SMS SPAM Classification

```
from google .colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

Import Libraries

```
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
```

```
!pip install tensorflow
```

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

Read the Dataset

```
df=
pd.read_csv('/content/drive/MyDrive/archive.zip',delimiter=',',encodin
g='latin-1')
df.head()
```

```
     v1                                                  v2 Unnamed: 2
\
0   ham  Go until jurong point, crazy.. Available only ...         NaN

1   ham                      Ok lar... Joking wif u oni...         NaN

2  spam  Free entry in 2 a wkly comp to win FA Cup fina...         NaN

3   ham  U dun say so early hor... U c already then say...         NaN

4   ham  Nah I don't think he goes to usf, he lives aro...         NaN
```

```
  Unnamed: 3 Unnamed: 4
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN
```

Pre-processing the Dataset

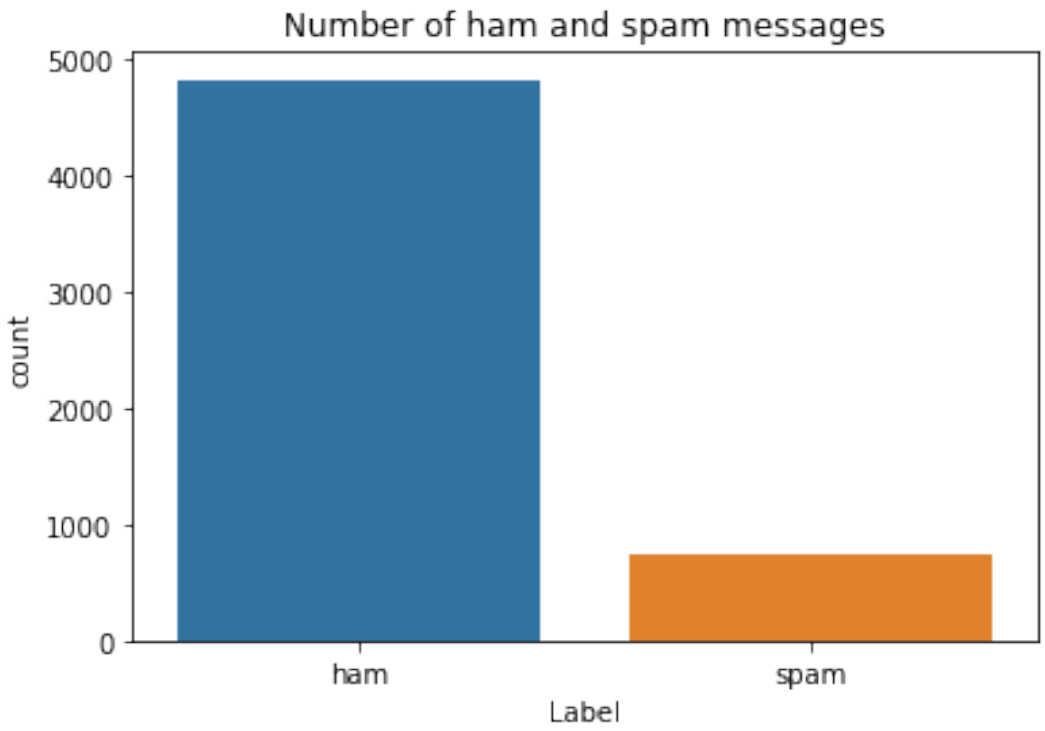
```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis= 1,inplace=
True)
df.info()
```

```
<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: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  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)
```

Create Model & Add 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)
 moddel = Model(inputs=inputs,outputs=layer)
 return model
```

Compile the Model

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

```
Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 150)]             0

 embedding_4 (Embedding)     (None, 150, 50)           50000

 lstm_4 (LSTM)               (None, 64)                29440

 FC1 (Dense)                 (None, 256)               16640

 activation_8 (Activation)   (None, 256)               0

 dropout_4 (Dropout)         (None, 256)               0

 out_layer (Dense)           (None, 1)                 257

 activation_9 (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()
```

