```
#import the library
import numpy as np
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
```

```
#load the dataset
df=pd.read_csv(r"spam.csv",encoding='Windows-1252')
```

```
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.describe()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
count	5572	5572	50	12	6
unique	2	5169	43	10	5
top	ham	Sorry, I'll call later	bt not his girlfrnd... G o o d n i g h t . . .@"	MK17 92H. 450Ppw 16"	GNT:-)"

```
ps=PorterStemmer()
```

```
nltk.download('stopwords')
```

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

```
data=[]
for i in range(0,5572):
    message=df["v2"][i]
    message=message.lower()
    message=re.sub('[^a-z]',' ',message)
    message=message.split()
    message=[ps.stem(word) for word in message if not word in set(stopwords.words("english
    message=' '.join(message)
    data.append(message)
```

```
data
```

```
cv=CountVectorizer(max_features=7000)
x=cv.fit_transform(data).toarray()
x.shape
```

```
     (5572, 6221)
```

```
df["v1"].loc[df["v1"]=="spam"]=0.0
df["v1"].loc[df["v1"]=="ham"]=1.0
df["v1"]
```

```
     0       1.0
     1       1.0
     2       0.0
     3       1.0
     4       1.0
            ...
     5567    0.0
     5568    1.0
     5569    1.0
     5570    1.0
     5571    1.0
     Name: v1, Length: 5572, dtype: object
```

```
y=df.iloc[:,0:1].values
y=np.asarray(y).astype("float64")
```

```
y
```

```
     array([[1.],
            [1.],
            [0.],
            ...,
            [1.],
            [1.],
            [1.]])
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

```
model=Sequential()
```

```
#input layer
```

```
model.add(Dense(units=5572,activation='relu',kernel_initializer='random_uniform'))
#hidden layer
model.add(Dense(units=6000,activation='relu',kernel_initializer='random_uniform'))
model.add(Dense(units=6000,activation='relu',kernel_initializer='random_uniform'))
model.add(Dense(units=6000,activation='relu',kernel_initializer='random_uniform'))
model.add(Dense(units=6000,activation='relu',kernel_initializer='random_uniform'))
model.add(Dense(units=6000,activation='relu',kernel_initializer='random_uniform'))
#output layer
model.add(Dense(units=1,activation='sigmoid',kernel_initializer='random_uniform'))
```

```
#Compile the Model
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
# Fit the Model
tr=model.fit(x_train,y_train,epochs=10,batch_size=32)
```

```
Epoch 1/10
140/140 [==============================] - 232s 2s/step - loss: 0.7672 - accuracy: 0
Epoch 2/10
140/140 [==============================] - 241s 2s/step - loss: 0.0094 - accuracy: 0
Epoch 3/10
140/140 [==============================] - 232s 2s/step - loss: 0.0193 - accuracy: 0
Epoch 4/10
140/140 [==============================] - 235s 2s/step - loss: 0.8794 - accuracy: 0
Epoch 5/10
140/140 [==============================] - 233s 2s/step - loss: 0.0258 - accuracy: 0
Epoch 6/10
140/140 [==============================] - 233s 2s/step - loss: 0.2907 - accuracy: 0
Epoch 7/10
140/140 [==============================] - 239s 2s/step - loss: 0.1515 - accuracy: 0
Epoch 8/10
140/140 [==============================] - 236s 2s/step - loss: 0.0457 - accuracy: 0
Epoch 9/10
140/140 [==============================] - 234s 2s/step - loss: 0.0118 - accuracy: 0
Epoch 10/10
140/140 [==============================] - 235s 2s/step - loss: 0.0074 - accuracy: 0
```

```
# Save The Model
model.save("sms.h5")
```

```
# Test The Model
ypred=model.predict(x_test)
ypred
```

```
35/35 [==============================] - 16s 361ms/step
array([[1.],
       [1.],
       [1.],
       ...,
       [1.],
       [1.],
       [1.]], dtype=float32)
```

```
y_test
```

```
array([[1.],
       [1.],
       [1.],
       ...,
       [1.],
       [1.],
       [1.]])
```

```
text=model.predict(cv.transform(["Wishing you a very happy Birthday to you ! "]))
text>0.5
```

```
1/1 [==============================] - 0s 250ms/step
array([[ True]])
```

```
class_name=["ham","spam"]
pred_id=text.argmax(axis=1)[0]
pred_id
print(str(class_name[pred_id]))
```

```
ham
```

✓ 0s completed at 12:10 PM ● ✕