ML_homework1-v2

October 11, 2017

1 Machine Learning Homework 1

Ivan Puhachov

1.1 Problem

Read a file of training set - comma separated values (.csv) - as integers.
Train a model for classification problem.
Make a prediction to given set of queries.

1.1.1 How it works

I implemented few basics models for classification problem - baseline (most common outcome), Nearest Neighbour, K-Nearest Neighbours. The output file is generated by 5-Nearest Neighbours Methods, as it shown the best performance.

I will update this file with new technics.

1.1.2 Data

I used data from Wholesale Customers Data Set (http://archive.ics.uci.edu/ml/datasets/Wholesale+customers) splitted in Training Set and Testing Set (Query Set) (300 rows and 140 rows).

Query data: Channel (last column) - binary (1 or 2, details in description of the data set).

The only edits I made on data is shifting Channel value to last column in table (instead of first column in original file) and removig headers.

Files: * TS.csv - Training Set * QS.csv - Query Set * QS_answ.csv - Query Set with true outcome (in last column) - to obtain accuracy of prediction Absence of "QS_answ" file will not cause any problem in algorithms work, it is using only for validation and fancy graphs.

1.1.3 Algorithms

- **Baseline**: Predict the most common outcome from TrainingSet
- **Nearest Neighbour**: Find the nearest item from TrainingSet and predict the same output
- **K-Nearest Neighbours**: Find K (fixed value) nearest items from TrainingSet and predict average output (most common) For calculating distances between objects I'm using different metrics: Euclidean, Taxicab, Discrete

Important: I did not test this program on wrong data (unexpected symbols, errors, missing values etc.) so crashes may occur under these conditions.

1.1.4 Imports

```
In [1]: import numpy as np
        import csv
        import matplotlib.pyplot as plt
        from collections import Counter
```

1.2 Inputs

```
In [2]: filename = 'TS.csv'
        with open(filename) as f:
            reader = csv.reader(f)
            #next(reader, None) #to skip headers
            TrainSet = np.array(list(reader)).astype(int)

        filename = 'QS.csv'
        with open(filename) as f:
            reader = csv.reader(f)
            #next(reader, None) #to skip headers
            TestSet = np.array(list(reader))
            TestSet[TestSet == '?'] = 0
            TestSet = TestSet.astype(int)

        (N, K) = TrainSet.shape
        print (N,K)

        #Column with variable we are trying to predict
        Query = K-1 #Last column
        print(TestSet)
```

```
300 8
[[    2 16448  6243 ...,  2662  2005     0]
 [    2  5283 13316 ...,  8752   172     0]
 [    2  2886  5302 ...,  6236   555     0]
 ...,
 [    3 14531 15488 ..., 14841  1867     0]
 [    3 10290  1981 ...,   168  2125     0]
 [    3  2787  1698 ...,   477    52     0]]
```

1.3 Baseline model

The simplest model is to predict the most common outcome.

Statistics of outputs:

```
In [3]: TrainOutputs = [i[Query] for i in TrainSet]
        Counter(TrainOutputs).most_common()
```

```
Out[3]: [(1, 197), (2, 103)]
```

```
In [4]: baseline = Counter(TrainOutputs).most_common(1)[0][0]
        predictionBaseline = np.array([baseline for i in TestSet])
        print(predictionBaseline)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

1.4 Nearest Neighbour Methods

Description: for every new query point Q find the nearest data point from Train Set (or set of K such points). Prediction for Q is the outcome value of nearest point (or the most common output of K nearest points).

For calculating distance between datapoints we need a measure.

Consider the following metrics: 1. Euclidean 2. Taxicab 3. Discrete

```
In [5]: def measureDistEuclid(a,b):
            dist = 0;
            for i in range(K):
                #measure distance only on "input" values
                if (i!=Query):
                # to prevent overflowing of numpy int type
                    dist += (np.asscalar(a[i]-b[i].item()))**2
            return np.sqrt(dist)

        def measureDistTaxi(a,b):
            dist = 0;
            for i in range(K):
                if (i!=Query):
                    dist += np.absolute(a[i]-b[i])
            return dist

        def measureDistDiscrete(a,b):
            dist = 0;
            for i in range(K):
                if (i!=Query):
                    dist += (a[i]==b[i])
            return dist
```

Implementing methods

