```
#Import necessary Libraries
import pandas as pd import
numpy as np import
matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('dark_background'
) import seaborn as sns from
nltk import word_tokenize

from sklearn.model_selection import train_test_split from sklearn.preprocessing import
LabelEncoder from sklearn.metrics import auc, roc_auc_score, roc_curve, confusion_matrix,
classification_r

from keras.models import Model, load_model from keras.layers import LSTM,
Activation, Dense, Dropout, Input, Embedding from keras.optimizers import
RMSprop from keras.preprocessing.text import Tokenizer from
tensorflow.keras.preprocessing.sequence import pad_sequences from
keras.utils import to_categorical, plot_model from keras.callbacks import
EarlyStopping, ModelCheckpoint, TensorBoard nltk.download('stopwords')
nltk.download('wordnet') nltk.download('punkt') nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date! True
```

```
df=pd.read_csv("/content/drive/MyDrive/IBM/Assignment - 4/spam.csv",encoding="latin-1");
df.head()
```

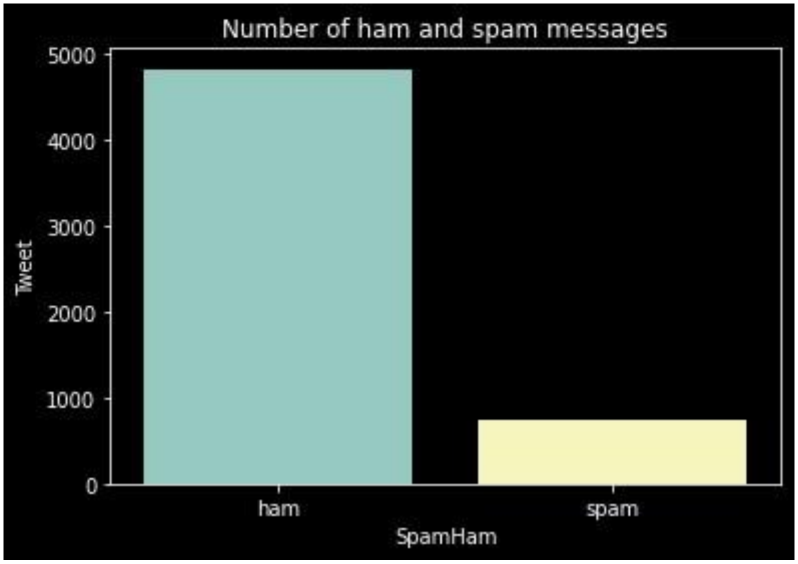
	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN

```
df.shape
```

```
(5572, 5)
```

```
fig, ax = plt.subplots()
sns.countplot(df.v1, ax=ax)
ax.set_xlabel('SpamHam')
ax.set_ylabel('Tweet')
ax.set_title('Number of ham and spam messages')
```

```
Text(0.5, 1.0, 'Number of ham and spam messages')
```



```
X = df.loc[:, 'v2']
y = df.loc[:, 'v1']
```

```
X
```

```
0                           Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2                           Free entry in 2 a wkly comp to win FA Cup fina...
3                           U dun say so early hor... U c already then say...
4                           Nah I don't think he goes to usf, he lives aro...
                              ...
5567                This is the 2nd time we have tried 2 contact u...
5568                Will Ì_ b going to esplanade fr home?
5569                Pity, * was in mood for that. So...any other s...
5570                The guy did some bitching but I acted like i'd...
5571                Rofl. Its true to its name Name: v2, Length: 5572, dtype: object
```

```
X_train_data, X_test_data, y_train_labels, y_test_labels = train_test_split(X, y, test_size=0
print(X_train_data.shape) print(X_test_data.shape)
```

