In [1]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from imblearn.over_sampling import RandomOverSampler
#from sklearn.cross_validation import cross_val_score
#!pip3 install imbalanced-learn
```

```
Using TensorFlow backend.
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
df_messages = pd.read_csv('spam.csv', encoding='latin-1', \
                          sep=',', names=['labels','message'])
```

In [4]:
```
df_messages.head(3)
```

Out[4]:

	labels	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...

In [5]:
```
df_messages.describe()
```

Out[5]:

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

In [6]:
```
df_messages.groupby('labels').describe()
```

Out[6]:

	message			
	count	**unique**	**top**	**freq**
labels				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

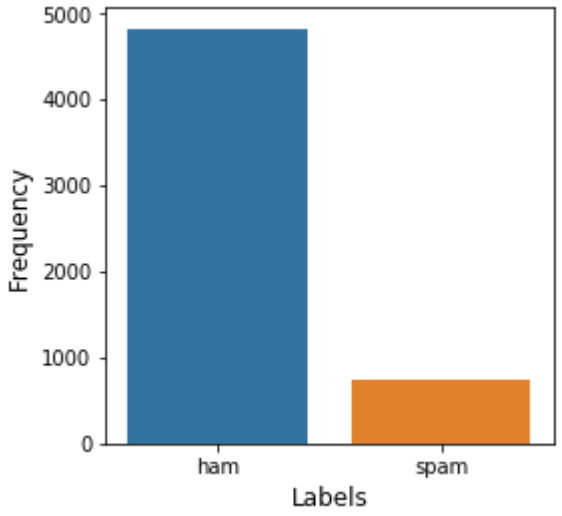
Number of words & number of characters

In [7]:
```
df_messages['word_count'] = df_messages['message'].apply(lambda x:
len(str(x).split(" ")))
df_messages['character_count'] = df_messages['message'].str.len()

df_messages[['message','word_count', 'character_count']].head()
```

Out[7]:

	message	word_count	character_count
0	Go until jurong point, crazy.. Available only ...	20	111
1	Ok lar... Joking wif u oni...	6	29
2	Free entry in 2 a wkly comp to win FA Cup fina...	28	155
3	U dun say so early hor... U c already then say...	11	49
4	Nah I don't think he goes to usf, he lives aro...	13	61

In [8]:
```
labels_count = pd.DataFrame(df_messages.groupby('labels')['message'].count())
labels_count.reset_index(inplace = True)
plt.figure(figsize=(4,4))
sns.barplot(labels_count['labels'], labels_count['message'])
plt.ylabel('Frequency', fontsize=12)
plt.xlabel('Labels', fontsize=12)
plt.show()
```



In [9]:
```
class_labels = {"ham":0,"spam":1}
df_messages['labels']=df_messages['labels'].map(class_labels)
df_messages.head()
```

Out[9]:

	labels	message	word_count	character_count
0	0	Go until jurong point, crazy.. Available only ...	20	111
1	0	Ok lar... Joking wif u oni...	6	29
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	28	155
3	0	U dun say so early hor... U c already then say...	11	49
4	0	Nah I don't think he goes to usf, he lives aro...	13	61

CountVectorizer

In [10]:
```
# Split your data into train & test set
X_train, X_test, Y_train, Y_test =
train_test_split(df_messages['message'],df_messages['labels'],test_size=0.3,
random_state=1)
```

In [11]:
```
from collections import Counter
os =  RandomOverSampler(ratio=1.0)
#os =  RandomOverSampler(random_state=42)
X_train_res, Y_train_res = os.fit_sample(pd.DataFrame(X_train),
pd.DataFrame(Y_train))
print ("Distribution of class labels before resampling
{}".format(Counter(Y_train)))
print ("Distribution of class labels after resampling
{}".format(Counter(Y_train_res)))
```

```
Distribution of class labels before resampling Counter({0: 3371, 1: 529})
Distribution of class labels after resampling Counter({0: 3371, 1: 3371})
```

In [12]:
```
X_test.shape
```

Out[12]:
```
(1672,)
```

In [13]:
```
# Creating an instance of the CountVectorizer class
# If 'english', a built-in stop word list for English is used.
# There are known issues with 'english' and you should consider an alternative
vectorizer = CountVectorizer(lowercase=True, stop_words='english',
analyzer='word')

# Learn a vocabulary from one or more message using the fit_transform() function
vect_train = vectorizer.fit_transform(X_train)
print(vect_train.toarray())
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

In [14]:
```
from collections import Counter
os =  RandomOverSampler(ratio=1.0)
vect_train_res, Y_train_res = os.fit_sample(vect_train, Y_train)

print ("Distribution of class labels before resampling
{}".format(Counter(Y_train)))
print ("Distribution of class labels after resampling
{}".format(Counter(Y_train_res)))
```

```
Distribution of class labels before resampling Counter({0: 3371, 1: 529})
Distribution of class labels after resampling Counter({0: 3371, 1: 3371})
```

