SVM, CART, and Random Forest: A Comparative Study

Dror Ayalon (dda290)

CUSP-GX-5006 Machine Learning for Cities (NYU)
Assigment # 3

Abstract

Classification in one of the major tasks in machine learning. This study compares the restuls of multiple classification algorithms, to find the most effective technique for the given dataset. Many of the results of this study were made under the assumeption that different classification techniques might perform better on different datasets. Therefore, the results of this study could be relevant to this dataset, or to datasets of similar nature.
The classification techniques that were studied are: [1] Support Vector Machines (SVM with Kernels), [2] Decision Trees (CART), [3] Decision Trees (CART) with Bagging, [4] Decision Trees (CART) with Boosting, [5] Random Forest, [6] Random Forest with Bagging, and [7] Random Forest with Boosting.
The main result is that out of the tested algorithms, Random Forest turned out to be the most accurate one for the given dataset.

1 Methods and Data Sets

1. The dataset used for this study is 'manhattan-dof.csv', which was made available to us by NYU. The dataset includes 2645 samples. The attributes that were used from this dataset are the following:

    - BldClassif - Building class. Used as the classes for the classification procedure.
    - GrossSqFt, GrossIncomeSqFt, MarketValueperSqFt - Indipendent variables that were used to generate the prediction model.

2. The data was cleaned to remove outlies. This process improved the results dramatically, probably due to the fact that tree algorithms are very sensitive to outliers. See figure 2.

3. The data was normalized to values between 0 and 1. This process improved the results of the SVM algorith dramatically, from 22.5% - 25.5% error rate to 8.2% - 9.3% error rate after the normalization.

4. The entire study was done using Python3 and

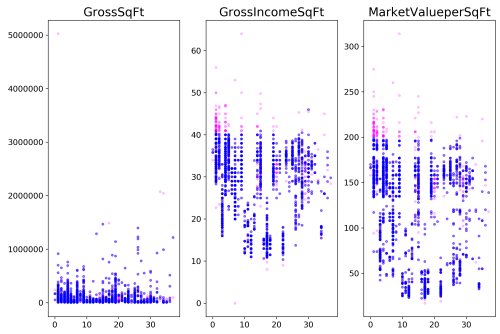

Figure 1: Original dataset VS. Cleaned dataset

the machine learning Python package scikit-learn (http://scikit-learn.org/).

5. To validate the results, a cross-validation process was used, based on the 'train_test_split' method of the scikit-learn package. During the process, 5 batches of data were generated. 4 of which were used as training sets and 1 was used as a validation set. The random split process was done 50 times for each classfication method. Moreover, a Bootstraping procedure was used to improve the restuls of a few of the classification algorithms. This topic will be discussed below.

6. The following scikit-learn algorithms were used to the generate the results for this study:

   - sklearn.svm.svc - Was used to perform SVM with kernels (gamma=500).
   - DecisionTreeClassifier - Was used to classify the data using a regular Decision Tree algorithm. The maximum depth of the tree was set to values between 4 and 8. More about the effect of the depth of the Decision Tree on the overall accuracy of the results on the 'Results' section.
   - RandomForestClassifier - Was used to classify the data using a Random Forest algorithm. This method was most effective with low number of trees (no improvement pass n_estimators=6), and high in depth (more than 6 levels). This observation could be explained by the small size of the dataset, and that Random Forest does not need many iterations to achieve good results. The downside of using a small number of small trees is that these configuration effected very badly on the Boosting procedure. More about the effect of the depth of the Decision Tree on the overall accuracy of the results on the 'Results' section.
   - BaggingClassifier - Was used to apply a Bagging process on a Decision Tree algorithm or a Random Forest algorithm. In most cases, this method showed a slight improvemet over the results of the algorithm without Bagging. Furthermore, in most cases, allowing the algorithm to choose features randomally with replacement (bootstrap_features = True) improved the results.
   - AdaBoostClassifier - Was used to apply a Boosting process on a Decision Tree algorithm or a Random Forest algorithm. The AdaBoostClassifier algorithm showed better results than scikit-learn's GradientBoostingClassifier algorithm across the board.

