NamUs: Unidentified Persons

Using Machine Learning to Predict the 5-Star "Identification Potential"

1. Introduction

Missing and unidentified persons have been referred to as the nation's silent mass disaster. There are many websites - from amateur to state-run - that attempt to facilitate finding and naming these individuals. One such site, funded by the Department of Justice, is the **National Missing and Unidentified Persons System (NamUs)** which does this at the nationwide level. It consists of three databases: Missing Persons, Unidentified Persons, and Unclaimed Persons. Upon entering a new missing persons or unidentified persons case, the information is cross checked with the other respective database to search for matches. This project focusses on data from the Unidentified Persons database.

2. Goal: To Reverse Engineer the Star-Rating

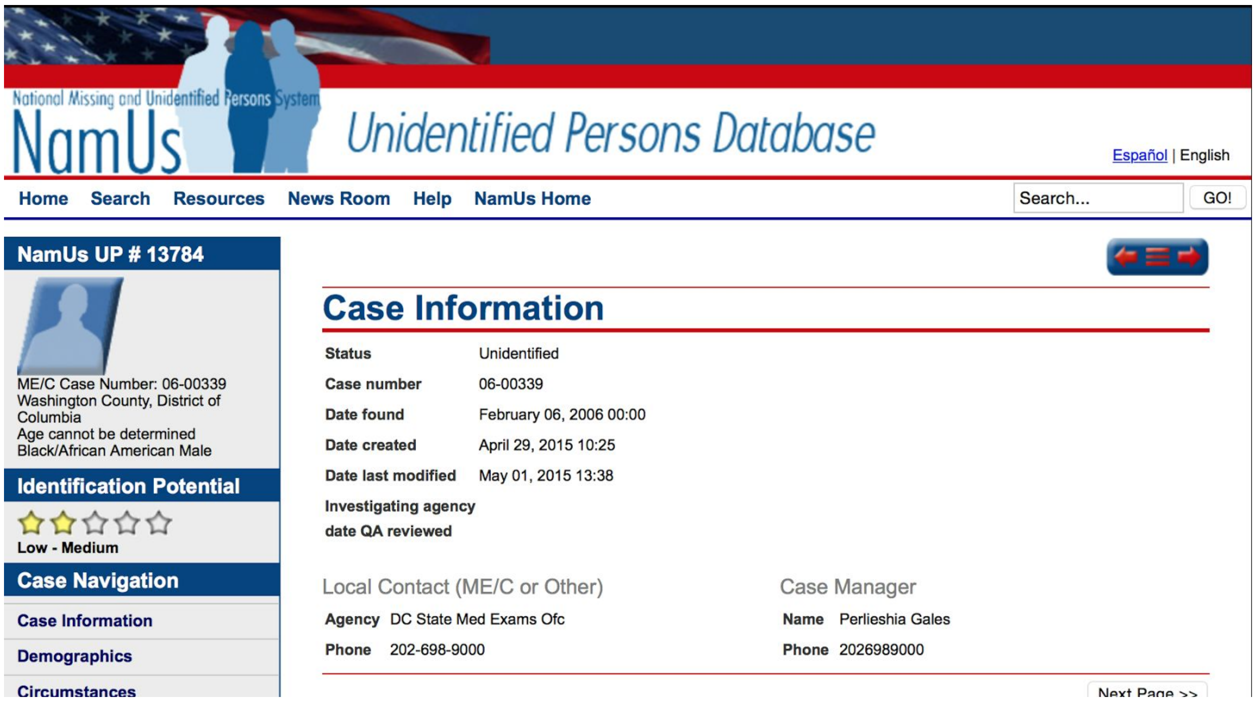
The database rates each case with an Identification Potential - hereon referred to as IP. Although it is a 5-star rating, there are 6 levels as a case can have zero stars. They are as follows: `'Extremely Low'`, `'Low'`, `'Low-Medium'`, `'Medium'`, `'Medium-High'`, and `'High'`). The NamUs website recently updated their Help section, adding a PDF that explains (somewhat vaguely) that the stars are determined automatically upon entering case details. The rating, therefore, can be assumed to be consistent case to case. My goal is to predict this rating based on the data that is freely available to the public within the database.

If I *can* predict the rating:

- Which features predict "Identification Potential"? And how do these compare with that NamUs state are important? While it is outside the scope of this paper, I'd be curious to know if these features are used to search for matches against the Missing Persons Database.

If I can*not* predict the rating:

- This may indicate key information used to calculate the predictions is not available to the public on NamUs;
- Or, this may indicate that there is something inconsistent with the NamUs algorithm. This can only be verified upon a successful FOIA request.


National Missing and Unidentified Persons System
NamUs
Unidentified Persons Database
Español | English
Home
Search
Resources
News Room
Help
NamUs Home
Search...
GO!

NamUs UP # 13784

ME/C Case Number: 06-00339
Washington County, District of Columbia
Age cannot be determined
Black/African American Male

Identification Potential

Low - Medium

Case Navigation

Case Information
Demographics
Circumstances

Case Information

Status	Unidentified
Case number	06-00339
Date found	February 06, 2006 00:00
Date created	April 29, 2015 10:25
Date last modified	May 01, 2015 13:38
Investigating agency	
date QA reviewed	

Local Contact (ME/C or Other)

Agency DC State Med Exams Ofc
Phone 202-698-9000

Case Manager

Name Perlieshia Gales
Phone 2026989000

Next Page >>

Modelling

This project could be considered a regression problem since the IP rating is a linear scale of 0 - 5 stars, or a classification problem due to the fact that it is not possible to be 'between' stars.

3. Data Collection

The data were scraped from the site by looping through case numbers with range(0, 14000) using `Namus_scrape2.py`. This script passed the case number to `get_html.py` which appended the case number to the basic URL (eg. `'https://identifyus.org/en/cases/' + '3191' + '/'`), submit the request, and made it look nice with BeautfulSoup4, and saved it to an html file identifed by case.

Once collected, the HTML files were looped through by case number using `Namus_fromFile.py`, and pertinent data were extracted (script `get_namus_fromFile.py`), and then added to a list, which was then saved as a dataframe and pickled for later. Pickling is a way to save dataframes directly, rather than saving them in a csv file.

Challenge: Initially, the steps were run all together using `Namus_scrape.py`: loop through the case numbers, submit a request to the website, save to an html file, extract the data, append to a list, and create a dataframe. However, I found that the number of case files collected were fewer than the number of cases on the website. Not every number my script looped through had a corresponding case. This makes sense if the case has been solved, as solved cases are unavailable to view online, but I was missing more than just solved cases. This was made clear when I found cases online that I did *not* have on file. It is possible that the missing cases were due to a timeout issue. Whatever the cause, it was fixed when I separated the processes of scraping the HTML and saving it to a file, and extracting data from those saved HTML files.. The case-number loop was rewritten to avoid collecting html files that I already had, and only collected the files it had previously missed. These were stored in a separate directory so I could keep track of files from my original scrape and the modified one.

Once all the html files were collected, and all the data extracted from them, a dataframe consisting of the first "batch" of data was created and pickled. The same was done for the second "missing data" batch. I kept them separate to begin with in case the reason they were missed initially was not due to timeout error but due to something about the data itself. Once unpickled (same process as reading in a csv) the two dataframes were joined together to achieve a complete dataset.

4. Data Extraction, Cleanup, Re-coding, and Feature Engineering

Data Extraction:

Data within the HTML files consisted of:

- Text (e.g. case number, date found, address found);
- Binary (radio buttons/check boxes on the web page), which appear as 0 or 1;
- Long-form text entry (e.g. descriptions);
- Standardized text options (e.g. dropdown selection or a specific list of available options) used to;
- Some combination of standardized text options and text;

- Images.

Tables of original data types:

Text in light grey were excluded as likely having no bearing on IP, or data was limited to very few cases..

Text (standard options or short form)	Binary
RESPONSE: case_rating (recoded to 0-5)	all_parts_recovered
case_status	head_not_recovered
case_number	torso_not_recovered
date_found	n-limbs_not_recovered
est_age	n-hands_not_recovered
min_age	amputations
max_age	deformities
race	scars_and_marks
ethnicity	tattoos
sex	piercings
weight	artificial_parts_aids
height	finger_toe_nails
min_year_of_death	other_distinctive features
postmortem_interval	medical_implants
address_1, address_2, city, state, zip, county,	foreign_objects
fingerprints	skeletal_findings
dna	organ_absent
dental	prior_surgery
hair_color	other_medical_information
left_eye_color, right_eye_color	
recognizable_face	

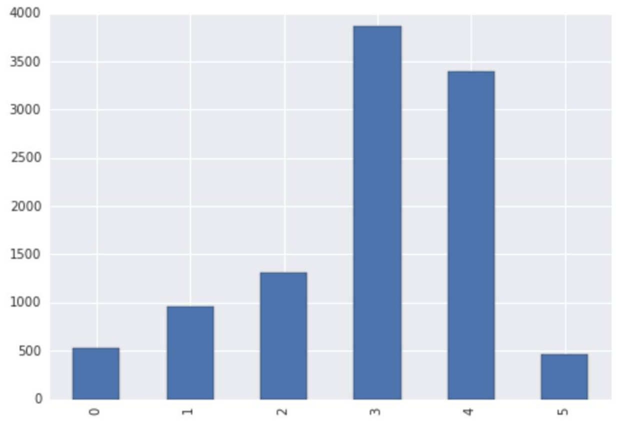
Table describing features that may be used in text analysis:

Long-Form Text	**Long-Form Text**
circumstances	eye_description
head_hair	body_hair
facial_hair	amputations_description
deformities_description	scars_and_marks_description
tattoos_description	piercings_description
finger_toe_nails_description	other_distinctive_features_description
medical_implants_description	foreign_objects_description
skeletal_findings_description	organ_absent_description
prior_surgery_description	other_medical_information_description
clothing_on_body	clothing_with_body
footwear	jewelry
eyewear	other_items_with_body
artificial_parts_aids_description	

Images were converted to a **count**, rather than getting the actual images themselves. The idea being that more pictures might lead to a higher IP. Future work might involve downloading the images and processing them so as to have counts for 'face' vs. 'non-face' pictures.

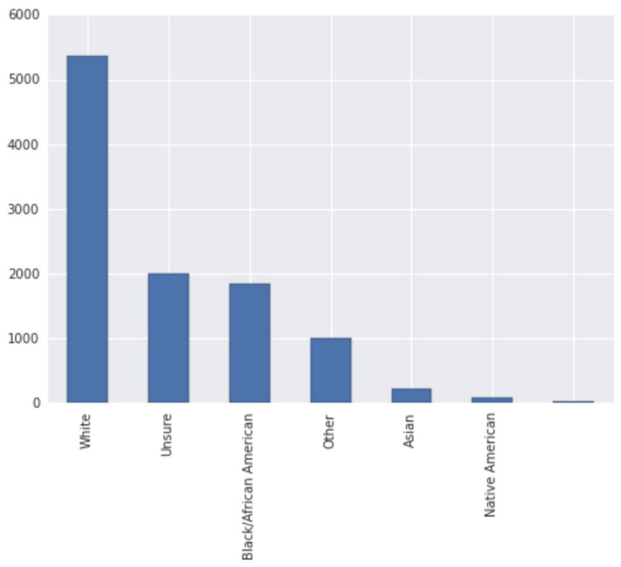
Brief Overview of the Data:

Shape (10529, 72) -> 10529 cases, 72 features



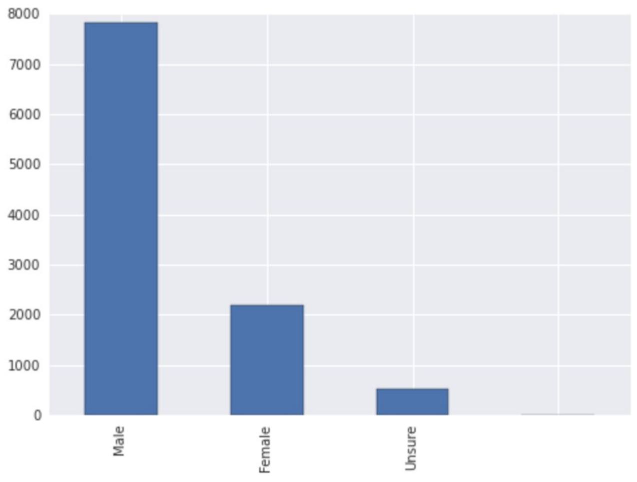
Response: 5-Star Rating

- 0 526,
- 1 962,
- 2 1312,
- 3 3870,
- 4 3396,
- 5 463



Race

● White	5357
● Unsure	2001
● Black/African American	1838
● Other	1006
● Asian	216
● Native American	89
● 22 -> missing	



Sex

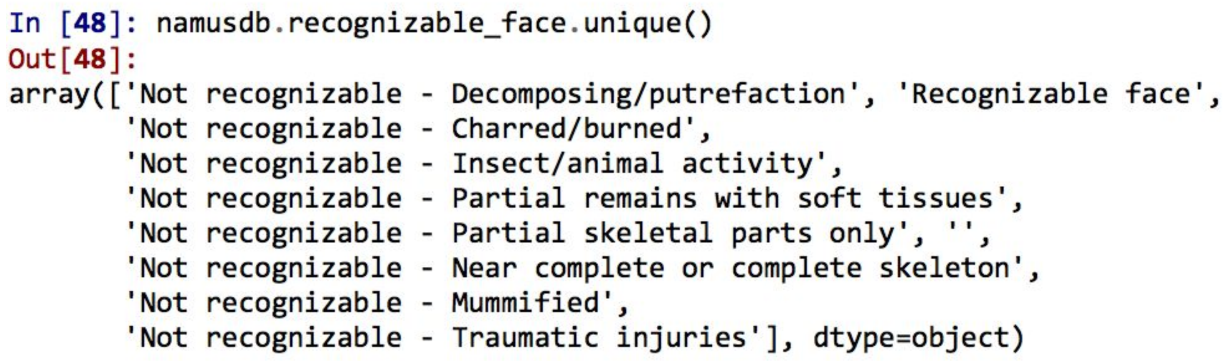
● Male	7815
● Female	2194
● Unsure	518
● 2 -> missing values	

Case Status

● Unidentified	10527
● Unidentified Living	2

Re-coding / Feature Engineering:

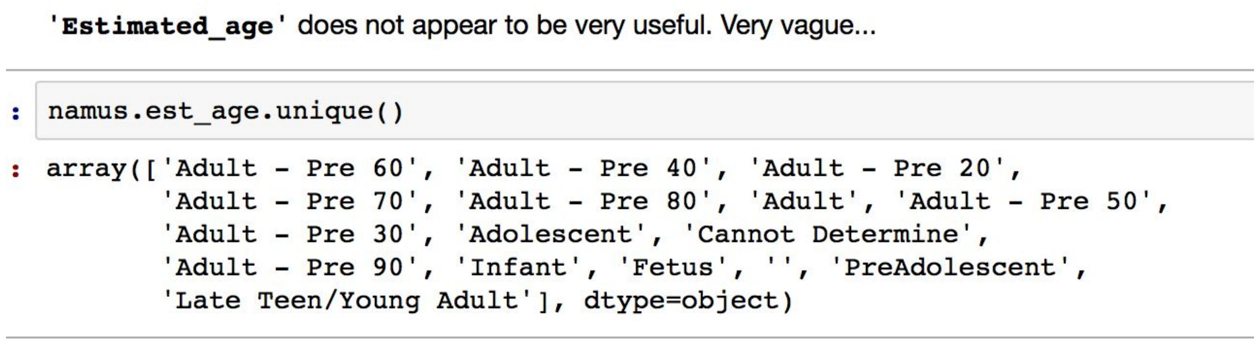
The "standard text" options were recoded into binary to `'1'` for 'yes, data exists for this feature for this case' and `'0'` for 'no, data does not exist for this feature for this case'. For example, recognizable-face provided the options seen in the figure below. Only `'Recognizable face'` was coded as `'1'`, while all other options, plus the empty string `' '` was coded as `'0'`.



```
In [48]: namusdb.recognizable_face.unique()
Out[48]:
array(['Not recognizable - Decomposing/putrefaction', 'Recognizable face',
       'Not recognizable - Charred/burned',
       'Not recognizable - Insect/animal activity',
       'Not recognizable - Partial remains with soft tissues',
       'Not recognizable - Partial skeletal parts only', '',
       'Not recognizable - Near complete or complete skeleton',
       'Not recognizable - Mummified',
       'Not recognizable - Traumatic injuries'], dtype=object)
```

Similarly for `'sex'` ('Male' = 1, 'Female' = 1; 'Unsure' = 0, and ' ' = 0), `'dna'`, `'fingerprints'`, and `'dental'`, `'left_eye'`, and `'right_eye'`. New features names were `'_face'`, `'_sex'`, `'_dna'`, `'_fingerprints'`, `'_dental'`, `'l_eye'`, `'r_eye'`.

Next came ages, weights and heights. Age was described in three features: `'est_age'` for estimated age, and `'min_age'` and `'max_age'` for the range. Because `'est_age'` was rather vague (see below) I excluded it from further analysis.

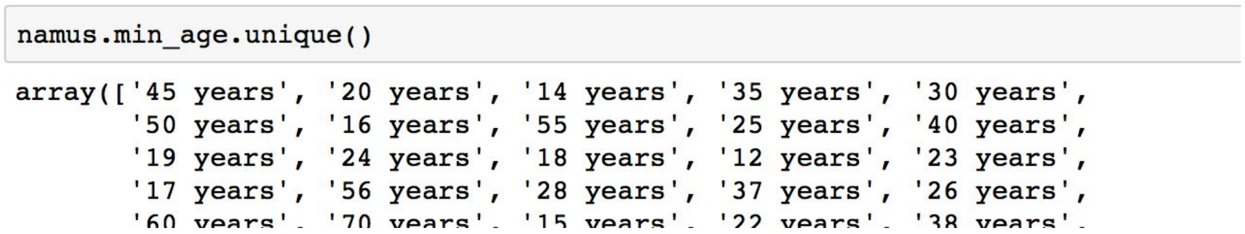


'Estimated_age' does not appear to be very useful. Very vague...

```
: namus.est_age.unique()

: array(['Adult - Pre 60', 'Adult - Pre 40', 'Adult - Pre 20',
         'Adult - Pre 70', 'Adult - Pre 80', 'Adult', 'Adult - Pre 50',
         'Adult - Pre 30', 'Adolescent', 'Cannot Determine',
         'Adult - Pre 90', 'Infant', 'Fetus', '', 'PreAdolescent',
         'Late Teen/Young Adult'], dtype=object)
```

The data for `'min_age'` and `'max_age'` were entered as an integer and text.

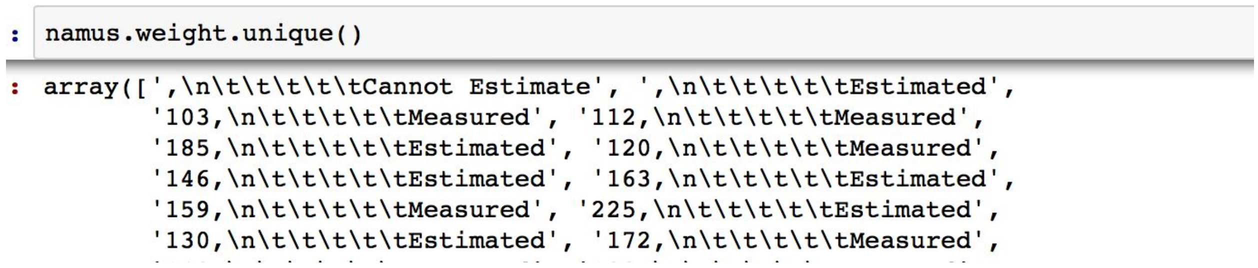


```
namus.min_age.unique()
```

```
array(['45 years', '20 years', '14 years', '35 years', '30 years',
       '50 years', '16 years', '55 years', '25 years', '40 years',
       '19 years', '24 years', '18 years', '12 years', '23 years',
       '17 years', '56 years', '28 years', '37 years', '26 years',
       '60 years', '70 years', '15 years', '22 years', '38 years',
```

Finding that all text was `'years'` (no `'months'` for example) I decided to extract the integer from the minimum and maximum, and then subtract the two values to get `'age_range'`. The idea being that more confidence in the age (a smaller range) might indicate a higher IP, while less confidence in the estimation of age (a larger range) might lead to a lower IP. Negative values were made positive using Numpy's `'abs'` (absolute) function with the assumption that the values were switched unintentionally during data entry, resulting in a negative value when subtracted. Empty strings were initially converted to `'nan'`. Later, because regression models dislike nan values, and there were too many to drop, nan values were replaced with `'99'`. I chose this number because if I believe the range indicates confidence in estimated age, a missing input would indicate an inability to estimate, and therefore the range might as well be 99 years. This value also already exists in the data, suggesting that it is a valid entry in some cases.

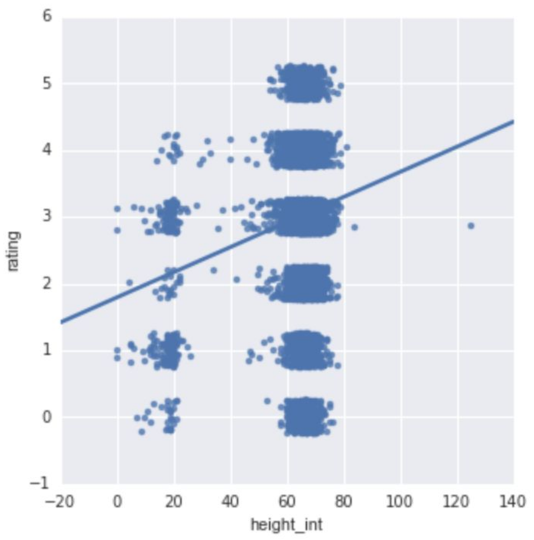
Height and weight were more tricky. They comprised text fields containing either empty strings, text (`'Estimated'`, `'Measured'`, `'Cannot Estimate'`) with no value for height or weight, strings of values without the text, or the entry would be complete with values and text.



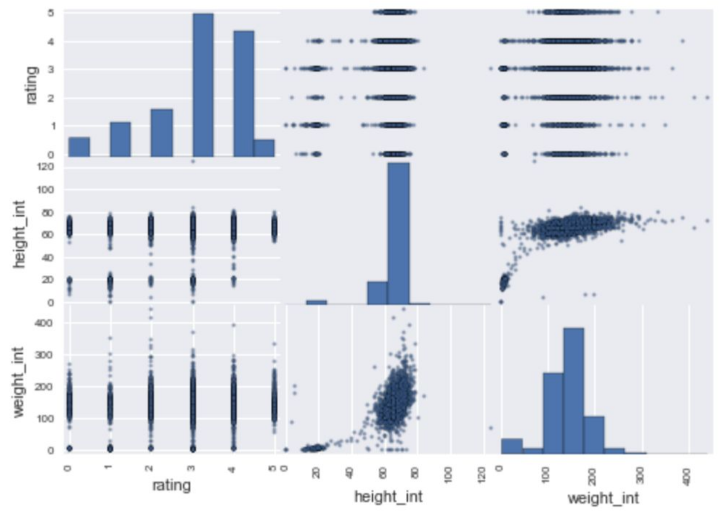
```
: namus.weight.unique()
```

```
: array([',\n\t\t\t\t\tCannot Estimate', ',\n\t\t\t\t\tEstimated',
        '103,\n\t\t\t\t\tMeasured', '112,\n\t\t\t\t\tMeasured',
        '185,\n\t\t\t\t\tEstimated', '120,\n\t\t\t\t\tMeasured',
        '146,\n\t\t\t\t\tEstimated', '163,\n\t\t\t\t\tEstimated',
        '159,\n\t\t\t\t\tMeasured', '225,\n\t\t\t\t\tEstimated',
        '130,\n\t\t\t\t\tEstimated', '172,\n\t\t\t\t\tMeasured',
```

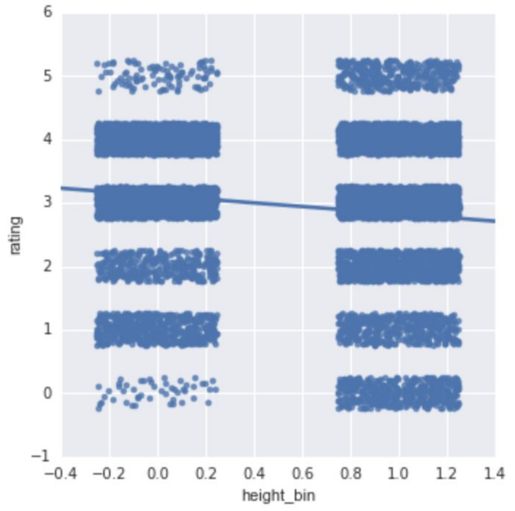
Options included making height and weight features consisting of just the values (integer feature) or based on the text (string feature). Height and weight integers were put into columns named 'height_int' and 'weight_int'.



The (super jittered) scatterplot to the left shows that there is a strong relationship between `'height_int'` and `'rating'`. This pattern is very similar for `'weight_int'` vs `'rating'`. However, I believe this is misleading, and that it is more reflective of the unbalanced data (number of observations per star), than a predictive relationship between height/weight and rating. The height/weight distributions appear very similar across IPs 0-4. An IP of 5-stars is exclusively taller people, however. Therefore it is not clear how best to use these features in a model. The scatter matrix plot (left) displays this same data a little differently.



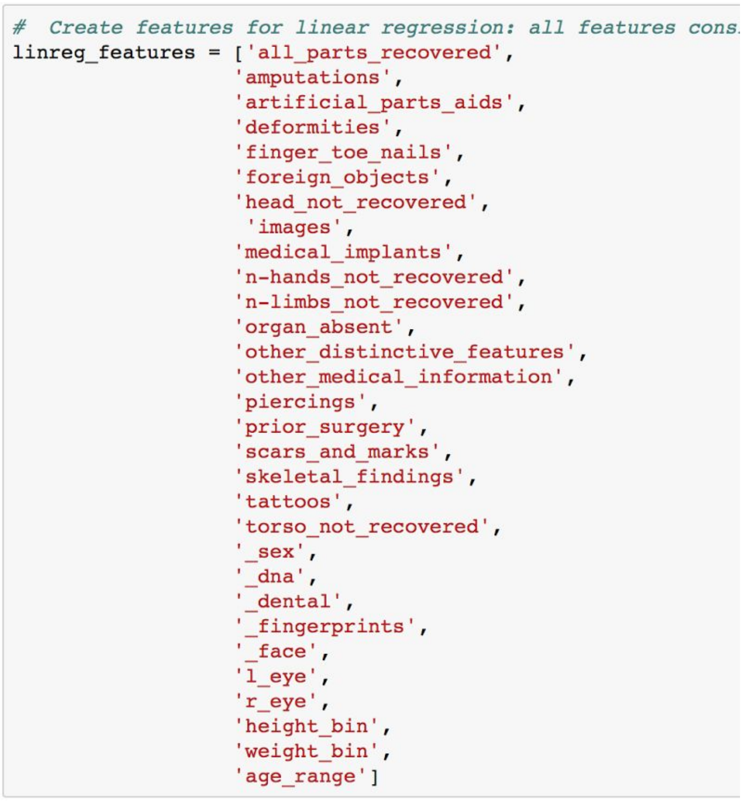
Unlike `'age_range'`, it was unclear to me how to deal with missing values for heights and weights.. For `'age_range'` where the size of the number indicates the accuracy of the estimation (or the confidence in the accuracy), it is a single value for height and weight, and the confidence is in the text associated with that value (`'Cannot Estimate'`, `'Estimated'`, `'Measured'`). I decided, therefore, that the text might be more relevant than the numeric value, as a proxy for 'available or known' data, vs 'unknown, unavailable' data. To achieve that, text were recoded as `'Measured'` = 1, `'Estimated'` = 1, `'Cannot Estimate'` = 0, and `' '` = 0 for both height and weight, and the new features were named `'weight_bin'` and `'height_bin'` for 'binarized'. Surprisingly, when binarized the relationship between 'known data' and 'unknown data' versus rating is now negative.



5. Feature Selection and Modelling

Linear Regression:

Ignoring all text features, I started with the following 30 binary/numeric features:



```
#  Create features for linear regression: all features cons
linreg_features = ['all_parts_recovered',
                   'amputations',
                   'artificial_parts_aids',
                   'deformities',
                   'finger_toe_nails',
                   'foreign_objects',
                   'head_not_recovered',
                    'images',
                   'medical_implants',
                   'n-hands_not_recovered',
                   'n-limbs_not_recovered',
                   'organ_absent',
                   'other_distinctive_features',
                   'other_medical_information',
                   'piercings',
                   'prior_surgery',
                   'scars_and_marks',
                   'skeletal_findings',
                   'tattoos',
                   'torso_not_recovered',
                   '_sex',
                   '_dna',
                   '_dental',
                   '_fingerprints',
                   '_face',
                   'l_eye',
                   'r_eye',
                   'height_bin',
                   'weight_bin',
                   'age_range']
```

A linear regression model was instantiated with 'normalize=True' to account for the different scales between the binary features and the numeric ones (images being a count, and age_rage). Using train-test-split, an RMSE of 0.8159 was achieved. For a first attempt, this is not *too* bad, and means that a prediction of IP from 0-5 can be off by approximately 0.82 stars.

Feature Selection/Regularization:
Using 'RidgeCV' regularization testing a large alpha range, the RMSE reduced to 0.8158 (alpha = 0.01). Using 'LassoCV' regularization also only improved the RMSE minimally to 0.8154 with a tiny alpha of 5.29e-05. The advantage of Lasso is that it reduces the coefficients of features that are unimportant to zero: essentially removing them from the model. In this case, Lasso removed `'artificial_parts_aids'`, `'deformities'`, `'foreign_objects'`, `'r_eye'`, `'medical_implants'`, and `'torso_not_recovered'`. Encouragingly, based on these same coefficients, LassoCV ranked the most *important* features as, from most to least, `'_dna'`, `'_dental'`, `'_fingerprints'`, `'_sex'`, and `'_face'`. NamUs explain that a 5-star IP requires these same features with with exception of `'_sex'` and the addition of *facial* images. I stressed facial, there, because I have made not processed the images to separate images of faces and images of other objects.

Null hypothesis testing:

For perspective, I tested the Null for this linear regression model which returns an RMSE of 1.1927, suggesting my model is a little better than chance.

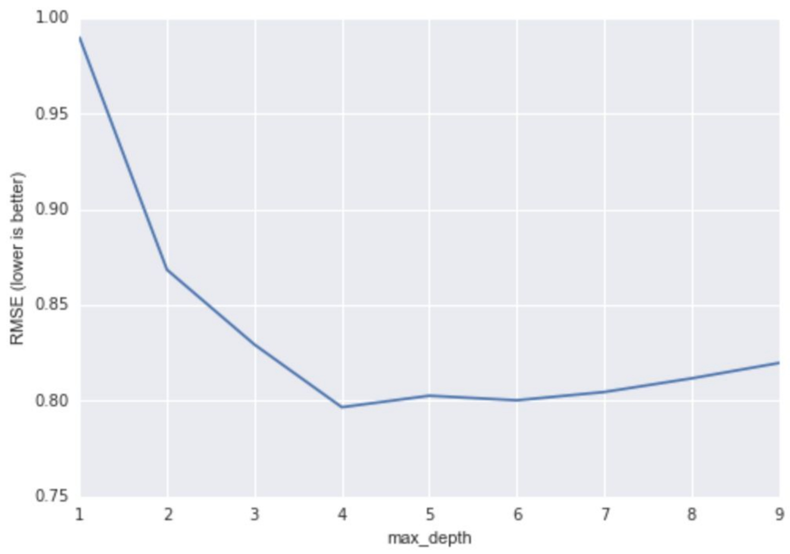
Decision Tree: Regression

Namus' IP algorithm was likely constructed by humans based on professional experience (e.g. coroner, police), and the general description of how it works suggests a hierarchy of important features. Therefore, decision trees, which use information hierarchy and rules, should perform well in predicting IP. Other advantages are that trees are insensitive to features that have little to no predictive value, and they are nonparametric, and so should still perform well with this data which has a leftward skew.

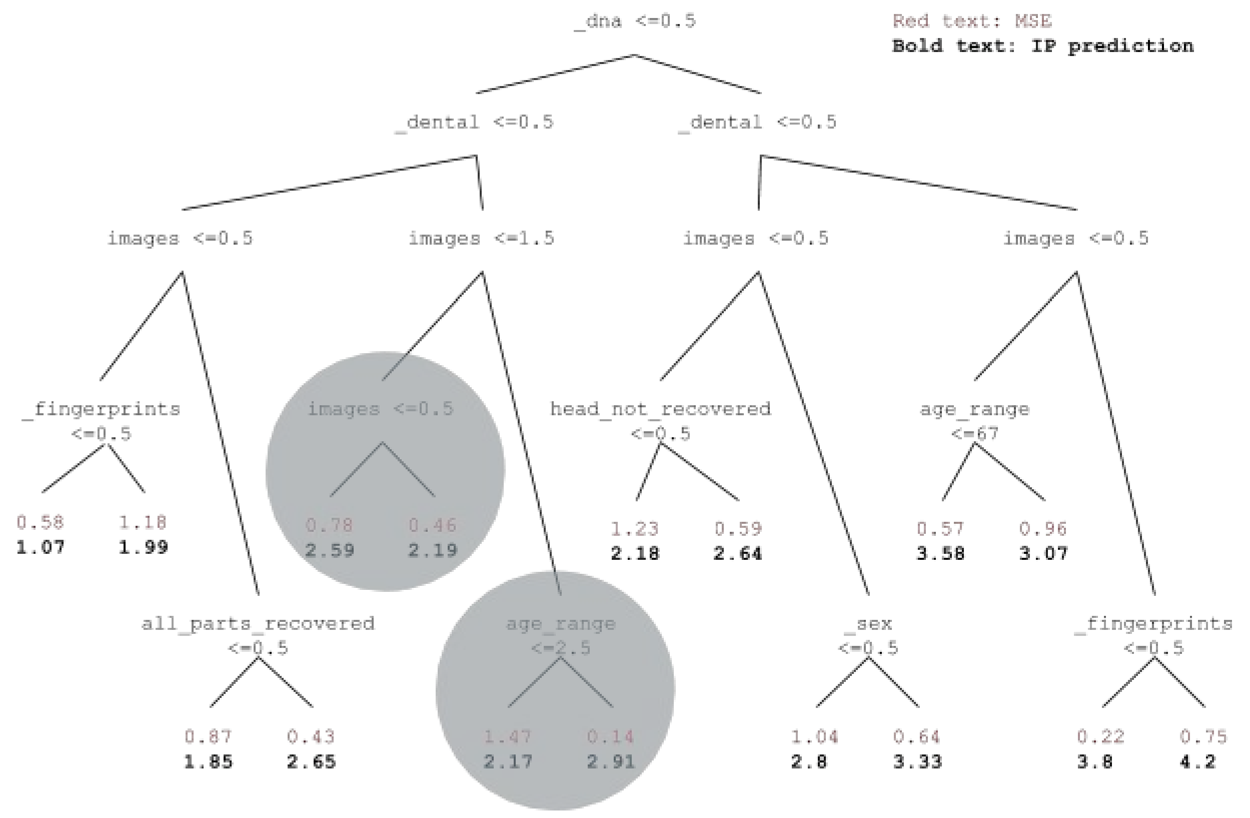
I started with a decision tree *regression* model. Despite the fact it is known to be insensitive to irrelevant features, removing the features suggested by LassoCV reduced the RMSE by ~0.01 from 0.9782 to 0.9665. This is a tiny improvement, however.

Tuning:

Looping through maximum depths 1-10 (random_state = 1, number of folds [cv] = 10), a depth of 4 splits provided an RMSE of 0.7966. Using cv=1000, resulting in folds containing ~10 observations, the RMSE reduced further to 0.7316. This large improvement with more folds might be due to the non-parametric distribution of the data: there are far fewer 0-star, 1-star and 5-star cases, but they might be given more influence when the fold number is so large that the number of observations include ~10 cases. Trees are known to be high variance, so this difference with the higher number of folds reflects this, I think.



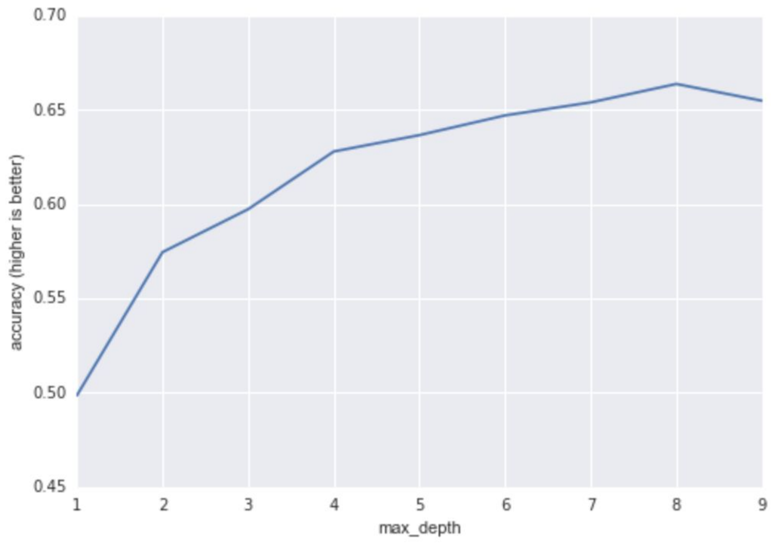
Regression Tree-Splits:



With a depth of 4 splits, the tree fails to predict any 0-Star or 5-Star cases (predicted ratings in bold text; MSE in red text). This may be due to the limited number of splits, which limits the complexity of the model. There are also some unexpected decisions (blue circles) where fewer images or a larger age range leads to a higher star prediction. Using a depth of 7 splits *did* predict 5-star cases, but there remained decisions that did not make sense. For example, in treereg_depth7.png, following the chart along right-decisions we come to `'images'<=14.5`. If it is true, we go left where the star prediction is 4.6 (round up to 5 stars), and if the images are more than 14.5, we go right and see a star prediction of 3.3 (round down to 3 stars). My original assumption was that more images leads to higher IP. Some branches support, while others go against this assumption. Continuing in treereg_depth7.png in the /images directory, the last decision on the right is `'height_bin'<0.5`: height_bin <0.5 must be 0 because it is binary. If true, and there *is no* height information, we go left to a prediction of 4.7 (or 5-stars) and if false (so there *is* height information) we go right for a prediction of 4-stars. Clearly this does not make sense.

Minimum leaf size was explored momentarily, but when considering there are several different routes one might take through the features to arrive at a 3-star rating, for example (i.e. DNA and/or dental and/or images and/or fingerprints etc, according to NamUS), a minimum leaf size no longer made sense, so this tuning parameter was abandoned.

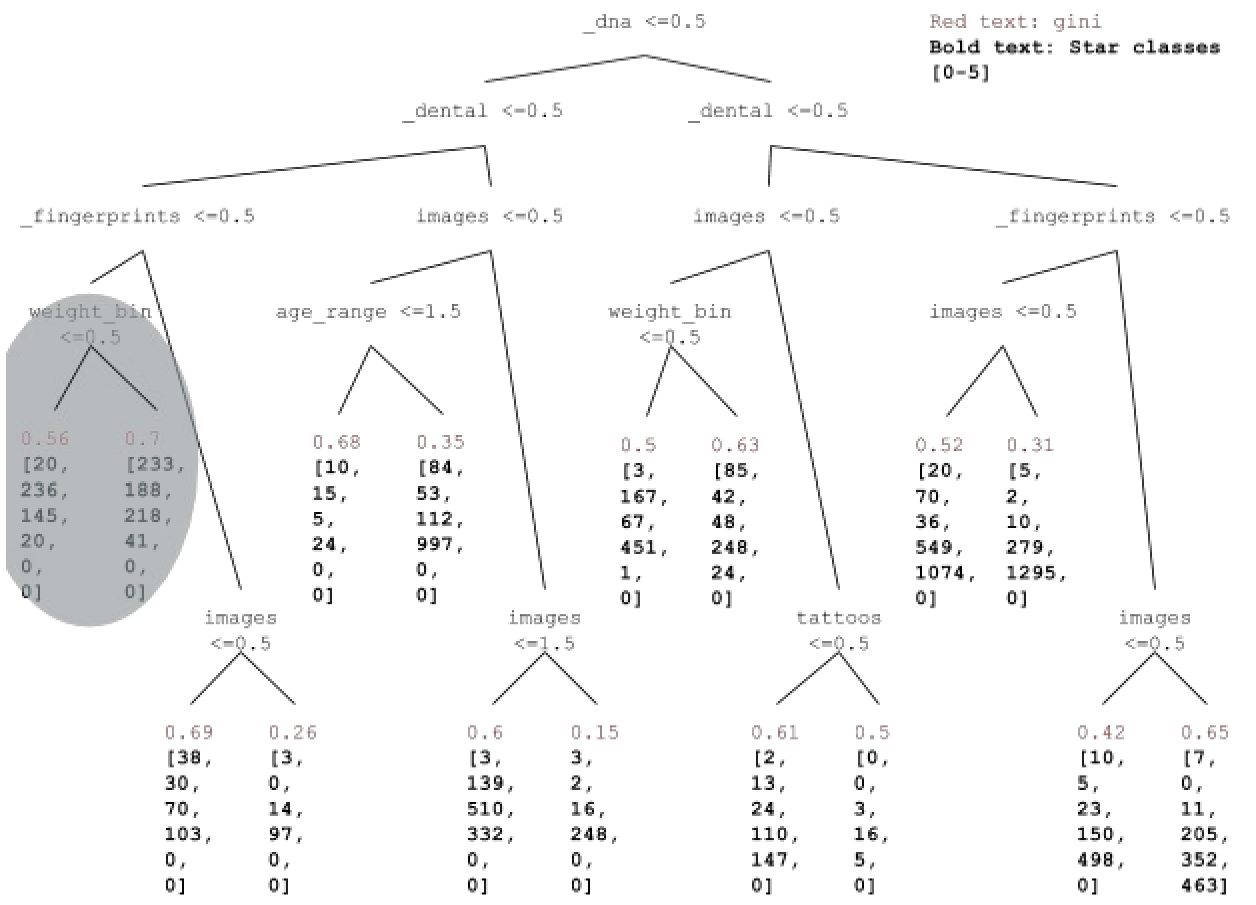
Decision Tree: Classification



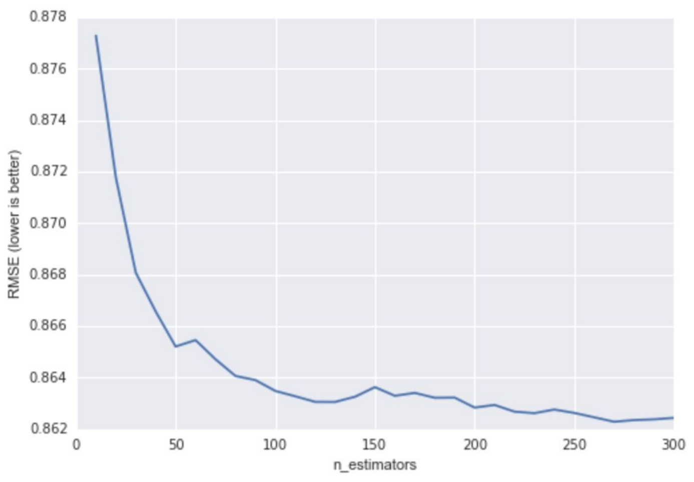
Because I am trying to predict discrete categories (though they are also ordinal) I explored a classification tree. Looping through maximum depth values, a depth of 8 leads to the best accuracy score of 66.37%, which is rather low. In addition, a depth of 8 splits is very complex and therefore is unlikely to generalize well. A depth of 4 splits, however, produces a tree with an accuracy of 62.79% and places all 5-Star cases in one leaf with a gini purity of 0.65 (a score of 0 is the best). Here, the five most important features are, in descending order: `'_dna'`, `'_dental'`, `'images'`, `'_fingerprints'`, and `'weight_bin'`. Surprisingly, `'_face'`, which is essential to getting a 5-Star IP along with DNA, dental, fingerprints, and facial images, is 20th. Although this may be considered an improvement on the Regression Tree, some decisions by the tree are unexpected. For example, highlighted in blue is `'weight_bin'` which reached by having no DNA, no dental records, and no fingerprints. At this point we would expect a low star rating for `'weight_bin'` <=0 being true (take the left branch), and a higher rating for it being false (right branch; meaning there *is* a weight measured or estimated). However, we actually get a majority 1-Star rating if there is no weight information, and majority 0-Star if there *is* weight information. It is reasonable to assume that there are features being accounted for in the NamUs algorithm that are not being considered in this model: one such feature being `'_face'`.

A figure of the tree can be seen on the next page.

Classification Tree-Splits:



Random Forest: Regression

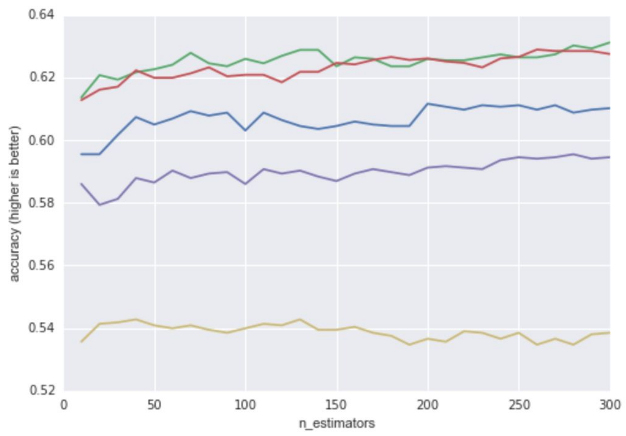


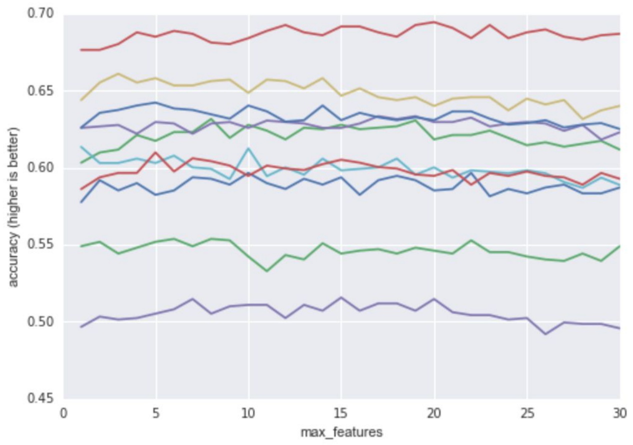
Disadvantages to single decision trees include sensitivity to imbalanced classes such as mine, and high variability. Running multiple trees as in Random Forests, and growing them deep and then combining them together removes these disadvantages, as well as removes bias towards the most influential feature so they also provide a better estimate of feature importance. So far, Linear Regression identified all features essential to a 5-Star rating except `'images'`, while both decision trees got all except `'_face'`. For the Random Forest, I used the full feature list.

An 'out-of-bag' score of 0.56 R^2 (the best score would be 1) was achieved with 270 estimators, 5 features considered at each split, and 5 cross-validation folds.

The top 5 most important features in descending order were `'_dna'`, `'_dental'`, `'age_range'`, `'images'`, `'_fingerprints'`, and `'_face'` as the 6th most important.

Random Forest: Classification





A single classification tree performed differently (and perhaps better) than the single regression tree, so I also wanted to explore a classification random forest. Looping through several different estimator sizes did not reveal an optimal number (see graph on the left-top: each line represents one of the 5 folds). Therefore, I chose the 'peak' in green which looks to be about 65. Assessing the number of features to consider per split also failed to reveal an optimum number of features (see left-bottom: each line represents one of the ten folds). Picking a feature number of 10 and 65 estimators to fit a new model provided an accuracy score of just 61% and an out-of-bag score of 0.65 R^2, which is significantly better than the regression forest. Using just 5 features per split (optimum number identified for the regression forest) achieved an accuracy of 60%. With just 5 features per split, the out-of-bag error is 0.64 R^2. Important features in descending order are: `'age_range'`, `'_dna'`, `'images'`, `'_dental'`, `'_fingerprints'`, and `'_face'` as 6th most important. These are the same regardless of 5 or 10 features per split is selected. Still, I feel that these scores can be improved.

6. Findings, Conclusions, and Future Work

I tested a variety of regression and classification models to predict "Identification Potential" of unidentified persons listed in the National Missing and Unidentified Persons System (NamUs) online database. I found that classification may tend to perform better than regression models, but it is extremely difficult to compare between them with their different performance metrics (RMSE vs. accuracy versus R^2).

Going forward, there are several approaches and modifications I would like to try. Most pressing is to address the difficulty in predicting the 5-star cases. The NamUs website updated their Help section about halfway through my project. Having the new Help file on Star Rating informed me that a case requires all five of `'_dna'`, `'_fingerprints'`, `'_dental'`, `'_face'` and facial images. I think a significant issue with my data is that my images are a count. Knowing now that a 5-Star rating has to have face images (either photographs or artist rendering), I plan to scrape all the images and process them (will be challenging) to identify which images contain faces and which do not. I'll then make a new column for "has facial image" = 1, and "no facial image" = 0; and another column for a count of non-facial images. I believe this alone will greatly improve modelling success! On a less challenging note, I unintentionally neglected to binarize `'hair_color'` into "hair color is known" = 1, and "hair color is unknown" = 0. Further, although the requirements for a , 0-Star, 1-Star or a 5-Star are very strict, and based specifically on what data (or lack thereof in terms of 0-Star) each case has, unrelated to amount of data overall. Between those stars, it seems to be more about which set of specific feature a case has, whereas for stars 2-4 it seems that how many of all the features a case has, which may be successfully modeled in a linear regression. It might be helpful, therefore, to either merge some of these features together, or use an ensembling approach to model the 0-Star, 1-Star and 5-Star cases using a classification model, for example, added to a model that learns stars 2-4 perhaps using a regression model.

Other models that might be worth exploring is One Versus Rest, which might be able to better learn the 0-star and 5-star cases. Ordinal Logistic Regression is good for ordered, multi-class responses, like a 5-Star rating, for example. There is also the text data that I have not addressed in this paper. I began looking at the descriptions of certain features, but with a standard Naive Bayes model, I could only achieve an accuracy of less than 50%.

Using different model performance assessment metrics would be helpful, such as AUC which is useful for unbalanced classes such as mine.

Submitting a FOIA request for data on solved cases would be informative to understand what features actually contributed to the identification of an individual. I am curious to know whether they update their algorithm based on evidence from solved cases.

Finally, I would like to create a map - possibly interactive where one could explore more about the case, for the find-locations for these unidentified individuals.