Deconstructing a Random Forest

Abhijit Dasgupta

0.1 Preamble that we load for every session

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

0.2 Developing a random forest regressor

In order to program a random forest, we need to develop several functions that will serve necessary roles in the process. These roles are:

1. Draw a bootstrap sample
2. Determine what observations are Out-of-Bag (OOB)
3. Train multiple decision trees, one on each bootstrap sample
4. Make predictions on new data

0.2.1 Get a bootstrap sample

```
def get_bootstrap(n_obs, n_boot, rng = np.random.RandomState(20)):
    """Summary
    *n_obs*: number of observations in data set
    *n_boot*: number of bootstrap samples to generate
    *rng*: (Optional) seed of random number generator

    RESULTS:
    An array that gives the indices of each bootstrap sample as a column in a
    2-d numpy array.
    """
    indx = np.arange(n_obs)
    boot_indx = rng.choice(indx, size = (n_obs, n_boot), replace=True)
    return(boot_indx)
```

0.2.2 Determining the OOB samples for each bootstrap sample

```
def find_oob(x, n_obs):
    """
    *x*: index of rows that are in a bootstrap sample
```

```
    *n_obs*: Number of observations in the original data

    RESULTS:
    A list with the indices of the OOB sample.
    """
    oob = list(set(range(n_obs)).difference(set(x)))
    return(oob)

def get_oob(boots, n_obs):
    """
    *boots*: A 2-d array that is the output of get_bootstrap
    *n_obs*: Number of observations in the data

    RESULTS:
    A list of OOB indices, one for each bootstrap sample. Note that this is essentially
    a ragged array
    """
    return([find_oob(x, n_obs) for x in boots.T])
```

0.2.3 Building a regression random forest

```
def myRFRegressor(dat, target_var, n_boot = 250,
                  max_features = 5,
                  rng = np.random.RandomState(35)):
    """
    Summary

    dat: A pandas DataFrame object
    target_var: A string denoting the column name of the target variable in dat
    n_boot: number of bootstrap samples to take
    max_features: Maximum number of features to consider at each split
    rng: (Optional) random number seed.

    RESULTS:
    A dictionary containing the trained decision trees as well as the OOB predictions fi
    """
    feature_names = list(dat.columns)
    feature_names.remove(target_var) # Removes the name of the target variable from the
    X, y = dat[feature_names], dat[target_var]
    boot_indx = get_bootstrap(X.shape[0], n_boot, rng=rng) # Generate bootstrap samples
    oob_indx = get_oob(boot_indx, X.shape[0]) # Get OOB samples
    oob_preds = np.zeros_like(boot_indx) - 1 # Storage for OOB predictions. -1 is meant
    baseLearner = DecisionTreeRegressor()
    engines = [] # Storage for fitted decision tree objects
    for i in range(n_boot):
        X_boot = X.iloc[boot_indx[:,i],:]
        y_boot = y[boot_indx[:,i]]
        X_oob = X.iloc[np.array(oob_indx[i]),:]
        baseLearner.fit(X_boot, y_boot)
        engines.append(baseLearner)
        oob_preds[np.array(oob_indx[i]),i] = baseLearner.predict(X_oob)
    oob_preds = pd.DataFrame(oob_preds)
    oob_preds[oob_preds==-1] = np.nan
    predictions = oob_preds.agg(np.nanmean, axis=1)
    return({'engines': engines, 'predictions': predictions})
```

