Beer Data Set Analysis

About the data source

- For this exercise, you will be working with beer data which can be downloaded from here https://drive.google.com/open?id=1e-kyoB97a5tnE7X4T4Es4FHi4g6Trefq

- Unzip the file and you should see a CSV file, called “BeerDataScienceProject.csv”

- The columns are
  1. beer_ABV
  2. beer_beerId
  3. beer_brewerId
  4. beer_name
  5. beer_style
  6. review_appearance
  7. review_palette
  8. review_overall
  9. review_taste
  10. review_profileName
  11. review_aroma
  12. review_text
  13. review_time

Questions

- Rank top 3 Breweries which produce the strongest beers?
- Which year did beers enjoy the highest ratings?
- Based on the user’s ratings which factors are important among taste, aroma, appearance, and palette?
- If you were to recommend 3 beers to your friends based on this data which ones will you recommend?
- Which Beer style seems to be the favorite based on reviews written by users?
- How does written review compare to overall review score for the beer styles?
- How do find similar beer drinkers by using written reviews only?

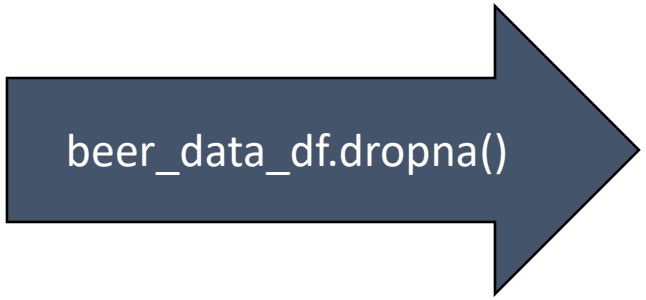
Data Profile

beer_ABV	• Measure of how much alcohol is contained in a given volume of an alcoholic beverage
beer_beerId	• Unique identifier of the beer
beer_brewerId	• Unique identifier of the brewery
beer_name	• Name of the beer
beer_style	• Style of the beer
review_appearance	• Rating indicating impact of visual appearance of the beer by the consumer on the scale of 0-5
review_palette	• Palate refers to mouthfeel (i.e. the way an item of food or drink feels in the mouth)
review_overall	• Overall rating of the beer by the consumer on the scale of 0-5
review_taste	• Rates taste of the beer by the consumer on the scale of 0-5
review_profileName	• Reviewer name
review_aroma	• Rates on scale of 0-5 and aroma of the beer often has the most varied sensory response
review_text	• Describes textual feedback by reviewer on that particular beer
review_time	• Indicates time of the review registered

EDA – Handling Null Data

Null Values per column in data set

beer_ABV	20280
beer_beerId	0
beer_brewerId	0
beer_name	0
beer_style	0
review_appearance	0
review_palette	0
review_overall	0
review_taste	0
review_profileName	115
review_aroma	0
review_text	119
review_time	0



No Null Values per column in data set

beer_ABV	0
beer_beerId	0
beer_brewerId	0
beer_name	0
beer_style	0
review_appearance	0
review_palette	0
review_overall	0
review_taste	0
review_profileName	0
review_aroma	0
review_text	0
review_time	0

Questions and Solution

Q1 Rank top 3 Breweries which produce the strongest beers?

- **Columns to consider**
  - beer_ABV
  - beer_brewerId
- **Logic**
  - step 1 - Group by beer_brewerId and compute average beer_ABV
  - step 2 - Create DataFrame on the resulted series
  - step 3 - Sort DataFrame by beer_abv_mean in descending order and get top 3 records

Q2 Which year did beers enjoy the highest ratings?

- **Columns to consider**
  - 'review_year' dervied from 'review_time'
  - 'avg review' dervied from 'review_appearance', 'review_palette','review_taste','review_aroma','review_overall'
- **Logic**
  - step 1 - Derive review_year from review_time
  - step 2 - Add a column say 'avg review' in existing dataframe indicating average rating per review with mean of ('review_appearance', 'review_palette','review_taste','review_aroma','review_overall')
  - step 3 - Group by review_year and compute average 'avg review' rating
  - step 4 - Create DataFrame on the resulted series
  - step 5 - Sort DataFrame by 'avg review' in descending order and get top 1 record
  - step 6 - Convert result set to string

Q3 Based on the user's ratings which factors are important among taste, aroma, appearance, and palette?

