Machine Learning Lab Assignment

Nikhil Kapu

17BCD7043

Slot- L39+L40

Here in this Jupyter notebook i have performed KNN and weigted KNN onAuto MPG Data Set available at https://archive.ics.uci.edu/ml/datasets/auto+mpg.

I have used three parameters for comparision which are- Displacement, Horsepower and mpg.

First of all lets load the Training and testing dataset-

In [0]:

```
import pandas

training_data = pandas.read_csv("auto_train.csv")
print(training_data.head())
test_data = pandas.read_csv("auto_test.csv")
print(test_data.head())

x = training_data.iloc[:,:-1]
y = training_data.iloc[:,-1]

x_test = test_data.iloc[:,:-1]
y_test = test_data.iloc[:,-1]
```

```
   displacement  horsepower   mpg
0         307.0         130  18.0
1         350.0         165  15.0
2         318.0         150  18.0
3         304.0         150  16.0
4         302.0         140  17.0
   displacement  horsepower   mpg
0            89          71  31.9
1            86          65  34.1
2            98          80  35.7
3           121          80  27.4
4           183          77  25.4
```

k-NN

Implemented k Nearest Neighbor from scratch. Using the data in the training set, predicted the output for each example in the test, for k = 1, k = 3, and k = 20. Reported the squared error Err on the test set.

In [0]:

```
from kNN import kNN
from sklearn.metrics import mean_squared_error

for k in [1, 3, 20]:
    classifier = kNN(x,y, k)
    pred_test = classifier.predict(x_test)

    test_error = mean_squared_error(y_test, pred_test)
    print("Test error with k={}: {}".format(k, test_error * len(y_test)/2))
```

```
Test error with k=1: 2868.0049999999997
Test error with k=3: 2794.729999999999
Test error with k=20: 2746.1914125
```

Weighted k-NN

Instead of computing an average of the k neighbors, I computed a weighted average of the neighbors using a gaussian function to retrieve the weight for each neighbor.

In [0]:

```python
from kNN import kNN

for k in [1, 3, 20]:
    classifier = kNN(x,y, k, weighted=True)
    pred_test = classifier.predict(x_test)

    test_error = mean_squared_error(y_test, pred_test)
    print("Test error with k={}: {}".format(k, test_error * len(y_test)/2))
```

```
Test error with k=1: 2868.005
Test error with k=3: 2757.3065023859417
Test error with k=20: 2737.9437262401907
```

So based on these three parameters its clear that weighted KNN is more accurate as it has less error by a measure of 2746-2736 = 10.