REACHING OUT TO OLD FRIENDS: HOW DO WE RECOMMEND WINES TO OUR CUSTOMERS?

A study of X-Wines and X-Wine customers

Capstone 3 Final Report
Jessica Williams

OBJECTIVE

X-wines has been running into trouble getting retaining their customer base. The percentage rate for repeat customers is at the lowest it’s been in the last five years. The company is trying to generate ideas to keep current customers making more purchases. X wines would like to enhance their customer experiences by recommending wines to customers that they might be interested in. They are looking to build a recommender system based on the properties of their wine and the customer ratings.

DATA

To investigate this question we I will use an X-wines dataset consisting of the following two dataframes:

- XWines_Full_100K_wines (csv)-This file contains a list of 100,000 wines and some of their attributes. The attributes include Type, Elaborate, Grapes, Harmonize, ABV, Body, Acidity, Region, Winery and Vintages.
- XWines_Full_21M_ratings(csv file)- This file contains a set of 21 million user ratings of wines on a scale of 1-5.

DATA WRANGLING

- **General cleaning and examination**

This was a fairly clean dataset so most of my data wrangling time was spent examining the column values. Here are a few of the steps that I took:

- Dropped the website column from the wine list dataframe as I know that won’t be useful for my analysis.

-Examined the value counts of most of the attribute columns in the wine list dataframe to view the available responses for each variable.

-Checked the wine list dataframe for null values and found none.

-Made sure no specific wine ID was repeated in the wine list dataframe.

-Found out the value range for the ratings in the wine ratings dataframe is1-5.

-Counted the number of unique raters in the wine ratings dataframe.

-Examined the frequency of the different individual raters.

-Created a mean rating column for the wine ratings dataframe that represented the mean of the ratings given for each unique wine.

EXPLORATORY DATA ANALYSIS (EDA)

- **Creating more features**

To aid in more effective analysis, I added a few extra variable columns to the data. In order to take a look at how many times our raters rated different wines, I added a column called rater count. This column summed the number of rating rows for each unique rater. The rater count column displayed a range of 5 to 2986 and confirmed that multiple ratings were available for all of our users.

I also added a column for the mean rating of each rater. From this column I was able to see that the higher frequency raters have lower rating means as would be expected with having more data to summarize over.

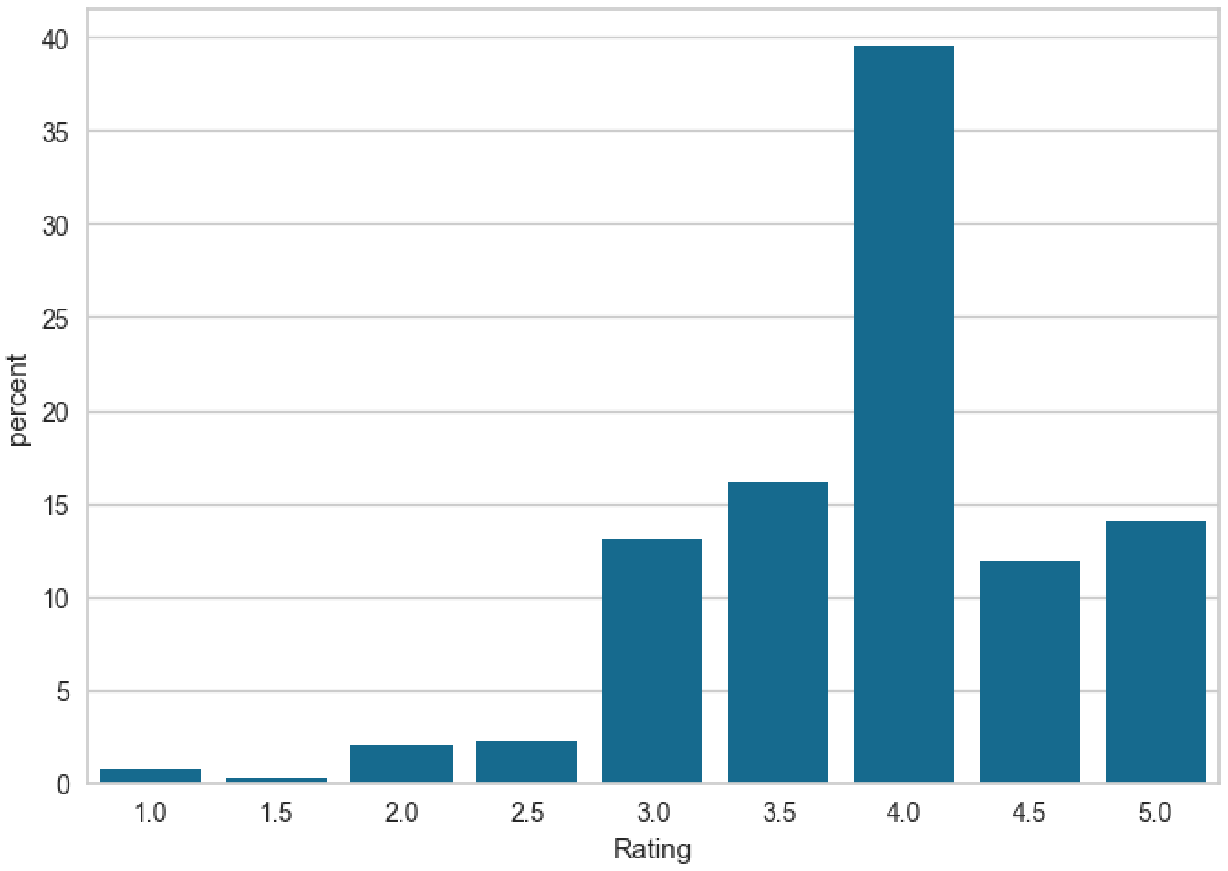
Since the dataset contained a lot of categorical variables, I decided to create dummy variables for a few of the categories that seemed most useful. I created dummy variables for the ‘Type’, ‘Elaborate’, ‘Body’, and ‘Acidity’ Columns.

- **Merging effectively**

Since the dataset existed in two different dataframes(wine_list and wine_ratings) I needed to merge the two dataframes to combine the information. I mergered the wine_list dataframe to the wine_ratings dataframe on the WineID column. The columns I pulled from the wine _list dataframe were the 'WineID', 'WineName', 'Type', 'Elaborate', 'Grapes', 'Harmonize', 'ABV', 'Body', 'Acidity', 'Country', 'RegionID', 'RegionName', 'WineryID', 'WineryName', and the 'Vintages' columns. To make this a simpler process I merged the dataframes before adding the dummy variables to the final dataframe.