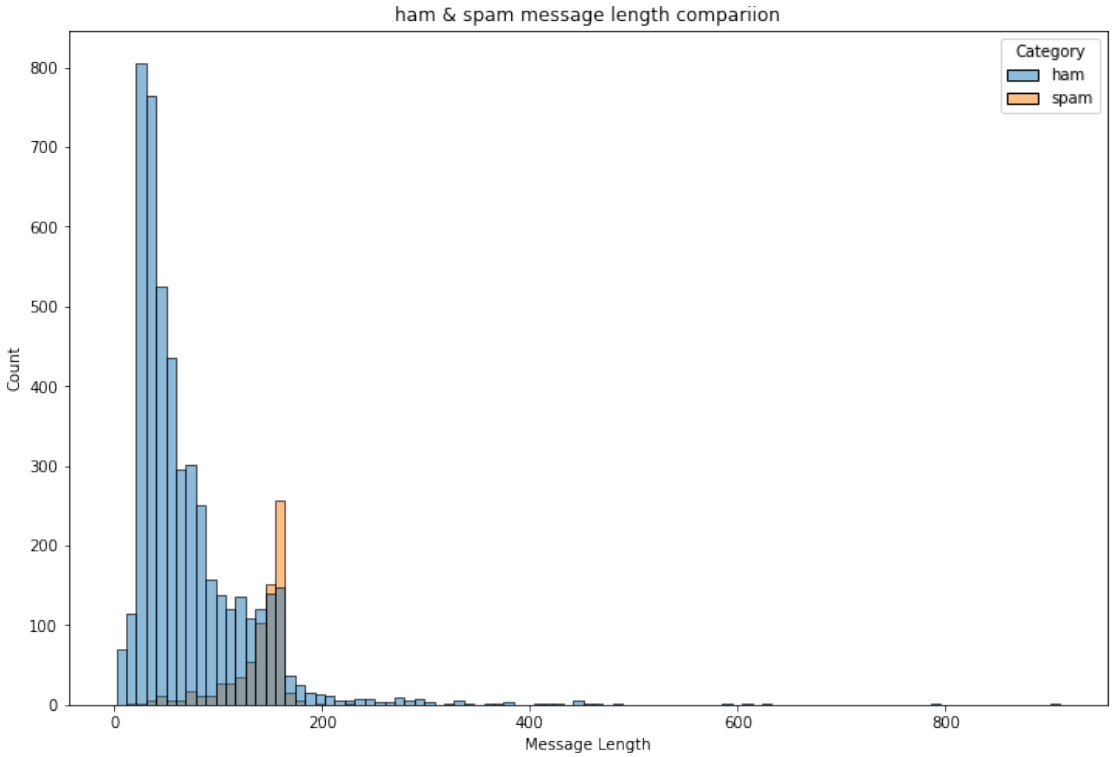
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
```

```
---   ------  --------------  -----
 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 message length compariion")
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```




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

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

```
Ham Message 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
```

