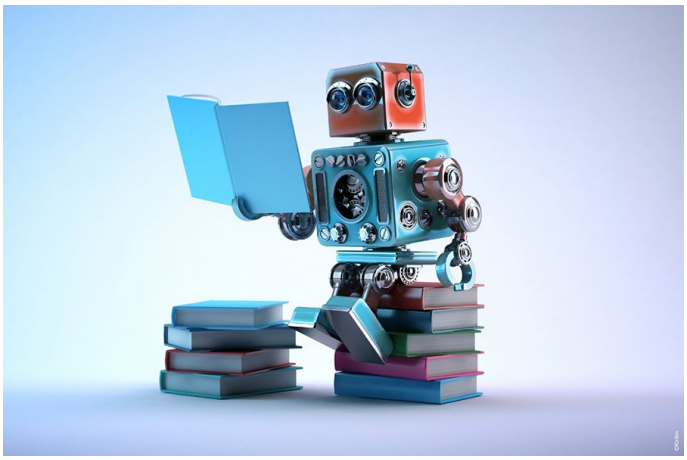
IIT Bombay

Machine Learning

Assignment 1

-Simple Classification



Ushasi Chaudhuri - 174310003

June 19, 2018

Contents

1 Objective

In this assignment, we have to perform three class classification. Based on the given attributes about houses in a particular county, we have to predict the class in the last column, which is "SaleStatus". For this purpose, we are given training data in the file trainSold.csv, and testing data in the file testSold.csv. The latter has the last column missing, for which we have to generate labels.

2 Classification Models

For the assignment, we have submitted two models, random forest classification and extreme gradient boosting based classification. They have been explained as follows.

2.1 Random Forest

A random forest takes a subset of observations and a subset of variables to build decision trees. This method builds multiple decision trees and combines them together to get a more accurate and stable prediction.

2.2 Extreme Gradient Boosting

This method is an advanced form of gradient boosting algorithm. This model provides us with a lot of parameters to tune and adjust. The advantages of this algorithm over the standard gradient boost are,

1. It provides regularization.
2. Parallel processing (making the model extremely faster)
3. Higher flexibility
4. Handling missing values
5. Tree pruning
6. It continues in the existing model (from the last iteration).

2.3 Support Vector Machine

In this algorithm, we plot each data attribute as a point in n-dimensional space (n is number of attributes) with the value of each attribute being a value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes most distinguishably.

3 Pre-processing Strategies

What strategy worked best for data pre-processing and why? Mention any strategies tried that didn't work.

3.1 Strategies Used

We performed some pre-processing on the data and the following were found to give us good results.

1. Replaced all the Nan values in the training data.
2. Replaced all the not a number values in the test data.
3. The "SoldStatus" column had three categorical values, namely SoldFast, SoldSlow, and NotSold. We denoted them with 3, 2, and 1 instead, to denote the different categorical values.
4. From the modified training data, we dropped the SaleStatus column, and appended the training data with the test data samples.
5. The above step was done, since it was noticed that after the dummy encoding, the number of columns in the test data and the train data did not match (This is due to the fact that a few of the columns in test data contains different categorical values than the values in the corresponding column of the train data).
6. After the the dummy encoding, we then separated out the test data and the training data, to perform our experiments.
7. Lastly, we also converted the test and training data samples into valid numeric values, using the `pandas` library.

The pre-processing step is same in both the models.

3.2 Strategies that did not work

A few strategies were tried that did not seem to affect the accuracy much were like, upper-case and lower-case mismatch adjustments, spelling mistakes in a few of the columns, and other such small discrepancies. Hence, we have not included them in the final submission model.

4 Classifiers

Which classifier worked the best? Why do you think those were appropriate for this data? What else was tried that didn't work?
We tried our results in four different models.

- Extreme Gradient Boosting classifier
- Gradient Boosting classifier
- Random forest classifier
- Support vector machine

Out of these, we saw that gradient boosting algorithm did not give as good result as the extreme gradient boosting classifier (approximately 97% accuracy on train data set). Hence, we choose the extreme gradient boosting classifier and did not show the results with the normal gradient booting classifier.
We then implemented the SVM classifier, and it gave us an accuracy of about 51%. Hence, we did discarded svm classifier also from our results. The random forest model provided us with fairly good results (approximately 93%).

5 Hyper Parameter Tuning

Which hyper-parameters were tuned and how? You may include hyper-parameter tuning graphs here.

5.1 Extreme Gradient Boosting

In the Extreme gradient boost method, while the classifier has many hyper parameter, we used only the following parameters in our training:

- Learning rate- 0.2 A higher value drops the accuracy of the system. The default value is 0.3.
- N-estimators- 14 selected.
- Maximum depth- Maximum depth of a tree, Chosen value 9.
- Minimum child weight- Defines the minimum sum of weights of all observations required in a child. This parameter is used to control over-fitting. Too high values can also lead to under fitting. We choose a value of 3.
- Gamma We choose 0.5. Higher values reduces the accuracy.
- Objective -Softmax for multi class classification.
- N-thread- This is used for parallel processing. The number of threads working parallely to increase the speed. We kept a value of 8.
- Seed- 27

5.2 Random Forest Classifier

The following hyper parameter were used in our training-

- N-estimators: Number of trees you want to build before taking the maximum voting or averages of predictions -400. Increasing this value further affects the speed of the performance drastically. However, the accuracy increases partially.
- N-jobs: How many processors is it allowed to use. We choose 2. Ths does not affect the accuracy, but affects the efficiency.
- oob-score: Random forest cross validation method (True).
- Random state: 0
- Max features: Auto: This will simply take all the features which make sense in every tree.

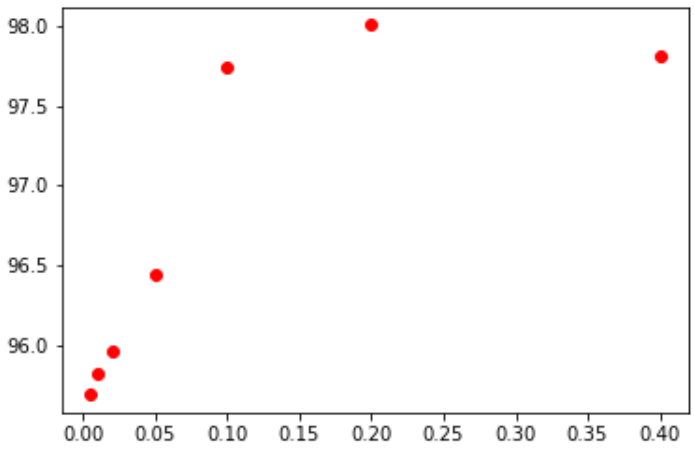


Figure 1: Varying `learning_rate` and keeping others constant. `n_estimator`=14. Chosen value of `learning_rate` as seen from the plot is 0.2.

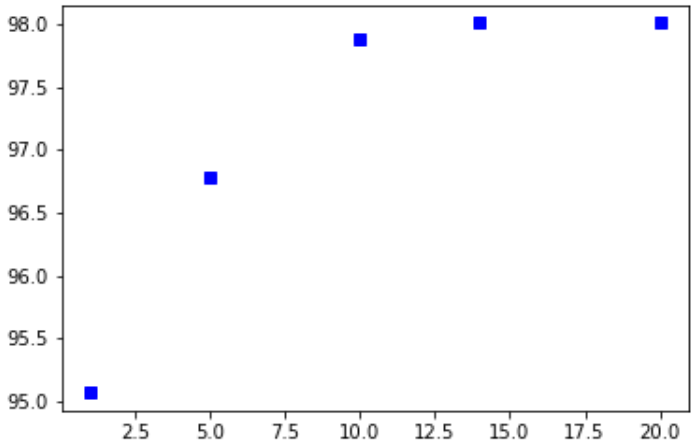


Figure 2: Varying `n_estimator` and keeping others constant. `learning_rate`=0.2. Chosen value of `n_estimator` as seen from the plot is 14.

The two graphs Fig. 1 and Fig. 2 show the hyper-parameter tuning for the XGBoost model. We vary a single parameter every time and note the difference in its performance. In case of XGBoost algorithm, the `learning_rate` and `n_estimator` parameters were found to affect the accuracy the most.

6 Results

We got the following accuracy as shown in Table 1 on the two models with the training and the test data set.

Table 1: Accuracy Table

	Extreme Gradient Boost	Random Forest	SVM
Training	98.01%	93.63%	51.5%
Testing	94.17%	94.58%	53.87%