Bayesian Analysis in Baseball

Introduction

One of my dreams is to one day be able to apply the toolkit of a data scientist towards domains that I am especially interested. One of which is baseball. Major League Baseball in the United States has changed dramatically over the last decade and it's primarily due to the ubiquitous application of data science concepts towards the game. Having a team of data analysts help formulate game strategy, draft selection, prospect development, and the business side of Major League Baseball is what is now known as the "Moneyball" approach. The concept of Moneyball was popularized by a book and film based on the true story of the Oakland Athletics' unprecedented approach to the game of baseball in the early 2000s, in which they employed the help of statisticians, not regular baseball scouts, to construct their team of what the stats deemed to be "value" picks. This allowed the Oakland Athletics, and many teams since, to be highly competitive in a sport that is often an uphill battle for teams without much capital to acquire superstars. In the 2020 MLB World Series, the Tampa Bay Rays had the **3rd lowest payroll** in baseball and managed to give the winners, the Los Angeles Dodgers, a fairly competitive series with just 28.3 million dollars in total salary. The Dodgers's combined salary in 2020 was $191.2 million. How did they manage to do so? Of course with incredibly talented players, but the Tampa Bay Rays are also regarded as *the* team in the MLB that employs analytics more than any other team–often using unorthodox strategies that go against conventional baseball wisdom, yet work more often than not.

My goal with this project was to explore a very hot button topic in the realm of Major League Baseball analytics: **Bayesian Analysis**. In looking through job postings over the years for MLB Data Scientist jobs (mostly to get an understanding of what to aim for in my skillset), one of the most common traits they desire in statisticians is knowledge of Bayesian concepts.

Bayesian Analysis is seemingly a very deep rabbit hole of a concept, and I could only hope to scratch the surface in the scope of this project. In essence, it revolves around Bayes' Theorem to combine **prior** knowledge with new information/data to come to a new, **posterior probability** (more on this later). In this project I will apply this concept to help determine the potential of a draft prospect to make it to the Major Leagues. More specifically, I will use a model known as a Naive Bayes Classifier to help determine the outcome of a binary class based on a particular player's attributes: "Eventual MLB player" and "Will not make it to the MLB."

Here's a brief description of how a typical career in the MLB works: one goes to college and/or high school, gets drafted by a Major League organization, becomes a **prospect** in the team's "farm system"/minor league teams, and, if good enough, will someday make their debut as a player in the Major Leagues. As we'll see later, the probability of making it to the majors after being drafted is very, very low.

The Data

First and foremost, I owe a data courtesy to several institutions/individuals:

- **Baseball Almanac** - I used this site to pull tables of draftee information
- **Michael Lee (https://www.mikelee.co/projects/)** - Michael created a very wonderful script that makes it very easy to spider through tables of stats sites to pull data on players in the Minor League system–a relatively difficult task compared to finding data on major league players
- **Baseball-Reference** - THE institution for anything on baseball stats. Most of the data was collected from their tables.
- **Lahman's Baseball Archive** - A very popular dataset for basic baseball analysis. There's even a very useful package for R that I did not use.

Starting with Draft Data For this project, I decided to focus on just 4 years of draft classes from 2007-2010. Again, this data was pulled from Baseball Almanac with simple copy/pasting into a csv. This

table forms the basis for the rest of the data by providing keys (player names) on which to join later tables (the rest of the attributes will later be pruned):

```
head(draft)
```

```
## # A tibble: 6 x 6
##   Rd    Number PlayerName    DraftedBy       POS   DraftedFrom
##   <fct>  <int> <chr>         <fct>           <fct> <fct>
## 1 1          1 Bryce Harper  Washington Nat~ OF    College of Southern Nevada
## 2 1          2 Jameson Tail~ Pittsburgh Pir~ RHP   The Woodlands High School (T~
## 3 1          3 Manny Machado Baltimore Orio~ SS    Brito Miami Private School (~
## 4 1          4 Christian Co~ Kansas City Ro~ SS    Cal State Fullerton
## 5 1          5 Drew Pomeranz Cleveland Indi~ LHP   University of Mississippi
## 6 1          6 Barret Loux   Arizona Diamon~ P     Texas A&M University
```

Creating our binary class: This step was easily the most important as I used a join between the above table and a dataset on Major League Baseball player information to create our attribute of focus in this project: major_league_player. This factor is binary with two classes: "Yes" and "No". In other words, "Yes" indicates that the player *has* eventually made it to the Major Leagues and "No" means that player has yet to make after 10 years. The table of interest is the "People" table from Lahman's Baseball Archive. Here is what it looks like:

```
lehman <- as_tibble(read.csv("People.txt"))
lehman <- lehman %>%
  mutate(playerName = paste(nameFirst, nameLast),
         debut = as.Date(debut),
         finalGame = as.Date(finalGame),
         bats = factor(bats),
         throws = factor(throws)) %>%
  select(playerName, playerID, bats, throws, debut, finalGame) %>%
  filter(debut >= "2007-01-01")
```

```
head(lehman)
```

```
## # A tibble: 6 x 6
##   playerName    playerID  bats  throws debut      finalGame
##   <chr>         <chr>     <fct> <fct>  <date>     <date>
## 1 Fernando Abad abadfe01  L     L      2010-07-28 2019-09-28
## 2 Albert Abreu  abreual01 R     R      2020-08-08 2020-08-08
## 3 Bryan Abreu   abreubr01 R     R      2019-07-31 2020-08-04
## 4 José Abreu    abreujo02 R     R      2014-03-31 2020-08-16
## 5 Juan Abreu    abreuju01 R     R      2011-08-29 2011-09-27
## 6 Tony Abreu    abreuto01 B     R      2007-05-22 2014-07-28
```

By default, the People table *only* contains information on Major League Baseball professionals. This trait allowed me to simply conduct a left join onto the previous "Draft" table. Then, using the code below, if the "debut" column for the joined table was NA for a particular player, they would be labeled "No" in the new column "major_league_player" and otherwise "Yes".

```
joined.data <- left_join(draft, lehman, by = c("PlayerName" = "playerName"))
# Add a column to determine if the draftee has made it to the majors
joined.data <- joined.data %>%
  mutate(major_league_player = ifelse(is.na(debut),  "No", "Yes")) %>%
  mutate(major_league_player = factor(major_league_player))

# Shortened table to show new column made
head(joined.data %>% select(PlayerName, major_league_player))
```

```
## # A tibble: 6 x 2
##   PlayerName      major_league_player
##   <chr>           <fct>
## 1 Bryce Harper    Yes
## 2 Jameson Taillon Yes
## 3 Manny Machado   Yes
## 4 Christian Colon Yes
## 5 Drew Pomeranz   Yes
## 6 Barret Loux     No
```

No Pitchers In baseball, one of the positions played is called the "Pitcher". Generally speaking, pitchers are *not* expected to hit the ball, therefore they are mostly evaluated on different metrics. For this project, I decided to focus on non-pitchers and their offensive stats–more on this limitation later.

Adding primitive stats Now, we will add relevant stats for players who play a position that is expected to hit by doing another join. This time, I'm appending the previous table with data scraped from Baseball-Reference. **Note**: players often spend *years* developing their talent before they are deemed ready for the Majors. Adding stats for more than one year across players who spend a varying amount of time in the minor league system would be terribly messy as players typically become *much* better players even a year after their debut in the minor leagues system. Presumably it's beneficial to train the model on a players stats in the first year, too: better to identify the better players early, right?

The stats added in this dataset are relatively primitive: things like "AB" (at bats) and "H" (hits) are relatively meaningless statistics, as it's the proportion of at-bats that result in hits that is more important. I decided to leave this table unmodified first to see if altering the stats later would yield better results. Here is what the data going into our first model looks like.

```
## # A tibble: 6 x 26
##   PlayerName POS   major_league_pl~     G    PA    AB     R     H   X2B   X3B
##   <chr>      <fct> <fct>            <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Bryce Har~ OF    Yes                109   452   387    63   115    24     2
## 2 Manny Mac~ SS    Yes                  9    39    36     3    11     1     1
## 3 Christian~ SS    Yes                 60   271   245    38    68    12     2
## 4 Delino De~ CF    Yes                 18    83    76    14    22     6     1
## 5 Michael C~ CF    Yes                 30   130   109    21    29    10     2
## 6 Yasmani G~ C     Yes                  8    33    28     4     8     1     0
## # ... with 16 more variables: HR <dbl>, RBI <dbl>, SB <dbl>, CS <dbl>,
## #   BB <dbl>, SO <dbl>, BA <dbl>, OBP <dbl>, SLG <dbl>, OPS <dbl>, TB <dbl>,
## #   GDP <dbl>, HBP <dbl>, SH <dbl>, SF <dbl>, IBB <dbl>
```

Adding more advanced statistics As mentioned, the stats above are primitive compared to the stats that are commonly accepted as key indicators of player performance in today's game. I only included a handful of *many* stats used by professional analysts, as these were a lot less complex as other statistics have formulas that weigh attributes as detailed as altitude in a particular baseball park. I decided to focus on the following stats, which are all derived from manipulation of the above primitive stats:

1. **Strike Percentage** - This is simply one of many measures of a player's "plate discipline" and is the proportion of at-bats that result in a strikeout, which is markedly negative outcome of an at-bat.
2. **ISO** - This is a measure of a player's "raw power". It is derived by subtracting a players "Slugging Percentage" by their "Batting Average", essentially resulting in a player's ability to hit for extra bases. Hitting for power (i.e. "swinging for the fences" or trying to always hit the ball out of the park) is increasingly becoming a trend in today's game, largely due to analytics, so I thought it might be a good statistic to include.

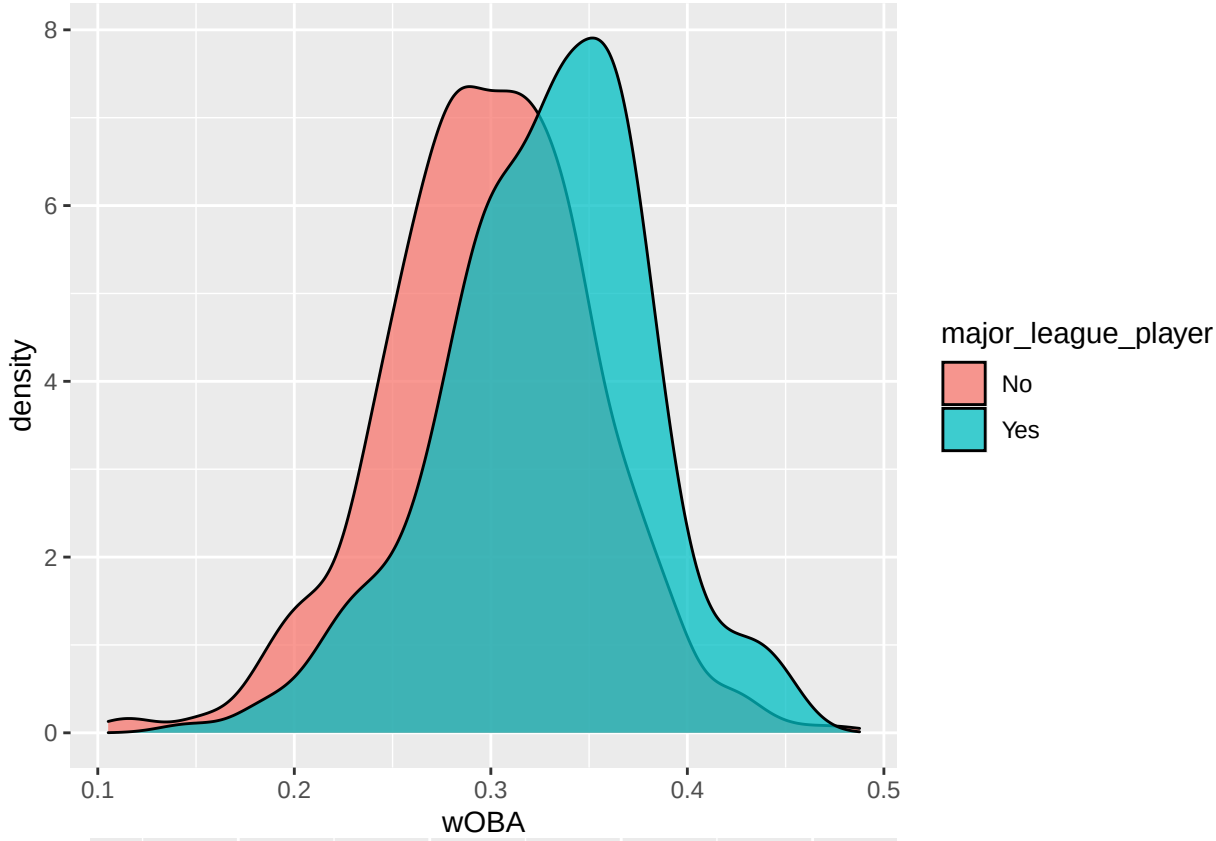
3. **wOBA** - Weighted On-Base Average. This is a slightly more sophisticated stat than batting average, which is simply the proportion of at-bats that result in a hit, as it adds a weight to each outcome of a hit: a double, which is worth two bases, is worth more than a single, and a home run has a higher weight than any other outcome. Again, the home run is increasingly becoming the stat to pad in recent years, so hopefully this stat will prove more useful than batting average for our model.

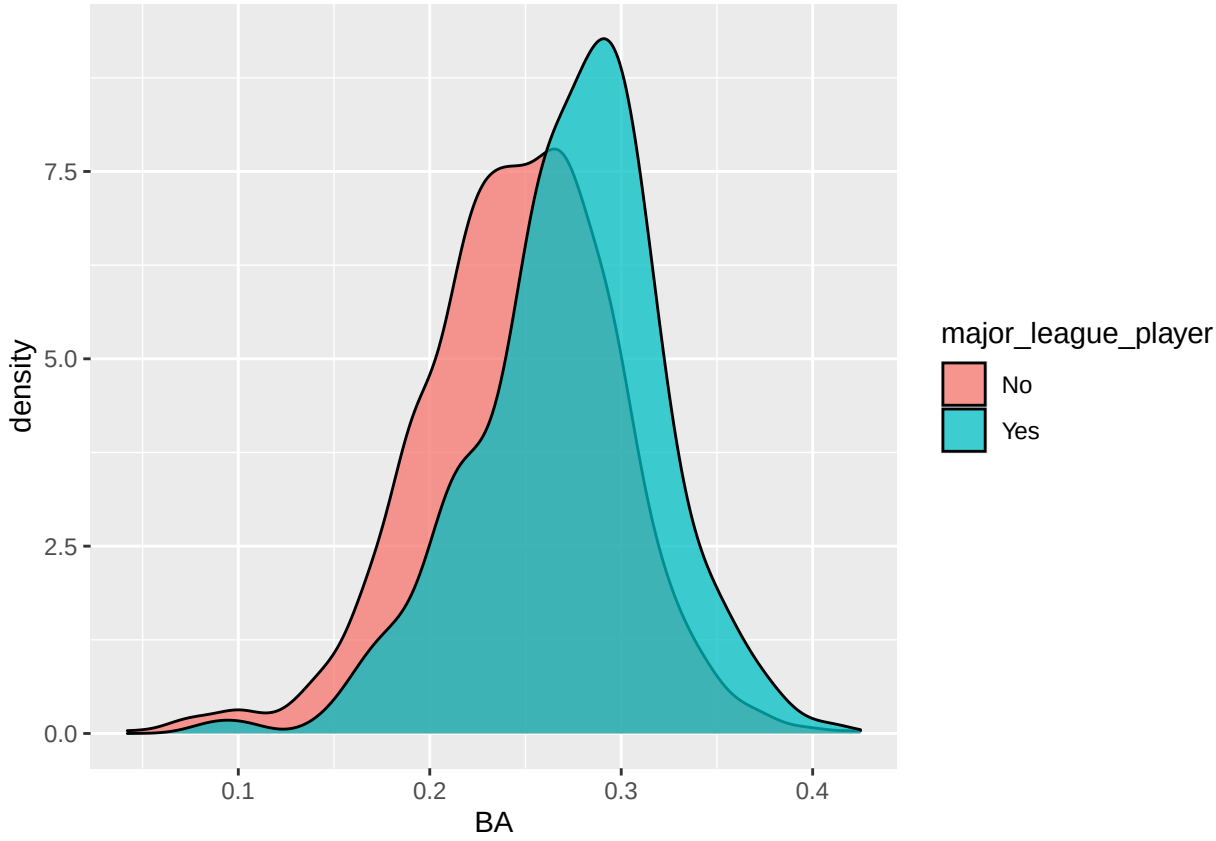
Finally, this is what the second dataset for our second model looks like:

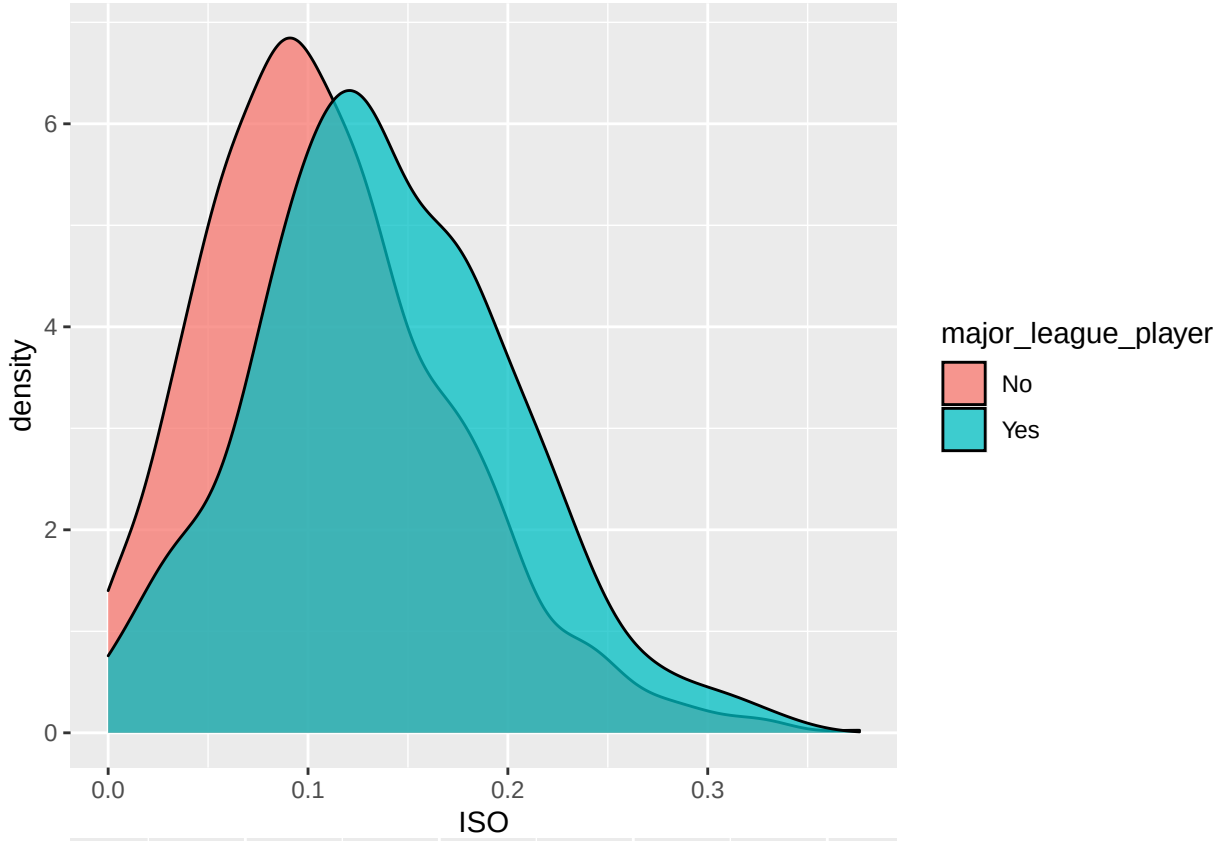
```
##         PlayerName major_league_player POS   ISO    BA      wOBA
## 1     Bryce Harper                  Yes  OF 0.204 0.297 0.3737005
## 2    Manny Machado                  Yes  SS 0.166 0.306 0.3516923
## 3  Christian Colon                  Yes  SS 0.102 0.278 0.3024286
## 4 Delino DeShields                  Yes  CF 0.106 0.289 0.3135060
## 5   Michael Choice                  Yes  CF 0.321 0.266 0.3930620
## 6  Yasmani Grandal                  Yes   C 0.035 0.286 0.3268485
##   StrikePercentage
## 1           0.2248
## 2           0.0833
## 3           0.1347
## 4           0.2632
## 5           0.4128
## 6           0.1429
```

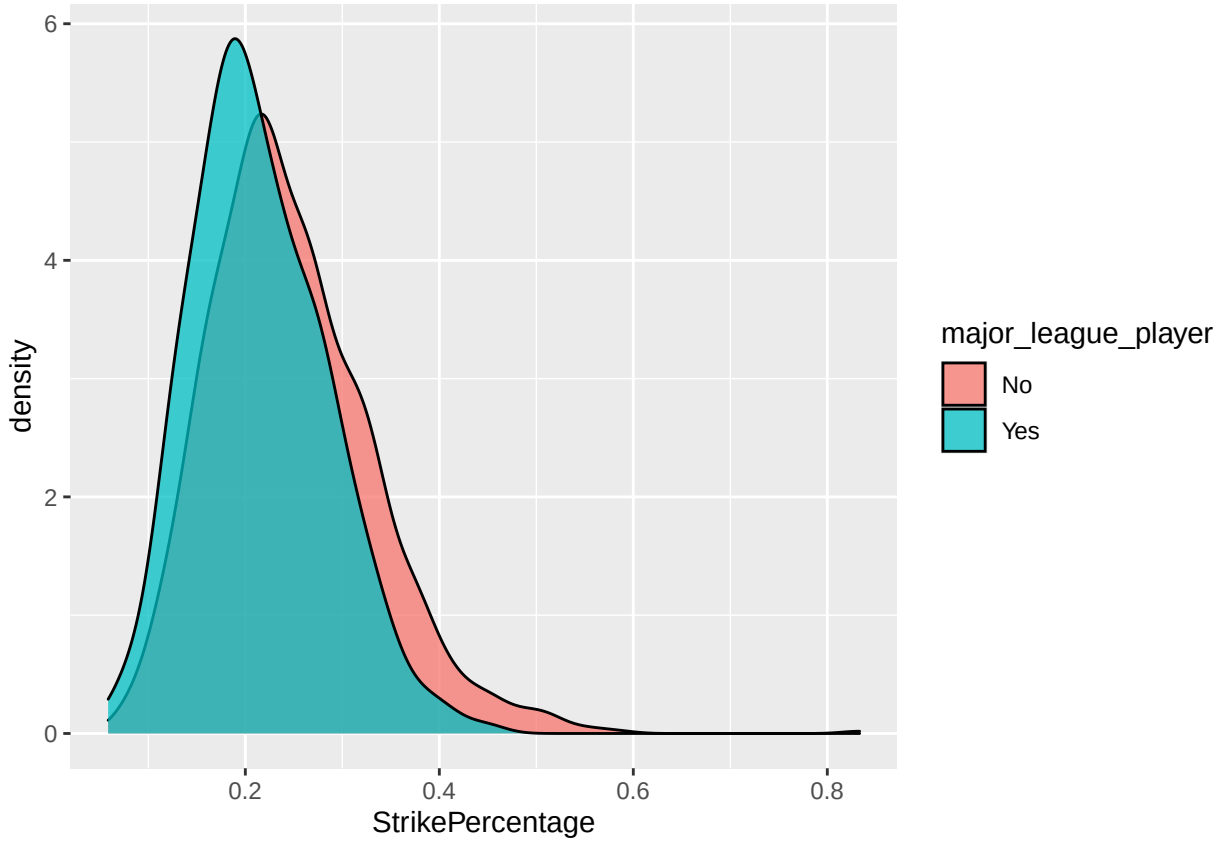
Exploratory Data Visualization

I realize the stats above might be a bit intimidating to anyone who might not be too familiar with baseball, so here are some exploratory plots to give a better idea of what those advanced stats actually mean in terms of how they differ between players who do *not* make it to the majors, and those who do. Based on their distributions, a major league caliber player generally has a higher ISO, a higher BA, a higher wOBA, and a lower Strike Percentage. Note the normality of the graphs, as this will be important when we discuss the model itself.


density
8
6
4
2
0
0.1
0.2
0.3
0.4
0.5
wOBA
major_league_player
No
Yes


density
7.5
5.0
2.5
0.0
0.1
0.2
0.3
0.4
BA
major_league_player
No
Yes


density
6
4
2
0
0.0
0.1
0.2
0.3
ISO
major_league_player
No
Yes


density
6
4
2
0
0.2
0.4
0.6
0.8
StrikePercentage
major_league_player
No
Yes

Naive Bayes Classifier

Bayes' formula is a simple formula using conditional probability to predict an event A given B has occurred. First, we must define a *prior*, that is, the **probability of event A happening** across a population, with no additional information given about that particular observation. In this case, the **prior** will be defined as A_{mlb}, i.e. the probability that a player in the MLB draft will make it to the Major Leagues (with A^C_{mlb} = the probability a player *doesn't* make it to the MLB). We can count this using the following code:

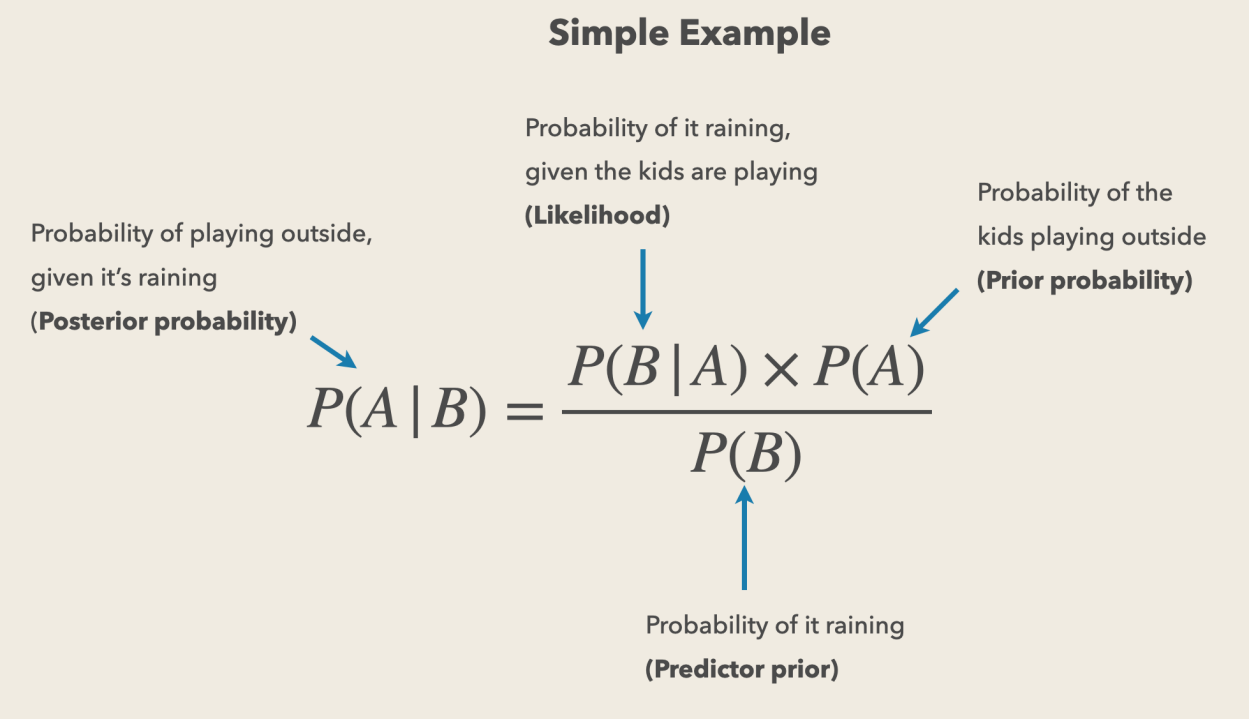
```
complete %>%
  summarize(
    `Percentage of draftees that make it to the majors within 10 years` = sum(major_league_player == "Y
```

```
## # A tibble: 1 x 1
##   `Percentage of draftees that make it to the majors within 10 years`
##                                                                 <dbl>
## 1                                                               0.193
```

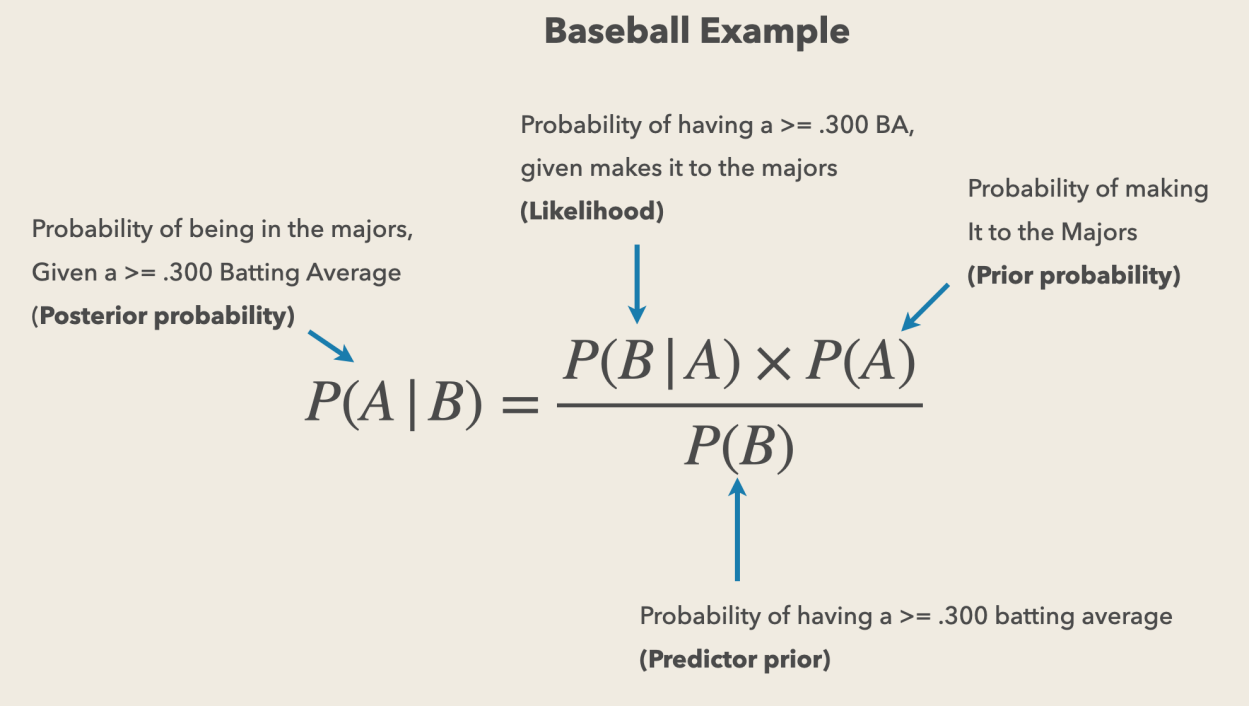
So, our prior probability is $A_{mlb} \approx .1933$, meaning just under 20% of players drafted make it to the major leagues within 10 years after being drafted. However, we using Bayes' Theorem, in theory we can *more accurately* find the probability a particular draftee will make it to the majors using Bayes' Theorem:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Here is a very simple example of how the formula applies to classifying problems. Consider a simple model that determines whether or not a group of children will play outside



And now, here is that same graphic but an example of how an attribute like, for example, a Batting Average $> .300$ might alter the probability of a player making it to the major leagues:

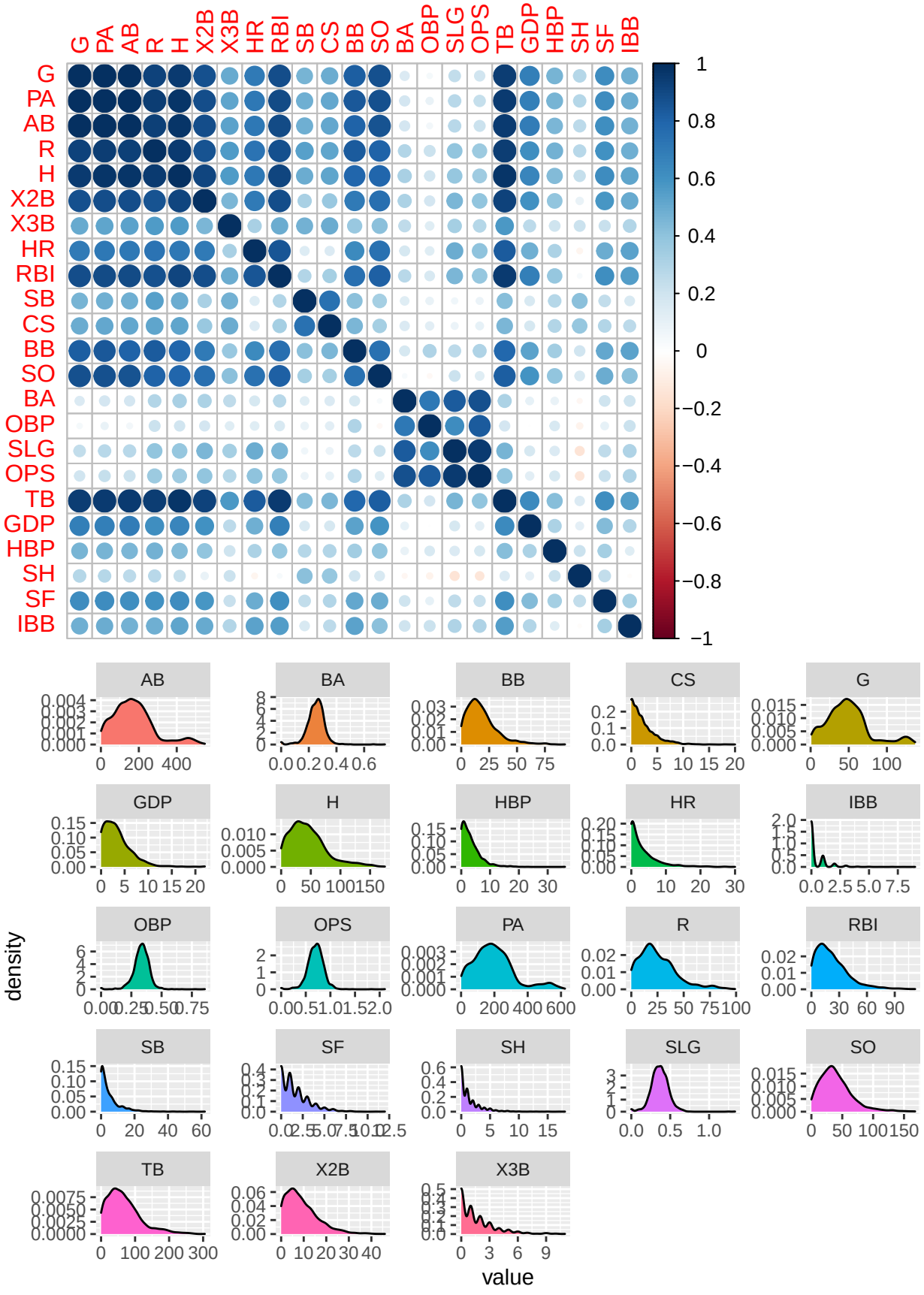


The goal of implementing a Naive Bayes Classifier is to factor in all those those predictors, $P(B)$, that maximize the algorithm's ability to discern players with Major League potential. If the model performs well, it can then be applied to *new* players that have been drafted. We'll test this by applying the model twice: once to the primitive stats table, and once to the advanced stats table. For both, the datasets will be split into a training and a testing set.

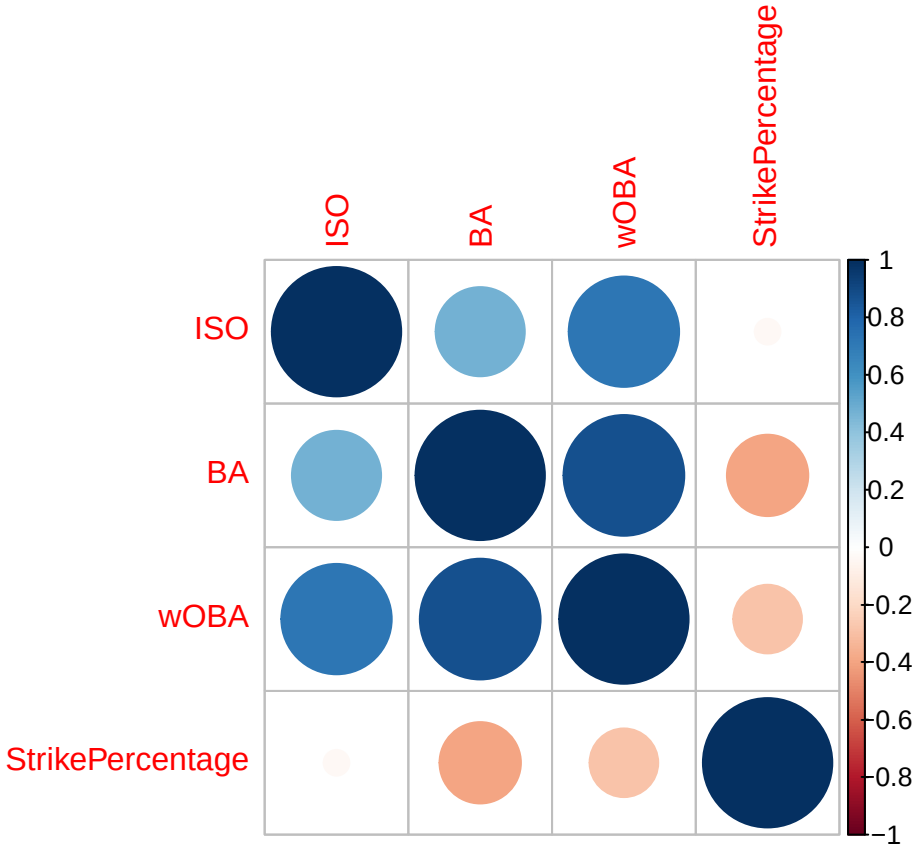
Note: Two key assumptions are made in a Naive Bayes Classifier:

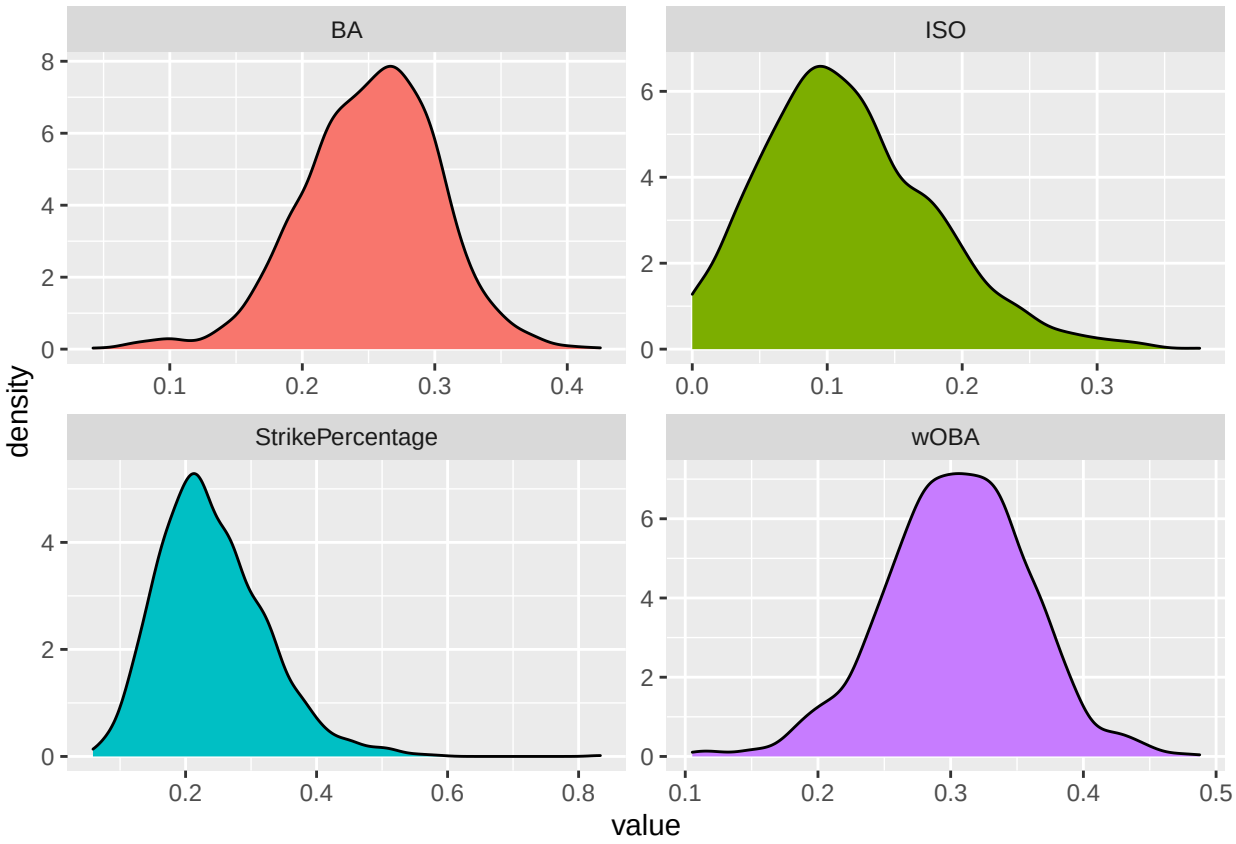
1. The predictors (the Bs) are assumed to be conditionally independent of one another
2. Continuous predictors are normally distributed.

