Predicting Beer Ratings

Emily Miller

April 28, 2017

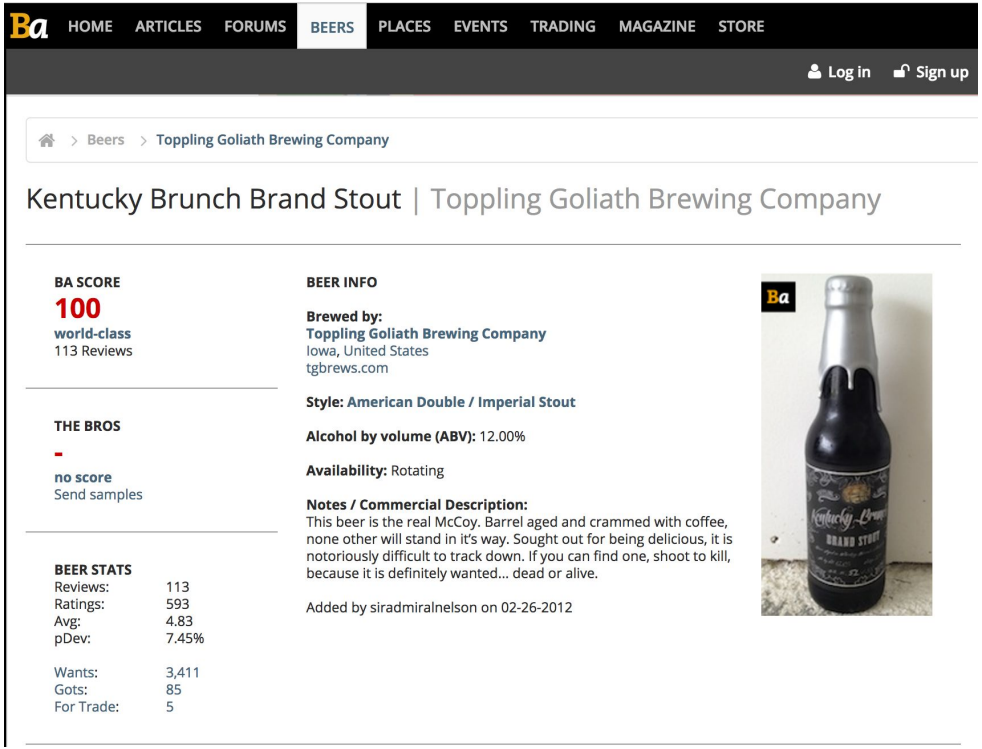
Beer Advocate

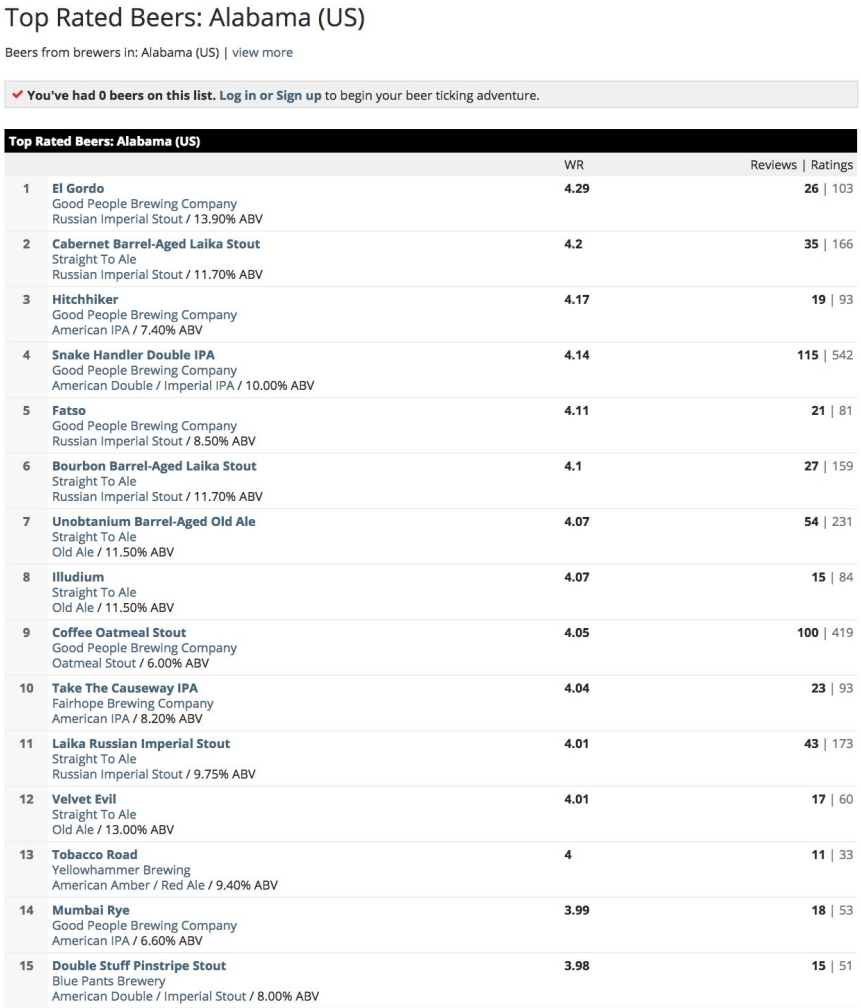
Beer attributes

- Name
- Style
- Alcohol by volume (ABV)
- Brewery
- State
- Year released

Target data:

- Avg user rating





Top Rated Beers: Alabama (US)

Beers from brewers in: Alabama (US) | view more

✔ **You've had 0 beers on this list.** Log in or Sign up to begin your beer ticking adventure.

Top Rated Beers: Alabama (US)

		WR	Reviews | Ratings
1	**El Gordo** Good People Brewing Company Russian Imperial Stout / 13.90% ABV	**4.29**	**26** | 103
2	**Cabernet Barrel-Aged Laika Stout** Straight To Ale Russian Imperial Stout / 11.70% ABV	**4.2**	**35** | 166
3	**Hitchhiker** Good People Brewing Company American IPA / 7.40% ABV	**4.17**	**19** | 93
4	**Snake Handler Double IPA** Good People Brewing Company American Double / Imperial IPA / 10.00% ABV	**4.14**	**115** | 542
5	**Fatso** Good People Brewing Company Russian Imperial Stout / 8.50% ABV	**4.11**	**21** | 81
