Machine Learning

Program assignment #3

0410001

Hong-Shuo, Chen

1. Abstract

   According to the class, we know what Decision Tree, K- nearest neighbor and naïve Bayes do. This time we use different classifiers/regressors to analyze two data sets.

2. Problem
   - In this assignment you need to use all the classifiers/regressors in the lecture (i.e. Decision Tree, K-nearest neighbor and naïve Bayes) to analyze two data sets.
   - In naïve Bayes, category features need to do laplace smooth and continuous features need to calculate the PDF (probability density function).
   - You need to submit your code and report. The report should include results, using library, language and explanation of your code. Also you need to tell us your idea about the results. (e.g. You can say why a classifier is better or worse than another)
   - Notice: You can call library this time
3. Data set
   Download the data sets from the following websites and split the data randomly to training data and test data (70% / 30%) then do your analysis
   - Iris data set
     https://archive.ics.uci.edu/ml/datasets/Iris
   - Forest Fires Data Set
     https://archive.ics.uci.edu/ml/datasets/Forest+Fires

4. Environment
    - Ubuntu 16.04.3 LTS
5. Using library and language
    - Library:
        - numpy
        - graphviz: plot the decision tree
        - pandas: load csv
        - sklearn: call the function
    - Language: Python 3.5.2
6. Results

```
max@max-VirtualBox:~/Desktop/ML/hw3$ python3 hw3.py
Iris
DecisionTree accuracy:  1.0
K = 5, KNN accuracy:    0.961904761905
Naive Bayes accuracy:   0.977777777778
```

PDF of Iris data, and this is the first ten data.

```
The probability of the samples for each target class.
[[  7.77533683e-087   9.90336750e-001   9.66325045e-003]
 [  2.57456781e-083   9.90031188e-001   9.96881240e-003]
 [  7.80065769e-198   2.91465938e-008   9.99999971e-001]
 [  1.94431658e-160   1.62098246e-006   9.99998379e-001]
 [  9.81492713e-086   9.15029471e-001   8.49705294e-002]
 [  2.89476597e-116   1.51108456e-001   8.48891544e-001]
 [  1.00000000e+000   7.16075787e-016   2.26232717e-024]
 [  1.04353168e-058   9.99806351e-001   1.93648626e-004]
 [  1.00000000e+000   4.97158926e-015   9.40947612e-024]
 [  1.49889084e-087   7.58213888e-001   2.41786112e-001]
```

```
Forestfires
DecisionTree accuracy:  0.403846153846
K = 5, KNN accuracy:    0.614958448753
Naive Bayes accuracy:   0.2564102564102564
```

PDF of Forest fires data, and this is the first five data.

```
The probability of the samples for each target class.
[[  3.72448526e-002   1.78143396e-001   7.32435232e-001   2.36871415e-003
    4.98078055e-002   0.00000000e+000]
 [  3.72486328e-002   1.94394131e-001   7.11792386e-001   6.85496622e-003
    4.97098842e-002   0.00000000e+000]
 [  9.60907896e-003   2.35461005e-001   2.96450590e-001   1.74965649e-003
    4.56729670e-001   0.00000000e+000]
 [  3.35548350e-002   1.73743020e-001   7.85090197e-001   2.49733098e-003
    5.11461664e-003   0.00000000e+000]
 [  1.73383749e-002   4.60394143e-001   5.20397055e-001   1.87042648e-003
    1.49657348e-010   0.00000000e+000]
```

7. Algorithm and Code Explanation

   Decision Tree:

   First, I call this two function to construct my Decision Tree Classifier, and also fit the training data into this model.

```
#Decision Tree
clf = tree.DecisionTreeClassifier()
clf = clf.fit(train_data, train_target)
```

   Because I want to visualize my Decision Tree, I import a library graphviz to help me plot this tree.

```
#Plot the Decision Tree
dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=iris.feature_names,
                                class_names=iris.target_names,
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("iris")
```

   Finally, I fit my testing data into my model to predict the result. The function clf.socre is powerful. We can fit the testing data and target simultaneously, and get the accuracy right away.

```
#Print the accuracy
accracy = clf.score(test_data,test_target)
print('DecisionTree accuracy:', end = "\t")
print(accracy)
```

KNN:

This is much easier to call the function KNeighborsClassifier. I just fit the trainning data and testing data respectively, and I can get the accuracy.

