KNN

Wavelix

1 Requirements

In this assignment, we need to analyse the Breast Cancer Wisconsin (Original) data using K-Nearest Neighbors algorithm. Details:

1. Divide the dataset into a training set and a testing set with a ratio of 7:3.
2. Write a code for the KNN algorithm.
3. Test the algorithm with varying k from 1 to 10 using the testing set, and provide the accuracy for each k. Analyse how the accuracy changes with different values of k.

2 Ideas and Analysis

2.1 First Approach

For the dataset, firstly, we normalize each feature to have a mean of 0 and a standard deviation of 1. Then the target labels are encoded:

- $'M'$ as 1
- $'B'$ as 0

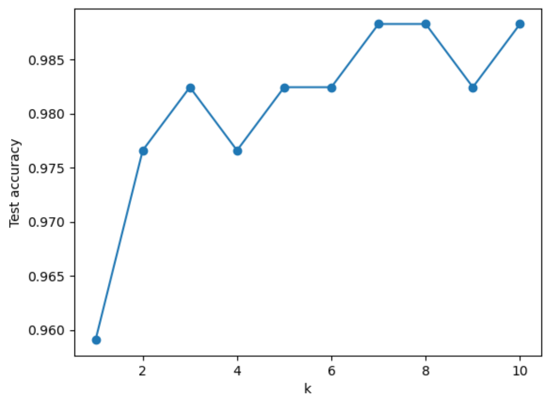
After that, the dataset is split into a training set and a testing set in a ratio of $7:3$ using random indices.

Next, in the function `knn`, we loops over a range of k values from 1 to 10. For each k, build a `KDTree` for efficient neighbor search in the training set. Then, query the k nearest neighbors for each points in the test set, and determine `y_pred` by finding the majority label among the k nearest neighbors. For each k we use a `score` to store the correct number of predictions in the testing set. Accuracy is caculated by $\frac{\texttt{score}}{\text{length of ytest}}$, which is stored in a `scores` list. For each k, we plot its accuracy in one chart.

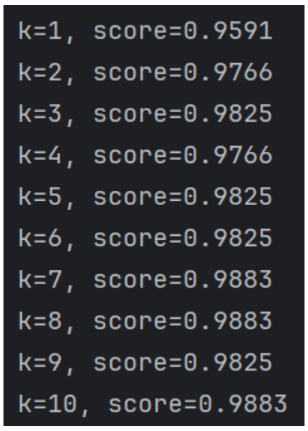
The related code is in `KNN.py`.

2.2 Result and Analysis

We plot the accuracies for each k, however, find that the result varies each time. This may result from the random split of the data into training and testing set, and each run can produce different results due to the random selection of instances in each set.

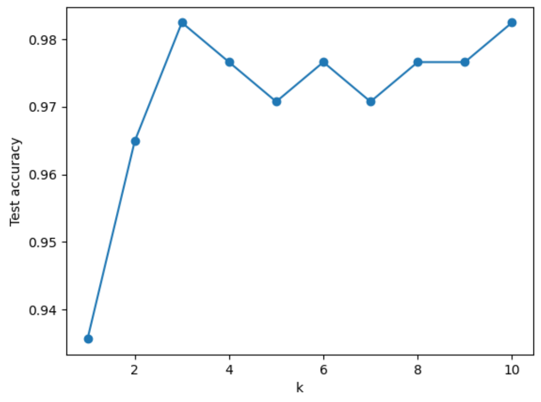


(a) plot

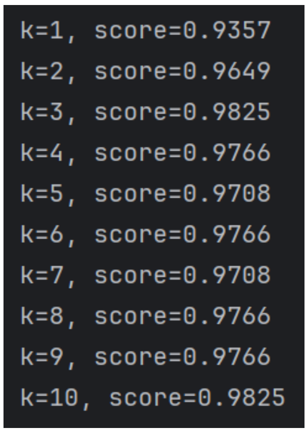


```
k=1, score=0.9591
k=2, score=0.9766
k=3, score=0.9825
k=4, score=0.9766
k=5, score=0.9825
k=6, score=0.9825
k=7, score=0.9883
k=8, score=0.9883
k=9, score=0.9825
k=10, score=0.9883
```

(b) accuracy

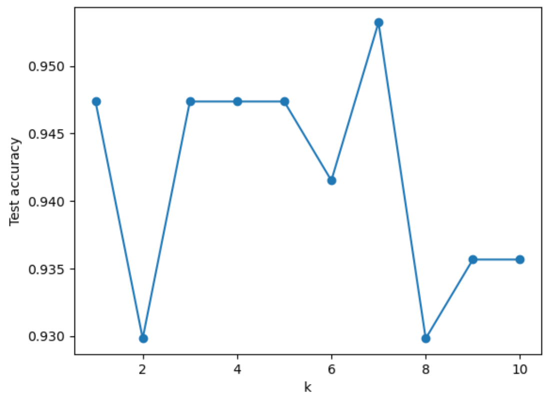


(c) plot

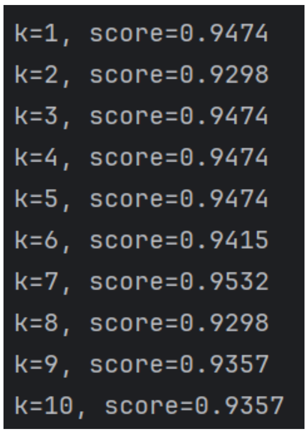


```
k=1, score=0.9357
k=2, score=0.9649
k=3, score=0.9825
k=4, score=0.9766
k=5, score=0.9708
k=6, score=0.9766
k=7, score=0.9708
k=8, score=0.9766
k=9, score=0.9766
k=10, score=0.9825
```

