

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

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.callbacks import EarlyStopping
import tensorflow
%matplotlib inline
```

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
df = pd.read_csv('spam.csv',delimiter=',',encoding='latin-1')
df.head()
```

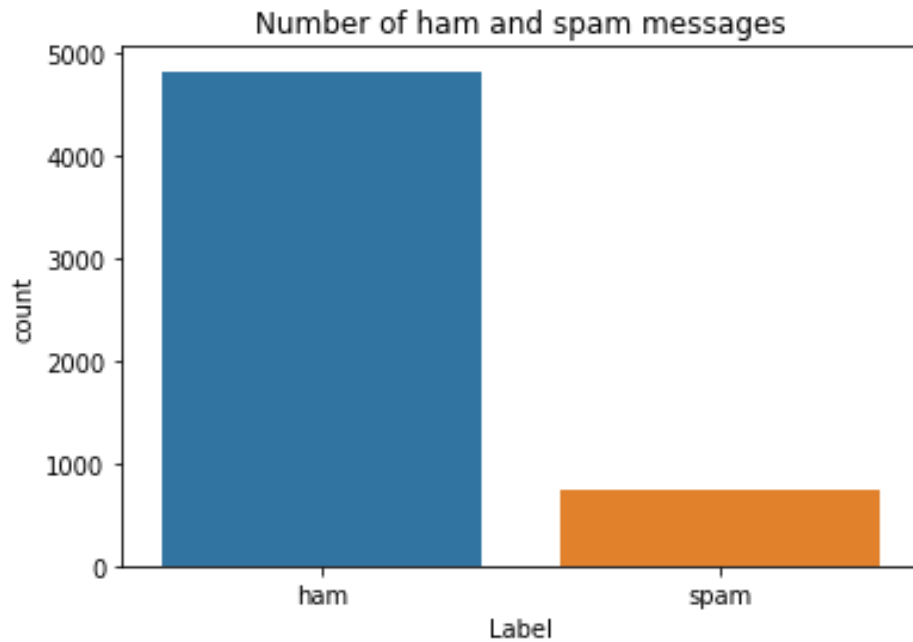
	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
...	...	U dun sav so early hor... U c already then

```
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
```

```
sns.countplot(df.v1)
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning
FutureWarning
Text(0.5, 1.0, 'Number of ham and spam messages')
```



```
X = df.v2
Y = df.v1
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)
```

```
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

```

```

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

```

Model: "model_1"