When we construct the KNeighborsClassifier, we can set the number of the nearest neighbors. After some try and error, I found that K = 5 is the best choice. So I set the n_beighbors as 5 here.

```
#KNN
neigh = KNeighborsClassifier(n_neighbors=5)
accracy = neigh.fit(train_data, train_target).score(train_data, train_target)
print('K = 5, KNN accuracy:', end = "\t")
print(accracy)
```

Naïve Byes:

In this model, we need to separate the training data into category features and continuous features first. I also do some mapping before this step. Map day and month into integer.

```
#Naive Bayes
cat_train_data = train_data[:,:4]
cot_train_data = train_data[:,4:]
cat_test_data = test_data[:,:4]
cot_test_data = test_data[:,4:]
```

The category features need to do with Laplace smooth. I call the function MultinomialNB here, which deals with the category features. There is a parameter α to construct the model with Laplace smooth. The α is set as 1 as default, which means Laplace smooth, so we don't need to modify any parameter.

PDF_cat means the probability of the samples for each target class in category features

```
#Laplace smooth
mnb = MultinomialNB()
PDF_cat = mnb.fit(cat_train_data, train_target).predict_proba(cat_test_data)
```

On the other hand, the continuous features need to be deal with GaussianNB. I also fit in the training data and testing data to get the probability.

This is the PDF that we predict. Predict the model as normal distribution.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

PDF_cot means the probability of the samples for each target class in continuous features.

```
gnb = GaussianNB()
PDF_cot = gnb.fit(cot_train_data, train_target).predict_proba(cot_test_data)
```

We get two PDF, so we need to do some process to determine the final PDF. I multiply this two PDF to get the final PDF to determine predict result.

```
mul = np.multiply(PDF_cat,PDF_cot)
```

In the end, I calculate the accuracy without calling any function, and print the result.

```
correct = 0
for i in range(len(test_target)):
        if np.argmax(mul[i]) == test_target[i]:
                correct = correct + 1
accuracy = correct/len(test_target)
print('Naive Bayes accuracy:', end = "\t")
print(accuracy)
print('The probability of the samples for each target class.')
print(PDF_cot)
```

8. Discussion

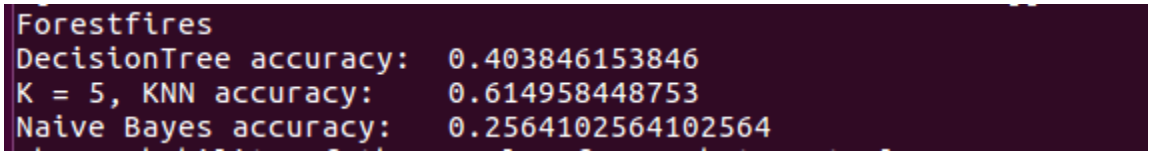
```
max@max-VirtualBox:~/Desktop/ML/hw3$ python3 hw3.py
Iris
DecisionTree accuracy:  1.0
K = 5, KNN accuracy:    0.961904761905
Naive Bayes accuracy:   0.977777777778
```

In the first model, the accuracy of them are all very high and almost close to one. I test for many times, I found that we cannot find the order which is best. Sometimes Decision Tree is the best, and sometimes Naïve Bayes is the best. However, I found that the accuracy of KNN never equal to one. So in Iris dataset I may give the order as

DecisionTree = Naïve Byes > KNN.

```
The probability of the samples for each target class.
[[  7.77533683e-087   9.90336750e-001   9.66325045e-003]
 [  2.57456781e-083   9.90031188e-001   9.96881240e-003]
 [  7.80065769e-198   2.91465938e-008   9.99999971e-001]
 [  1.94431658e-160   1.62098246e-006   9.99998379e-001]
 [  9.81492713e-086   9.15029471e-001   8.49705294e-002]
 [  2.89476597e-116   1.51108456e-001   8.48891544e-001]
 [  1.00000000e+000   7.16075787e-016   2.26232717e-024]
 [  1.04353168e-058   9.99806351e-001   1.93648626e-004]
 [  1.00000000e+000   4.97158926e-015   9.40947612e-024]
 [  1.49889084e-087   7.58213888e-001   2.41786112e-001]
```

From the PDF, we can found out that there are always have two classes are very close, and the remaining one will be very close to one, so I think this model is very good to eliminate one class which is less relevant.



In the forestfires dataset, we found that the accuracy is different from each other in a large scale in different model.

