
2IMM20 - Foundations of datamining

Assignment 1

Students:

Joris van der Heijden (0937329)
Bram van der Pol ()

Email addresses:

j.j.m.v.d.heijden@student.tue.nl
Bram email adres here

Supervisors:

Dr.ir. Joaquin Vanschoren

Eindhoven, March 5, 2018

Contents

1	Appendix	7
1.1	Code question 2.1	7
1.2	Code question 2.2	7

Nepalese character recognition

1

Evaluate k -Nearest Neighbors, Logistic Regression and RandomForests with their default settings.

- *Take a stratified 10% subsample of the data.*
- *Use the default train-test split and predictive accuracy. Is predictive accuracy a good scoring measure for this problem?*
- *Try to build the same models on increasingly large samples of the dataset (e.g. 10%, 20%,...). Plot the training time and the predictive performance for each. Stop when the training time becomes prohibitively large (this will be different for different models).*

Because the logistic regression algorithm took more time than the k -Nearest and the Random Forest algorithm we made two figures; one based on maximum 20% of the data and one based on maximum 6% of the data. For this problem we used 75% of the data to train the model and 25% to validate the model.

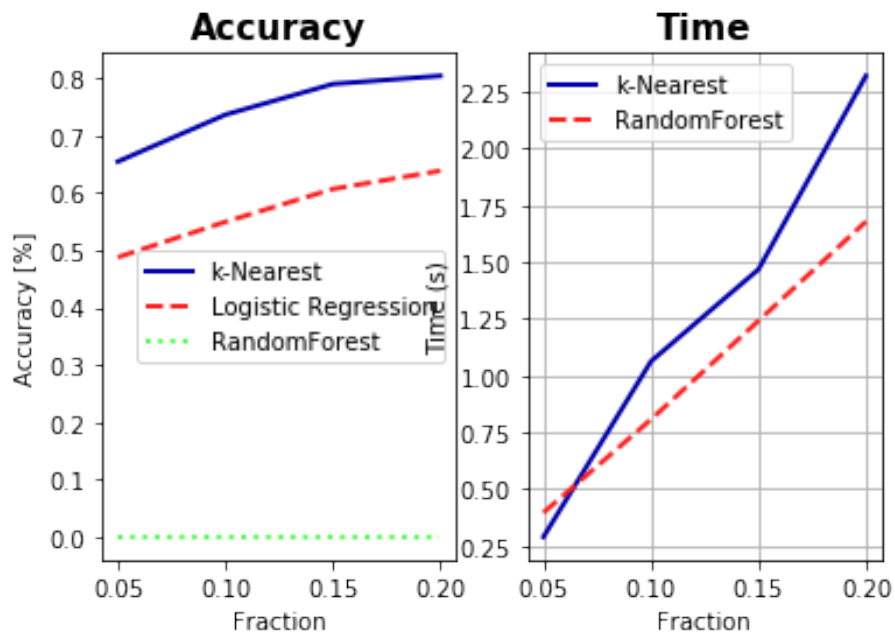


Figure 1: Performance of the k -Nearest and Logistic regression algorithms.

