Project Report

Ranen Oomen-Danckert

19/04/2024

Problem Context

When performing technical wine evaluations the taster aims to roughly quantify different aspects of a wine such as a wine's body, sweetness, acidity, and bitterness. While roughly quantifying these different characteristics is not extremely difficult after some training and calibration it is unclear if there is a clear correlation between these characteristics and a wine's quality.

Inference Task

This project will attempt to understand which wine characteristics have the strongest effect on wine quality. Our predictor variables will be a wine's body, acidity, and price, and our response variable will be wine quality. To understand which characteristics have the strongest effect on quality, we will perform L1 regularized linear regression, forward variable selection, and MCMC to estimate which predictors better correspond to wine quality in linear models. We hypothesize that L1 regularized linear regression will include all three predictors, with small positive slopes on price and body and a small negative slope on acidity, forward variable selection will select price, followed by body and acidity, and that 95% Bayesian credible intervals will all include 0 as a possible slope. We expect the Bayesian method to provide a more comprehensive view of the slope values and the linear models to all perform roughly the same with respect to mean squared error in prediction quality.

Modeling Challenge

The modeling challenge lies in determining a quantitative way of evaluating wine quality. Given the diverse attributes of different wines it may be difficult to determine rules of thumb, but it would not be completely surprising to associate high acidity, often caused by premature grape harvest, with poor quality or fuller bodies to be associated with better quality wines. Being able to differentiate wine quality from price would be beneficial for shoppers looking to enjoy their wine without unnecessary financial strain. This is why we will compare three different and interpretable methods for judging wine quality from body, acidity and price, seeing if there are useful rules of thumb when buying wine.

Literature Review

Most of the existing analysis on Kaggle with the Spanish wine quality data that we are using for our analysis is concerned with predicting price from the other variables. The closest analysis to the one we will be performing uses categorical regression trees with boosting and Shapley Additive Explanation values to estimate the importance of different predictors. The source code for this analysis can be found here. Our analysis of wine quality predictors is simpler than one with regression trees and introduces the Bayesian approach to the analysis.

Lasso Model

We preprocess our data by standardizing all of our predictors, and centering the ratings. For our Lasso model, we choose the strongest regularization strength within 1 standard error of the minimum cross-validation mean squared error. This model assigns zero slope to acidity and body, and a small positive slope to price. The cross-validation mean squared error is 0.0111.

Forward Variable Selection

Using the same design matrix as we did for our Lasso model we now perform forward variable selection. Our forward variable selection algorithm chooses price, followed by body then acidity. This is a similar to result to what we found with our Lasso Model, and it explicitly ranks body ahead of acidity in terms of usefulness for prediction.

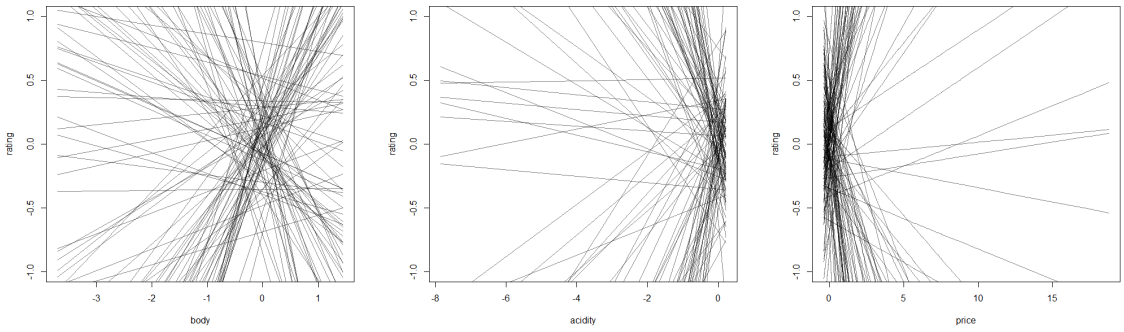
Bayesian Model

We will construct a Bayesian Linear model where our predictors are body, acidity and price, and our response variable is rating. β_1 is the slope corresponding to the wine's body, β_2 corresponds to the wine's acidity and β_3 corresponds to the wine's price. Our model is specified below.

