Description of Project and First Steps

November 17, 2023

1 Introduction

The ultimate vision for this project is a web based interface (ideally point and click where appropriate) that assists with every step of the data science process (for lack of a better expression). Namely:

1. Initial cleaning and preparation of the raw data
2. First investigations of the data (think univariate statistics in SAS)
3. More complicated investigations into the data, e.g. multivariate analysis, principal component analysis, etc.
4. Building visualizations
5. If necessary for the particular project a model development suite. (To be clear I want this included, I just recognize not every data project involves model development.).

The way I see it, each of these can be thought of as separate ”suites” of a larger application, and can be developed more or less independently. I don't necessarily think this will be a ”finish one section and move on to the next section” type of project either, and I anticipate there will be lots of back and for changes as we learn things like what format we want the data to come out of step 1 in, or maybe save visualizations for the end etc.

In the end, I think having something built like this will take quite a bit of time to do properly, but will make it MUCH easier to analyze data, develop models, and create impressive visualizations. In theory if anyone wants or needs to analyze data (or just wants to look at some for fun) this tool will make it very easy to do so. Things like formatting or picking variables to plot in matplotlib can be tedious and this will eliminate a lot of the hassle.

Now for a little more detail about each of the steps, followed by some next steps.

2 Initial Cleaning and Preparation

The most important part of any data project BY FAR is the cleaning and preparation of the data. Data cleaning is essentially a never ending job and is paramount to the success of the project. Anywho, with that out of the way, here are my initial thoughts for what the process might look like:

1. Take in a raw dataset. Just to get the ball rolling it should able to accommodate a pandas dataframe, a numpy array, and a regular old python array. Probably couldn't hurt to have it able to accommodate text files and csv's either.

2. Run through a list of standard data cleaning prompts for each column. Note these prompts may differ based on the desired column type (date, number, name, etc.). It's important that this list of prompts is easy to alter or modify. The final version would ideally be point and click (e.g. "Do you want to remove apostrophes from names yes/no")

3. Allow a round of user driven data cleaning to account for any additional and potentially idiosyncratic steps. For example, consumer financial data typically includes FICO scores, and at KeyBank 999 was *could* be used to indicate a missing value. A user may want to remove/alter values based on range. Not 100% what this may look like.

4. Push the data to a sql server somewhere. I don't think it's a bad idea to save the initial raw data somewhere, and I also think it's vital to store all of the cleaning steps that were performed. This was a major source of replicability issues in my modeling days.

We'll have to find the right balance to strike between (2) and (3), and I'd like to make the list of "standard" techniques we prompt to the user easy to modify. As we get more ideas for I could see this list changing a lot so it needs to be dynamic in some sense.

Also the open ended cleaning on the part of the user will be very important. As I mentioned above, there will often be project specific cleaning that's required and we need to allow for that. As a first pass, I think the ability to remove or alter rows based on arbitrary criteria (simple conditional statements or even regex removal) is a must, as is manually removing rows. Sometimes you don't really know what you're looking for until you see it, and it might be hard to describe. I think a user should be able to click on a row and drop it from the dataset. And to conform with the "save the cleaning steps" paradigm, it could prompt them to enter why it's being dropped and then that would get stored wherever the cleaning steps get stored.

As a final note, it was not uncommon at all in my modeling days to find some random data quirk six months into modeling and have to start over completely. If your data is big and messy enough it's almost inevitable. To make your life as easy as possible you need to be able to easily make changes to the data and then have them flow through your process again. I think it's very likely that the next step will result in the person realizing additional cleaning is required. We'll want to design things in a way so it won't be difficult to come back over to the cleaning suite and make additional cleaning steps.

