Machine Learning, Fall 2019: Project 1

Zhibo Sheng

Header: **I use Python, and use pandas to read csv file, use numpy to manipulate the matrix, and use matplotlib to do data visualizations.**

Datasets: The project will explore two datasets, the famous MNIST dataset of very small pictures of handwritten numbers, and a dataset that explores the prevelance of diabetes in a native american tribe named the Pima. You can access the datasets here:

1. https://www.kaggle.com/uciml/pima-indians-diabetes-database
2. https://www.kaggle.com/c/digit-recognizer/data

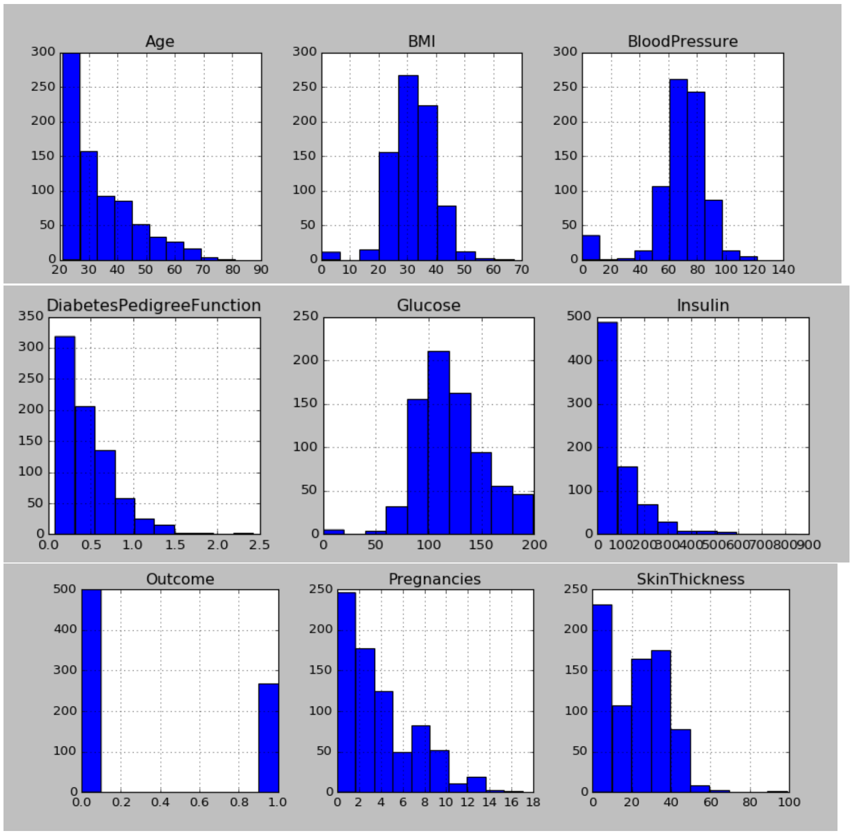
Programming Task: For each dataset, you must create a K-NN classifier that uses the training data to build a classifier, and evaluate and report on the classifier performance.

(30 points) Dataset details:

1. The pima dataset has 768 samples, it has 8 feature vector(Pregnancies, Glucose, BloodPressure SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age) and 1 label (Outcome). However, it has several missing data in the csv file. We can see the distribution of the data below.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin
count	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479
std	3.369578	31.972618	19.355807	15.952218	115.244002
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000
75%	6.000000	140.250000	80.000000	32.000000	127.250000
max	17.000000	199.000000	122.000000	99.000000	846.000000

	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000
mean	31.992578	0.471876	33.240885	0.348958
std	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.078000	21.000000	0.000000
25%	27.300000	0.243750	24.000000	0.000000
50%	32.000000	0.372500	29.000000	0.000000
75%	36.600000	0.626250	41.000000	1.000000
max	67.100000	2.420000	81.000000	1.000000



(15 points) Algorithm Description For the pima dataset, as we can see below, there are some missing data in the csv file, especially for SkinThickness and Insulin, so I choose to replace the data with mean(Glucose, BloodPressure , BMI) and median (SkinThickness, Insulin) and I normalize the feature vector to 0-1. And I split 80% data of dataset as training data and 20% data of dataset as test data.

```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

(45 points) Algorithm Results:

1) Manhattan Distance

For the Manhattan Distance, I iterate K from 1 to 5. As we can see in the picture below, when k = 3, we can get the best result. So I choose k = 3 as the default.

