Chase Hendley & Neal Sakash

CSCI 334

May 1, 2017

Comparing Various Data Mining Techniques on MNIST Data

The MNIST database of handwritten numbers is one of the most widely known data sets available, and as such, many techniques on data mining have been used in order to set up predictive analysis of those numbers. We have taken a number of methods together in order to compare which method works best for this dataset. To evaluate the differences, a comparison in the area under the curve for ROC(AUROC) curves was done.

Over the course of the semester, various forms of analysis was done using Support Vector Machines, MultiLayer Perceptron, Random Forest and Decision Trees. As Random Forest and Decision Trees were a new component, we first discuss the procedure for these.

```
from mnist import MNIST
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn import cross_validation
import matplotlib.pyplot as plt

mndata = MNIST('./Datasets')
trainingImages, trainingLabels = mndata.load_training()
testImages, testLabels = mndata.load_testing()

trainingImagesCount = len(trainingImages)
testingImagesCount = len(testImages)

clf = RandomForestClassifier(n_estimators=150, criterion="gini", max_depth=32, max_features="auto")
#clf = RandomForestClassifier()
clf = clf.fit(trainingImages[:1000], trainingLabels[:1000])
#clf = clf.fit(trainingImages[:60000], trainingLabels[:60000])
```

After analyzing through Random Forest, we used a simpler Decision Tree algorithm to go over the data as well.

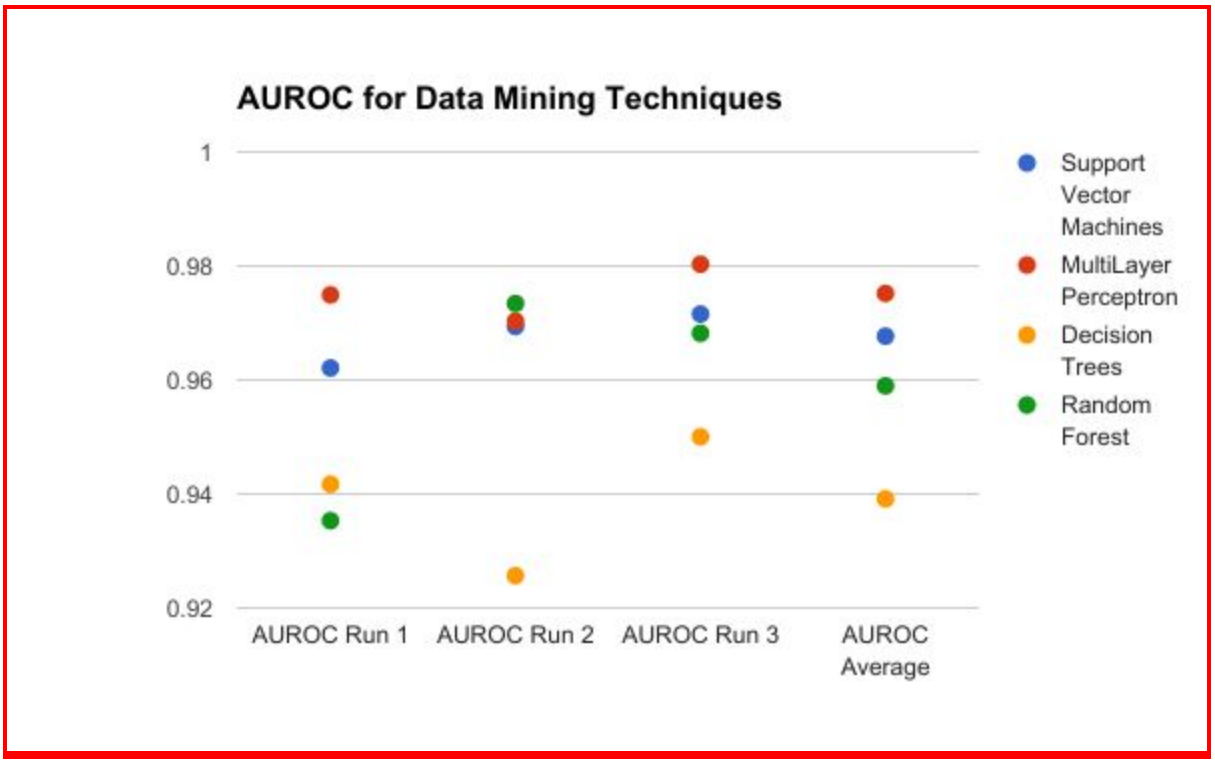
```
from mnist import MNIST
from sklearn import tree
from sklearn import metrics
from sklearn import cross_validation
import matplotlib.pyplot as plt
import StringIO, pydot

mndata = MNIST('./Datasets')
trainingImages, trainingLabels = mndata.load_training()
testImages, testLabels = mndata.load_testing()

trainingImagesCount = len(trainingImages)
testingImagesCount = len(testImages)

clf = tree.DecisionTreeClassifier(criterion="gini", max_depth=32, max_features=784)
#clf = tree.DecisionTreeClassifier()
clf = clf.fit(trainingImages[:1000], trainingLabels[:1000])
#clf = clf.fit(trainingImages[:60000], trainingLabels[:60000])
```

After completing these two methods, results were also gathered from previous SVM and MLP techniques to compare the AUROC. This had the following results with 3 runs using different parameters to each use, and the average of those:



As you can see MLP had the highest results for AUROC with SVM coming in close behind. These were with certain parameters selected. By changing the size of the data analyzed and

factoring in other differences, this table could be completely different. It is important to take not only these factors in when using different data mining techniques, but also what information you wish to find and the size of the dataset. One thing this class has gone over is that there is not one catch all technique. In order to properly gain insight into the data, multiple methods should always be initially used. Once a proper technique is used, as the data is updated or changes, it can be further refined to develop models that efficiently analyze that particular set.

References Used:

https://github.com/efebozkir/handwrittendigit-recognition

https://github.com/moeabdol/mlp-mnist/blob/master/MLP.py

http://yann.lecun.com/exdb/mnist/index.html

http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html