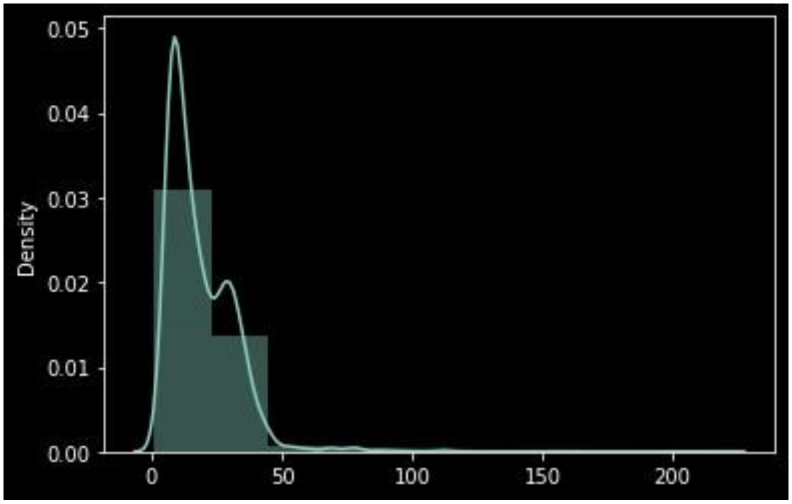
```
(4457,)
(1115,)
```

```
sent_lens = [] for sent
in X_train_data:
    sent_lens.append(len(word_tokenize(sent)))

print(max(sent_lens))
```

```
220
```

```
sns.distplot(sent_lens, bins=10, kde=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f81b671abd0>
```



```
np.quantile(sent_lens, 0.95)
```

```
39.0
```

```
max_sequence_length = 38

tok = Tokenizer()
tok.fit_on_texts(X_train_data.values)

vocab_length = len(tok.word_index) #len(tok.word_counts) or len(tok.index_word.keys()) will a
print('No. of unique tokens(vocab_size): ', vocab_length)

X_train_sequences = tok.texts_to_sequences(X_train_data.values) X_test_sequences =
tok.texts_to_sequences(X_test_data.values) print('No of sequences:', len(X_train_sequences))
#No of sequences will be same as the number print(X_train_sequences[:2])

#make all sequences of equal length
X_train = pad_sequences(X_train_sequences, maxlen=max_sequence_length)
X_test = pad_sequences(X_test_sequences, maxlen=max_sequence_length)
X_train[:2]
```

```
No. of unique tokens(vocab_size):  7954
No of sequences: 4457
[[38, 30, 8, 5, 273, 1989, 81, 116, 26, 11, 1656, 322, 10, 53, 18, 299, 30, 349, 1990],
array([[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,            0,
0,    0,    0,    0,    0,    0,    0,   38,   30,    8,
           5,  273, 1989,   81,  116,   26,   11, 1656,  322,   10,   53,
          18,  299,   30,  349, 1990],
       [   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,  799,   15, 2555, 1442, 1127,  192, 2556,
           171,    12,    98, 1991,    44,  195, 1657, 2557, 1992, 2558,    21,
     9,     4,  203, 1025,  225]], dtype=int32)
```

```
y_train_labels.values array(['ham', 'spam', 'ham', ..., 'ham', 'ham',

     'ham'], dtype=object)
```

```
le = LabelEncoder() y_train =
le.fit_transform(y_train_labels) y_test =
le.fit_transform(y_test_labels)
print(y_train)

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

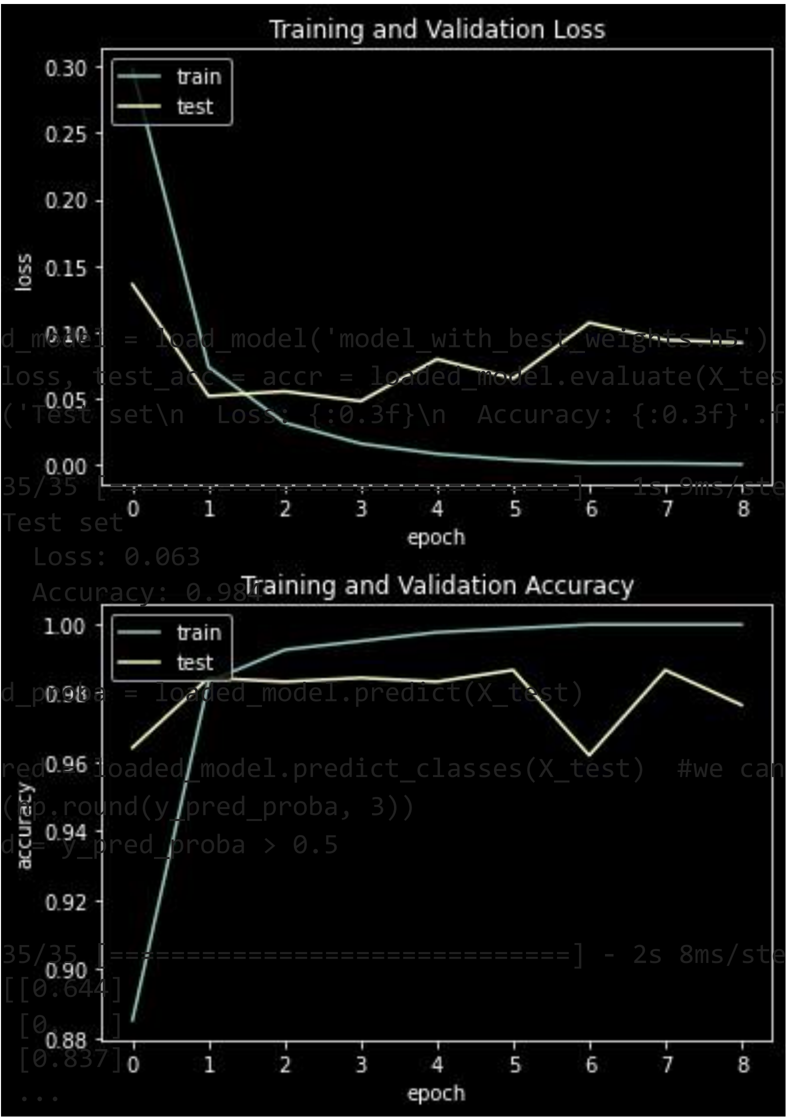
```
def create_model(vocab_len, max_seq_len):
    inputs = Input(name='inputs', shape=[max_seq_len])   #None, 150     layer =
Embedding(vocab_length + 1, 50, input_length=max_seq_len)(inputs) #None, 150, 50     layer
= LSTM(64)(layer)  #None, 64     layer = Dense(256,name='FC1')(layer) #None, 256     layer
= Activation('relu')(layer) #None, 256     layer = Dropout(0.5)(layer) #None, 256     layer
= Dense(1,name='out_layer')(layer) #None, 1     layer = Activation('sigmoid')(layer) #None,
1     model = Model(inputs=inputs,outputs=layer)
model.compile(loss='binary_crossentropy',optimizer=RMSprop(), metrics=['acc'])     return
model

