Lecture 10: Introduction to scikit-learn

CS 167: Machine Learning

scikit-learn

`scikit-learn` is Python's main Machine Learning library

- Builds on NumPy, SciPy, and matplotlib
- Plays nicely with Pandas

Check if you have it:

```
  import sklearn
```

`http://scikit-learn.org/stable/install.html` if you don't

Try some things out

```
import pandas
from sklearn.ensemble import RandomForestClassifier as RFC
from sklearn import cross_validation as cv
from sklearn import metrics

iris_data = pandas.read_csv('irisData.csv')
print(iris_data) #remember what this data looks like

(iris_train, iris_test) = cv.train_test_split(iris_data,\
                                test_size = 0.2)
rfc = RFC()

predictors = ["sepal length","sepal width",\
              "petal length","petal width"]
rfc.fit(iris_train[predictors],iris_train["species"])
```

Try some more things out

```
iris_predictions = rfc.predict(iris_test[predictors])

print(iris_predictions)
print(iris_test["species"])

print(metrics.accuracy_score(iris_test["species"],\
                             iris_predictions))
```

Tweaking the learning algorithm parameters

When you create the Random Forest object, you can send optional arguments to tweak the algorithm's parameters.

Try this to change the number of tress from the default of 10 to 100:

```
rfc = RFC(n_estimators=100)
```

Check out the documentation here:
`http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

Exercises

With the aid of the documentation, try the following:

- Does it allow for any early stopping or pruning of decision trees? If so, do it.
- By default, it uses a method other than entropy-based information gain to measure which attribute to split on. Change it so that it uses information gain instead.
- Make it run faster by having it build trees and predict in parallel (simultaneously running on different cores of your CPU).
- Determine which of the attributes (petal length, sepal length, etc.) were most important (i.e., used by more of the trees in the random forest). Hint: the documentation uses the word *features* for what we have been calling *attributes*. Whenever the documentation says "attributes", it means data fields of the Random Forest object.
- Can this be used to do the original single-decision-tree algorithm? Hint: try messing with the `max_features` and `bootstrap` parameters.

Exercise

Check out the `scikit-learn` documentation here:

`http://scikit-learn.org/stable/modules/classes.html`

Find the documentation for the k-Nearest-Neighbor classifier (i.e., the *classifier*, not an unsupervised algorithm). Answer the following questions:

- What is the default value of k it uses?
- Does it do weighted or unweighted k-Nearest-Neighbor by default?
- What is the accuracy on the Iris data set for $k = 100$, both for weighted and unweighted?
- How do the weighted and unweighted versions do for small values of k?

Challenge

I have posted a version of the German Credit dataset called `numeric_german_credit.csv` which has all of the categorical data converted into numbers (don't accidentally use this with Project 2!).

Using scikit-learn's Random Forest and K-Nearest-Neighbor classifiers, find the best-performing set of parameters.