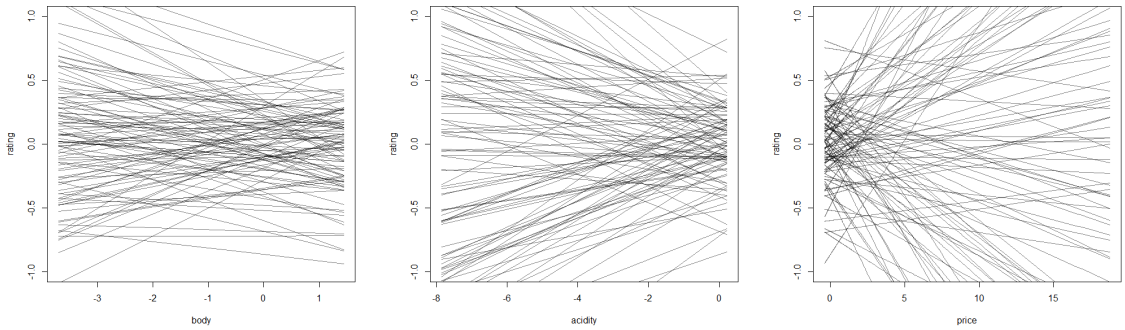
$$\begin{aligned}
\beta_1 &\sim \mathcal{N}(0,1) \\
\beta_2 &\sim \mathcal{N}(0,1) \\
\beta_3 &\sim \mathcal{N}(0,1) \\
\text{intercept} &\sim \mathcal{N}(0,0.3) \\
\sigma^2 &\sim \text{Exp}(2) \\
y_i|X &\sim \mathcal{N}(\beta X_i + \text{intercept}, \sigma^2)
\end{aligned}$$

The prior for the slope parameters is standard normal, since we have standardized our data we will want to use the same prior for all three predictor slopes in order to avoid any bias. We have a prior for an intercept that we will center at 0 since we don't want to bias the posterior intercept away from zero. Since the centered ratings range from -0.06 to 0.6 we don't want a massive standard deviation for our likelihood which is why we use the relatively high rate parameter on the prior.

We perform a prior predictive check to confirm that our prior choices are reasonable. Plots of potential outputs for our three slopes along with our potential intercept are shown below.



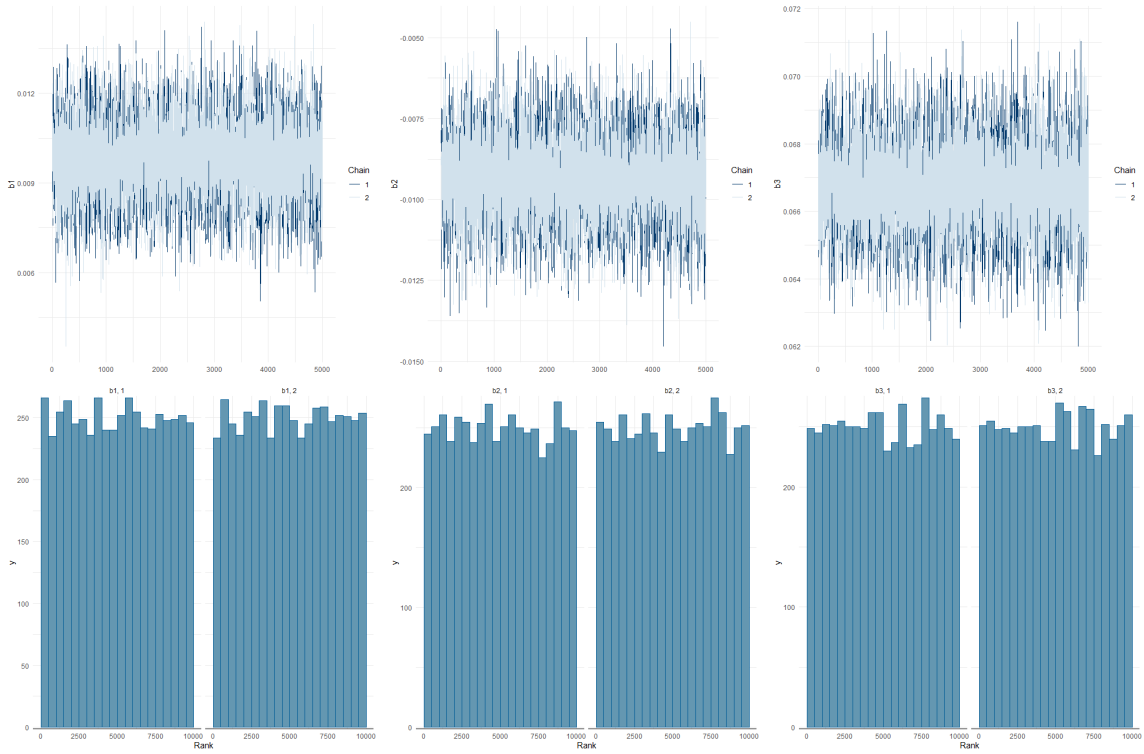
It looks like our prior is producing reasonable intercept values, but the slopes are far too extreme so we need to significantly reduce the standard deviation of our slope priors. We will try 0.1 as our standard deviation instead of 1. The updated prior predictive plots are shown below.