- **Columns to consider**
  - 'review_overall'
  - 'review_appearance'
  - 'review_aroma'
  - 'review_overall'
  - 'review_palette'
  - 'review_taste'

- **Logic Approach 1 : Corelation**
  - step 1 - Compute co-relation between 'review_overall' and ['review_palette','review_taste','review_aroma','review_appearance']
  - step 2 - Create DataFrame on the resulted series
  - step 3 - Sort DataFrame by 'review_overall' in descending order and get top 1 record
  - step 4 - Convert result set to string

- **Logic Approach 2 : Permutation Feature Importance**
  - step 1 – Split features in two variables
  - step 2 – X Dependent Variables ['review_palette','review_taste','review_aroma','review_appearance']
  - step 3 – y Independent Variable ('review_overall)
  - step 4 – Initiate RandomForestRegressor
  - step 5 – Fit the model
  - step 6 – Compute rf.feature_importances_

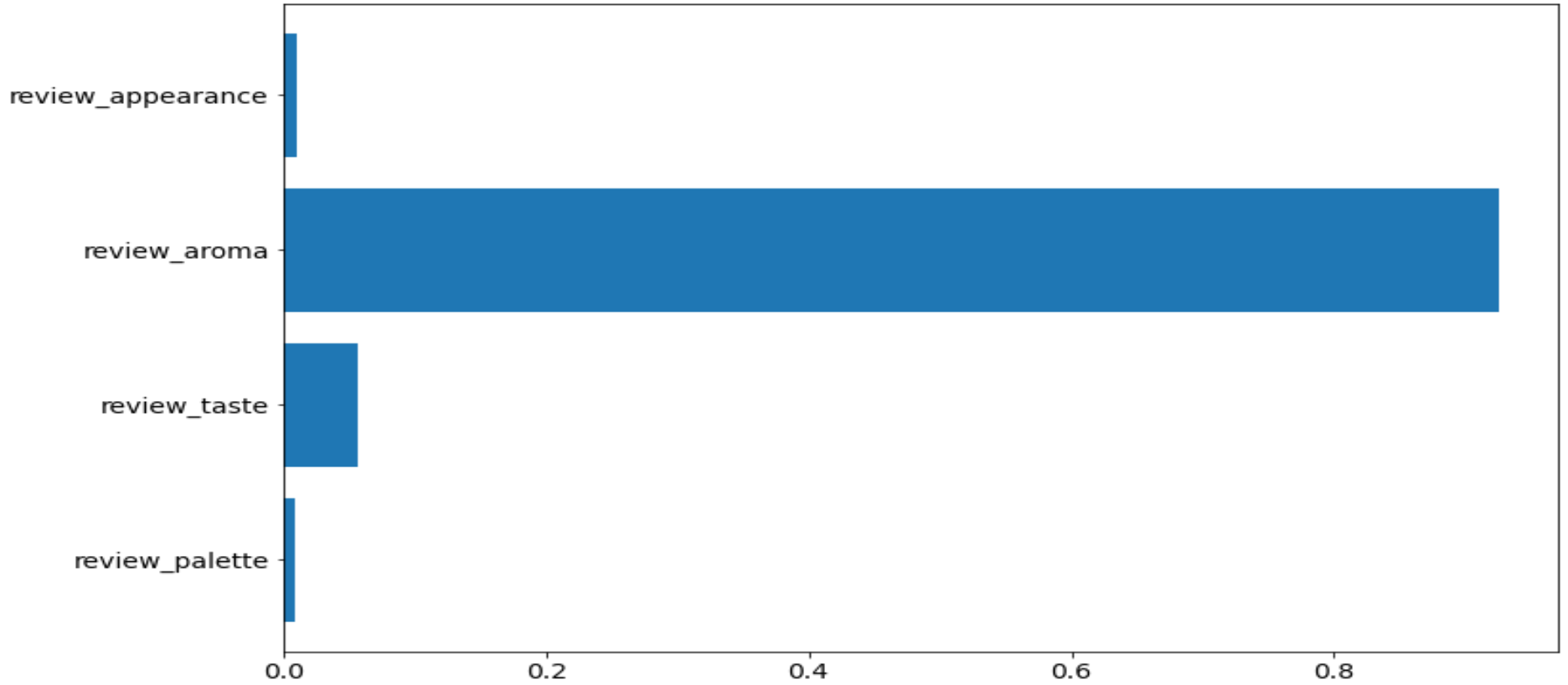
- **Permutation Feature Importance method over co-relation**
  - These feature importance's are based on the mean decrease in criterion, like gini impurity for random forests
  - Importance's are based on measuring the increase of the prediction error when you permute the feature's values.
  - So you compute the prediction error two times, before and after permutation of the feature.
  - The higher the difference between the prediction errors, the more important the feature.