- **Distributions**

First I examined the distribution of the ratings.



From this plot I noticed a few things

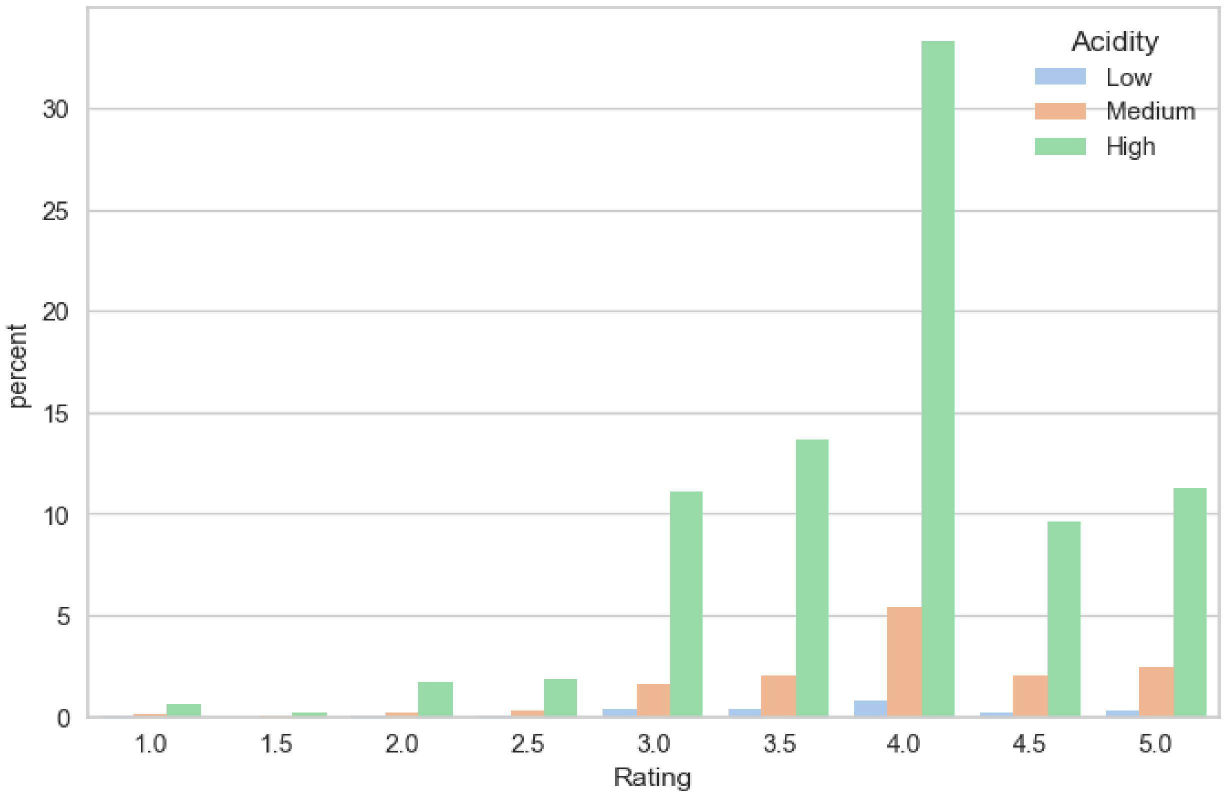
-4.0 is the most frequent rating with 40% of the distribution.

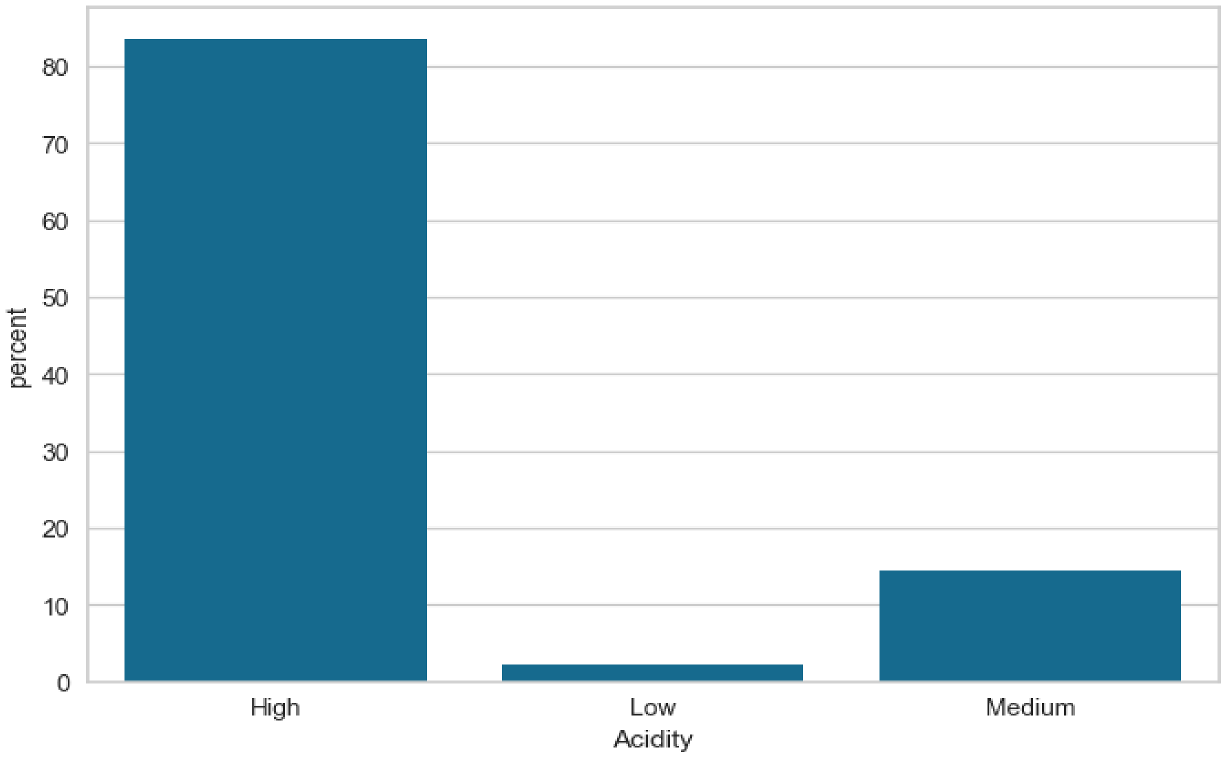
-Most of the ratings for the wines are above 3.0.

-the users can give .5 ratings as a part of the value ranges (ex. 1.5, 2.5)

-there is a significant count for all of the possible ratings.

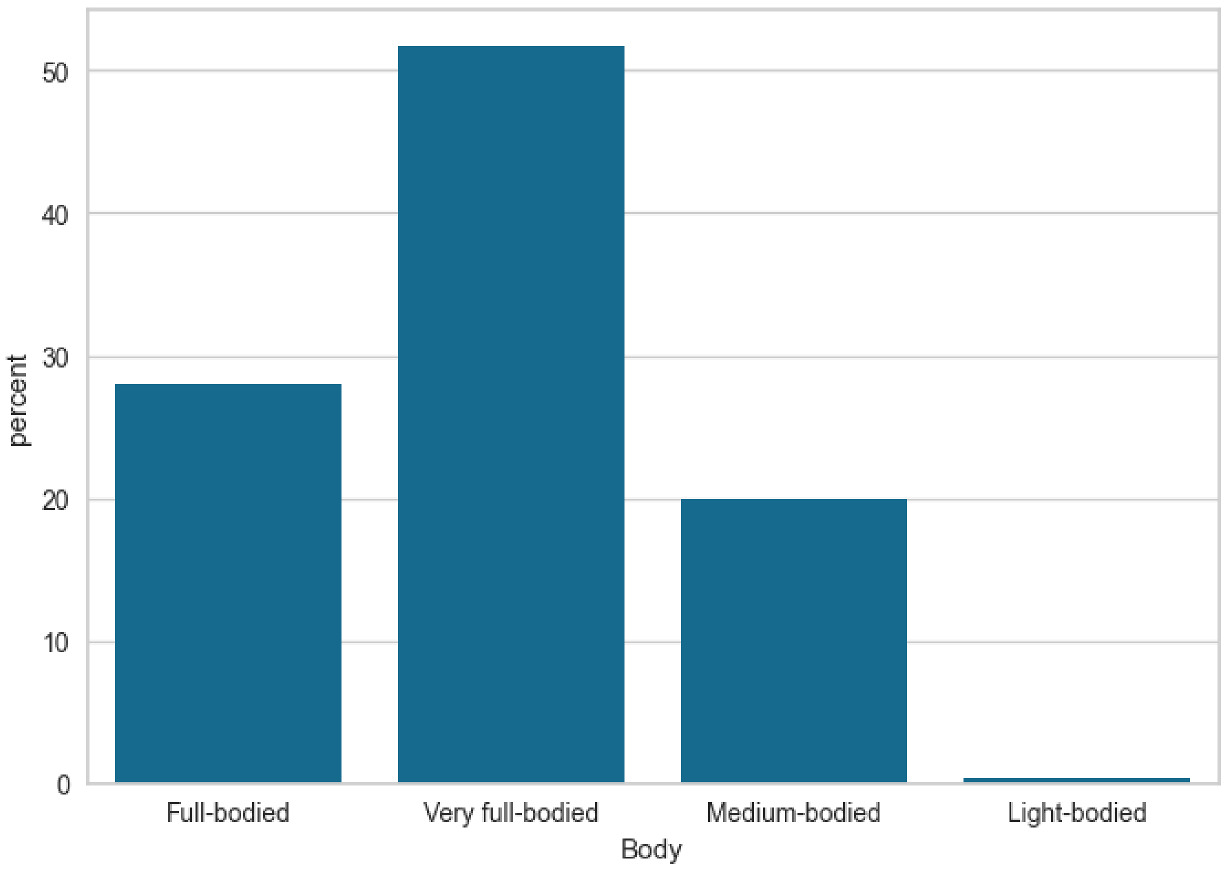
I also took a look at the rating distribution for some of the wine attribute variables starting with the ‘Acidity’ variable.

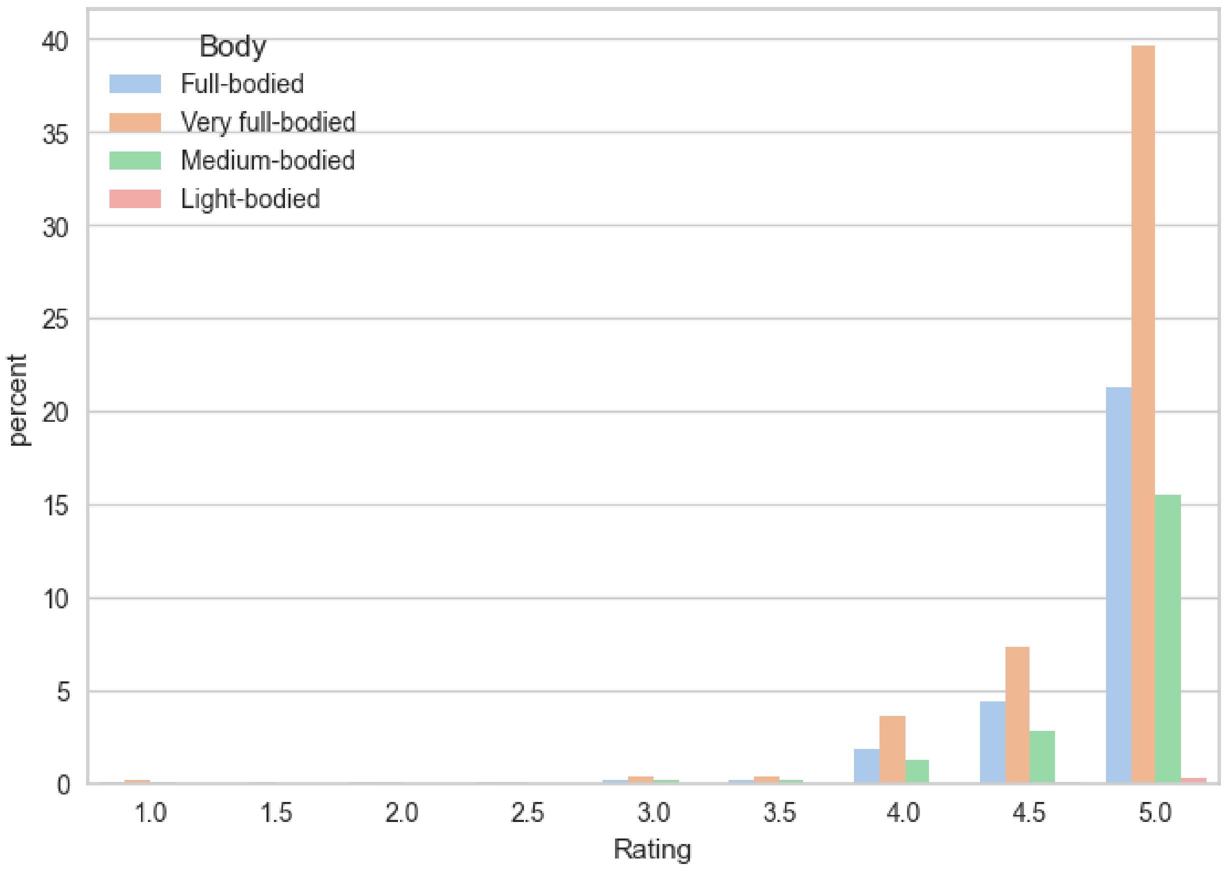

Acidity
Low
Medium
High
percent
0
5
10
15
20
25
30
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Rating


percent
0
10
20
30
40
50
60
70
80
High
Low
Medium
Acidity

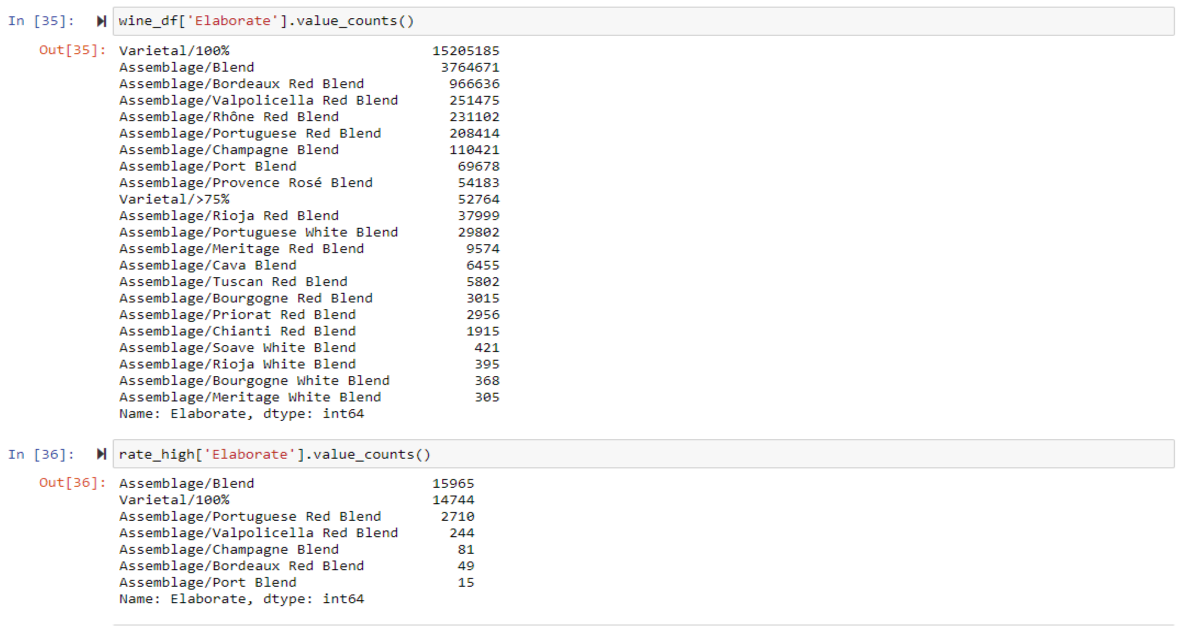
'High' acidity wines seemed to have a large representation in all of the rating values but it also accounted for 80% of the wine distribution. A similar pattern emerged with the distribution of the 'Body' variable. 'Full bodied' wines had the largest representation on the high ratings but this category accounted for almost 50% of the wine distribution.

-Since looking at this information over our entire dataset wasn't quite as useful as I would like it to be, I tried to examining the wines with the highest average rating. I started with a subset of rating means above 4.8. Within this subset the distribution for the 'Acidity' attribute remained the same. However, the distribution for the 'Body' variable did change.





The ‘very full-bodied’ wines now represents the largest percentage of rating values of 5.0 despite ranking third in the percentage of wines in this data subset. I also made this comparison with the ‘Elaborate’ variable.

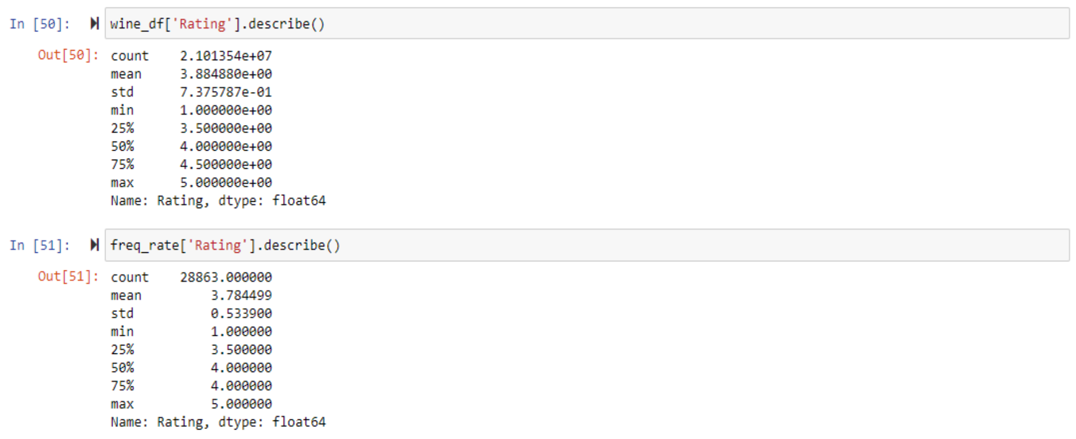

```
In [35]:  wine_df['Elaborate'].value_counts()
Out[35]:  Varietal/100%                       15205185
          Assemblage/Blend                     3764671
          Assemblage/Bordeaux Red Blend         966636
          Assemblage/Valpolicella Red Blend     251475
          Assemblage/Rhône Red Blend            231102
          Assemblage/Portuguese Red Blend       208414
          Assemblage/Champagne Blend            110421
          Assemblage/Port Blend                  69678
          Assemblage/Provence Rosé Blend         54183
          Varietal/>75%                          52764
          Assemblage/Rioja Red Blend             37999
          Assemblage/Portuguese White Blend      29802
          Assemblage/Meritage Red Blend           9574
          Assemblage/Cava Blend                   6455
          Assemblage/Tuscan Red Blend             5802
          Assemblage/Bourgogne Red Blend          3015
          Assemblage/Priorat Red Blend            2956
          Assemblage/Chianti Red Blend            1915
          Assemblage/Soave White Blend             421
          Assemblage/Rioja White Blend             395
          Assemblage/Bourgogne White Blend         368
          Assemblage/Meritage White Blend          305
          Name: Elaborate, dtype: int64

In [36]:  rate_high['Elaborate'].value_counts()
Out[36]:  Assemblage/Blend                    15965
          Varietal/100%                       14744
          Assemblage/Portuguese Red Blend      2710
          Assemblage/Valpolicella Red Blend     244
          Assemblage/Champagne Blend             81
          Assemblage/Bordeaux Red Blend          49
          Assemblage/Port Blend                  15
          Name: Elaborate, dtype: int64
```

For values in the ‘Elaborate’ variable, the top represented categories are ‘Assemblage/Blend’ and ‘Varietal/100%’. Although these two categories have the highest representation in the dataset, the highest ratings subset of the data contains the largest percentage of the Assemblage/Blend where the overall data set contains the highest portion of Varietal/100%.

I decided to stick to examining the highest rated subset of the data and move onto the representation of specific wineries in this subset. ‘Vega Sicilia’ sits at the top of the list with 8670 out of 33808 entries. I decided to narrow or top subset further to see if the top represented wineries would change. When examining rating means above 4.9, it looks like our top 5 wineries are now completely different than they were for rating means above 4.8. Krug sits at the top of the list with 157 entries out of 663.

Next I took a look at how the rating frequency of a user affects the data. I examined a subset of data with only users who rated a wine more than 2000 times.


```
In [50]:  wine_df['Rating'].describe()
Out[50]: count    2.101354e+07
         mean     3.884880e+00
         std      7.375787e-01
         min      1.000000e+00
         25%      3.500000e+00
         50%      4.000000e+00
         75%      4.500000e+00
         max      5.000000e+00
         Name: Rating, dtype: float64

In [51]:  freq_rate['Rating'].describe()
Out[51]: count    28863.000000
         mean         3.784499
         std          0.533900
         min          1.000000
         25%          3.500000
         50%          4.000000
         75%          4.000000
         max          5.000000
         Name: Rating, dtype: float64
```

I noticed a few key things from this comparison:

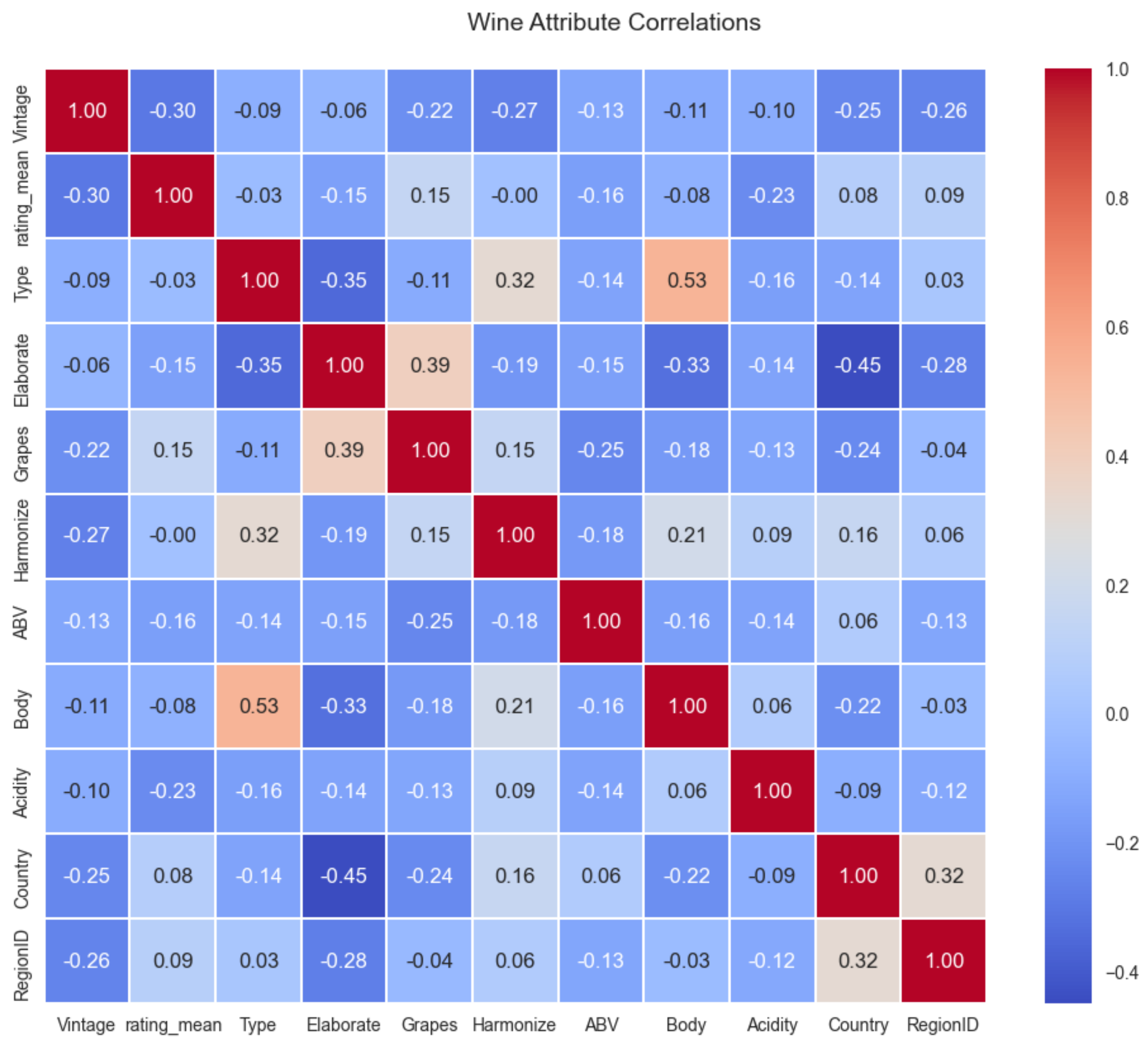
-The most frequently included wineries have changed again. ‘M. Chapoutier’ is most represented.

- It looks like the frequent raters rate a little more critically as the 50% and 75% quartiles are lower than the total dataset.

-Both the ‘Type’ and ‘Elaborate’ variable still have the same category with the highest representation (‘Red’, ‘Varietal/100%’)

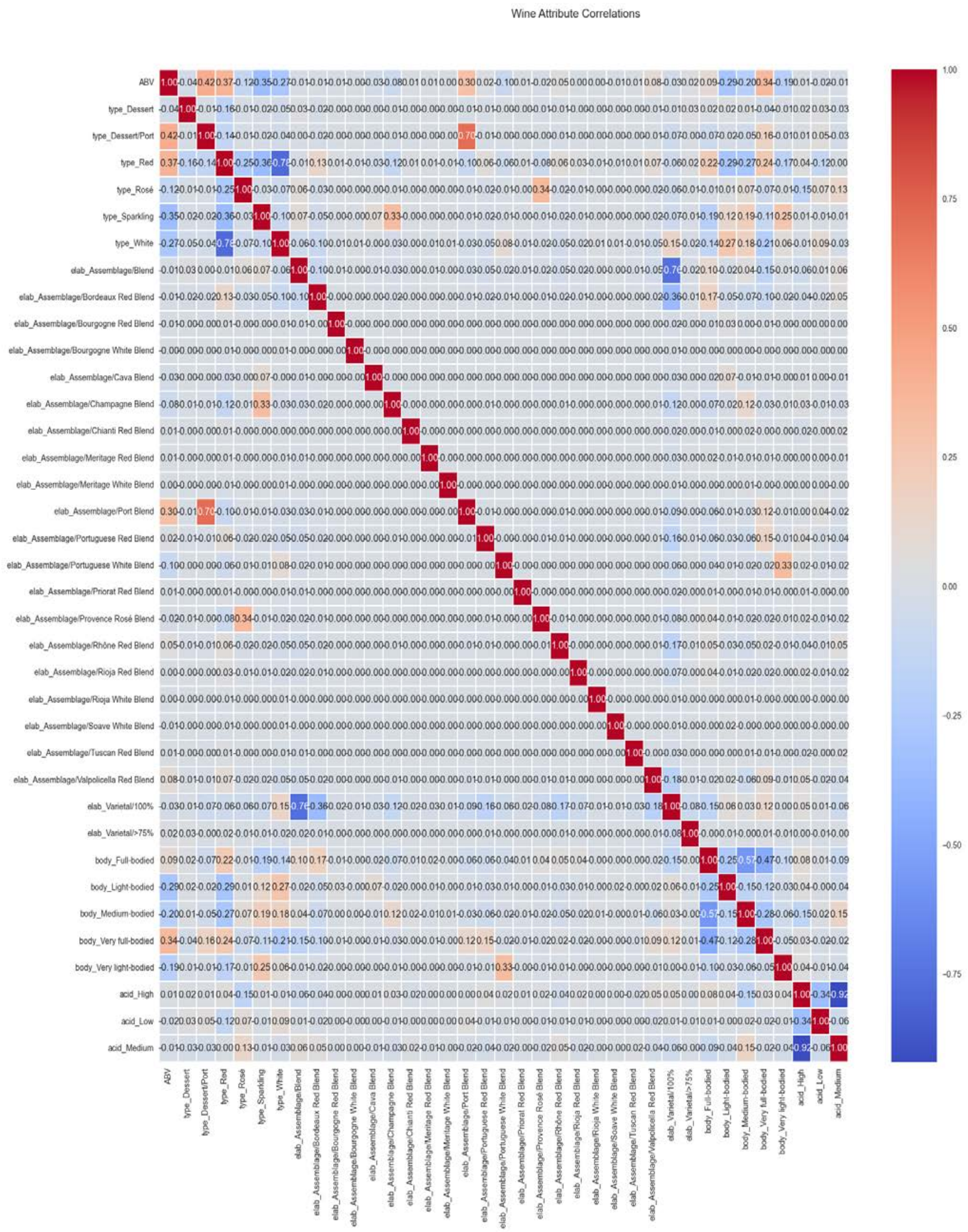
- **Correlations?**

First I looked for correlations between the wine attributes.



We don’t have any strong correlations here but we have some moderate ones. We can see the highest correlation is between ‘Body’ and ‘Type’ at 0.53. There are also a lot of negative correlations here which could mean that some of the attributes indicate the absence of others. For instance, a wine with more ‘ABV’ may have lower ‘Acidity’. I also examined correlations between the wine attributes and the rating mean of the wines and found no significant correlations.

Lastly I examined some of the correlations within the dummy variables I created.



As would be expected we have a few variables that have high negative correlations because they are in the same category. For example, low, medium and high acidity are all negatively correlated because they are all different levels of the same scale.

- **EDA Findings**

After exploring this data further here are my key takeaways:

1. It is hard to pinpoint specific features of the wines that correlate to higher or lower ratings because of the distribution of the wines with different features. Most of the wine attributes have a disproportionate representation of some values over others. For example, our wine list contains 56,162 red wines with the closest second being 29,196 red wines. All categories after that are 8000 or less. This makes it hard to discover patterns for large representation in high ratings outside of just being a large portion of the dataset overall.

2. There are a good amount of negative correlation between the variables because the different variables are what make each wine distinct.

3. Most of our data is categorical in some way and need a dummy variable to be created for predictive purposes.

4. We have a mixture of wine data and user rating data that will serve as a good base for creating an accurate recommender system.

MODELING

- **Model choice**

Since our recommender system will be based on user ratings of wine, let’s start by parsing out the user id, wine id and rating. From there I created the surprise data set to cross-validate our models on as well as create an anti-set to run predictions on later. The models I tested on the data were as follows:

- Normal Predictor model

-SVD Model

- KNNWithMeans

- KNNBaseline

I started with a Normal Predictor model to examine a baseline RMSE for our dataset. As expected our mean RMSE (0.906) is fairly high for a rating scale of 1-5. The SVD model gives us a much better mean RMSE of 0.488. - The KNNWithMeans model has a higher mean RMSE than the previous model. The test time is also much longer than the both of the previous models. The KNNBaseline model seems to have the best mean RMSE at 0.486. It also has the longest mean fit and test time of all of the models.

From the results outputted from testing different models on the data, I moved forward with fine tuning and applying both the KNNBaseline model as well as the SVD model. Even though the KNNBaseline model technically has the best mean RMSE, The SVD model has an RMSE that is very close and also would save a lot of test and fit time. I think it would be worth comparing the predictions for both models to find out if we can save time in the long run.

-First let's run a grid search for the best hyperparameters for our KNNBaseline Model.

- **Hyperparameters**

First I ran a grid search for the best hyperparameters for our KNNBaseline Model. The best parameters for this model are as follows:

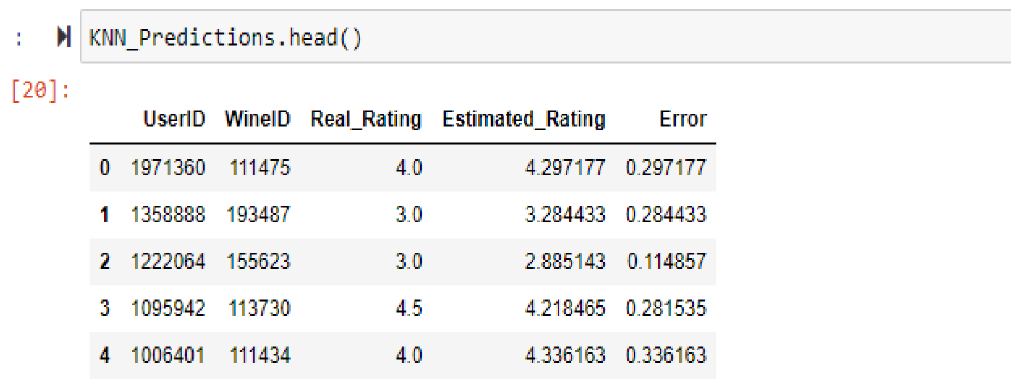
{'n_epochs': 5, 'lr_all': 0.002, 'reg_all': 0.4}

Next I ran the parameter grid search for the SVD model as well.The best parameters for the SVD model are as follows:

{'n_epochs': 15, 'lr_all': 0.005, 'reg_all': 0.4}

- **Results**

To apply the model and get predictions, I created a dataframe that will display the userID, wineID, the actual rating and the predicted rating from our model. I also included an error column that displays the difference between the actual rating and the predicted rating.

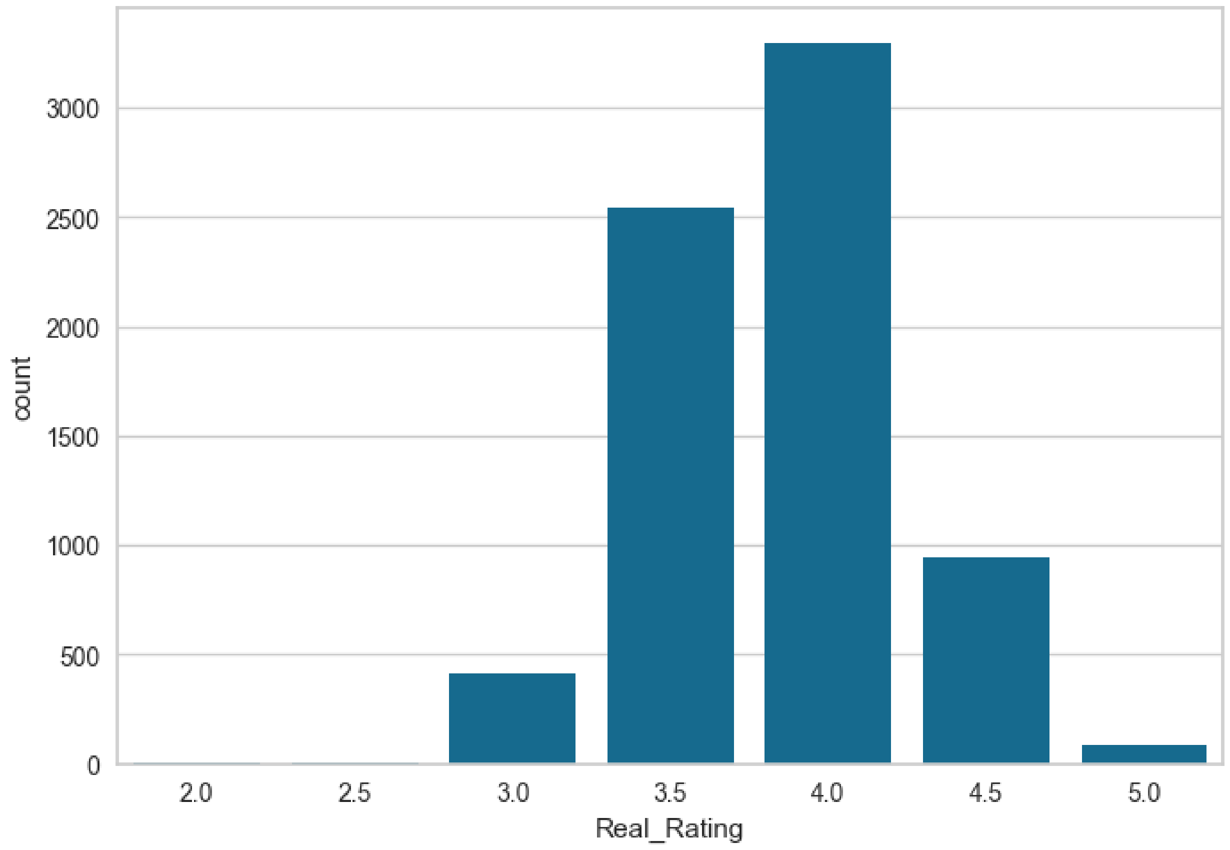


```
KNN_Predictions.head()
```

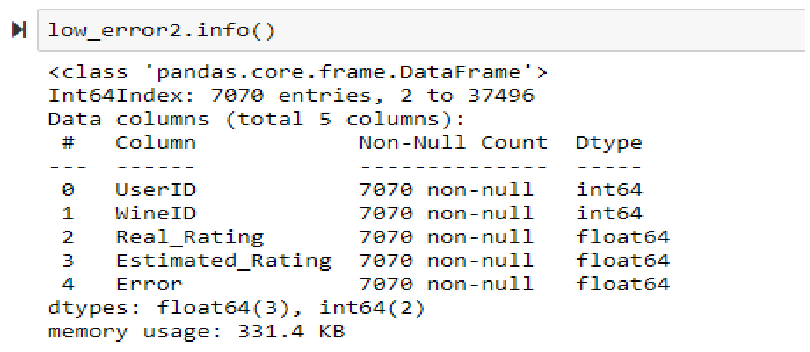
[20]:

	UserID	WineID	Real_Rating	Estimated_Rating	Error
0	1971360	111475	4.0	4.297177	0.297177
1	1358888	193487	3.0	3.284433	0.284433
2	1222064	155623	3.0	2.885143	0.114857
3	1095942	113730	4.5	4.218465	0.281535
4	1006401	111434	4.0	4.336163	0.336163

I stared with examining how many estimated ratings have a low error. The KNN Baseline Model produced 7278 estimated ratings out of 150,000 with errors less than 0.1. Next I examined the distribution of the ratings in the low error subset.



This model is best at predicting ratings of 4.0. It is also worth noting that this rating distribution is similar to the total data set. I followed the same steps using the SVD model as well.

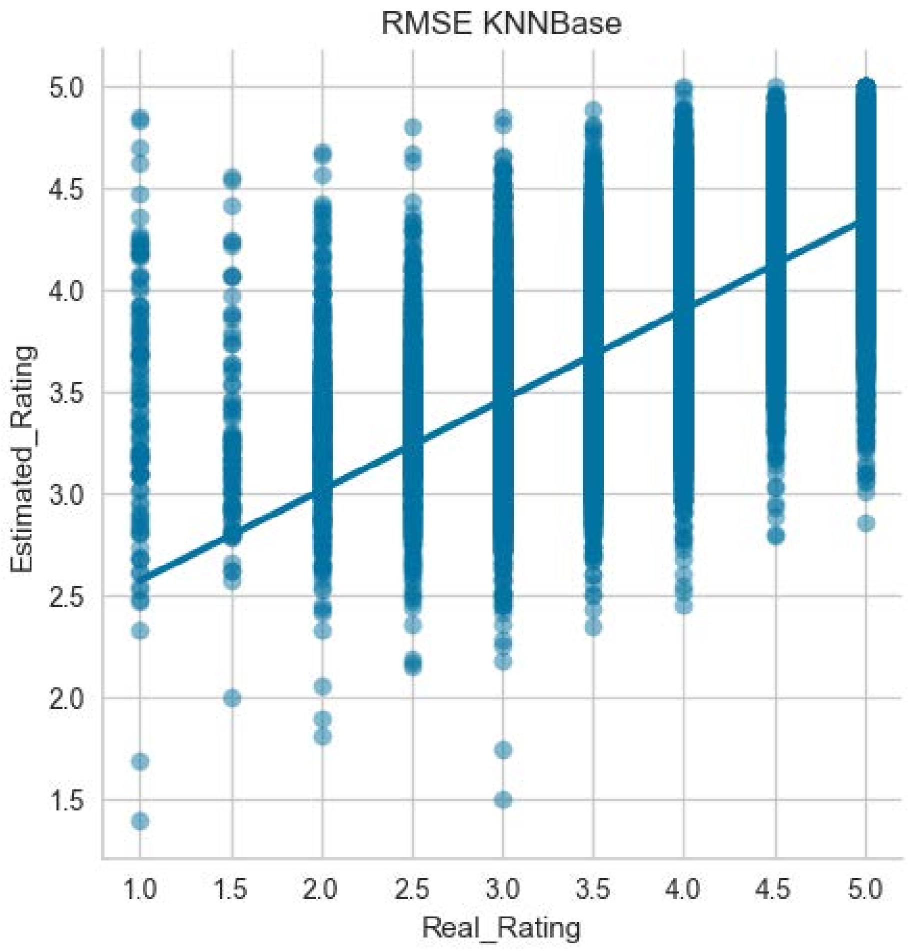


```
low_error2.info()
```

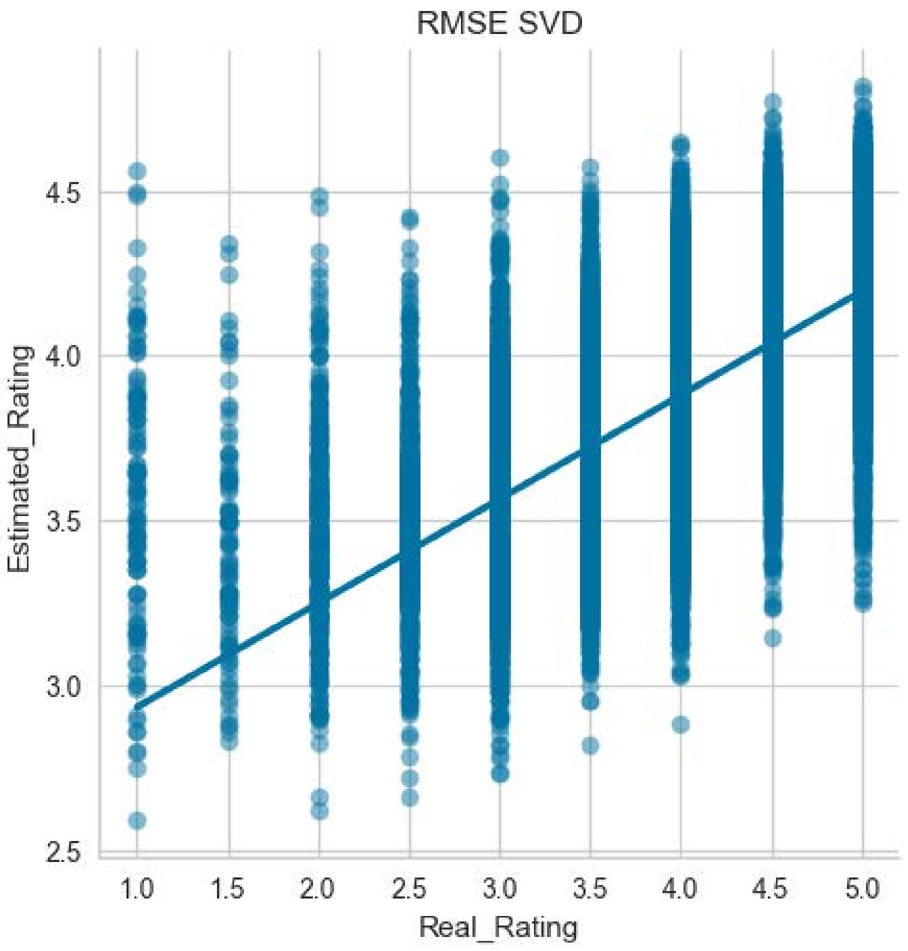
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7070 entries, 2 to 37496
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   UserID            7070 non-null   int64  
 1   WineID            7070 non-null   int64  
 2   Real_Rating       7070 non-null   float64
 3   Estimated_Rating  7070 non-null   float64
 4   Error             7070 non-null   float64
dtypes: float64(3), int64(2)
memory usage: 331.4 KB
```

The number of predictions in this data subset is 7070. This is similar to but less than the number of low error predictions from the KNN baseline model. The rating distribution for this model is also very similar to the KNNBaseline model with the top represented rating being 4.0. Finally compared the RMSE for the different models on our traintest set.

```
RMSE:0.48
```



```
RMSE:0.50
```



The RMSE for the KNN baseline model is pretty much the same as our initial cross validation mean. The SVD model is a bit higher than our initial cross validation mean.

CONCLUSIONS

After using each model to create predictions it looks like the KNNBaseline model produces better results than the SVD model. The KNN Baseline model has a better RMSE and has more estimated ratings with low errors than the SVD model. It seems that the KNNBaseline would be the better model to use to increase the chance of predicting ratings more accurately. Considering that the difference in outcomes between the two models is small and the time difference for applying the models is fairly large, it may be worth it to our stake holders to use the SVD model. A choice would have to be made here on whether accuracy or resource usage is the most important factor.