Aim: - Implement Fake News Classifier using LSTM-Deep Learning Model.
Requirement: -Python versions 3.7, 3.8, 3.9, 3.10 or 3.11, jupyter notebook

Theory: -

Long Short-Term Memory (LSTM):
LSTM is a type of recurrent neural network (RNN) designed to overcome the vanishing gradient problem in traditional RNNs. It can capture long-range dependencies in sequential data by using memory cells and gates to selectively store and retrieve information.

Fake News Classification:
Text Preprocessing:

- Clean and tokenize text data, removing stop words and special characters.
- Convert words to lowercase**.**

Word Embeddings:

- Represent words as vectors using pre-trained embeddings or train embeddings on the dataset**.**

LSTM Model:

- Build a deep learning model using LSTM layers to capture sequential patterns.
- Use dropout layers to prevent overfitting**.**
- Apply a dense layer with softmax activation for binary classification.

Training:

- Split the dataset into training and testing sets.
- Feed preprocessed text data into the LSTM model.
- Optimize using binary cross-entropy loss and an optimizer.

Evaluation:

- Assess model performance on the test set using metrics like accuracy, precision, recall, and F1 score.
- Adjust hyperparameters and architecture based on performance.

Code Implementation (using Python and TensorFlow/Keras):

- Tokenize and pad sequences for input.
- Build an LSTM model with embedding, LSTM, and dense layers.
- Compile the model with binary cross-entropy loss and an optimizer.
- Train the model on the training set and evaluate on the test set.
- Adjust hyperparameters for better performance and consider additional techniques like hyperparameter tuning and cross-validation.

Implement Fake News Classifier using LSTM-Deep Learning Model.

```
import pandas as pd
```

```
df = pd.read_csv('/content/train.csv')
```

```
df = df.dropna()
X = df.drop('label',axis=1)
y = df['label']
X.shape
y.shape
```

```
(18285,)
```

```
import tensorflow as tf
```

```
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
```

```
voc_size=5000
```

```
msgs = X.copy()
```

```
msgs['title'][1]
```

```
'FLYNN: Hillary Clinton, Big Woman on Campus - Breitbart'
```

```
msgs.reset_index(inplace=True)
```

```
import nltk
import re
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
corpus = []
for i in range(len(msgs)):
  print(i)
  review = re.sub('[^a-zA-Z]', ' ', msgs['title'][i])
  review = review.lower()
  review = review.split()
  review = [ps.stem(word) for word in review if not word in stopwords.words('english')]
  review = ' '.join(review)
  corpus.append(review)
```

```
Streaming output truncated to the last 5000 lines.
13285
13286
13287
13288
13289
13290
13291
13292
13293
13294
13295
13296
13297
13298
13299
13300
13301
13302
13303
```

```
review
```

```
review
```

```
'keep f aliv'
```

```
onehot_repr = [one_hot(words, voc_size) for words in corpus]
onehot_repr
```

```
[[3205, 2867, 979, 1957, 481, 4095, 1167, 1149, 3623, 471],
 [1290, 1932, 1351, 1811, 1219, 2271, 2933],
 [666, 1929, 1330, 4035],
 [952, 3625, 3904, 4198, 3239, 4150],
 [1519, 1219, 306, 2432, 1374, 3471, 1219, 3137, 135, 2594],
 [483,
  4916,
  814,
  4653,
  1608,
  606,
  4287,
  995,
  1399,
  4011,
  4033,
  2232,
  4364,
  577,
  2933],
 [3421, 82, 81, 4942, 2036, 3174, 2554, 322, 4610, 4356, 161],
 [616, 790, 3015, 2327, 3301, 4114, 606, 3919, 4610, 4356, 161],
 [3854, 2470, 4021, 4130, 420, 1818, 2645, 886, 606, 3126],
 [4409, 274, 3732, 3678, 846, 1709, 123, 885],
 [3330, 2367, 4421, 1051, 1131, 1981, 4233, 748, 4808, 2242, 2303],
 [4198, 193, 481, 1818, 606, 3301],
 [2637, 787, 3741, 3031, 3750, 2316, 219, 3451, 3074],
 [3386, 156, 2067, 806, 433, 820, 4451, 4610, 4356, 161],
 [4113, 4115, 1338, 2760, 4737, 4610, 4356, 161],
 [4772, 4973, 2244, 4950, 3850, 1093, 165, 410, 30, 452]
```

```
sent_length = 20
embedded_docs = pad_sequences(onehot_repr, padding='pre', maxlen=sent_length)
print(embedded_docs)
```

```
[[   0    0    0 ... 1149 3623  471]
 [   0    0    0 ... 1219 2271 2933]
 [   0    0    0 ... 1929 1330 4035]
 ...
 [   0    0    0 ... 4610 4356  161]
 [   0    0    0 ... 3220 1334 4321]
 [   0    0    0 ...  346 2578 1064]]
```

```
embedded_docs[0]
```

```
array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0, 3205,
       2867,  979, 1957,  481, 4095, 1167, 1149, 3623,  471], dtype=int32)
```

+ Code + Text

```
from nltk.tag import sequential
embedding_vector_features=40
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(LSTM(100))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 20, 40)            200000

 lstm (LSTM)                 (None, 100)               56400

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 256501 (1001.96 KB)
Trainable params: 256501 (1001.96 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```