label[i])
```

In [114]:

```
# printing the data after cleaning

for i in range(0,5):
    print("Text: ",text[i])
    print("Label: ",label[i])
```

```
Text:  میں نے ایٹم بم بنایا ھے او بھائی ایٹم بمب کوٹ لکھپت والی اتفاق فیکٹری میں نہیں بنت
ا ایٹم بم کھوٹہ کی ایٹمی
Label:  P
Text:  چندے سے انقلاب اور عمران خان وزیر اعظم نہیں بن سکتے
Label:  N
Text:  سرچ انجن گوگل کے نائب صدر نے فضا میں  130000 فٹ کی بلندی پر چھلانگ لگا کر عالمی ریک
ارڈ قائم کرلیا چھلانگ کی
Label:  P
Text:  ابھی تک اسکی لہریں کبھی کبھی آ جاتی ہیں یار أ
Label:  P
Text:  گندی زبان اور گٹر جیسے دماغ والے جاھل جیالے ھو تم جیالا ھو اور جاھل نہ ھو یہ ممکن ن
ھیں
Label:  N
```

In [115]:

```
# splitting the data

X_train, X_test, y_train, y_test = train_test_split(text, label, test_size = 0.25, rando
m_state = 42)
```

In [116]:

```
# Encoding labels
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.transform(y_test)
```

In [117]:

```
# tokenizing

tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
```

In [167]:

```
# padding the sequences to ensure that shorter sequences will have same length as the lon
gest sequence or max length

max_length = 200
X_train = pad_sequences(X_train, maxlen = max_length, padding = 'post')
X_test = pad_sequences(X_test, maxlen = max_length, padding = 'post')
```

In [168]:

```
number_of_layers = [2, 3]
dropout_rates = [0.3, 0.7]
rnn_result = []
gru_result = []
lstm_result = []
bilstm_result = []
```

RNN MODEL

In [189]:

```
# Defining RNN model

def rnn(layers, dropout_rate):
    model_rnn = Sequential()
    model_rnn.add(Embedding(input_dim = len(tokenizer.word_index) + 1, output_dim = 100,
input_length = max_length))

    for _ in range(layers - 1):
        model_rnn.add(SimpleRNN(units = 64, return_sequences = True))
        model_rnn.add(Dropout(dropout_rate))

    model_rnn.add(SimpleRNN(units = 64))
    model_rnn.add(Dropout(dropout_rate))
    model_rnn.add(Dense(units = 1, activation = 'sigmoid'))

    # Compiling and training the RNN model
    model_rnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accu
racy'])
    model_rnn.fit(X_train, y_train, epochs = 10, batch_size = 20, validation_data = (X_t
est, y_test))
    return model_rnn
```

In [190]:

```
for n in number_of_layers:
    for d in dropout_rates:
        model = rnn(n, d)
        pred_rnn = np.round(model.predict(X_test))
        accuracy_rnn = accuracy_score(y_test, pred_rnn)
        precision_rnn = precision_score(y_test, pred_rnn, zero_division = 1)
        recall_rnn = recall_score(y_test, pred_rnn)
        f1_rnn = f1_score(y_test, pred_rnn)
        rnn_result.append({
            'Number of layers':n,
            'Dropout rate':d,
            'Accuracy':accuracy_rnn,
            'Precision':precision_rnn,
            'Recall':recall_rnn,
            'F1 score':f1_rnn
        })
```