```
[0.6009771986970684, 0.6726384364820847, 0.6579804560260586, 0.6856677524429967, 0.6840390879478827]
[0.6363636363636364, 0.6753246753246753, 0.6818181818181818, 0.6753246753246753, 0.6493506493506493]
```

For the accuracy, we can see the picture that, the true positive is 80, the true negatives is 25, the false positive is 20 and the false negative is 29.

```
[[80 20]
 [29 25]]
```

2) Euclidean Distance

For the Euclidean Distance, I iterate K from 1 to 5. As we can see in the picture below, when k = 4, we can get the best result. So I choose k = 4 as the default.

```
[0.6205211726384365, 0.6726384364820847, 0.6677524429967426, 0.6840390879478827, 0.6840390879478827]
[0.6428571428571429, 0.6623376623376623, 0.6493506493506493, 0.6883116883116883, 0.6428571428571429]
```

For the accuracy, we can see the picture that, the true positive is 88, the true negatives is 18, the false positive is 12 and the false negative is 36.

```
[[88 12]
 [36 18]]
```

3) Chebychev Distance

For the Chebychev Distance, I iterate K from 1 to 5. As we can see in the picture below, when k = 2, we can get the best result. So I choose k = 2 as the default.

```
[0.6302931596091205, 0.6661237785016286, 0.6547231270358306, 0.6889250814332247, 0.6840390879478827]
[0.6038961038961039, 0.6558441558441559, 0.5844155844155844, 0.6298701298701299, 0.6233766233766234]
```

For the accuracy, we can see the picture that, the true positive is 86, the true negatives is 15, the false positive is 14 and the false negative is 39

