StateFarm Data Science Case Study

Austin Collins*

We are tasked with utilizing 32 potential data fields to create two conceptually different models to predict the interest rate assigned to a loan. We are provided approximately 400,000 records of previous interest rate assignments with which to fit and/or train our model. Further, we are also tasked to fill missing values in the training data, to simulate 'cleaning' dirty data. The predictor variables include scale, ordinal, and nominal variable types. We choose Stochastic Gradient Descent (SGD) and Random Forest regressions as the primary bases for our models, employing elements of robust covariance types and Generalized Linear Estimators along the way. And finally, we provide guidance on how to combine these two models for improved predictions via Bayesian Model Averaging.

OVERVIEW

In this document we discuss and compare the two primary models we employ: Stochastic Gradient Descent SGD and Random Forest Regression RFR. Further discussion on topics such as feature selection, collinearity, heteroscedasticity, and outliers are also discussed in sections following the main analysis.

Similarly, our solutions to the broken code found in BasicFunctions.py is also included as a following section, as are details on our Python environment and our approach to data cleaning and processing.

GENERAL CONSIDERATIONS

Much as one's favorite food may depend on mood, we believe model selection should be driven by the characteristics of your data set and your ultimate modeling goal. Indeed, a good indication one has a "'favorite"' model is effort spent cramming data into the model; model selection should accommodate your data to the extent possible.

As such, our selection of distinct models will necessarily refer to many of the same characteristics of our data set – though their approach to modeling it will be quite different. Rather than reference them in our description of both models, we begin by identifying here the characteristics of our data set:

- Comparably large number of records

  A large number of records on which to fit/train compared to the dimensionality of our data (including the extension of dimensionality from the presence of ordinal and categorical variables) reduces concerns about over-fitting. While this removes some model options due to computational constraints (for example, Nearest Neighbors based modeling), it enables other techniques that require a greater degree of cross-validation or benefit from optimization over a large parameter space.

- Presence of many ordinal variables

  Indeed, the bulk of our predictors are either categorical (such as State) or ordinal (such as loan rank). Some variables that naively seem to be scale variables (such as number of delinquencies) are in fact ordinal – as there is an artificial floor, ceiling, or strong categorical difference. As an example, the number of months since last delinquency cannot be scale outside of a group that only includes records that indicate there has been a delinquency.

  This greatly expands the dimensionality of our predictor space, but also guides us in our selection of dimensionality-reduction techniques.

- Non-normal regressand and predictor distributions

  Of the scale values we have available to us, few are normally distributed. Importantly, our target regressand – the quantity we ultimately wish to predict – is itself highly non-normally distributed[1]. Were they clearly of a particular distribution, this might be useful. In the instant case, however, both the shape and nature of the regressand distribution does not narrow it down much. Poisson, due to the low values and positivity constraint? Log-normal, as a product of factors? Negative Binomial, reflecting choices between different pass/fail categorizations? Gamma, because almost anything can be Gamma with the right parameters?

  The practical result of this is guidance in our choice of metrics, models, and dimensionality-reduction methods that are not essentially predicated on normality. Indeed, this is one reason we ultimately decide on hierarchical clustering-based dimensionality reduction methods (in this case, feature agglomeration via Ward's algorithm) over feature selection methods such as Randomized Principal Component Analysis (which operates on correlation/covariance matrices), though there are several other reasons we make this choice, which we discuss later.