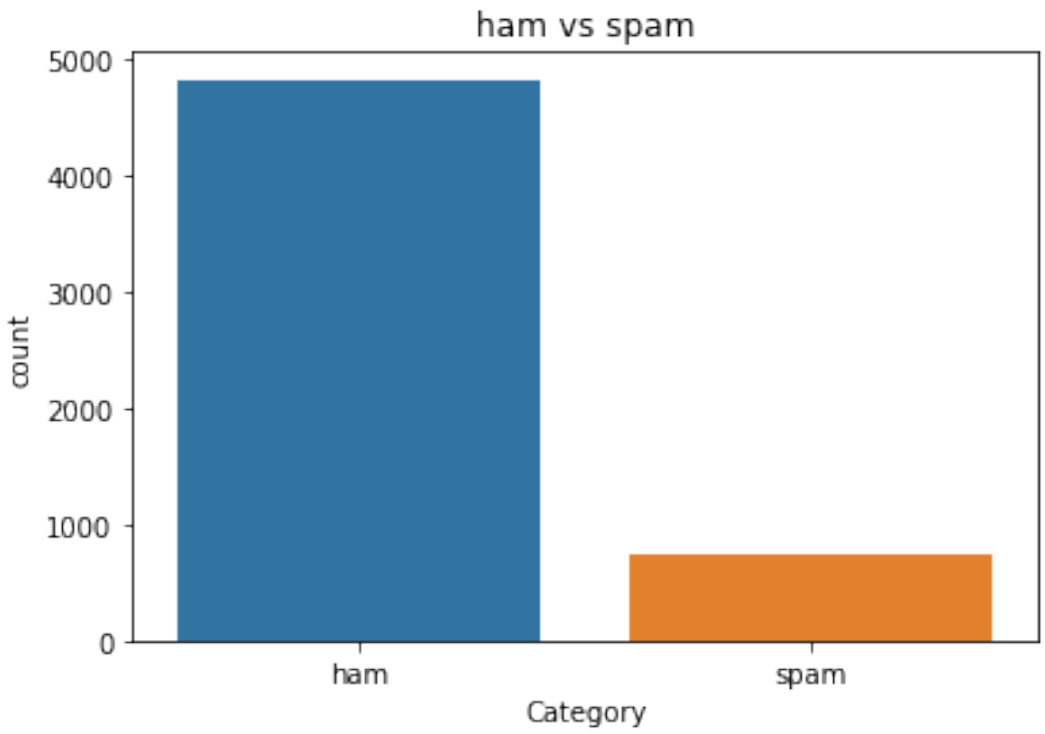
```
data.describe(include="all")
```

	Category	Message	Message Length
count	5572	5572	5572.000000
unique	2	5169	NaN
top	ham	Sorry, I'll call later	NaN
freq	4825	30	NaN
mean	NaN	NaN	80.118808
std	NaN	NaN	59.690841
min	NaN	NaN	2.000000
25%	NaN	NaN	36.000000
50%	NaN	NaN	61.000000
75%	NaN	NaN	121.000000
max	NaN	NaN	910.000000

```
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:86.593683% of total data.
Spam Contains:13.406317% 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(
```

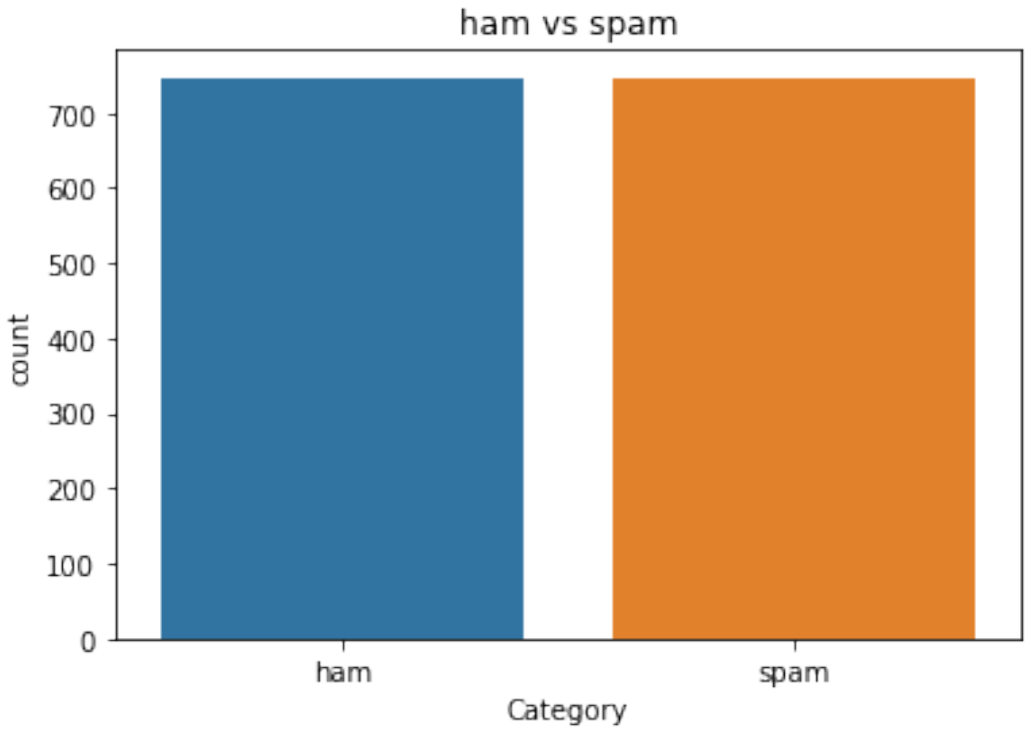
```
    columns=["index"],
)

df.shape
```

```
(1494, 3)
```

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

```
ham     747
spam    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      ham  Sorry completely forgot * will pop em round th...
73
1      ham                                Are you this much buzy
22
2      ham   sure, but make sure he knows we ain't smokin yet
```

```
48
3      ham  S:-)if we have one good partnership going we w...
61
4     spam  REMINDER FROM O2: To get 2.50 pounds free call...
147
```

