In [28]:
```
from flask import Flask
from flask import Flask, flash, redirect, render_template, request, ses
sion, abort
import os,easyimap,json
import pandas as pd
from sklearn import preprocessing
import nltk
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics
```

In [29]:
```
# download stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\DoubleW\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[29]:
```
True
```

In [30]:
```
"""
TODO: Inspect dataset
"""

# read dataset
dataset = pd.read_csv('dataset/emails.csv')

# remove duplicates
dataset.drop_duplicates(inplace = True)

# get label spam
y = dataset['spam']

# Encode label
```

```
le = preprocessing.LabelEncoder()
y_enc = le.fit_transform(y)

#print(dataset.shape) # (5695,2)
#print(dataset.groupby('spam').count())  # 1:spam , 0: not spam
#print(dataset.head(5))
```

In [31]:
```
"""
TODO: Preprocessing
"""
# list of word has no meaningful
stop_words = nltk.corpus.stopwords.words('english')
#Stemming ( eg : distribute , distribution ,distributing , distributor
 ,...) can replace with distribute
porter = nltk.PorterStemmer()
#print(stop_words)
```

In [32]:
```
# every mail start with 'Subject' so remove it
processed=dataset['text'].map(lambda text: text[9:])
```

In [33]:
```
# normalize : replace format email , http , phone number , number with
 general form
processed = processed.str.replace(r'\b[\w\-.]+?@\w+?\.\w{2,4}\b','email
addr')
processed = processed.str.replace(r'(http[s]?\S+)|(\w+\.[A-Za-z]{2,4}\S
*)','httpaddr')
processed = processed.str.replace(r'£|\$', 'moneysymb')
processed = processed.str.replace(r'\b(\+\d{1,2}\s)?\d?[\-(.]?\d{3}\)?
[\s.-]?\d{3}[\s.-]?\d{4}\b','phonenumbr')
processed = processed.str.replace(r'\d+(\.\d+)?', 'numbr')

# today with todays : the same , collapse all white space ( spaces , li
ne breaks ,tabs ) into a single space
processed = processed.str.replace(r'[^\w\d\s]', ' ')
processed = processed.str.replace(r'\s+', ' ')
processed = processed.str.replace(r'^\s+|\s+?$', '')

processed = processed.str.lower()  # to lower case
```

```
In [34]: # filter stop-words
processed = processed.apply(lambda x: ' '.join(
    term for term in x.split() if term not in set(stop_words))
)
```

```
In [8]: # filter stemming
processed = processed.apply(lambda x: ' '.join(
    porter.stem(term) for term in x.split())
)
```

```
In [44]: # feature engineering
vectorizer = TfidfVectorizer(ngram_range=(1, 2))
X_ngrams = vectorizer.fit_transform(processed)
X_train, X_test, y_train, y_test = train_test_split(
    X_ngrams,
    y_enc,
    test_size=0.3,
    random_state=42,
    stratify=y_enc
)

clf = svm.LinearSVC(loss='hinge')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

```
In [45]: # Normalization the email
def preprocess_text(messy_string):
    #assert(type(messy_string) == str)
    cleaned = re.sub(r'\b[\w\-.]+?@\w+?\.\w{2,4}\b', 'emailaddr', messy
_string)    # replace email addr with 'emailaddr'
    cleaned = re.sub(r'(http[s]?\S+)|(\w+\.[A-Za-z]{2,4}\S*)', 'httpadd
r',           # replace http link with 'httpaddr'
                     cleaned)
    cleaned = re.sub(r'£|\$', 'moneysymb', cleaned)
                      # replace money symbol with moneysymb
    cleaned = re.sub(
        r'\b(\+\d{1,2}\s)?\d?[\-(.]?\d{3}\)?[\s.-]?\d{3}[\s.-]?\d{4}\b'
,              # replace phone number
        'phonenumbr', cleaned)

    cleaned = re.sub(r'\d+(\.\d+)?', 'numbr', cleaned)
                # replace number

    # today with todays : the same , collapse all white space ( spaces
 , line breaks ,tabs ) into a single space.... and lower case it
    cleaned = re.sub(r'[^\w\d\s]', ' ', cleaned)
    cleaned = re.sub(r'\s+', ' ', cleaned)
    cleaned = re.sub(r'^\s+|\s+?$', '', cleaned.lower())



    return ' '.join(
        porter.stem(term)
        for term in cleaned.split()
        if term not in set(stop_words)
    )
```

In [46]:
```
def spam_filter(message):
    if clf.predict(vectorizer.transform([preprocess_text(message)])):
        return 'spam'
    else:
        return 'not spam'
```

In [47]:
```
pd.DataFrame(
    metrics.confusion_matrix(y_test, y_pred),
    index=[['actual', 'actual'], ['spam', 'ham']],
    columns=[['predicted', 'predicted'], ['spam', 'ham']]
)
```

Out[47]:

		predicted	
		spam	**ham**
actual	**spam**	1296	2
	ham	10	401

In [49]:
```
#this function computes subset accuracy
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
print(accuracy_score(y_test, y_pred,normalize=False)) #1697 / 1709
```

```
0.9929783499122293
1697
```

In [50]:
```
# Applying k-Fold Cross Validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = clf, X = X_train, y = y_train,
 cv = 10)
print(accuracies)
print(accuracies.mean())
print(accuracies.std()) # most value within the range 0.04 from mean va
lue
```

```
[0.97744361 0.98245614 0.98997494 0.9924812  0.98496241 0.98746867
 0.98241206 0.98743719 0.98994975 0.98994975]
0.9864535711113714
0.00436913106075129
```

In [ ]:
```
```