- Empirical, functional goal

  Our goal is simply to predict an interest rate. This is a functional, empirical task as opposed to a more

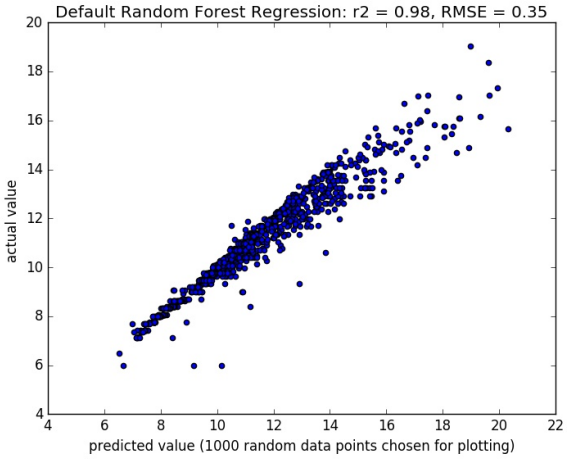
exploratory or research effort to 'understand' the structure in an underlying data set or to identify new areas of inquiry. This does not mean no understanding of the data is necessary for the task, simply that exploratory methods tend to favor identification of underlying patterns rather than utilizing them. This is another factor that persuades us – over the voice of our inner theorist – to employ feature agglomeration rather than PCA. Even though both methods ultimately end up identifying a reduced basis of eigenstates, agglomeration does so as a consequence of clustering, rather than the fundamental purpose of the algorithm as is the case with PCA.

FIRST APPROACH: RANDOM FOREST REGRESSION

The large number of central categorical predictors, the likely high dimensionality of even our post-reduction data, and empirical nature of the goal immediately suggest an ensemble method. Of the ensemble methods, the graph-centric approach of Random Forest Regression (RFR) is a natural (though not necessarily optimal) approach to the influence of our categorical predictors and is also straightforward to implement.

Though we are aware of the existence of Robust Linear Methods employing robust norm and covariance structures (our personal favorite being the draconian Tukey's Biweight), the scalable number of separate, randomized estimators in RFR is immediately appealing in cases where collinearity and non-normality are certain, heteroscedasticity likely, and options for thorough for cross-validation are limited (too many independent cross-validation sets would bring the dimensionality problem into play).

Further, RFR is a more natural extension of our decision to employ feature agglomeration rather than feature selection given the contrast between "top-down"' and "bottom-up" approaches. Taken together with RFR's comparatively limited parameter space, we naively expect it to "just work" – that is, we expect a non-terrible result with the default parameters and no modifications to our default data set. Arbitrarily picking 100 as the number of clusters to consider (under 1/3 of our total dimensionality), we see this is disturbingly true:



Indeed, we spent quite a bit of time confirming that this result was valid. One hesitates any time a model seems as if it is a "win button". Other figures for this model are included in the appendices, but the primary re-assurance we took was from two factors that are not included in the published code – but easily can be should you wish to see them. They were 1) consistency over randomly sub-sampled cross-validations (though with reduced accuracy to be expected from partitioning our training set), and 2) the monotonicity of results over the parameters space of cluster size and number of estimators. Both factors strongly suggest a sufficiency in dimensionality reduction, model choice, and data characteristics that provide confidence in the accuracy of the model.

Of course, once one has convinced one's self that the results are genuine, one is free to post-facto claim it was the truly impressive insight into the field of data science that led to a model selection that ultimately optimized to... a graph looking essentially the same, but

achieving an RMSE of 0.31

Not having anticipated such a robust and immediately relevant result, we had already coded methods of grouping by presence/absence of features such as public records and delinquency. Our intention was to model dominant sub-groups of data, then utilize the most specific (given that the test data has no missing values). For example, in the test data we had expected to have a different model for people with no delinquencies, and people with delinquencies with which we could then treat the time of the last delinquency as a scale variable and gain valuable additional information. This proved to be unnecessary, however, so is not included in our published code. Once more, should you wish to see the code and results of this sub-group analysis, simply ask and I will provide them.

One can explain this in terms of RFR's natural tendency to identify and incorporate different groups, but this feels more akin to a theorist's "'hand-waving"' than an actual insight. So, without further commentary, we proceed to a far less straightforward method...

