Baseball Analytics Questions

Patrick Mellady

1

Using the attached "pitch_data.csv" (column description below), develop a statistical model to predict the probability of a pitch being called a strike, conditional on the batter not swinging. Your code should append your predicted probability to the original dataset and calculate your model's Brier score.

Solution

Theory

We will consider a model with two binary random variables, Y_{i1} and Y_{i2}, where:

$$Y_{i1} = \begin{cases} 1 & \text{if strike} \\ 0 & \text{else} \end{cases}$$

$$Y_{i2} = \begin{cases} 1 & \text{if swing} \\ 0 & \text{else} \end{cases}$$

With the above random variables, we define

$$\pi_{ijk} = P(Y_{k1} = i, Y_{k2} = j) \qquad i, j = 0, 1 \quad k = 1, 2, 3, \cdots, n$$

Thus, the probability we want to model is

$$\pi_k = P(Y_{k1} = 1 | Y_{k2} = 0) = \frac{P(Y_{k1} = 1, Y_{k2} = 0)}{P(Y_{k2} = 0)}$$

By the law of total probability

$$P(Y_{k2} = 0) = P(Y_{k1} = 1, Y_{k2} = 0) + P(Y_{k1} = 0, Y_{k2} = 0)$$

So, we can rewrite our desired probability as

$$\pi_k = \frac{\pi_{10k}}{\pi_{10k} + \pi_{00k}}$$

We will model π_{10k} and π_{00k} using a logistic regression with the logit link, i.e.

$$logit(\pi_{10k}) = \log(\frac{\pi_{10k}}{1-\pi_{10k}}) = x^T\beta$$
$$logit(\pi_{00k}) = \log(\frac{\pi_{00k}}{1-\pi_{00k}}) = x^T\gamma$$

where x^T is a vector of covariates given in the dataset. By fitting the above models, we can find estimates for the weights, $\hat{\beta}$ and $\hat{\gamma}$ and use those to predict π_{10k} and π_{00k} using the following formulas:

$$\hat{\pi}_{10k} = \frac{e^{x^T\hat{\beta}}}{1+e^{x^T\hat{\beta}}}, \quad \hat{\pi}_{00k} = \frac{e^{x^T\hat{\gamma}}}{1+e^{x^T\hat{\gamma}}}$$

We then plug the above estimates into the formula above to estimate π_k with:

$$\hat{\pi}_k = \frac{\hat{\pi}_{10k}}{\hat{\pi}_{10k}+\hat{\pi}_{00k}}$$

Implementation

Since each unique game state, pitch type, pitcher, catcher, and umpire will necessitate a large design matrix with many covariates, we will use the LASSO penalty to perform variable selection. First, we will import the data set, ensure that the covariates are of the correct R variable type, and remove NA entries from the data

```
library(glmnet)
pitch<-read.csv("pitch_data.csv", header=TRUE)
noms<-colnames(pitch)
pitch[,3:17] <- lapply(pitch[,3:17] , factor)
pitch$spin_rate<-as.numeric(pitch$spin_rate)
pitch<-pitch[is.na(pitch$spin_rate)==FALSE,]
```

Next, we will define the events $A_k : Y_1 = 1$ and $Y_2 = 0$ and $B_k : Y_1 = 0$ and $Y_2 = 0$, so that $P(A_k) = \pi_{10k}$ and $P(B_k) = \pi_{00k}$, and append the outcomes to the pitch dataframe

```
strike<-pitch$is_strike
swing<-pitch$is_swing
A<-rep(0,nrow(pitch))
B<-rep(0,nrow(pitch))
pitch<-cbind(A, B, pitch)
colnames(pitch)<-c("A", "B", noms)

pitch[pitch$is_strike==1 & pitch$is_swing==0, 1]<-1
pitch[pitch$is_strike==0 & pitch$is_swing==0, 2]<-1
pitch<-pitch[,c(1,2,5:25)]
```