Layer (type)	Output Shape	Param #
=====		
inputs (InputLayer)	[(None, 150)]	0
embedding_1 (Embedding)	(None, 150, 50)	50000
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: 96,337		
Trainable params: 96,337		
Non-trainable params: 0		
=====		

```
df.columns
```

```
Index(['v1', 'v2'], dtype='object')
```

```

data=df.rename(
{
    "v1":"Category",
    "v2":"Message"
},
axis=1
)

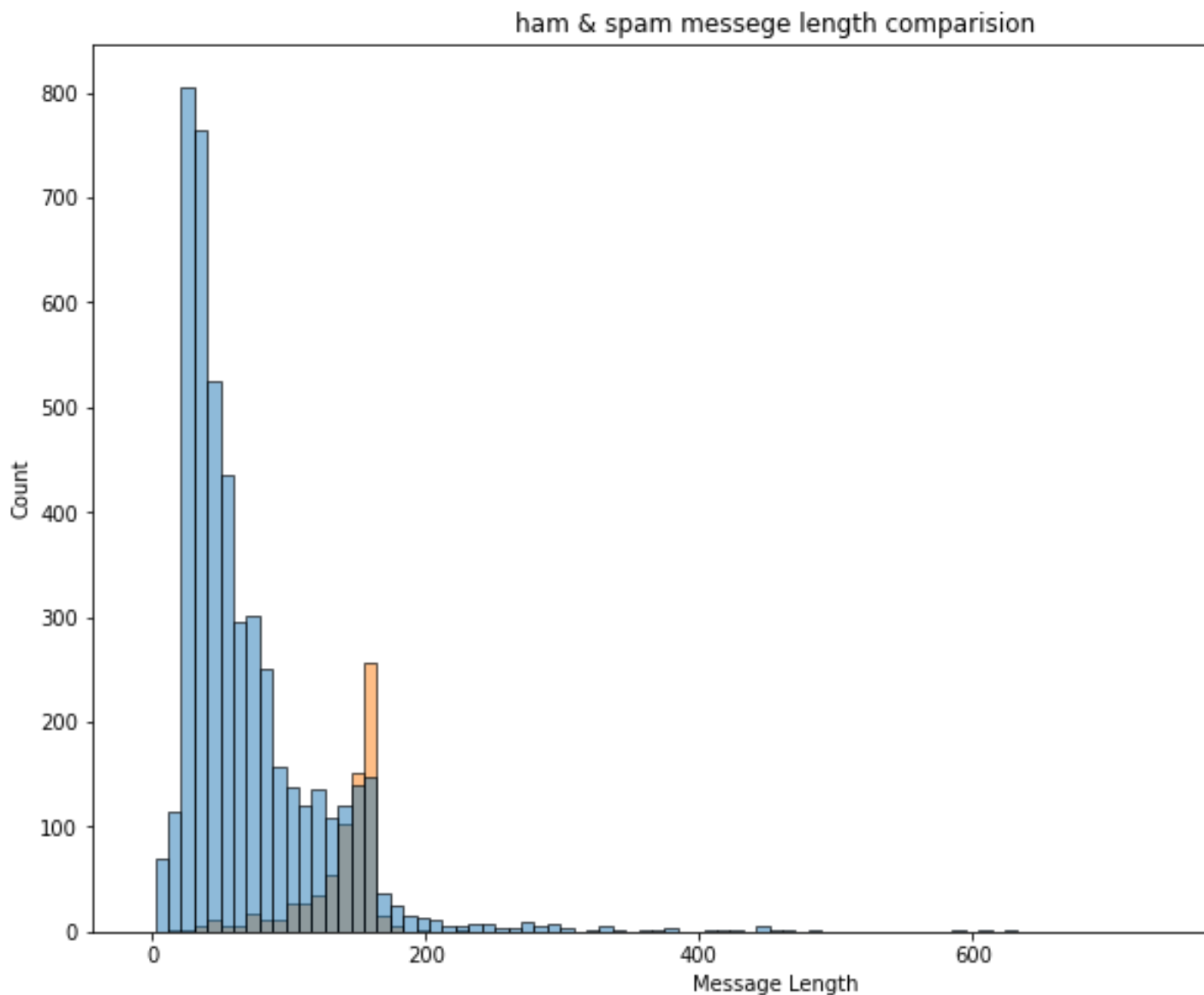
```

```
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
```

```
data["Message Length"]=data["Message"].apply(len)
```

```
fig=plt.figure(figsize=(12,8))
sns.histplot(
    x=data["Message Length"],
    hue=data["Category"]
)
plt.title("ham & spam messege length comparision")
plt.show()
```



```
ham_desc=data[data["Category"]=="ham"]["Message Length"].describe()
```

```
spam_desc=data[data["Category"]=="spam"]["Message Length"].describe()

print("Ham Messege Length Description:\n",ham_desc)
print("*****")
print("Spam Message Length Description:\n",spam_desc)
```

