Building Machine Learning Classifiers: Model selection

Read in & clean text

In [1]:
```
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.pun
ctuation])
    tokens = re.split('\W+', text)
    text = [ps.stem(word) for word in tokens if word not in stopwords]
    return text
```

Split into train/test

In [2]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data[['body_text', 'bod
y_len', 'punct%']], data['label'], test_size=0.2)
```

Vectorize text

In [3]:
```
tfidf_vect = TfidfVectorizer(analyzer=clean_text)
tfidf_vect_fit = tfidf_vect.fit(X_train['body_text'])

tfidf_train = tfidf_vect_fit.transform(X_train['body_text'])
tfidf_test = tfidf_vect_fit.transform(X_test['body_text'])

X_train_vect = pd.concat([X_train[['body_len', 'punct%']].reset_index(drop=
True),
           pd.DataFrame(tfidf_train.toarray())], axis=1)
X_test_vect = pd.concat([X_test[['body_len', 'punct%']].reset_index(drop=Tr
ue),
           pd.DataFrame(tfidf_test.toarray())], axis=1)

X_train_vect.head()
```

Out[3]:

	body_len	punct%	0	1	2	3	4	5	6	7	...	7153	7154	7155	7156	7157	71
0	19	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	115	3.5	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	106	2.8	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	29	3.4	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
4	152	4.6	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

5 rows × 7165 columns

Final evaluation of models

In [4]:
```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassi
fier
from sklearn.metrics import precision_recall_fscore_support as score
import time
```

In [5]:
```
rf = RandomForestClassifier(n_estimators=150, max_depth=None, n_jobs=-1)

start = time.time()
rf_model = rf.fit(X_train_vect, y_train)
end = time.time()
fit_time = (end - start)

start = time.time()
y_pred = rf_model.predict(X_test_vect)
end = time.time()
pred_time = (end - start)

precision, recall, fscore, train_support = score(y_test, y_pred, pos_label=
'spam', average='binary')
print('Fit time: {} / Predict time: {} ---- Precision: {} / Recall: {} / Ac
curacy: {}'.format(
    round(fit_time, 3), round(pred_time, 3), round(precision, 3), round(rec
all, 3), round((y_pred==y_test).sum()/len(y_pred), 3)))
```

```
Fit time: 1.782 / Predict time: 0.213 ---- Precision: 1.0 / Recall: 0.81 /
Accuracy: 0.975
```

In [6]:
```
gb = GradientBoostingClassifier(n_estimators=150, max_depth=11)

start = time.time()
gb_model = gb.fit(X_train_vect, y_train)
end = time.time()
fit_time = (end - start)

start = time.time()
y_pred = gb_model.predict(X_test_vect)
end = time.time()
pred_time = (end - start)

precision, recall, fscore, train_support = score(y_test, y_pred, pos_label=
'spam', average='binary')
print('Fit time: {} / Predict time: {} ---- Precision: {} / Recall: {} / Ac
curacy: {}'.format(
    round(fit_time, 3), round(pred_time, 3), round(precision, 3), round(rec
all, 3),
    round((y_pred==y_test).sum()/len(y_pred), 3)))
```

```
Fit time: 186.61 / Predict time: 0.135 ---- Precision: 0.889 / Recall: 0.81
6 / Accuracy: 0.962
```

In [ ]:
```

```