These prior predictive plots seem much more reasonable while maintaining sufficient flexibility across possible outcomes. Our updated model is then

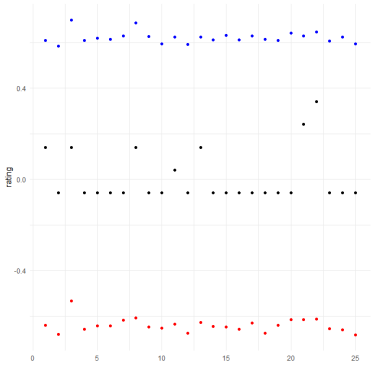
$$\begin{aligned}
\beta_1 &\sim \mathcal{N}(0, 0.1) \\
\beta_2 &\sim \mathcal{N}(0, 0.1) \\
\beta_3 &\sim \mathcal{N}(0, 0.1) \\
\text{intercept} &\sim \mathcal{N}(0, 0.3) \\
\sigma^2 &\sim \text{Exp}(2) \\
y_i | X &\sim \mathcal{N}(\beta X_i + \text{intercept}, \sigma^2)
\end{aligned}$$

We now estimate our posterior distributions for our body, acidity, and price slopes by running Markov chain Monte Carlo (MCMC) using STAN. We use 2 chains for 10000 iterations and examine the trace and rank plots to ensure the chains mix well. The trace and rank plots for the body, acidity, and price slopes are shown below.



The trace plots for all three predictors show that the chains are mixing well and all of the rank plots are well balanced so it appears that slow mixing is not a problem for our MCMC algorithm.

We now perform our posterior predictive check. To do this we use a validation set that is roughly 10% of our data and generate predictions for this validation set in our MCMC algorithm. We find that the actual coverage of our 95% credible interval for the validation set of 632 observations is 100%, so our credible interval is over-conservative. A plot of the first 25 observations and their corresponding intervals is shown below.



The mean of the predictions from our Bayesian model yielded a mean squared prediction error of 0.0106.

Discussion

With respect to the importance of body, acidity and price, our MCMC algorithm returned similar results to our L1 regularized linear regression and our forward variable selection method. All three methods, as expected, placed greater magnitude slopes on price, followed by body, and then acidity. The mean squared errors of L1 regularized linear regression and Bayesian Linear regression were also very close, but none of the 95% credible intervals for the slope parameters contained zero. The Bayesian linear model specified in this report determined that price has the strongest correlation with wine quality, followed by fuller body, and lower acidity. The key limitation of this analysis is that it is too general to provide good rules of thumb for assessing wine quality. While it is often true that wines high in acidity are unpleasant it is not always true. Champagnes for example, are usually higher in acidity but widely considered high quality. An analysis of wines grouped by grape varietal, or processing method may provide more useful rules of thumb on a per wine basis. With a more general dataset, like the one used in this analysis a method like decision trees, which have been used by other analysts, may do a better job of identifying patterns in wine quality, although they will sacrifice interpretability as the trees grow. Wine quality is dependent on many factors, and determining it mechanically is difficult. Great care should be taken in any analysis of wine quality.

Appendix

STAN Code

```
// The input data is a vector 'y' of length 'N'.
data {
  int<lower=0> N;  // number of wines
  int<lower=0> N_valid; // number of wines to predict with
  vector<lower=-4,upper=2>[N] body; // body rating
  vector<lower=-8, upper=1>[N] acidity; // acidity rating
  vector<lower=-1>[N] price; // price of the wine
  vector<lower=-1, upper=1>[N] y; // quality rating
  vector<lower=-4,upper=2>[N_valid] body_pred; // predictors
  vector<lower=-8, upper=1>[N_valid] acidity_pred;
  vector<lower=-1>[N_valid] price_pred;
```

```
}

parameters {
  real b1; // body slope
  real b2; // acidity slope
  real b3; // price slope
  real intercept; // model intercept
  real <lower=0>sigma; // standard deviation of linear model
}

model {
  // prior
  b1 ~ normal(0,0.1);
  b2 ~ normal(0,0.1);
  b3 ~ normal(0,0.1);
  intercept ~ normal(0, 0.3);
  sigma ~ exponential(0.1);

  // likelihood
  y ~ normal(intercept + b1*body + b2*acidity + b3*price, sigma^2);
}

generated quantities {
  vector[N_valid] quality_pred;
  for (i in 1:N_valid)
    quality_pred[i] = normal_rng(body_pred[i]*b1 + acidity_pred[i]*b2 + price_pred[i]*b3 + intercept, si

}
```