model = create_model(vocab_length, max_sequence_length)
model.summary()

     Model: "model_1"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #
     =================================================================
     inputs (InputLayer)         [(None, 38)]              0
     embedding_1 (Embedding)     (None, 38, 50)            397750
     lstm_1 (LSTM)               (None, 64)                29440
      FC1 (Dense)                 (None, 256)               16640

      activation_2 (Activation)   (None, 256)               0
     dropout_1 (Dropout)         (None, 256)               0
     out_layer (Dense)           (None, 1)                 257
     activation_3 (Activation)   (None, 1)                 0
     =================================================================
     Total params: 444,087
     Trainable params: 444,087
     Non-trainable params: 0
     _________________________________________________________________
```

```
filepath='model_with_best_weights.h5'
callbacks = [EarlyStopping(monitor='val_loss', patience=5, verbose=1),
             ModelCheckpoint(filepath=filepath, monitor='val_loss', save_best_only=True, verb
             ]
```

```
history = model.fit(X_train, y_train, batch_size=128, epochs=20, validation_split=0.2, callba
```

```
Epoch 1/20
28/28 [==============================] - ETA: 0s - loss: 0.2985 - acc: 0.8853
Epoch 1: val_loss improved from inf to 0.13599, saving model to model_with_best_weights
28/28 [==============================] - 6s 99ms/step - loss: 0.2985 - acc: 0.8853 - val
Epoch 2/20
28/28 [==============================] - ETA: 0s - loss: 0.0737 - acc: 0.9832
Epoch 2: val_loss improved from 0.13599 to 0.05181, saving model to model_with_best_weig
28/28 [==============================] - 2s 76ms/step - loss: 0.0737 - acc: 0.9832 - val
Epoch 3/20
28/28 [==============================] - ETA: 0s - loss: 0.0318 - acc: 0.9924
Epoch 3: val_loss did not improve from 0.05181
28/28 [==============================] - 2s 75ms/step - loss: 0.0318 - acc: 0.9924 - val
Epoch 4/20
28/28 [==============================] - ETA: 0s - loss: 0.0160 - acc: 0.9950
Epoch 4: val_loss improved from 0.05181 to 0.04797, saving model to model_with_best_weig
28/28 [==============================] - 2s 78ms/step - loss: 0.0160 - acc: 0.9950 - val
Epoch 5/20
28/28 [==============================] - ETA: 0s - loss: 0.0083 - acc: 0.9975
Epoch 5: val_loss did not improve from 0.04797
28/28 [==============================] - 2s 74ms/step - loss: 0.0083 - acc: 0.9975 - val
Epoch 6/20
28/28 [==============================] - ETA: 0s - loss: 0.0039 - acc: 0.9986
Epoch 6: val_loss did not improve from 0.04797
28/28 [==============================] - 2s 74ms/step - loss: 0.0039 - acc: 0.9986 - val
Epoch 7/20
28/28 [==============================] - ETA: 0s - loss: 0.0014 - acc: 0.9997
Epoch 7: val_loss did not improve from 0.04797
28/28 [==============================] - 2s 74ms/step - loss: 0.0014 - acc: 0.9997 - val
Epoch 8/20
28/28 [==============================] - ETA: 0s - loss: 0.0012 - acc: 0.9997
Epoch 8: val_loss did not improve from 0.04797
28/28 [==============================] - 2s 75ms/step - loss: 0.0012 - acc: 0.9997 - val
Epoch 9/20
28/28 [==============================] - ETA: 0s - loss: 4.6690e-04 - acc: 0.9997
Epoch 9: val_loss did not improve from 0.04797
28/28 [==============================] - 2s 75ms/step - loss: 4.6690e-04 - acc: 0.9997
Epoch 9: early stopping
```