Q3 Based on the user's ratings which factors are important among taste, aroma, appearance, and palette?

Co-relation

	review_overall	review_palette	review_taste	review_aroma	review_appearance
review_overall	1.000000	0.598048	0.689276	0.780310	0.483091
review_palette	0.598048	1.000000	0.600842	0.703428	0.544724
review_taste	0.689276	0.600842	1.000000	0.722737	0.551979
review_aroma	0.780310	0.703428	0.722737	1.000000	0.531204
review_appearance	0.483091	0.544724	0.551979	0.531204	1.000000

Permutation Feature Importance



Q4 If you were to recommend 3 beers to your friends based on this data which ones will you recommend?

- **Columns to consider**
  - beer_name
  - review_overall

- **Logic**
  - step 1 - Group by beer_name and compute average review_overall
  - step 2 - Create DataFrame on the resulted series
  - step 3 - Sort DataFrame by review_overall_mean in descending order and get top 3 records

Q5 Which Beer style seems to be the favorite based on reviews written by users?

- **Columns to consider**
  - beer_style
  - review_text
  - review_overall

- **Logic**
  - Filter records based on review_overall > 4
  - Cleansing text
    - Uniform casing - lower()
    - Stopwords
    - Normalization
    - Pos tagging
    - Compute polarity score of review_text
  - Computing polarity score using TextBlob Calculate the polarity score for each review
  - Group by the 'beer_style' and calculate mean of polarity score
  - Create DataFrame on the resulted series
  - Sort DataFrame by polarity score mean in descending order and get top 1 record

Textblob vs Vader

- Textblob sentiment analyzer returns two properties for a given input sentence:
  - Polarity is a float that lies between [-1,1], -1 indicates negative sentiment and +1 indicates positive sentiments.
  - Subjectivity is also a float that lies in the range of [0,1]. Subjective sentences generally refer to opinion, emotion, or judgment.

- VADER (Valence Aware Dictionary and sEntiment Reasoner)
  - It uses a list of lexical features (e.g. word) which are labeled as positive or negative according to their semantic orientation to calculate the text sentiment.
  - Vader sentiment returns the probability of a given input sentence to be positive, negative, and neutral

- For this data and requirement Textblob's polarity score was more suitable

Q6 How does written review compare to overall review score for the beer styles?

- **Columns to consider**
  - beer_style
  - polarity_score – Generated during previous question
  - review_overall
- **Logic**
  - step 1 - Group by beer_style and compute average of 'polarity_score' and 'review_overall'
  - step 2 - Create DataFrame on the resulted series
  - step 3 - Sort DataFrame by 'polarity_score' and 'review_overall' in descending order
  - step 4 - Analyse association of 'polarity_score' and 'review_overall'

Q7 How do find similar beer drinkers by using written reviews only?

- **Columns to consider**
  - review_text
  - review_profileName

- **Logic**
  - Considering review text only where the reviewer has reviewed at least 500 beers
  - Merge text review and group by each reviewer name
  - Perform cross join on the same data frame so that each review text can be compared by every other text in same data frame
  - Initialize TfidfVectorizer to convert a review text to vector
  - Normalize text using lemmatization
  - Remove punctuation
  - Compute cosine_similarity score between two cleansed texts
  - Sort records by cosine_similarity

Thank You!