Non-Bayesian Code

```
library(tidyverse)
library(glmnet)
library(stats)

spain_wine_ratings <- read_csv("C:/Users/ranen/Downloads/wines_SPA.csv")

## Lasso

## We have enough full observations to not worry about omitting missing values
spain_wine_ratings <- na.omit(spain_wine_ratings)

response <- c("rating")

rating <- select(spain_wine_ratings, all_of(response))

## Transform Tibble to be compatible with glmnet
## standardize columns
acidity_mean <- mean(spain_wine_ratings$acidity)
acidity_sd <- sd(spain_wine_ratings$acidity)
body_mean <- mean(spain_wine_ratings$body)
body_sd <- sd(spain_wine_ratings$body)
```

```
price_mean <- mean(spain_wine_ratings$price)
price_sd <- sd(spain_wine_ratings$price)
rating_mean <- mean(rating$rating)

rating <- rating %>%
  mutate(rating = rating-rating_mean)

spain_wine_ratings <- spain_wine_ratings %>%
  select(c(acidity, body, price)) %>%
  mutate(acidity = (acidity - acidity_mean)/acidity_sd) %>%
  mutate(body = (body - body_mean)/body_sd) %>%
  mutate(price = (price - price_mean)/price_sd)


## glmnet takes matrices as input

X <- as.matrix(spain_wine_ratings)
y <- as.matrix(rating)

cvfit <- cv.glmnet(X,y)

plot(cvfit)

coef(cvfit, s = cvfit$lambda.1se)
coef(cvfit, s = cvfit$lambda.min)


## Forward variable selection

require(leaps)

forward_var <- regsubsets(x=X,y=y, method = "forward")
summary(forward_var)
```

