BeerRatingsAnalysis

August 13, 2018

1 Project: Multiple Regression Analysis for Beer Ratings

1.0.1 Kate Grosch and Lucas Baker

We would like to determine whether factors such as ABV, reviewer age, and beer appearance contribute to the overall rating of a beer, using data from online craft beer ratings.

To begin our analysis, we will load the project packages and the dataset.

```
In [1]: # loading the packages and modules

        %matplotlib inline

        # general packages
        import numpy as np
        import pandas as pd
        import sklearn
        import statsmodels.api as sm
        from collections import defaultdict

        # for statistics
        import scipy.stats as stats
        import statsmodels.api as sm

        # for visualizations
        import matplotlib.pyplot as plt
        from matplotlib import rcParams

        import seaborn as sns
        sns.set_style("whitegrid")
        sns.set_context("poster")
```

1.0.2 Important Notes about the Data

We do some basic data cleaning below: We remove ABV values outside of 3 standard deviations from the mean (to remove some junk ABV values we had) and we remove rows without a valid reviewer gender and age.

```
In [2]: #loading and cleaning the data

        df = pd.read_csv("beer.csv")

        df = df.rename(index=str, columns={'beer/ABV': 'ABV',
                                           'user/gender':'gender',
                                           'user/ageInSeconds':'reviewerAgeInSeconds',
                                           'review/overall':'overall',
                                           'beer/beerId':'beerId',
                                           'beer/brewerId':'brewerId',
                                           'review/appearance':'appearance',
                                           'review/aroma':'aroma',
                                           'review/palate':'palate',
                                           'review/taste':'taste',
                                           'review/timeUnix': 'unixPostTime'})

        # removing outliers and null values
        df = df[np.abs(df.ABV-df.ABV.mean()) <= (3*df.ABV.std())]
        df = df[df.reviewerAgeInSeconds < 2838240000]
        df = df[df.reviewerAgeInSeconds.notnull()]
        df = df[df.gender.notnull()]

        Y = df['overall']
        df = df.drop(['review/text',
                      'review/timeStruct',
                      'user/birthdayRaw',
                      'user/birthdayUnix',
                      'index',
                      'beer/style',
                      'user/profileName'], axis=1)

        df.describe()
```

```
Out[2]:                ABV        beerId      brewerId   appearance        aroma  \
        count  7664.000000   7664.000000   7664.000000  7664.000000  7664.000000
        mean      7.453027  21757.470511   3065.612474     3.911600     3.896399
        std       2.271367  18520.543349   5185.519409     0.591697     0.672122
        min       0.500000    175.000000      1.000000     1.000000     1.000000
        25%       5.500000   5441.000000    395.000000     3.500000     3.500000
        50%       7.000000  18313.000000   1199.000000     4.000000     4.000000
        75%       9.400000  34146.000000   1315.000000     4.500000     4.500000
        max      14.000000  77207.000000  26990.000000     5.000000     5.000000

                   overall       palate        taste  unixPostTime  \
        count  7664.000000  7664.000000  7664.000000  7.664000e+03
        mean      3.912317     3.866323     3.951983  1.236586e+09
        std       0.698713     0.669257     0.714298  6.580694e+07
        min       1.000000     1.000000     1.000000  9.932515e+08
```

```
25%       3.500000     3.500000     3.500000  1.198031e+09
50%       4.000000     4.000000     4.000000  1.248079e+09
75%       4.500000     4.500000     4.500000  1.289380e+09
max       5.000000     5.000000     5.000000  1.326257e+09

       reviewerAgeInSeconds
count          7.664000e+03
mean           1.154863e+09
std            2.556664e+08
min            7.034366e+08
25%            9.769754e+08
50%            1.094307e+09
75%            1.270825e+09
max            2.577622e+09
```

1.0.3 Model and Variables

Our full list of variables is ABV, brewer, appearance rating, aroma rating, palate rating, taste rating, review submission date, reviewer age, and reviewer gender. We got this dataset from Kaggle, and the CSV is available at https://github.com/katiegrosch/beer-data-analysis.

Before performing the analysis, we will preprocess the data to extract our explanatory features. Our desired y and xi's are as follows:

y: Overall rating, on a range of [1.0, 5.0]. x1 The number of reviews for the beer. x2 Average reviewer age. x3 Fraction of female reviewers. x4 Beer appearance rating, in the range [1.0, 5.0]. x5 Alcohol by volume. x6 Length of beer name.

We will be modeling the impact of the xi's on y in the form of the following multiple regression model, where i's are constants fit by least squares regression: y = 0 + 1x1 + 2x2 + 3x3 + 4x4 + 5x5 + 6x6

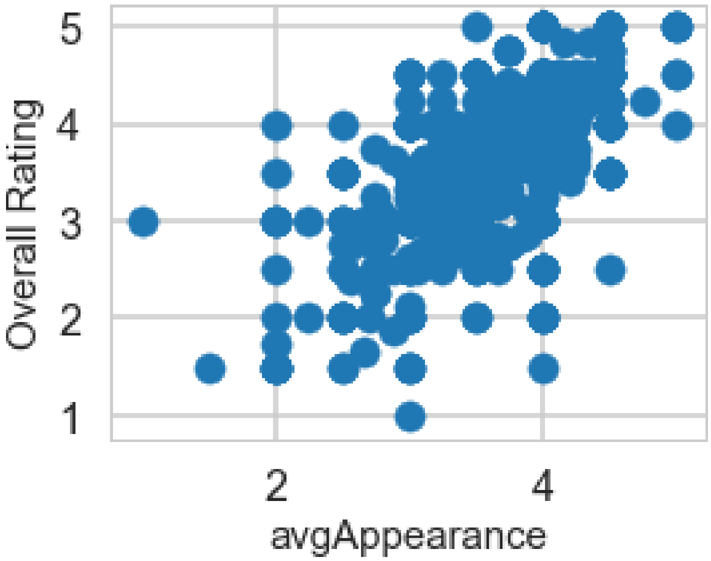
Let's reshape and group the data by beer:

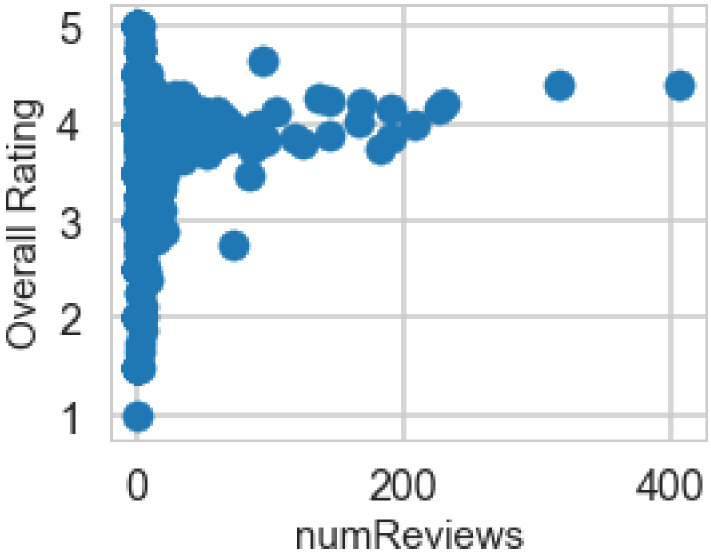
```
In [3]: df['beerNameLength'] = pd.Series([len(x) for x in df['beer/name']], index=df.index)

        beer_abv = dict()
        beer_overall = defaultdict(list)
        beer_appearance = defaultdict(list)
        reviewer_ages = defaultdict(list)
        reviewer_genders = defaultdict(list)
        for index, row in df.iterrows():
            name = row['beer/name']
            beer_abv[name] = row['ABV']  # Will overwrite, but same values
            beer_overall[name].append(row['overall'])
            beer_appearance[name].append(row['appearance'])
            reviewer_ages[name].append(row['reviewerAgeInSeconds'])
            reviewer_genders[name].append(0.0 if row['gender'].startswith("M") else 1.0)
        names = sorted(set(df['beer/name']))

        combined = {
            'beerName': names,
            'numReviews': [len(beer_overall[name]) for name in names],
            'ABV': [beer_abv[name] for name in names],
            'avgAppearance': [np.mean(beer_appearance[name]) for name in names],
            'avgReviewerAgeInYears': [np.mean(reviewer_ages[name]) /
                                      31557600 for name in names],
            'fractionFemale': [np.mean(reviewer_genders[name]) for name in names],
            'beerNameLength': [len(name) for name in names],
            'avgOverall': [np.mean(beer_overall[name]) for name in names]
        }

        # Save for later
        original = df.drop(['beerId',
                            'brewerId',
                            'unixPostTime',
                            'reviewerAgeInSeconds',
                            'gender'
                           ], axis=1)
        df = pd.DataFrame(combined)
        df.describe()
```

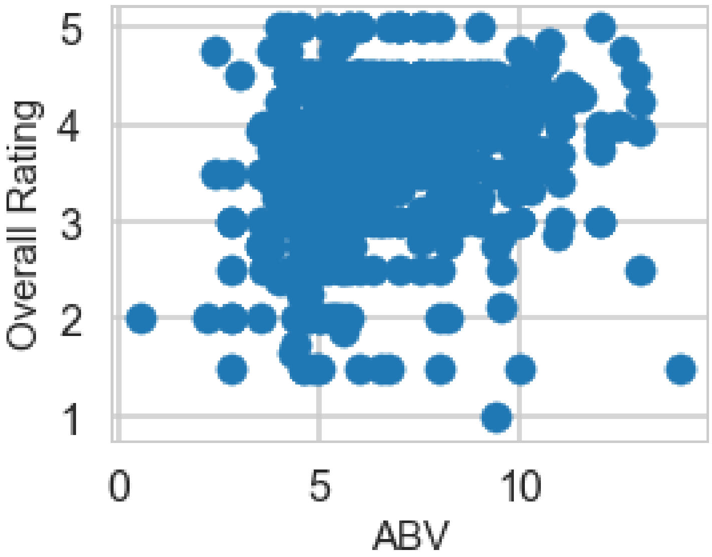
```
Out[3]:        numReviews         ABV  avgAppearance  avgReviewerAgeInYears  \
        count  758.000000  758.000000     758.000000             758.000000
        mean    10.110818    6.332691       3.676301              37.248549
        std     31.550733    1.985983       0.543039               6.424929
        min      1.000000    0.500000       1.000000              25.104959
        25%      1.000000    5.000000       3.500000              33.544099
        50%      2.000000    5.800000       3.756250              36.490340
        75%      5.000000    7.500000       4.000000              40.270028
        max    407.000000   14.000000       5.000000              74.991338

               fractionFemale  beerNameLength  avgOverall
        count      758.000000      758.000000  758.000000
        mean         0.011534       19.525066    3.677237
        std          0.082317        9.087787    0.676843
        min          0.000000        2.000000    1.000000
        25%          0.000000       13.000000    3.500000
        50%          0.000000       18.000000    3.833333
        75%          0.000000       25.000000    4.000000
        max          1.000000       72.000000    5.000000
```

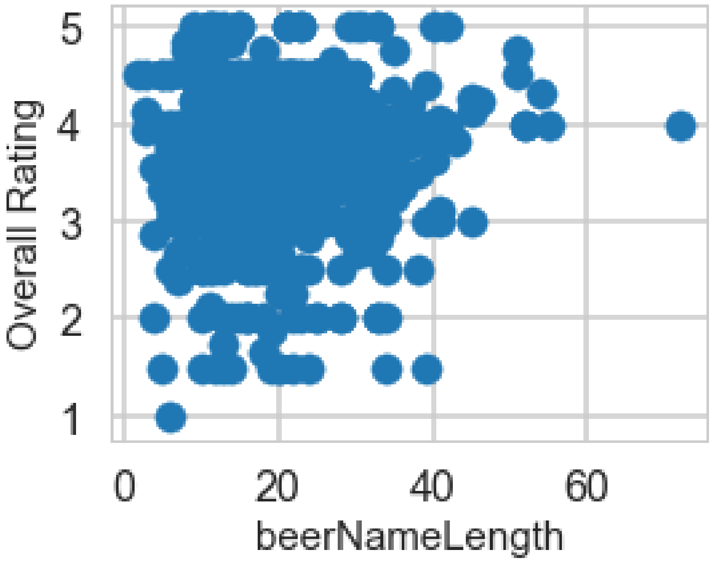
To begin our analysis, let's look at a graph of every xi versus our target variable (overall rating).

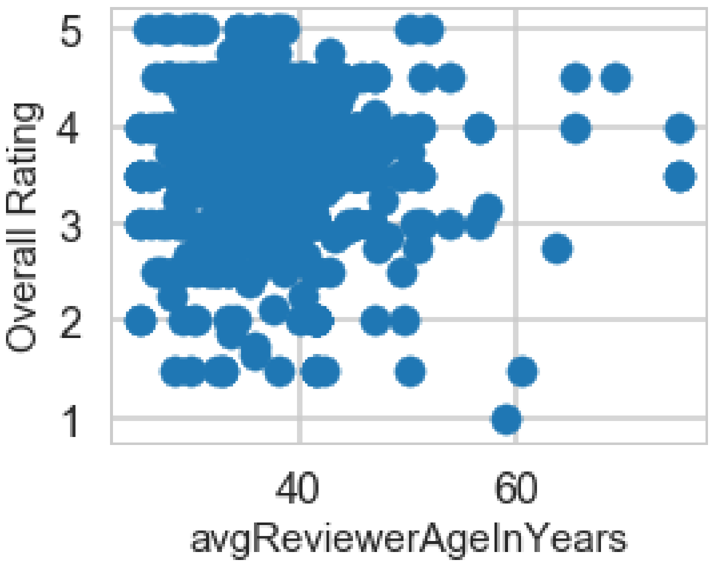
```
In [4]: Y = df['avgOverall']
        ratings = df.drop(['avgOverall', 'beerName'], axis=1)

        for column in ['avgAppearance',
                       'numReviews',
                       'ABV',
            'beerNameLength',
            'avgReviewerAgeInYears',
            'fractionFemale']:
plt.figure(figsize=(4, 3))
plt.scatter(ratings[column], Y)
plt.ylabel('Overall Rating', size=15)
plt.xlabel(column, size=15)
```

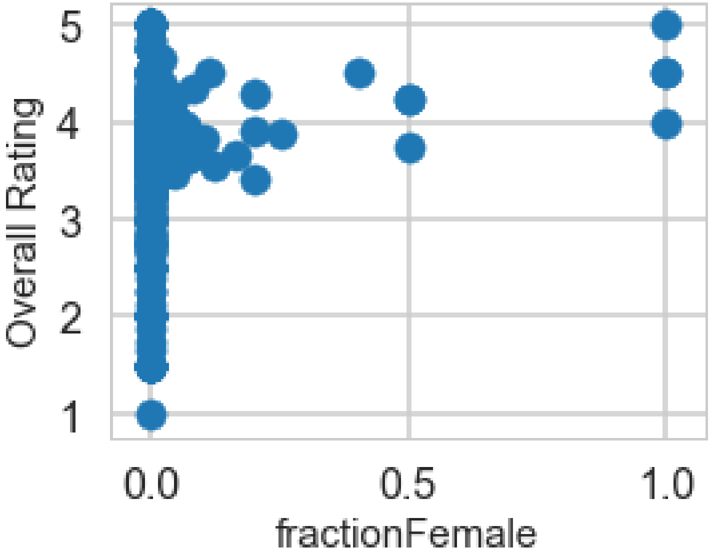





Overall Rating
5
4
3
2
1
0
5
10
ABV


Overall Rating
5
4
3
2
1
0
20
40
60
beerNameLength



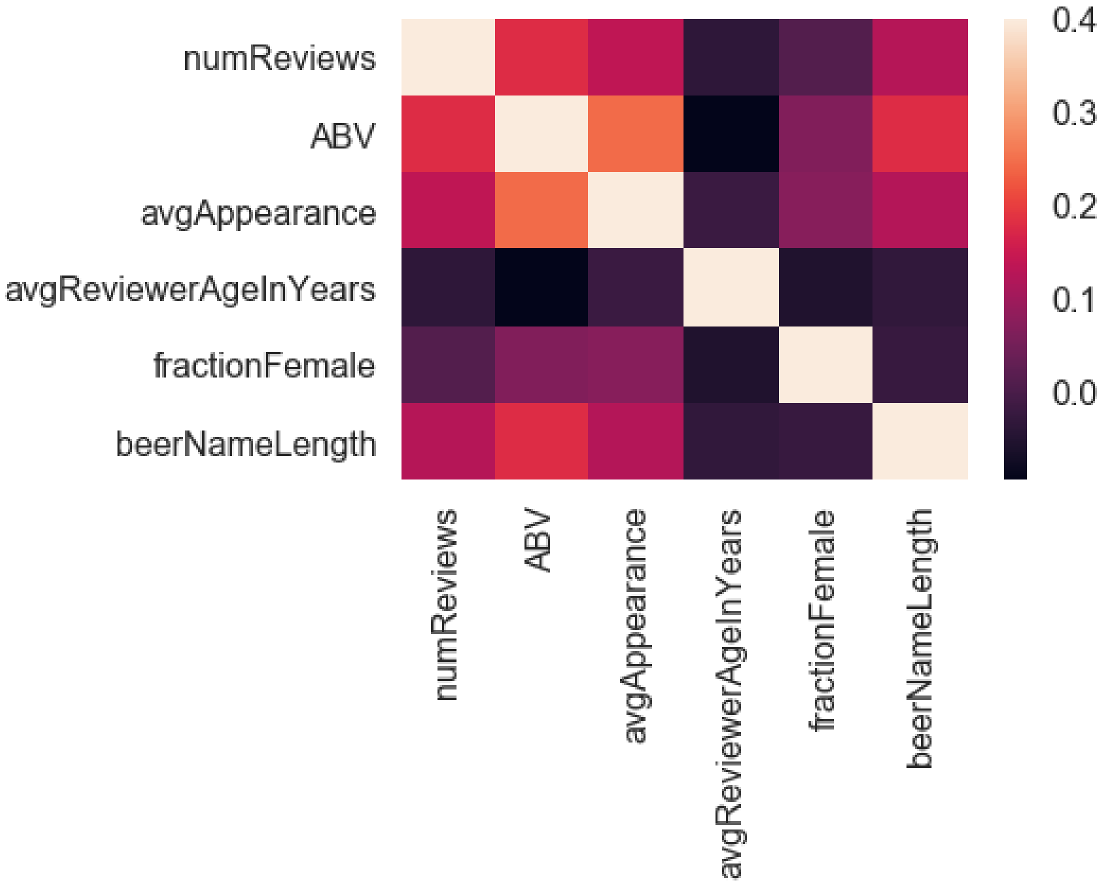


1.0.4 Variable Correlations

Visually speaking, there appears to be a strong correlation between appearance rating and overall rating, as might be expected. Number of reviews also appears at least somewhat linked to quality: while there are many beers with few reviews at all levels, the most frequently reviewed beers are rated above 4.0, relative to a mean average overall rating of 3.67. The other explanatory variables seem to have a more tenuous connection. Plots for ABV, beer name length, and average reviewer age in years appear as a fairly undifferentiated mass, and while it is clear that female reviewers give a higher rating on average, the average fraction of female viewers is 1.1%! Detecting a strong effect on such a lopsided distribution may be difficult.

Let's also see if any of the variables correlate with each other. If we encounter multicollinearity, it may make sense to remove one of the correlated variables:

```
In [5]: corrmat = ratings.corr()
        sns.heatmap(corrmat, vmax=.4);
```



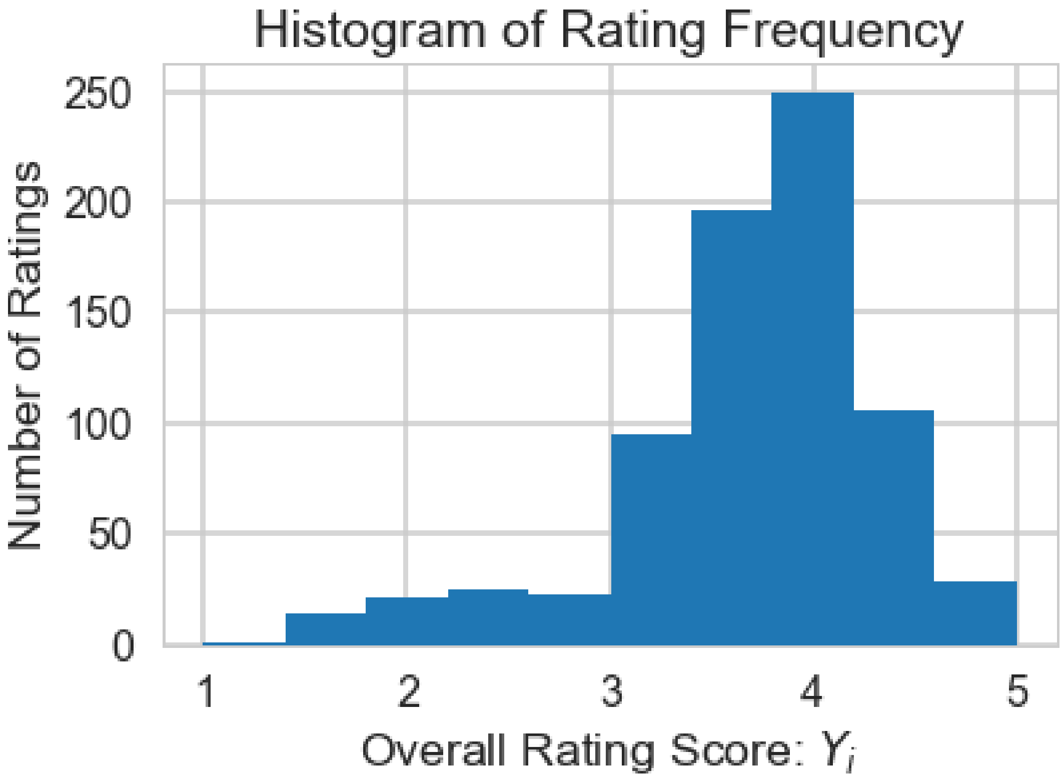
1.0.5 Variable correlations in the heatmap

In the confusion matrix above, lighter colors imply higher correlation. Trivially, each variable has a perfect correlation with itself. We can confirm that none of these variables correlate strongly with one another, which is reasonable given the idiosyncratic selection. However, ABV and appearance rating show at least a weak relationship, which may suggest a correlation between ABV, appearance, and a third variable such as style. (For example, an Imperial Stout might tend to be both stronger and darker in color.)

Before we create our regression model, let's look at a histogram of the ratings to get a sense of where they cluster. It looks like the ratings are left-skewed, but overall have a near-normal distribution. We confirm by calculating the skew of the data to be -0.9899, which might affect our model.

```
In [6]: plt.hist(Y)
        plt.xlabel("Overall Rating Score: $Y_i$")
        plt.ylabel("Number of Ratings")
        plt.title("Histogram of Rating Frequency")

        print("Skew: %f" % Y.skew())
```

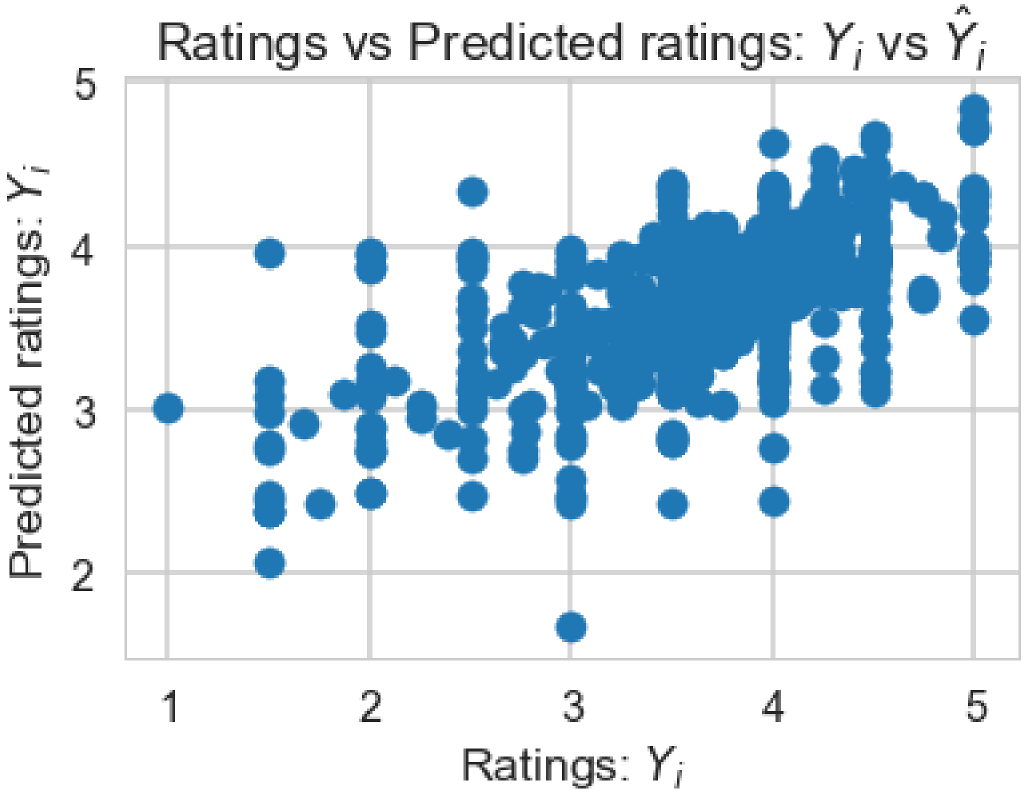
```
Skew: -0.989947
```



1.0.6 Building the Linear Regression

```
In [7]: from sklearn.linear_model import LinearRegression

        lr = LinearRegression()
        lr = lr.fit(ratings, Y)
        y_pred = lr.predict(ratings)

        plt.scatter(Y, y_pred)
        plt.xlabel("Ratings: $Y_i$")
        plt.ylabel("Predicted ratings: $\hat{Y}_i$")
        plt.title("Ratings vs Predicted ratings: $Y_i$ vs $\hat{Y}_i$")
```

```
Out[7]: Text(0.5,1,'Ratings vs Predicted ratings: $Y_i$ vs $\\hat{Y}_i$')
```



```
In [8]: from sklearn.metrics import mean_squared_error

        mse = mean_squared_error(Y, y_pred)
        print("Mean Squared Error: %f" % mse)
```

```
Mean Squared Error: 0.281744
```

The variance of the estimator 2 is close to MSE, but takes into account the degrees of freedom lost by the 6 regressors and the intercept:

```
In [9]: print("Variance of error: SSE / (n - (k + 1)) = ", mse * len(Y) / (len(Y) - 7))
```

```
Variance of error: SSE / (n - (k + 1)) =  0.28436978879024455
```

```
In [10]: print("Beta coefficients: ")
         print(lr.coef_)
         print("Largest coefficient: %f" % np.amax(lr.coef_))
         print("R-squared: %f" % sklearn.metrics.r2_score(Y, y_pred))
```

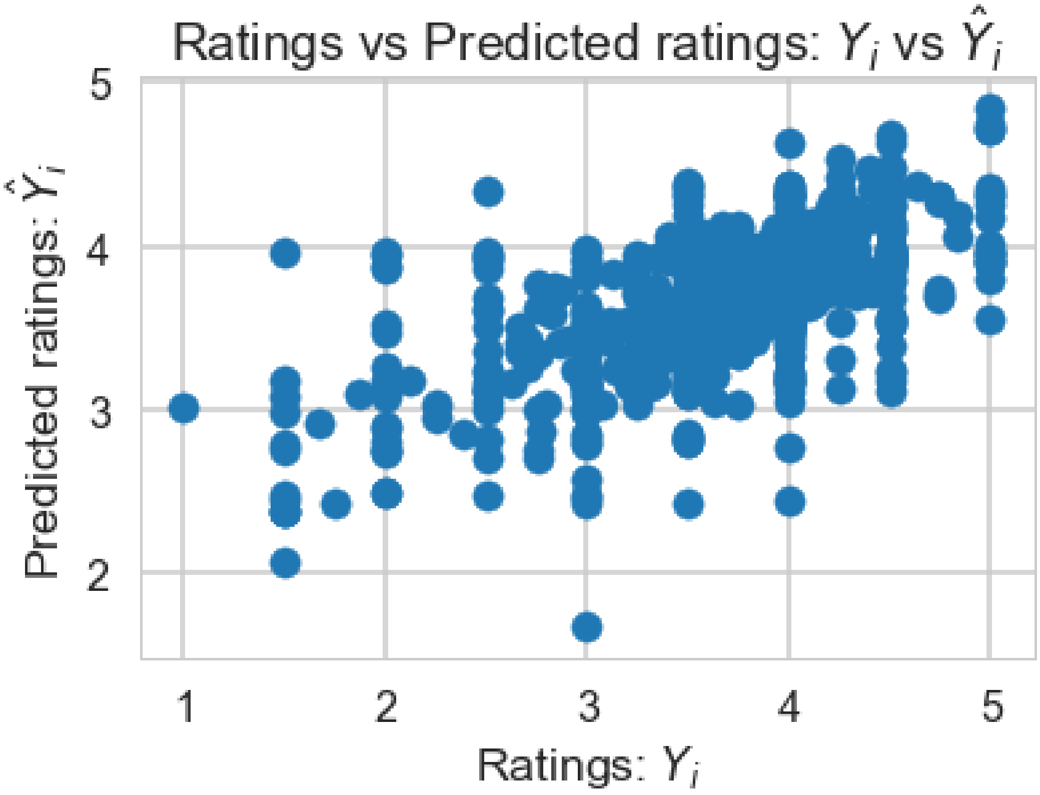
```
Beta coefficients:
[ 5.42577301e-04  1.99656227e-03  7.50113235e-01 -7.39405511e-03
  5.36400647e-01 -1.01190727e-03]
Largest coefficient: 0.750113
R-squared: 0.384182
```

The R-squared value is disappointingly low, but not so low as to suggest that our predictors are useless. Does the model have any significant variables?

The time has come to calculate confidence intervals, but remarkably, scikit-learn does not provide a built-in method of accessing model statistics. Let's switch to statsmodel, which does. The scikit model will provide a reference by which to confirm our numbers. (This sanity check is quite valuable: for instance, if the intercept is not explicitly added in statsmodel, it will not be included and the resulting R-squared will be unreasonably high.)

```
In [11]: ratings['intercept'] = np.ones(len(ratings))
         reg = sm.OLS(Y, ratings).fit()
         y_pred = reg.predict(ratings)

         plt.scatter(Y, y_pred)
         plt.xlabel("Ratings: $Y_i$")
         plt.ylabel("Predicted ratings: $\hat{Y}_i$")
         plt.title("Ratings vs Predicted ratings: $Y_i$ vs $\hat{Y}_i$")

         print(reg.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             avgOverall   R-squared:                       0.384
Model:                            OLS   Adj. R-squared:                  0.379
Method:                 Least Squares   F-statistic:                     78.09
Date:                Mon, 13 Aug 2018   Prob (F-statistic):           9.20e-76
Time:                        20:45:08   Log-Likelihood:                -595.45
No. Observations:                 758   AIC:                             1205.
Df Residuals:                     751   BIC:                             1237.
Df Model:                           6
Covariance Type:            nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
numReviews               0.0005      0.001      0.861      0.390      -0.001       0.002
ABV                      0.0020      0.010      0.193      0.847      -0.018       0.022
avgAppearance            0.7501      0.037     20.167      0.000       0.677       0.823
avgReviewerAgeInYears   -0.0074      0.003     -2.437      0.015      -0.013      -0.001
fractionFemale           0.5364      0.237      2.265      0.024       0.071       1.001
beerNameLength          -0.0010      0.002     -0.463      0.644      -0.005       0.003
intercept                1.1905      0.182      6.530      0.000       0.833       1.548
==============================================================================
Omnibus:                       83.947   Durbin-Watson:                   1.871
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              166.129
Skew:                          -0.671   Prob(JB):                     8.43e-37
Kurtosis:                       4.860   Cond. No.                         557.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```



As shown, the 95% (= .05) confidence intervals for the regressors are as follows:

CI(intercept, = .05) = [0.833, 1.548] CI(numReviews, .05) = [-0.001, 0.002] CI(ABV, .05) = [-0.018, 0.022] CI(avgAppearance, .05) = [0.677, 0.823] CI(avgReviewerAgeInYears, .05) = [-0.013, -0.001] CI(fractionFemale, .05) = [0.071, 1.001] CI(beerNameLength, .05) = [-0.005, 0.003]

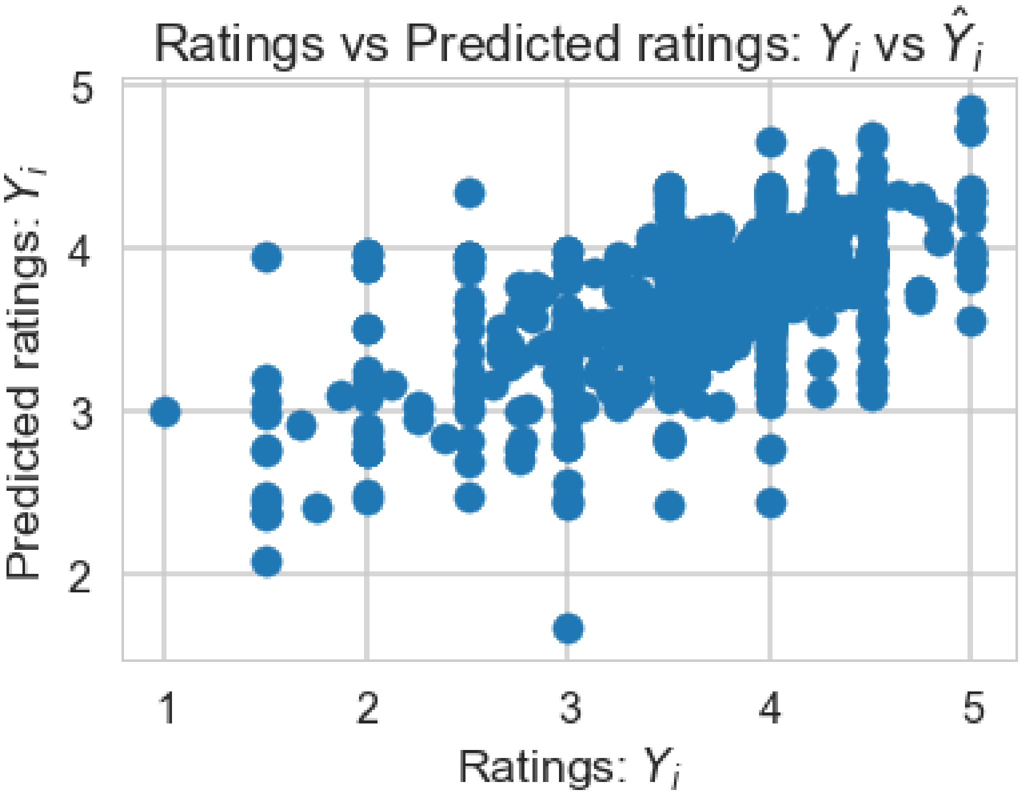
A one-star increase in appearance rating, all else equal, produces an extra .75 stars in overall rating. Women also rate just over half a star higher, and older reviewers about .0074 stars lower per year. All three of these variables are significant at the = .05 level, even the reviewer age, which is an excellent demonstration of the distinction between significance and effect size. The other three variables (name length, number of reviews, and ABV) have high p-values, are not significant, and bear little apparent relation to overall rating.

The F-statistic of the model is 78.09, well over the critical value, as suggested by the F-test p-value of 9.20e-76. Thus, despite the quirky choice of variables, the resulting model is clearly significant.

Having evaluated the significance of all regressors, we will leave in appearance rating, fraction of female reviewers, and average reviewer age while removing the others to build the final model.

```
In [12]: ratings_final = ratings.drop(['numReviews', 'ABV', 'beerNameLength'], axis=1)
         reg_final = sm.OLS(Y, ratings_final).fit()
         y_pred_final = reg_final.predict(ratings_final)

         plt.scatter(Y, y_pred_final)
         plt.xlabel("Ratings: $Y_i$")
         plt.ylabel("Predicted ratings: $\hat{Y}_i$")
         plt.title("Ratings vs Predicted ratings: $Y_i$ vs $\hat{Y}_i$")

         print(reg_final.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             avgOverall   R-squared:                       0.383
Model:                            OLS   Adj. R-squared:                  0.381
Method:                 Least Squares   F-statistic:                     156.3
Date:                Mon, 13 Aug 2018   Prob (F-statistic):           9.24e-79
Time:                        20:45:08   Log-Likelihood:                -595.94
No. Observations:                 758   AIC:                             1200.
Df Residuals:                     754   BIC:                             1218.
Df Model:                           3
Covariance Type:            nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
avgAppearance             0.7541      0.036     21.096      0.000       0.684       0.824
avgReviewerAgeInYears    -0.0075      0.003     -2.481      0.013      -0.013      -0.002
fractionFemale            0.5424      0.236      2.297      0.022       0.079       1.006
intercept                 1.1776      0.175      6.726      0.000       0.834       1.521
==============================================================================
Omnibus:                       86.648   Durbin-Watson:                   1.868
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              170.540
Skew:                          -0.691   Prob(JB):                     9.28e-38
Kurtosis:                       4.868   Cond. No.                         464.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```



In the new model, the F-statistic has roughly doubled (156.3 vs 78.09) and the p-value dropped by a factor of 1000 (9.24e-79 vs 9.20e-76), while the adjusted R-squared has also increased slightly (0.381 vs 0.379). The raw R-squared has decreased slightly (0.383 vs 0.384), but this is to be expected because additional regressors can only increase the R-squared value. From the rise in adjusted R-squared and F-statistic values, we may conclude that the new model with fewer variables is superior.

We would also like to look at the importance of each i. How large is the average impact of each variable on the final prediction? This can be evaluated by looking at the product of each i with its average, minimum, and maximum input.

```
In [13]: for c in ['avgAppearance', 'avgReviewerAgeInYears', 'fractionFemale']:
             col = ratings_final[c]
             vals = [min(col), np.mean(col), max(col)]
             print('Min, mean, max effect sizes for %s:' % c)
             print('%f, %f, %f\n' % tuple(reg_final.params[c] * x for x in vals))
```

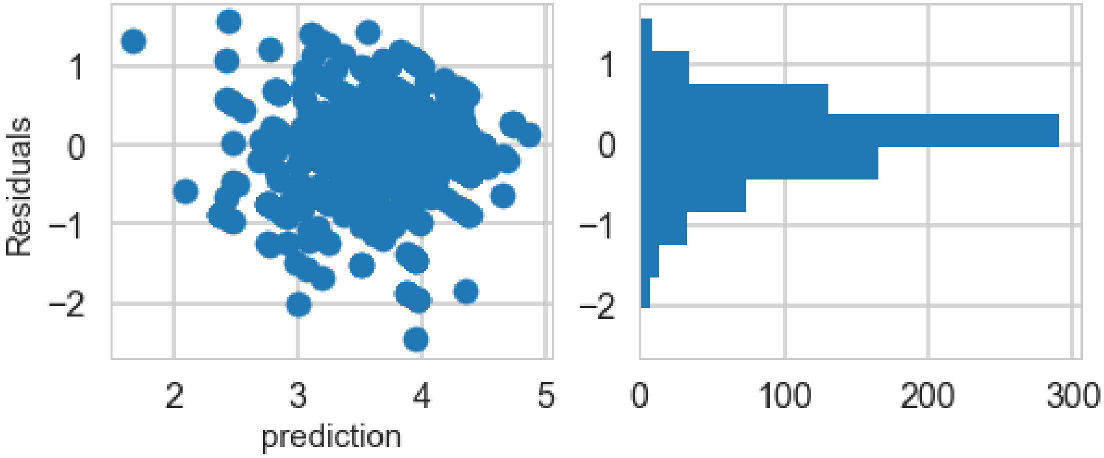
```
Min, mean, max effect sizes for avgAppearance:
0.754090, 2.772262, 3.770450

Min, mean, max effect sizes for avgReviewerAgeInYears:
-0.187956, -0.278873, -0.561445
```

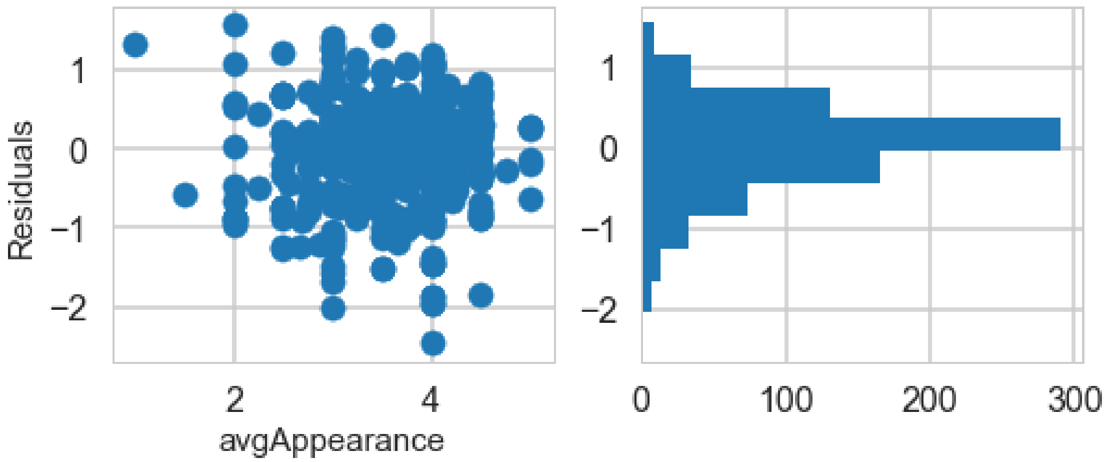
```
Min, mean, max effect sizes for fractionFemale:
0.000000, 0.006256, 0.542356
```

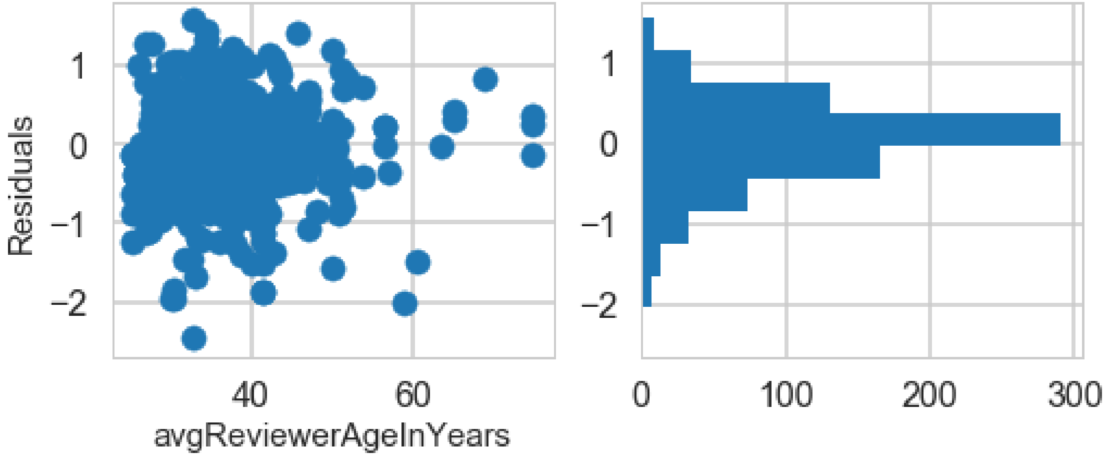
Appearance is obviously the most important predictor, but surprisingly, the impact of reviewer age is much larger than that of gender even though the gender coefficient is much larger. This is an issue of units: if we had used, for instance, fractions of an average human lifespan of 80 years, the reviewer age coefficient would have been much higher in magnitude. In any case, we can conclude that the order of regressor importance is (1) average appearance, (2) average reviewer age, and (3) fraction female.

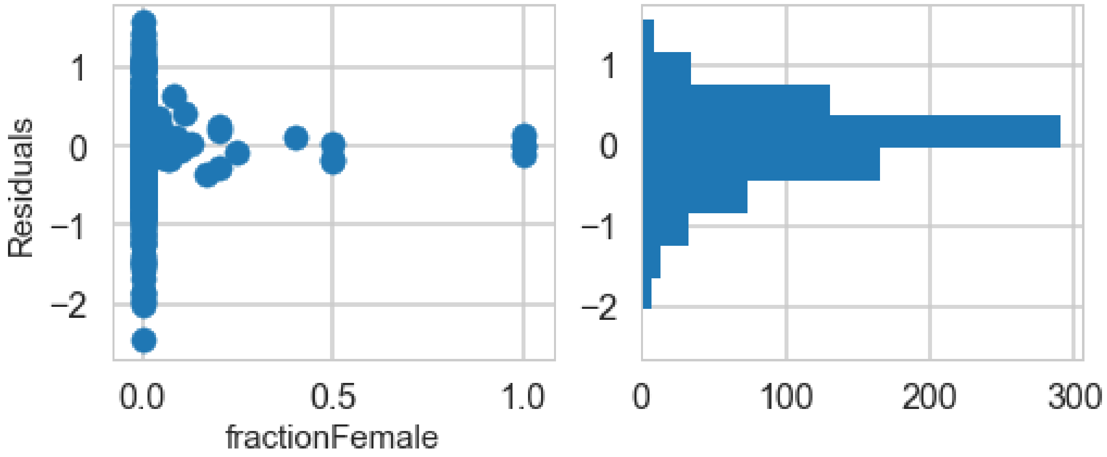
Looking back, does the error in our data fit the Gaussian assumptions of a normal, zero-mean, constant variance random variable? Earlier, we calculated the skew to be -0.9899, so the ratings themselves are not quite normally distributed. However, multiple linear regression can still work effectively if the residuals are near-random. We now plot the residuals versus the predictions as well as each xi to confirm this assumption:

```
In [14]: columns = [y_pred_final]
         columns.extend(ratings_final[c] for c in ['avgAppearance',
                                                   'avgReviewerAgeInYears',
                                                   'fractionFemale'])


         for i, name in enumerate(['prediction',
                                   'avgAppearance',
                                   'avgReviewerAgeInYears',
                                   'fractionFemale']):
             plt.figure(figsize=(8, 3))
             plt.subplot(1, 2, 1)
             plt.scatter(columns[i], Y - y_pred_final)
             plt.ylabel('Residuals', size=15)
             plt.xlabel(name, size=15)
             plt.subplot(1, 2, 2)
             plt.hist(Y - y_pred_final, 10, orientation='horizontal')
```




Residuals
1
0
−1
−2
2
4
avgAppearance
1
0
−1
−2
0
100
200
300


Residuals
1
0
−1
−2
40
60
avgReviewerAgeInYears
1
0
−1
−2
0
100
200
300



The distribution of the data is not ideal, as we would much prefer to see an even spread of values along the x-axis. However, there is no clear evidence of heteroscedasticity, nonlinear patterns, or serious outliers.

There does appear to be asymmetry across the x-axis, where there are more negative residuals, with higher average magnitudes, than the positive ones. This imbalance reflects the skew in the initial distribution, where ratings are generally clustered around the 3-4 star range and rarely fall below. Since ratings are bounded at 1.0 and 5.0, there is more room for error by predicting too low than too high. Interpreting this asymmetry is a judgment call, but there is nothing here to suggest that the data would be better served by something other than a linear model.

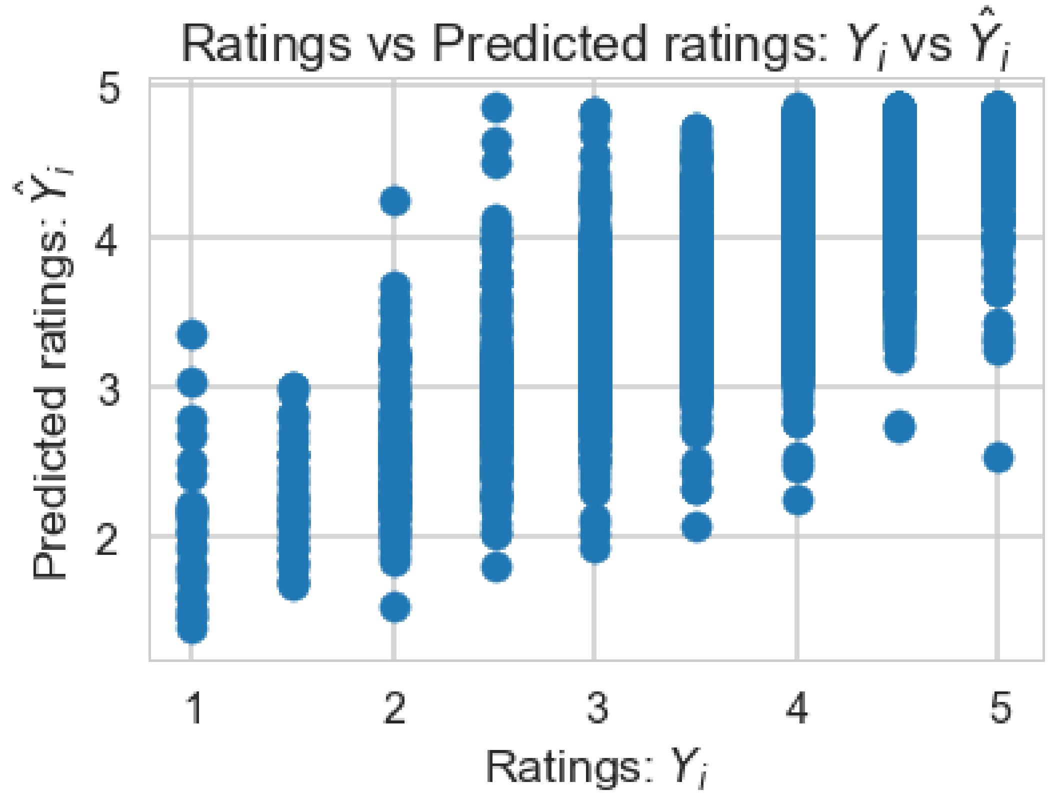
1.1 Discussion

Our project aimed to investigate how well multiple linear regression can predict overall beer rating based on the six features selected. We report that our selection of variables predicts overall rating with significance, but with low precision. One variable, average appearance rating, correlated extremely well with overall rating, while the fraction of female reviewers and average reviewer age also proved significant at the = .05 level. The other three variables, ABV, beer name length, and number of reviews, showed no significant predictive ability. Our final model predicted overall rating from appearance rating, fraction of female reviewers, and average reviewer age with an R-squared of 0.383 and adjusted R-squared of 0.381, an F-statistic of 156.3 with p-value of 9.24e-79, and p-values of 0.000, 0.022, and 0.013, respectively, for the three regressors. While our original data exhibited skew that led to a somewhat asymmetric residual distribution, there was no observed heteroscedasticity, nonlinear patterns, or serious outliers that would argue against the selection of a linear model.

Overall, the above investigation serves as a good illustration of the difference between significance and usefulness. The relationships identified by regression were certainly significant, but if model effectiveness were a priority, the most desirable option would be to gather more predictive data. For example, the pure rating data is more predictive than our derived features, with an adjusted R-squared of 0.629 even before aggregation:

```
In [15]: original['intercept'] = np.ones(len(original))
         original_ratings = original[['appearance',
                                      'aroma',
                                      'palate',
                                      'taste',
                                      'intercept']]
         original_Y = original['overall']
         reg_original = sm.OLS(original_Y, original_ratings).fit()
         y_pred_original = reg_original.predict(original_ratings)

         plt.scatter(original_Y, y_pred_original)
         plt.xlabel("Ratings: $Y_i$")
         plt.ylabel("Predicted ratings: $\hat{Y}_i$")
         plt.title("Ratings vs Predicted ratings: $Y_i$ vs $\hat{Y}_i$")

         print(reg_original.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                overall   R-squared:                       0.629
Model:                            OLS   Adj. R-squared:                  0.629
Method:                 Least Squares   F-statistic:                     3246.
Date:                Mon, 13 Aug 2018   Prob (F-statistic):               0.00
Time:                        20:45:09   Log-Likelihood:                -4327.2
No. Observations:                7664   AIC:                             8664.
Df Residuals:                    7659   BIC:                             8699.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
appearance     0.0349      0.010      3.400      0.001       0.015       0.055
aroma          0.0618      0.011      5.820      0.000       0.041       0.083
palate         0.2459      0.011     22.508      0.000       0.224       0.267
taste          0.5335      0.011     47.522      0.000       0.511       0.556
intercept      0.4760      0.036     13.208      0.000       0.405       0.547
==============================================================================
Omnibus:                      391.814   Durbin-Watson:                   1.999
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              901.117
Skew:                          -0.322   Prob(JB):                    2.11e-196
Kurtosis:                       4.552   Cond. No.                         59.6
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```



Sometimes one finds fascinating relationships in data sources that do not initially appear connected, and this is the type of data science journalists love to write stories about. The rest of the time, the data that looks connected likely is and that which doesn't isn't. Nonetheless, we have learned at least one non-obvious lesson: all else being equal, young women appear to look more kindly on the beer they drink.