Natural Language Processing using Python

In [4]:
```
import pandas as pd
# import and instantiate CountVectorizer (with the default parameters)
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
```

In [1]:
```
import seaborn as sns
```

In [2]:
```
df = sns.load_dataset('tips')
```

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

Out[3]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2

In [48]:
```
#url = 'https://raw.githubusercontent.com/justmarkham/pycon-2016-tutorial/master/data/sms.tsv'
#sms = pd.read_table(url, header=None, names=['label', 'message'])
```

In [ ]:
```

```

In [54]:
```
#sms.to_csv('sms.csv', index=False)
```

In [ ]:
```
# read file into pandas using a relative path
# path = 'sms.tsv'
# sms = pd.read_table(path, header=None, names=['label', 'message'])
```

In [6]:
```
sms = pd.read_csv('sms.csv')
```

In [7]:
```
sms.shape
```

Out[7]:
```
(5572, 2)
```

In [8]:
```
sms.head(4)
```

Out[8]:

	label	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...

In [4]:
```
sms.shape
```

Out[4]:
```
(5572, 2)
```

In [5]:
```
# examine the first 10 rows
sms.head(10)
```

Out[5]:

	label	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...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...

In [9]:
```
# examine the class distribution
sms.label.value_counts()
```

Out[9]:
```
ham     4825
spam     747
Name: label, dtype: int64
```

In [10]:
```
# convert label to a numerical variable
sms['label_num'] = sms.label.map({'ham':0, 'spam':1})
```

In [11]:
```
# check that the conversion worked
sms.head(10)
```

Out[11]:

	label	message	label_num
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0
5	spam	FreeMsg Hey there darling it's been 3 week's n...	1
6	ham	Even my brother is not like to speak with me. ...	0
7	ham	As per your request 'Melle Melle (Oru Minnamin...	0
8	spam	WINNER!! As a valued network customer you have...	1
9	spam	Had your mobile 11 months or more? U R entitle...	1

In [12]:
```
# how to define X and y (from the SMS data) for use with COUNTVECTORIZER
X = sms.message
y = sms.label_num
print(X.shape)
print(y.shape)
```

```
(5572,)
(5572,)
```

In [13]:
```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4179,)
(1393,)
(4179,)
(1393,)
```

In [11]:
```
#Part 4: Vectorizing our dataset
```

In [14]:
```
# instantiate the vectorizer
vect = CountVectorizer()
```

In [15]:
```
# learn training data vocabulary, then use it to create a document-term matrix
vect.fit(X_train)
X_train_dtm = vect.transform(X_train)
```

In [16]:
```
# equivalently: combine fit and transform into a single step
X_train_dtm = vect.fit_transform(X_train)
```

In [17]:
```
# examine the fitted vocabulary
vect.get_feature_names()[:-10:-1]
```

Out[17]:
```
['〓ud', 'èn', 'zyada', 'zouk', 'zoom', 'zoe', 'zindgi', 'zhong', 'zeros']
```

In [18]:
```
# total feature count
len(vect.get_feature_names())
```

Out[18]:
```
7456
```

In [19]:
```
# examine the document-term matrix
X_train_dtm
```

Out[19]:
```
<4179x7456 sparse matrix of type '<class 'numpy.int64'>'
	with 55209 stored elements in Compressed Sparse Row format>
```

In [20]:
```
# transform testing data (using fitted vocabulary) into a document-term matrix
X_test_dtm = vect.transform(X_test)
X_test_dtm
```

Out[20]:
```
<1393x7456 sparse matrix of type '<class 'numpy.int64'>'
	with 17604 stored elements in Compressed Sparse Row format>
```

In [22]:
```
#Part 5: Building and evaluating a model
```

In [22]:
```
# import and instantiate a Multinomial Naive Bayes model
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
```

In [23]:
```
# train the model using X_train_dtm (timing it with an IPython "magic command")
%time nb.fit(X_train_dtm, y_train)
```

```
CPU times: user 4 ms, sys: 0 ns, total: 4 ms
Wall time: 5.75 ms
```

