Building Machine Learning Classifiers: Explore Gradient Boosting 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)*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

5 rows × 8106 columns

Explore GradientBoostingClassifier Attributes & Hyperparameters

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
print(dir(GradientBoostingClassifier))
print(GradientBoostingClassifier())
```

```
['_SUPPORTED_LOSS', '__abstractmethods__', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__',
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_split=1e-07, min_samples_leaf=1,
              min_samples_split=2, min_weight_fraction_leaf=0.0,
              n_estimators=100, presort='auto', random_state=None,
              subsample=1.0, verbose=0, warm_start=False)
```

Build our own Grid-search

```
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_GB(est, max_depth, lr):
    gb = GradientBoostingClassifier(n_estimators=est, max_depth=max_depth, learning_rate=lr)
    gb_model = gb.fit(X_train, y_train)
    y_pred = gb_model.predict(X_test)
    precision, recall, fscore, train_support = score(y_test, y_pred, pos_label='spam', average='binary')
    print('Est: {} / Depth: {} / LR: {} ---- Precision: {} / Recall: {} / Accuracy: {}'.format(
        est, max_depth, lr, round(precision, 3), round(recall, 3),
        round((y_pred==y_test).sum()/len(y_pred), 3)))
```

```
for n_est in [50, 100, 150]:
    for max_depth in [3, 7, 11, 15]:
        for lr in [0.01, 0.1, 1]:
            train_GB(n_est, max_depth, lr)
```

```
/Users/djedamski/.pyenv/versions/3.5.3/lib/python3.5/site-packages/sklearn/metrics/classification.py:1113: UndefinedMetricWarning
  'precision', 'predicted', average, warn_for)
Est: 50 / Depth: 3 / LR: 0.01 ---- Precision: 0.0 / Recall: 0.0 / Accuracy: 0.868
Est: 50 / Depth: 3 / LR: 0.1 ---- Precision: 1.0 / Recall: 0.687 / Accuracy: 0.959
Est: 50 / Depth: 3 / LR: 1 ---- Precision: 0.88 / Recall: 0.796 / Accuracy: 0.959
Est: 50 / Depth: 7 / LR: 0.01 ---- Precision: 0.0 / Recall: 0.0 / Accuracy: 0.868
Est: 50 / Depth: 7 / LR: 0.1 ---- Precision: 0.968 / Recall: 0.83 / Accuracy: 0.974
Est: 50 / Depth: 7 / LR: 1 ---- Precision: 0.917 / Recall: 0.823 / Accuracy: 0.967
Est: 50 / Depth: 11 / LR: 0.01 ---- Precision: 1.0 / Recall: 0.027 / Accuracy: 0.872
Est: 50 / Depth: 11 / LR: 0.1 ---- Precision: 0.962 / Recall: 0.871 / Accuracy: 0.978
Est: 50 / Depth: 11 / LR: 1 ---- Precision: 0.926 / Recall: 0.85 / Accuracy: 0.971
Est: 50 / Depth: 15 / LR: 0.01 ---- Precision: 0.0 / Recall: 0.0 / Accuracy: 0.868
Est: 50 / Depth: 15 / LR: 0.1 ---- Precision: 0.977 / Recall: 0.857 / Accuracy: 0.978
Est: 50 / Depth: 15 / LR: 1 ---- Precision: 0.919 / Recall: 0.85 / Accuracy: 0.97
Est: 100 / Depth: 3 / LR: 0.01 ---- Precision: 0.987 / Recall: 0.51 / Accuracy: 0.934
Est: 100 / Depth: 3 / LR: 0.1 ---- Precision: 0.991 / Recall: 0.776 / Accuracy: 0.969
Est: 100 / Depth: 3 / LR: 1 ---- Precision: 0.901 / Recall: 0.803 / Accuracy: 0.962
Est: 100 / Depth: 7 / LR: 0.01 ---- Precision: 0.989 / Recall: 0.612 / Accuracy: 0.948
Est: 100 / Depth: 7 / LR: 0.1 ---- Precision: 0.985 / Recall: 0.871 / Accuracy: 0.981
Est: 100 / Depth: 7 / LR: 1 ---- Precision: 0.922 / Recall: 0.81 / Accuracy: 0.966
Est: 100 / Depth: 11 / LR: 0.01 ---- Precision: 0.991 / Recall: 0.741 / Accuracy: 0.965
Est: 100 / Depth: 11 / LR: 0.1 ---- Precision: 0.984 / Recall: 0.864 / Accuracy: 0.98
Est: 100 / Depth: 11 / LR: 1 ---- Precision: 0.912 / Recall: 0.844 / Accuracy: 0.969
Est: 100 / Depth: 15 / LR: 0.01 ---- Precision: 0.992 / Recall: 0.796 / Accuracy: 0.972
Est: 100 / Depth: 15 / LR: 0.1 ---- Precision: 0.977 / Recall: 0.871 / Accuracy: 0.98
Est: 100 / Depth: 15 / LR: 1 ---- Precision: 0.932 / Recall: 0.844 / Accuracy: 0.971
Est: 150 / Depth: 3 / LR: 0.01 ---- Precision: 0.988 / Recall: 0.537 / Accuracy: 0.938
Est: 150 / Depth: 3 / LR: 0.1 ---- Precision: 0.992 / Recall: 0.81 / Accuracy: 0.974
Est: 150 / Depth: 3 / LR: 1 ---- Precision: 0.902 / Recall: 0.816 / Accuracy: 0.964
Est: 150 / Depth: 7 / LR: 0.01 ---- Precision: 0.99 / Recall: 0.687 / Accuracy: 0.958
```

```
Est: 150 / Depth: 7 / LR: 0.1 ---- Precision: 0.977 / Recall: 0.857 / Accuracy: 0.978
Est: 150 / Depth: 7 / LR: 1 ---- Precision: 0.937 / Recall: 0.81 / Accuracy: 0.968
Est: 150 / Depth: 11 / LR: 0.01 ---- Precision: 0.983 / Recall: 0.796 / Accuracy: 0.971
Est: 150 / Depth: 11 / LR: 0.1 ---- Precision: 0.985 / Recall: 0.871 / Accuracy: 0.981
Est: 150 / Depth: 11 / LR: 1 ---- Precision: 0.904 / Recall: 0.837 / Accuracy: 0.967
Est: 150 / Depth: 15 / LR: 0.01 ---- Precision: 0.975 / Recall: 0.796 / Accuracy: 0.97
Est: 150 / Depth: 15 / LR: 0.1 ---- Precision: 0.977 / Recall: 0.864 / Accuracy: 0.979
Est: 150 / Depth: 15 / LR: 1 ---- Precision: 0.913 / Recall: 0.857 / Accuracy: 0.97
```