Wild Mushrooms Classification – Edible or Poisonous

Yulin Shen

ECE 539 Project Report

Professor: Hu Yu Hen

2013 Fall

(I allow code to be released in the public domain)

Contents

Introduction

Practicality of the project

Mushrooms, as a kind of food, are very special due to their edibility. Some countries treat mushrooms as a kind of high nutrition food. However, only small portions of them are edible. It is really dangerous to eat a poisonous mushroom. Thus, I want to use some classification algorithms to develop a best model to predict whether new emerging mushrooms are edible based on the detected data of the mushrooms. Furthermore, it is an opportunity to compare the classifiers and also understand how they operate.

Aim of the project

The project uses the data from UCI Machine Learning Repository. I intend to implement 2 classification algorithms to build models for prediction. In the process, the project aim to increase the accuracies of them. And also, I intend to compare the 2 classifiers to know their advantages and disadvantages.

Work Performed

Data analysis

The original dataset is recorded with alphabetic characters. Although each alphabetic character is also represented as an integer, the range of each feature is not in an integral scale.

First, I calculate a lowest common multiple number from each number of different kinds in features. Then, I use this number to be the range of them, and convert each alphabetic character to a reasonable integer. Now, it can help the K-NN to get better order of the distances.

Contrast to the K-NN, Naïve Bayer does not need the dataset to be recorded in numerical character. Thus, I just keep the original dataset to do implementation.

The dataset contains about 8000 combinations. Although the number of data is enough, we still need to divide them to get the training samples and testing samples. In this case, I implement 4-way cross validation to get 4 pairs of training dataset and testing dataset. The size ratio of them is 3:1.

K-NN

After getting 4 pairs of training dataset and testing dataset, I need to calculate distances between each point in training dataset and each point in testing dataset. Because each feature's importance is same, I choose way of Euclidean distance to do calculations without weights. After storing them in a matrix, I sort them in ascending order. Now, I can start to consider how to determine the final result. As we know, the first element in the array should take the biggest weight. Hence, I try to set the weights for each of elements in the matrix. Because the feature size is 22, the number of feature difference is 22 at most. Now, I can implement an exhaustive way from 1 to 22 to find which K is best in the model. Finally, I can compare the predicted results with the true results to get the classification rates.

$$y(i^*) = \underset{1 \le i \le n}{Min.} \| y(i) - x \|$$

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Naïve Bayer

The algorithm just needs the original dataset, because it is to calculate the possibility of each feature. In another sentence, it counts the number of features instead of calculating. For example, in the first feature column, I need to count the number of 'b', and it when the mushroom is edible. Later, I need to calculate the 2 possibilities that 'b' emerging and mushroom is edible and also poisonous. After getting all 2 possibilities of features, I need to multiply them to do comparison. The result of comparison can determine whether the mushroom is edible or poisonous. Now, I can calculate the classification rate as above.

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

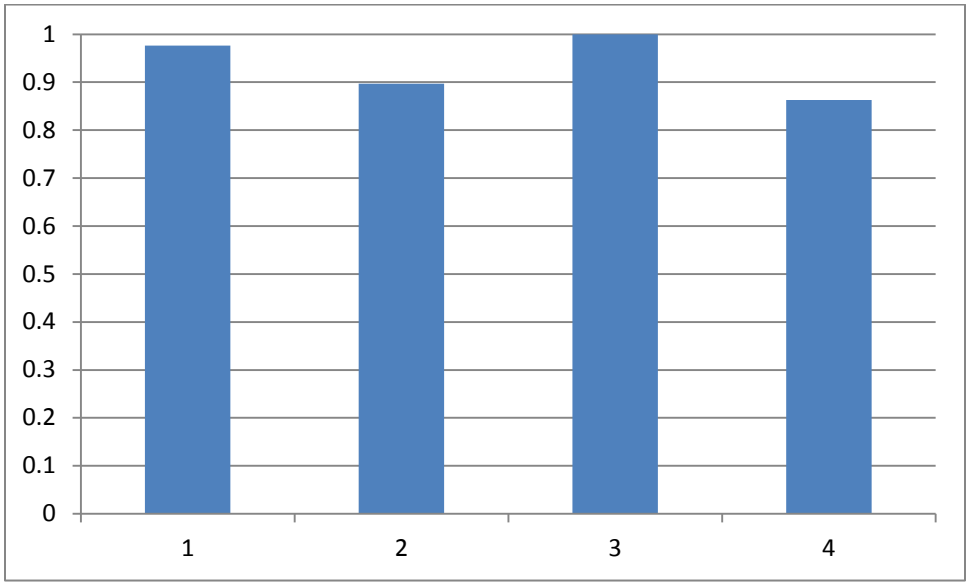
$$\begin{aligned}
&p(C, F_1, \ldots, F_n) \\
&\quad = p(C)\ p(F_1, \ldots, F_n|C) \\
&\quad = p(C)\ p(F_1|C)\ p(F_2, \ldots, F_n|C, F_1) \\
&\quad = p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3, \ldots, F_n|C, F_1, F_2) \\
&\quad = p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3|C, F_1, F_2)\ p(F_4, \ldots, F_n|C, F_1, F_2, F_3) \\
&\quad = p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ \ldots p(F_n|C, F_1, F_2, F_3, \ldots, F_{n-1}).
\end{aligned}$$

