P2P Loan Analytics Project

Overview:

1) Explore the Prosper dataset. We will do some data cleaning and preparation together. We will use 3 year loans only.
2) Build a default model using loans from the early part of the sample. Cutoff dates will be announced in class.
3) Test your default model in more recent data.
4) Prepare a presentation to deliver on 3/26.
5) Submit a 3 page write-up (not including tables/figures), your presentation slides, and your Jupyter notebook on 4/1.

To include in your write-up:

1) How does your model perform in sample and out of sample?
2) Which variables are most important in your model?
3) What key modelling choices did you make?
4) References.

To include in your presentation:

1) A brief description of your algorithm and modelling choices.
2) Model performance and variable importance, in-sample and out-of-sample.
3) What would your next steps be if you were to continue this project?

Detailed Timeline

1) 2/27 – 3/12: Explore dataset and the Prosper website, read about your algorithm choices.
2) 3/12: Choose algorithms for projects in class. See the announcement section on Canvas for details of an “early selection” option, email choices to professor on 3/4 if participating.
3) In class on 3/12 and 3/19 we will do some data analysis, data cleaning, and example models together.
4) By the end of class on 3/12, we will have a final set of data cleaning guidelines/code to share. But some teams will probably want to go farther in “feature engineering” for variables of specific interest.
5) 3/12 – 3/19: Decide what variables you want to use, complete custom data preparation, estimate a first draft of your model in the early part of the data and assess in-sample fit.
6) 3/19: Resolve any problems with draft models in class. I will also demonstrate how to assess the performance of the example models out-of-sample in more recent data.
7) 3/19 – 3/26: Assess performance out-of-sample, prepare presentation.
8) 3/26: Deliver presentation
9) 4/1: Submit write-up, slides, and notebook on Canvas.

Notes:

1) I realize the algorithms may not all have the same degree of difficulty. If you have one of the easier algorithms or one we started for in-class examples, you should be able to do more in terms of thoughtful variable choices, feature engineering, and model tuning. If you have one of the harder ones, it is fine to just get it to run with some default settings. But be prepared to share what the challenges are in that case.
2) Given the limited time we have, it is fine to treat these algorithms as "gray boxes." Meaning you should have a high-level understanding about what the algorithm does and the most important tuning parameters available, but you are not expected to become an expert in the underlying details.
3) My advice is to get a crude version of the model running with default choices and variables that need little or no modification as soon as possible, and then worry about improving it once it works minimally. In general, there is a Python function or set of functions that implement your algorithm. Focus on just getting the data into that function in the proper format and getting results out first.
4) Choose one in-sample model after the in-sample analysis and stick with it for your out-of-sample tests. This approximates/simplifies a real-world process where you have made your best decisions at the end of training period and have to live with it before you get to see the OOS data. In other words, you should <u>not</u> retrain your model with the OOS data, assume you have to make decisions using your model before any of the OOS data is revealed. And do not confuse your writeup/presentation with comparisons of multiple OOS models.
5) I am not specifying the performance metrics and graphical output to present, so you can use the simplest reasonable ones to get out of your algorithm.
6) You are not graded on the performance of the algorithm.
7) Submitted notebooks should be runnable by grader from end-to-end for full credit. The only modifications they should have to make are possibly installing extra packages and replacing paths to data files.

Algorithms:

Logit: statsmodels formula.api.logit()

CART: scikit-learn tree.DecisionTreeClassifier()

Random Forests: scikit-learn ensemble.RandomForestClassifier()

Gradient Tree Boosting: scikit-learn ensemble.GradientBoostingClassifier()

SVM: scikit-learn svm.SVC()

Neural Networks: scikit-learn neural_network.MLPClassifier()

k-Nearest Neighbors: scikit-learn neighbors.KNeighborsClassifier()