Building Machine Learning Classifiers: Explore Random Forest model with grid-search

Grid-search: Exhaustively search all parameter combinations in a given grid to determine the best model.

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)

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	0.047	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	0.041	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	0.032	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	0.071	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	0.044	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

5 rows × 8106 columns

Build our own Grid-search

```
from sklearn.ensemble import RandomForestClassifier
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)
```

```
def train_RF(n_est, depth):
    rf = RandomForestClassifier(n_estimators=n_est, max_depth=depth, n_jobs=-1)
    rf_model = rf.fit(X_train, y_train)
    y_pred = rf_model.predict(X_test)
    precision, recall, fscore, support = score(y_test, y_pred, pos_label='spam', average='binary')
    print('Est: {} / Depth: {} ---- Precision: {} / Recall: {} / Accuracy: {}'.format(
        n_est, depth, round(precision, 3), round(recall, 3),
        round((y_pred==y_test).sum() / len(y_pred), 3)))
```

```
for n_est in [10, 50, 100]:
    for depth in [10, 20, 30, None]:
        train_RF(n_est, depth)
```

```
Est: 10 / Depth: 10 ---- Precision: 1.0 / Recall: 0.216 / Accuracy: 0.892
Est: 10 / Depth: 20 ---- Precision: 0.975 / Recall: 0.516 / Accuracy: 0.932
Est: 10 / Depth: 30 ---- Precision: 1.0 / Recall: 0.647 / Accuracy: 0.952
Est: 10 / Depth: None ---- Precision: 0.984 / Recall: 0.784 / Accuracy: 0.969
Est: 50 / Depth: 10 ---- Precision: 1.0 / Recall: 0.235 / Accuracy: 0.895
```

```
Est: 50 / Depth: 20 ---- Precision: 1.0 / Recall: 0.562 / Accuracy: 0.94
Est: 50 / Depth: 30 ---- Precision: 1.0 / Recall: 0.667 / Accuracy: 0.954
Est: 50 / Depth: None ---- Precision: 0.985 / Recall: 0.843 / Accuracy: 0.977
Est: 100 / Depth: 10 ---- Precision: 1.0 / Recall: 0.242 / Accuracy: 0.896
Est: 100 / Depth: 20 ---- Precision: 1.0 / Recall: 0.601 / Accuracy: 0.945
Est: 100 / Depth: 30 ---- Precision: 0.981 / Recall: 0.686 / Accuracy: 0.955
Est: 100 / Depth: None ---- Precision: 1.0 / Recall: 0.83 / Accuracy: 0.977
```