```
In [6]: def nearestNeighbour(q,measureDist):
            minDist = measureDist(q,TrainSet[0])
```

```
            minDistValue = TrainSet[0][Query]
            for i in TrainSet:
                d = measureDist(q,i);
                if (d<minDist):
                    minDist = d
                    minDistValue = i[Query]
            return minDistValue

        def KnearestNeighbour(q,k,measureDist):
            distances = [{'dist':measureDist(i,q), 'Query':i[Query]} for i in TrainSet]
            distances = sorted(distances, key = lambda x: x['dist'])
            neighbours = [distances[i]['Query'] for i in range(k)]
            return Counter(neighbours).most_common(1)[0][0]
```

Predicting

```
In [7]: predictionEuclid = np.array([nearestNeighbour(i,measureDistEuclid) for i in TestSet])
        predictionTaxi = np.array([nearestNeighbour(i,measureDistTaxi) for i in TestSet])
        predictionDiscrete = np.array([nearestNeighbour(i,measureDistDiscrete) for i in TestSet]
```

1.5 Output

As will be shown after, K-Nearest Neighbours method witk K=4 and Euclid distance shows the best accuracy on testing set. This will be the output.

```
In [8]: for i in TestSet:
            i[Query] = KnearestNeighbour(i,4,measureDistEuclid)

        np.savetxt("OUT.csv",TestSet, delimiter=',', fmt='%d')
        print(TestSet)
```

```
[[    2 16448  6243 ...,  2662  2005     2]
 [    2  5283 13316 ...,  8752   172     2]
 [    2  2886  5302 ...,  6236   555     2]
 ...,
 [    3 14531 15488 ..., 14841  1867     2]
 [    3 10290  1981 ...,   168  2125     1]
 [    3  2787  1698 ...,   477    52     1]]
```

1.6 Comparison

Additional part.

1.6.1 Requesting true outcome

```
In [9]: filename = 'QS_answ.csv'
        with open(filename) as f:
            reader = csv.reader(f)
            #next(reader, None) #to skip headers
            TestSetANSW = np.array(list(reader)).astype(int)

        truePrediction = np.array([i[Query] for i in TestSetANSW])
```

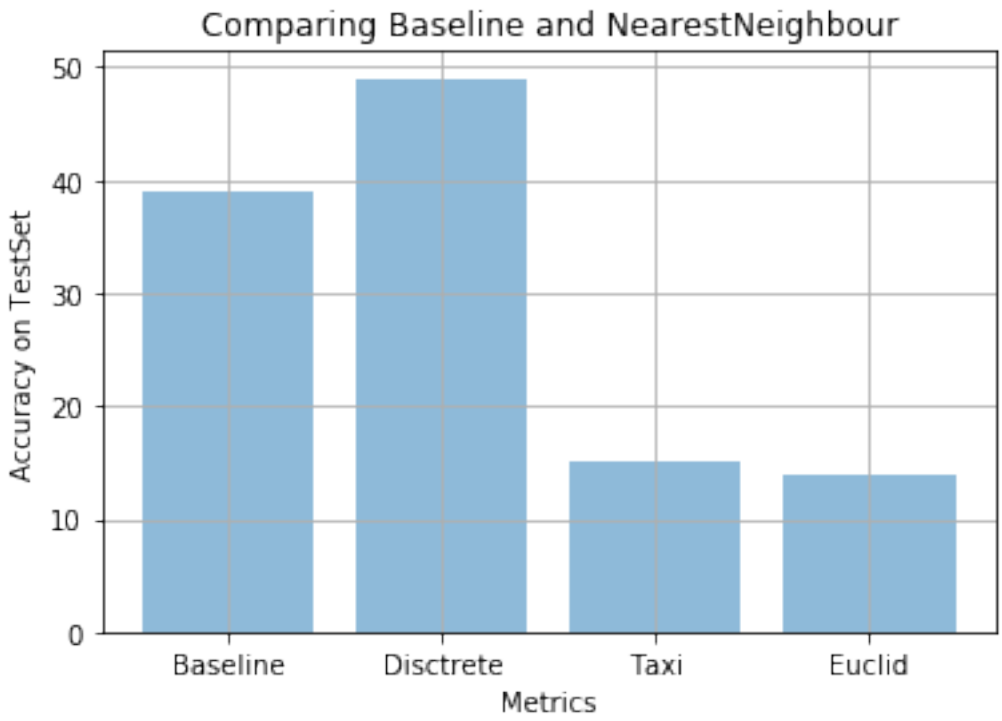
```
In [10]: def accuracy(pred):
             err = 0;
             for i in range(truePrediction.shape[0]):
                 if (truePrediction[i]!=pred[i]):
                     err+=1
             return err
```