6	**Bourbon Barrel-Aged Laika Stout** Straight To Ale Russian Imperial Stout / 11.70% ABV	**4.1**	**27** | 159
7	**Unobtanium Barrel-Aged Old Ale** Straight To Ale Old Ale / 11.50% ABV	**4.07**	**54** | 231
8	**Illudium** Straight To Ale Old Ale / 11.50% ABV	**4.07**	**15** | 84
9	**Coffee Oatmeal Stout** Good People Brewing Company Oatmeal Stout / 6.00% ABV	**4.05**	**100** | 419
10	**Take The Causeway IPA** Fairhope Brewing Company American IPA / 8.20% ABV	**4.04**	**23** | 93
11	**Laika Russian Imperial Stout** Straight To Ale Russian Imperial Stout / 9.75% ABV	**4.01**	**43** | 173
12	**Velvet Evil** Straight To Ale Old Ale / 13.00% ABV	**4.01**	**17** | 60
13	**Tobacco Road** Yellowhammer Brewing American Amber / Red Ale / 9.40% ABV	**4**	**11** | 33
14	**Mumbai Rye** Good People Brewing Company American IPA / 6.60% ABV	**3.99**	**18** | 53
15	**Double Stuff Pinstripe Stout** Blue Pants Brewery American Double / Imperial Stout / 8.00% ABV	**3.98**	**15** | 51

Scraped top beers for all 50 states plus D.C.

Total of 3,824 beers

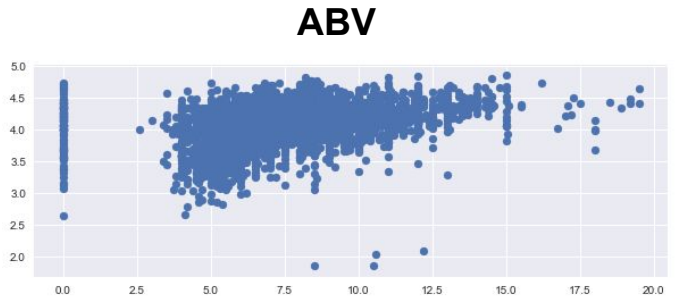
Feature engineering

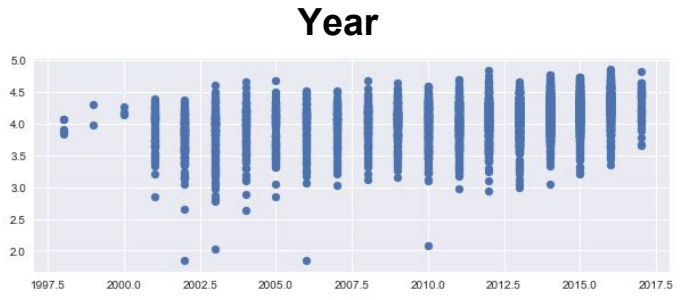
- Name
- Brewery
- Style
- Alcohol by volume (ABV)
- Year released
- State

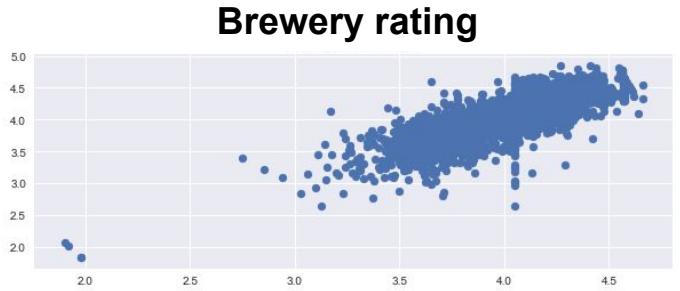
- Number of beers per brewery (proxy for brewery size)
- Brewery rating -- avg rating of all *other* beers by that brewery (avoid data leakage)
- Beer style taxonomy x2
  - Ales and lagers by region
  - Individual dummies
- State dummy variables

Models

- Linear regression with ridge or lasso regularization does well
  - R-squared around 0.63
- Lasso zeroes out nearly all beer style and state dummies







Can I predict beer ratings with only a few variables?

Tree models

- R-squared around 0.71 with random forest
- 30 trees in the forest with leaves no smaller than 10 observations
- Four variables
  - Brewery rating
  - Brewery size proxy
  - ABV
  - Year

Tree models

- R-squared around 0.71 with random forest
- 30 trees in the forest with leaves no smaller than 10 observations
- Four variables
  - Brewery rating
  - Brewery size proxy
  - ABV
  - Year

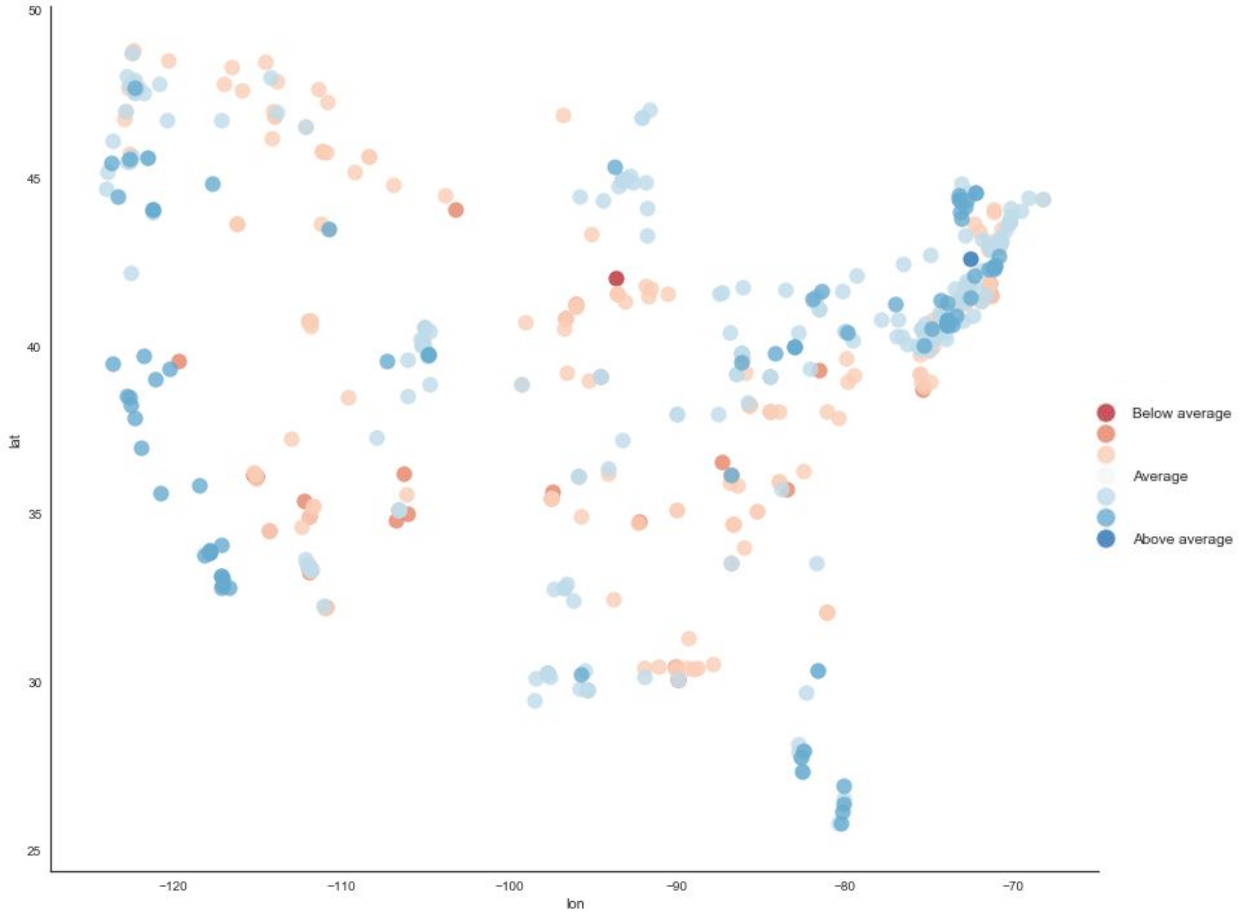
But can we do better?

Feature engineering

Take two

- Adjust treatment of missing values for brewery rating
  - Initially set to zero for breweries with one beer
  - Replace with average of all other beers (across breweries)
- Scrape zip code data

Average beer rating for breweries across the U.S.



Random forest

- Brewery avg rating
- Brewery size proxy
- ABV
- Year
- Zipcode

Random forest

- Brewery avg rating
- ABV
- Zipcode
- Year
- Brewery size proxy

R-squared ~0.73
with test data

With more time and data...

- Improve current model
  - Imputed values for missing ABV data
- Extend model
  - Lower rated beers
  - Other countries
- Data visualizations
  - Interactive preference map in d3
- Additional data
  - Beer description and tasting notes ➜ NLP
  - User reviews ➜ preference prediction

Appendix

Best model: feature importance

```
sorted_features(trees['rftree2'], Xtrain_lim3, Xtest_lim3)
```

```
Score: 0.720791216872
('brewery_avg_rating2', 0.774463770886770 21)
('abv', 0.10833019979025442)
('zipcode', 0.05460976310282061 9)
('year', 0.03568533744608153 7)
('brew_counts', 0.02691092877407327 5)
```

Modifying variables within best model

```
best model: five vars
Model: rftree2
Score: 0.728063155376

without avg_rating
Model: rftree2
Score: 0.604045114502

without avg_rating and zipcode
Model: rftree2
Score: 0.377929913887
```

Highest rated beer styles

	beer_style	rating
27	Bière de Champagne / Bière Brut	4.450000
59	Gueuze	4.373333
40	Eisbock	4.355000
11	American Double / Imperial Stout	4.283497
9	American Double / Imperial IPA	4.244880
20	American Wild Ale	4.242000
41	English Barleywine	4.237931
67	Lambic - Fruit	4.228667
54	Flanders Red Ale	4.208571
79	Russian Imperial Stout	4.207260

Breweries with highest avg beer rating

	brewery	lat	lon	rating
0	Brick & Feather Brewery	42.59	-72.55	4.650000
1	Monkish Brewing Co.	33.83	-118.31	4.600000
2	Sand City Brewing Co.	40.90	-73.34	4.584286
3	Bottle Logic Brewing	33.83	-117.86	4.583333
4	Moonraker Brewing Company	39.00	-121.09	4.570000
5	Tree House Brewing Company	42.09	-72.31	4.568750
6	Night Shift Brewing	42.40	-71.05	4.560000
7	Angry Chair Brewing	27.95	-82.48	4.514000
8	Sante Adairius Rustic Ales	36.97	-121.95	4.506000
9	Hangar 24 Brewery	34.06	-117.17	4.495000

Linear models

```
def run_linear_models(xtrain_data, xtest_data):
    models = {}
    models['lin_reg'] = LinearRegression()
    models['ridge'] = Ridge()
    models['lasso1'] = Lasso(alpha=.2)
    models['lasso2'] = Lasso(alpha=.02)
    models['lasso3'] = Lasso(alpha=.002)
    models['lasso4'] = Lasso(alpha=.0002)
    #models['lasso5'] = Lasso(alpha=.00002)
    models['elasticnet'] = ElasticNet()

    for name,model in models.items():
        model.fit(xtrain_data, ytrain)
        print('Model: ' + name)
        print("Score: " + str(model.score(xtest_data, ytest)))
        print("")
```

Tree models

```
def run_tree_models(xtrain_data, xtest_data):
    trees = {}

    trees['cart'] = tree.DecisionTreeRegressor(max_depth=7)
    trees['extratrees'] =tree.ExtraTreeRegressor(max_depth=7)
    trees['randomForest'] = RandomForestRegressor()
    trees['rftree1'] = RandomForestRegressor(n_estimators = 10, max_features='auto', min_samples_split=10)
    trees['rftree2'] = RandomForestRegressor(n_estimators = 30, max_features='auto', min_samples_split=10)
    trees['rftree3'] = RandomForestRegressor(n_estimators = 50, max_features='auto', min_samples_split=10)
    trees['rftree4'] = RandomForestRegressor(n_estimators = 100, max_features='auto', min_samples_split=10)
    trees['bagged_randomForest'] = BaggingRegressor(RandomForestRegressor())
    trees['adaboostedTrees'] = ensemble.AdaBoostRegressor()
    trees['gradboostedTrees'] = ensemble.GradientBoostingRegressor()

    for name,model in trees.items():
        model.fit(xtrain_data, ytrain)
        print('Model: ' + name)
        print("Score: " + str(model.score(xtest_data, ytest)))
        print("")
```

A quarter of states have 100 beers listed

State

```
count_per_state = df.groupby(['state'])['rating'].count()
```

```
# 25 states have 100 beers
# 51 'states' as DC is included
count_per_state.hist();
```