```
Epoch 1/10
Epoch 1/10
37/37 [==============================] - 5s 85ms/step - loss: 0.7339 - accuracy: 0.4905 -
val_loss: 0.7234 - val_accuracy: 0.4735
Epoch 2/10
37/37 [==============================] - 3s 79ms/step - loss: 0.7294 - accuracy: 0.5054 -
val_loss: 0.7368 - val_accuracy: 0.4816
Epoch 3/10
37/37 [==============================] - 3s 79ms/step - loss: 0.7242 - accuracy: 0.5191 -
val_loss: 0.7375 - val_accuracy: 0.5061
Epoch 4/10
37/37 [==============================] - 3s 75ms/step - loss: 0.7492 - accuracy: 0.4918 -
val_loss: 0.6961 - val_accuracy: 0.4857
Epoch 5/10
37/37 [==============================] - 3s 88ms/step - loss: 0.7359 - accuracy: 0.4782 -
val_loss: 0.7129 - val_accuracy: 0.4939
Epoch 6/10
37/37 [==============================] - 3s 80ms/step - loss: 0.7311 - accuracy: 0.5041 -
val_loss: 0.6991 - val_accuracy: 0.4694
Epoch 7/10
37/37 [==============================] - 3s 79ms/step - loss: 0.7332 - accuracy: 0.4877 -
val_loss: 0.6941 - val_accuracy: 0.5306
Epoch 8/10
37/37 [==============================] - 3s 71ms/step - loss: 0.7344 - accuracy: 0.4741 -
val_loss: 0.6940 - val_accuracy: 0.5020
Epoch 9/10
37/37 [==============================] - 3s 84ms/step - loss: 0.7160 - accuracy: 0.4946 -
val_loss: 0.6988 - val_accuracy: 0.4857
Epoch 10/10
37/37 [==============================] - 3s 86ms/step - loss: 0.7327 - accuracy: 0.4837 -
val_loss: 0.7059 - val_accuracy: 0.4816
8/8 [==============================] - 0s 28ms/step
Epoch 1/10
37/37 [==============================] - 5s 99ms/step - loss: 0.8020 - accuracy: 0.5095 -
val_loss: 0.7096 - val_accuracy: 0.4653
Epoch 2/10
37/37 [==============================] - 4s 97ms/step - loss: 0.8129 - accuracy: 0.4932 -
val_loss: 0.6925 - val_accuracy: 0.5102
Epoch 3/10
37/37 [==============================] - 4s 101ms/step - loss: 0.7827 - accuracy: 0.4959
- val_loss: 0.6977 - val_accuracy: 0.5306
Epoch 4/10
37/37 [==============================] - 3s 84ms/step - loss: 0.8000 - accuracy: 0.5027 -
val_loss: 0.7097 - val_accuracy: 0.5388
Epoch 5/10
37/37 [==============================] - 3s 91ms/step - loss: 0.8164 - accuracy: 0.4986 -
val_loss: 0.7260 - val_accuracy: 0.4898
Epoch 6/10
37/37 [==============================] - 3s 83ms/step - loss: 0.8107 - accuracy: 0.5068 -
val_loss: 0.6935 - val_accuracy: 0.4898
Epoch 7/10
37/37 [==============================] - 3s 83ms/step - loss: 0.8090 - accuracy: 0.5095 -
val_loss: 0.6952 - val_accuracy: 0.5184
Epoch 8/10
37/37 [==============================] - 3s 94ms/step - loss: 0.7982 - accuracy: 0.5123 -
val_loss: 0.7000 - val_accuracy: 0.5429
Epoch 9/10
37/37 [==============================] - 3s 83ms/step - loss: 0.7818 - accuracy: 0.5163 -
val_loss: 0.6912 - val_accuracy: 0.5306
Epoch 10/10
37/37 [==============================] - 3s 86ms/step - loss: 0.8040 - accuracy: 0.4673 -
val_loss: 0.7049 - val_accuracy: 0.4735
8/8 [==============================] - 0s 20ms/step
Epoch 1/10
37/37 [==============================] - 6s 118ms/step - loss: 0.7795 - accuracy: 0.4986
- val_loss: 0.6993 - val_accuracy: 0.5143
Epoch 2/10
37/37 [==============================] - 4s 119ms/step - loss: 0.7841 - accuracy: 0.4714
- val_loss: 0.7337 - val_accuracy: 0.5020
Epoch 3/10
37/37 [==============================] - 5s 147ms/step - loss: 0.7397 - accuracy: 0.5368
- val_loss: 0.7476 - val_accuracy: 0.4898
Epoch 4/10
```

