Homework 4

Helen Ngo

November 1, 2016

Problem 1 Use sklearn's implementation of k-Nearest Neighbors for regression purposes. Find the best value of k using 10-fold Cross-Validation (CV).

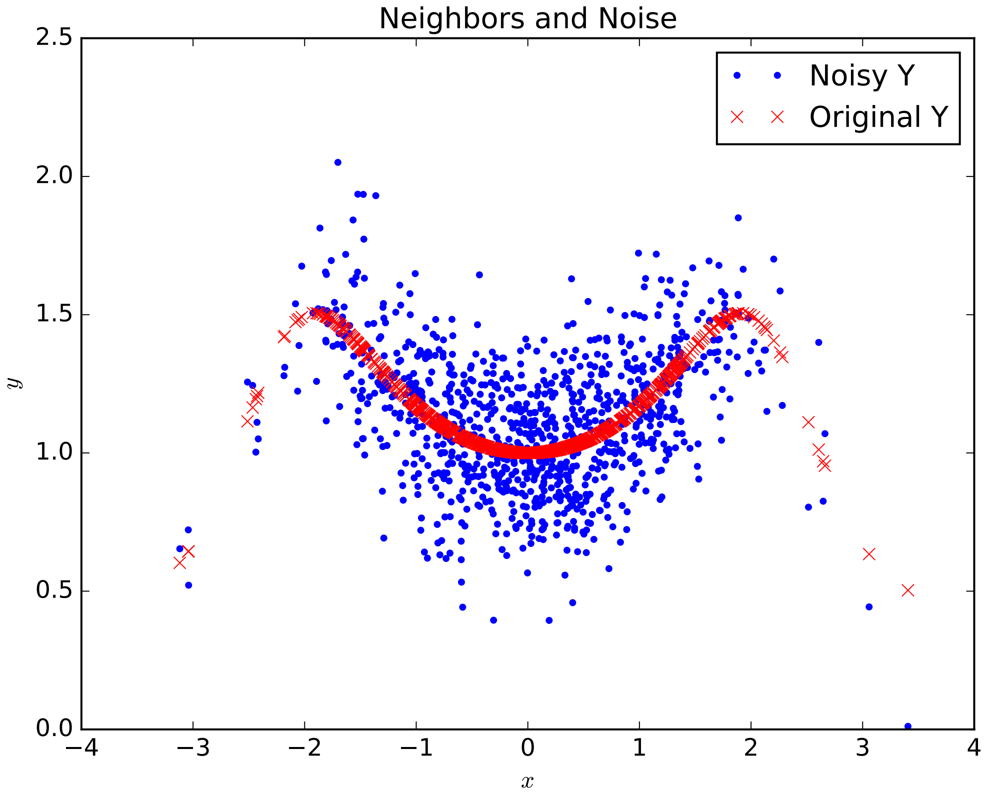
(a) Generate 1000 data points.

(b) Use 10-fold CV to report the three best values of k-neighbors that yield the best CV E_{out}; vary the values of k in the following range: $k = 1, 3, 5, \ldots, 2\left\lfloor \frac{N+1}{2} \right\rfloor - 1$.

(c) Report the best CV E_{out}.

Solution:

(a) The data was generated using the given genDataSet function, modified so that it only returned the input x and target y values. The "Noisy Y" data was used as the target y values:



This can be seen in the python document attached.

(b)
```
        def bestK(self,folds = 10):
                # Reshape dataset for k-neighbors regression
                x = (self.X).reshape(-1, 1)
                y = (self.Y).reshape(-1, 1)
                # Housekeeping
                Eout = []
                N = len(x)
                maxk = int(2*np.floor((N*(folds-1)/folds+1)/2)-1)
                # Output Eout list for k-neighbor range
                for n_neighbors in range(1, maxk, 2):
                        kf = KFold(n_splits = folds)
                        kscore = []
                        for train, test in kf.split(x):
                                x_train, x_test = x[train], x[test]
                                y_train, y_test = y[train], y[test]
                                reg = neighbors.KNeighborsRegressor(
                                        n_neighbors, weights = 'distance')
                                reg.fit(x_train, y_train)
                                kscore.append(reg.score(x_test,
                                        y_test))
                        Eout.append(sum(kscore)/len(kscore))
                return Eout
```

Using the python code above, a list of E_{out} for the given range of k values was found. The index i of the max value for the list of E_{out} was determined, then used to calculate the corresponding k. Note that

k	index
1	0
3	1
5	2
$\vdots$	$\vdots$
$2\left\lfloor \frac{N+1}{2} \right\rfloor - 1$	$\frac{k-1}{2}$

,

therefore the k value can be found from the index using the formula

$$k = 2(i) + 1.$$

The index of the three best values of k-neighbors that yield the best CV E_{out} were found: $k = 289, 296$ and 303. [1]

(c) The best CV $E_{out} = 0.245024051068$ was given by $k = 289$.