In [15]:
```
# Create an instance of MultinomialNB()
model_nb = MultinomialNB()

# Fit your data to the model
model_nb.fit(vect_train_res,Y_train_res)

# Use predict() to predict target class
predict_train = model_nb.predict(vect_train_res)
```

In [16]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

In [17]:
```
# Calculate Train Accuracy
print('Accuracy score: {}'.format(accuracy_score(Y_train_res, predict_train)))

# Calculate other metrics on your train results
print('Precision score: {}'.format(precision_score(Y_train_res, predict_train)))
print('Recall score: {}'.format(recall_score(Y_train_res, predict_train)))
print('F1 score: {}'.format(f1_score(Y_train_res, predict_train)))
```

```
Accuracy score: 0.9906555918125185
Precision score: 0.9861845972957084
Recall score: 0.9952536339365173
F1 score: 0.9906983611398199
```

In [18]:
```
# We apply the model into our test data
vect_test = vectorizer.transform(X_test)
prediction = model_nb.predict(vect_test)

# Calculate Test Accuracy
print('Accuracy score: {}'.format(accuracy_score(Y_test, prediction)))

# Calculate other metrics on your test data
print('Precision score: {}'.format(precision_score(Y_test, prediction)))
print('Recall score: {}'.format(recall_score(Y_test, prediction)))
print('F1 score: {}'.format(f1_score(Y_test, prediction)))
```

```
Accuracy score: 0.9706937799043063
Precision score: 0.8535564853556485
Recall score: 0.9357798165137615
F1 score: 0.8927789934354486
```

In [19]:
```
print(X_test[4:5])
vect_test1 = vectorizer.transform(X_test[4:5])
prediction1 = model_nb.predict(vect_test1)
```

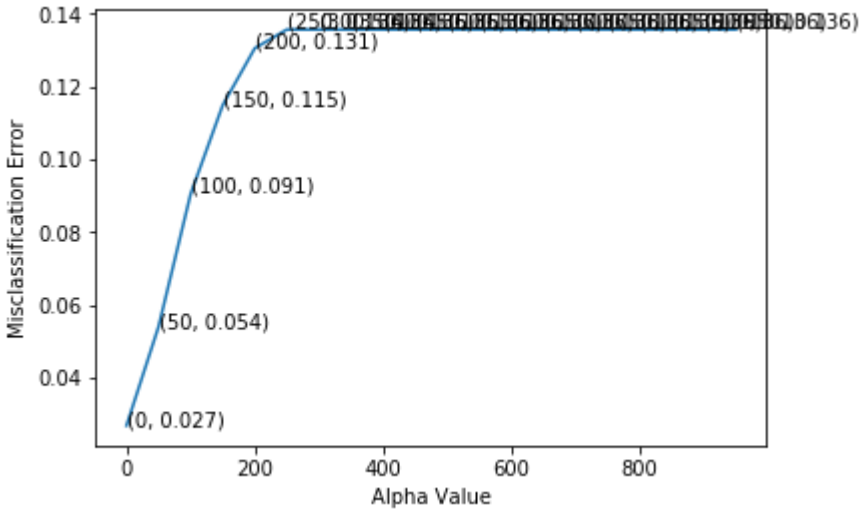
```
4674    Hi babe its Chloe, how r u? I was smashed on s...
Name: message, dtype: object
```

In [20]:
```
print(prediction1)
```

```
[1]
```

In [21]:
```
from sklearn.model_selection import cross_val_score
# creating odd list of Alpha for NB
myList = list(range(0,1000,50))
neighbors = myList
#list(filter(lambda x: x % 2 != 0, myList))

# empty list that will hold cv scores
cv_scores = []

# perform 10-fold cross validation
for k in neighbors:
    knn = MultinomialNB(alpha=k)
    scores = cross_val_score(knn, vect_train,Y_train, cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())

# changing to misclassification error
MSE = [1 - x for x in cv_scores]

# determining best k
optimal_k = neighbors[MSE.index(min(MSE))]
print('\nThe optimal number of neighbors is %d.' % optimal_k)

# plot misclassification error vs k
plt.plot(neighbors, MSE)

for xy in zip(neighbors, np.round(MSE,3)):
    plt.annotate('(%s, %s)' % xy, xy=xy, textcoords='data')

plt.xlabel('Alpha Value')
plt.ylabel('Misclassification Error')
plt.show()

print("the misclassification error for each Alpha value is : ", np.round(MSE,3))
```

```
The optimal number of neighbors is 0.
```



```
the misclassification error for each Alpha value is :  [0.027 0.054 0.091 0.115
0.131 0.136 0.136 0.136 0.136 0.136 0.136 0.136
 0.136 0.136 0.136 0.136 0.136 0.136 0.136 0.136]
```

In [ ]: