Wine Recommender

August 19, 2020

1 Wine Recommender Based on Sommelier Reviews

This project will implement a recommender system based on sommelier wine reviews. The goal of this project will be to produce a recommender system for wines based on the similarity to a bottle of wine the user has tried. The similarity will be determined by reviews with common descriptive words. The recommender will also predict a rating based on the average of the ratings weighted by the Jaccard similarity of the descriptions.

[1]:
```
import string
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

The data was taken from Kaggle https://www.kaggle.com/zynicide/wine-reviews

The dataset has ~120K individual reviews using sommelier descriptions like "snappy", "earthy" and various, very specific, fruit or spice flavors.

[2]:
```
dataset = pd.read_csv('data/1442_8172_compressed_winemag-data-130k-v2.csv.zip')
```

[3]:
```
dataset.drop(dataset.columns[0],axis = 1,inplace=True) # Drop the index column␣
↪and use the pandas index
```

[4]:
```
dataset.head()
```

[4]:

	country	description	designation	points	price	province
0	Italy	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	NaN	Sicily & Sardinia
1	Portugal	This is ripe and fruity, a wine that is smooth...	Avidagos	87	15.0	Douro
2	US	Tart and snappy, the flavors of lime flesh and...	NaN	87	14.0	Oregon
3	US	Pineapple rind, lemon pith and orange blossom ...	Reserve Late Harvest	87	13.0	Michigan

```
 4  Vintner's Reserve Wild Child Block      87   65.0             Oregon

              region_1           region_2         taster_name  \
0                 Etna                NaN       Kerin O'Keefe
1                  NaN                NaN          Roger Voss
2    Willamette Valley  Willamette Valley        Paul Gregutt
3  Lake Michigan Shore                NaN  Alexander Peartree
4    Willamette Valley  Willamette Valley        Paul Gregutt

  taster_twitter_handle                                                title  \
0          @kerinokeefe                     Nicosia 2013 Vulkà Bianco  (Etna)
1            @vossroger      Quinta dos Avidagos 2011 Avidagos Red (Douro)
2           @paulgwine       Rainstorm 2013 Pinot Gris (Willamette Valley)
3                   NaN  St. Julian 2013 Reserve Late Harvest Riesling ...
4           @paulgwine   Sweet Cheeks 2012 Vintner's Reserve Wild Child...

          variety               winery
0     White Blend              Nicosia
1  Portuguese Red  Quinta dos Avidagos
2      Pinot Gris            Rainstorm
3        Riesling           St. Julian
4      Pinot Noir         Sweet Cheeks
```

[5]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129971 entries, 0 to 129970
Data columns (total 13 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   country                129908 non-null  object
 1   description            129971 non-null  object
 2   designation            92506 non-null   object
 3   points                 129971 non-null  int64
 4   price                  120975 non-null  float64
 5   province               129908 non-null  object
 6   region_1               108724 non-null  object
 7   region_2               50511 non-null   object
 8   taster_name            103727 non-null  object
 9   taster_twitter_handle  98758 non-null   object
 10  title                  129971 non-null  object
 11  variety                129970 non-null  object
 12  winery                 129971 non-null  object
dtypes: float64(1), int64(1), object(11)
memory usage: 12.9+ MB
```