Results

K-NN:

Classification rate1: 0.9764 Classification rate2: 0.8966

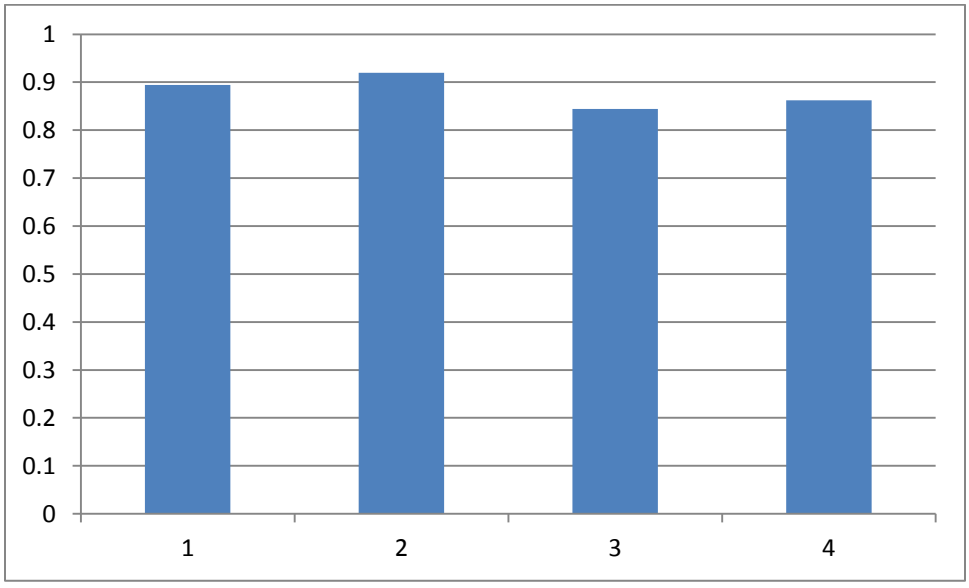
Classification rate3: 1 Classification rate4: 0.8631



Naïve Bayer:

Classification rate1: 0.9764 Classification rate2: 0.8966

Classification rate3: 1 Classification rate4: 0.8631



Discussion

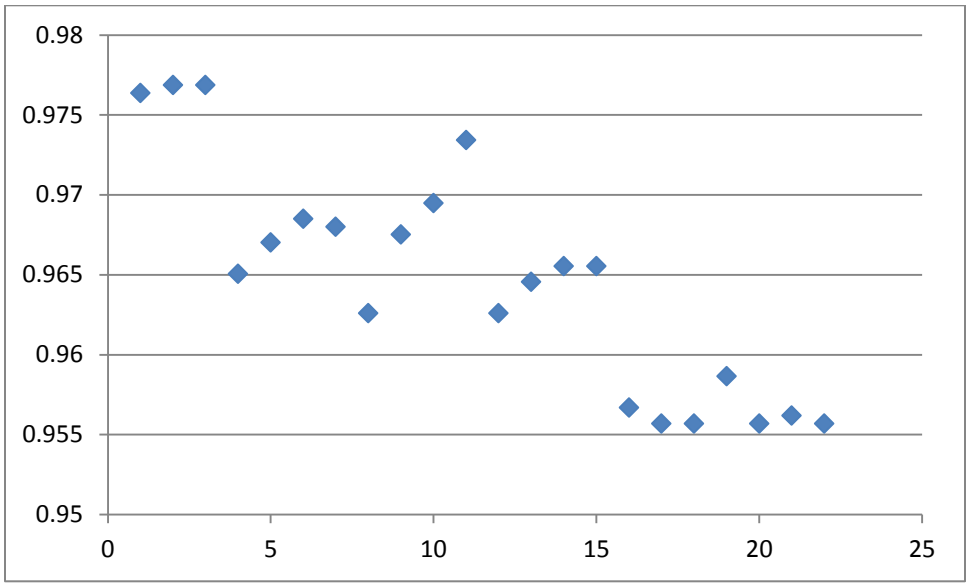
I tried to use the original dataset to implement the K-NN. The accuracy is obviously lower than the dataset converted in an integral scale. In order to increase prediction accuracy, the way to dealt with data is also important. They should be in the same range. It can help classifiers to get better results.

