Team Leader : Ariharasudhan S

Team Member 1 : Shenbaga Maharaja A

Team Member 3 : Esakki Durai R

Team Member 4 : Chandru S

Import required library

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Embedding
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing import sequence
```

Read Dataset & Do Pre-processing

```
df = pd.read_csv('/content/drive/MyDrive/ibm/spam.csv', delimiter=',', encoding='latin-1')
df.head()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

```
X = df.v2
Y = df.v1
encoder = LabelEncoder()
Y = encoder.fit_transform(Y)
Y = Y.reshape(-1,1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

```
tokenizer = Tokenizer(num_words=2000, lower=True)
tokenizer.fit_on_texts(X_train)
sequences = tokenizer.texts_to_sequences(X_train)
X_train = sequence.pad_sequences(sequences, maxlen=200)
```

Creating Model

```
model  = Sequential()
```

Adding Layers

```
model.add(Embedding(2000, 50, input_length=200))
model.add(LSTM(64))
model.add(Dense(256, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(1,activation="sigmoid"))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 50)           100000

 lstm (LSTM)                 (None, 64)                29440

 dense (Dense)               (None, 256)               16640

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 146,337
Trainable params: 146,337
Non-trainable params: 0
```

Model Compilation

```
model.compile(loss='binary_crossentropy', optimizer=RMSprop(), metrics=['accuracy'])
```

Fitting The Model

```
model.fit(X_train, y_train, batch_size=128, epochs=10, validation_split=0.2)
```

```
Epoch 1/10
28/28 [==============================] - 13s 344ms/step - loss: 0.3511 - accuracy: 0.8670 - val_loss: 0.1703 - val_accuracy: 0.9574
Epoch 2/10
28/28 [==============================] - 9s 317ms/step - loss: 0.1027 - accuracy: 0.9742 - val_loss: 0.0565 - val_accuracy: 0.9798
Epoch 3/10
28/28 [==============================] - 9s 321ms/step - loss: 0.0438 - accuracy: 0.9874 - val_loss: 0.0531 - val_accuracy: 0.9787
Epoch 4/10
28/28 [==============================] - 9s 332ms/step - loss: 0.0329 - accuracy: 0.9905 - val_loss: 0.0493 - val_accuracy: 0.9832
Epoch 5/10
28/28 [==============================] - 9s 325ms/step - loss: 0.0219 - accuracy: 0.9938 - val_loss: 0.0523 - val_accuracy: 0.9888
Epoch 6/10
28/28 [==============================] - 9s 324ms/step - loss: 0.0160 - accuracy: 0.9961 - val_loss: 0.0589 - val_accuracy: 0.9843
Epoch 7/10
28/28 [==============================] - 9s 322ms/step - loss: 0.0143 - accuracy: 0.9955 - val_loss: 0.0638 - val_accuracy: 0.9798
Epoch 8/10
28/28 [==============================] - 9s 318ms/step - loss: 0.0091 - accuracy: 0.9969 - val_loss: 0.0693 - val_accuracy: 0.9798
Epoch 9/10
28/28 [==============================] - 9s 319ms/step - loss: 0.0068 - accuracy: 0.9980 - val_loss: 0.0853 - val_accuracy: 0.9809
Epoch 10/10
28/28 [==============================] - 9s 323ms/step - loss: 0.0069 - accuracy: 0.9975 - val_loss: 0.0835 - val_accuracy: 0.9843
<keras.callbacks.History at 0x7f528e02bb90>
```

Saving The Model

```
model.save("model.h5")
```

Testing The Model

```
test_sequences = tokenizer.texts_to_sequences(X_test)
X_test = sequence.pad_sequences(test_sequences, maxlen=200)
acc = model.evaluate(X_test, y_test)
```

```
35/35 [==============================] - 1s 28ms/step - loss: 0.0609 - accuracy: 0.9865
```

```
def predict(message):
    txt = tokenizer.texts_to_sequences(message)
    txt = sequence.pad_sequences(txt, maxlen=200)
    preds = model.predict(txt)
    if preds > 0.5:
        print("Span")
    else:
        print("Not Span")
```

```
predict(["Sorry, I'll call after the meeting."])
```

```
1/1 [==============================] - 1s 508ms/step
Not Span
```

```
predict(["Congratulations!!! You won $50,000. Send message LUCKY100 to XXXXXXXXX to recieve your prize."])
```

```
1/1 [==============================] - 0s 28ms/step
Span
```

```
predict(["you won rupess 10,0000"])
```

```
1/1 [==============================] - 0s 32ms/step
Span
```

```
predict(["This is the very important problem"])
```

```
1/1 [==============================] - 0s 27ms/step
Not Span
```