Assignment -4
SMS Spam Classification

Project Name	AI-powered Nutrition Analyzer for Fitness Enthusiasts
Student Name	Kavinaya V
Student Roll Number	1921148
Maximum Marks	2 Marks

1.Download the dataset

2.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 keras.models import Model
    from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
    from keras.optimizers import RMSprop
    from keras.preprocessing.text import Tokenizer
    from keras_preprocessing import sequence
    from keras.utils import to_categorical
    from keras.models import load_model
```

3. Read Dataset and do preprocessing

```
df = pd.read_csv('/content/spam (1).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) #dropping
unwanted columns
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
```

```
# Count of Spam and Ham values
df.groupby(['v1']).size()
```

```
[ ] v1
    ham     4825
    spam     747
    dtype: int64
```

```
# Test and train split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)
```

```
# Label Encoding target column
X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

```
# Tokenisation function
max_words = 1000
max_len = 150
tok = Tokenizer(num_words=max_words)
tok.fit_on_texts(X_train)
sequences = tok.texts_to_sequences(X_train)
sequences_matrix = sequence.pad_sequences(sequences,maxlen=max_len)
```

4. Create Model and 5. Add Layers (LSTM, Dense-(Hidden Layers), Output)

```
# Creating LSTM model
inputs = Input(name='InputLayer',shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
layer = LSTM(64)(layer)
layer = Dense(256,name='FullyConnectedLayer1')(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1,name='OutputLayer')(layer)
layer = Activation('sigmoid')(layer)
```

6. Compile the model

```
model = Model(inputs=inputs,outputs=layer)
model.summary()
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 InputLayer (InputLayer)     [(None, 150)]             0

 embedding (Embedding)       (None, 150, 50)           50000

 lstm (LSTM)                 (None, 64)                29440

 FullyConnectedLayer1 (Dense  (None, 256)              16640
 )

 activation (Activation)     (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 OutputLayer (Dense)         (None, 1)                 257

 activation_1 (Activation)   (None, 1)                 0

=================================================================
Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0
_________________________________________________________________
```

7. Fit the Model

```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
validation_split=0.2)
```

```
Epoch 1/10
30/30 [==============================] - 12s 288ms/step - loss: 0.3478 - accuracy: 0.8704 - val_loss: 0.1900 - val_accuracy: 0.9262
Epoch 2/10
30/30 [==============================] - 8s 267ms/step - loss: 0.0927 - accuracy: 0.9770 - val_loss: 0.0929 - val_accuracy: 0.9747
Epoch 3/10
30/30 [==============================] - 8s 268ms/step - loss: 0.0432 - accuracy: 0.9876 - val_loss: 0.0740 - val_accuracy: 0.9800
Epoch 4/10
30/30 [==============================] - 8s 269ms/step - loss: 0.0309 - accuracy: 0.9918 - val_loss: 0.0648 - val_accuracy: 0.9810
Epoch 5/10
30/30 [==============================] - 8s 264ms/step - loss: 0.0255 - accuracy: 0.9931 - val_loss: 0.0674 - val_accuracy: 0.9810
Epoch 6/10
30/30 [==============================] - 8s 269ms/step - loss: 0.0178 - accuracy: 0.9952 - val_loss: 0.0605 - val_accuracy: 0.9852
Epoch 7/10
30/30 [==============================] - 8s 269ms/step - loss: 0.0162 - accuracy: 0.9960 - val_loss: 0.0633 - val_accuracy: 0.9852
Epoch 8/10
30/30 [==============================] - 8s 266ms/step - loss: 0.0129 - accuracy: 0.9958 - val_loss: 0.0723 - val_accuracy: 0.9863
Epoch 9/10
30/30 [==============================] - 8s 266ms/step - loss: 0.0097 - accuracy: 0.9976 - val_loss: 0.0704 - val_accuracy: 0.9884
Epoch 10/10
30/30 [==============================] - 8s 269ms/step - loss: 0.0070 - accuracy: 0.9976 - val_loss: 0.0700 - val_accuracy: 0.9863
<keras.callbacks.History at 0x7f6077793ad0>
```

8. Save the Model

```
model.save("model_1")
```

```
WARNING:absl:Function `_wrapped_model` contains input name(s) InputLayer with unsupported characters which will be renamed to inputlayer in the SavedModel.
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn, lstm_cell_layer_call_and_return_conditional_losses while saving (showing 2 of 2). These fur
```

9. Test the model

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = sequence.pad_sequences(test_sequences,maxlen=max_len)
accuracy = model.evaluate(test_sequences_matrix,Y_test)
print('Accuracy: {:0.3f}'.format(accuracy[1]))
```

```
27/27 [==============================] - 1s 22ms/step - loss: 0.0945 - accuracy: 0.9797
Accuracy: 0.980
```

```
y_pred = model.predict(test_sequences_matrix)
print(y_pred[25:40].round(3))
```

```
27/27 [==============================] - 1s 21ms/step
[[0.   ]
 [1.   ]
 [1.   ]
 [0.   ]
 [1.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [1.   ]
 [0.002]
 [0.   ]
 [1.   ]
 [0.   ]
 [0.   ]]
```

print(Y_test[25:40])

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