```
[[86 14]
 [39 15]]
```

.

(10 points) Runtime: Describe the run-time of your algorithm and also share the actual ”wall-clock” time that it took to compute your results.

For the KNN with Manhattan Distance, it take 0.0812s to finish the task.

For the KNN with Euclidean Distance, it take 0.1185s to finish the task.

For the KNN with Chebychev Distance, it take 0.070s to finish the task.

Machine Learning, Fall 2019: Project 1

Zhibo Sheng

Header: **I use Python, and use pandas to read csv file, use numpy to manipulate the matrix, and use matplotlib to do data visualizations.**

Datasets: The project will explore two datasets, the famous MNIST dataset of very small pictures of handwritten numbers, and a dataset that explores the prevelance of diabetes in a native american tribe named the Pima. You can access the datasets here:

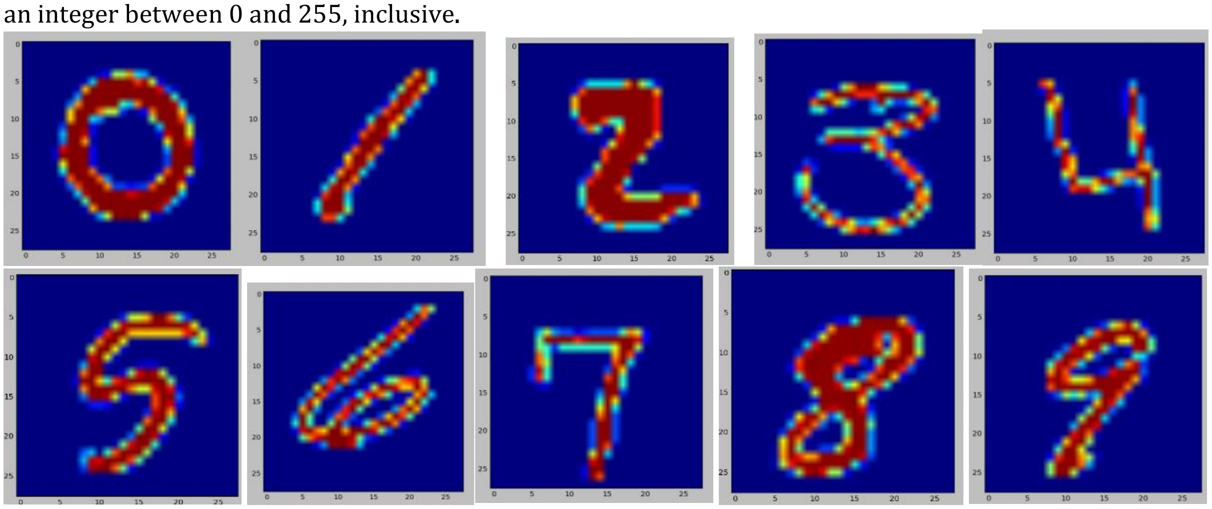
3. https://www.kaggle.com/uciml/pima-indians-diabetes-database

4. https://www.kaggle.com/c/digit-recognizer/data

Programming Task: For each dataset, you must create a K-NN classifier that uses the training data to build a classifier, and evaluate and report on the classifier performance.

(30 points) Dataset details:

The MNIST dataset has 42000 sample for training set and 28000 sample for test set. And it has 10 category. As we can see below, there are the example for each category. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.



(15 points) Algorithm Description For the MNIST dataset, we normalize the data from 0 to 255 to 0 to 1. And we reshape the data demenation from [1,784] to [28,28]. And for training set, it has label, so we should divide the label from the traing set.

(45 points) Algorithm Results:

When K == 1, the confusion matrix is show as below, and the total accuracy is 0.97

```
[[803   0   0   1   1   1   5   0   1   1]
 [  0 952   3   1   0   0   1   2   2   0]
 [  7   2 836   2   1   0   2   7   3   0]
 [  0   0   3 832   0  14   1   6   3   4]
 [  1   7   0   0 792   0   3   3   0  21]
 [  3   1   0  11   1 730   8   0   1   1]
 [  1   1   0   0   1   2 836   0   0   0]
 [  0   7   5   1   3   0   0 877   0   6]
 [  0   5   3  13   2   8   3   2 726   6]
 [  4   2   0   2  13   2   1  16   0 772]]
```

When K == 2, the confusion matrix is show as below, and the total accuracy is 0.96

```
[[808   0   2   0   0   1   2   0   0   0]
 [  0 956   3   0   0   0   0   1   1   0]
 [  9   8 835   1   0   0   0   5   2   0]
 [  2   0   9 842   0   3   0   3   1   3]
 [  1   7   0   0 806   0   2   1   0  10]
 [  4   1   0  26   2 721   2   0   0   0]
 [  1   1   0   0   3   3 833   0   0   0]
 [  0   8   6   1   6   0   0 875   0   3]
 [  3  10   3  26   4  26   5   3 684   4]
 [  5   3   1   4  23   3   2  29   1 741]]
```

When K == 3, the confusion matrix is show as below, and the total accuracy is 0.97

```
[[805   0   1   0   1   2   3   0   1   0]
 [  0 957   1   1   0   0   0   1   1   0]
 [  9   6 831   1   0   0   2   9   2   0]
 [  1   0   3 842   0   5   0   4   4   4]
 [  1   6   0   0 788   0   4   1   0  27]
 [  1   0   0   7   1 738   8   0   0   1]
 [  0   1   0   0   2   2 836   0   0   0]
 [  0   9   5   0   2   0   0 876   0   7]
 [  4   9   1  16   2  17   4   0 708   7]
 [  4   2   1   4  11   3   1  21   0 765]]
```

When K == 4, the confusion matrix is show as below, and the total accuracy is 0.97

