Natural Language Processing for Text Classification with NLTK and Scikit-learn

We will use:

1.Regular Expressions

2.Feature Engineering

3.Multiple scikit-learn Classifiers

4.Ensemble Methods

1. Import Necessary Libraries

To ensure the necessary libraries are installed correctly and up-to-date, print the version numbers for each library. This will also improve the reproducibility of our project.

In [1]:

```
import sys
import nltk
import sklearn
import pandas
import numpy

print('Python: {}'.format(sys.version))
print('NLTK: {}'.format(nltk.__version__))
print('Scikit-learn: {}'.format(sklearn.__version__))
print('Pandas: {}'.format(pandas.__version__))
print('Numpy: {}'.format(numpy.__version__))
```

```
Python: 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.190
0 64 bit (AMD64)]
NLTK: 3.3
Scikit-learn: 0.19.1
Pandas: 0.23.0
Numpy: 1.14.3
```

2. Load the Dataset

Now that we have ensured that our libraries are installed correctly, let's load the data set as a Pandas DataFrame. Furthermore, let's extract some useful information such as the column information and class distributions.

The data set we will be using comes from the UCI Machine Learning Repository. It contains over 5000 SMS labeled messages that have been collected for mobile phone spam research. It can be downloaded from the following URL:

https://archive.ics.uci.edu/ml/datasets/sms+spam+collection (https://archive.ics.uci.edu/ml/datasets/sms+spam+collection)

In [3]:

```
import pandas as pd
import numpy as np

# load the dataset of SMS messages
df = pd.read_table('SMSSPamCollection', header=None, encoding='utf-8')
```

In [4]:

```
# print useful information about the dataset
print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
0    5572 non-null object
1    5572 non-null object
dtypes: object(2)
memory usage: 87.1+ KB
None
      0                                                  1
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 [5]:

```
# check class distribution
classes = df[0]
print(classes.value_counts())
```

```
ham     4825
spam     747
Name: 0, dtype: int64
```

2. Preprocess the Data

Preprocessing the data is an essential step in natural language process. In the following cells, we will convert our class labels to binary values using the LabelEncoder from sklearn, replace email addresses, URLs, phone numbers, and other symbols by using regular expressions, remove stop words, and extract word stems.

In [6]:

```
from sklearn.preprocessing import LabelEncoder

# convert class labels to binary values, 0 = ham and 1 = spam
encoder = LabelEncoder()
Y = encoder.fit_transform(classes)

print(Y[:10])
```

```
[0 0 1 0 0 1 0 0 1 1]
```

In [7]:

```
# store the SMS message data
text_messages = df[1]
print(text_messages[:10])
```

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

2.1 Regular Expressions

Some common regular expression metacharacters - copied from wikipedia

^ Matches the starting position within the string. In line-based tools, it matches the starting position of any line.

. Matches any single character (many applications exclude newlines, and exactly which characters are considered newlines is flavor-, character-encoding-, and platform-specific, but it is safe to assume that the line feed character is included). Within POSIX bracket expressions, the dot character matches a literal dot. For example, a.c matches "abc", etc., but [a.c] matches only "a", ".", or "c".

[ ] A bracket expression. Matches a single character that is contained within the brackets. For example, [abc] matches "a", "b", or "c". [a-z] specifies a range which matches any lowercase letter from "a" to "z". These forms can be mixed: [abcx-z] matches "a", "b", "c", "x", "y", or "z", as does [a-cx-z]. The - character is treated as a literal character if it is the last or the first (after the ^, if present) character within the brackets: [abc-], [-abc]. Note that backslash escapes are not allowed. The] character can be included in a bracket expression if it is the first (after the ^) character: []abc].

[^] Matches a single character that is not contained within the brackets. For example, [^abc] matches any character other than "a", "b", or "c". [^a-z] matches any single character that is not a lowercase letter from "a" to "z". Likewise, literal characters and ranges can be mixed.

$ Matches the ending position of the string or the position just before a string-ending newline. In line-based tools, it matches the ending position of any line.

( ) Defines a marked subexpression. The string matched within the parentheses can be recalled later (see the next entry, \n). A marked subexpression is also called a block or capturing group. BRE mode requires ( ).

\n Matches what the nth marked subexpression matched, where n is a digit from 1 to 9. This construct is vaguely defined in the POSIX.2 standard. Some tools allow referencing more than nine capturing groups.

***** Matches the preceding element zero or more times. For example, ab*c matches "ac", "abc", "abbbc", etc. [xyz]* matches "", "x", "y", "z", "zx", "zyx", "xyzzy", and so on. (ab)* matches "", "ab", "abab", "ababab", and so on.

{m,n} Matches the preceding element at least m and not more than n times. For example, a{3,5} matches only "aaa", "aaaa", and "aaaaa". This is not found in a few older instances of regexes. BRE mode requires {m,n}.

In [8]:

```
# use regular expressions to replace email addresses, URLs, phone numbers, other number
s

# Replace email addresses with 'email'
processed = text_messages.str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                      'emailaddress')

