In [1]:
```
import numpy as np
import pandas as pd
import nltk
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn import metrics
from nltk import word_tokenize, FreqDist
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re
from nltk.stem.porter import PorterStemmer
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
```

In [2]:
```
df = pd.read_csv(r"C:\Users\adity\Downloads\archive (1)\spam.csv", encoding = 'latin-1
df.drop(columns = ['Unnamed: 2','Unnamed: 3', 'Unnamed: 4'], inplace = True)
df.rename(columns = {'v1': 'Target', 'v2': 'Message'}, inplace = True)
df.head()
```

Out[2]:

	Target	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	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...

In [3]:
```
df['Target'] = df['Target'].apply(lambda x: 0 if x == 'ham'else 1)
df['Target'].value_counts()
```

Out[3]:
```
0    4825
1     747
Name: Target, dtype: int64
```

In [4]:
```
ham_df = df[df['Target']==0]
spam_df = df[df['Target']==1]
ham_downsampled = ham_df.sample(spam_df.shape[0])
df_balanced = pd.concat([ham_downsampled, spam_df])
df_balanced.reset_index(drop = True, inplace = True)
df_balanced.isna().sum()
```

Out[4]:
```
Target     0
Message    0
dtype: int64
```

In [5]:
```
print(df_balanced['Target'].value_counts())
df_balanced.head()
```

```
0    747
1    747
Name: Target, dtype: int64
```

Out[5]:

	Target	Message
0	0	A bloo bloo bloo I'll miss the first bowl
1	0	Havent shopping now lor i juz arrive only
2	0	So u wan 2 come for our dinner tonight a not?
3	0	I'm eatin now lor, but goin back to work soon....
4	0	Still i have not checked it da. . .

In [6]:

```
txt = []
corpus = list(df_balanced['Message'])
for i in range(len(corpus)):
    r = re.sub('[^a-zA-Z]', ' ', corpus[i])
    r = r.lower()
    r = r.split()
    r = ' '.join(r)
    txt.append(r)
df_balanced['Message'] = txt
print(df_balanced['Message'][:3])
```

```
0      a bloo bloo bloo i ll miss the first bowl
1      havent shopping now lor i juz arrive only
2     so u wan come for our dinner tonight a not
Name: Message, dtype: object
```

In [7]:

```
df_balanced['Message'] = df_balanced.apply(lambda row: word_tokenize(row['Message']),
print(df_balanced['Message'][:3])
```

```
0    [a, bloo, bloo, bloo, i, ll, miss, the, first,...
1    [havent, shopping, now, lor, i, juz, arrive, o...
2    [so, u, wan, come, for, our, dinner, tonight, ...
Name: Message, dtype: object
```

In [8]:

```
stop_words = set(stopwords.words('english'))
stop_list = []
for i in range(len(df_balanced)):
        msg = df_balanced['Message'][i]
        msg = [word for word in msg if word not in stop_words]
        stop_list.append(msg)
df_balanced['Message'] = stop_list
print(df_balanced['Message'][:3])
```

```
0    [bloo, bloo, bloo, miss, first, bowl]
1     [havent, shopping, lor, juz, arrive]
2          [u, wan, come, dinner, tonight]
Name: Message, dtype: object
```

In [9]:

```
ps = PorterStemmer()
stem_list = []
for i in range(len(df_balanced)):
    txt = df_balanced['Message'][i]
    txt = [ps.stem(word) for word in txt]
    stem_list.append(txt)
df_balanced['Message'] = stem_list
print(df_balanced['Message'][:3])
```

```
0    [bloo, bloo, bloo, miss, first, bowl]
1          [havent, shop, lor, juz, arriv]
2          [u, wan, come, dinner, tonight]
Name: Message, dtype: object
```