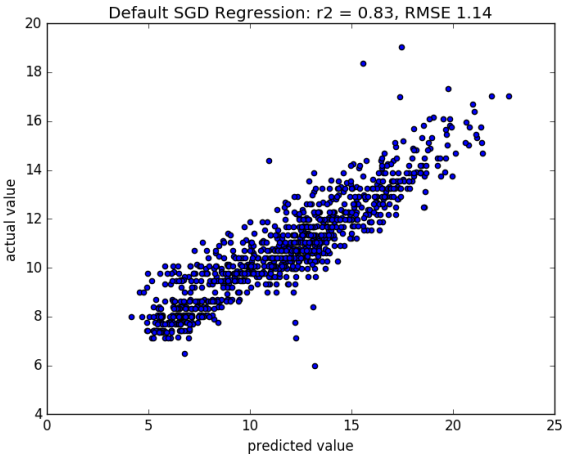
SECOND APPROACH: STOCHASTIC GRADIENT DESCENT

If one believes that there is already a previously-established approach to the problem at hand, Stochastic Gradient Descent (SGD) is a solid first choice for model. The truly expansive nature of its parameter space enables one to leverage the ever-increasing computational resources available to scan parameter space for the "correct" (or at least a functional) approach to the problem. One can iterate over types of norms (squared distance, Huber, distance-insensitive, or even squaring your distance sensitivity), regularization types (L1, L2, elastic net, or even none), regularization values (the seemingly omnipresent α), and number of iterations – in addition to occasional additional parameters depending on your previous choices (for example L1 ratio for elastic net).

If it can be adequately (though not necessarily optimally) represented in a linear model[2], then SGD is likely able to find it for you. Ahem, we meant to say that one may succeed with a wise/clever/judicious choice of parameters.

However, one must also be aware of the adage: "For every question there a simple, easy to understand answer – that is completely wrong." Exploring/abusing the phase space of parameters extensively requires greater cross-validation efforts. This is but one reason among many that SGD is only formally appropriate for large-ish to huge data sets.

Our naive expectation here, then, is that SGD will produce less impressive "out of the box" results than would a well suited, but limited in parameters, model such as RFR. Once more, this appears to be borne out by running SGD with the default parameters:



Another important distinction between this approach and the Random Forest regression approach is sensitivity to scaling. SGD is sensitive not only to the standardization (zero mean, unit variance) of both the regressand and regressor, but can also be influenced by steps prior to your standardization. For example, it is suggested by the authors of the SGD algorithm we employ that if you employ PCA or clustering that a separate scale factor should be added to ensure the L2 norms are identical.

Further, the comparative utility of "pseudo-ordinal" variables differs as well. For example, if a variable is only "technically" ordinal and behaves much like a scale variable when it is present, SGD sometimes benefits from either subgroup analysis in which one group assumes it is scale or a transformation into a quantile-regressive ranking. Further, SGD suffers more from missing data (on average) than does Random Forest regression.

As such, it is unsurprising that our optimized SGD regression achieves only

a RMSE of 0.071.

We do note that such quantile transformations could have been employed, as could have utilizing the set of training data with no regressand present to "fill in" the missing income data. Indeed, we supply the routine we would have used to do so. However, we did not ultimately employ it as the Random Forest regression performed so well.

LIMITATIONS OF OUR APPROACH(ES)

In both analyses we make a crucial, central assumption – that the characteristics of our training data set and test data set are essentially identical. This approach would likely fail spectacularly when this is not the case – for example, if our training data was from 1980 and our target predictions were for 2030.

ON DATA CLEANING, AND ITS SURPRISING INUTILITY

As we often do, we began our coding with analyses of the data set aimed at "filling in" missing data points. [3].

The important metrics in this case were the test data columns that were missing values. Discounting purely string fields such as "reason for loan provided by user", the count was:

```
348845 X26,Number of months since the last public record
276439 X16,Reason for loan provided by borrower
218802 X25,Number of months since the borrower's last del
61028   X13,Annual income of borrower
61010   X1,Interest Rate on the loan
```

The top 3 were all fields for which a null value is entirely natural. Should an applicant never have had a delinquency, for example, zero is not the appropriate value – null is. The next two values are similar in size, but very different in context.

