Explorations in ML Part I: Predicting Orientation

Github Repo: here

Data Source: here

What this is >>> *Part I in a series of applying fundamental ML techniques to explore data and make predictions. This first installment is a classification exercise that serves as a capstone project for Codecademy's Data Science: Machine Learning Specialist career path program. The project requires students to use a Codecademy-provided OKCupid dataset, but grants them leeway in how they work with it. While not entirely in my realm of interest, this project did serve as a proxy for exploring domains that DO interest me, namely queer online dating (or, to be frank, hookup) products such as Grindr, Scruff, and Recon.*

This document and the links in it summarize my exploration, model-building process, and takeaways..

Codecademy Prompt

In recent years, there has been a massive rise in the usage of dating apps to find love. Many of these apps use sophisticated data science techniques to recommend possible matches to users and to optimize the user experience. These apps give us access to a wealth of information that we've never had before about how different people experience romance. In this portfolio project, you will analyze some data from OKCupid, an app that focuses on using multiple-choice and short answers to match users.

The purpose of this project is to practice formulating questions and implementing machine learning techniques to answer those questions. However, the questions you ask and how you answer them are entirely up to you.

TL: DR >>

1. **What: The goal was to see whether we could predict users' sexual orientation from their OKCupid profiles. The data ultimately did not provide a strong signal for orientation, with the highest-performing model achieving a joint accuracy and recall of 74% but a precision of only 36% at the same threshold. (Much more on thresholds below)**
2. **Why: The dataset contains orientation labels, which stood out as the only target of any interest.* I chose this label to practice supervised classification ML**
3. **How: Building and tuning classifiers to predict orientation using a combination of numeric (e.g., age) and textual (e.g., paragraph summarizing interests) user profile information**

**I'm not going to argue that predicting orientation for a platform that allows users to disclose orientation has much business value. However, the dataset does not include labels for any business-relevant outcomes, such as revenue/user, retention, satisfaction, or success rate. As such, I chose to focus on an area of personal interest as a queer person. For context, the 'example' preview option that Codecademy shows is a classification of star sign.....Lastly, predicting orientation DOES have value in other contexts, such as recommendation engines, where orientation is not disclosed but relevant to a user's behavior.*

Data Processing

Codecademy provided a single CSV file containing 59,946 entries and 31 columns, each representing an individual user. The set came in a relatively 'clean' state with no detected duplication issues. With some exceptions detailed below, the dataset displayed almost no erroneous or 'outlier' values. No changes to the level of aggregation were required (row = user). Transformations to the feature sets are listed below:

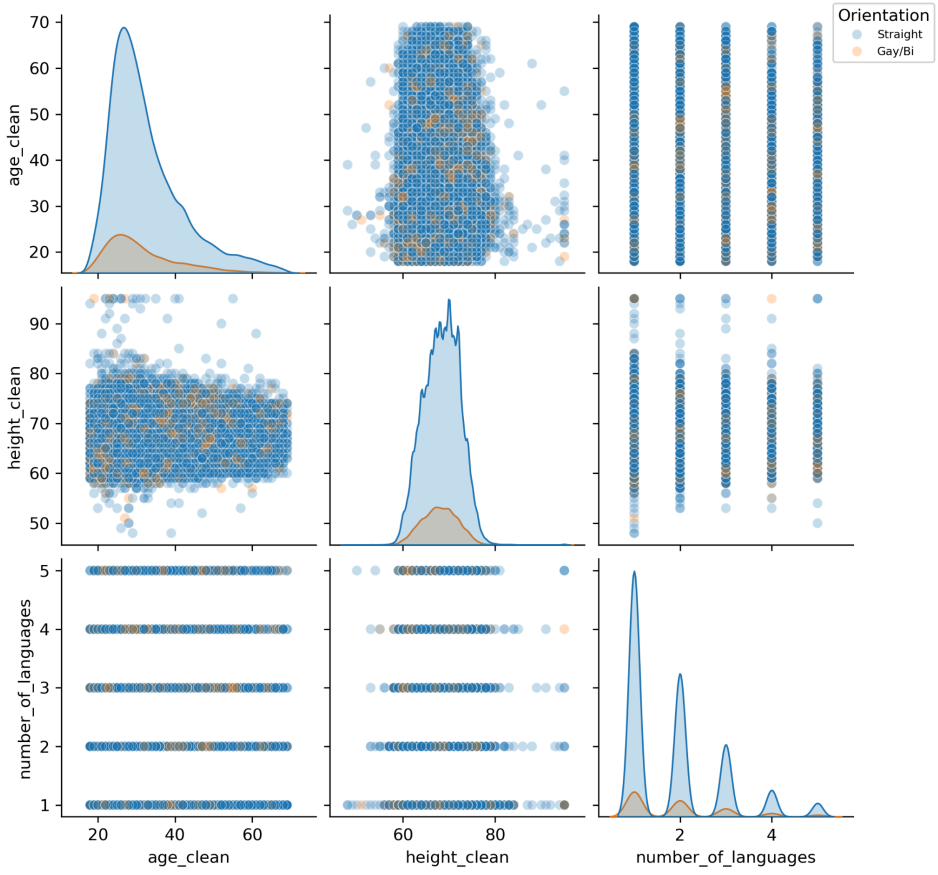
- **Target transformation:** The orientation category provided users three options to choose from: (1)Straight, (2)Gay, (3)Bisexual. I simplified the target into two categories: (1)Straight, (2)Gay or Bisexual, in part due to the low number of users who selected bisexual. Just as importantly, I also assumed that the signals for gay and bisexual were similar and was ultimately more concerned with predicting 'not straight' users than more nuanced identities.
- **Dropping Income:** The most noteworthy data issue was apparent in the 'income' field: most users left it blank, had a value of -1, or had a value that appeared to be heavily rounded to the nearest 10,000. Given the lack of metadata to explain how the values were generated and the number of missing or ambiguous values, I dropped the feature before the model-building process.
- **Dummifying categorical variables**: The bulk of the data transformation work involved creating a series of dummy variables from standardized categories that users chose to describe themselves. In some cases, no underlying changes to the category were required (for example, body type). For many others, however, the categories described more than one set of information, that is, more than one category. In these cases, the categories were split into separate fields before being converted to dummy variables.
    - *Assumptions*: Across all categorical fields, I assumed that a null value signaled that a user chose not to provide an answer. Since not disclosing information is a relevant user behavior, these records were preserved, with null values set to 'Not Selected.'
    - *Has vs. Wants and similar splits*: OKCupid allows users to capture different aspects of themselves within the same field. The offspring dropdown, for example, enables users to select whether they have children AND if they want children at the same time (e.g., 'Does not have kids but wants them'). Having and wanting children are two distinct signals and should be treated as separate features. Similar field structures were observed across the dataset, such as the pets field (Having vs. liking pets) and the religion and diet fields, which provided qualifiers for the chosen category ('Has cats and likes dogs). In each of these instances, I made a separate field to capture a single fundamental aspect of what the user was revealing about themselves. I then converted the resulting standalone features into dummies.
- **Simplifying Education:** The data show that OKCupid offered 32 unique education dropdown options for users to select. Those options often drilled down to a level of detail that I assumed carried little signal (for example, distinguishing a user with a law degree from one with any graduate degree). I simplified the variable to reflect only the highest level of education a user marked themselves as having, assuming that the level reached is the most pertinent information the field captures (Users could discuss their occupation and interests in other parts of their profiles).
- **Note on Race and Languages**: Users could select any number of languages spoken and races/ethnicities. I did not use every unique *combination*. Instead, I created separate fields for

each unique language and ethnicity a user could have selected to capture the exclusive impact of each language or race/ethnicity.

- **Dropping Location and Last Login**: All users were located in the Bay Area. Due to the lack of variation across location, I did not use it as a feature. Similarly, the dataset came from users active in 2012. No other variables captured user behavior with respect to the OKCupid application itself beyond Last Login. I assumed that Last Login, by itself, had no usable relationship with orientation and did not include it in the modeling.
- **Building a vocabulary**: In the latter stages of model building, I incorporated textual profile information for model training. Namely, I combined all user-entered text — the fields in which users could talk about themselves, their interests, what they are looking for, etc. — and vectorized it into unigrams—more on this as we discuss NLP.

