**Header:**

My Name: Qi Zhao

Major resources: A laptop, java libraries, Pima Indians Diabetes Database (Kaggle).

Programming Language: java.

**Dataset details:**

The total 768 examples in this dataset are divided into three parts.

Training set: 201 ~ 768 ( About 74%);

Validation Set: 1~100(13%);

Test Set: 101~200 (13%);

**Algorithm description:**

K-NN algorithms.

Feature scaling: I calculate a mediate value for each feature. mediate value = (max + min)/2.

For each feature: the scaling value = original value/mediate value; Such that all the data

are in the same order.

Tips: For the features except for “pregnancies”:

First, ignore the “0” value during the process in which we calculate the mediate value of that

feature;

Second, replace the “0” value with the mediate value during the process in which all the feature

values are scaled.

**Algorithm results:**

Confusion matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | (actual)Non-diabetes | (actual)diabetes | total |
| (predict)Non-diabetes | TN=58 | FN=26 | 84 |
| (predict)diabetes | FP=4 | TP=12 | 16 |
| total | 62 | 38 | 100 |

Accuracy: (TP + TN)/total = 70%;

Misclassification Rate: (FP + FN)/total = 30%;

True Positive Rate (sensitivity): TP/(TP + FN) = 32%

False Positive Rate: FP/(TN + FP) = 6.5%;

Specificity: TN/(TN + FP) = 93.5%;

Precision: TP/(FP + TP) = 75%;

Prevalence: (FN + TP)/total = 38%;

**Runtime**

**Analysis:**

We assume there are m training examples, and there are n features for each example.

First, convert the dataset from line to line to an arraylist that store the “example” objects and scale them. O (m \* n).

Then, for each test case, iterate the whole arraylist and find the K nearest point. O(m\*n logk).

Finally, we assume the amount of test case is t;

So, the total time complexity is O(t \* m \* n \* logk).

**Actual:**

For Pima Data Set : the number of test cases is 100; K =5; run Time = 31ms;