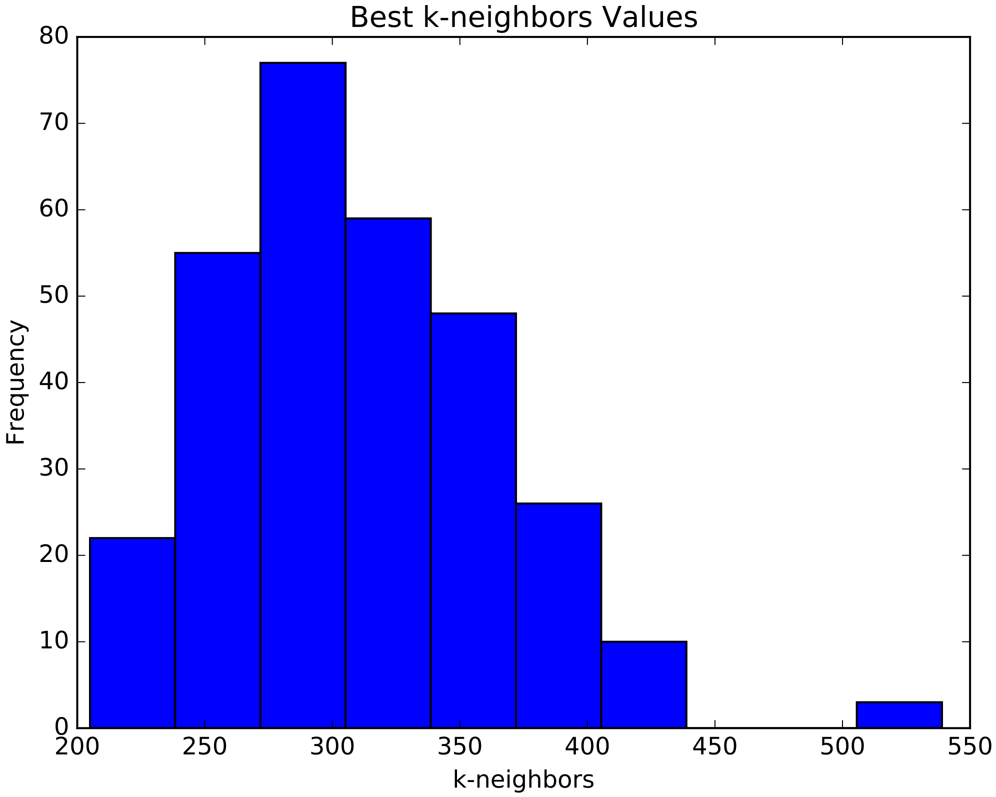
Problem 2 Generate the dataset in the same way as in Problem 1. Repeat the experiment 100 times storing the best three k number of neighbors in every single trial, and at the end of all the trials plot a histogram of all the saved values.

Solution: Commenting out the plotting feature of the genDataSet function, the experiment was repeated a 100 times.

```
def main():
        kvalues = np.empty([300,1]) # vector of best k's
        # Conducting 100 trials
        for trial in range(0,100):
                d = Neighbors(N=1000) # generate data
                Eout = d.bestK() # Eout array
                # Determining best k
                for j in range(0,3):
                        i = Eout.index(max(Eout))
                        k = 2*i+1
                        kvalues[trial*3+j] = k
                        print('k␣=␣', k)
                        print('Eout␣=␣', max(Eout))
                        del Eout[i]
        # Plotting histogram
        plt.hist(kvalues)
        plt.title("Best␣k-neighbors␣Values")
        plt.xlabel("k-neighbors")
        plt.ylabel("Frequency")
        plt.savefig('khistogram.pdf', bbox_inches='tight')
        plt.show()
```

[1] Read footnote(2), these k values were properly adjusted.

The following histogram was generated using the best k values. [2]



[2] The k number of neighbors may be 1 to 2 k values off due to method of finding them, but this will do for examining the trend, such as for a histogram.