```
In [11]: print(accuracy(predictionBaseline))
         print(accuracy(predictionEuclid))
         print(accuracy(predictionTaxi))
         print(accuracy(predictionDiscrete))
```

```
39
14
15
49
```

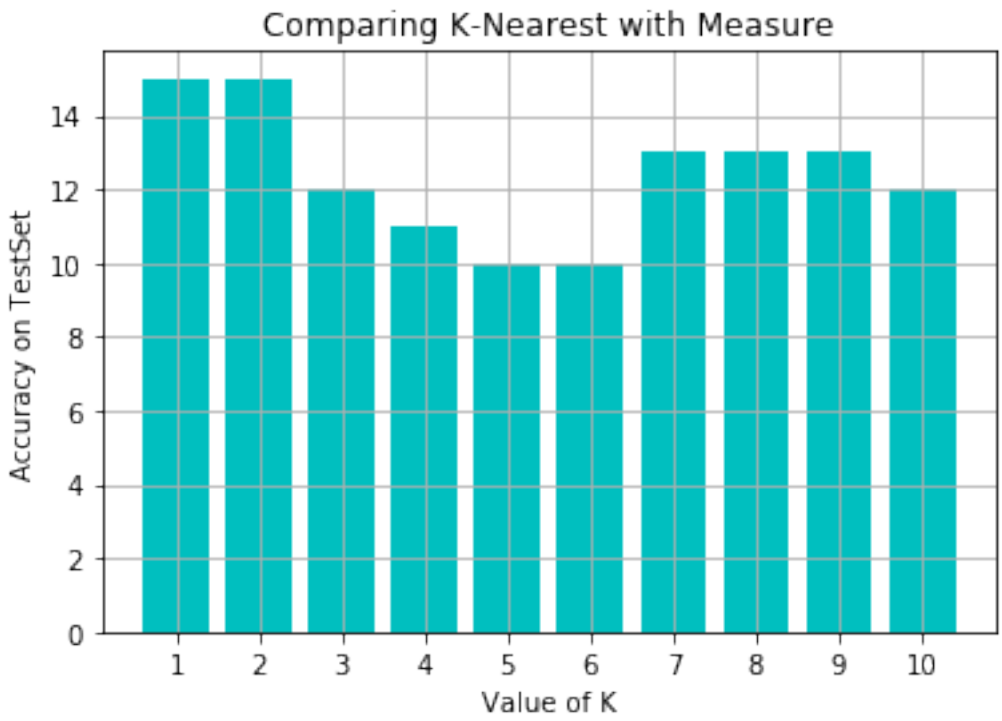
1.6.2 Graphs

```
In [12]: performance = [accuracy(predictionBaseline), accuracy(predictionDiscrete), accuracy(pre
         plt.bar(np.arange(len(performance)), performance, align='center', alpha=0.5)
         plt.xticks(np.arange(len(performance)), ['Baseline','Disctrete', 'Taxi', 'Euclid'])
         plt.ylabel('Accuracy on TestSet')
         plt.xlabel('Metrics')
         plt.title('Comparing Baseline and NearestNeighbour')
         plt.grid(True)
         plt.show()
```



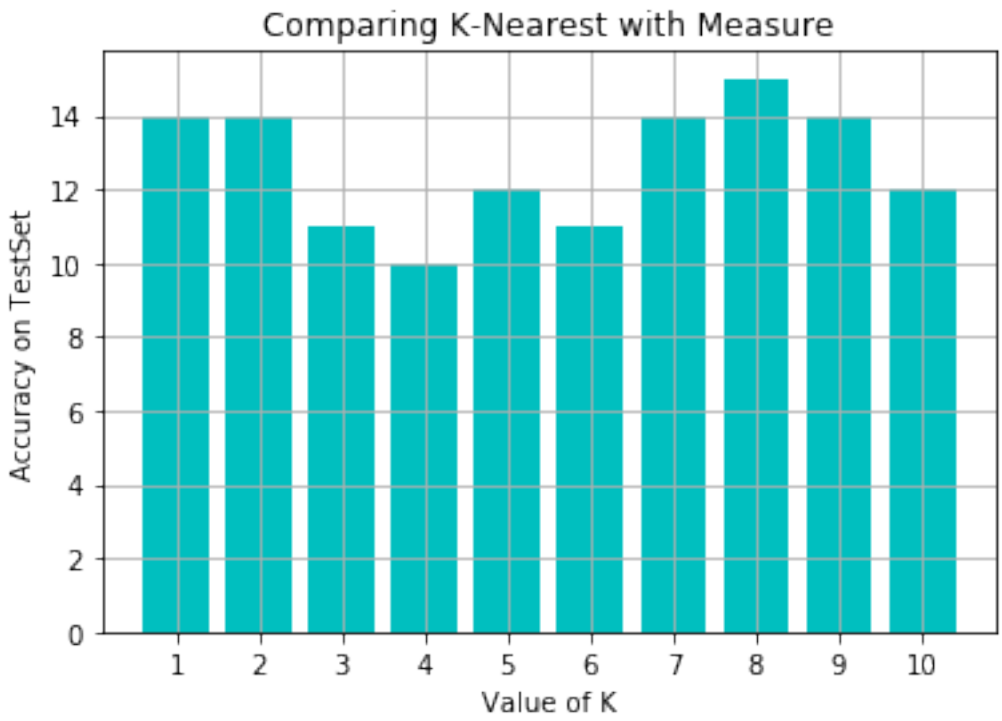
Comparing K-Nearest methods, changing K and Measure

```
In [13]: def predictionKNearest(k,measureDist):
             return np.array([KnearestNeighbour(i,k,measureDist) for i in TestSet])

In [14]: Kmax = 10

In [15]: def plotPerfKNeigh(measureDist):
             performance = [accuracy(predictionKNearest(i,measureDist)) for i in range(1,Kmax+1)
             plt.bar(np.arange(len(performance)), performance, align='center', alpha=1, color =
             plt.xticks(np.arange(len(performance)), range(1,Kmax+1))
             plt.title('Comparing K-Nearest with Measure')
             plt.ylabel('Accuracy on TestSet')
             plt.xlabel('Value of K')
             plt.grid(True)
             plt.show()

In [16]: plotPerfKNeigh(measureDistTaxi)
         print("TaxiCab Distance")
```



TaxiCab Distance

