## Importing required libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from keras_preprocessing.sequence import pad_sequences
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.sequence import pad_sequences
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
import re
%matplotlib inline
```

## Reading Dataset

```
data = pd.read_csv("/content/spam.csv", encoding="ISO-8859-1")
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

## Data Preprocessing

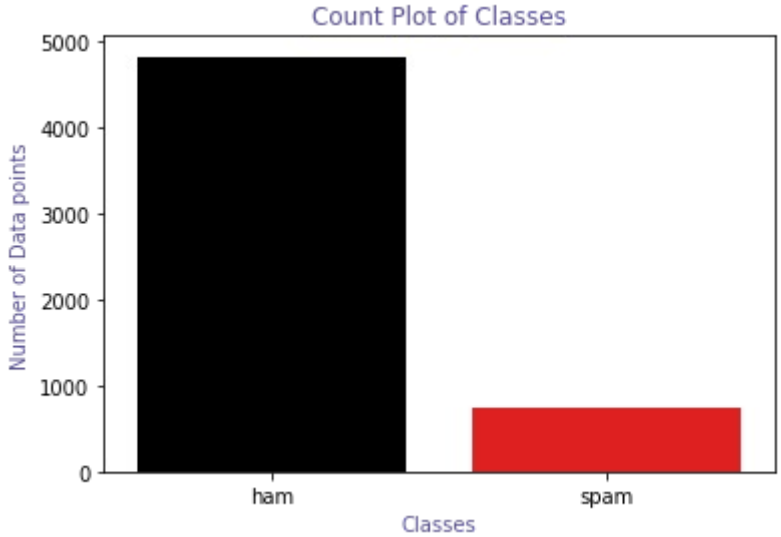
```
df = data.drop(data[["Unnamed: 2","Unnamed: 3","Unnamed: 4"]], axis=1)
df.rename(columns = {"v1":"Target", "v2":"Text"}, inplace = True)
df
```

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

5572 rows × 2 columns

```
plt.figure(figsize=(6,4))
fg = sns.countplot(x= df["Target"], palette= ["black", "red"] )
fg.set_title("Count Plot of Classes", color="#58508d")
fg.set_xlabel("Classes", color="#58508d")
fg.set_ylabel("Number of Data points", color="#58508d")
```

```
Text(0, 0.5, 'Number of Data points')
```




Double-click (or enter) to edit

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
df["No_of_Characters"] = df["Text"].apply(len)
```

```
df["No_of_Words"]=df.apply(lambda row: nltk.word_tokenize(row["Text"]), axis=1).apply(le
df["No_of_sentence"]=df.apply(lambda row: nltk.sent_tokenize(row["Text"]), axis=1).apply
df.describe().T
```

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **No_of_Characters** | 5572.0 | 80.118808 | 59.690841 | 2.0 | 36.0 | 61.0 | 121.0 | 910.0 |
| **No_of_Words** | 5572.0 | 18.695621 | 13.742587 | 1.0 | 9.0 | 15.0 | 27.0 | 220.0 |
| **No_of_sentence** | 5572.0 | 1.970747 | 1.417778 | 1.0 | 1.0 | 1.0 | 2.0 | 28.0 |

```
df.head()
```

| | Target | Text | No_of_Characters | No_of_Words | No_of_sentence |
|---|---|---|---|---|---|
| **0** | ham | Go until jurong point, crazy.. Available only ... | 111 | 24 | 2 |
| **1** | ham | Ok lar... Joking wif u oni... | 29 | 8 | 2 |
| **2** | spam | Free entry in 2 a wkly comp to win FA Cup fina... | 155 | 37 | 2 |
| **3** | ham | U dun say so early hor... U c already then say... | 49 | 13 | 1 |

```
plt.figure(figsize=(18,12))
fg = sns.pairplot(data=df, hue="Target",palette=["green","red"])
plt.show(fg)
```