7. Using these algorithms, the following methods were compared:

   (a) Support Vector Machines (SVM with Kernels)
   (b) Decision Trees (CART)
   (c) Decision Trees (CART) with Bagging
   (d) Decision Trees (CART) with Boosting
   (e) Random Forest
   (f) Random Forest with Bagging
   (g) Random Forest with Boosting

2 Results

- The result of running a comparison between all the tested algorithms shows that the Random Forest algorithm, using 24 trees (n_estimators=24), with maximum depth of 14 levels (max_depth=14), and bootstraping is the most accurate method for the given dataset with 1.06% error.
- The Boosting mechanism did not improve the results of the Random Forest algorithm (see figure 2). This outcome might be because the Random Forest algorithm is generating a variety of trees, and use them to reduce variance, which is general intent behind the Boosting mechanism. Another reason could be the nature of the dataset.
- The best SVM result showed 9% error in classification, and therefore, it is not the best method to predict classes on the given dataset.

Tree Max Depth	Decision Trees	Decision Trees + Bagging	Decision Trees + Boosting	Random Forest	Random Forest + Bagging	Random Forest + Boosting
2	7.82%	7.40%	49.68%	8.25%	7.40%	16.70%
3	7.40%	7.82%	16.28%	7.82%	6.77%	35.94%
4	7.40%	7.19%	10.78%	6.77%	6.77%	8.67%
5	7.82%	7.19%	10.99%	6.13%	6.77%	9.73%
6	10.36%	7.40%	8.88%	6.77%	6.98%	14.80%
7	8.88%	7.40%	8.88%	5.50%	7.19%	9.09%
8	9.09%	7.82%	8.46%	4.65%	7.40%	8.46%
9	9.09%	8.25%	8.25%	4.23%	6.77%	11.42%
10	8.88%	7.82%	8.67%	2.33%	6.77%	9.51%
11	9.30%	9.51%	9.94%	2.54%	6.77%	8.46%
12	9.94%	6.98%	9.09%	2.54%	6.55%	8.46%
13	9.94%	8.25%	8.25%	1.90%	7.40%	7.82%
14	9.94%	9.09%	9.09%	1.06%	7.40%	8.03%
15	9.09%	7.19%	8.03%	1.90%	6.98%	8.25%

Figure 2: A comparison between error rates of different classification methods with a changing maximum depth. It is clear that Random Forest methods improve with depth, while regular decisiton trees do not.
Boosting improved the regular Decision Tree results.

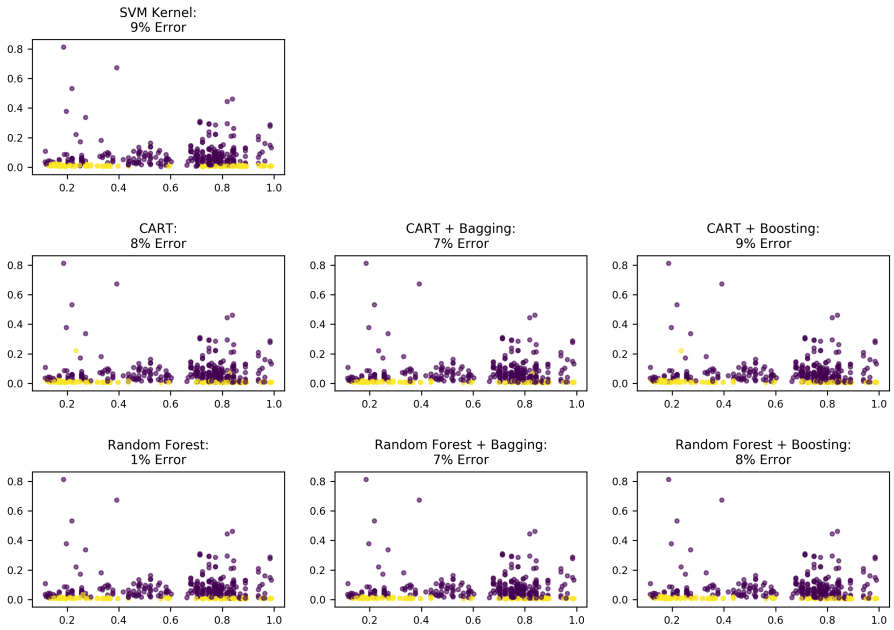

Figure 3: The classification results from all tested algorithms. The graphs show the best consistent result from each algorithm.