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

```
df.head()
```

```
  Category                                            Message  Message
Length  \
0      ham  Sorry completely forgot * will pop em round th...
73
1      ham                                Are you this much buzy
22
2      ham   sure, but make sure he knows we ain't smokin yet
48
3      ham  S:-)if we have one good partnership going we w...
61
4     spam  REMINDER FROM O2: To get 2.50 pounds free call...
147

   label
0      0
1      0
2      0
3      0
4      1
```

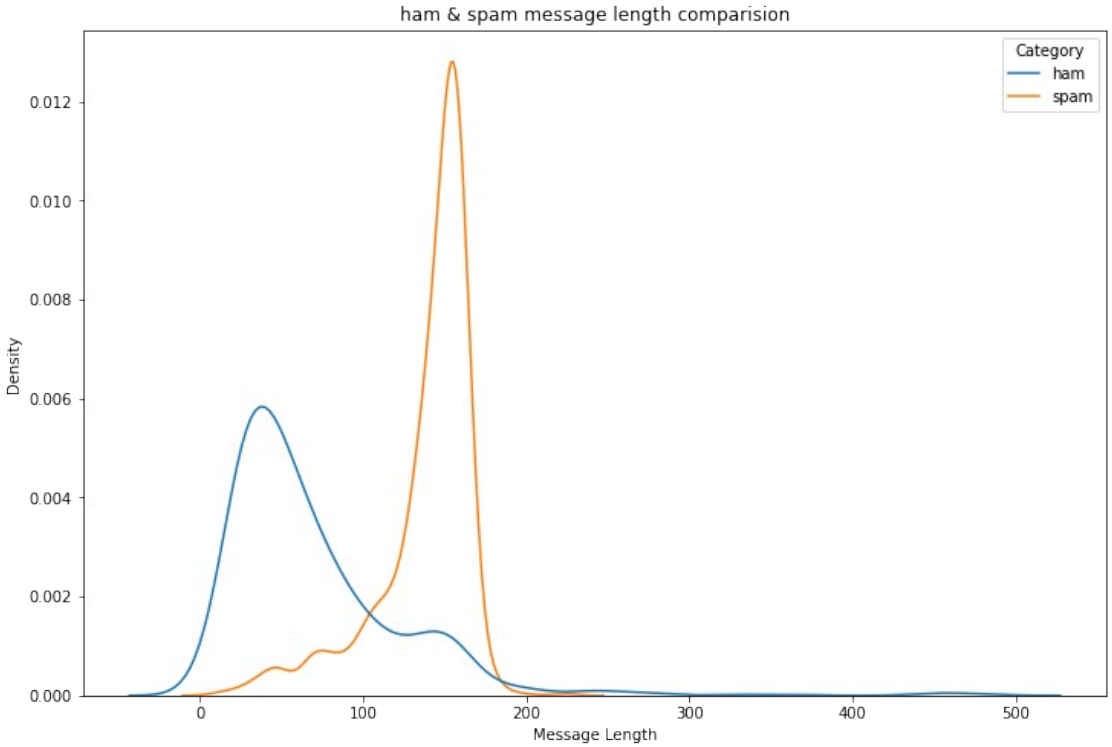
```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

stemmer=PorterStemmer

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

!pip install tensorflow
```

```
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      104.647256
std        56.391151
min         2.000000
25%        51.000000
50%       115.000000
75%       153.000000
max       482.000000
Name: Message Length, dtype: float64
```

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



```
from tensorflow.keras.preprocessing.sequence import pad_sequences
sentence_len=100
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  ...  91  92  93  94  95  96
97  \
0 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN  ... NaN NaN NaN NaN NaN NaN
NaN
1 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN  ... NaN NaN NaN NaN NaN NaN
NaN
2 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN  ... NaN NaN NaN NaN NaN NaN
NaN
3 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN  ... NaN NaN NaN NaN NaN NaN
NaN
4 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN  ... NaN NaN NaN NaN NaN NaN
NaN

   98  99  label
0 NaN NaN      0
1 NaN NaN      0
2 NaN NaN      0
3 NaN NaN      0
4 NaN NaN      1

[5 rows x 101 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_trainval,x_val,y_trainval,y_test=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_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 150)]             0

 embedding_4 (Embedding)     (None, 150, 50)           50000

 lstm_4 (LSTM)               (None, 64)                29440

 FC1 (Dense)                 (None, 256)               16640

 activation_8 (Activation)   (None, 256)               0

 dropout_4 (Dropout)         (None, 256)               0

 out_layer (Dense)           (None, 1)                 257

 activation_9 (Activation)   (None, 1)                 0

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