Economics 1660: Big Data Prof. Daniel Björkegren

PS 4: Prediction

In this problem set, we will work with loans in the Kiva loans dataset. Kiva is a crowd funding platform for microfinance loans in developing countries. Visitors to the site can browse loan requests and select borrowers to lend to. Requests for a loan include descriptions of the borrowers, what they plan to do with the money, the full amount being requested, and so on. Usually each request accumulates many lenders over time until it has been fully funded. You can take a look at http://www.kiva.org/lend to get an idea of what lenders see when they browse the site.

For this assignment, we will use trees to predict the time it takes for a loan to be fully funded by lenders on Kiva: the difference between the posted date of a loan and the funded date of the loan.

The data for this assignment can be found in the file loans_A_labeled.csv. Each line of the file represents a loan that was fully funded on Kiva. The variables are as follows:

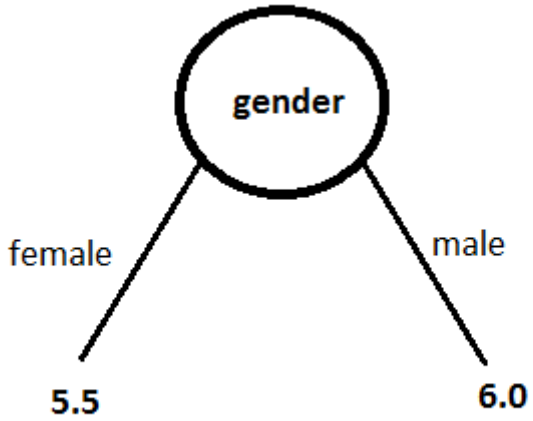
id	A unique identifier for each loan
name	The name of the borrower
gender	The gender of the borrower (M/F)
pictured	An indicator (0/1) for whether or not the borrower is present in a picture that accompanies the loan description
description	A description of the loan and the borrower, made available to potential lenders on Kiva
loan_amount	The amount the borrower is requesting
activity	The activity for which the loan will be used
sector	Business sector of the activity
country	The borrower's country
town	The borrower's town
posted_date	The date on which the loan was first posted to Kiva
repayment_term	The number of months by which the borrower expects to have repaid the loan
languages	A list of the languages in which the loan description is available. Languages are represented as 2-letter codes (e.g. "en" for English) followed by a vertical bar ("|"), which acts as a separator
days_until_funded	The number of days a loan takes to be fully funded after it is initially posted on Kiva

Regression Trees

As we did in PS3, we will use trees for this prediction problem. An important difference to notice, however, is that in this problem set we are dealing with a *continuous* variable

(days_until_funded), as opposed to the binary variable (edible) we were predicting in PS3. We must therefore adjust our approach; we will move from classification trees to regression trees.

A regression tree looks like a classification tree, but with leaf nodes that can be equal to any number—not just 0 or 1. Take the following one-level tree as a simple example:



If this were our tree, we would predict that *days_until_funded* is 5.5 for all female borrowers, and 6.0 for all male borrowers.

The process of building a regression tree is different from building a classification tree in two ways:

1. At each leaf, our predictions may be a number, not a binary classification
2. We may choose a loss function that considers not just whether a prediction is correct but also how far it is from correct (here we'll use mean-squared-error)

Your Task

You will use the stencil code PS4_stencil.py to make predictions on the variable *days_until_funded*. There are 4 functions that you will need to fill in:

- **partition_loss(subsets)**: Since we are building a regression tree as opposed to a classification tree, we need to adjust our loss function from PS3. This function takes in *subsets*, a list of lists of observations (tuples) where the first element in the tuple is the feature dictionary, and the second is the value of *days_until_funded*. The function should turn a value representing the weighted mean squared error of *subsets*. Take the following example:

  partition_loss([[({…}, 1), ({…}, 2), ({…}, 3)], [({…}, 5), ({…}, 9))]]

  subsets has 2 subsets. The first with 3 observations and the second with 2.

We will compute the MSE of each of the subsets and return the weighted (by number of observations) sum of the MSEs:

$$\frac{3}{5}\frac{[(1-2)^2+\ (2-2)^2+\ (3-2)^2]}{3}+\frac{2}{5}\frac{[(5-7)^2+\ (9-7)^2]}{2}=2$$

- **build_tree(inputs, num_levels, split_candidates = None)**: This function is where we will build the regression tree. You may use the same code you wrote here in PS3, but when you get to the bottom level (leaf nodes) of your tree, return the average *days_until_funded* value of all the observations in the data that reach that node.

