Boston Housing Prices

Udacity Machine Learning Engineer
Nanodegree Program: Project 1

Anderson Daniel Trimm

January 29, 2016

Abstract

In this report, we present two analyses using the Boston housing dataset: First, we perform a statistical analysis of the dataset using NumPy. Following this analysis, we optimize a decision tree regression algorithm and use it to predict the value of a house using scikit-learn.

1   Introduction

2   Statistical Analysis of the Boston Housing Dataset

In this section we compute basic statistics of the Boston Housing dataset using NumPy. To begin, we need to import the Boston Housing dataset as well as Numpy:

```
import numpy as np
from sklearn import datasets
```

We can now define the following function to load the Boston dataset:

```
def load_data():
    """Load the Boston dataset."""

    boston = datasets.load_boston()
    return boston
```

After loading the Boston dataset, we can look at its attributes `boston.data` and `boston.target` to access the features and housing prices. The attribute `boston.data` gives a two-dimensional ndarray, where each row is the list of features for a given house. The attribute `boston.target` gives a one-dimensional ndarray of the housing prices. The total number of houses is therefore just the length of the ndarray boston.target:

```
>>> np.shape(boston.target)
(506,)
```

which we see is 506. The number of features per house then is the row length of the ndarray boston.data, which is the one-eth entry of

```
>>> np.shape(boston.data)
(506, 13)
```

which is 13. We can encapsulate these in functions as

```
def size_of_data(city_data):
    number_of_houses = np.shape(city_data.data)[0]
    return number_of_houses

def number_of_features(city_data):
    number_of_features = np.shape(city_data.data)[1]
    return number_of_features
```

To compute the minimum, maximum, mean, and meadian price and the standard deviation, we simply use the methods `np.min(boston)`,`np.max(boston)`, `np.mean(boston)`, `np.meadian(boston)`, and `np.std(boston)`, respectively. As before, we encapsulate these in functions as

```
def get_min_price(city_data):
    min_price = np.min(city_data.target)
    return min_price

def get_max_price(city_data):
    max_price = np.max(city_data.target)
    return max_price

def get_mean_price(city_data):
    mean_price = np.mean(city_data.target)
    return mean_price

def get_median_price(city_data):
    median_price = np.median(city_data.target)
    return median_price

def get_standard_deviation(city_data):
    standard_deviation = np.std(city_data.target)
    return standard_deviation
```