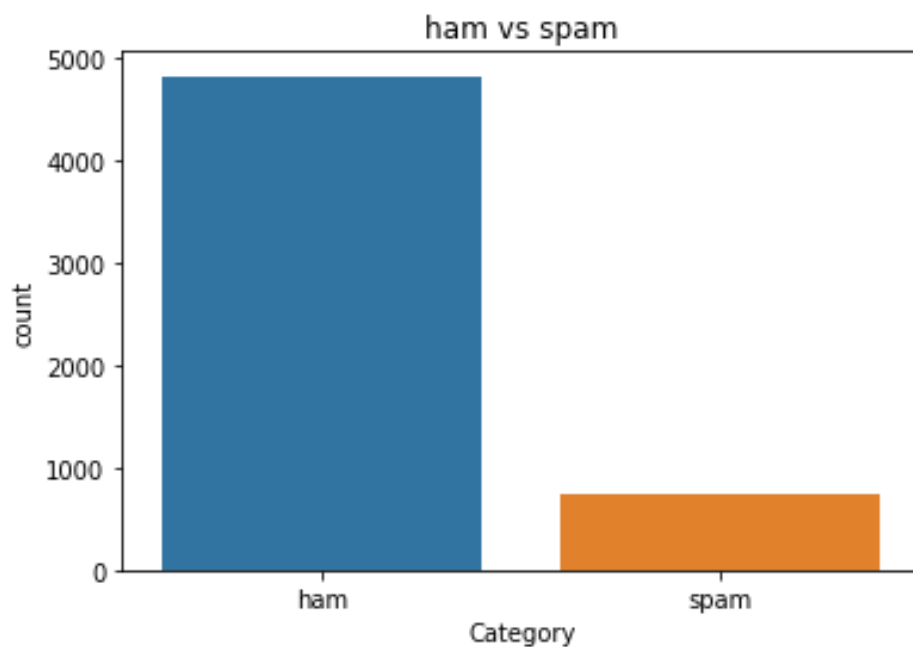
```
Ham Messege Length Description:
count      4825.000000
mean        71.023627
std         58.016023
min          2.000000
25%         33.000000
50%         52.000000
75%         92.000000
max        910.000000
Name: Message Length, dtype: float64
*****
Spam Message Length Description:
count       747.000000
mean       138.866131
std        29.183082
min        13.000000
25%       132.500000
50%       149.000000
75%       157.000000
max       224.000000
Name: Message Length, dtype: float64
```

```
data.describe(include="all")
```

```
data["Category"].value_counts()
```

```
ham      4825
spam      747
Name: Category, dtype: int64
```

```
sns.countplot(
    data=data,
    x="Category"
)
plt.title("ham vs spam")
plt.show()
```



```
ham_count=data["Category"].value_counts()[0]
spam_count=data["Category"].value_counts()[1]
```

```
total_count=data.shape[0]
```

```
print("Ham contains:{:.2f}% of total data.".format(ham_count/total_count*100))
print("Spam contains:{:.2f}% of total data.".format(spam_count/total_count*100))
```

```
Ham contains:86.59% of total data.
Spam contains:13.41% of total data.
```

```
minority_len=len(data[data["Category"]=="spam"])
majority_len=len(data[data["Category"]=="ham"])
minority_indices=data[data["Category"]=="spam"].index
majority_indices=data[data["Category"]=="ham"].index
random_majority_indices=np.random.choice(
    majority_indices,
```

```
        size=minority_len,  
        replace=False  
    )  
undersampled_indices=np.concatenate([minority_indices,random_majority_indices])  
df=data.loc[undersampled_indices]  
df=df.sample(frac=1)  
  
df=df.reset_index()  
df=df.drop(  
    columns=["index"],  
)
```

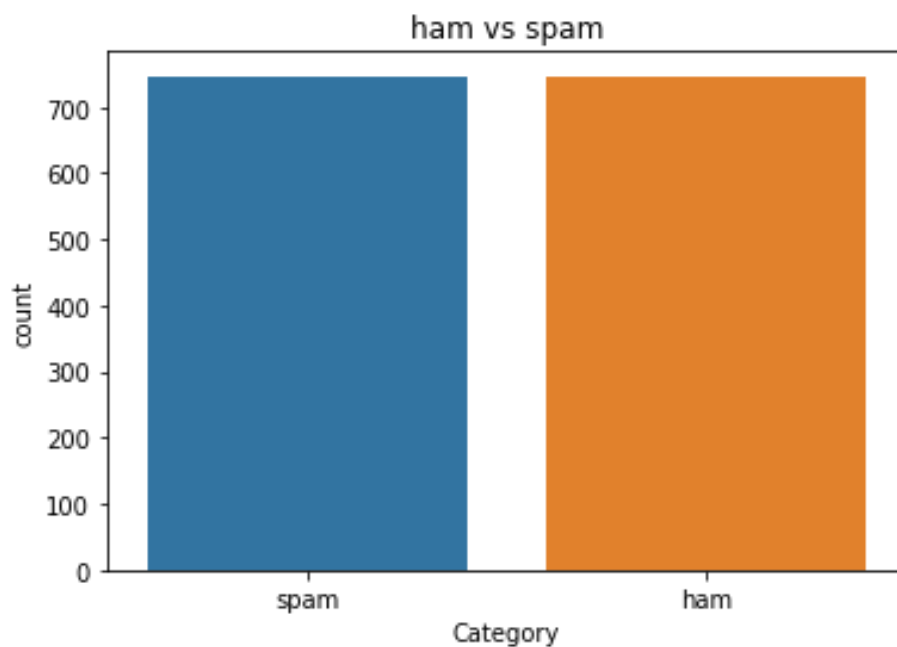
```
df.shape
```

```
(1494, 3)
```

```
df["Category"].value_counts()
```

```
spam    747  
ham     747  
Name: Category, dtype: int64
```

```
sns.countplot(  
    data=df,  
    x="Category"  
)  
plt.title("ham vs spam")  
plt.show()
```



```
df.head()
```