```
history_dict = history.history

# list all data in history
print(history_dict.keys())

# summarize history for loss
plt.plot(history_dict['loss'])
plt.plot(history_dict['val_loss'])
plt.title('Training and Validation Loss')
plt.ylabel('loss') plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper
left') plt.show()

# summarize history for accuracy
plt.plot(history_dict['acc'])
plt.plot(history_dict['val_acc'])
plt.title('Training and Validation Accuracy')
plt.ylabel('accuracy') plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper
left') plt.show()
     dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
y_pred
```



```
loaded_model = load_model('model_with_best_weights.h5')
test_loss, test_acc = accr = loaded_model.evaluate(X_test, y_test)
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(test_loss, test_acc))

     35/35 [==============================] - 1s 9ms/step - loss: 0.0634 - acc: 0.9839
     Test set
       Loss: 0.063
       Accuracy: 0.984


y_pred_proba = loaded_model.predict(X_test)

# y_pred = loaded_model.predict_classes(X_test)  #we can't use it on Model object. Can be use
print(np.round(y_pred_proba, 3))
y_pred = y_pred_proba > 0.5


     35/35 [==============================] - 2s 8ms/step
     [[0.644]
      [0.   ]
      [0.837]
      ...
      [0.   ]
      [0.001]
      [0.987]]
     array([[ True],
     [False],
            [ True],
            ...,
            [False],
            [False],
            [ True]])
```

```
# summarize the first few cases for i in range(5):     print('%s => %d
(expected %d)' % (X_test[i].tolist(), y_pred[i], y_test[i]))
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1366, 1578, 1432, 19, 78
[1, 188, 11, 6440, 2, 7, 1, 135, 2, 28, 12, 4, 290, 7931, 1, 104, 33, 3, 22, 647, 15, 28
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 39, 54, 258, 144, 3, 54, 21
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 64, 33, 3, 1528, 13, 263, 53, 79, 228, 79, 3, 31, 7, 838,
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 715, 29, 357, 532, 622, 15, 1107, 528, 706, 4
```

```
print(confusion_matrix(y_test, y_pred))
```

```
[[964   1]
 [ 17 133]]
```

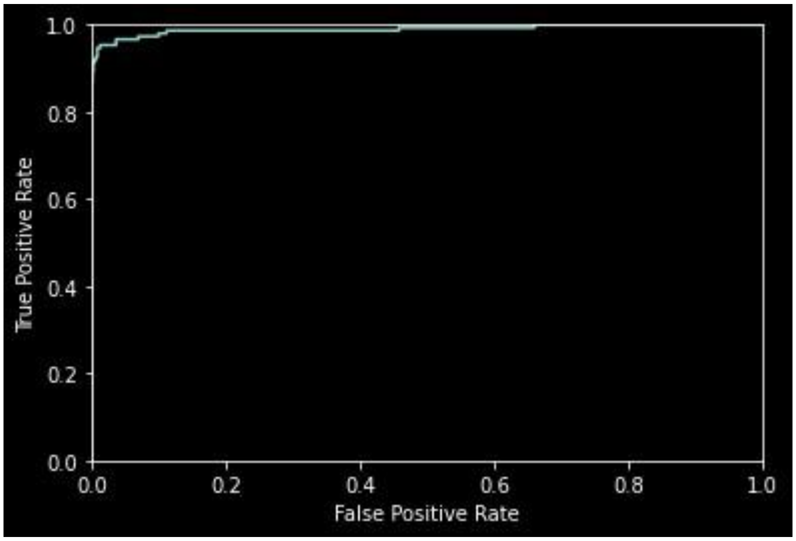
```
print(classification_report(y_test, y_pred))
```

```
precision    recall  f1-score   support

           0       0.98      1.00      0.99       965
     1       0.99      0.89      0.94       150

    accuracy                           0.98      1115
macro avg       0.99      0.94      0.96      1115 weighted
avg       0.98      0.98      0.98      1115
```

```
auc = roc_auc_score(y_test, y_pred_proba)
print('AUC: %.3f' % auc)
```

```
AUC: 0.990
```

```
fpr_keras, tpr_keras, thresholds_keras = roc_curve(y_test, y_pred_proba)

def plot_roc_curve(fpr,tpr):
import matplotlib.pyplot as plt
plt.plot(fpr,tpr)
plt.axis([0,1,0,1])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
   plot_roc_curve (fpr_keras,
tpr_keras)
```


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate