Predicting the Popularity of Different Beers

Eric Folsom, Delaney Green, Cameron Bayer, Eva Halsne

6/6/2023

Team Name: Stout-asticians

Dataset link: https://www.kaggle.com/datasets/rdoume/beerreviews

Research Goal:

The goal of this project is to predict the future popularity of beers given certain factors. We will be analyzing a dataset to determine which keywords, type, brand, aroma, style, appearance, taste and so forth affect what beers people prefer. Using this will we create predictions on beer popularity.

Abstract:

Beer and modern statistics share a surprising historical link. If it were not for William S. Gosset, a brewer at Guiness at the turn of the 20th century, we would not have Student's t-test, a hypothesis testing method that we use to this day. As statisticians from Washington state, which grows approximately three quarters of all the hops in the US, we have a natural interest in the creation and quality of beer. In this project, we use a dataset of ~1.5 million reviews from BeerAdvocate, one of the largest and oldest online resources for beer reviews, discussions, and events. By using a combination of classification and regression methods, our project takes a given beer and personal review to predict and recommend future beers to the reviewer.

Background:

Beer is one of the most popular alcoholic beverages in the world, with a global market worth billions of dollars. Understanding what factors contribute to the popularity of certain beers can be crucial for breweries and other businesses in the industry. With the rise of craft beer culture and the increasing number of beer options available, predicting what beers will be popular in the future can be a valuable tool for businesses looking to stay ahead of trends and improve their product offerings.

Using a dataset of beer reviews from BeerAdvocates, our goal was to create our own beer recommendation model. In the final product, the user would be able to input the name of a beer that is within the dataset and receive a list of recommendations of similar beers based on our model. This model takes factors such as brand, aroma, style, appearance, and taste into account to make a list of recommendations; by default, the model gives five recommendations, though more or less may be specified. While treating our dataset as continuous we will use a mixture of K-means classification, Lasso regression, and item based collaborative filtering methods to develop models that can predict future popularity based on these attributes.

Methods:

The dataset used for this project is a collection of user reviews and ratings for a myriad of beers. The dataset includes various attributes of the beers such as style, aroma, taste, appearance, and overall rating. We will be using this data to predict the popularity of beers based on these attributes. The original dataset contains over 1.5 million reviews in total. However, working with such a large quantity of data is computationally expensive, and as we preprocessed our data, we were able to reduce the number of reviews to around 45,000.

To preprocess the data, we first removed any unnecessary columns to our research such as time of review and the reviewer's user ID. Additionally, we wanted to reduce the size of our dataset without compromising the accuracy of a beer's reviews by removing a subset of a given beer's reviews.Therefore, we elected to take an average of all reviews for a unique beer and use these values to create one data point. Thus, for each of the 56,857 unique beers in the dataset, there is one review that is representative of all given reviews in the original data. This reduces the size of our data set from 1.5 million to 56,857, which is somewhat more manageable for conducting analysis. We ended up with about $45,000$ beers for which we had useable data for creating a model.

We experimented with further separating each beer by style. There are 104 unique styles present in the data set, and we attempted to perform analysis within each style of beer. One benefit to this is that our graphs would be easier to interpret (and also feasible to graph in R for some computers), and more significantly, this would help ensure our analysis is useful in application. For example, if a dark beer such as a Russian imperial stout has a high overall review rating, it would make sense to compare its taste and aroma to other dark beers of the same style to draw inferences about why it is well-liked instead of comparing it to a light lager beer, which often has a completely different flavor profile and should not be held to the same standards. Additionally, some beer drinkers may have strong preferences for beer styles, and to ensure they are recommended a beer they like, it would be best to recommend beers within a style they enjoy.

However, we elected to not use this method for several reasons. Primarily, we wanted a simpler model, and the method above would have created a lot more complexity than the scope of the project allows. Additionally, we had some concerns about the usefulness of only recommending beers within the same style; while this would be simple, we did not feel as if it was truly creating a recommendation based on the characteristics of the specific inputted beer, which was our goal.

Instead, we ended up using K-means clustering with k=8 on the whole dataset, then assigned the cluster labels to our data. Using a for loop, we tested values of k from 1 to 100, and then selected the value of k at which the within-group sum of squares decreased by less than 5% as k increased by 1; this is essentially the same as selecting the "elbow" point from a graph. After deciding on k=5, we performed K-means clustering on the data. We then wrote a function that calculated the Euclidean distance between the input beer and beers within the same cluster. The function returned the names of the beers that have the shortest Euclidean distance to the given input beer.

In addition to K-means clustering, we used a Lasso regression. By doing so we can employ a shrinkage method which allows us to narrow down the suggestions in which our models give. This allows to identify which predictor variables influenced the overall review score the most.

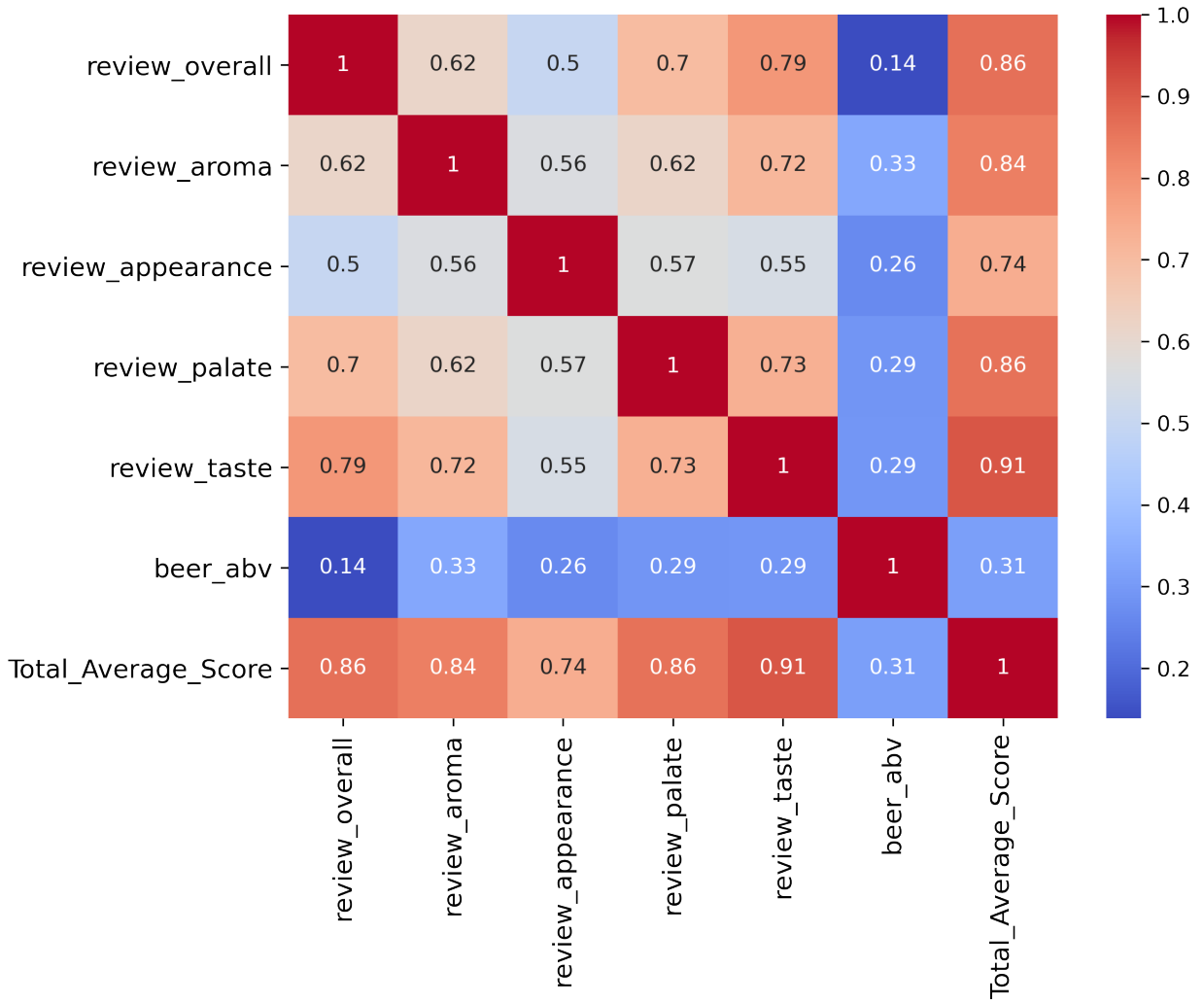
Another facet of this model is the use of item based collaborative filtering. Item-based collaborative filtering analyzes similarities between items based on user behavior or item characteristics to provide personalized recommendations. In the case of this project, our items are different kinds of beer, and the characteristics are a pre-specified list of features. This approach encompasses gathering data, computing item similarities, identifying similar beers, predicting user preferences and generating recommendations.

Overall, we will use a combination of clustering and regression methods to develop models that can accurately predict the popularity of beers based on their given scores.

Results:

After cleaning our data, we were able to summarize our important variables as follows:

```
##     beer_abv      review_overall   review_aroma   review_appearance
##  Min.   : 0.010   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.: 5.000   1st Qu.:3.429   1st Qu.:3.265   1st Qu.:3.500
##  Median : 5.700   Median :3.750   Median :3.625   Median :3.778
##  Mean   : 6.295   Mean   :3.655   Mean   :3.565   Mean   :3.690
##  3rd Qu.: 7.300   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :57.700   Max.   :5.000   Max.   :5.000   Max.   :5.000
##  review_palate    review_taste   total_average_score
##  Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:3.312   1st Qu.:3.333   1st Qu.:3.400
##  Median :3.667   Median :3.714   Median :3.702
##  Mean   :3.581   Mean   :3.608   Mean   :3.620
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:3.975
##  Max.   :5.000   Max.   :5.000   Max.   :5.000
```

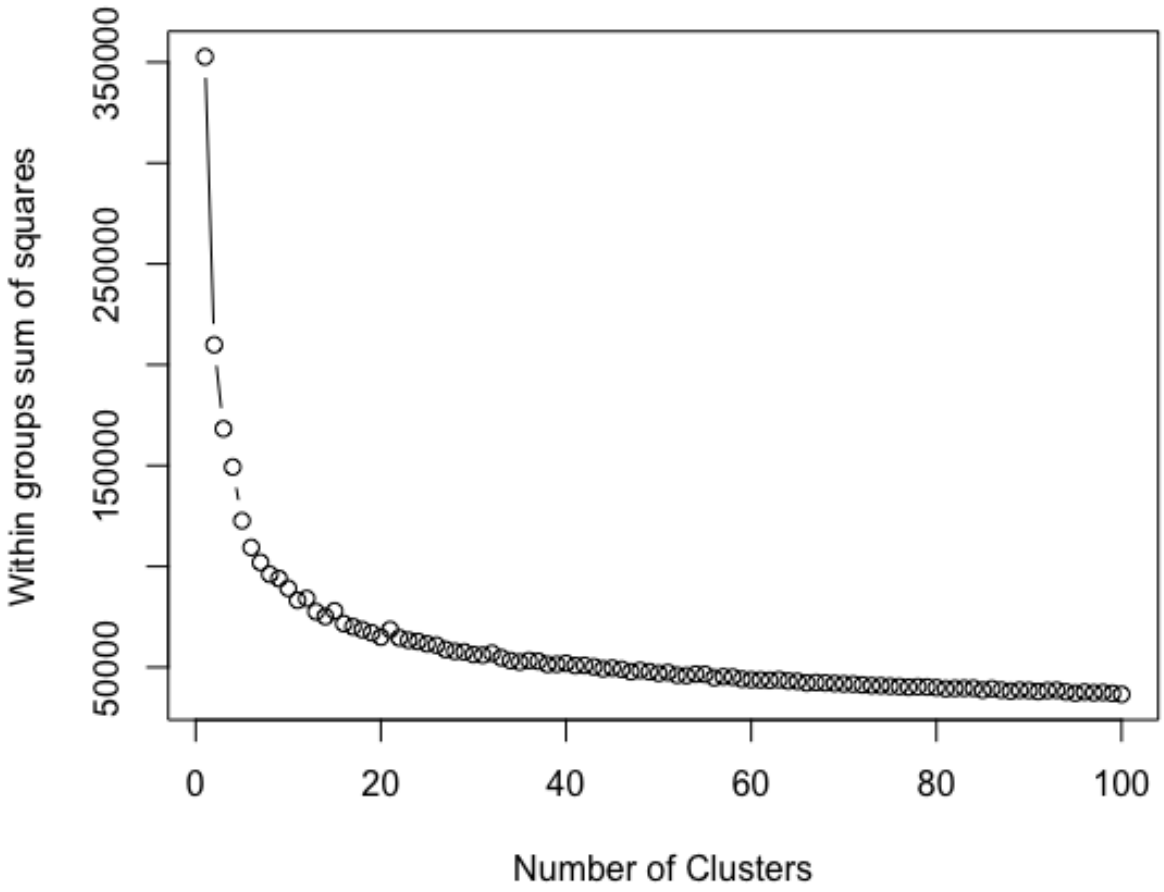


We created a heatmap of the data to see which features were most correlated with each other. Following this, we performed Lasso regression to see which features were most important in predicting the `overall_review` score for the beers. The results of our Lasso regression can be summarized in the following table:

	Mean Magnitude	Proportion selected	"Importance"	Average MSE
`beer_abv`	-0.0298	1.0	0.0298	.0691
`n_reviews`	0.0000	0.013	0.0000	
`review_aroma`	0.0905	1.0	0.0905	
`review_appearance`	0.0363	1.0	0.0363	
`review_palate`	0.2866	1.0	0.2866	
`review_taste`	0.5826	1.0	0.5826	

From the results of our Lasso regression the most important characteristics which influence the overall review of a beer is the taste and palate of the beer.

We then performed K-means clustering on our dataset so that we could recommend similar beers in each clusters. We present the following graph to show how we chose our value of K.



The choice was not super obvious from the graph so we wrote a function which chose the optimal value of K, by choosing the first value of K which decreased the with group sum of squares by less than 5%. In our case, $K = 8$. From here we created a function to find the closest beers within the cluster of the specified beer. By default our model returns the 5 most similar beers, but the amount of beers which are recommended by the model can be specified by the user.

The results of our function for 5 beers is as follows:

```
beer_recommend("Coors")
```

```
## [1] "Cluster for Coors (Brewery: Coors Brewing Company ): 6"
## [1] "Other beers in the same cluster: 4958"
## [1] "Number of recommendations: 5"

## [1] "Tsingtao"                 "Foster's Lager"
## [3] "Winter's Bourbon Cask Ale" "Kirin Ichiban"
## [5] "Sapporo Premium Beer"
```

```
beer_recommend("Corona Extra")
```

```
## [1] "Cluster for Corona Extra (Brewery: Grupo Modelo S.A. de C.V. ): 8"
## [1] "Other beers in the same cluster: 538"
## [1] "Number of recommendations: 5"

## [1] "Bass Pale Ale"        "Don de Dieu"          "Brooklyn Lager"
## [4] "India Pale Ale (IPA)" "Festina Pêche"
```

```
beer_recommend("Rainier Lager")
```

```
## [1] "Cluster for Rainier Lager (Brewery: Pabst Brewing Company ): 6"
## [1] "Other beers in the same cluster: 4958"
## [1] "Number of recommendations: 5"

## [1] "Rheingold Beer"
## [2] "San Miguel Pale Pilsen"
## [3] "Original Draught"
## [4] "Castello Birra Italiana / Friulana Blonda"
## [5] "Cerveza Aguila"
```

```
beer_recommend("Blue Moon Belgian White")
```

```
## [1] "Cluster for Blue Moon Belgian White (Brewery: Coors Brewing Company ): 8"
## [1] "Other beers in the same cluster: 538"
## [1] "Number of recommendations: 5"

## [1] "Oktoberfest"                    "Smuttynose IPA \"Finest Kind\""
## [3] "Samuel Adams Black Lager"       "Budweiser"
## [5] "ApriHop"
```

```
beer_recommend("Bud Light")
```

```
## [1] "Cluster for Bud Light (Brewery: Anheuser-Busch ): 8"
## [1] "Other beers in the same cluster: 538"
## [1] "Number of recommendations: 5"

## [1] "Mocha Porter"                  "Nut Brown Ale"
## [3] "Samuel Smith's Nut Brown Ale"  "Bell's Kalamazoo Stout"
## [5] "Green Flash West Coast I.P.A."
```

From our limited experience of drinking beers, we would say that our model creates reasonable recommendations.

Limitations:

There are several limitations to consider in our project. Firstly, the reduction of the dataset from over 1.5 million reviews to about 45,000 may lead to a loss of valuable information. Although this reduction was necessary to handle computational limitations, it could potentially result in a less comprehensive analysis.

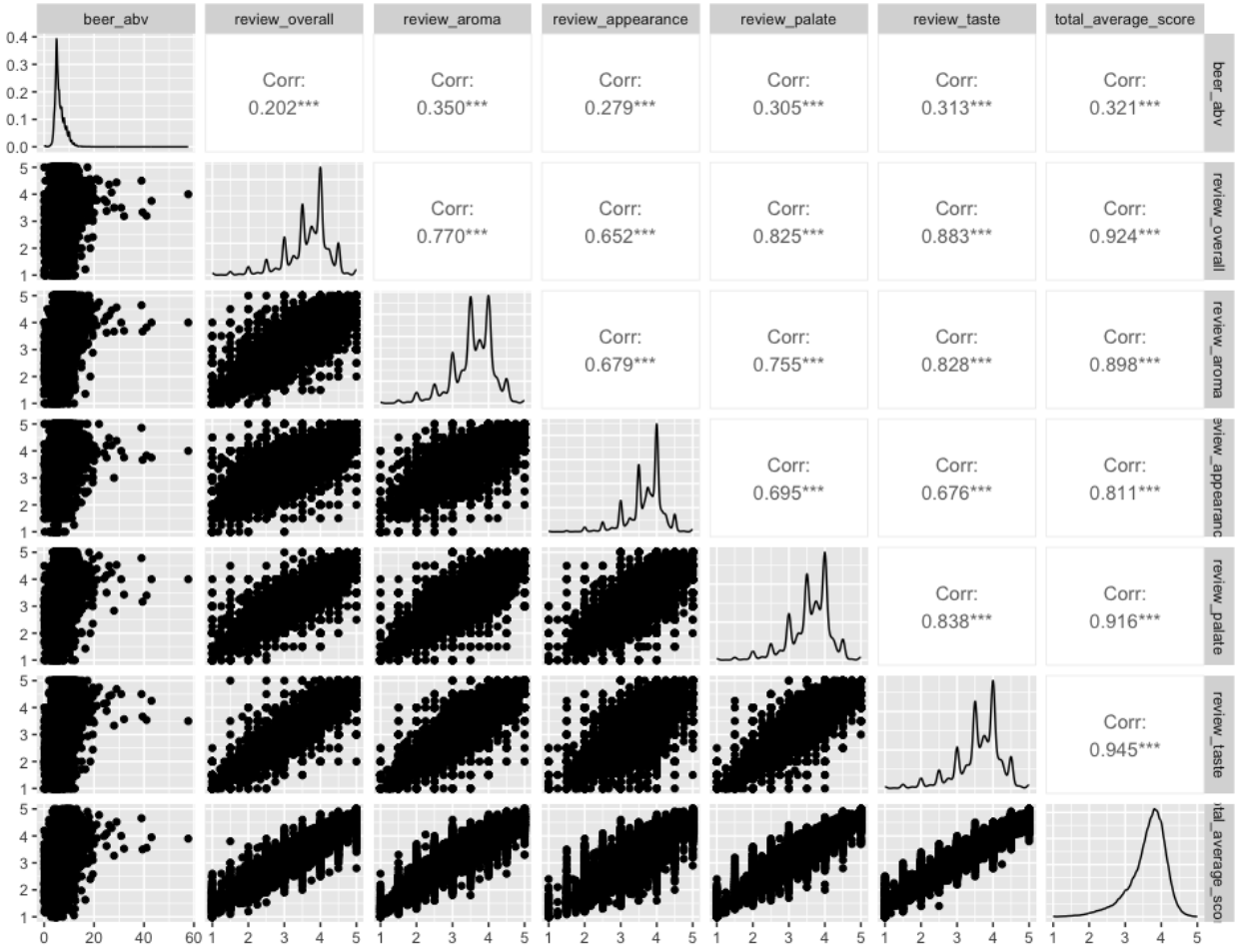
Additionally, by taking the average of all reviews for a unique beer in each category, we create one representative data point per beer. While this helps reduce the dataset size, it overlooks potential variations within a beer's reviews, potentially oversimplifying the analysis. (We ended up not utilizing this new Total_Average variable, however, in a more detailed analysis in which this new column could be used, this would be a limitation.)

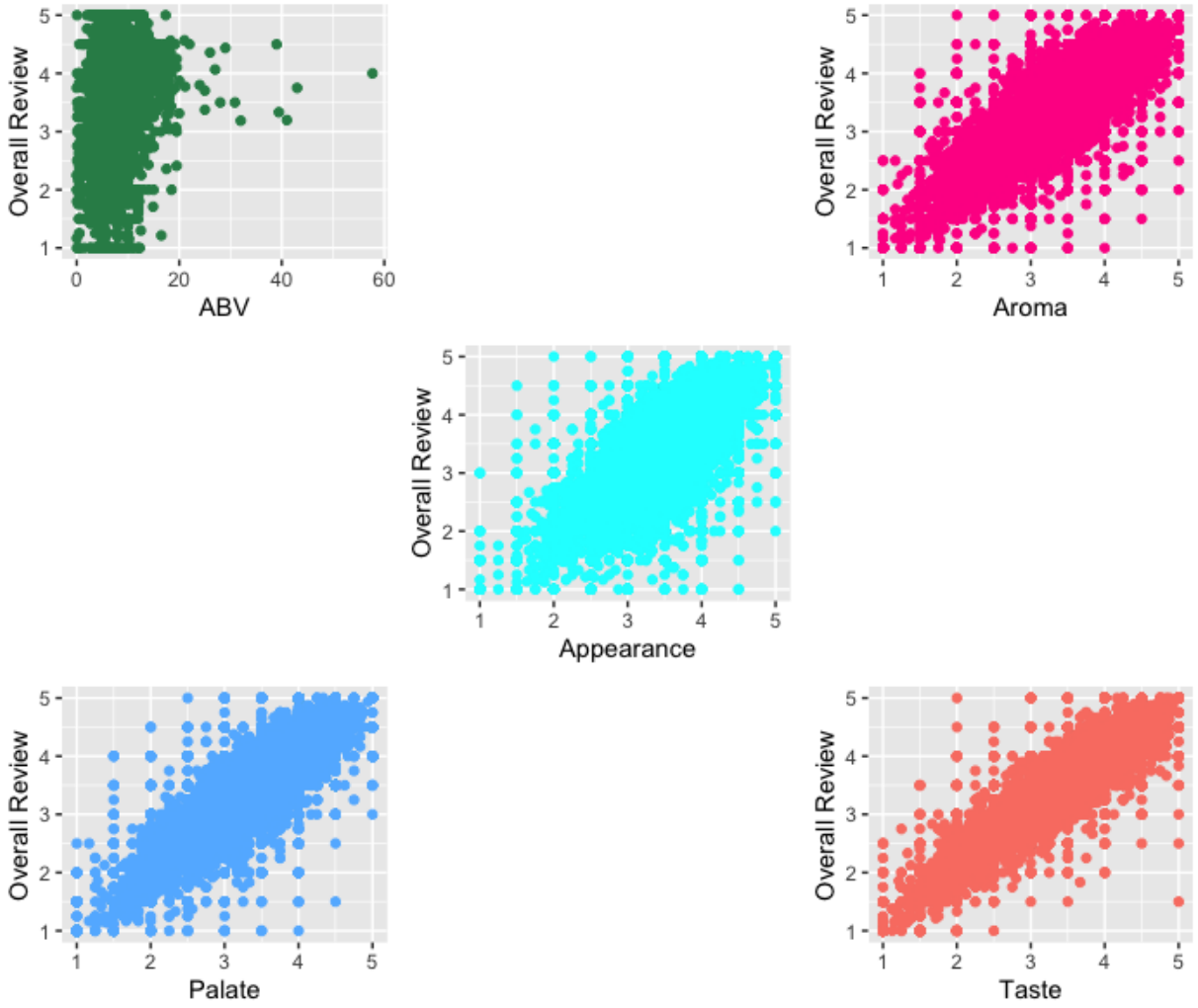
Furthermore, the choice of classification method, specifically K-nearest neighbors (KNN), and the use of Lasso regression introduce potential limitations. For example, KNN relies heavily on the choice of the number of neighbors (k) and the distance metric used, which can significantly impact the predictions. Regarding Lasso regression, it is a widely used method for feature selection and regularization. However, Lasso regression tends to shrink coefficients towards zero, leading to sparsity in the model and potentially overlooking important attributes. This can result in biased predictions and the exclusion of relevant variables from the model.

These limitations should be taken into account when interpreting the results and may require further investigation or refinement of the methodology in future research.

References

Appendix


beer_abv
review_overall
review_aroma
review_appearance
review_palate
review_taste
total_average_score
Corr: 0.202***
Corr: 0.350***
Corr: 0.279***
Corr: 0.305***
Corr: 0.313***
Corr: 0.321***
Corr: 0.770***
Corr: 0.652***
Corr: 0.825***
Corr: 0.883***
Corr: 0.924***
Corr: 0.679***
Corr: 0.755***
Corr: 0.828***
Corr: 0.898***
Corr: 0.695***
Corr: 0.676***
Corr: 0.811***
Corr: 0.838***
Corr: 0.916***
Corr: 0.945***



In the image seen above we have scatterplots comparing individual beer attributes to their overall score, this includes the ABV, aroma, apperance, palatte and taste. This was to identify if there were any highly correlated attributes to the overall review.

R Code:

```
#### 447 Project R Code ####
rm(list=ls())
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
#### Loading Required Libraries ####
library(tidyverse)
library(conflicted)
library(GGally)
library(reshape)
library(patchwork)
library(glmnet)

conflicts_prefer(dplyr::select)
conflicts_prefer(dplyr::filter)
conflicts_prefer(dplyr::rename)

#### Loading Data ####
#downloading and reading the beer reviews dataset as a tibble
beer<-as_tibble(read.csv("https://www.dropbox.com/s/nc3dygybis4wkbk/beer_reviews.csv?dl=1", header=TRUE)

#### Cleaning Data ####
## We are cleaning the data so that we have a dataset
#with the average review of each beer ###
#We will remove some of the non-important variables,
#but we may need them for later
beer_map<-select(beer, brewery_id,brewery_name, beer_name, beer_beerid,
                 beer_style, beer_abv)
#removing some of the unnecessary columns in the dataset
beer.clean<-select(beer,-review_profilename, -brewery_id,-beer_beerid,
                   -review_time)
#a lot of abv values were missing so we need to take care of those
#filling in the missing values that we can from other reviews of the same beer
beer.clean <- beer.clean %>%
  group_by(beer_name) %>%
  fill(beer_abv, .direction = "downup")
#removing the rows with missing values which we couldn't fill in
beer.clean<-beer.clean%>%
  filter(!is.na(beer_abv))
#now beer.clean is clean raw data if we need to use it later


#this will be useful later
map2<-select(beer_map, beer_name, beer_style, brewery_name)
map2<-map2%>%distinct(beer_name,.keep_all = TRUE)

##condensing the data down to what we want

#taking the average review of each beer for each category
beer_avg<-beer.clean%>%
  group_by(beer_name)%>%
  summarise(across(c(review_overall, review_aroma, review_appearance,
                     review_palate, review_taste, beer_abv), mean))
```

```r
#adding the style of each beer
beer_avg<-left_join(beer_avg, map2, by="beer_name")
beer_avg<-as_tibble(beer_avg)
#getting the standard deviation of each review
beer_sd<-beer.clean%>%
  group_by(beer_name)%>%
  summarise(across(c(review_overall, review_aroma, review_appearance,
                     review_palate, review_taste), sd))
#for the beers where there were only one review, there is 0 variance,
#but because of the sample variance formula, we would be dividing by 0, which induced NAs
beer_sd<-beer_sd%>%replace_na(list(review_overall=0,review_aroma=0,
                                   review_appearance=0,review_palate=0,review_taste=0))

#renaming some columns for later use
colnames.new.sd<-c(review_overall_sd="review_overall",
                   review_aroma_sd="review_aroma",
                  review_appearance_sd= "review_appearance",
                   review_palate_sd= "review_palate",
                  review_taste_sd="review_taste")
beer_sd<-rename(beer_sd, all_of(colnames.new.sd))

#calcualting the number of reviews for each beer
beer_n<-beer.clean%>%
  group_by(beer_name)%>%
  summarise(n=n())

#putting everything from above together
beer_avg<-beer_avg%>%
  add_column(n_reviews=beer_n$n,review_overall_sd=beer_sd$review_overall_sd,
             review_aroma_sd=beer_sd$review_aroma_sd,
             review_appearance_sd=beer_sd$review_appearance_sd,
             review_palate_sd= beer_sd$review_palate_sd,
             review_taste_sd=beer_sd$review_taste_sd)
#ordering the columns in a way that it is easy to view raw
beer_avg<-beer_avg%>%
  select(beer_name,brewery_name, beer_style, beer_abv,
         n_reviews, review_overall, review_overall_sd, review_aroma,
         review_aroma_sd, review_appearance, review_appearance_sd, review_palate,
         review_palate_sd,  review_taste, review_taste_sd)
## Creating a new variable which is the avrage of all the review scores
beer_avg<-beer_avg%>%mutate(beer_avg, total_average_score= (review_overall+review_aroma+review_appearanc
head(beer_avg)

#### Some summary statistics for each category
summary(beer_avg[,c(4,6,8,10,12,14,16)])

##Data visiaulization
ggpairs(beer_avg, columns=c(4,6,8,10,12,14,16))
p1<-ggplot(beer_avg, aes(x=beer_abv, y=review_overall))+
  geom_point(color="#2E8B57")+
  labs(x = "ABV", y = "Overall Review")
p2<-ggplot(beer_avg, aes(x=review_aroma, y=review_overall))+
  geom_point(color="#FF1493")+
```

```
  labs(x = "Aroma", y = "Overall Review")
p3<-ggplot(beer_avg, aes(x=review_appearance, y=review_overall))+
  geom_point(color="#00FFFF")+
  labs(x = "Appearance", y = "Overall Review")
p4<-ggplot(beer_avg, aes(x=review_palate, y=review_overall))+
  geom_point(color="#63B8FF")+
  labs(x = "Palate", y = "Overall Review")
p5<-ggplot(beer_avg, aes(x=review_taste, y=review_overall))+
  geom_point(color="#FA8072")+
  labs(x = "Taste", y = "Overall Review")