Bayesian Code

```
body_vec <- spain_wine_ratings$body
acidity_vec <- spain_wine_ratings$acidity
price_vec <- spain_wine_ratings$price
rating_vec <- rating$rating

## Prior Predictive check

set.seed(1)
intercepts <- rnorm(100,0,0.3)
bodys <- rnorm(100,0,1)
aciditys <- rnorm(100,0,1)
prices <- rnorm(100,0,1)

## x-values
body_vals <- sort(unique(body_vec))
```

```r
acidity_vals <- sort(unique(acidity_vec))
price_vals <- sort(unique(price_vec))

## Plot possible priors

plot(1, type = "n", xlab = "body",
     ylab = "rating", xlim = c(min(body_vals), max(body_vals)),
     ylim = c(-1, 1))
for(i in (1:100)){
  lines(x = body_vals, y = bodys[i]*body_vals+intercepts[i],
        col = rgb(red = 0, green = 0, blue = 0, alpha = 0.5))
}


plot(1, type = "n", xlab = "acidity",
     ylab = "rating", xlim = c(min(acidity_vals), max(acidity_vals)),
     ylim = c(-1, 1))
for(i in (1:100)){
  lines(x = acidity_vals, y = aciditys[i]*acidity_vals+intercepts[i],
        col = rgb(red = 0, green = 0, blue = 0, alpha = 0.5))
}

plot(1, type = "n", xlab = "price",
     ylab = "rating", xlim = c(min(price_vals), max(price_vals)),
     ylim = c(-1, 1))
for(i in (1:100)){
  lines(x = price_vals, y = prices[i]*price_vals+intercepts[i],
        col = rgb(red = 0, green = 0, blue = 0, alpha = 0.5))
}


## update priors
bodys_updated <- rnorm(100, mean = 0, 0.1)
aciditys_updated <- rnorm(100, mean = 0, 0.1)
prices_updated <- rnorm(100, mean = 0, 0.1)

plot(1, type = "n", xlab = "body",
     ylab = "rating", xlim = c(min(body_vals), max(body_vals)),
     ylim = c(-1, 1))
for(i in (1:100)){
  lines(x = body_vals, y = bodys_updated[i]*body_vals+intercepts[i],
        col = rgb(red = 0, green = 0, blue = 0, alpha = 0.5))
}


plot(1, type = "n", xlab = "acidity",
     ylab = "rating", xlim = c(min(acidity_vals), max(acidity_vals)),
     ylim = c(-1, 1))
for(i in (1:100)){
  lines(x = acidity_vals, y = aciditys_updated[i]*acidity_vals+intercepts[i],
        col = rgb(red = 0, green = 0, blue = 0, alpha = 0.5))
}
```

```
plot(1, type = "n", xlab = "price",
     ylab = "rating", xlim = c(min(price_vals), max(price_vals)),
     ylim = c(-1, 1))
for(i in (1:100)){
  lines(x = price_vals, y = prices_updated[i]*price_vals+intercepts[i],
        col = rgb(red = 0, green = 0, blue = 0, alpha = 0.5))
}

## Run MCMC

suppressPackageStartupMessages(require(rstan))

fit = stan(
  "wine.stan",
  seed = 1,
  data = list(
    y = rating_vec,
    body = body_vec,
    acidity = acidity_vec,
    price = price_vec,
    N = length(rating_vec)
  ),
  chains = 2,
  iter = 10000
)

## check for slow mixing

suppressPackageStartupMessages(require(bayesplot))

# Trace plots
mcmc_trace(fit, pars = c("b1")) + theme_minimal()
mcmc_trace(fit, pars = c("b2")) + theme_minimal()
mcmc_trace(fit, pars = c("b3")) + theme_minimal()

# rank plots
mcmc_rank_hist(fit, pars = c("b1")) + theme_minimal()
mcmc_rank_hist(fit, pars = c("b2")) + theme_minimal()
mcmc_rank_hist(fit, pars = c("b3")) + theme_minimal()

## Posterior Predictive

##randomly choose an observation
n = length(spain_wine_ratings$body)

## slightly smaller than 10% of observations used in validation set
validation_indices <- round(runif(0.1*n,-0.5, n+0.5))
validation_indices <- unique(validation_indices)

spain_wine_train <- filter(spain_wine_ratings, !row_number() %in% validation_indices)
spain_wine_valid <- spain_wine_ratings[validation_indices,]

rating_train <- filter(rating, !row_number() %in% validation_indices)
```

```
rating_valid <- rating[validation_indices,]

rating_train_vec <- rating_train$rating
body_train_vec <- spain_wine_train$body
body_valid_vec <- spain_wine_valid$body
acidity_train_vec <- spain_wine_train$acidity
acidity_valid_vec <- spain_wine_valid$acidity
price_train_vec <- spain_wine_train$price
price_valid_vec <- spain_wine_valid$price

## Run MCMC, generating predictions for validation set

calibration_fit <- stan(
  "wine.stan",
  seed = 1,
  data = list(
    y = rating_train_vec,
    body = body_train_vec,
    acidity = acidity_train_vec,
    price = price_train_vec,
    N = length(rating_train_vec),
    N_valid = length(body_valid_vec),
    body_pred = body_valid_vec,
    acidity_pred = acidity_valid_vec,
    price_pred = price_valid_vec
  ),
  chains = 2,
  iter = 10000
)

calibration_quantiles <- summary(calibration_fit, probs = c(0.025, 0.975))$summary
calibration_quantiles <- calibration_quantiles[,c("2.5%", "97.5%")]
calibration_quantiles <-
  calibration_quantiles[!rownames(calibration_quantiles) %in%
                          c("b1", "b2", "b3", "intercept", "sigma", "lp__"),]

## Now calculate proportion of ratings in credible interval

inside <- 0
n_valid <- nrow(rating_valid)
for  (i in 1:n_valid) {
  if(rating_valid[i,] > calibration_quantiles[i,1] &&
     rating_valid[i,] < calibration_quantiles[i,2]) {
    inside <- inside + 1
  }
}

calibration_tib <-
  tibble("rating" = rating_valid$rating,
         "lower bound" = calibration_quantiles[,1],
         "upper bound" = calibration_quantiles[,2])

## Show credible interval
```

```r
ggplot() +
  geom_point(aes(x = seq(1,25), y=rating_valid$rating[1:25])) +
  geom_point(aes(x = seq(1,25), y = calibration_tib$`lower bound`[1:25]), colour = "red") +
  geom_point(aes(x = seq(1,25), y = calibration_tib$`upper bound`[1:25]), colour = "blue") +
  labs(y = "rating",x = "") +
  theme_minimal()

## Predict for all data

full_fit <- stan(
  "wine.stan",
  seed = 1,
  data = list(
    y = rating_vec,
    body = body_vec,
    acidity = acidity_vec,
    price = price_vec,
    N = length(rating_vec),
    N_valid = length(body_vec),
    body_pred = body_vec,
    acidity_pred = acidity_vec,
    price_pred = price_vec
  ),
  chains = 2,
  iter = 10000
)

full_quantiles <- summary(full_fit)$summary
preds <- full_quantiles[,"mean"]
preds <- preds[6:6334]

## Bayes prediction mean MSE
stopifnot(length(preds)==length(rating_vec))

bayes_mse = sum((rating_vec - preds)**2)/length(preds)
```