Classification Problem: Wine Quality

Abstract:

We got a dataset about the quality of wine, the dataset has 11 attributes of wine and the quality(score between 0 and 10) for each wine. By using decision tree, neural network and svm algorithm to build a model separately for the dataset, I have compared the accuracy of each model. What's more, to find a best result, I have adjusted some parameter for each model. Also, I have discussed the advantages and disadvantages for each algorithm.

The description of the dataset:

The dataset(4898 rows, 12 columns) is related to white variants of the Portuguese "Vinho Verde" wine.

Attributes for each column:
1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol

Output variable (based on sensory data):
12 - quality (score between 0 and 10)

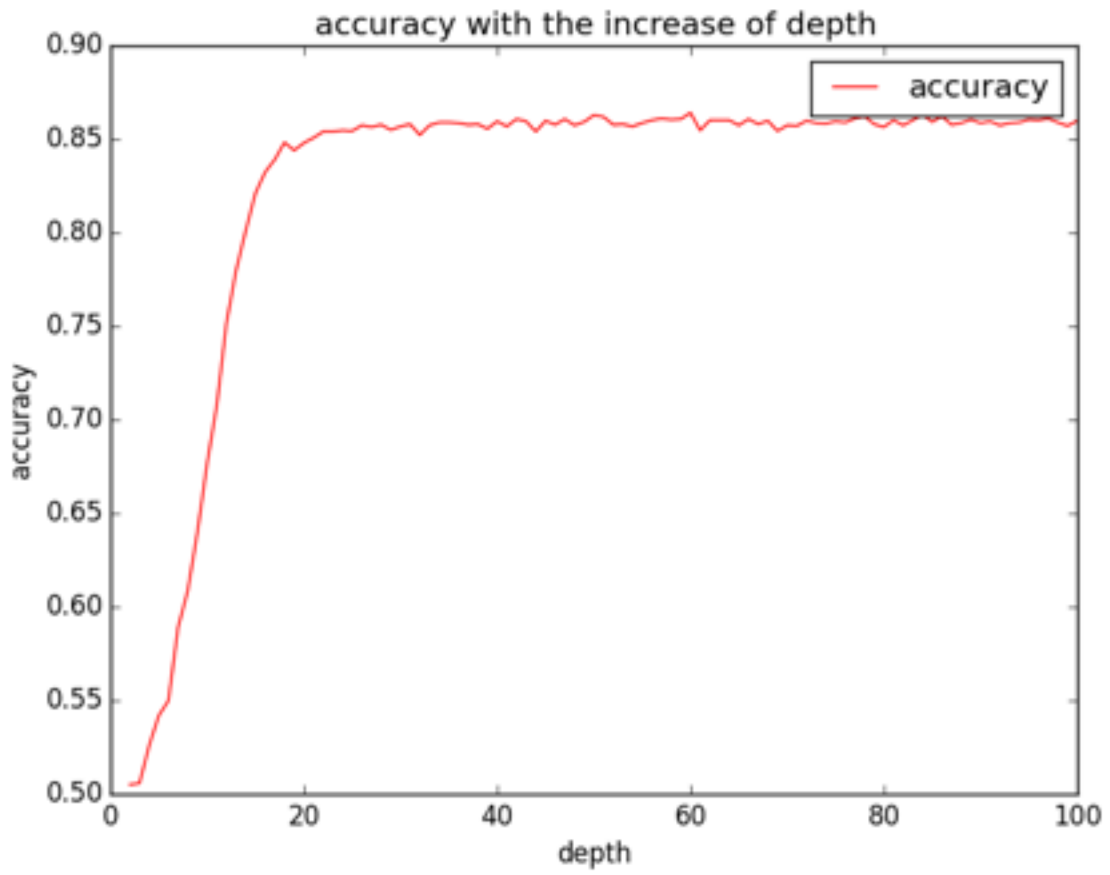
The idea of building model:

By searching information online, we need to split the dataset randomly into train and datasets with a ratio of 67% train because this is a common ratio for testing an algorithm on a dataset. And use the other 67% random data in dataset to test the model and get results.

