In [1]:

```
import pandas as pd
import numpy as np
import nltk
import csv
import re

nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
STOPWORDS = set(stopwords.words('english'))
```

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

In [2]:

```
import os
import seaborn as sns
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud
from keras import utils
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, Embedding
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.text import Tokenizer
import keras
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
```

In [3]:

```
data = pd.read_csv('/content/spam.csv', encoding='latin-1')
data.head()
```

Out[3]:

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

In [4]:

```
data['v2'][4]
```

Out[4]:

```
"Nah I don't think he goes to usf, he lives around here though"
```

In [5]:

```
data.columns
```

Out[5]:

```
Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')
```

In [6]:

In [6]:

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

Out[6]:

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

In [7]:

```
data.shape
```

Out[7]:

```
(5572, 2)
```

In [8]:

```
data.describe()
```

Out[8]:

	v1	v2
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

In [9]:

```
data.isna().sum()
```

Out[9]:

```
v1    0
v2    0
dtype: int64
```

In [10]:

```
data.duplicated().sum()
```

Out[10]:

```
403
```

In [11]:

```
data=data.drop_duplicates()
```

In [12]:

```
data.duplicated().sum()
```

Out[12]:

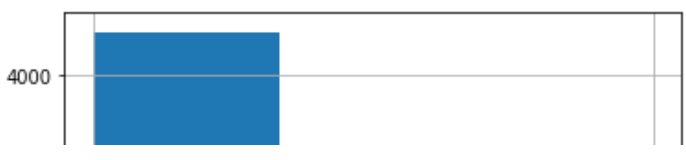
```
0
```

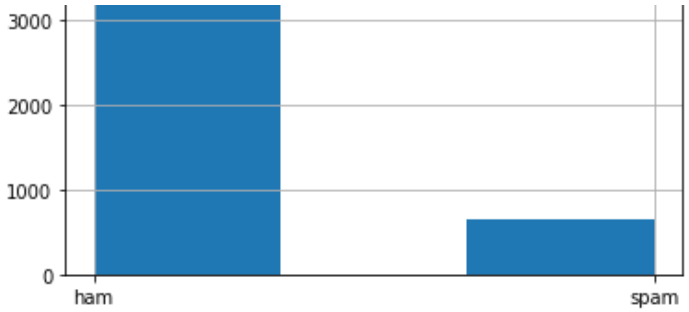
In [13]:

```
data['v1'].hist(bins=3)
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff43a53e2d0>
```





In [14]:

```
data['alpha_text'] = data['v2'].apply(lambda x: re.sub(r'[^a-zA-Z ]+', '', x.lower()))
data.head()
```

Out[14]:

	v1	v2	alpha_text
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...
3	ham	U dun say so early hor... U c already then say...	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...	nah i dont think he goes to usf he lives aroun...

In [15]:

```
nltk.download('stopwords')
data['imp_text'] = data['alpha_text'].apply(lambda x : ' '.join([word for word in x.spli
t() if not word in set(stopwords.words('english'))]))
data.head()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[15]:

	v1	v2	alpha_text	imp_text
0	ham	Go until jurong point, crazy.. Available only ...	go until jurong point crazy available only in ...	go jurong point crazy available bugis n great ...
1	ham	Ok lar... Joking wif u oni...	ok lar joking wif u oni	ok lar joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in a wkly comp to win fa cup final...	free entry wkly comp win fa cup final tkts st ...
3	ham	U dun say so early hor... U c already then say...	u dun say so early hor u c already then say	u dun say early hor u c already say
4	ham	Nah I don't think he goes to usf, he lives aro...	nah i dont think he goes to usf he lives aroun...	nah dont think goes usf lives around though

In [16]:

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer()
```

In [17]:

```
x = cv.fit_transform(data).toarray()
x
```

Out[17]:

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

```
       [1, 0, 0, 0],
       [0, 1, 0, 0]])
```

