1. For k = 1, which examples were not correctly classified?

	index	sepal length[cm]	petal width	true_label	predicted_label
0	10	6.1	1.4	2	1
1	11	6.4	1.5	1	2
2	16	6.0	1.5	1	2

2. Report the accuracy on the test set for k = 1.

```
Accuracy for k = 1: 0.9210526315789473
```

3. For k = 3, which examples were not correctly classified?

	index	sepal length[cm]	petal width	true_label	predicted_label
0	10	6.1	1.4	2	1

4. Report the accuracy on the test set for k = 3.

```
Accuracy for k = 3: 0.9736842105263158
```

5. For k = 5, which examples were not correctly classified?

	index	sepal length[cm]	petal width	true_label	predicted_label
0	10	6.1	1.4	2	1
1	37	6.0	1.6	1	2

6. Report the accuracy on the test set for k = 5.

```
Accuracy for k = 5: 0.9473684210526315
```

7. Suppose we used the very simple Zero-R classifier on this dataset, rather than k-NN. That is, we classify all examples in the test set as belonging to the class that is most common in the training set. What is the resulting accuracy?

```
The predicted label of Zero-R classifier is:  2
The accuracy of Zero-R classifier is:  0.23684210526315788
```

8. Choose a new distance function for your k-NN algorithm.

```python
# change distance function
def new_distance(x1, x2):
    d = 0.
    for i in range(len(x1)):
        d += (abs(x1[i]-x2[i]))**1
    # print(d)
    return d
```

Manhattan-distance

You can choose whatever distance function you like, but should choose something that you think might yield higher accuracy than the Euclidean distance function.

Run k-NN with your distance function, using the same training and test sets, to classify the examples in the test set.

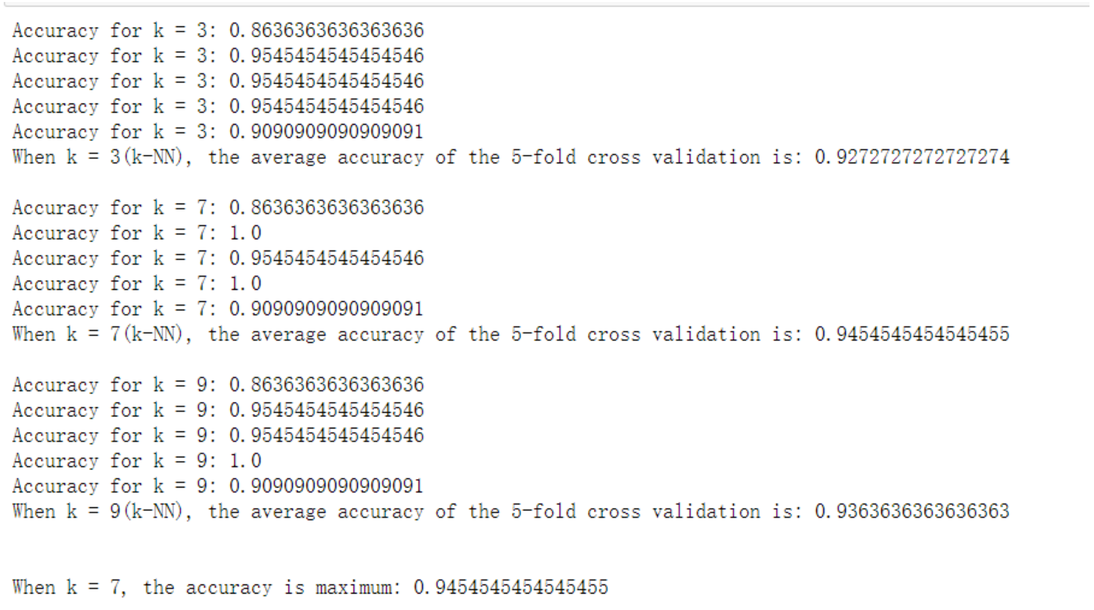
Did your distance function achieve higher accuracy (for k = 1 , k = 3, and k = 5) than the first distance function? If it didn't, what is a possible reason that it didn't?

The result is same with the result of Euclidean distance. Possible reason could be the misclassified data are actually noise data. Even if you use all the feature, you still get the same result.

9. (extra credit) Implement 5-fold cross-validation on the training set to determine which of the following values of k works better in k-NN: 3; 7; 9.

More particularly, to implement the 5-fold cross-validation, divide the training set into 5 sets of equal size. In practice you may want to randomly permute the data before dividing it into 5 sets, but for this assignment, just take the first 1=5 of the examples listed in the _le, then the second 1=5, etc. (and DON'T PERMUTE) the examples first.

Then for each of the 3 values of k, do the following. (1) For each of the 5 sets, train on the examples in the other four sets, and test on the examples in the 5th set. The result is that a prediction has been made on each example in the training set. (2) Calculate the percentage of these predictions that were correct. This is the cross-validation accuracy (for this value of k).



```
Accuracy for k = 3: 0.8636363636363636
Accuracy for k = 3: 0.9545454545454546
Accuracy for k = 3: 0.9545454545454546
Accuracy for k = 3: 0.9545454545454546
Accuracy for k = 3: 0.9090909090909091
When k = 3(k-NN), the average accuracy of the 5-fold cross validation is: 0.9272727272727274

Accuracy for k = 7: 0.8636363636363636
Accuracy for k = 7: 1.0
Accuracy for k = 7: 0.9545454545454546
Accuracy for k = 7: 1.0
Accuracy for k = 7: 0.9090909090909091
When k = 7(k-NN), the average accuracy of the 5-fold cross validation is: 0.9454545454545455

Accuracy for k = 9: 0.8636363636363636
Accuracy for k = 9: 0.9545454545454546
Accuracy for k = 9: 0.9545454545454546
Accuracy for k = 9: 1.0
Accuracy for k = 9: 0.9090909090909091
When k = 9(k-NN), the average accuracy of the 5-fold cross validation is: 0.9363636363636363


When k = 7, the accuracy is maximum: 0.9454545454545455
```