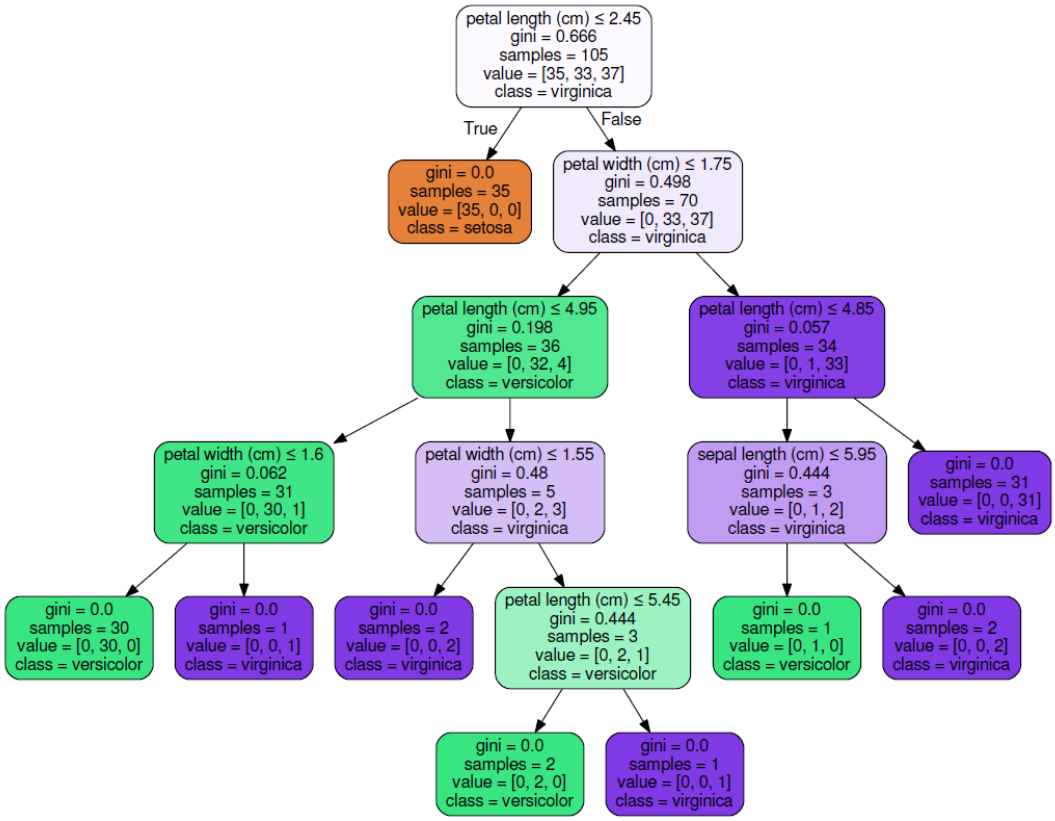
The order is

KNN > DecisionTree > Naïve Byes

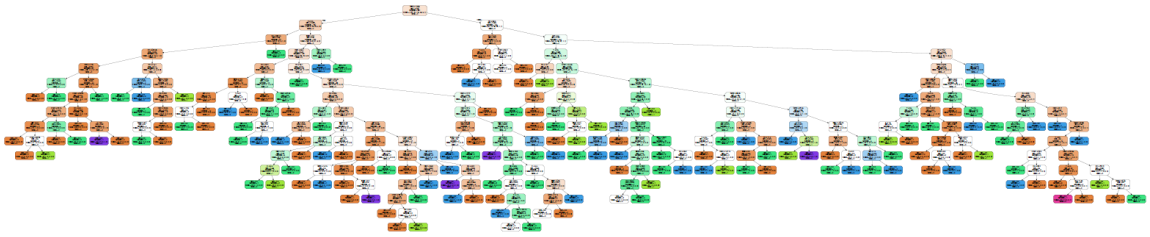
I guess the reason that Naïve Byes is the lowest one is that our model may not the normal distribution, but we use Guassian model to train it, so the accuracy is very low.

The KNN is highest. I think it is due to most of data gathering in specific class. Most data lay in 10~1000. We find the k nearest neighbors, when the K is big the accuracy will increase, it means that the misprediction will not influence so much. However, it means the data which minority will have low predict accuracy, so we need to choose appropriate K.

The decisiontree lay between this two model. I think it is a nice model, and maybe we can use random forest to increase accuracy.



The Iris data Decision Tree



The forest data Decision Tree