IEOR 242: Applications in Data Analysis, Fall 2021

Homework Assignment #4

October 30, 2021

Problem 0:

Please self-grade Problem 1 from Homework 3. Please use the following rubric to assign yourself a score of $0 - 2$ for each part of the problem:

1. 0 points = Did not attempt or very wrong
2. 1 point = Showed something non-trivial, but some mistakes or some missing justification
3. 2 points = 100% correct

The GSIs will make an announcement concerning how to submit your self-grade scores on Gradescope.

Problem 1: Letter Recognition (Adapted from Bertsimas 22.3)

One of the most widespread and classical applications of machine learning is to do optical character/letter recognition, which is used in applications from physical sorting of mail at post offices, to reading license plates at toll booths, to processing check deposits at ATMs. Optical character recognition by now involves very well-tuned and sophisticated models. In this problem, you will build a simple model that uses attributes of images of four letters in the Roman alphabet – A, B, P, and R – to predict which letter a particular image corresponds to.

In this problem, we have four possible labels of each observation, namely whether the observation is the letter A, B, P, or R. Hence this is a *multi-class classification* problem.

The file `Letters_train.csv` and `Letters_test.csv` contains 2181 and 935 observations, each of which corresponds to a certain image of one of the four letters A, B, P and R. The images came from 20 different fonts, which were then randomly distorted to produce the final images; each such distorted image is represented as a collection of pixels, and each pixel is either "on" or "off". For each such distorted image, we have available certain attributes of the image in terms of these pixels, as well as which of the four letters the image is. These features are described in Table 1. Note that **feature selection is not required** for this problem.

NOTE: For the questions in this problem, provide your numerical answer as well as any necessary code, calculations, or explanations needed to justify your answer.

a) (25 points) To warm up, we'll start by predicting whether or not the letter is "B". To do so, first create a new variable called `isB` in your dataset, which takes value "Yes" if the letter is B, and "No" if the letter is not B.

i) Before building any models, first consider a baseline method that always predicts the most frequent outcome, which is "not B". What is the accuracy of this baseline method on the test set?

ii) Construct a logistic regression model to predict whether or not the letter is a B, using the training set to build your model. (Remember to not use the the variable `letter` as one of the independent variables in any of the models in this part, since it is not a valid feature.) What is the accuracy of your logistic regression model on the test set, using a threshold of $p = 0.5$?

iii) What is the AUC of your logistic regression model?

iv) Construct a CART tree to predict whether or not a letter is a B, using the training set to build your model. Select the `ccp_alpha` value for the tree through cross-validation, and explain how you did the cross-validation and how you selected the `ccp_alpha` value. What is the accuracy of your CART model on the test set?

v) Now construct a Random Forest model to predict whether or not the letter is a B. Just leave the Random Forest parameters at their default values (i.e., leave them out of the function call). What is the accuracy of this Random Forest model on the test set?

vi) Compare the accuracy of your logistic regression, CART, and Random Forest models. Which one performs best on the test set? For this application, which do you think is more important: interpretability or accuracy?

b) (50 points) Now let us move on to the original problem of interest, which is to predict whether or not a letter is one of the four letters A, B, P or R. The variable in the dataset which you will try to predict is `letter`.

i) In a multi-class classification problem, a simple baseline model is to predict (for every observation) the most frequent class over all of the classes. For this problem, what does the baseline method predict, and what is the baseline accuracy on the test set?

ii) Now construct a CART model to predict `letter` using the training data. Again, select the `ccp_alpha` value for the tree through cross-validation. Again, explain how you did the cross-validation and how you selected the `ccp_alpha` value. What is the test set accuracy of your CART model?

iii) Next build a model for predicting `letter` using "vanilla" bagging of CART models. This can be achieved, for example, by setting `max_features` equal to the total number of features (i.e., set $m = p$) in the `RandomForestClassifier` package in Python. What is the test set accuracy of your bagging model?

iv) Now build a Random Forest model, and this time use cross-validation to select the `max_features` value for the Random Forests method. Explain how you did the cross validation and how you selected the `max_features` value. What is the test set accuracy for your Random Forest model?

v) Let us now apply boosting to this problem using the `GradientBoostingClassifier` function Set `n_estimators` to 3300, `max_leaf_nodes` to 10, and leave all other parameters at their default values. What is the test set accuracy of your gradient boosting model?

(Note: If you are interested in more details about boosting, I highly recommend perusing Chapter 10 of "The Elements of Statistical Learning" by Hastie, Tibshirani, and Friedman.)

c) (25 points) Compare the test set accuracy of your CART, bagging, Random Forest, and boosting models from part (*b*). Use the **bootstrap** to carefully test which model performs best. Which model would you recommend for this problem? For your chosen model, again use the bootstrap to construct a confidence interval for its accuracy.

Table 1: Variables in the dataset `Letters`.

Variable	Description
`letter`	The letter that the image corresponds to (A, B, P or R).
`xbox`	The horizontal position of where the smallest box enclosing the letter shape begins.
`ybox`	The vertical position of where the smallest box enclosing the letter shape begins.
`width`	The width of this smallest box.
`height`	The height of this smallest box.
`onpix`	The total number of "on" pixels in the character image.
`xbar`	The mean horizontal position of all of the "on" pixels.
`ybar`	The mean vertical position of all of the "on" pixels.
`x2bar`	The mean squared horizontal position of all of the "on" pixels in the image.
`y2bar`	The mean squared vertical position of all of the "on" pixels in the image.
`xybar`	The mean of the product of the horizontal and vertical position of all of the "on" pixels in the image.
`x2ybar`	The mean of the product of the squared horizontal position and the vertical position of all of the "on" pixels.
`xy2bar`	The mean of the product of the horizontal position and the squared vertical position of all of the "on" pixels.
`xedge`	The mean number of edges (the number of times an "off" pixel is followed by an "on" pixel, or the image boundary is hit) as the image is scanned from left to right, along the whole vertical length of the image.
`xedgeycor`	The mean of the product of the number of horizontal edges at each vertical position and the vertical position.
`yedge`	The mean number of edges as the images is scanned from top to bottom, along the whole horizontal length of the image.
`yedgexcor`	The mean of the product of the number of vertical edges at each horizontal position and the horizontal position.