Out[23]:
```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

In [24]:
```
# make class predictions for X_test_dtm
y_pred_class = nb.predict(X_test_dtm)
```

In [25]:
```
# calculate accuracy of class predictions
from sklearn import metrics
metrics.accuracy_score(y_test, y_pred_class)
```

Out[25]:
```
0.9885139985642498
```

In [26]:
```
# print the confusion matrix
metrics.confusion_matrix(y_test, y_pred_class)
```

Out[26]:
```
array([[1203,    5],
       [  11,  174]])
```

In [28]:
```
# print message text for the false positives (ham incorrectly classified as spam)
X_test[(y_pred_class==1) & (y_test==0)]
```

Out[28]:
```
574               Waiting for your call.
3375             Also andros ice etc etc
45      No calls..messages..missed calls
3415             No pic. Please re-send.
1988    No calls..messages..missed calls
Name: message, dtype: object
```

In [29]:
```
# print message text for the false positives (ham incorrectly classified as spam)
X_test[(y_pred_class > y_test)]
```

Out[29]:
```
574               Waiting for your call.
3375             Also andros ice etc etc
45      No calls..messages..missed calls
3415             No pic. Please re-send.
1988    No calls..messages..missed calls
Name: message, dtype: object
```

In [30]:
```
# print message text for the false negatives (spam incorrectly classified as ham)
X_test[(y_pred_class < y_test)]
```

Out[30]:
```
3132    LookAtMe!: Thanks for your purchase of a video...
5       FreeMsg Hey there darling it's been 3 week's n...
3530    Xmas & New Years Eve tickets are now on sale f...
684     Hi I'm sue. I am 20 years old and work as a la...
1875    Would you like to see my XXX pics they are so ...
1893    CALL 09090900040 & LISTEN TO EXTREME DIRTY LIV...
4298    thesmszone.com lets you send free anonymous an...
4949    Hi this is Amy, we will be sending you a free ...
2821    INTERFLORA -  It's not too late to order Inter...
2247    Hi ya babe x u 4goten bout me?' scammers getti...
4514    Money i have won wining number 946 wot do i do...
Name: message, dtype: object
```

In [31]:
```
# example false negative
X_test[2247]
```

Out[31]:
```
"Hi ya babe x u 4goten bout me?' scammers getting smart..Though this is a regular vodafone no, if you respond you get further prem rate msg/subscription. Othe
r nos used also. Beware!"
```

In [32]:
```
# calculate predicted probabilities for X_test_dtm (poorly calibrated)
y_pred_prob = nb.predict_proba(X_test_dtm)[:, 1]
y_pred_prob
```

Out[32]:
```
array([2.87744864e-03, 1.83488846e-05, 2.07301295e-03, ...,
       1.09026171e-06, 1.00000000e+00, 3.98279868e-09])
```

In [33]:
```
# calculate AUC
metrics.roc_auc_score(y_test, y_pred_prob)
```

Out[33]:
```
0.9866431000536962
```

In [34]:
```
#Part 6: Comparing models
```

Logistic Regression

In [27]:
```
# import and instantiate a logistic regression model
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [28]:
```
# train the model using X_train_dtm
%time logreg.fit(X_train_dtm, y_train)
```

```
CPU times: user 168 ms, sys: 20 ms, total: 188 ms
Wall time: 795 ms
```

Out[28]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [29]:
```
# make class predictions for X_test_dtm
y_pred_class = logreg.predict(X_test_dtm)
```

In [30]:
```
# calculate predicted probabilities for X_test_dtm (well calibrated)
y_pred_prob = logreg.predict_proba(X_test_dtm)[:, 1]
y_pred_prob
```

Out[30]:
```
array([0.01269556, 0.00347183, 0.00616517, ..., 0.03354907, 0.99725053,
       0.00157706])
```

In [31]:
```
# calculate accuracy
metrics.accuracy_score(y_test, y_pred_class)
```

Out[31]:
```
0.9877961234745154
```

In [32]:
```
# calculate AUC
metrics.roc_auc_score(y_test, y_pred_prob)
```

Out[32]:
```
0.9936817612314301
```

In [ ]:
```

```