Project McNulty: Identifying craft beer style

Natalia Lichagina

Presentation date: Feb 13th, 2019

Summary of completed work

Problem statement

The goal was to build a model that would identify what style a beer is depending on its key characteristics: ABV (alcohol by volume), IBU (measure of bitterness), and color (measure of how dark the beer is). The data is a set of self-reported brew-at-home recipes, so not all labels were to be fully trusted.

Gathering data

I downloaded the file from a Kaggle dataset: https://www.kaggle.com/jtrofe/beer-recipes/version/3#recipeData.csv

It's 74K recipes scraped from website Brewer's Friend, covering 176 beer styles in total.

Data processing

Jupyter notebook: *Code.ipynb*

Tableau notebook (packaged with data): *Beer_20190204_with_data.twbx*

I started by looking at my data in Tableau, identifying what bad data to filter out, which beer styles to group together, and which ones to exclude altogether from analysis. I wanted to retain as much data as possible while making the groups distinctive enough across the three dimensions that the model would be able to separate them out.

Once I was satisfied, I reworked this logic in SQL. (In reality, of course, it took several rounds of group -> model -> go back to revise group...)

Here's the final SQL code so you don't have to dig for it in my notebook :)

```sql
CREATE VIEW step1 as (
    SELECT
        beer_list.*
        ,CASE
            WHEN UPPER(style) LIKE '%DOUBLE IPA%' or UPPER(style)
                LIKE '%IMPERIAL IPA%' then 'Double IPA'
            WHEN UPPER(style) LIKE '%IPA%' and not UPPER(style)
                LIKE '%BLACK%' then 'IPA'
            WHEN UPPER(style) LIKE '%AMERICAN PALE ALE%'
                then 'American Pale Ale'
            WHEN UPPER(style) LIKE '%BLOND%' or UPPER(style)
                LIKE '%LSCH%' or UPPER(style) LIKE '%CREAM%'
                or UPPER(style) LIKE '%WHEAT %' or
                UPPER(style) LIKE '%WITBIER%'
                or UPPER(style) LIKE '%WEISSBIER%' or
                UPPER(style) LIKE '%PILS%'
                or UPPER(style) LIKE '%PROHIBITION LAGER%'
                then 'Blonde Ale/Wheat Beer/Pilsner'
            WHEN UPPER(style) LIKE '%IMPERIAL STOUT%'
                then 'Imperial Stout'
            WHEN UPPER(style) LIKE '%PORTER%' or UPPER(style)
                LIKE '%STOUT%'  then 'Porter/Stout'
            WHEN UPPER(style) LIKE '%BELGIAN TRIPEL%'
                or UPPER(style) LIKE '%GOLDEN STRONG%'
                then 'Belgian Tripel'
            WHEN UPPER(style) LIKE '%BROWN ALE%' or
                UPPER(style) LIKE '%AMBER%' or
                UPPER(style) LIKE '%RED%'
                or ( UPPER(style) LIKE '%DARK%'
                    and UPPER(style) LIKE '%LAGER%' )
                or UPPER(style) LIKE '%DUNKLES%'
                or UPPER(style) LIKE '%BELGIAN%'
                then 'Brown Ale/Amber/Belgian'
            ELSE 'zOther'
            END as style_group
    FROM beer_list
    WHERE ibu > 0
        and abv >= 2 and abv <= 15
        and color >= 2 and color <= 80
        and style not in ('N/A','Specialty Beer',
                          'Experimental Beer',
                          'Spice  Herb  or Vegetable Beer',
                        'Holiday/Winter Special Spiced Beer')
    );
```

Since my data was quite messy, I removed all outliers from each group - any observation that was over 2 standard deviations away from the group mean on any dimension.

```
WITH outliers as (
SELECT
    step1.*
    ,AVG(abv) over (PARTITION BY style_group) as group_avg_abv
    ,STDDEV(abv) over (PARTITION BY style_group) as group_stdev_abv
    ,AVG(ibu) over (PARTITION BY style_group) as group_avg_ibu
    ,STDDEV(ibu) over (PARTITION BY style_group) as group_stdev_ibu
    ,AVG(color) over (PARTITION BY style_group) as group_avg_color
    ,STDDEV(color) over (PARTITION BY style_group) as group_stdev_color
FROM step1
)

SELECT
    style_group
    ,abv
    ,ibu
    ,color
FROM outliers
WHERE ABS(abv - group_avg_abv)<=2*group_stdev_abv
    AND ABS(ibu - group_avg_ibu)<=2*group_stdev_ibu
    AND ABS(color - group_avg_color)<=2*group_stdev_color
    AND style_group <> 'zOther'
```

Modeling

- I used **Random Forest, KNN, Logistic, SVC**, and **Naive Bayes** for this project
- Since I changed my project topic halfway through and was short on time, I left figuring out how to get the Random Search to work until after the presentation. Unfortunately, I was not able to get it to work in the end - I shut down the kernel late last night after it had been running for over 5 hours for just one of the models. Therefore, my modeling was done with base versions of the models without hyperparameter tuning (the only slight exception was that the base Logistic Regression failed to converge, so I increased the number of iterations to 200 and then it worked, but that was purely heuristic.)
- The metrics I focused on were **accuracy** and **F1** - there was no obvious reason to prioritize precision or recall given the nature of my problem: no life-or-death stakes in classifying beer, and the labels are quite suspect to begin with!
- I ran a 4-fold cross-validation.
- All of the models performed very similarly in their prediction quality. While SVC did the best, it also took the longest and I wasn't particularly eager to try tunning it on the whole CV dataset, so I chose Random Forest as my final winner.

:

	Model	Avg. Accuracy	Avg. Precision	Avg. Recall	Avg. F1-score	Total fit time	Total score time	Total CV time
0	Random Forest	0.78	0.80	0.79	0.79	8.51	2.75	11.26
1	KNN	0.77	0.79	0.78	0.78	0.17	0.82	0.99
2	Logistic	0.78	0.80	0.78	0.79	2.69	0.37	3.06
3	SVC	0.79	0.81	0.79	0.80	15.79	28.29	44.08
4	Naive Bayes	0.77	0.78	0.78	0.78	0.17	0.37	0.54

- The final scores for Random Forest trained on the whole CV dataset and fitted on test were:

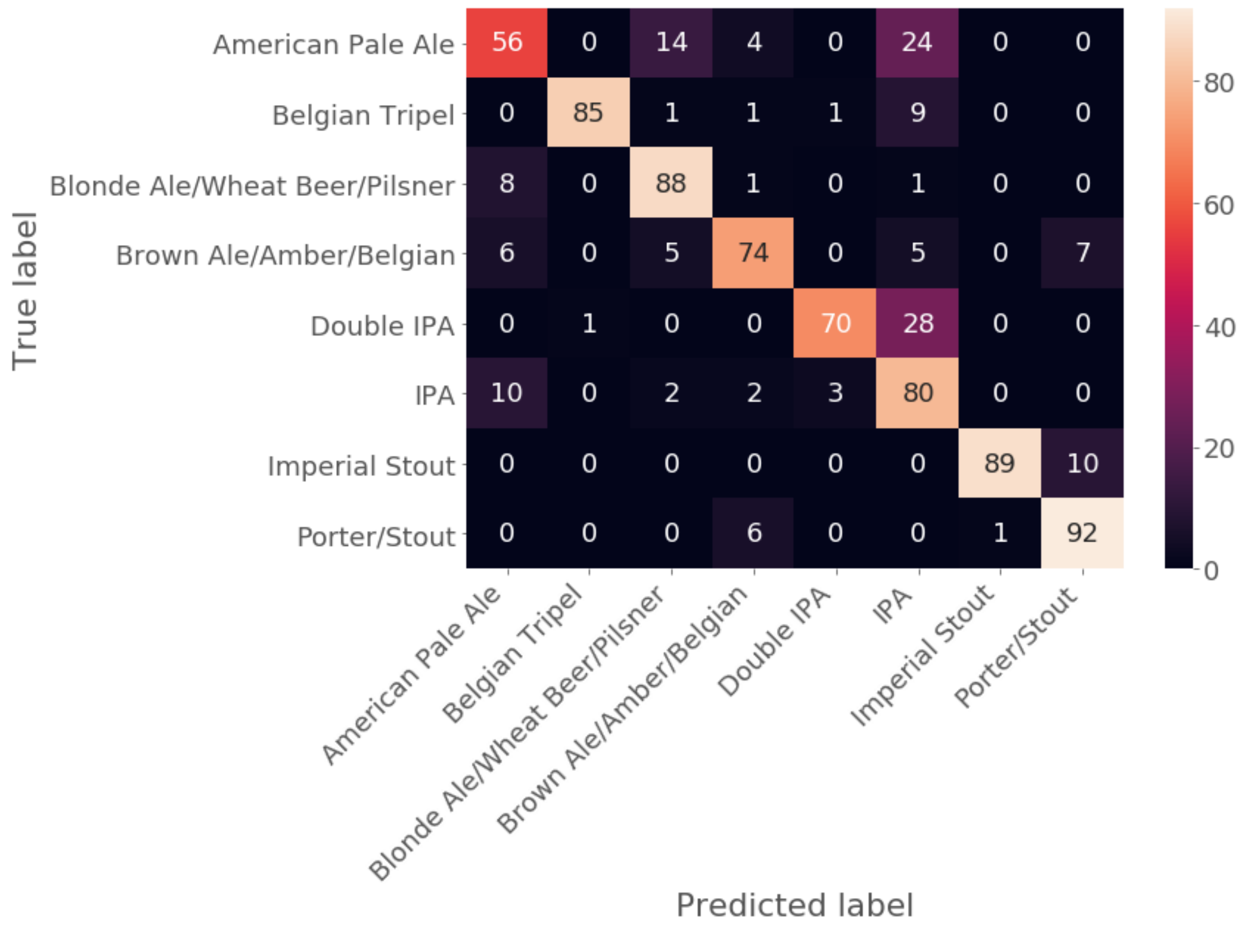
  Accuracy: 0.79

Precision: 0.81

Recall: 0.80

F-score: 0.81

- To summarize final predictions in an intuitive way for my unbalanced dataset, I normalized the confusion matrix to show predictions as % of original population rather than raw counts.



(note that this is slightly different from what I showed in my presentation - that's because I've rerun the code since, and every time the results change a little due to random train/test splits)

Conclusion

Overall, I'm satisfied with how the model worked out - the final results made sense and were easy to visualize in Tableau for intuitive double-checking on why some styles were harder for the model to classify than others. This model is mostly just for fun; the output can be used to identify beer styles adjacent to the ones you already like to branch out to, and, of course, to help label whatever concoction you might brew up at home. 😎👩🏻‍🍳

Future work

- Figure out Random Search for all those models, maybe use AWS to run it?
- Use a clustering algorithm to create the initial style groups
- Build a pretty Flask app to let people plug in values for ABV/IBU/Color and get predictions out