```
37/37 [==============================] - 6s 158ms/step - loss: 0.7910 - accuracy: 0.4619
- val_loss: 0.7145 - val_accuracy: 0.4612
Epoch 5/10
37/37 [==============================] - 5s 141ms/step - loss: 0.7574 - accuracy: 0.4741
- val_loss: 0.7187 - val_accuracy: 0.4776
Epoch 6/10
37/37 [==============================] - 5s 135ms/step - loss: 0.7447 - accuracy: 0.5109
- val_loss: 0.7135 - val_accuracy: 0.4571
Epoch 7/10
37/37 [==============================] - 6s 174ms/step - loss: 0.7367 - accuracy: 0.5054
- val_loss: 0.6946 - val_accuracy: 0.5469
Epoch 8/10
37/37 [==============================] - 5s 136ms/step - loss: 0.7590 - accuracy: 0.4809
- val_loss: 0.7252 - val_accuracy: 0.4776
Epoch 9/10
37/37 [==============================] - 6s 155ms/step - loss: 0.7201 - accuracy: 0.5245
- val_loss: 0.7230 - val_accuracy: 0.4776
Epoch 10/10
37/37 [==============================] - 5s 132ms/step - loss: 0.7428 - accuracy: 0.4687
- val_loss: 0.7155 - val_accuracy: 0.4571
8/8 [==============================] - 1s 32ms/step
Epoch 1/10
37/37 [==============================] - 7s 147ms/step - loss: 0.9304 - accuracy: 0.4973
- val_loss: 0.7135 - val_accuracy: 0.4694
Epoch 2/10
37/37 [==============================] - 5s 130ms/step - loss: 0.9122 - accuracy: 0.4850
- val_loss: 0.6914 - val_accuracy: 0.5306
Epoch 3/10
37/37 [==============================] - 5s 127ms/step - loss: 0.9477 - accuracy: 0.4605
- val_loss: 0.7282 - val_accuracy: 0.4694
Epoch 4/10
37/37 [==============================] - 5s 129ms/step - loss: 0.8710 - accuracy: 0.5123
- val_loss: 0.7019 - val_accuracy: 0.4694
Epoch 5/10
37/37 [==============================] - 5s 130ms/step - loss: 0.8690 - accuracy: 0.4905
- val_loss: 0.6969 - val_accuracy: 0.4694
Epoch 6/10
37/37 [==============================] - 6s 151ms/step - loss: 0.7990 - accuracy: 0.5341
- val_loss: 0.6938 - val_accuracy: 0.4694
Epoch 7/10
37/37 [==============================] - 5s 140ms/step - loss: 0.7909 - accuracy: 0.5272
- val_loss: 0.6915 - val_accuracy: 0.5306
Epoch 8/10
37/37 [==============================] - 7s 188ms/step - loss: 0.7908 - accuracy: 0.4973
- val_loss: 0.6916 - val_accuracy: 0.5306
Epoch 9/10
37/37 [==============================] - 6s 155ms/step - loss: 0.7854 - accuracy: 0.5000
- val_loss: 0.6938 - val_accuracy: 0.5306
Epoch 10/10
37/37 [==============================] - 5s 145ms/step - loss: 0.7981 - accuracy: 0.4809
- val_loss: 0.6917 - val_accuracy: 0.5306
8/8 [==============================] - 1s 36ms/step
```

In [191]:

```
# printing results

rnn_result
```

Out[191]:

```
[{'Number of layers': 2,
  'Dropout rate': 0.3,
  'Accuracy': 0.5142857142857142,
  'Precision': 0.4830508474576271,
  'Recall': 0.4956521739130435,
  'F1 score': 0.4892703862660944},
 {'Number of layers': 2,
  'Dropout rate': 0.7,
  'Accuracy': 0.5142857142857142,
  'Precision': 0.4830508474576271,
  'Recall': 0.4956521739130435,
  'F1 score': 0.4892703862660944},
  'F1 score': 0.4892703862660944},
 {'Number of layers': 3,
  'Dropout rate': 0.3,
  'Accuracy': 0.5142857142857142,
  'Precision': 0.4830508474576271,
  'Recall': 0.4956521739130435,
  'F1 score': 0.4892703862660944},
 {'Number of layers': 3,
  'Dropout rate': 0.7,
  'Accuracy': 0.5142857142857142,
  'Precision': 0.4830508474576271,
  'Recall': 0.4956521739130435,
  'F1 score': 0.4892703862660944},
 {'Number of layers': 2,
  'Dropout rate': 0.3,
  'Accuracy': 0.4816326530612245,
  'Precision': 0.4423076923076923,
  'Recall': 0.4,
  'F1 score': 0.4200913242009133},
 {'Number of layers': 2,
  'Dropout rate': 0.7,
  'Accuracy': 0.47346938775510206,
  'Precision': 0.4406779661016949,
  'Recall': 0.45217391304347826,
  'F1 score': 0.4463519313304721},
 {'Number of layers': 3,
  'Dropout rate': 0.3,
  'Accuracy': 0.45714285714285713,
  'Precision': 0.41964285714285715,
  'Recall': 0.40869565217391307,
  'F1 score': 0.41409691629955947},
 {'Number of layers': 3,
  'Dropout rate': 0.7,
  'Accuracy': 0.5306122448979592,
  'Precision': 1.0,
  'Recall': 0.0,
  'F1 score': 0.0}]
```

In [192]:

```
print(classification_report(y_test, pred_rnn))
```

```
              precision    recall  f1-score   support

           0       0.53      1.00      0.69       130
           1       0.00      0.00      0.00       115

    accuracy                           0.53       245
   macro avg       0.27      0.50      0.35       245
weighted avg       0.28      0.53      0.37       245
```

```
c:\Users\Mannahil Miftah\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\Mannahil Miftah\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\Mannahil Miftah\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
```

GRU MODEL

In [211]:

```python
# Defining GRU model

def gru(layers, dropout_rate):
    model_gru = Sequential()
    model_gru.add(Embedding(input_dim = len(tokenizer.word_index) + 1, output_dim = 100,
input_length = max_length))

    for _ in range(layers - 1):
        model_gru.add(GRU(units = 64, return_sequences = True))
        model_gru.add(Dropout(dropout_rate))

    model_gru.add(GRU(units = 64))
    model_gru.add(Dropout(dropout_rate))
    model_gru.add(Dense(units = 1, activation = 'sigmoid'))

    # Compiling and training the GRU model
    model_gru.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accu
racy'])
    model_gru.fit(X_train, y_train, epochs = 10, batch_size = 20, validation_data = (X_t
est, y_test))
    return model_gru
```

In [212]:

```python
for n in number_of_layers:
    for d in dropout_rates:
        model = gru(n, d)
        pred_gru = np.round(model.predict(X_test))
        accuracy_gru = accuracy_score(y_test, pred_gru)
        precision_gru = precision_score(y_test, pred_gru, zero_division = 1)
        recall_gru = recall_score(y_test, pred_gru)
        f1_gru = f1_score(y_test, pred_gru)
        gru_result.append({
            'Number of layers':n,
            'Dropout rate':d,
            'Accuracy':accuracy_gru,
            'Precision':precision_gru,
            'Recall':recall_gru,
            'F1 score':f1_gru
        })
```

```
Epoch 1/10
37/37 [==============================] - 18s 265ms/step - loss: 0.6945 - accuracy: 0.5054
- val_loss: 0.6961 - val_accuracy: 0.4694
Epoch 2/10
37/37 [==============================] - 9s 236ms/step - loss: 0.6958 - accuracy: 0.5068
- val_loss: 0.6923 - val_accuracy: 0.5306
Epoch 3/10
37/37 [==============================] - 7s 186ms/step - loss: 0.6945 - accuracy: 0.4946
- val_loss: 0.6923 - val_accuracy: 0.5306
Epoch 4/10
37/37 [==============================] - 7s 200ms/step - loss: 0.6942 - accuracy: 0.4700
- val_loss: 0.6923 - val_accuracy: 0.5306
Epoch 5/10
37/37 [==============================] - 7s 192ms/step - loss: 0.6938 - accuracy: 0.4946
- val_loss: 0.6930 - val_accuracy: 0.5306
Epoch 6/10
37/37 [==============================] - 17s 471ms/step - loss: 0.6944 - accuracy: 0.4823
- val_loss: 0.6921 - val_accuracy: 0.5306
Epoch 7/10
37/37 [==============================] - 17s 467ms/step - loss: 0.6966 - accuracy: 0.4700
- val_loss: 0.6917 - val_accuracy: 0.5306
Epoch 8/10
37/37 [==============================] - 17s 453ms/step - loss: 0.6933 - accuracy: 0.5109
- val_loss: 0.6930 - val_accuracy: 0.5306
Epoch 9/10
37/37 [==============================] - 15s 395ms/step - loss: 0.6949 - accuracy: 0.4714
- val_loss: 0.6948 - val_accuracy: 0.4694
Epoch 10/10
37/37 [==============================] - 16s 436ms/step - loss: 0.6951 - accuracy: 0.5068
- val_loss: 0.6917 - val_accuracy: 0.5306
8/8 [==============================] - 3s 140ms/step
```