```
In [17]: plotPerfKNeigh(measureDistEuclid)
         print("Euclid Distance")
```

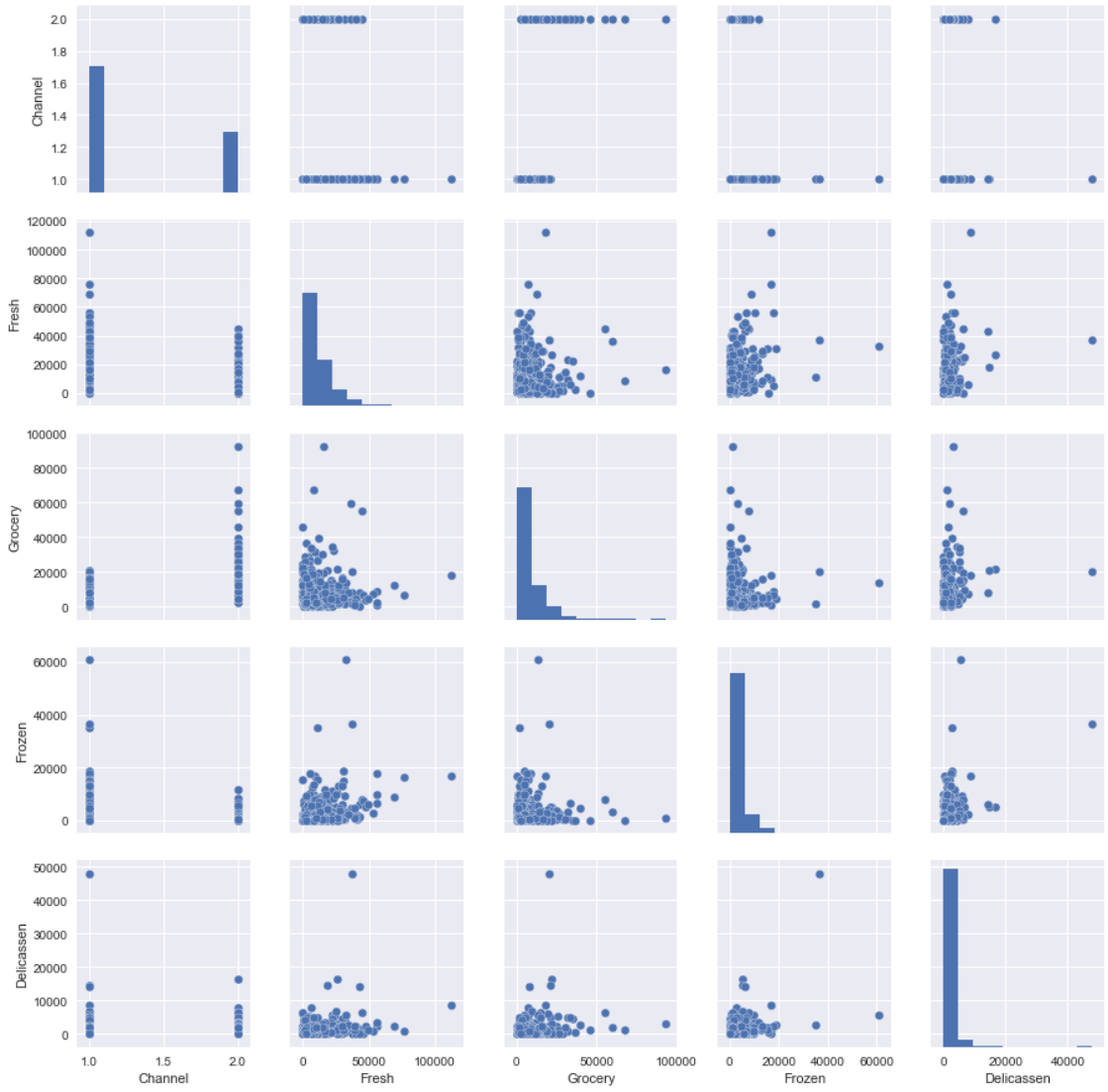



```
Euclid Distance
```

1.6.3 Plots about the data

(Under construction)

```
In [18]: import seaborn as sns
         import pandas as pd

In [19]: filename = 'Wholesale.csv'
         data = pd.read_csv(filename)
         data.columns

Out[19]: Index(['Region', 'Fresh', 'Milk', 'Grocery', 'Frozen', 'Detergents_Paper',
                'Delicassen', 'Channel'],
               dtype='object')

In [20]: sns.set()
         cols = ['Channel', 'Fresh', 'Grocery', 'Frozen','Delicassen']
         sns.pairplot(data[cols], size=2.5)
         plt.show()
```


Channel
Fresh
Grocery
Frozen
Delicassen
2.0
1.8
1.6
1.4
1.2
1.0
120000
100000
80000
60000
40000
20000
0
50000
60000
30000
10000
1.0
1.5
2.0
0
50000
100000
0
20000
40000
60000
0
20000
40000