Predicting Orientation

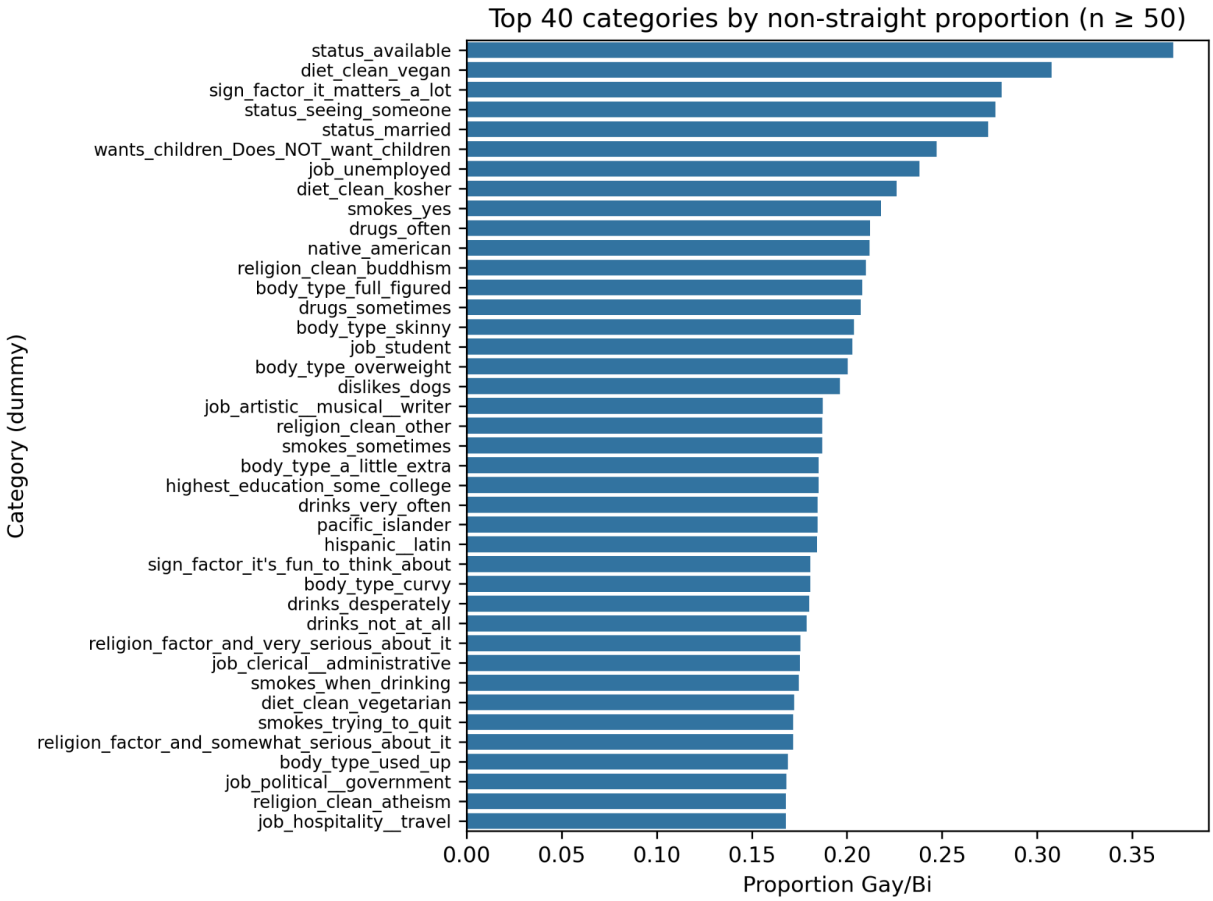
Let's start by exploring the continuous variables. As an a priori hypothesis, it's reasonable to assume that the numeric variables, on their own, would not tell us much beyond the distribution of straight vs. not across the dataset. While potentially useful for prediction, it would be naive to assume that there are any fundamental relationships between orientation and these variables. As it happens, we only have 3 in this data:



I could have spent more time making these charts more straightforward, but why bother? They are clear enough to indicate that there are no immediately noteworthy relationships between height, age, the

number of languages spoken, and orientation. These charts show what we already know: the majority of users are straight.

Slightly more interesting are the categorical variables. With the same caveat as above, we can show the proportion of non-straight users by each dummy:

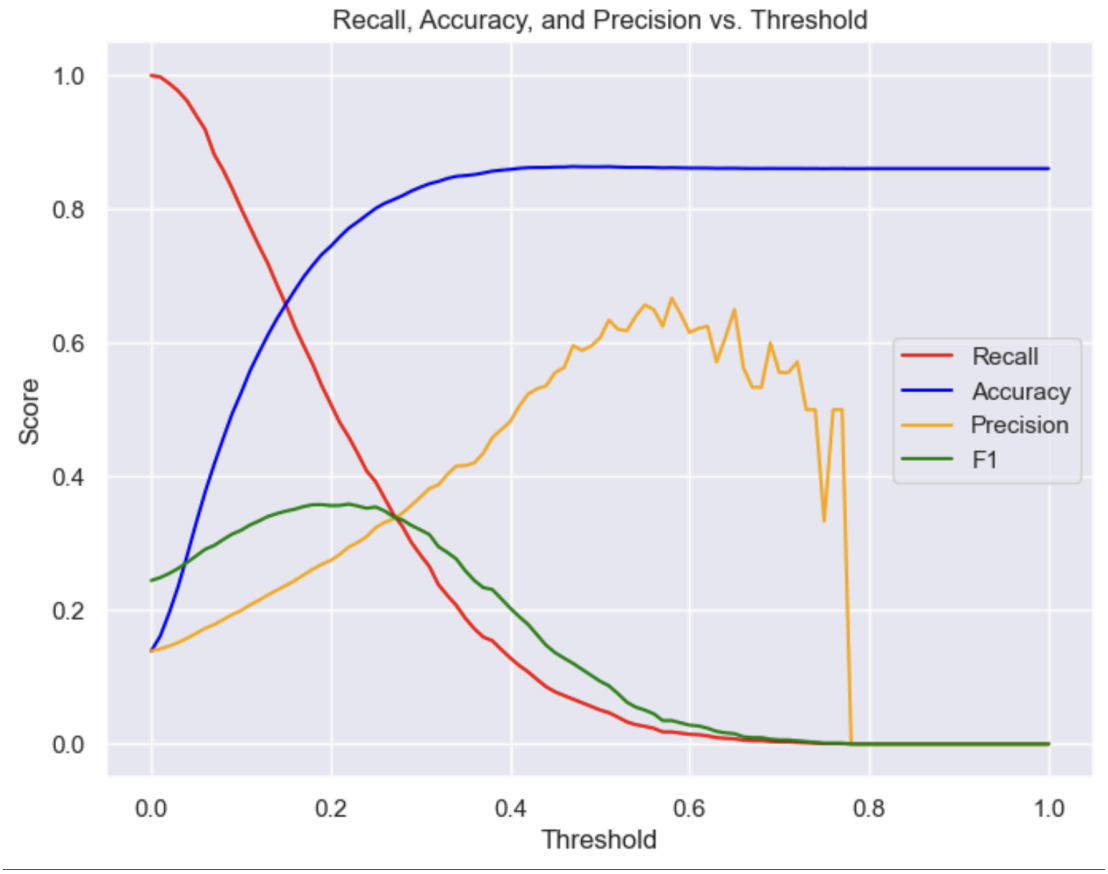


(Note: For reference, ~14% of the entire data set is non-straight class)

As you can see, non-straight users are more likely to be available, married, or partnered, unemployed vegans who are really into star signs. The 'star signs mattering' category, having well above the average proportion of gays, is particularly amusing to see, given the anecdotally observed obsession all my gay friends have with astrology. At the same time, non-straight people never make up the majority in any category, so plenty of straights take star signs seriously, too, I guess.

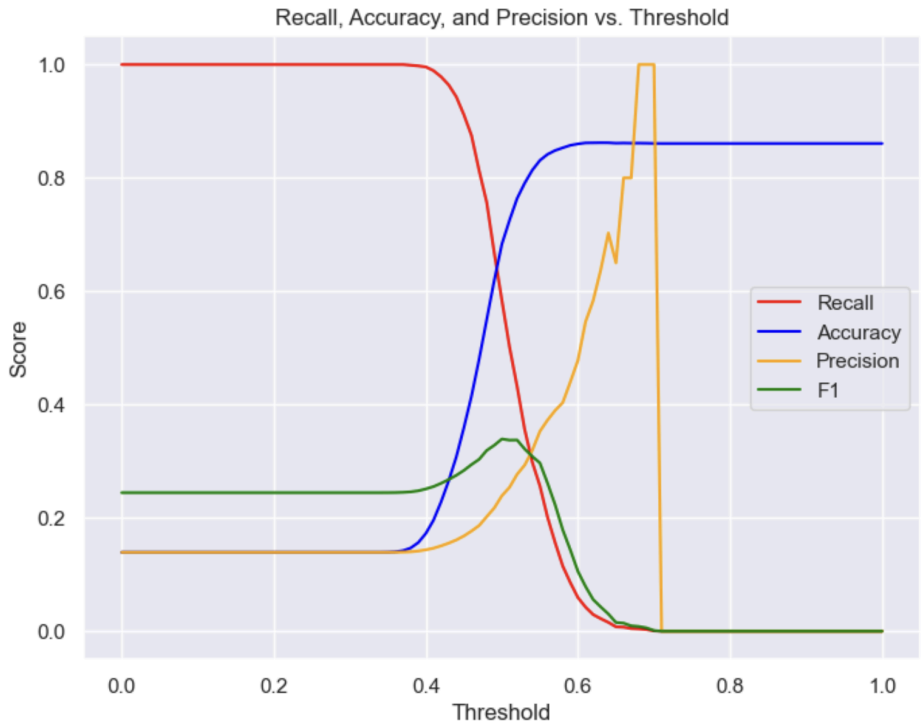
At this point, I was ready to see how much predictive signal these features had before diving into what users wrote. As seen in the Jupyter file, I ran a series of models. To start, I trained models solely on numeric variables such as age and height. Subsequently, I ran some NLP models trained solely on textual profile information, such as user essays, and finally, models trained on both sets of features.

Random Forest 1 Scoring



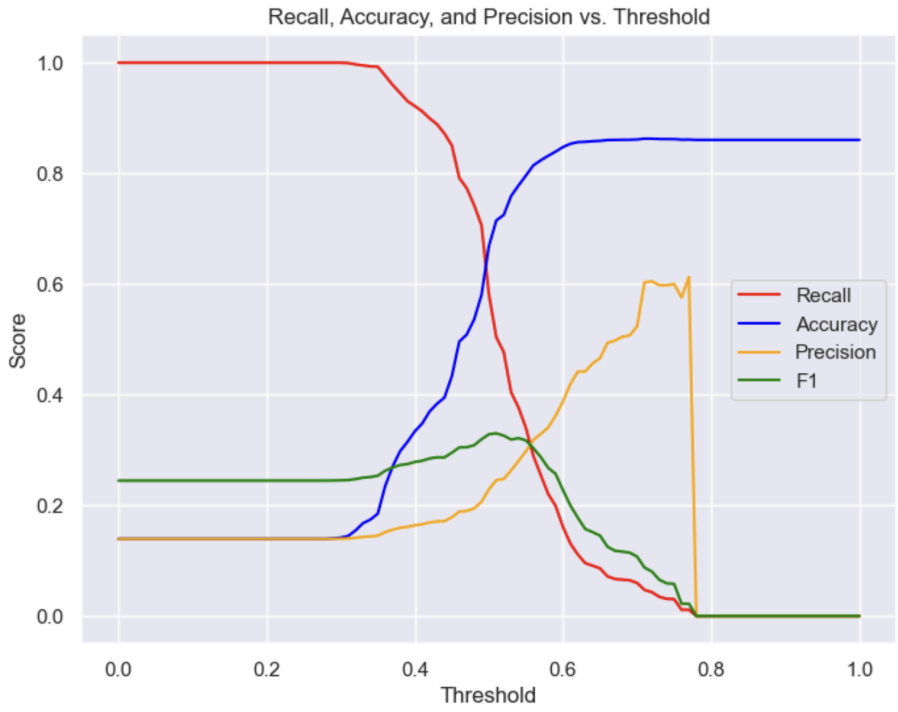
The recall was abysmal, to say the least, and accuracy plateaued at the population proportion of non-straight users. While I did not expect these features to reveal much about a user's orientation, I was shocked by how sharply accuracy and recall diverged across various thresholds. The model essentially needed to drastically over-classify users as gay/bi to classify any of them accurately. While scored to accuracy, these results alone revealed the lack of a relevant signal. Before moving on, I ran another RF search but chose to score on recall. Here is the same chart for that search:

Random Forest 2 Scoring

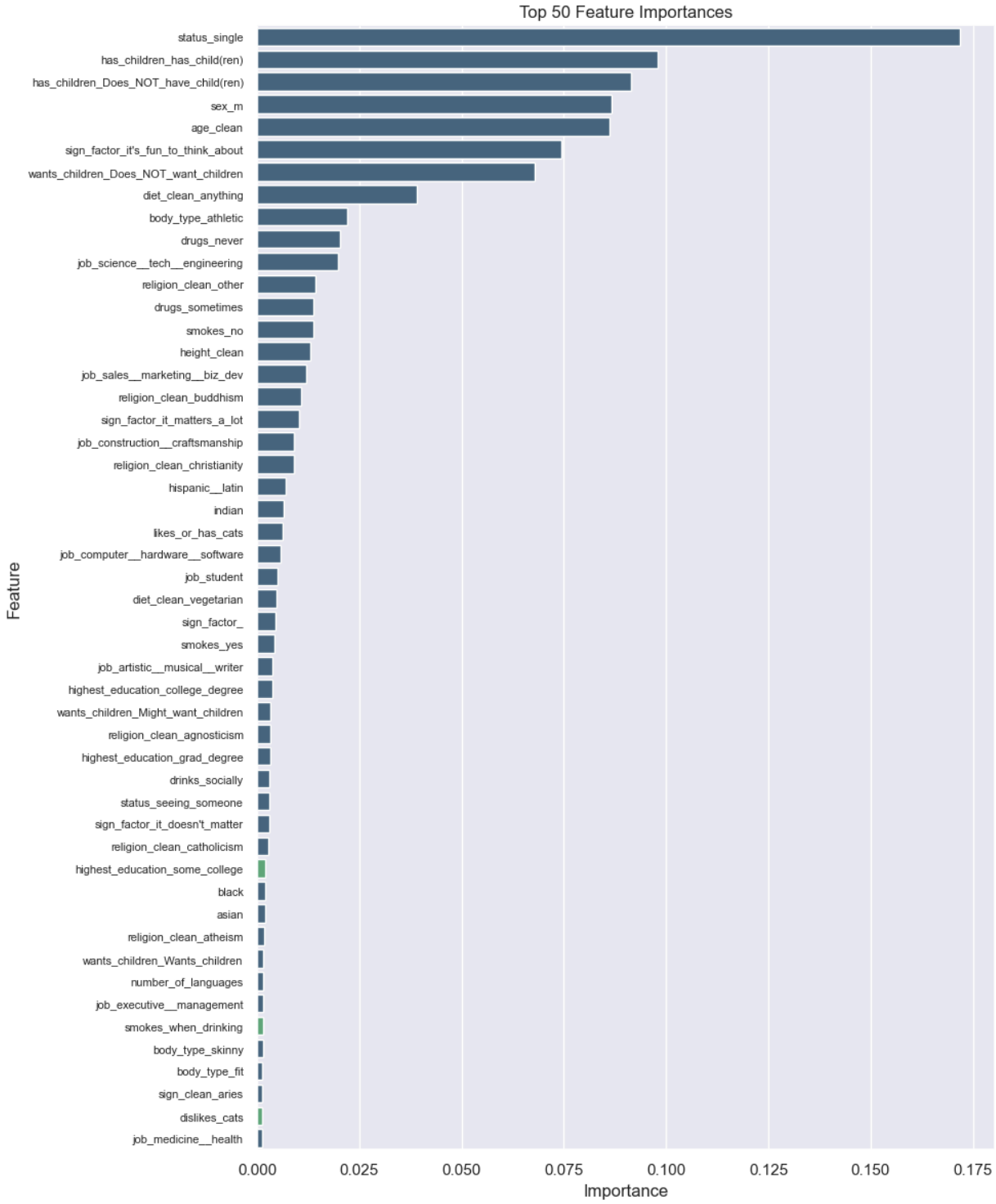


....No change other than the threshold location. For the next try, I balanced the class weights when initializing the random forest, but kept the parameter distribution elements and ranges the same. The results were...

Random Forest 3 Scoring



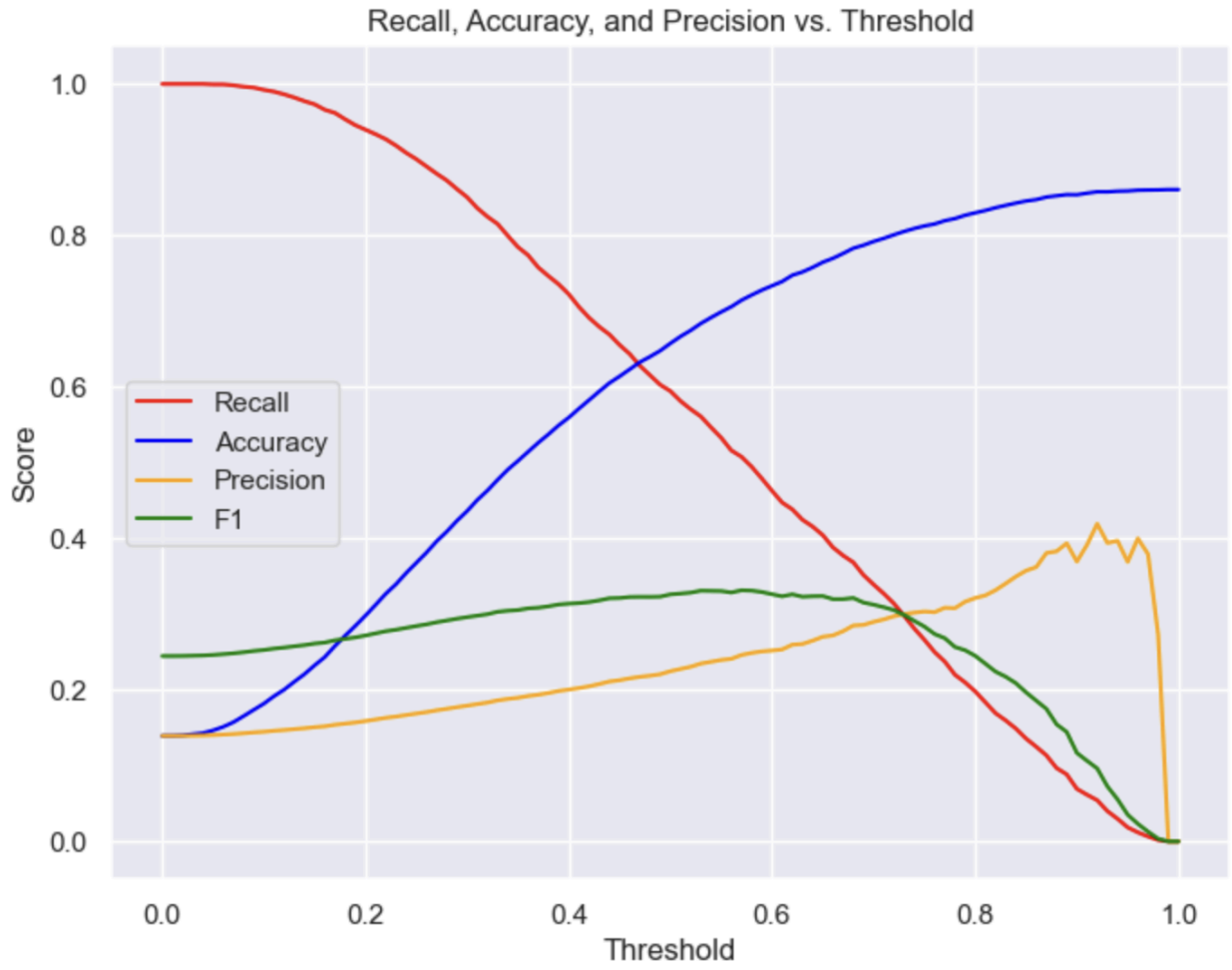
...pretty much the same. Not terribly surprising, and crystal clear that I would need different data for a clearer signal. I did not bother with other estimators for this portion. Before delving into the user essays, I wanted to see which features the forests had discovered as most important. Like the simple proportion summary, relationship status, desire for children, and star signs emerged at the top of the list, along with a handful of others.



Some NPL

Now I wanted to see what would happen if I only used non-numerical data to make predictions. I started with a simple count vectorizer and a Naive Bayes model.

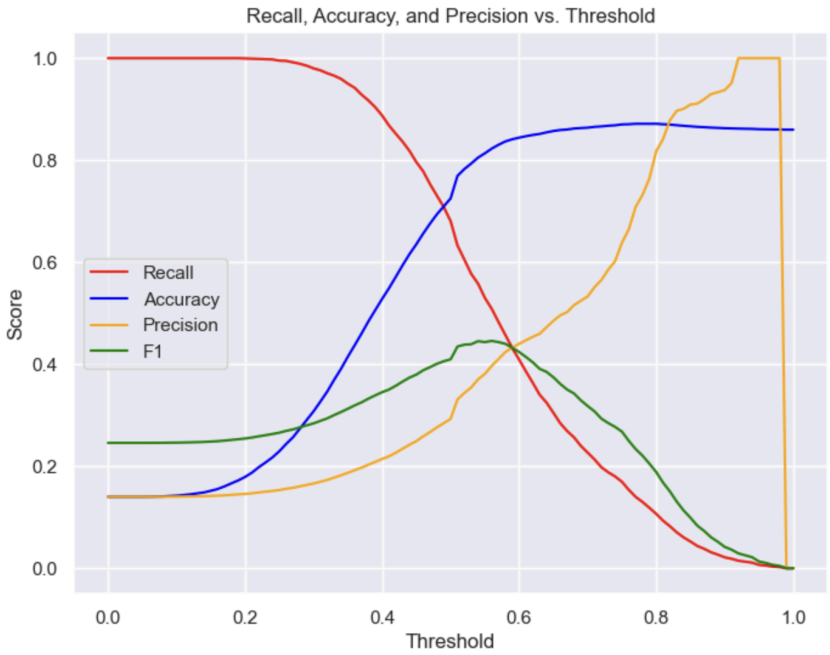
Multinomial Naive Bayes



Not only did I not get any improvement, but the intersection of recall and accuracy was nearly identical. A count vector did not yield any more signal whatsoever. While mildly disappointed, I was not surprised, given how likely it is that users wrote similar things in these prompt portions of their profiles. As mentioned, the users were all from the Bay Area, further increasing the chances of repeated localized references.

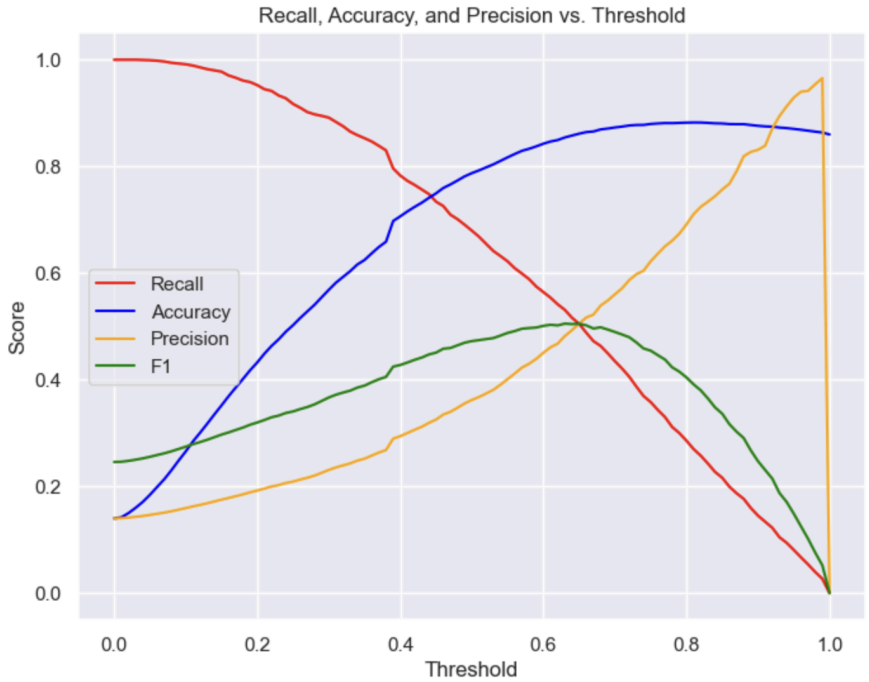
But perhaps focusing on *rarer* words would out the non-straight users to me. After all, how many straight guys would gush about, say, a love of Lady Gaga? (Particularly in 2012). So I repeated the exercise with a TfidfVectorizer to boost the importance of rare words and used a ComplementNB model to reduce bias towards the majority class and to find nuggets of insight to guide the classification.

Complement Naive Bayes



While slight, the intersection of accuracy and recall increased, right around the 50% probability threshold. This improvement gave me some hope that there was indeed previously uncaptured signal and that there could be more of it. Naturally, I now wanted to combine all possible features, as the interaction between the vectorized text and numerical variation could further improve scores. I fit a logistic regression using a randomized search over regularization terms. Here are the scores that the best-performing model produced:

Logistic Regression 1



Again, we see a slight but identifiable improvement. As the code file shows, the best-performing model used a balanced class weight, L2 regularization, and the SAGA solver.

At this point, I could push further by trying different estimators or a more fine-tuned grid search. Still, the pattern was clear: Additional improvements, whether from a different estimator, finer tuning, or more profound exploration of the text, would likely be marginal rather than stepwise.

Some Concluding Thoughts

1. **2012**: A dataset featuring a few tens of thousands of users from over a decade ago, all confined to the same geographical location, naturally limits how well we can predict their orientations. As a queer man myself, I can distinctly remember a more restrictive culture at that time, which understandably could reduce the language variation across straight and non-straight people, particularly in public. Alternatively, it was and remains possible that straight and non-straight people really do use similar language on dating apps and have similar backgrounds.
2. **Type I vs. Type II**: Classifying minority classes can be difficult across a variety of contexts. The nature of incorrect predictions and how we treat them varies by context. Here we're just having some fun. But if we were to use this information to make orientation predictions for, say, product recommendations, one would need to weigh the costs of false-positive vs. false-negative predictions. There are more straight people in the world, so if it's a product recommendation, it's generally better to assume straight and recommend the 'straight' products (e.g., 3, 1-bath products). Suppose this is a public health initiative (Though I sincerely hope no public health institution is predicting sexual orientation from OKCupid profile information) aimed at queer individuals. In that case, false positives may be more tolerable and indeed ideal.

What would I do differently if I re-did this?

1. **Try more estimators**: While I was limited in how far I could push this particular dataset for my classification goals, other estimators might have done marginally better. More importantly, testing more models could have forced me to understand the data more deeply if they had required transformations or more robust performance assessments.
2. **Clean up the scoring**: I realized, as I was writing this, that the best estimators were chosen with the default 50% threshold before I plotted performance across all thresholds. I doubt this made any substantive difference, but it would have been better practice to include a threshold in the parameter distribution rather than as a printout of the best-performing model.
3. **Cleaner Visualizations:** I made a function to plot the scoring across thresholds, but could have added more, such as the title and best estimator params.