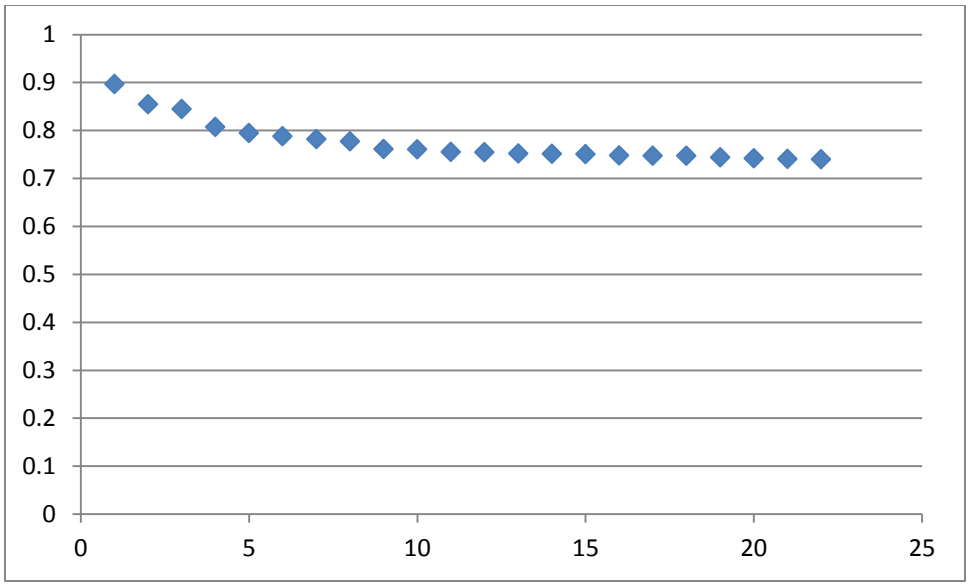
The crucial procedures in the K-NN are how to calculate the distances and how to determine the results. Maybe in other cases, some features are more important than other ones. In this time, the algorithm can give more weight in this feature. It can help to improve the order of distances. And also, the algorithm can give weights in the process of determining results. The first distance is the smallest one, so it is the closest point to the testing data. The result should be most convincing. Actually, it is like the linear regression. However, at this case, I do not think implement regression in the last procedure is a good idea. Because if I implement that, I may have 4 different models of regression because of 4 pairs of datasets, now it is possible to determine which one is better. We do not know the true testing sample for future prediction and also the true training dataset should include both testing samples and training sample in the project. Hence, it is useless to implement the regression. However, to find some reasonable weights for them is still a good choice.

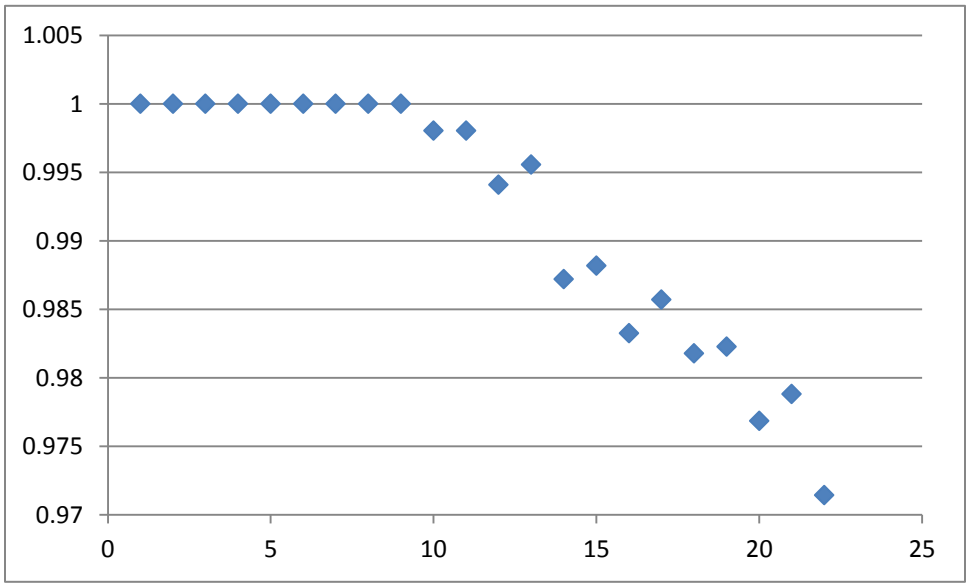
We can see that the sample 3 have the highest classification rate, and the sample 1 is also good in prediction. However, the sample 2 and 4 are not very well. Because I parse the dataset into 4 separate sets. Maybe in some sets, the data is not in an average manner. For example, the set 3 may have a lot of poisonous mushrooms or the feature 1 in this set is almost ‘x’. In this case, it is

