Exploring Multiple Methods of Classification for the Result of a Baseball Pitch

Alex Chandy

November 14, 2023

1 Abstract

Attempting to predict the result of a baseball pitch can be extremely beneficial to baseball pitchers from a strategic standpoint. Given the data at our hands nowadays, classification algorithms can be tuned to a batter's specific tendencies, and allow pitchers to predict what will happen if they throw a certain pitch. In this paper, 3 different classification algorithms were explored and evaluated: K-Nearest Neighbors, Gaussian Naive Bayes, and Logistic Regression. PCA was used to reduce dimensionality, and storage space, for future applications of this project. Initial results were respectable, with 75% accuracy from the K-Nearest Neighbors and Naive Bayes algorithm, and after PCA, there was very little accuracy loss. Althoughh Logistic Regression performed poorly initially, when the RBF kernel trick was applied, there was an accuracy boost that allowed it to reach the 75% mark set by the other two algorithms. This sets up possible avenues for further research on this topic, as different kernels could be experimented with.
Code: https://github.com/alexchandy13/pitch-classify

2 Introduction

2.1 Background

The intersection of data and sports has become increasingly prevalent recently, especially so for the game of baseball. Compared to most other sports, baseball is a particularly appropriate platform for data-driven prediction and classification. Although it is a team sport, the crux of the game is very individual, simply a batter versus a pitcher. This makes it relatively easy to make predictions. Due to the fairly closed environment, almost all variables can be quantified.

With the advent of tracking technology in baseball (StatCast), almost every feature of a baseball pitch is now quantified and available to the public. Variables such as pitch velocity, pitch movement, and pitch spin rate are all major factors in the result of a pitch. In this paper, when using the phrase "pitch result" or "result of a pitch", we will be referring to what **the batter** does, after a certain pitch has been thrown.

The core of this project is to use variables such as those listed above in order to predict the pitch result. Of course, there are many ways to classify the result of a pitch, but for this project's case, we will simply classify the results as a "Swing" or a "Take" (Batter does not swing). These two classes cover the entire sample space of a pitch result. The majority of this paper will cover this binary classification problem, but a small subsection will be dedicated to exploring this as a multiclass classification problem.

With regards to the binary classification problem, the decision of a batter to swing or not is dependent on many different features of a pitch. This paper will explore how 8 different features of a pitch affect the result after the pitch has been thrown.

2.2 Motivation and Application

For a baseball pitcher, knowing what pitch to throw, where and when to throw it can provide tremendous benefit in strategically trying to get a batter out. This project aims to provide a pitcher with an idea of what a batter will do, given a certain pitch.

Every time a pitcher throws a pitch, it is subject to variations in velocity, movement, spin, and so on. However, given the average for each of these features, these averages can be used as inputs to a function which outputs the pitch result. Of course, the pitch result is highly dependent on tendencies of the batter who is being pitched to. Thus, for each batter, a new function must be made.

We propose a system in which a classification model is derived for each batter in professional baseball. This model will be trained on a dataset of every pitch thrown to the batter in the last 3 seasons, labeled by the pitch result (”Swing” or ”Take”). Given this model, the attributes of a pitch can be input to yield a result.

3 Methodology

3.1 Dataset and Collection

The initial pitch features chosen to be used for classification were:

- Pitch Velocity (Miles per hour)
- Count (Balls and Strikes) - Affects mindset of the batter
- Pitch Movement (Feet) - Measured with 2 dimensions/features
- Pitch Location (Feet) - Measured with 2 dimensions/features
- Pitch Spin Rate (Revolutions per minute) - Affects deceleration and path shape of pitch

The dataset was scraped from Baseball Savant [1], which is Major League Baseball’s database of StatCast tracked data. Using BaseballR [2], an R package written by Bill Petti for easily scraping data from baseball related databases.

As aforementioned, in a fully functional system, a classifier would be built for each player. For the purposes of this paper, only one player-specific classifier will be built and evaluated. I chose Rafael Devers, as he has a large sample (6,237 pitches) from 2019-2021, and a relatively balanced distribution (3,397 Swings, 2,840 Takes).

So, the dataset we are working with consists of 6,237 training samples, each with 8 attributes, and a binary label.

3.2 Initial Experiment

The nature of this problem is greatly influenced by spatial factors, as the location and movement of a pitch are strong contributors to the swing decision. We can see in Figure 1 the class of a point, based on just the x and z (horizontal and vertical) location of a pitch when it reaches the batter. Clearly, there is a pattern of concentric circles, with there being a high concentration of ”Swing” labeled data in the center, surrounded by ”Take” labeled data on the edges. The clusters of points are clearly not linearly separable on a 2-dimensional basis, and likely not linearly separable even when all features are included. Because of the non-linear nature of this problem, a K-Nearest Neighbors and Naive Bayes Classifier were chosen.