[6]:
```
print("Unique Wines: ", len(dataset.title.value_counts()))
print("Top 20 wines with the most reviews:\n", dataset.title.value_counts().
 ↪head(20))
```

```
Unique Wines:  118840
Top 20 wines with the most reviews:
 Gloria Ferrer NV Sonoma Brut Sparkling (Sonoma County)                11
Korbel NV Brut Sparkling (California)                                  9
Segura Viudas NV Extra Dry Sparkling (Cava)                            8
Segura Viudas NV Aria Estate Extra Dry Sparkling (Cava)                7
Gloria Ferrer NV Blanc de Noirs Sparkling (Carneros)                   7
Ruinart NV Brut Rosé  (Champagne)                                      7
Korbel NV Sweet Rosé Sparkling (California)                            6
J Vineyards & Winery NV Brut Rosé Sparkling (Russian River Valley)     6
Boizel NV Brut Réserve  (Champagne)                                    6
Pierre Sparr NV Brut Réserve Sparkling (Crémant d'Alsace)              6
Jacquart NV Brut Mosaïque  (Champagne)                                 6
Mumm Napa NV Brut Prestige Sparkling (Napa Valley)                     6
Bailly-Lapierre NV Brut  (Crémant de Bourgogne)                        6
Korbel NV Blanc de Noirs Sparkling (California)                        5
Jean Laurent NV Blanc de Noirs Brut Pinot Noir (Champagne)             5
Mailly Grand Cru NV Délice Demi-Sec  (Champagne)                       5
G. H. Mumm NV Cordon Rouge Brut  (Champagne)                           5
Mailly Grand Cru NV Blanc de Noirs Brut Pinot Noir (Champagne)         5
Roederer Estate NV Brut Rosé Sparkling (Anderson Valley)               5
Henri Abele NV Brut  (Champagne)                                       5
Name: title, dtype: int64
```

[7]:
```
print("Unique Reviewers: ", len(dataset.taster_twitter_handle.value_counts()))
```

```
Unique Reviewers:  15
```

[8]:
```
dataset.taster_twitter_handle.isna().sum()
```

[8]: 31213

2 Finding Similarities

There are only 15 reviewers, and they are not necessarily reviewing the wines based on preference, so the user/item similarity approach from the class probably won't yield good results. Also, 31k reviews do not have a taster, so to avoid removing these from the dataset, I'll try looking at the description text, and calculate similarity based on the words in the description.

[9]:
```
dataset.loc[dataset['title'] == 'Gloria Ferrer NV Sonoma Brut Sparkling (Sonoma␣
 ↪County)'].head(3)
```

[9]:
```
      country                                        description  designation  \
3209       US  Creamy, lush and somewhat robust, this dry spa...  Sonoma Brut
4399       US  Made predominantly from Pinot Noir, this is an...  Sonoma Brut
27773      US  A wonderfully drinkable sparkling wine that ap...  Sonoma Brut

       points  price    province       region_1 region_2     taster_name  \
3209       90   22.0  California  Sonoma County   Sonoma  Virginie Boone
4399       88   22.0  California  Sonoma County   Sonoma  Virginie Boone
27773      90   20.0  California  Sonoma County   Sonoma             NaN

      taster_twitter_handle  \
3209                 @vboone
4399                 @vboone
27773                    NaN

                                                    title          variety  \
3209   Gloria Ferrer NV Sonoma Brut Sparkling (Sonoma...  Sparkling Blend
4399   Gloria Ferrer NV Sonoma Brut Sparkling (Sonoma...  Sparkling Blend
27773  Gloria Ferrer NV Sonoma Brut Sparkling (Sonoma...  Sparkling Blend

              winery
3209   Gloria Ferrer
4399   Gloria Ferrer
27773  Gloria Ferrer
```

There are multiple descriptions for each wine name. Let's create a reduced dataset that has only the title and description, and then merge the descriptions for each wine and clean up and tokenize the strings and finally, remove common words such as 'i' and 'there'.

[10]:
```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
#print(stop_words)
```

[11]:
```
import time
start_time = time.time()
# initialize the lists that will store the data
names = []
descriptions = []
pointss = []
i = 0
for name in list(dataset['title'].unique()):
    catd = ' '.join(dataset.loc[dataset['title'] == name].description)
    cleaned = catd.translate(str.maketrans('', '', string.punctuation)).lower().
 ↪split(sep=' ')
    tokenized = set([w for w in cleaned if not w in stop_words])
    names.append(name)
    descriptions.append(tokenized)
```

```
    pointss.append(int(dataset.loc[dataset['title'] == name].points.values[0]))
    if not i % 10000:
        print("element {} --- {:d} minute ---".format(i, int((time.time() -␣
 ↪start_time) // 60)))
    i +=1
#     if i > 15000: # quit early for debugging
#         break
# create the empty dataframe for the merged descriptions
merged_desc = pd.DataFrame()
merged_desc['title'] = names
merged_desc['description'] = descriptions
merged_desc['points'] = pointss
```

```
element 0 --- 0 minute ---
element 10000 --- 3 minute ---
element 20000 --- 6 minute ---
element 30000 --- 10 minute ---
element 40000 --- 13 minute ---
element 50000 --- 17 minute ---
element 60000 --- 20 minute ---
element 70000 --- 23 minute ---
element 80000 --- 27 minute ---
element 90000 --- 30 minute ---
element 100000 --- 33 minute ---
element 110000 --- 37 minute ---
```