```
<Figure size 1296x864 with 0 Axes>
```


```
def Clean(Text):
  sms = re.sub('[^a-zA-Z]', ' ', Text)
  sms = sms.lower() #converting to lowecase
  sms = sms.split()
  sms = ' '.join(sms)
  return sms
df["Clean_Text"] = df["Text"].apply(Clean)
df["Tokenize_Text"]=df.apply(lambda row: nltk.word_tokenize(row["Clean_Text"]), axis=1)
nltk.download('stopwords')
```

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

```
def remove_stopwords(text):
  stop_words = set(stopwords.words("english"))
  filtered_text = [word for word in text if word not in stop_words]
  return filtered_text
df["Nostopword_Text"] = df["Tokenize_Text"].apply(remove_stopwords)
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

```
lemmatizer = WordNetLemmatizer()
def lemmatize_word(text):
  lemmas = [lemmatizer.lemmatize(word, pos ='v') for word in text]
  return lemmas
df["Lemmatized_Text"] = df["Nostopword_Text"].apply(lemmatize_word)
corpus= []
for i in df["Lemmatized_Text"]:
  msg = ' '.join([row for row in i])
  corpus.append(msg)
corpus[:5]
```

```
['go jurong point crazy available bugis n great world la e buffet cine get amore
wat',
 'ok lar joke wif u oni',
 'free entry wkly comp win fa cup final tkts st may text fa receive entry question
std txt rate c apply',
```

```
      'u dun say early hor u c already say',
      'nah think go usf live around though']
```

```
df.tail()
```

| | Target | Text | No_of_Characters | No_of_Words | No_of_sentence | Clean_Text | T |
|---|---|---|---|---|---|---|---|
| **5567** | spam | This is the 2nd time we have tried 2 contact u... | 161 | 35 | 4 | this is the nd time we have tried contact u u ... | |
| **5568** | ham | Will Ì_ b going to esplanade fr home? | 37 | 9 | 1 | will b going to esplanade fr home | |
| **5569** | ham | Pity, * was in mood for that. So...any other s... | 57 | 15 | 2 | pity was in mood for that so any other suggest... | |
| **5570** | ham | The guy did some bitching but I acted like | 125 | 27 | 1 | the guy did some bitching but i acted like i d... | |

```
X = df.Clean_Text
Y = df.Target
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)
```

```
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 = pad_sequences(sequences,maxlen=max_len)
```

## Layers

```
def RNN():
  inputs = Input(name='inputs',shape=[max_len])
  layer = Embedding(max_words,50,input_length=max_len)(inputs)
  layer = LSTM(64)(layer)
  layer = Dense(256,name='FC1')(layer)
```

```
  layer = Activation('relu')(layer)
  layer = Dropout(0.5)(layer)
  layer = Dense(1,name='out_layer')(layer)
  layer = Activation('sigmoid')(layer)
  model = Model(inputs=inputs,outputs=layer)
  return model
```

## Compiling

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

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

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

 lstm (LSTM)                 (None, 64)                29440

 FC1 (Dense)                 (None, 256)               16640

 activation (Activation)     (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 out_layer (Dense)           (None, 1)                 257

 activation_1 (Activation)   (None, 1)                 0

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

## Fill model

```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=15,validation_split=0.2,callbac
```

```
Epoch 1/15
30/30 [==============================] - 1s 17ms/step - loss: 0.0427 - accuracy: 0.
Epoch 2/15
30/30 [==============================] - 0s 13ms/step - loss: 0.0301 - accuracy: 0.
<keras.callbacks.History at 0x7f5ac6ecddd0>
```

## Save model

```
model.save('lstm_model')
```

```
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn, lstm_cell_la
```

## Test model

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
accr = model.evaluate(test_sequences_matrix,Y_test)
```

```
27/27 [==============================] - 0s 6ms/step - loss: 0.0591 - accuracy: 0.9
```