(d) accuracy



(e) plot



```
k=1, score=0.9474
k=2, score=0.9298
k=3, score=0.9474
k=4, score=0.9474
k=5, score=0.9474
k=6, score=0.9415
k=7, score=0.9532
k=8, score=0.9298
k=9, score=0.9357
k=10, score=0.9357
```

(f) accuracy

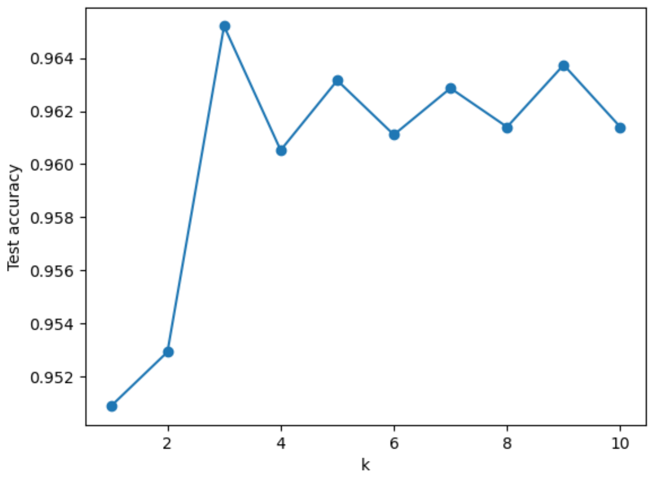
In most cases, the output have the tendency that, the accuracy climbs up and then dropped as k increases. To visualize the tendency, we have the second approach.

2.3 Second Approach and Analysis

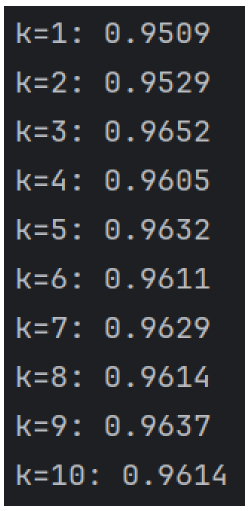
We applied an informal method to look for the tendency. Before the random split, we add a new loop called `repetition`. The final accuracy is caculated by the mean of the scores in each repetition. Typically, this method is used in validation.

The related code is in `KNN_2.py`

Now the tendency is more clear:



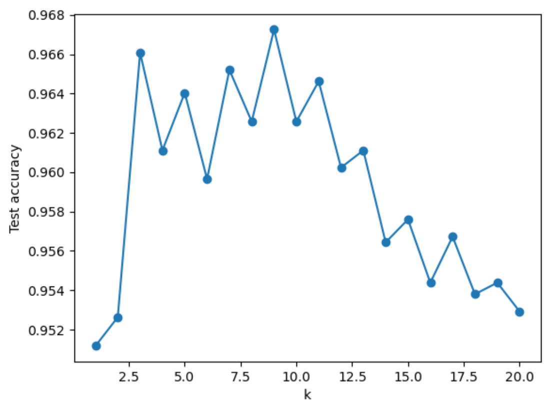
(g) plot



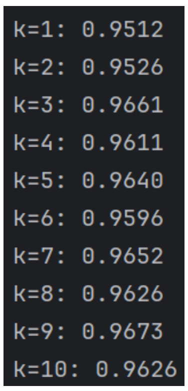
```
k=1: 0.9509
k=2: 0.9529
k=3: 0.9652
k=4: 0.9605
k=5: 0.9632
k=6: 0.9611
k=7: 0.9629
k=8: 0.9614
k=9: 0.9637
k=10: 0.9614
```

(h) accuracy

The accuracy climbs up as k increases, and when k is larger than 3, it seems that there is no obvious changes on accuracy. Then, we re-test with varying k from 1 to 20:

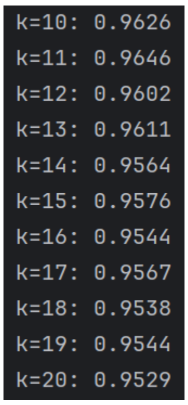


(i) plot



```
k=1: 0.9512
k=2: 0.9526
k=3: 0.9661
k=4: 0.9611
k=5: 0.9640
k=6: 0.9596
k=7: 0.9652
k=8: 0.9626
k=9: 0.9673
k=10: 0.9626
```

(j) accuracy



```
k=10: 0.9626
k=11: 0.9646
k=12: 0.9602
k=13: 0.9611
k=14: 0.9564
k=15: 0.9576
k=16: 0.9544
k=17: 0.9567
k=18: 0.9538
k=19: 0.9544
k=20: 0.9529
```

(k) accuracy

It is clear that as k increases, the accuracy climbs up first and then drops down.

When k is small, the model is highly sensitive to local changes in the training data. A single noisy can dominate the prediction. When k is large, the model looks almost every data rather than focusing on local neighborhoods, which loses sensitivity to local features. When we choose a moderate k, the algorithm consider a proper range of neighbors, making predictions perform better.