[12]:
```
merged_desc.tail(5)
```

[12]:

	title	description	points
118835	Dr. H. Thanisch (Erben Müller-Burggraef) 2013 ...	{featherlight, tangerine, acidity, honey, hone...	90
118836	Citation 2004 Pinot Noir (Oregon)	{coconut, means, secondary, given, gracefully,...	90
118837	Domaine Gresser 2013 Kritt Gewurztraminer (Als...	{crisp, subdued, spice, serious, favor, ripe, ...	90
118838	Domaine Marcel Deiss 2012 Pinot Gris (Alsace)	{acidity, crisp, pinot, powerful, spice, gris,...	90
118839	Domaine Schoffit 2012 Lieu-dit Harth Cuvée Car...	{dominate, drink, rounded, profile, feel, powe...	90

That took a very long time, but it looks like I have what I need now.

The Jaccard distance function will take two sets of words (cleaned, concatenated descriptions) and calculate the Intersection over Union for words in the two description sets.

[13]:
```
def Jaccard(s1,s2):
    num = len(s1.intersection(s2))
    den = len(s1.union(s2))
    return num/den
```

[14]:
```
def mostSimilar(name, n):
    similarities = []
    # I have to loop over the whole dataset and calculate my Jaccard similarities
    ws1 = merged_desc.loc[merged_desc['title'] == name].description.values[0]
    for idx, dat in merged_desc.iterrows():
        name2 = dat.title
        if name2 == name:
            continue
        ws2 = dat.description
        sim = Jaccard(ws1,ws2)
        similarities.append((sim,name2))
    similarities.sort(reverse=True)
    return similarities[:n]
```

[15]:
```
test1 = dataset.iloc[1908]
print('Suggested similar wines to: ', test1.title, '\n')
mostSimilar(test1.title,10)
```

```
Suggested similar wines to:  Nugan Family Estates 2004 Cabernet Sauvignon (South
Eastern Australia)
```

[15]:
```
[(0.22857142857142856, 'Tahbilk 2006 Shiraz (Nagambie Lakes)'),
 (0.22857142857142856, 'Château la Genestière 2014 Red (Lirac)'),
 (0.2222222222222222,
  'Mt. Difficulty 2013 Bannockburn Long Gully Single Vineyard Pinot Noir
(Central Otago)'),
 (0.2222222222222222, 'Celestial Bay 2004 Shiraz (Margaret River)'),
 (0.21875, 'Marcolino Sebo 2013 Visconde de Borba Red (Alentejo)'),
 (0.21212121212121213,
  'Kestrel 2013 Falcon Series Estate Sangiovese (Yakima Valley)'),
 (0.21052631578947367,
  'Wakefield 2010 St. Andrews Single Vineyard Release Cabernet Sauvignon (Clare
Valley)'),
 (0.20588235294117646, "Maimai 2011 Syrah (Hawke's Bay)"),
 (0.20588235294117646, 'Lavau 2012 White (Châteauneuf-du-Pape)'),
 (0.20588235294117646,
  "Kendall-Jackson 2015 Vintner's Reserve Pinot Gris (California)")]
```

[16]:
```
test2 = dataset.iloc[3284]
print('Suggested similar wines to: ', test2.title, '\n')
```

```
mostSimilar(test2.title,10)
```

```
Suggested similar wines to:  Melipal 2006 Reserve Malbec (Mendoza)
```