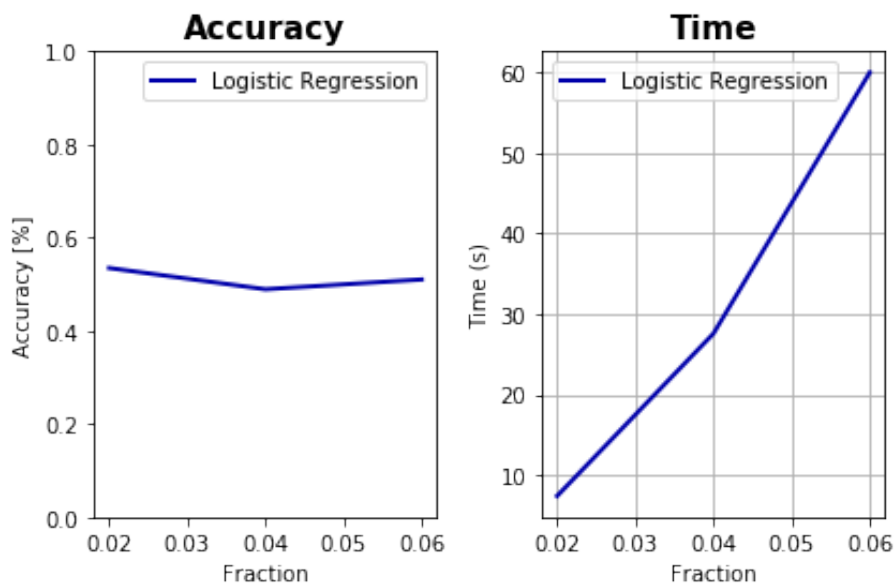


Figure 2: Performance of the Logistic regression algorithm.

Because the data size is sufficiently large the predictive accuracy is a good scoring measure. The characteristics of the k-Nearest and randomForest algorithm is shown in figure 1. This figures shows that the accuracy of the models goes up when more data is used, which is expected. However figure 2 does not show this trend. We do not recommend this algorithm for this problem because of the time needed to train this algorithm and the accuracy of the model.

2

Optimize the value for the number of neighbors k (keep $k < 50$) and the number of trees (keep $n_{estimators} < 100$) on the stratified 10% subsample. Use 10-fold crossvalidation and plot k and $n_{estimators}$ against the predictive accuracy. Which value of $k, n_{estimators}$ should you pick?

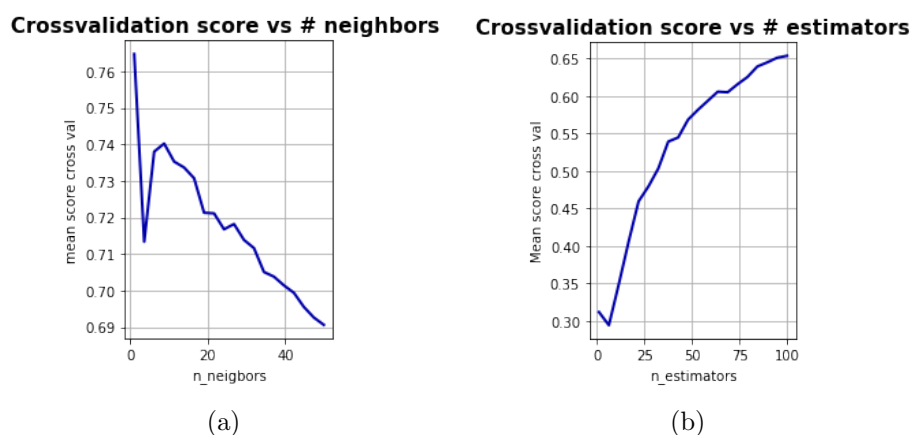


Figure 3

As can be seen in figure 3 is that the number of neighbors should be set to 1 and the number of trees to 100.

3

For the RandomForest, optimize both $n_{estimators}$ and $max_{features}$ at the same time on the entire dataset. - Use a nested cross-validation and a random search over the possible values, and measure the accuracy. Explore how fine-grained this grid/random search can be, given your computational resources. What is the optimal performance you find? - Hint: choose a nested cross-validation that is feasible. Dont use too many folds in the outer loop. -

Repeat the grid search and visualize the results as a plot (heatmap) $n_{estimators} \cdot \max_{features} \rightarrow ACC$ with ACC visualized as the color of the data point. Try to make the grid as fine as possible. Interpret the results. Can you explain your observations? What did you learn about tuning RandomForests?