# Replace URLs with 'webaddress'
processed = processed.str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',
                                  'webaddress')

# Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
processed = processed.str.replace(r'£|\$', 'moneysymb')

# Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashe
s) with 'phonenumber'
processed = processed.str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',
                                  'phonenumbr')

# Replace numbers with 'numbr'
processed = processed.str.replace(r'\d+(\.\d+)?', 'numbr')
```

In [9]:

```
# Remove punctuation
processed = processed.str.replace(r'[^\w\d\s]', ' ')

# Replace whitespace between terms with a single space
processed = processed.str.replace(r'\s+', ' ')

# Remove leading and trailing whitespace
processed = processed.str.replace(r'^\s+|\s+?$', '')
```

In [10]:

```
# change words to lower case - Hello, HELLO, hello are all the same word
processed = processed.str.lower()
print(processed)
```

```
0       go until jurong point crazy available only in ...
1                             ok lar joking wif u oni
2       free entry in numbr a wkly comp to win fa cup ...
3             u dun say so early hor u c already then say
4       nah i don t think he goes to usf he lives arou...
5       freemsg hey there darling it s been numbr week...
6       even my brother is not like to speak with me t...
7       as per your request melle melle oru minnaminun...
8       winner as a valued network customer you have b...
9       had your mobile numbr months or more u r entit...
10      i m gonna be home soon and i don t want to tal...
11      six chances to win cash from numbr to numbr nu...
12      urgent you have won a numbr week free membersh...
13      i ve been searching for the right words to tha...
14                    i have a date on sunday with will
15      xxxmobilemovieclub to use your credit click th...
16                             oh k i m watching here
17      eh u remember how numbr spell his name yes i d...
18      fine if that s the way u feel that s the way i...
19      england v macedonia dont miss the goals team n...
20              is that seriously how you spell his name
21      i m going to try for numbr months ha ha only j...
22         so ü pay first lar then when is da stock comin
23      aft i finish my lunch then i go str down lor a...
24      ffffffffff alright no way i can meet up with y...
25      just forced myself to eat a slice i m really n...
26                        lol your always so convincing
27      did you catch the bus are you frying an egg di...
28      i m back amp we re packing the car now i ll le...
29      ahhh work i vaguely remember that what does it...
                              ...
5542            armand says get your ass over to epsilon
5543               u still havent got urself a jacket ah
5544    i m taking derek amp taylor to walmart if i m ...
5545        hi its in durban are you still on this number
5546            ic there are a lotta childporn cars then
5547    had your contract mobile numbr mnths latest mo...
5548                     no i was trying it all weekend v
5549    you know wot people wear t shirts jumpers hat ...
5550           cool what time you think you can get here
5551    wen did you get so spiritual and deep that s g...
5552    have a safe trip to nigeria wish you happiness...
5553                           hahaha use your brain dear
5554    well keep in mind i ve only got enough gas for...
5555    yeh indians was nice tho it did kane me off a ...
5556    yes i have so that s why u texted pshew missin...
5557    no i meant the calculation is the same that lt...
5558                              sorry i ll call later
5559    if you aren t here in the next lt gt hours imm...
5560                      anything lor juz both of us lor
5561    get me out of this dump heap my mom decided to...
5562    ok lor sony ericsson salesman i ask shuhui the...
5563                              ard numbr like dat lor
5564    why don t you wait til at least wednesday to s...
5565                                           huh y lei
5566    reminder from onumbr to get numbr pounds free ...
5567    this is the numbrnd time we have tried numbr c...
5568                  will ü b going to esplanade fr home
5569    pity was in mood for that so any other suggest...
5570    the guy did some bitching but i acted like i d...
```

```
5571                          rofl its true to its name
Name: 1, Length: 5572, dtype: object
```

In [11]:

```
from nltk.corpus import stopwords

# remove stop words from text messages

stop_words = set(stopwords.words('english'))

processed = processed.apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words))
```

In [12]:

```
# Remove word stems using a Porter stemmer
ps = nltk.PorterStemmer()

processed = processed.apply(lambda x: ' '.join(
    ps.stem(term) for term in x.split()))
```

3. Generating Features

Feature engineering is the process of using domain knowledge of the data to create features for machine learning algorithms. In this project, the words in each text message will be our features. For this purpose, it will be necessary to tokenize each word. We will use the 1500 most common words as features.

In [13]:

```
from nltk.tokenize import word_tokenize

# create bag-of-words
all_words = []

for message in processed:
    words = word_tokenize(message)
    for w in words:
        all_words.append(w)

all_words = nltk.FreqDist(all_words)
```

In [14]:

```
# print the total number of words and the 15 most common words
print('Number of words: {}'.format(len(all_words)))
print('Most common words: {}'.format(all_words.most_common(15)))
```

```
Number of words: 6579
Most common words: [('numbr', 2648), ('u', 1207), ('call', 674), ('go', 45
6), ('get', 451), ('ur', 391), ('gt', 318), ('lt', 316), ('come', 304),
('moneysymbnumbr', 303), ('ok', 293), ('free', 284), ('day', 276), ('kno
w', 275), ('love', 266)]
```

In [15]:

```
# use the 1500 most common words as features
word_features = list(all_words.keys())[:1500]
```

In [17]:

```
# The find_features function will determine which of the 1500 word features are contain
ed in the review
def find_features(message):
    words = word_tokenize(message)
    features = {}
    for word in word_features:
        features[word] = (word in words)

    return features

# Lets see an example!
features = find_features(processed[0])
for key, value in features.items():
    if value == True:
        print (key)
```

```
go
jurong
point
crazi
avail
bugi
n
great
world
la
e
buffet
cine
got
amor
wat
```

In [20]:

```
# Now lets do it for all the messages
messages = list(zip(processed, Y))

# define a seed for reproducibility
seed = 1
np.random.seed = seed
np.random.shuffle(messages)

# call find_features function for each SMS message
featuresets = [(find_features(text), label) for (text, label) in messages]
```

In [21]:

```
# we can split the featuresets into training and testing datasets using sklearn
from sklearn import model_selection

# split the data into training and testing datasets
training, testing = model_selection.train_test_split(featuresets, test_size = 0.25, ran
dom_state=seed)
```

In [22]:

```
print(len(training))
print(len(testing))
```

```
4179
1393
```

4. Scikit-Learn Classifiers with NLTK

Now that we have our dataset, we can start building algorithms! Let's start with a simple linear support vector classifier, then expand to other algorithms. We'll need to import each algorithm we plan on using from sklearn. We also need to import some performance metrics, such as accuracy_score and classification_report.

In [23]:

```
# We can use sklearn algorithms in NLTK
from nltk.classify.scikitlearn import SklearnClassifier
from sklearn.svm import SVC

model = SklearnClassifier(SVC(kernel = 'linear'))

# train the model on the training data
model.train(training)

# and test on the testing dataset!
accuracy = nltk.classify.accuracy(model, testing)*100
print("SVC Accuracy: {}".format(accuracy))
```

```
SVC Accuracy: 98.49246231155779
```

In [24]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

# Define models to train
names = ["K Nearest Neighbors", "Decision Tree", "Random Forest", "Logistic Regression"
, "SGD Classifier",
         "Naive Bayes", "SVM Linear"]

classifiers = [
    KNeighborsClassifier(),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    LogisticRegression(),
    SGDClassifier(max_iter = 100),
    MultinomialNB(),
    SVC(kernel = 'linear')
]

models = zip(names, classifiers)

for name, model in models:
    nltk_model = SklearnClassifier(model)
    nltk_model.train(training)
    accuracy = nltk.classify.accuracy(nltk_model, testing)*100
    print("{} Accuracy: {}".format(name, accuracy))
```

```
K Nearest Neighbors Accuracy: 94.18521177315147
Decision Tree Accuracy: 97.48743718592965
Random Forest Accuracy: 98.20531227566404
Logistic Regression Accuracy: 98.42067480258436
SGD Classifier Accuracy: 97.98994974874373
Naive Bayes Accuracy: 98.56424982053123
SVM Linear Accuracy: 98.49246231155779
```

In [26]:

```
# Ensemble methods - Voting classifier
from sklearn.ensemble import VotingClassifier

names = ["K Nearest Neighbors", "Decision Tree", "Random Forest", "Logistic Regression"
, "SGD Classifier",
         "Naive Bayes", "SVM Linear"]

classifiers = [
    KNeighborsClassifier(),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    LogisticRegression(),
    SGDClassifier(max_iter = 100),
    MultinomialNB(),
    SVC(kernel = 'linear')
]

models = list(zip(names, classifiers))

nltk_ensemble = SklearnClassifier(VotingClassifier(estimators = models, voting = 'hard'
, n_jobs = -1))
nltk_ensemble.train(training)
accuracy = nltk.classify.accuracy(nltk_model, testing)*100
print("Voting Classifier: Accuracy: {}".format(accuracy))
```

```
Voting Classifier: Accuracy: 98.49246231155779
```

In [30]:

```
# make class label prediction for testing set
txt_features, labels = zip(*testing)

prediction = nltk_ensemble.classify_many(txt_features)
```

```
C:\Users\user\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:1
51: DeprecationWarning: The truth value of an empty array is ambiguous. Re
turning False, but in future this will result in an error. Use `array.size
> 0` to check that an array is not empty.
  if diff:
```

In [31]:

```
# print a confusion matrix and a classification report
print(classification_report(labels, prediction))

pd.DataFrame(
    confusion_matrix(labels, prediction),
    index = [['actual', 'actual'], ['ham', 'spam']],
    columns = [['predicted', 'predicted'], ['ham', 'spam']])
```

```
             precision    recall  f1-score   support

          0       0.98      1.00      0.99      1218
          1       0.99      0.89      0.94       175

avg / total       0.98      0.98      0.98      1393
```

Out[31]:

		predicted	
		ham	**spam**
actual	**ham**	1216	2
	spam	19	156