plot.lout<-"
11##22
##33##
44##55
"
p1+p2+p3+p4+p5+plot_layout(design = plot.lout)

#### Modeling ####

#selecting only the numeric variables in our dataset
numeric_df <- beer_avg %>%
  select(beer_abv, n_reviews, review_overall, review_aroma, review_appearance,
         review_palate, review_taste, total_average_score)
numeric_df<-data.frame(numeric_df)

## Fitting a Lasso regression model to see which variables affect the overall review
set.seed(11)
#Number of times we regenerate training sets
B<-1000
#initializing empty vectors for storing results
lasso01mat<-double()
lassobetamat<-double()
mse.lasso<-double()

for (i in 1:B){
  #Setting a random seed for each iteration
  set.seed(i+sample(1:B,1))

  #splitting the data into training and testing sets
  train<-sample(1:dim(numeric_df)[1], dim(numeric_df)[1]*.8)
  test<- -train
  beer.train<-numeric_df[train,]
  beer.test<-numeric_df[test,]

  train.mat<-model.matrix(review_overall ~. -total_average_score,
                          data=beer.train)[,-1]
  test.mat<-model.matrix(review_overall~. -total_average_score,
                         data=beer.test)[,-1]
  #Fitting lasso regression on the training set
  #note alpha = 1 is default
  fit.lasso<-glmnet(train.mat,beer.train$review_overall, alpha=1)
  #performing cross-fold validation, default is 10 fold
  cv.lasso<-cv.glmnet(train.mat,beer.train$review_overall, alpha=1)
```

```
  #choosing the minimum lambda value
  bestlam.lasso<-cv.lasso$lambda.min
  bestlam.lasso
  #predicted values and MSE calculations
  pred.lasso<-predict(fit.lasso, s=bestlam.lasso, newx=test.mat)
  mse.lasso[i]<-mean((pred.lasso-beer.test$review_overall)^2)
  #storing the beta coefficients
  beta.lasso<-predict(fit.lasso, s=bestlam.lasso, type="coefficients")
  lassobetamat<-rbind(lassobetamat, beta.lasso[-1,1])
  #storing whether or not the explanatory variable was included in the model
  #stored as a 1 if it is included and a 0 if it is not included
  lasso01<-as.integer(abs(beta.lasso[-1,1])>0)
  lasso01mat<-rbind(lasso01mat,lasso01)
}
#naming the columns of the matricies of the magnitude of coefficients or if
#predictor variables were chosen to match the original data
colnames(lasso01mat)<-colnames(numeric_df)[c(-8,-3)]
colnames(lassobetamat)<-colnames(numeric_df)[c(-3,-8)]
## Proportion of time that the predictor variables were chosen
colMeans(lasso01mat)
##  Mean value for each coefficient
colMeans(lassobetamat)
## Average MSE
mean(mse.lasso)
## Ranking our explanatory variables
colMeans(lasso01mat)*colMeans(lassobetamat)

## performing k-means clustering on the dataset to cluster similar beers
# Normalize the data
numeric_df <- scale(numeric_df)
## Choosing K by looking at a graph
set.seed(1234)
wss2 <- (nrow(numeric_df)-1)*sum(apply(numeric_df,2,var))
for (i in 2:100) {
  wss2[i] <- sum(kmeans(numeric_df, centers=i)$withinss)
}
plot(1:400, wss2, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")

## Finding the "optimal" K by seeing which value of K decreases the WSS by less than 5%
set.seed(1234)
wss <- (nrow(numeric_df)-1)*sum(apply(numeric_df,2,var))
old_wss <- Inf
for (i in 2:100) {
  wss[i] <- sum(kmeans(numeric_df, centers=i)$withinss)
  percentage_decrease <- ((old_wss - wss[i]) / old_wss) * 100
  if (i > 1 && !is.nan(percentage_decrease) && percentage_decrease < 5) {
    break
  }
  old_wss <- wss[i]
}
optimal_k <- which.min(wss)
```

```r
#Performing K means clustering using our "optimal" K
set.seed(80085)
kmeans_result <- kmeans(numeric_df, centers=optimal_k, iter.max = 100)

# Add the cluster assignments back to the original data
beer_avg$cluster <- kmeans_result$cluster

#function to recommend beers based on how we clustered them
# We are choosing the 5 most similar beers by default but that can be changed
beer_recommend <- function(beer_name, N = 5) {
  beer_cluster <- beer_avg$cluster[which(beer_avg$beer_name == beer_name)]
  print(paste("Cluster for", beer_name, "(Brewery:",
              beer_avg$brewery_name[which(beer_avg$beer_name == beer_name)],
              "):", beer_cluster))

  # Filter beers in the same cluster without excluding the input beer
  same_cluster_beers <- beer_avg %>%
    filter(cluster == beer_cluster)

  cluster_beers <- same_cluster_beers[same_cluster_beers$beer_name != beer_name, ]

  print(paste("Other beers in the same cluster:", nrow(cluster_beers)))


  if (nrow(cluster_beers) > 0) {
    # Extract numeric features for the given beer and the other beers in the cluster
    beer_features <- matrix(as.numeric(beer_avg[beer_avg$beer_name == beer_name,
                                                c(4,5,6,8,10,12,14,16)]), nrow = 1)
    # Compute distances from the given beer to all other beers in the cluster
    distances <- sapply(1:nrow(cluster_beers), function(i) {
      cluster_features <- as.matrix(cluster_beers[i, c(4,5,6,8,10,12,14,16)])
      dist(rbind(beer_features, cluster_features))
    })


    # Select the N beers with the smallest distances
    closest_beers <- cluster_beers$beer_name[order(distances)[1:N]]

    print(paste("Number of recommendations:", length(closest_beers)))

    return(closest_beers)
  } else {
    return(character(0))
  }
}

#Seeing which beers our model recommends based on which beers we like

beer_recommend("Coors")
beer_recommend("Modelo Especial")
beer_recommend("Pacífico")
beer_recommend("Pacifico Light")
beer_recommend("Corona Extra")
```

```
beer_recommend("Corona Light")
beer_recommend("Rainier Lager")
beer_recommend("Bud Light")
beer_recommend("Bud Light Lime")
beer_recommend("Blue Moon Belgian White")
beer_recommend("Wainwright")
```