In [10]:
```
corpus = []
for i in df_balanced['Message']:
    msg = ' '.join(row for row in i)
    corpus.append(msg)
df_balanced['Message'] = corpus
print(df_balanced['Message'][:3])
```

```
0    bloo bloo bloo miss first bowl
1         havent shop lor juz arriv
2         u wan come dinner tonight
Name: Message, dtype: object
```

In [11]:
```
x_train, x_test, y_train, y_test = train_test_split(df_balanced['Message'], df_balance
```

In [12]:
```
tv = TfidfVectorizer()
x_train_tv = tv.fit_transform(x_train)
x_test_tv = tv.transform(x_test)
```

In [13]:
```
nb_model = MultinomialNB()
nb_model.fit(x_train_tv, y_train)
nb_predict = nb_model.predict(x_test_tv)
print('Precision', ' ', round(metrics.precision_score(y_test, nb_predict), 2))
print('Accuracy', ' ', round(metrics.accuracy_score(y_test, nb_predict), 2))
```

```
Precision   0.95
Accuracy   0.96
```

In [14]:
```
cv_score = cross_val_score(nb_model, x_train_tv, y_train, scoring='accuracy', cv=10)
print('Cross Validated Accuracy:', round(cv_score.mean(),2))
```

```
Cross Validated Accuracy: 0.95
```

In [15]:
```
lr_model = LogisticRegression()
lr_model.fit(x_train_tv, y_train)
lr_predict = lr_model.predict(x_test_tv)
print('Precision', ' ', round(metrics.precision_score(y_test, lr_predict), 2))
print('Accuracy', ' ', round(metrics.accuracy_score(y_test, lr_predict), 2))
```

```
Precision   0.96
Accuracy   0.94
```

In [16]:
```
cv_score = cross_val_score(lr_model, x_train_tv, y_train, scoring='accuracy', cv=10)
print('Cross Validated Accuracy:', round(cv_score.mean(),2))
```

```
Cross Validated Accuracy: 0.95
```

In [17]:
```
rf_model = RandomForestClassifier()
rf_model.fit(x_train_tv, y_train)
rf_predict = rf_model.predict(x_test_tv)
print('Precision', ' ', round(metrics.precision_score(y_test, rf_predict), 2))
print('Accuracy', ' ', round(metrics.accuracy_score(y_test, rf_predict), 2))
```

```
Precision   0.99
Accuracy   0.95
```

```
In [18]: cv_score = cross_val_score(rf_model, x_train_tv, y_train, scoring='accuracy', cv=10)
         print('Cross Validated Accuracy:', round(cv_score.mean(),2))
```

```
Cross Validated Accuracy: 0.95
```

```
In [19]: svm_model = svm.SVC()
         svm_model.fit(x_train_tv, y_train)
         svm_predict = svm_model.predict(x_test_tv)
         print('Precision', ' ', round(metrics.precision_score(y_test, svm_predict), 2))
         print('Accuracy', ' ', round(metrics.accuracy_score(y_test, svm_predict), 2))
```

```
Precision   0.97
Accuracy   0.94
```

```
In [20]: cv_score = cross_val_score(svm_model, x_train_tv, y_train, scoring='accuracy', cv=10)
         print('Cross Validated Accuracy:', round(cv_score.mean(),2))
```

```
Cross Validated Accuracy: 0.95
```

```
In [21]: k_model = KNeighborsClassifier()
         k_model.fit(x_train_tv,y_train)
         k_predict = k_model.predict(x_test_tv)
         print('Precision', ' ', round(metrics.precision_score(y_test, k_predict), 2))
         print('Accuracy', ' ', round(metrics.accuracy_score(y_test, k_predict), 2))
```

```
Precision   0.97
Accuracy   0.91
```

```
In [22]: cv_score = cross_val_score(k_model, x_train_tv, y_train, scoring='accuracy', cv=10)
         print('Cross Validated Accuracy:', round(cv_score.mean(),2))
```

```
Cross Validated Accuracy: 0.91
```

```
In [ ]:
```