```
8/8 [==============================] - 3s 140ms/step
Epoch 1/10
37/37 [==============================] - 25s 417ms/step - loss: 0.6962 - accuracy: 0.4837
- val_loss: 0.6938 - val_accuracy: 0.4694
Epoch 2/10
37/37 [==============================] - 7s 185ms/step - loss: 0.6930 - accuracy: 0.5109
- val_loss: 0.6942 - val_accuracy: 0.4694
Epoch 3/10
37/37 [==============================] - 9s 231ms/step - loss: 0.6982 - accuracy: 0.4946
- val_loss: 0.6921 - val_accuracy: 0.5306
Epoch 4/10
37/37 [==============================] - 6s 170ms/step - loss: 0.6974 - accuracy: 0.4918
- val_loss: 0.6935 - val_accuracy: 0.4694
Epoch 5/10
37/37 [==============================] - 6s 170ms/step - loss: 0.6976 - accuracy: 0.4414
- val_loss: 0.6926 - val_accuracy: 0.5306
Epoch 6/10
37/37 [==============================] - 7s 179ms/step - loss: 0.6941 - accuracy: 0.5204
- val_loss: 0.6924 - val_accuracy: 0.5306
Epoch 7/10
37/37 [==============================] - 7s 187ms/step - loss: 0.6929 - accuracy: 0.5218
- val_loss: 0.6928 - val_accuracy: 0.5306
Epoch 8/10
37/37 [==============================] - 7s 194ms/step - loss: 0.7028 - accuracy: 0.4687
- val_loss: 0.6928 - val_accuracy: 0.5306
Epoch 9/10
37/37 [==============================] - 7s 191ms/step - loss: 0.6938 - accuracy: 0.4932
- val_loss: 0.6923 - val_accuracy: 0.5306
Epoch 10/10
37/37 [==============================] - 8s 217ms/step - loss: 0.6956 - accuracy: 0.4932
- val_loss: 0.6924 - val_accuracy: 0.5306
8/8 [==============================] - 1s 63ms/step
Epoch 1/10
37/37 [==============================] - 16s 289ms/step - loss: 0.6946 - accuracy: 0.5082
- val_loss: 0.6913 - val_accuracy: 0.5306
Epoch 2/10
37/37 [==============================] - 9s 256ms/step - loss: 0.6982 - accuracy: 0.5082
- val_loss: 0.6926 - val_accuracy: 0.5306
Epoch 3/10
37/37 [==============================] - 10s 280ms/step - loss: 0.6930 - accuracy: 0.5177
- val_loss: 0.6921 - val_accuracy: 0.5306
Epoch 4/10
37/37 [==============================] - 12s 321ms/step - loss: 0.6946 - accuracy: 0.5027
- val_loss: 0.6928 - val_accuracy: 0.5306
Epoch 5/10
37/37 [==============================] - 27s 749ms/step - loss: 0.6935 - accuracy: 0.4973
- val_loss: 0.6940 - val_accuracy: 0.4694
Epoch 6/10
37/37 [==============================] - 24s 649ms/step - loss: 0.6951 - accuracy: 0.4959
- val_loss: 0.6934 - val_accuracy: 0.4694
Epoch 7/10
37/37 [==============================] - 23s 611ms/step - loss: 0.6952 - accuracy: 0.4605
- val_loss: 0.6948 - val_accuracy: 0.4694
Epoch 8/10
37/37 [==============================] - 23s 622ms/step - loss: 0.6950 - accuracy: 0.4728
- val_loss: 0.6931 - val_accuracy: 0.4694
Epoch 9/10
37/37 [==============================] - 23s 621ms/step - loss: 0.6936 - accuracy: 0.4823
- val_loss: 0.6924 - val_accuracy: 0.5306
Epoch 10/10
37/37 [==============================] - 23s 624ms/step - loss: 0.6947 - accuracy: 0.4891
- val_loss: 0.6941 - val_accuracy: 0.4694
8/8 [==============================] - 5s 211ms/step
Epoch 1/10
37/37 [==============================] - 24s 347ms/step - loss: 0.6937 - accuracy: 0.4755
- val_loss: 0.6915 - val_accuracy: 0.5306
Epoch 2/10
37/37 [==============================] - 15s 408ms/step - loss: 0.6961 - accuracy: 0.4986
- val_loss: 0.6928 - val_accuracy: 0.5306
Epoch 3/10
37/37 [==============================] - 24s 651ms/step - loss: 0.6951 - accuracy: 0.4782
- val_loss: 0.6933 - val_accuracy: 0.4694
Epoch 4/10
Epoch 4/10
37/37 [==============================] - 26s 700ms/step - loss: 0.6945 - accuracy: 0.5286
- val_loss: 0.6930 - val_accuracy: 0.5306
Epoch 5/10
37/37 [==============================] - 24s 662ms/step - loss: 0.6976 - accuracy: 0.4877
- val_loss: 0.6923 - val_accuracy: 0.5306
Epoch 6/10
37/37 [==============================] - 24s 658ms/step - loss: 0.6948 - accuracy: 0.5259
- val_loss: 0.6955 - val_accuracy: 0.4694
Epoch 7/10
37/37 [==============================] - 24s 658ms/step - loss: 0.6962 - accuracy: 0.4959
- val_loss: 0.6925 - val_accuracy: 0.5306
Epoch 8/10
37/37 [==============================] - 24s 654ms/step - loss: 0.6965 - accuracy: 0.5136
- val_loss: 0.6950 - val_accuracy: 0.4694
Epoch 9/10
37/37 [==============================] - 22s 584ms/step - loss: 0.6957 - accuracy: 0.5000
- val_loss: 0.6920 - val_accuracy: 0.5306
Epoch 10/10
37/37 [==============================] - 24s 637ms/step - loss: 0.6989 - accuracy: 0.4728
- val_loss: 0.6935 - val_accuracy: 0.4694
8/8 [==============================] - 4s 224ms/step
```

In [213]:

```
# printing results

gru_result
```

Out[213]:

```
[{'Number of layers': 2,
  'Dropout rate': 0.3,
  'Accuracy': 0.5061224489795918,
  'Precision': 0.47368421052631576,
  'Recall': 0.46956521739130436,
  'F1 score': 0.47161572052401746},
 {'Number of layers': 2,
  'Dropout rate': 0.7,
  'Accuracy': 0.5061224489795918,
  'Precision': 0.47368421052631576,
  'Recall': 0.46956521739130436,
  'F1 score': 0.47161572052401746},
 {'Number of layers': 3,
  'Dropout rate': 0.3,
  'Accuracy': 0.5061224489795918,
  'Precision': 0.47368421052631576,
  'Recall': 0.46956521739130436,
  'F1 score': 0.47161572052401746},
 {'Number of layers': 3,
  'Dropout rate': 0.7,
  'Accuracy': 0.5061224489795918,
  'Precision': 0.47368421052631576,
  'Recall': 0.46956521739130436,
  'F1 score': 0.47161572052401746},
 {'Number of layers': 2,
  'Dropout rate': 0.3,
  'Accuracy': 0.5142857142857142,
  'Precision': 0.4830508474576271,
  'Recall': 0.4956521739130435,
  'F1 score': 0.4892703862660944},
 {'Number of layers': 2,
  'Dropout rate': 0.7,
  'Accuracy': 0.5142857142857142,
  'Precision': 0.4830508474576271,
  'Recall': 0.4956521739130435,
  'F1 score': 0.4892703862660944},
 {'Number of layers': 3,
  'Dropout rate': 0.3,
  'Accuracy': 0.5142857142857142,
  'Precision': 0.4830508474576271,
  'Recall': 0.4956521739130435,
  'F1 score': 0.4892703862660944},
 {'Number of layers': 3,
  'Dropout rate': 0.7,
  'Accuracy': 0.5142857142857142,
  'Precision': 0.4830508474576271,
  'Recall': 0.4956521739130435,
  'F1 score': 0.4892703862660944},
 {'Number of layers': 2,
  'Dropout rate': 0.3,
  'Accuracy': 0.5306122448979592,
  'Precision': 1.0,
  'Recall': 0.0,
  'F1 score': 0.0},
 {'Number of layers': 2,
  'Dropout rate': 0.7,
  'Accuracy': 0.46938775510204084,
  'Precision': 0.46938775510204084,
  'Recall': 1.0,
  'F1 score': 0.6388888888888888},
 {'Number of layers': 3,
  'Dropout rate': 0.3,
  'Accuracy': 0.5306122448979592,
  'Precision': 1.0,
  'Recall': 0.0,
  'F1 score': 0.0},
 {'Number of layers': 2,
  'Dropout rate': 0.3,
  'Accuracy': 0.5306122448979592,
  'Precision': 1.0,
  'Recall': 0.0,
  'F1 score': 0.0},
 {'Number of layers': 2,
  'Dropout rate': 0.7,
  'Accuracy': 0.5306122448979592,
  'Precision': 1.0,
  'Recall': 0.0,
  'F1 score': 0.0},
 {'Number of layers': 3,
  'Dropout rate': 0.3,
  'Accuracy': 0.46938775510204084,
  'Precision': 0.46938775510204084,
  'Recall': 1.0,
  'F1 score': 0.6388888888888888},
 {'Number of layers': 3,
  'Dropout rate': 0.7,
  'Accuracy': 0.46938775510204084,
  'Precision': 0.46938775510204084,
  'Recall': 1.0,
  'F1 score': 0.6388888888888888}]
```

In [214]:

```
print(classification_report(y_test, pred_gru))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       130
           1       0.47      1.00      0.64       115

    accuracy                           0.47       245
   macro avg       0.23      0.50      0.32       245
weighted avg       0.22      0.47      0.30       245
```

```
c:\Users\Mannahil Miftah\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\Mannahil Miftah\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
```

```
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\Mannahil Miftah\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
```

LSTM MODEL

In [196]:

```
# defining LSTM model

def lstm(layers, dropout_rate):
    model_lstm = Sequential()
    model_lstm.add(Embedding(input_dim = len(tokenizer.word_index) + 1, output_dim = 100
, input_length = max_length))

    for _ in range(layers - 1):
        model_lstm.add(LSTM(units = 64, return_sequences = True))
        model_lstm.add(Dropout(dropout_rate))

    model_lstm.add(LSTM(units = 64))
    model_lstm.add(Dropout(dropout_rate))
    model_lstm.add(Dense(units = 1, activation = 'sigmoid'))

    # compiling and training the LSTM model
    model_lstm.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['acc
uracy'])
    model_lstm.fit(X_train, y_train, epochs = 10, batch_size = 20, validation_data = (X_
test, y_test), verbose = 0)
    return model_lstm
```

In [197]:

```
for n in number_of_layers:
    for d in dropout_rates:
        model = lstm(n, d)
        pred_lstm = np.round(model.predict(X_test))
        accuracy_lstm = accuracy_score(y_test, pred_lstm)
        precision_lstm = precision_score(y_test, pred_lstm, zero_division = 1)
        recall_lstm = recall_score(y_test, pred_lstm)
        f1_lstm = f1_score(y_test, pred_lstm)
        lstm_result.append({
            'Number of layers':n,
            'Dropout rate':d,
            'Accuracy':accuracy_lstm,
            'Precision':precision_lstm,
            'Recall':recall_lstm,
            'F1 score':f1_lstm
        })
```

```
8/8 [==============================] - 2s 93ms/step
8/8 [==============================] - 1s 68ms/step
8/8 [==============================] - 2s 126ms/step
8/8 [==============================] - 3s 173ms/step
```

In [199]:

```
# printing the results

lstm_result
```

Out[199]:

```
[{'Number of layers': 2,
  'Dropout rate': 0.3,
  'Accuracy': 0.5306122448979592,
  'Precision': 1.0,
  'Recall': 0.0,
  'F1 score': 0.0},
 {'Number of layers': 2,
  'Dropout rate': 0.7,
  'Accuracy': 0.46938775510204084,
  'Precision': 0.46938775510204084,
  'Recall': 1.0,
  'F1 score': 0.6388888888888888},
 {'Number of layers': 3,
  'Dropout rate': 0.3,
  'Accuracy': 0.46938775510204084,
  'Precision': 0.46938775510204084,
  'Recall': 1.0,
  'F1 score': 0.6388888888888888},
 {'Number of layers': 3,
  'Dropout rate': 0.7,
  'Accuracy': 0.5306122448979592,
  'Precision': 1.0,
  'Recall': 0.0,
  'F1 score': 0.0}]
```

In [200]:

```
print(classification_report(y_test, pred_lstm))
```

```
              precision    recall  f1-score   support

           0       0.53      1.00      0.69       130
           1       0.00      0.00      0.00       115

    accuracy                           0.53       245
   macro avg       0.27      0.50      0.35       245
weighted avg       0.28      0.53      0.37       245
```

```
c:\Users\Mannahil Miftah\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\Mannahil Miftah\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\Mannahil Miftah\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1
248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 i
n labels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
```

BILSTM MODEL

In [205]:

```
# defining BiLSTM model

def bilstm(layers, dropout_rate):
    model_bilstm = Sequential()
    model_bilstm.add(Embedding(input_dim = len(tokenizer.word_index) + 1, output_dim = 1
00, input_length = max_length))

    for _ in range(layers - 1):
        model_bilstm.add(Bidirectional(LSTM(units = 64, return_sequences = True)))
        model_bilstm.add(Dropout(dropout_rate))

    model_bilstm.add(Bidirectional(LSTM(units = 64)))
    model_bilstm.add(Dropout(dropout_rate))
    model_bilstm.add(Dense(units = 1, activation = 'sigmoid'))

    # compiling and training the BiLSTM model
    model_bilstm.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['a
ccuracy'])
    model_bilstm.fit(X_train, y_train, epochs = 10, batch_size = 20, validation_data = (
X_test, y_test), verbose = 0)
    return model_bilstm
```

In [206]:

```
for n in number_of_layers:
    for d in dropout_rates:
        model = bilstm(n, d)
        pred_bilstm = np.round(model.predict(X_test))
        accuracy_bilstm = accuracy_score(y_test, pred_bilstm)
        precision_bilstm = precision_score(y_test, pred_bilstm, zero_division = 1)
        recall_bilstm = recall_score(y_test, pred_bilstm)
        f1_bilstm = f1_score(y_test, pred_bilstm)
        bilstm_result.append({
            'Number of layers':n,
            'Dropout rate':d,
            'Accuracy':accuracy_bilstm,
            'Precision':precision_bilstm,
            'Recall':recall_bilstm,
            'F1 score':f1_bilstm
        })
```

```
8/8 [==============================] - 2s 157ms/step
8/8 [==============================] - 2s 160ms/step
8/8 [==============================] - 3s 194ms/step
8/8 [==============================] - 7s 528ms/step
```

In [207]:

```
# printing the results

bilstm_result
```

Out[207]:

```
[{'Number of layers': 2,
  'Dropout rate': 0.3,
  'Accuracy': 0.5836734693877551,
  'Precision': 0.5355191256830601,
  'Recall': 0.8521739130434782,
  'F1 score': 0.6577181208053691},
 {'Number of layers': 2,
  'Dropout rate': 0.7,
  'Accuracy': 0.6571428571428571,
  'Precision': 0.624,
  'Recall': 0.6782608695652174,
  'F1 score': 0.65},
 {'Number of layers': 3,
  'Dropout rate': 0.3,
  'Accuracy': 0.6163265306122448,
  'Precision': 0.5755395683453237,
  'Recall': 0.6956521739130435,
  'F1 score': 0.6299212598425197},
 {'Number of layers': 3,
  'Dropout rate': 0.7,
  'Accuracy': 0.6204081632653061,
  'Precision': 0.5859375,
  'Recall': 0.6521739130434783,
  'F1 score': 0.6172839506172839}]
```

In [208]:

```
print(classification_report(y_test, pred_bilstm))
```

```
              precision    recall  f1-score   support

           0       0.66      0.59      0.62       130
           1       0.59      0.65      0.62       115

    accuracy                           0.62       245
```

```
   macro avg       0.62      0.62      0.62       245
weighted avg       0.62      0.62      0.62       245
```

```
In [ ]:
```