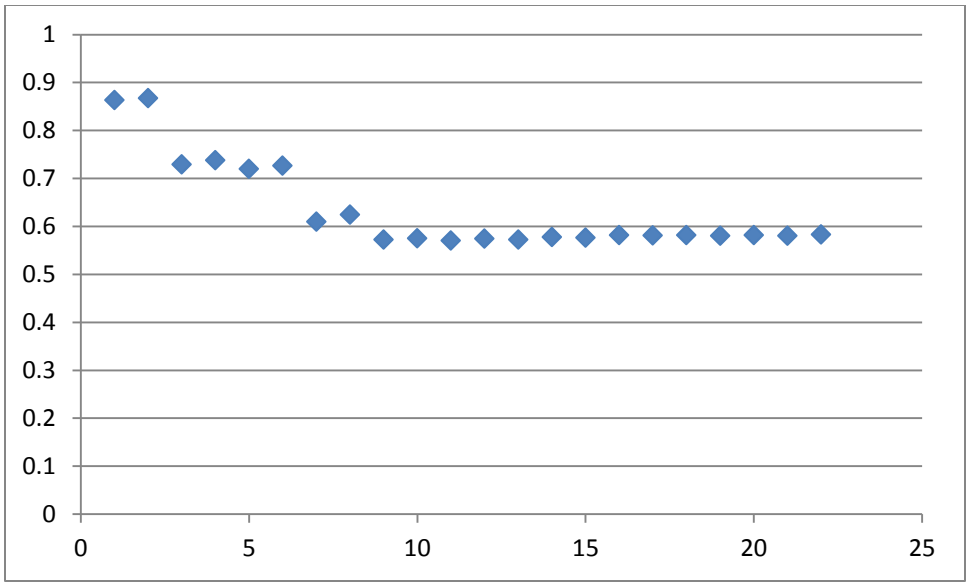
hard to find the similar point in this algorithm. However, the true training set is comprehensive. I think the model can perform higher accuracy if it is implemented in a realistic case.

In the project, I implement an exhaustive way to detect which K is better for prediction. We can see that K =1, 2, or 3 is better in the sample 1, 2, and 4. However, the sample 2 is very stable due to dataset in a good condition.




1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25


1.005
1
0.995
0.99
0.985
0.98
0.975
0.97
0
5
10
15
20
25



The classification rates of Naïve Bayer are all about 90%. Although I implement the same procedures as a reference, the classification rates are a little lower than those in the reference. This algorithm is mainly based on the probability. I think the result is not very good due to the same reason as K-NN, which is caused by the separate datasets. However, if I ignore this bad effect, the performance of the algorithm is worse than the K-NN in this problem. In my opinion, the problem has more than 20 features, if the algorithm is only to compare the multiple result of each possibility, the big deviation may be produced. I think in the last procedure, it is better to implement some other models to limit the deviation for prediction.

In the process, I calculate all probabilities for all features. We can see that some probabilities are very high, which means that if this feature emerge, the mushroom is likely to be edible. It is a hint to tell us which features can influence the result more. Hence, we may add weights to them.

References

University of California – Irvine. “Mushroom Dataset”, May 1989.

http://archive.ics.uci.edu/ml/datasets/Mushroom

Grayson Leonard, Matt Schartman. “Classifying Edibility of Mushrooms”, June 2012.

Min-Ling Zhang, Zhi-Hua Zhou. “A K-Nearest Neighbor Based Algorithm for Multi-label Classification”.

Wikipedia. “Naïve Bayes Classifier”. http://en.wikipedia.org/wiki/Naive_Bayes_classifier