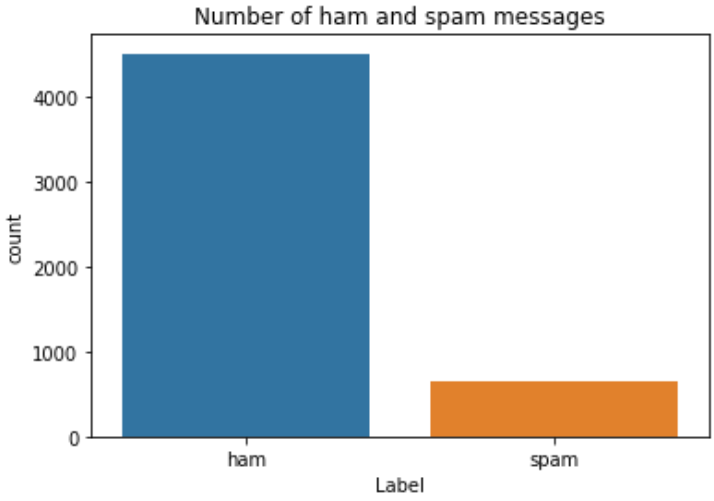
In [18]:

```
sns.countplot(data.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 argu
ment will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  FutureWarning
```

Out[18]:

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



In [19]:

```
X = data.v2
Y = data.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

In [20]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.15, random_state=4
2, stratify=Y)
```

In [21]:

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

In [22]:

```
sequences_matrix.shape
```

Out[22]:

```
(4393, 150)
```

In [23]:

```
sequences_matrix.ndim
```

Out[23]:

2

In [24]:

```
sequences_matrix = np.reshape(sequences_matrix,(4393,150,1))
```

In [25]:

```
sequences_matrix.ndim
```

Out[25]:

3

In [26]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
```

In [27]:

```
model = Sequential()
model.add(Embedding(max_words,50,input_length=max_len))
```

In [28]:

```
model.add(LSTM(units=64,input_shape = (sequences_matrix.shape[1],1),return_sequences=Tru
e))
model.add(LSTM(units=64,return_sequences=True))
model.add(LSTM(units=64,return_sequences=True))
model.add(LSTM(units=64))
model.add(Dense(units = 256,activation = 'relu'))
model.add(Dense(units = 1,activation = 'sigmoid'))
```

In [29]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 150, 50)           50000

 lstm (LSTM)                 (None, 150, 64)           29440

 lstm_1 (LSTM)               (None, 150, 64)           33024

 lstm_2 (LSTM)               (None, 150, 64)           33024

 lstm_3 (LSTM)               (None, 64)                33024

 dense (Dense)               (None, 256)               16640

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 195,409
Trainable params: 195,409
Non-trainable params: 0
_________________________________________________________________
```

In [30]:

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

In [31]:

```
M = model.fit(sequences_matrix,Y_train,batch_size=128,epochs=5,validation_split=0.2)
```

```
Epoch 1/5
28/28 [==============================] - 38s 1s/step - loss: 0.4562 - accuracy: 0.8480 -
val_loss: 0.3593 - val_accuracy: 0.8760
Epoch 2/5
28/28 [==============================] - 27s 963ms/step - loss: 0.2458 - accuracy: 0.9129
- val_loss: 0.1077 - val_accuracy: 0.9693
Epoch 3/5
28/28 [==============================] - 27s 965ms/step - loss: 0.0729 - accuracy: 0.9806
- val_loss: 0.0749 - val_accuracy: 0.9784
Epoch 4/5
28/28 [==============================] - 27s 965ms/step - loss: 0.0487 - accuracy: 0.9869
- val_loss: 0.0732 - val_accuracy: 0.9750
Epoch 5/5
28/28 [==============================] - 27s 964ms/step - loss: 0.0326 - accuracy: 0.9915
- val_loss: 0.0808 - val_accuracy: 0.9761
```

In [32]:

```
model.save('spam-classifier.h5')
```

In [33]:

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = utils.pad_sequences(test_sequences,maxlen=max_len)
```

In [34]:

```
accr = model.evaluate(test_sequences_matrix,Y_test)
```

```
25/25 [==============================] - 4s 77ms/step - loss: 0.1105 - accuracy: 0.9768
```

In [35]:

```
print("Accuracy of the model on Testing Data is - " , accr[1]*100 , "%")
```

```
Accuracy of the model on Testing Data is -  97.68041372299194 %
```

In [36]:

```
l = accr[0]
a =accr[1]
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(l,a))
```

```
Test set
  Loss: 0.111
  Accuracy: 0.977
```