	Category	Message	Message Length
0	spam	UR awarded a City Break and could WIN a å£200 ...	148
1	spam	No. 1 Nokia Tone 4 ur mob every week! Just txt...	143
2	ham	Don't worry, * is easy once have ingredients!	45
3	ham	I just made some payments so dont have that mu...	98
4	spam	Moby Pub Quiz.Win a å£100 High Street prize if...	156

```
df["Label"]=df["Category"].map(  
    {  
        "ham":0,  
        "spam":1  
    }  
)
```

```
df.head()
```

	Category	Message	Message Length	Label
0	spam	UR awarded a City Break and could WIN a å£200 ...	148	
1	spam	No. 1 Nokia Tone 4 ur mob every week! Just txt...	143	
2	ham	Don't worry, * is easy once have ingredients!	45	
3	ham	I just made some payments so dont have that mu...	98	
4	spam	Moby Pub Quiz.Win a å£100 High Street prize if...	156	

```
import re  
import nltk  
from nltk.corpus import stopwords  
from nltk.stem import PorterStemmer  
  
stemmer=PorterStemmer()
```

```
corpus=[]  
for message in df["Message"]:  
    message=re.sub("[^a-zA-Z]", " ",message)  
    message=message.lower()  
    message=message.split()
```



```
from tensorflow.keras.preprocessing.text import one_hot
vocab_size=10000

oneHot_doc=[one_hot(words,n=vocab_size)
            for words in corpus
            ]
```

```
df["Message Length"].describe()
```

```
count    1494.000000
mean      105.866801
std        56.509846
min         3.000000
25%        51.000000
50%       119.000000
75%       153.000000
max       450.000000
Name: Message Length, dtype: float64
```

```
fig=plt.figure(figsize=(12,8))
sns.kdeplot(
    x=df["Message Length"],
    hue=df["Category"]
)
plt.title("ham & spam messege length comparision")
plt.show()
```

ham & spam messege length comparision

0.012

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
sentence_len=200
embedded_doc=pad_sequences(
    oneHot_doc,
    maxlen=sentence_len,
    padding="pre"
)
```

```
extract_features=pd.DataFrame(
    data=embedded_doc
)
target=df["Label"]
```

```
df_final=pd.concat([extract_features,target],axis=1)
```

```
df_final.head()
```

	0	1	2	3	4	5	6	7	8	9	...	191	192	1
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N

5 rows × 201 columns

```
X=df_final.drop("Label",axis=1)
y=df_final["Label"]
```

```
from sklearn.model_selection import train_test_split
```

```
X_trainval,X_test,y_trainval,y_test=train_test_split(
    X,
    y,
    random_state=37,
```

```

        test_size=0.2
    )

X_train,X_val,y_train,y_val=train_test_split(
    X_trainval,
    y_trainval,
    random_state=37,
    test_size=0.2
)

```

```

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

```

Model: "model_2"

Layer (type)	Output Shape	Param #
=====		
inputs (InputLayer)	[(None, 150)]	0
embedding_2 (Embedding)	(None, 150, 50)	500000
lstm_2 (LSTM)	(None, 64)	29440
FC1 (Dense)	(None, 256)	16640
activation_4 (Activation)	(None, 256)	0
dropout_2 (Dropout)	(None, 256)	0
out_layer (Dense)	(None, 1)	257
activation_5 (Activation)	(None, 1)	0
=====		
Total params: 96,337		
Trainable params: 96,337		
Non-trainable params: 0		
=====		

[Colab paid products](#) - [Cancel contracts here](#)

✓ 0s completed at 6:46 PM