3 First Investigations

Generally speaking this step is really for the modeler to get an initial feel for the data. (Of course this step kind of happens along side the cleaning step, but for now let's think of it as a subsequent step). As crazy as it sounds, the first thing I would do is physically look at the data and then compute some basic summary statistics. Typical things I was looking for:

- Are the variables the type I'm expecting them to be? For example sometimes something like account number might contain lead zeros so it's not actually a "number" but a string of numbers.
- Are there lots of missing values? There's nothing worse than seeing a field in a data dictionary that looks useful, and then looking in the data and it's almost completely missing. If too many values are missing it's generally worthless.
- What format are date variables stored as? Sometimes it was YYYY-MM-DD, others it was MM/DD/YYYY, others it was MMDDYYYY and one time we had an Australian consultant working with us who made it DDMMYYYY. It actually caused a lot of issues.
- What does a typical value look like? At Key Loan-to-value was a percentage and stored as a float, but when you looked at it it would say "82.7" instead of "0.827". Even worse, we had multiple loans servicers over the years, and some of them DID write it as "0.827". We had to look for values less than 1 and multiply them by 100.
- Do I understand what the values mean? It wasn't uncommon to see "customer state" and it could either be CO for Colorado or D for delinquent.
- What's the distribution of the variable? We had multiple different credit scores and somehow no one had any idea what the appropriate values were for all of them. Sometimes it helped to look at the distribution and see it looked "normal" between 300 and 800, then there was a gap, and then there were many values in the 900s. It turns out those were all different flags and needed to be removed.

This is all that's coming to mind for right now, but it's really important to have a good grasp of your data and all the components. I think the best way to approach this is let the user click on a particular field, and the screen will display a few things based on the type:

- Selected Percentiles (numerical)
- A handful of random values to see what it looks like (all types)
- Most common values (all types, maybe not floats)
- Histogram (numerical)
- Mode and maybe counts of top X most frequent variables (strings, categories etc.)
- Missing percentage

These things will be conditional on the variable type, as well as some other aspects like how many distinct values does the variable have. I think this would generally be a good start for poking around though.

4 More Complicated Investigations

Surprisingly I don't actually have too much too say about this at the moment. A big one from my modeling days were bivariate plots with both dependent and independent variables. At a bear minimum users will need to be able to select two variables and see basic regressions or correlations between them. For model development you don't want two of your covariates to be highly correlated. You also don't want to include variables that have no discernible relationship with your dependent variable (where applicable).

Additionally, it wasn't uncommon to experiment with transformations at this stage. Two great examples that come to mind:

- FICO Score: Generally speaking default rate vs. FICO would go down until about 740, at which point it would level off. The idea is that once people are over 740 they really aren't better or worse borrowers and that reflects in the data. We would add a boolean variable to indicate if the FICO score was below this point
- $ln(x)$ transformations! It may not sound like it but there are extremely common and actually very applicable in the real world. I'll spare the math, but they take you "from the multiplication world to the linear world". Sometimes you'd see something like "when you increase FICO by 30 points default rate gets cut in half", and so taking log of your default rate make these variables linearly related.

I've also been obsessed with principal component analysis and clustering analysis lately and this seems like the right stage to do that (as well as any other applicable machine learning techniques). I know the python libraries can do the heavy lifting here so it won't be too difficult to implement them (I've also built my own algos for both in case we aren't satisfied, no big deal) and I do think these are both very modern ways to probe your data.

For the actual user experience I'm envisioning a few simple buttons indicating the type of analysis and any additional arguments required (such as the number of means for k-means clustering or k-nearest neighbors). This is definitely a bit down the road though so I'm not too worried at the moment. These also segue nicely into the next section...

5 Visualizations

This is the component that I think will be the most challenging but also the most important. Visualization is king with data, and we need to be able to have convenient and fast visualization capabilities at a minimum. High level, here are my plans:

1. Use available python graphics packages to create all visualizations. Matplotlib can handle basically everything and will make it easy to get graphs into the application. Some other features will require other visualization tools (like I know decision trees have their own means of showing the decision tree) but off the shelf stuff will be fine to get the ball rolling.

2. Once we have (1) handled, consider using seaborn for cleaner graphics or even making our own graphics packages where applicable. Building decision trees is a great example of something I wanna jazz up. The off the shelf visualization of them is fine (the tree structure is obviously necessary) but they're very recognizable and kind of bland. I can immediately notice it when people insert matplotlib plots into their work (whether it's blog posts or youtube videos) and it might be nice to have a unique presentation.