```
[[805   0   2   0   0   2   4   0   0   0]
 [  0 956   2   1   0   0   0   1   1   0]
 [ 10   5 835   1   0   0   0   8   1   0]
 [  1   1   4 838   0   4   0   5   5   5]
 [  1   8   0   0 799   0   2   0   0  17]
 [  1   0   0  14   0 733   6   0   0   2]
 [  0   1   0   0   2   4 834   0   0   0]
 [  0  10   4   1   5   0   0 874   0   5]
 [  2   9   1  18   4  23   6   2 696   7]
 [  5   2   1   3  12   4   2  25   0 758]]
```

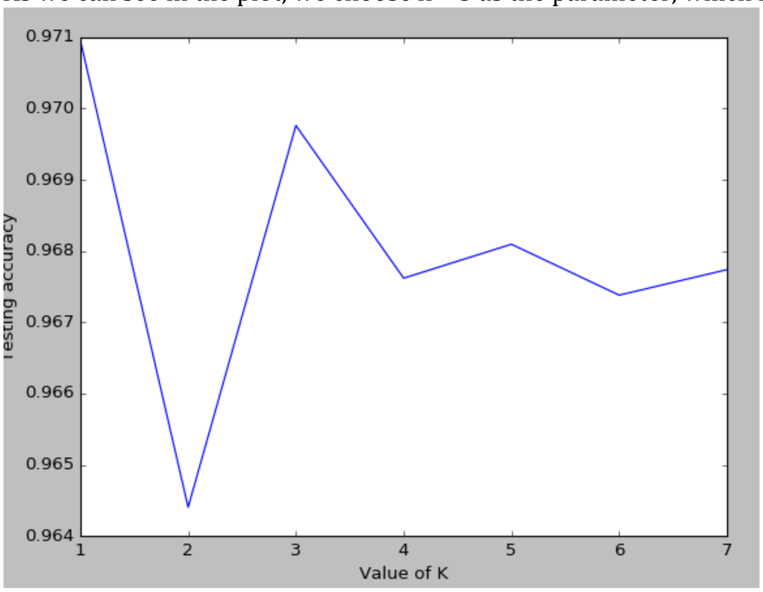
When K == 5, the confusion matrix is show as below, and the total accuracy is 0.97

```
[[806   0   2   0   0   2   3   0   0   0]
 [  0 956   2   1   0   0   0   1   1   0]
 [  9   6 829   3   0   0   1   9   3   0]
 [  1   1   4 836   0   6   0   6   4   5]
 [  1   7   0   0 795   0   4   1   0  19]
 [  2   1   1   8   0 734   7   0   1   2]
 [  0   1   0   0   1   2 837   0   0   0]
 [  0   9   4   1   4   0   0 874   0   7]
 [  3   8   0  16   4  20   8   3 699   7]
 [  5   2   1   5   9   3   2  19   0 766]]
```

When K == 6, the confusion matrix is show as below, and the total accuracy is 0.97

```
[[805   0   2   0   0   2   4   0   0   0]
 [  0 956   3   0   0   0   0   1   1   0]
 [ 10   6 830   3   0   0   1   8   2   0]
 [  1   1   3 839   0   4   0   7   4   4]
 [  1   7   0   0 801   0   3   0   0  15]
 [  1   1   0  12   1 731   7   0   1   2]
 [  0   1   0   0   1   4 835   0   0   0]
 [  0  11   3   1   2   0   0 873   0   9]
 [  5  12   0  19   4  18   6   2 695   7]
 [  4   2   1   4  12   4   2  22   0 761]]
```

As we can see in the plot, we choose k = 3 as the parameter, which has the best result in the testset



(10 points) Runtime: Describe the run-time of your algorithm and also share the actual ”wall-clock” time that it took to compute your results.

When k = 3 , the algorithm takes 379s to complete the task.