Now, we will create the design matrices for the two probabilities. Since the strikezone is a restricted space over the plate, considering pitch_location_x and pitch_location_z linearly does not make sense. To correct this, we will consider quadratic pitch location data, which should give higher strike probabilities to pitches centered in the strikezone.

```
xfactors1<-model.matrix(A~.-1,pitch[, c(1, 3:17)])
X1<-as.matrix(data.frame(xfactors1, as.matrix(pitch[, 18:23]),
                         as.matrix(pitch[, 18]^2),
                         as.matrix(pitch[, 19]^2),
                         as.matrix(pitch[, 18]*pitch[, 19])))
```

```
xfactors2<-model.matrix(B~.-1,pitch[, 2:17])
X2<-as.matrix(data.frame(xfactors2, as.matrix(pitch[, 18:23]),
                         as.matrix(pitch[, 18]^2),
                         as.matrix(pitch[, 19]^2),
                         as.matrix(pitch[, 18]*pitch[, 19])))
```

We fit the models with the LASSO penalty for variable selection.

```
y1<-as.vector(pitch$A)
mod1<-glmnet(X1, y1, family="binomial", alpha=1, standardize = TRUE, intercept=TRUE)

y2<-as.vector(pitch$B)
mod2<-glmnet(X2, y2, family="binomial", alpha=1, standardize = TRUE, intercept=TRUE)
```

Now that we have calculated the weights, we can predict the probability of a strike given the batter does not swing. We start by calculating π_{10k} and π_{00k}, called p_10 and p_00, respectively.

```
p_10<-predict(mod1, X1, s=mod1$lambda[length(mod1$lambda)],type="response")
p_00<-predict(mod2, X2, s=mod1$lambda[length(mod1$lambda)], type="response")
```

We now create an additional column in our dataframe to house the conditional probabilities and bring back the original structure of including the variables is_swing and is_strike

```
pitch<-cbind(pitch, p_10/(p_10+p_00))
pitch$A<-strike
pitch$B<-swing
colnames(pitch)<-c("is_strike", "is_swing",
                   colnames(pitch)[-c(1,2,length(colnames(pitch)))], "prob")
```

Now that we have the conditional probabilities in our dataframe, we can easily calculate the brier score. Furthermore, to check that the model is better than randomly guessing at each pitch, we can calculate a brier skill score using a reference model where $\hat{\pi}_k = 0.5$, giving a reference brier score of 0.25.

```
brier<-sum((pitch[pitch$is_swing==0,]$is_strike-pitch[pitch$is_swing==0,]$prob)^2)/
    nrow(pitch[pitch$is_swing==0,])
b_ref<-.25
bss<-1-brier/b_ref
```

```
## The model's Brier Score is  0.0704246
```

```
## The model's Brier Skill Score is  0.7183016
```

Additionally, we can use rounded predicted probabilities to classify each pitch as a strike or not and examine the classification error.

```
err<-sum(pitch[pitch$is_swing==0,1]!=round(pitch[pitch$is_swing==0,24]))/
    nrow(pitch[pitch$is_swing==0,])
(right<-round(1-err,2))
```

```
## [1] 0.91
```

This tells us that the model performs much better than random, as we can predict strikes with approximately 91% accuracy.

2

a

Build a projection for the strikeout rate (mean of is_strikeout) for each pitcher in the dataset for the 2025 season (for these purposes assume the 2024 season has ended)

Solution

Theory

For this problem, we will let Y_i be a multinomial random variable representing the number of strikeouts, walks, and other that a pitcher accumulates in a season, i.e.:

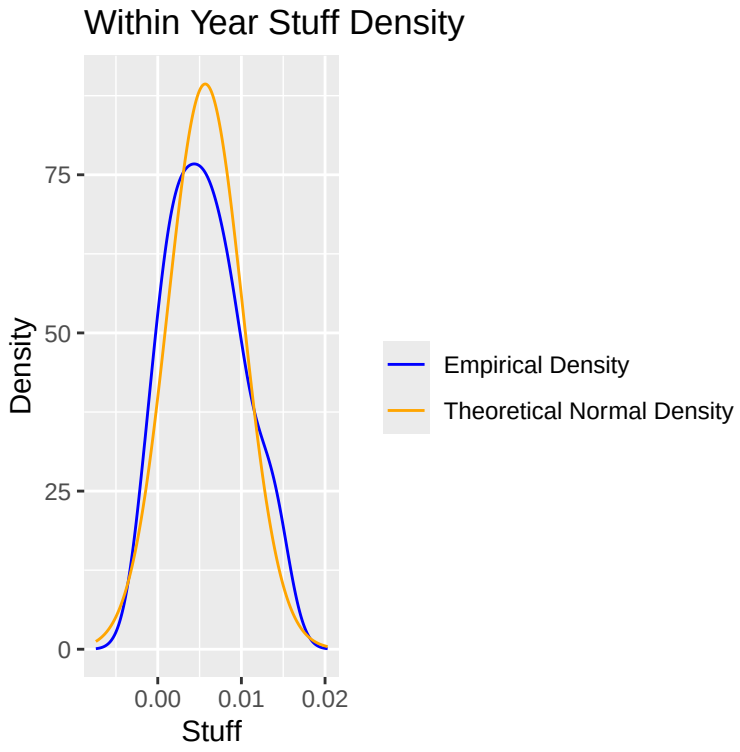
$$Y_i \sim MN_3(n_i, \pi_i)$$

where n_i is the total number of batters faced, $\pi_i = (\pi_{i1}, \pi_{i2}, \pi_{i3})$ is vector of probabilities representing the strikeout rate, walk rate, and other rate for pitcher i. We will use a multicategory logit model of the form

$$\ln(\frac{\pi_{ir}}{1 - \sum_{j \neq r} \pi_{ij}}) = x_i^T \beta_r$$

where x_i contains information about a pitcher's average stuff within a season.

To get an understanding of how a pitcher's average stuff affects the rates over a season, we examine the structure of the stuff metric for a pitcher within a year. To do this, we plot the density estimate for a random pitcher's stuff.



Overlaid on the stuff density estimate (in blue) is the normal approximation to stuff (in orange). This suggests that future stuff values can be predicted using a linear model of the form:

$$\text{Stuff}_i = \mu_i + \beta \text{year} + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2)$$

So, to solve this problem, we will fit our model to the data, estimate each pitchers average stuff for the 2025 season, and use the estimated stuff metric in to predict the strikeout rate in 2025. We now move to the implentation.

Implementation

First, we will import the data and aggregate all the pitcher-hitter match-ups within each year.

```
mat<-matrix(rep(0,8),nrow=1)
for(id in ids){
  for(yr in years){
    temp<-c(yr, id, mean(X[X$pitcher_id==id & X$year==yr,]$age),
            mean(X[X$pitcher_id==id & X$year==yr,]$stuff),
            sum(X[X$pitcher_id==id & X$year==yr,]$is_strikeout),
            sum(X[X$pitcher_id==id & X$year==yr,]$is_walk),
            length(X[X$pitcher_id==id & X$year==yr,]$age)-
              sum(X[X$pitcher_id==id & X$year==yr,]$is_strikeout)-
              sum(X[X$pitcher_id==id & X$year==yr,]$is_walk),
            length(X[X$pitcher_id==id & X$year==yr,]$age))
    mat<-rbind(mat, temp)
  }
}
mat<-data.frame(mat[-1,])
colnames(mat)<-c(noms, "other", "tot")
mat$pitcher_id<-as.factor(mat$pitcher_id)
```

Now that the data is formatted in terms of yearly counts, we can fit the models to find the weights we will use to predict next year's statistics.

```
Xr<-model.matrix(is_strikeout~.-1, mat[,c(2,5)])
Xr<-as.matrix(data.frame(Xr,as.matrix(mat[,c(1,3,4)])))
y<-matrix(c(as.vector(mat$is_strikeout),
            as.vector(mat$is_walk),
            as.vector(mat$other)), ncol=3)
mod<-glmnet(Xr, y, family="multinomial",
            penalty.factor = c(rep(0.0001,200),0,0,0))
```

With the models fit, we will predict stuff for 2025.

```
pred_stuff<-c()
for(id in ids){
  x<-cbind(rep(1, length.out=length(X[X$pitcher_id==id,]$stuff)),
           X[X$pitcher_id==id,]$year)
  beta<-solve(t(x)%*%x)%*%t(x)%*%(X[X$pitcher_id==id, ]$stuff)
  pred_stuff<-c(pred_stuff, t(beta)%*%c(1, 2025))
}
```

Now that we have predicted each pitcher's stuff, we can predict each pitcher's rates for 2025

```
pred_rates<-matrix(rep(0,4), nrow=1)
for(i in 1:length(ids)){
  pitcher<-c(ids[i], 2025, max(mat[mat$pitcher_id==ids[i],]$age)+1, pred_stuff[i])
  pred_rates<-rbind(pred_rates, pitcher)
}
pred_rates<-pred_rates[-1,]
pred_rates<-data.frame(pred_rates)
colnames(pred_rates)<-c("pitcher_id", "year", "age", "stuff")
```

```
pred_rates$pitcher_id<-as.factor(pred_rates$pitcher_id)
Xp<-model.matrix(year~pitcher_id-1, pred_rates)
Xp<-as.matrix(data.frame(Xp,as.matrix(pred_rates[,c(2,3,4)])))
rates<-predict(mod, Xp, s=mod$lambda[length(mod$lambda)], type="response")
```

With the rates predicted, we will look at four pitchers predictions and historical trends. The black lines represent the strikeout rate and the orange lines are the walk rates

To examine the performance of the model, we can use the data from the years 2020 through 2023 to predict strikeout rates in 2024 and compare the error of the model specified above to a model that assigns a 2024 strikeout rate equal to the career average strikeout rate and a linear model using stuff, age, and year.

```
mat1<-mat[mat$year!=2024,]

Xt<-model.matrix(is_strikeout~.-1, mat1[,c(2,5)])
Xt<-as.matrix(data.frame(Xt,as.matrix(mat1[,c(1,3,4)])))
yt<-matrix(c(as.vector(mat1$is_strikeout),
             as.vector(mat1$is_walk),
             as.vector(mat1$other)), ncol=3)
mod1<-glmnet(Xt, yt, family="multinomial",
             penalty.factor = c(rep(0.0001,200),0,0,0))

# predicting stuff
pred_stuff1<-c()
for(id in ids){
  x<-cbind(rep(1, length.out=length(X[X$pitcher_id==id & X$year!=2024,]$stuff)),
           X[X$pitcher_id==id & X$year!=2024,]$year)
```

```
  beta<-solve(t(x)%*%x)%*%t(x)%*%(X[X$pitcher_id==id & X$year!=2024, ]$stuff)
  pred_stuff1<-c(pred_stuff1, t(beta)%*%c(1, 2024))
}

pred_rates1<-matrix(rep(0,4), nrow=1)
for(i in 1:length(ids)){
  pitcher<-c(ids[i], 2024, max(mat1[mat1$pitcher_id==ids[i],]$age)+1, pred_stuff1[i])
  pred_rates1<-rbind(pred_rates1, pitcher)
}
pred_rates1<-pred_rates1[-1,]
pred_rates1<-data.frame(pred_rates1)
colnames(pred_rates1)<-c("pitcher_id", "year", "age", "stuff")
pred_rates1$pitcher_id<-as.factor(pred_rates1$pitcher_id)
Xpt<-model.matrix(year~pitcher_id-1, pred_rates1)
Xpt<-as.matrix(data.frame(Xpt,as.matrix(pred_rates1[,c(2,3,4)])))
rates1<-predict(mod1, Xpt, s=mod1$lambda[length(mod1$lambda)], type="response")
rates1<-data.frame(rates1)
colnames(rates1)<-c("K%", "BB%", "Other%")

bb_ref<-c()
k_ref<-c()
for(id in ids){
  k_ref<-c(k_ref, sum(mat1[mat1$pitcher_id==id,]$is_strikeout)/
             sum(mat1[mat1$pitcher_id==id,]$tot))
}

k_2024<-mat[mat$year==2024,]$is_strikeout/mat[mat$year==2024,]$tot

data_linear<-mat1[,1:4]
data_linear<-cbind(mat1[,5]/mat1[,8],data_linear)
colnames(data_linear)<-c("K_rate", colnames(mat1[,1:4]))

linear<-lm(K_rate~.-1, data=data_linear)
k_ref_linear<-predict(linear, pred_rates1)
```

The error ratio of the multinomial model to the average model is given by

```
mean((k_2024-rates1[,1])^2)/mean((k_ref-k_2024)^2)
```

```
## [1] 0.6806718
```

Since the ratio is less than one, the multinomial model has about 35% lower error and thus is better at predicting strikeout rate than simply using career average strikeout rate. Similarly, we find the ratio of the errors for the multinomial model and the linear model.

```
mean((k_2024-rates1[,1])^2)/mean((k_ref_linear-k_2024)^2)
```

```
## [1] 0.8252281
```

Again, since this ratio is less than one, the multinomial model has about 20% lower error and is thus better at predicting strikeout rate than using a linear model,