- **predict(tree, to_predict):** This function should predict the days_until_funded value of the given observation (to_predict) using the given tree. The code for this function may be identical or almost identical to the code you wrote for the *to_classify* function in PS3.

- **load_data()**: Here you should read in the loans data from the *loans_A_labeled.csv* file. Return the observations as a list of tuples (a, b), where 'a' is a dictionary of features and 'b' is the value of the *days_until_funded* variable. See the "Generating Features" section below.

1. **Generating Features**

   Explore the data to get a sense of what it contains. (On a Mac or Unix system you can do this without loading the file into memory using the *cat* terminal command)

   Consider: what information might be predictive? For example, are there keywords in the description of a loan that might indicate a person's reliability or chance of business success? Might there be characteristics of loans that make them more attractive to potential lenders? What other variables might have an effect on the length of time it takes to fund a loan?

   Create a set of numeric indicators or features from the dataset that can be used in a regression model. For example, you could define 'keywordCountPromptly' as the number of time the keyword 'promptly' appears in the project description. As another example, you could create a variable 'english' as a dummy for whether or not the loan description is available in English. Include these features in your feature dictionaries when you are writing the *load_data* function.

2. **Predictive Model**

Create tree models that predict *days_until_funded* using features that exist in the data and that you have generated. Think about which variables might be predictive, and which ones might be best left out.

You may (and should) recycle some of your code from PS3 to build your models. You have done much of the work already, but keep in mind a few aspects of this dataset that are different from the mushrooms dataset:

- **Most of the variables are not binary**. In the mushrooms dataset, we dealt with variables that had values of either 1 or 0. In this dataset, many of the variables are continuous, e.g. *loan_amount*. It is up to you how you choose to handle this. One strategy might be to create a dummy variable called *high_loan_amount*, which gets set equal to 1 if the loan amount exceeds $1,000, and gets set equal to 0 otherwise. It will certainly be helpful to explore the dataset before you begin building the tree. You should try a variety of different strategies.
- **Some of the variables are factor variables**. Factor variables can take on many different non-numeric values. One example is the *location_country* variable. One strategy for dealing with this type of variable could be to create dummy variables for each possible "level" to indicate whether a loan was given to a certain country. Keep in mind, however, that you will want your trees to be able to make predictions on new loans that may be from countries that you did not encounter in the "loans_A_labeled.csv" dataset. If you decide to include factor variables in your models, make sure your trees can handle a values of the factor variables that it did not encounter when it was built.
- **One of the variables is a text project description**. To generate features using the description variable, you may want to create dummy variables to represent whether a certain keyword is found in a loan's description. The easiest way do this in Python is to use *in*. If you have a string *s*, calling *keyword in s* will return True if the keyword is in *s* and False otherwise.

Create at least 5 models using different combinations of prediction variables, different numbers of levels on in the tree, and different strategies for handling non-binary variables. Test each model on the dataset and compute the mean squared error between the predicted values of *days_until_funded* and the actual values.

- **For each model, write a few sentences describing the model (variables, levels, and strategies you used), and record its mean squared error when tested on the dataset.**

3. Turn in Predictions

Of the models you estimated, select which one you feel worked best (judging from the accuracy rate). You will now use your model to make predictions on a new, unlabeled dataset, "loans_B_unlabeled.csv". This dataset is identical in format to loans_A.csv, but the classification variable *days_until_funded* is omitted. Use your model to predict the value of *days_until_funded* for each loan in loans_B_unlabeled.csv. Note that it may take a few minutes for your code to run due to the size of loans_B_unlabeled.csv. Save your predictions in the form of a CSV file. The first column should be the ID of the loan, and the second column should be your prediction (either 0 or 1). In the header row, name the first variable "ID", and the second variable *days_until_funded* followed by the first and last initials of your group members. For example, Professor Björkegren and Simon Freyaldenhoven's group would turn in results of the form:

```
ID,days_until_funded_DB_SF
1,4
2,0
3,7
...
```

Save these as *loans_B_predicted_[*Group Initials Here*].csv* and turn this in on Canvas.