As we saw above for the advanced stats table, many of those predictors *are* normally distributed. However, in the primitive stats table there are *many* predictors that are neither normally distributed nor conditionally independent. The first graph below shows a correlation table across all numeric variables in the primitive dataset (darker blue/red implies heavier correlation). For the second, simply note the shapes of the density plots on numeric stats aren't particularly normally distributed either.


G PA AB R H X2B X3B HR RBI SB CS BB SO BA OBP SLG OPS TB GDP HBP SH SF IBB
1 0.8 0.6 0.4 0.2 0 −0.2 −0.4 −0.6 −0.8 −1
AB BA BB CS G
GDP H HBP HR IBB
OBP OPS PA R RBI
SB SF SH SLG SO
TB X2B X3B
density
value

And again, the variables for the second model are normally distributed, however, still heavily correlated with one another and not conditionally independent:





This then begs the question: **why even bother with a Naive Bayes Classifier?** First, consider how we would compute a posterior probability across all predictors: our response variable is binary–either a player makes it to the majors, or they don't– this gives us 2 possibilities, which would be raised to the power of 23 (how many possible predictors we have in the data set). This results in this many probabilities needed to be computed:

```
2**23
```

```
## [1] 8388608
```

However, by assuming conditional independence and normality, only 2 x 23 probabilities are needed. A much less computationally expensive task, and still a somewhat powerful predictor. More importantly, as it applies to the domain of baseball, *we don't particularly care about the exact posterior probability.* Often, we simply care about whether or not a *specific* player is more likely to be a major league-caliber player than not (i.e. >50%) to determine whether it is worthwhile to continue developing their talent.

Model 1: Primitive Stats

I used the caret package to perform the algorithm for a Naive Bayes Classifier using the train() function and its "nb" method (naive bayes). There are a few tuning parameters like fL and Laplace smoothing which honestly somewhat fell over my head, but based on other papers I've seen on Naive Bayes, proper tuning of them can yield fairly decent results.

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##            Reference
## Prediction   No  Yes
```

```
##        No  66.5 13.2
##        Yes 14.2  6.1
##
##  Accuracy (average) : 0.7256
```

And now, using the predict() function on the testing data

```
pred <- predict(nb1, newdata = test)
confusionMatrix(pred, test$major_league_player)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  351  61
##        Yes  74  41
##
##                Accuracy : 0.7438
##                  95% CI : (0.7043, 0.7806)
##     No Information Rate : 0.8065
##     P-Value [Acc > NIR] : 0.9998
##
##                   Kappa : 0.2173
##
##  Mcnemar's Test P-Value : 0.3017
##
##             Sensitivity : 0.8259
##             Specificity : 0.4020
##          Pos Pred Value : 0.8519
##          Neg Pred Value : 0.3565
##              Prevalence : 0.8065
##          Detection Rate : 0.6660
##    Detection Prevalence : 0.7818
##       Balanced Accuracy : 0.6139
##
##        'Positive' Class : No
##
```

Not *terrible* results to say we can predict a player's career trajectory with a ~74% accuracy, but evaluating the Cohen's Kappa value shows that the model performed rather poorly. Let's see if we can do a little better.

Model 2: Advanced Baseball Stats

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   No  Yes
##        No  73.1 13.3
##        Yes  7.7  6.0
##
##  Accuracy (average) : 0.7906
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  No Yes
##        No  313  63
##        Yes  47  22
##
##                Accuracy : 0.7528
##                  95% CI : (0.71, 0.7922)
##     No Information Rate : 0.809
##     P-Value [Acc > NIR] : 0.9986
##
##                   Kappa : 0.1382
##
##  Mcnemar's Test P-Value : 0.1527
##
##             Sensitivity : 0.8694
##             Specificity : 0.2588
##          Pos Pred Value : 0.8324
##          Neg Pred Value : 0.3188
##              Prevalence : 0.8090
##          Detection Rate : 0.7034
##    Detection Prevalence : 0.8449
##       Balanced Accuracy : 0.5641
##
##        'Positive' Class : No
##
```

An improvement, but still not particularly good.

Conclusion

A win for the frequentists. The biggest problem is that if we were to train a toddler to simply shake their head "no" at every baseball prospect they saw, in total they'd have better predictive power than the model performed here. Here's a reminder on what the priori probability was:

```
## # A tibble: 1 x 1
##   `Percentage of draftees that make it to the majors within 10 years`
##                                                                 <dbl>
## 1                                                               0.193
```

This means that the toddler would have predicted their career trajectory with a >80% accuracy.

I'm not exactly sure *why* the model performed poorly, but I have some ideas. First, everything I researched was quick to point out that a Naive Bayes Classifier is far from the first choice for prediction/classifying problems. Its strength comes from its ease of implementation and computationally inexpensive nature.

It's also worth noting that a baseball organization obviously cannot just say "No" to every prospect they pick up–this would *truly* be a case of "paralysis by analysis." At the end of the day, as noted above, we aren't particularly interested in the exact posterior probability calculated using Bayes' theorem: we might simply want to know if a player is more likely than not to make it to the Majors. For this, a Naive Bayes Classifier could still be useful to some degree.

I speculate that the model performed poorly for a few reasons. For one, the graphs shown above tell us that the data's predictors are *far* from conditionally independent. Violating this assumption of the Naive Bayes Classifier likely came at the cost of predictive power. However, many data sets I've seen through researching this method show that despite violating the assumptions of normality in continuous variables and conditional independence among predictors, the model can still perform very well. There are dozens of articles online showing >95% accuracy in classifying the Iris dataset, for example.

This leads me to believe that the poor performance is largely due to my selection of predictors and the constraints I held on the dataset for simplicity's sake. Again, I only included data from a prospect's *first* year of playing in the minors after being drafted–perhaps an average across their first few years, or improvement from one year to the next would be better indicators. Additionally, a selection of more conditionally independent predictors would likely yield better predictive power Keep in mind that baseball isn't all about offensive power! Many players are more inclined towards defense, and their abilities on both sides of the field are what makes them Major League caliber players.

Of course, a better understanding of the tuning parameters could prove helpful as well.

For now, I'll let the frequentists have their laugh.