[16]:
```
[(0.14, 'Arboleda 2006 Shiraz (Aconcagua Valley)'),
 (0.13559322033898305, 'Baron De Ley 2001 7 Viñas Reserva  (Rioja)'),
 (0.13333333333333333, 'Château Darzac 2011 Heritage  (Bordeaux Supérieur)'),
 (0.12962962962962962,
  'TerraNoble 2007 Gran Reserva Cabernet Sauvignon (Colchagua Valley)'),
 (0.1276595744680851, 'Zumaya 2010  Ribera del Duero'),
 (0.1276595744680851,
  'Viña Alicia 2008 Paso de Piedra Cabernet Sauvignon (Luján de Cuyo)'),
 (0.125, 'Coelho 2010 Atração Pinot Noir (Willamette Valley)'),
 (0.12,
  'Real Sitio de Ventosilla 2004 RSV 1601 El Duque de Lerma  (Ribera del
Duero)'),
 (0.11904761904761904,
  'Alta Cima 2013 Speical Edition Reserva Syrah (Lontué Valley)'),
 (0.11764705882352941,
  'Vistamar 2010 Sepia Reserva Cabernet Sauvignon (Maipo Valley)')]
```

[ ]:
```
test3 = dataset.iloc[89021]
print('Suggested similar wines to: ', test3.title, '\n')
mostSimilar(test3.title,10)
```

I tested several different wines for suggestions. It seems to be doing a very good job finding similar wines. When a varietal or type is entered, the top suggestions are always for matching wines. The dataset was only sommelier reviews and was old. I think this recommender system could work well for an online wine store, and could be extended with more recent data, and with customer reviews as well as the sommelier reviews.

The recommender system is quite slow, right now. I think this could be improved by not considering the whole dataset, but instead breaking it up into categories and only considering similar varietals, countries, regions, etc. in the similarity.

3 Rating prediction using collaborative filtering

First, calculate the average rating

[17]:
```
avg_rating = merged_desc['points'].mean(axis=0)
print(avg_rating)
```

```
88.4437478963312
```

[18]:
```
def predictRating(name):
    ratings = []
    similarities = []
```

```
    ws1 = merged_desc.loc[merged_desc['title'] == name].description.values[0]
    for idx, dat in merged_desc.iterrows():
        name2 = dat.title
        if name2 == name:
            continue
        ws2 = dat.description
        ratings.append(dat.points)
        sim = Jaccard(ws1,ws2)
        similarities.append(sim)
    if (sum(similarities) > 0):
        weightedRatings = [(x*y) for x,y in zip(ratings,similarities)]
        return sum(weightedRatings) / sum(similarities)
    else:
        # There are no similar wines (unlikely, given the vast number of words␣
 ↪included)
        return avg_rating
```

3.1 Evaluate Performance

[19]:
```
test2 = merged_desc.iloc[3569]
print(test2)
name = test2['title']
predictRating(name)
```

```
title          Authentique 2014 Keeler Estate Vineyard Pinot ...
description    {wholecluster, gravelly, slightly, tannins, ed...
points                                                        89
Name: 3569, dtype: object
```

[19]:
```
88.47362716544629
```

[20]:
```
def MSE(predictions, labels):
    differences = [(x-y)**2 for x,y in zip(predictions,labels)]
    return sum(differences) / len(differences)
```

[26]:
```
always_predict_mean = [avg_rating]*1000 #len(merged_desc) # only look at a␣
 ↪subset to save time
cf_predictions = []
i = 0
for idx, dat in merged_desc.iterrows():
    cf_predictions.append(predictRating(dat['title']))
    i += 1
    if i >= 1000:
        break
truth = list(merged_desc['points'])[:1000]
```

[27]:
```
print('Always predict mean MSE: ', MSE(always_predict_mean, truth))
print('Colab Filtered Predictions MSE: ', MSE(cf_predictions, truth))
```

```
Always predict mean MSE:  6.910502148376197
Colab Filtered Predictions MSE:  6.33491842042439
```

So predicting the rating using the Jaccard similarity of words in the review text significantly reduces the MSE from only using the mean rating as the prediction. I believe this could be improved by further removal of irrelevant words, or perhaps applying some sentiment analysis to the review text.