Loan applications missing an assigned interest rate – our sole regressand – cannot be filled in without unacceptably influencing our training mechanisms. However – and importantly – that does not mean that data is useless. One may still utilize the records missing a regressand either via unsupervised learning for PCA or clustering, or by developing an independent regression model to "fill in the blanks" for other records that have a regressand but are missing other values.

The natural target for an auxilliary regression was income, which was missing in over 60,000 records. Another natural target would have been loan rank/subrank (obviously, via a categorization rather than regression model). Despite loan rank/subrank being a far stronger predictor of regressand than income, its appearance in only 267 records was unpersuasive.

Thus, we developed a regression model – accurate to less than 10 percent – to replace the missing income data. It was only after we had done so that we discovered what a weak predictor income was. Thus, we include the code for that regression, but did not employ it in our final predictions.

NOTE: Loan ID and Applicant ID also had numerous missing values. However, they served no computational purpose, so we ignored them. DataFrame index was a perfect substitute for Loan ID, and the only use we could conceive for Applicant ID was in filling in missing fields should the same applicant have applied for multiple loans. However, every Applicant ID was unique, rendering it also useless from a computational perspective.

All in all, we ended up not employing much "data cleaning", or substitution of missing values, as their impact on our final analyses was negligible.

ON OUTLIERS

Similarly, we did not correct for outliers in a deliberate fashion. We believed, and this is confirmed by our scatter plots, that both models we employed are naturally "robust" against outlier influence. In the end, we are not much concerned – in this case, graded by RMSE – with under- or over-estimating the interest rate assigned to the one person with an income exceeding 7 million dollars, [4] when the mean was so very much lower.th of that.

ON THE PYTHON ENVIRONMENT WE EMPLOYED

The primary tools we employed were

Pandas: primarily for its wonderful DataFrame structure. Though we originally anticipated employing it far more substantially – particularly it's group by and panel features – the need never arose. Should an examiner of our code wonder why several binary indicators were added during data processing, anticipating the need to group/empanel by those factors is the answer to that question.

StatsModels: though it did not feature in our final analyses, ad-hoc exploration of data via straightforward, and less straightforward, Ordinary Least Squares linear regression is an irreplaceable part of our arsenal. StatsModels was the tool we used for this, as well as for easier saving/loading of data structures. Note that StatsModels also requires Numpy, so we do not include Numpy separately here.

Patsy: though we are not ourselves a rabid devotee to "R-style" equation programming, the ease with with Patsy simplifies the creation of design matrices – as well as it's terrific categorical encoding options – make this a necessary feature for us.

Sci-Kit Learn: a fantastic set of routines, with unparalleled documentation.

Anaconda: Though Anaconda is its own, self-contained, Python environment – complete with its own IDE, shell, and update system – we employed it simply because we're developing on a windows machine, and building the BLAS/LAPACK libraries on our own would rob us of the substantial performance improvements brought by the Intel MKL libraries.

Python: We're using 3.5. Though we're not fond of 3+, it pays to stay ahead of the curve in terms of familiarity, so if we encounter someone who requires we use it we are already familiar. Clearly, from the broken code solutions, we are also comfortable with 2+ Python as well.

---

* Collins.Austin@gmail.com

[1] Thankfully so! One would be hesitant to finance in a world where interest rates could be normally distributed, unlimited by a positivity boundary.

[2] We note the many non-linearity options available, via changes to design and contrast matrices, in today's putatively linear models.

[3] It was merciful to include only missing data as "dirty data" – this is typically not the case and validating data to determine if it needs to be adjusted is far more cumbersome.

[4] We were far more interested in the person who provided "ilikehorses" as the reason for his loan!