3. This could be done during or after (2), but I eventually want the visualizations to be animated. For example, if you're plotting variable X against variable Y, and then you switch our variably Y for variable Z, I want the graph to smoothly transition from what was showing to the the new figure. The first two methods would result in something more akin to changing slides in powerpoint. The youtube channel 3blue1brown has phenomenal animations and is built on an opensource library called manim (or math animation or something like that) which could serve as a good starting point.

In the near term I'm only interested in 1, but longer term I REALLY want to get the visualizations to look good (and potentially even unique to this application).

6 Modeling

I'll come right out and say that as of the writing of this document (11/16/2023) this is least urgent part of the project in the initial stages. Of course it's still a key part of the final product, but I don't mind if we only get some very basic functionality in here during the "v1.0" stage. With that said...

It's a bit cliche but the expression "modeling is more art than science" is quite true and it can be tough to really pin down certain aspects of the model development process, but for my money the objective pieces (and how we can think about it in an interface) look something like this:

- Select model type (logistic, multivariate linear, random forest, xgboost etc.) It's up to the user to pick the right one.
- Select variables. This is easily the most "art" area of the process and needs to allow for both algorithmic and manual selection of variables. For example, in my modeling days we ran an algorithm in SAS called "best subsets" which would select the best N subsets of a given set variables using a selected criteria (such as R^2 or Akike Information Criterion). You could tell it how many variables each subset needed to have, and also specify a list of variables that had to be in there. Even though this sounds perfect, oftentimes you'd look at the best models and have to throw them out for various reasons and start over with extra refinements (e.g. "maybe I shouldn't include DJIA value in my default model"). Once again, I think the scikit-learn library and others will do most of the heavy lifting, but we should still make it convenient to "play around" at this stage.
- Asses model performance. There are a number of ways to go about this. In sample fit, out of sample fit etc.
- Visualize model results.

I'd really like to hammer home that modeling (at least in my experience) can be very iterative and this "suite" should make it easy to do that.

7 Conclusion

I imagine this project will take a while, but I think it will be a tremendous growth experience. The segmented nature of the different sections will allow for development in parallel, and most of the "suites" offer a multiple levels of refinement, as does the project itself. My mental list for the order we should do things:

1. Get each section built in a standalone interface. Need to identify inputs and outputs of each one.
2. Wrap all of them up in one cohesive application
3. Get a working prototype that can run on a local machine

4. Get the prototype set up in a web interface. If there is any way to do something like have a website "read" the code from our github main branch so that as we make changes to that it flows through to the website that would be ideal. At this point there should at least be basic functionality to each section.
5. Go back and add extra functionality to each section. This is where we will really flesh out the features of each suite.
6. Make major cosmetic changes. This is important, but low priority in my opinion. Personally I want to spend a ton of time fine tuning the looks and whatnot, but that's way down the road.

To this end....

8 First Steps

All progress is good progress. In my mind, the following would be good starting points, but anyone can work on whatever they like. I think the best order to begin:

1. Someone put a script in the github that pushes a pandas DataFrame to a SQL server, and also pulls data from that same server. I have no idea how paying for SQL servers goes and don't want to stick anyone with a bill (is it just memory based or access based) but I'd like to see how we are doing this. Even if it's just a dummy dataset with a few rows of made up data, I think it's best to start familiarizing where we want to put and pull data from
2. A super basic example of a python application. I only know how to run python code in an IDE or terminal and don't know where to begin with building a standalone application. Would be helpful to see what it looks like and how to build basic interactivity.
3. Build a python app that can display a pandas dataframe (can be anything for the time being) and allow for BASIC interaction with the data, as well as pushing the data to the SQL server. Pushing to the SQL server can come second, but being able to make additions and changes to the dataframe through the app is most important. This will essentially serve as a crude prototype of the cleaning suite.
4. Find a dynamic way to list to-dos that we can all access.

Once we have those 3 I think it will make it much easier for people to work on other sections. These feel like the "tutorial" missions in some sense and once they're done we can start building the basic variable explorer and fleshing out the cleaning piece more.