

In [55]:

```
import string
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re
```

In [56]:

```
df = pd.read_csv('spam_ham_dataset.csv')
```

In [57]:

```
df.head()
```

Out[57]:

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

In [58]:

```
df = df.drop(['Unnamed: 0'], axis=1)
df.head()
```

Out[58]:

	label	text	label_num
0	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	spam	Subject: photoshop , windows , office . cheap ...	1
4	ham	Subject: re : indian springs\r\nthis deal is t...	0

In [59]:

```
print('Total %s data email'% len(df))
```

Total 5171 data email

In [60]:

```
#total class memebbers
df['label'].value_counts()
```

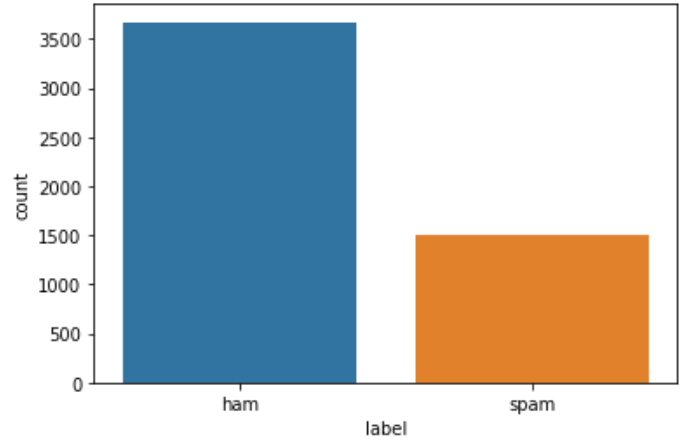
Out[60]:

```
ham      3672
spam     1499
Name: label, dtype: int64
```

In [61]:

```
#show graph
df_label = sns.countplot(df['label'])
df_label.set_xticklabels(df['label'].unique())
plt.show()
```

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.



In [62]:

```
#data preprocessing
#data text cleaning
# punctuations
punct = []
for char in string.punctuation:
    punct.append(char)
```

In [63]:

```
def cleaning(txt):
    # case folding
    text = txt.lower()

    # remove multiple space, tabs, dan newlines
    text = re.sub('\s+', ' ',text)
    # remove links
    text = text.replace("http://", " ").replace("https://", " ")
    # remove special characters
    text = text.encode('ascii', 'replace').decode('ascii')
    text = ' '.join(re.sub("([@#][A-Za-z0-9+])|(\w+:\/\/\S+)", " ", text).split())
    # remove punctuation
    text = ' '.join([word for word in text if word not in punct])
    #remove single character
    text = re.sub(r"\b[a-zA-Z]\b", "", text)
    #remove numbers
    text = re.sub(r"\d+", "", text)
    #remove multiple spaces (again)
    text = re.sub('\s+', ' ',text)
    return text
```

In [64]:

```
# call function for cleaning
# apply fungsi cleaning ke setiap text
df['text_cleaned'] = df['text'].apply(lambda x: cleaning(x))
df = df[['text', 'text_cleaned', 'label']]
df.head()
```

Out [64]:

	text	text_cleaned	label
0	Subject: enron methanol ; meter # : 988291\r\n...	subject enron methanol meter this is follow up...	ham
1	Subject: hpl nom for january 9 , 2001\r\n(see...	subject hpl nom for january see attached file ...	ham
2	Subject: neon retreat\r\nho ho ho , we ' re ar...	subject neon retreat ho ho ho we re around to ...	ham
3	Subject: photoshop , windows , office . cheap ...	subject photoshop windows office cheap main tr...	spam
4	Subject: re : indian springs\r\nthis deal is t...	subject re indian springs this deal is to book...	ham

In [65]:

```
#compare
print(df['text'][0])
print(df['text_cleaned'][0])
```

Subject: enron methanol ; meter # : 988291
this is a follow up to the note i gave you on monday , 4 / 3 / 00 { preliminary
flow data provided by daren } .
please override pop ' s daily volume { presently zero } to reflect daily
activity you can obtain from gas control .
this change is needed asap for economics purposes .
subject enron methanol meter this is follow up to the note gave you on monday preliminary flow da
ta provided by daren please override pop daily volume presently zero to reflect daily activity yo
u can obtain from gas control this change is needed asap for economics purposes

In [66]:

```
# to remove stop words
import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
from nltk.corpus import stopwords
stop = stopwords.words('english')
df['text_cleaned'] = df['text_cleaned'].apply(lambda x: ' '.join([word for word in x.split() if wo
rd not in stop]))
```

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

In [67]:

```
#lemmatization
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
lemmatizer = WordNetLemmatizer()
def get_wordnet_pos(word):
    """Map POS tag to first character lemmatize() accepts"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.NOUN)

def do_lemma(string):
    lemmatized = ' '.join([lemmatizer.lemmatize(word, get_wordnet_pos(word)) for word in nltk.word
_tokenize(string)])
    return lemmatized
```

In [68]:

```
import nltk
nltk.download('omw-1.4')
df['text_cleaned'] = df['text_cleaned'].apply(lambda x: do_lemma(x))
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Package omw-1.4 is already up-to-date!
```

In [69]:

```
df = df.drop(['text'], axis=1)
df = df.rename(columns = {'text_cleaned' : 'text'})
df.columns
```

Out[69]:

```
Index(['text', 'label'], dtype='object')
```

In [70]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
X = tfidf.fit_transform(df['text'])
y = df['label']
```

In [71]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [72]:

```
from time import time
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
# defining the classifier
clf = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
```

In [73]:

```
#predicting the time of train and testing
t0 = time()
clf.fit(X_train, y_train)
print("\nTraining time:", round(time()-t0, 3), "s\n")
```

Training time: 0.01 s

In [74]:

```
#predicting the time of testing
t1 = time()
pred = clf.predict(X_test)
print("Predicting time:", round(time()-t1, 3), "s\n")
```

Predicting time: 0.305 s

In [75]:

```
#calculating and printing the accuracy of the algorithm
print("Accuracy of KNN Algorithm: ", accuracy_score(pred,y_test))
```

Accuracy of KNN Algorithm: 0.9623188405797102