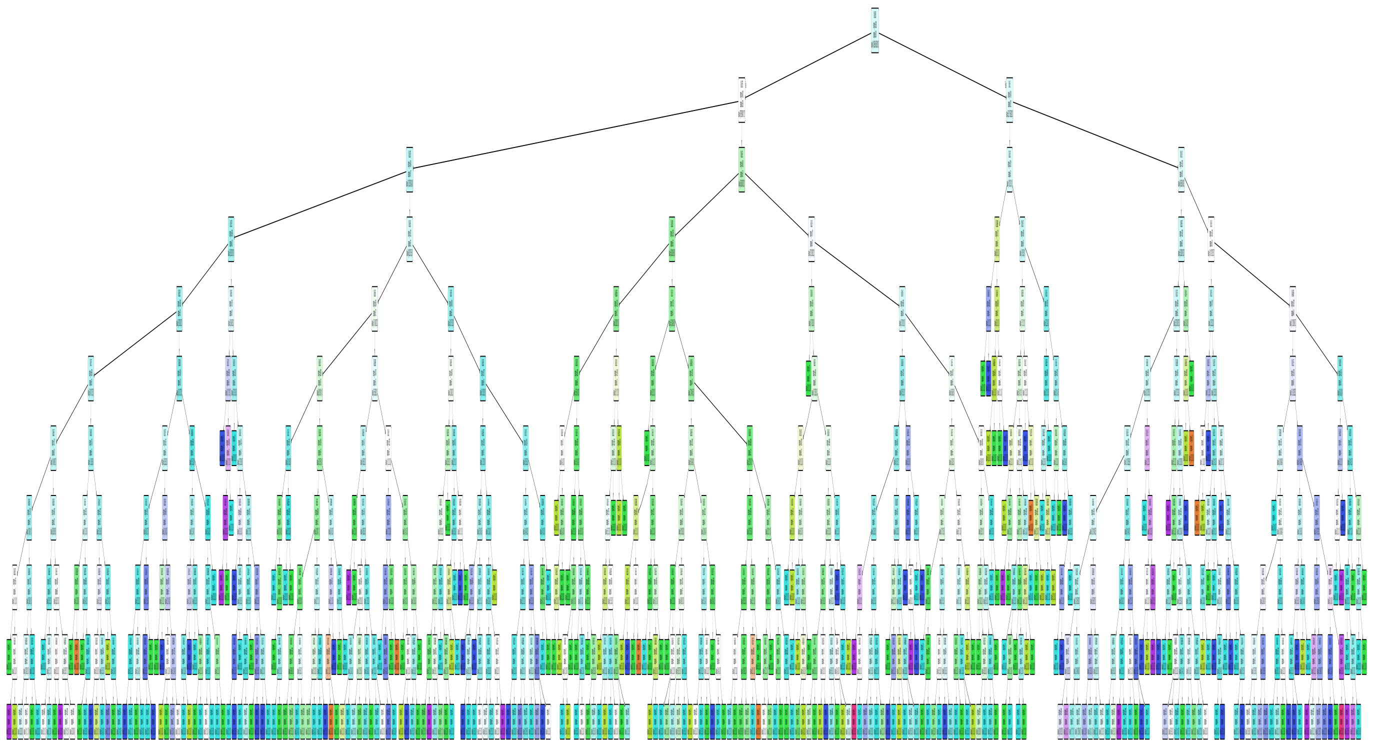
Code of three algorithm(python 2.7.9):

Decision Tree:

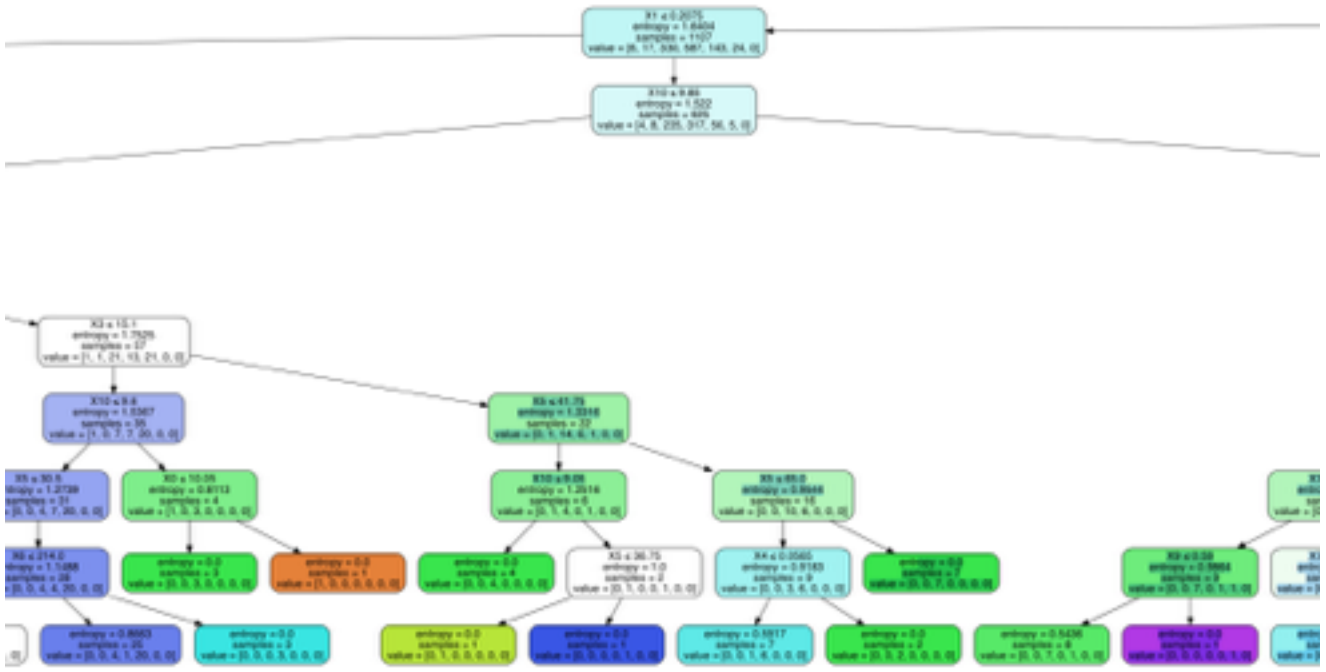
```
from sklearn import tree
import csv
import random
from sklearn.externals.six import StringIO
import pydot
from IPython.display import Image
from sklearn.metrics import accuracy_score
from matplotlib import pyplot

#load file
lines=csv.reader(open('final.csv',"rb"))
dataset=list(lines)

for i in range(len(dataset)):
    dataset[i]=[float(x) for x in dataset[i]]

#seperate dataset
def splitDataset(dataset):
    splitRatio = 0.67
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
    while len(trainSet) < trainSize:
        index = random.randrange(len(copy))
        trainSet.append(copy.pop(index))
    a=[row[0:11] for row in trainSet]
    b=[row[-1] for row in trainSet]
    return [a, b]
#X is training set and Y is its results
X, Y= splitDataset(dataset)
#A is testing data and B is its actual quality score
A, B= splitDataset(dataset)


#test accuracy in different depth and plot line chart from depth 2 to 100
def test_depth(max_depth, X_train, X_test, Y_train, Y_test):
    depths = []
    accuracy=[]
    for depth in range(2, max_depth + 1):
        depths.append(depth)
        clf = tree.DecisionTreeClassifier(criterion='entropy',
max_depth=depth)
        clf.fit(X_train, Y_train)
        test_result=clf.predict(X_test)
        accuracy.append(accuracy_score(test_result, Y_test))

    pyplot.plot([number for number in range(2, max_depth+1)], accuracy, '-r',
label="accuracy")
    pyplot.legend()
    pyplot.xlabel('depth')
    pyplot.ylabel('accuracy')
    pyplot.title('accuracy with the increase of depth')
    pyplot.show()

# call the function
test_depth(100, X, A, Y, B)


#Cause the decision tree picture would be too large if we define the depth to
100, so I set max_depth equal to 10 and output a pdf file for the picture.

clf = tree.DecisionTreeClassifier(criterion='entropy', max_depth=10)
clf.fit(X, Y)
dot_data = StringIO()
tree.export_graphviz(clf, out_file=dot_data,filled=True, rounded=True,
special_characters=True)
graph = pydot.graph_from_dot_data(dot_data.getvalue())
graph.write_pdf("quality of wine.pdf")
```



Overall View



Detail



SVM:

```
import csv
import random
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

#load file
lines=csv.reader(open('final.csv',"rb"))
dataset=list(lines)

for i in range(len(dataset)):
    dataset[i]=[float(x) for x in dataset[i]]

#seperate dataset
def splitDataset(dataset):
    splitRatio = 0.67
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
    while len(trainSet) < trainSize:
        index = random.randrange(len(copy))
        trainSet.append(copy.pop(index))
    a=[row[0:11] for row in trainSet]
    b=[row[-1] for row in trainSet]
    return [a, b]
#training data
X, Y=splitDataset(dataset)
#testing data
A, B=splitDataset(dataset)

# default
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape=None, degree=3, gamma='auto',
kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)

M=[0.01,0.1,1,10,100,1000,10^4,10^5,10^6,10^7]
for C in M:
    clf = SVC(C)
    clf.fit(X, Y)
    test_result=clf.predict(A)
    print accuracy_score(test_result, B)
```

```
accuracy_score=
0.44925327644
0.44925327644
0.735751295337
0.848826577263
0.856141420299
0.856141420299
0.851264858275
0.851874428528
0.850960073148
0.851264858275
```

