Building Machine Learning Classifiers: Random Forest on a holdout test set

Read in & clean text

```
import nltk
import pandas as pd
import re
from sklearn.feature_extraction.text import TfidfVectorizer
import string

stopwords = nltk.corpus.stopwords.words('english')
ps = nltk.PorterStemmer()

data = pd.read_csv("SMSSpamCollection.tsv", sep='\t')
data.columns = ['label', 'body_text']

def count_punct(text):
    count = sum([1 for char in text if char in string.punctuation])
    return round(count/(len(text) - text.count(" ")), 3)*100

data['body_len'] = data['body_text'].apply(lambda x: len(x) - x.count(" "))
data['punct%'] = data['body_text'].apply(lambda x: count_punct(x))

def clean_text(text):
    text = "".join([word.lower() for word in text if word not in string.punctuation])
    tokens = re.split('\W+', text)
    text = [ps.stem(word) for word in tokens if word not in stopwords]
    return text

tfidf_vect = TfidfVectorizer(analyzer=clean_text)
X_tfidf = tfidf_vect.fit_transform(data['body_text'])

X_features = pd.concat([data['body_len'], data['punct%'], pd.DataFrame(X_tfidf.toarray())], axis=1)
X_features.head()
```

	body_len	punct%	0	1	2	3	4	5	6	7	...	8094	8095	8096	8097	8098	8099	8100	8101	8102	8103
0	128	4.7	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
1	49	4.1	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
2	62	3.2	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
3	28	7.1	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
4	135	4.4	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

Explore RandomForestClassifier through Holdout Set

```
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(X_features, data['label'], test_size=0.2)


from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=50, max_depth=20, n_jobs=-1)
rf_model = rf.fit(X_train, y_train)


sorted(zip(rf_model.feature_importances_, X_train.columns), reverse=True)[0:10]
```

```
[(0.071067778644078275, 'body_len'),
 (0.040562335897847433, 7350),
 (0.035736155950968088, 3134),
 (0.025830800898315055, 2031),
 (0.020706891454006282, 1881),
 (0.020667459644832679, 5724),
 (0.020246234600271286, 4796),
 (0.016709671666146234, 5988),
 (0.016333631268556359, 1803),
 (0.015520152981795897, 2171)]
```

```
y_pred = rf_model.predict(X_test)
precision, recall, fscore, support = score(y_test, y_pred, pos_label='spam', average='binary')
```

```
print('Precision: {} / Recall: {} / Accuracy: {}'.format(round(precision, 3),
                                                         round(recall, 3),
                                                         round((y_pred==y_test).sum() / len(y_pred),3)))
```

```
Precision: 1.0 / Recall: 0.552 / Accuracy: 0.934
```