Ziyan Guo

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

(30 points) Dataset details: Describe the data and some simple visualizations (for images, a few examples from each category; for other data, perhaps some scatter plots or histograms that show a big picture of the data). Describe your training/test split for K-NN and justify your choices.

**pima-indians-diabetes-database:**

There is the changes of error rate under different values of k.

Here the K value is from 1 to 50, and the y-axis represents the error rate on the x-axis.

Graphical user interface, chart, application, line chart

Description automatically generated

（Now using the Minkowski distance test method）

Graphical user interface, chart, application

Description automatically generated

![Text

Description automatically generated]()（The euclidean metric method is now used）

Graphical user interface, chart, line chart

Description automatically generated

（The Manhattan Distance method is now used）

Here is some scatter plots show a big picture of the data:

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

(End of pima-indians-diabetes-database graph)

**digit-recognizer dataset:**

Here is Error rate of number 8 with different k values: Chart, line chart

Description automatically generated

I randomly divided the whole database into 20% test set and 80% sample set, The error rate of the learning results at this ratio is much lower than the 50% 50% error rate.

(15 points) Algorithm Description: K-NN is a very clear algorithm, so here describe any data pre-

processing, feature scaling, distance metrics, or otherwise that you did.

**data preprocessing:**

For the data of these two datasets, I created a unique class in the program called data class. Then put these data into the data class for preprocessing according to the different characteristics of each row of data.

**feature scaling:**

In the data set about diabetes, there is a feature value of insulin, and his value will vary greatly. So in order to narrow the effect of this value on the overall prediction results, I limited this value to be between zero and 1. In this way, its influence on other eigenvalues can be reduced and the overall accuracy can be improved.

**distance metrics:**

In k-means or kNN, we often use Euclidean distance to calculate the distance between nearest neighbors, and sometimes Manhattan distance. In the diabetes dataset I used all three distance test methods to better improve the accuracy. Then in the handwritten font recognition dataset, I chose to use the Manhattan distance to calculate the distance based on the accuracy of the diabetes dataset.

(45 points) Algorithm Results: Show the accuracy of your algorithm|in the case of the Pima Dataset, show accuracy with tables showing false positive, false negative, true positive and true negatives. For the Pima Dataset, use three di\_erent distance metrics and compare the results.

In the case of the MNIST digits show the complete confusion matrix. Choose a single digit to measure accuracy and show how that number varies as a function of K.

**Pima Dataset：**

This is happen when k = 2 and k = 3.

Text

Description automatically generated

Accuracy is 72% for k = 2.

**digit-recognizer dataset:**

**Text

Description automatically generated**

complete confusion matrix:

A picture containing text, electronics, keyboard

Description automatically generated

single digit 8 to measure accuracy and number varies as a function of K: Chart, line chart

Description automatically generated

Accuracy around 80%

(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.

You may find time counter on the end of the all the print. And

 the actual "wall-clock" time for Pima Dataset will be around 1 second.

And,

 the actual "wall-clock" time for digit-recognizer Dataset will be around 6 seconds. If use all the data to be train set the run time will be about 3mins.

For the run-time of your algorithm since the loop in the knn function, So the run-time of it will be O(N).