We can find that accuracy improves with the increased C, and the best accuracy is around 85.6% when C is 100 and 1000. The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

Neural Net:

```
import csv
import random
import neurolab as nl
from sklearn.metrics import accuracy_score

#load file
lines=csv.reader(open('final.csv',"rb"))
dataset=list(lines)

for i in range(len(dataset)):
    dataset[i]=[float(x) for x in dataset[i]]

#seperate dataset
def splitDataset(dataset):
    splitRatio = 0.67
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
    while len(trainSet) < trainSize:
        index = random.randrange(len(copy))
        trainSet.append(copy.pop(index))
    a=[row[0:11] for row in trainSet]
    b=[row[-1] for row in trainSet]
    b=zip(b)
    return [a, b]
X, Y= splitDataset(dataset)
A, B= splitDataset(dataset)

#node number denote the number of nodes in hiden layer
accuracy=[]
for node_number in range(100,150):
    # i neurons for hidden layer, 1 neuron for output
    # 2 layers including hidden layer and output layer

    net=nl.net.newff([[0,1],[0,1],[0,1],[0,1],[0,1],[0,1],[0,1],[0,1],
[0,1],[0,1],[0,1]],[node_number,1])
    net.trainf = nl.train.train_rprop
    net.train(X, Y, epochs = 100)
    test_result= net.sim(A)
    print test_result
    accuracy.append(accuracy_score(test_result, B))
print accuracy
```

I was stuck by this program for a long time, because all the output is 1, after I looked for a lot of information, I realized that I didn't not formalized the input data, they should be around between 0 to 1 in certain ratio. And I used excel to realize this function but it didn't work. Maybe some problem of file format. But I think the code is correct.

Advantages & Disadvantages of algorithms:

Decision Tree:

Advantages:
Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation.
Have value even with little hard data. Important insights can be generated based on experts describing a situation (its alternatives, probabilities, and costs) and their preferences for outcomes.
Allow the addition of new possible scenarios
Help determine worst, best and expected values for different scenarios
Use a white box model. If a given result is provided by a model.
Can be combined with other decision techniques.

Disadvantages:
For data including categorical variables with different number of levels, information gain in decision trees are biased in favor of those attributes with more levels.[4]

Calculations can get very complex particularly if many values are uncertain and/or if many outcomes are linked.

SVM:

Advantages:
Effective in high dimensional spaces.
Still effective in cases where number of dimensions is greater than the number of samples.
Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages:
If the number of features is much greater than the number of samples, the method is likely to give poor performances.
SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

NN:

Advantages:
· A neural network can perform tasks that a linear program cannot.
· When an element of the neural network fails, it can continue without any problem by their parallel nature.
· A neural network learns and does not need to be reprogrammed.
· It can be implemented in any application and without any problem.
Disadvantages:
· The neural network needs training to operate.
· The architecture of a neural network is different from the architecture of microprocessors therefore needs to be emulated.
· Requires high processing time for large neural networks.

Conclusion:

For decision tree, with the increase of depth, its accuracy would get better until depth 20. After depth 20, the accuracy is almost stable around 85%.

For SVM, when adjusting penalty parameter C, the larger C is, the better result it would be. The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly.

Reference:

1.Uciedu. (2015). Uciedu. Retrieved 16 December, 2015, from https://archive.ics.uci.edu/ml/datasets/Wine Quality

In-text citation: (Uciedu, 2015)

2.Wikipediaorg. (2015). Wikipediaorg. Retrieved 16 December, 2015, from https://en.wikipedia.org/wiki/Decision_tree

In-text citation: (Wikipediaorg, 2015)

3.Scikit-learnorg. (2015). Scikit-learnorg. Retrieved 16 December, 2015, from http://scikit-learn.org/stable/modules/svm.html

In-text citation: (Scikit-learnorg, 2015)

4.Answerscom. (2015). Answerscom. Retrieved 16 December, 2015, from http://www.answers.com/Q/Advantage_and_disadvantage_of_neural_network

In-text citation: (Answerscom, 2015)

Epilogue

I select python as my final project language cause all homework done before was used in R, so I want to practice my python skill through this project. I really learned a lot from this course because I never contacted such area in my undergraduate. Also, cause this is my first semester in America, this course seems a little difficult for me. However, I really like the way professor teaches and I make rapid progress by doing homework and searching information online. Not only did I get better in coding, but also I have a better comprehension on today's hot big data.