Accuracy for # estimators and # features

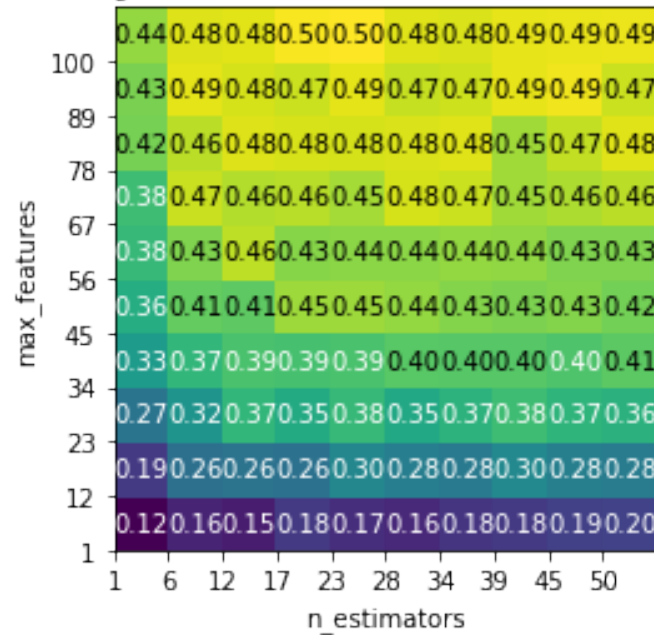


Figure 4: Accuracy for the random forest for different maximum features and number of estimators numbers. This is done for 1% of the total data set (due to the lack of computer power and time).

The finest grid which does not take too much time is a 10x10 grid. This grid gives a good insight in the performance of the randomforest algorithm. We used only 1% of the data to save computing power. This has a big influences on the results, however the characteristics of the algorithm can be examined. Figure 4 shows that the number of trees in the forest

4

1

Do a bias-variance analysis of both algorithms. For each, vary the number of trees on a log scale from 1 to 1024, and plot the bias error (squared),

variance, and total error (in one plot per algorithm). Interpret the results. Which error is highest for small ensembles, and which reduced most by each algorithm as you use a larger ensemble? When are both algorithms under- or over-fitting? Provide a detailed explanation of why random forests and gradient boosting behave this way. - See lecture 3 for an example on how to do the bias-variance decomposition - To save time, you can use a 10% stratified subsample in your initial experiments, but show the plots for the full dataset in your report. In []: from sklearn.model

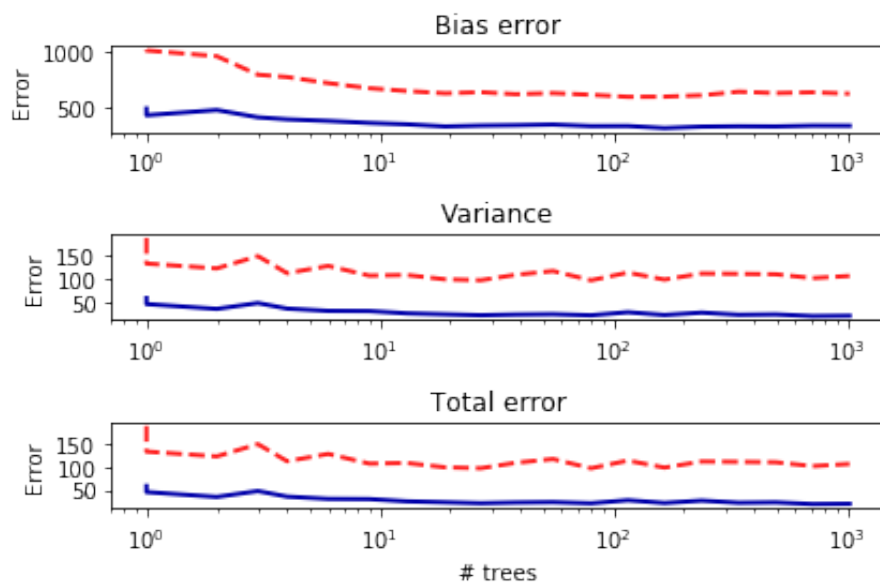


Figure 5:

2

A validation curve can help you understand when a model starts under- or overfitting. It plots both training and test set error as you change certain characteristics of your model, e.g. one or more hyperparameters. Build validation curves for gradient boosting, evaluated using AUROC, by varying the number of iterations between 1 and 500. In addition, use at least two values for the learning rate (e.g. 0.1 and 1), and tree depth (e.g. 1 and 4). This will yield at least 4 curves. Interpret the results and provide a clear explanation for the results. When is the model over- or underfitting? Discuss the effect of the different combinations learning rate and tree depth and provide a clear explanation. While scikit-learn has a validation curve function, we'll use a

modified version (below) that provides a lot more detail and can be used to study more than one hyperparameter. You can use a default train-test split. We added the `clf.fit(X train, y train)` function before the Gradient boosting algorithm in order to make sure estimator was defined. The result is shown in figure 6.