0.2.4 Determining predictions using RF from new data

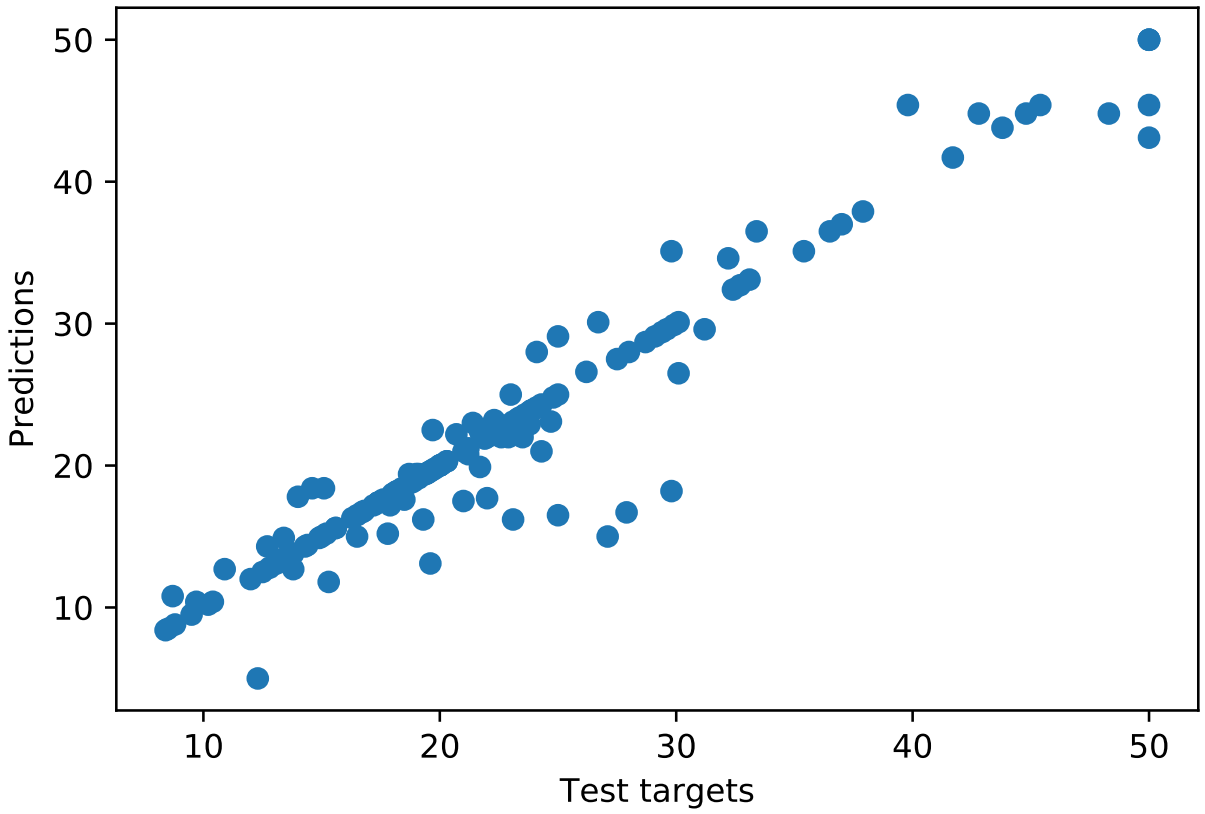
This is the method used to score data using Random Forests

```
def predictRF(engines, test_dat):
    """Summary

    engines: A list of fitted decision trees that will be used as scorers for new data
    test_dat: A test data set
    RESULTS:
    Predictions for each row of test_dat
    """
    prediction = np.zeros(test_dat.shape[0])
    for i in range(test_dat.shape[0]):
        x = test_dat.iloc[i,:]
        preds = [machine.predict(x.values.reshape(1,-1)) for machine in engines]
        prediction[i] = np.mean(np.array(preds))
    return(prediction)
```

0.3 Putting it all together

```
if __name__=='__main__':
    import pandas as pd
    from sklearn.datasets import load_boston
    from sklearn.model_selection import train_test_split
    boston = load_boston()
    X, y = boston['data'], boston['target']
    dat = pd.DataFrame(X, columns = boston['feature_names'])
    dat['target'] = y
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
    myRF = myRFRegressor(dat, 'target')
    p = predictRF(myRF['engines'], pd.DataFrame(X_test))
    plt.scatter(y_test, p)
    plt.xlabel('Test targets')
    plt.ylabel('Predictions');
```


Predictions
Test targets
10
20
30
40
50