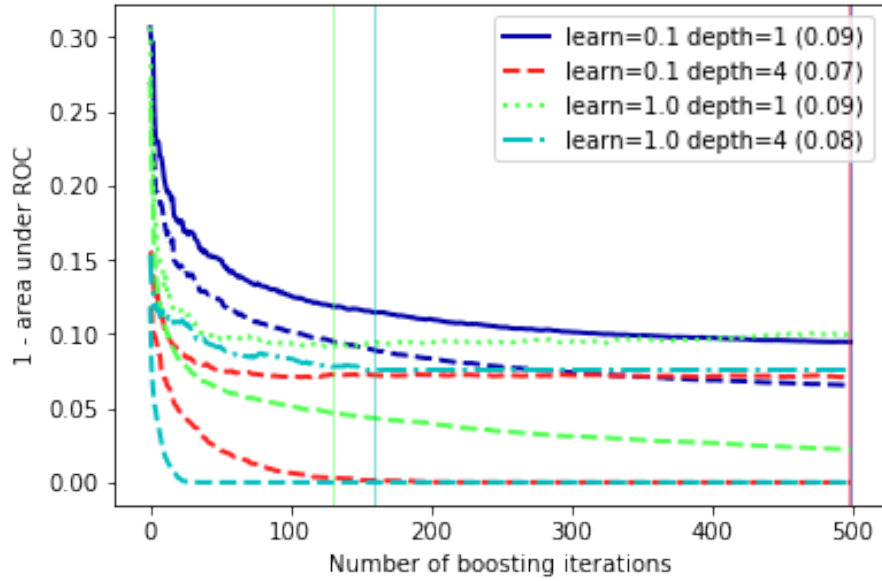


Figure 6: Compute Receiver operating characteristic score for different hyperparameters for the gradient boosting algorithm.

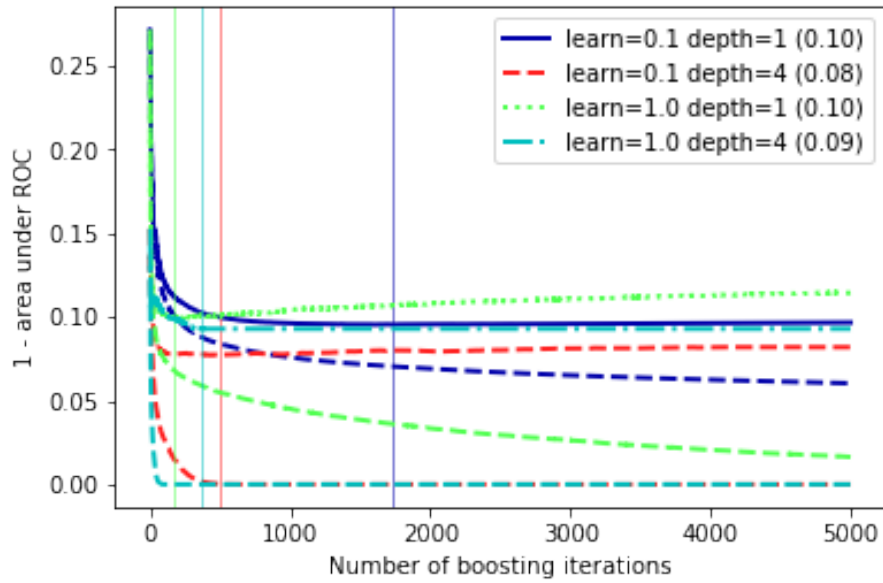


Figure 7: Compute Receiver operating characteristic score for different hyperparameters for the gradient boosting algorithm. The number of iterations is set to 5000 to show possible overfitting characteristics.

1 Appendix

1.1 Code question 2.1

1.2 Code question 2.2