K-Nearest Neighbors is appropriate for this task, because it does not require a linearly separable dataset [3]. Classification is simply determined by the data closest to the input point. In this case, a point on the edge of the distribution would most likely be closer to ”Take” labeled data, and would be classified as such. Difficulties could arise when attempting to classify points on the edge of the decision

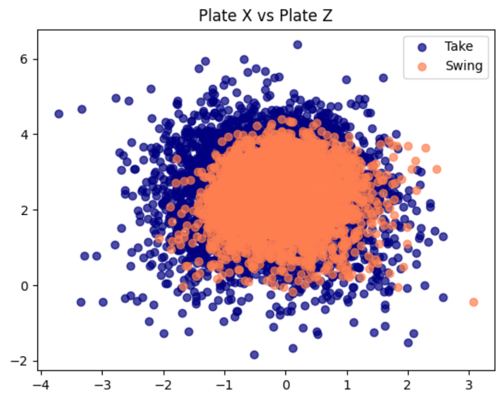

Figure 1: Pitch Result Displayed by Location

boundary. As can be seen in figure 1, there is not much separation towards the edge of the center cluster of "Swing" labeled data. Obviously, this is the case when only 2 features are being observed. Since the classifier will have access to 8 features, the decision boundary should become more distinct.

The Naive Bayes Classifier is also capable of creating a non-linear decision boundary. Depending on the standard deviation of the features of each class, different non-linear decision surfaces can be generated [4], so it should be feasible for Naive Bayes to create an appropriate decision boundary for this problem. The Gaussian Naive Bayes Classifier was chosen, as the distribution of all features is approximately Gaussian.

Additionally, a Logistic Regression classifier will be evaluated as a pseudo-standard. Both the K-Nearest Neighbors and Naive Bayes classifiers should perform much better than the Logistic Regression classifier. This is because Logistic Regression performs better on data that is linearly separable [5], as it is a linear classifier.

3.3 Dimensionality Reduction with PCA

As stated above, 8 features are being explored in this problem. This is a relatively high dimensionality, and it could be valuable to decrease the number of features used in the model. The scale of this application is thousands of times larger than what is explored in this paper, since a classifier must be derived for each professional baseball player, as mentioned. Although the model sizes may be small on the individual scale, the storage space dramatically increases when used in a real world application.

Dimensionality reduction can decrease the complexity and storage space of a model, and Principal Component Analysis (PCA) is a popular method for achieving this. We can determine the new dimensionality k, the PCA algorithm will calculate the directions of maximum variance, and give the top k components in terms of explained variance. By doing this, we can map a dataset of dimensionality d to one of dimensionality k [6]. We can visualize how much variance is explained by a component by plotting the explained variance ratio, as shown in Figure 2. For this problem, we see that by reducing the dimensionality from 8 to 6 components, we lose very little explained variance, as the 6 components cumulatively explain nearly 100% of the variance.

After performing PCA for 6 components, we can visualize the pairwise distributions of the data for each component in Figure 3. We can see a similar pattern for almost all the pairwise comparisons. There is a clear clustering of "Swing" labeled data towards the center of each distribution, with "Take" labeled data on either side of the cluster. This is comparable to the clustering shown in Figure 1, as "Swing" labeled data is close to the center, and "Take" labeled data on the edges.

Using the 6 components from PCA instead of the 8 original features, we train the same K-Nearest Neighbors, Naive Bayes, and Logistic Regression classifiers on the PCA transformed dataset. Since

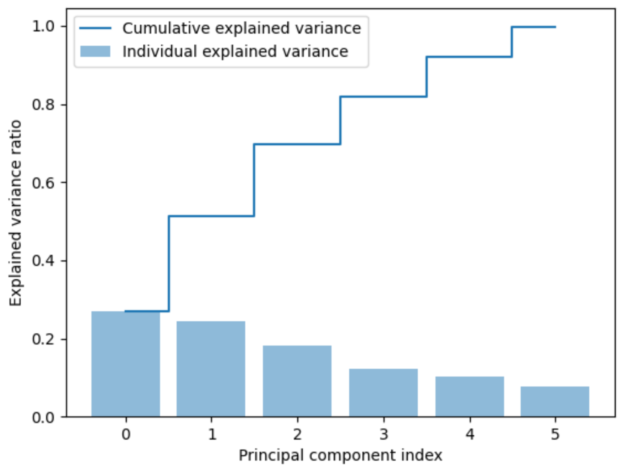

Figure 2: Explained Variance Plot

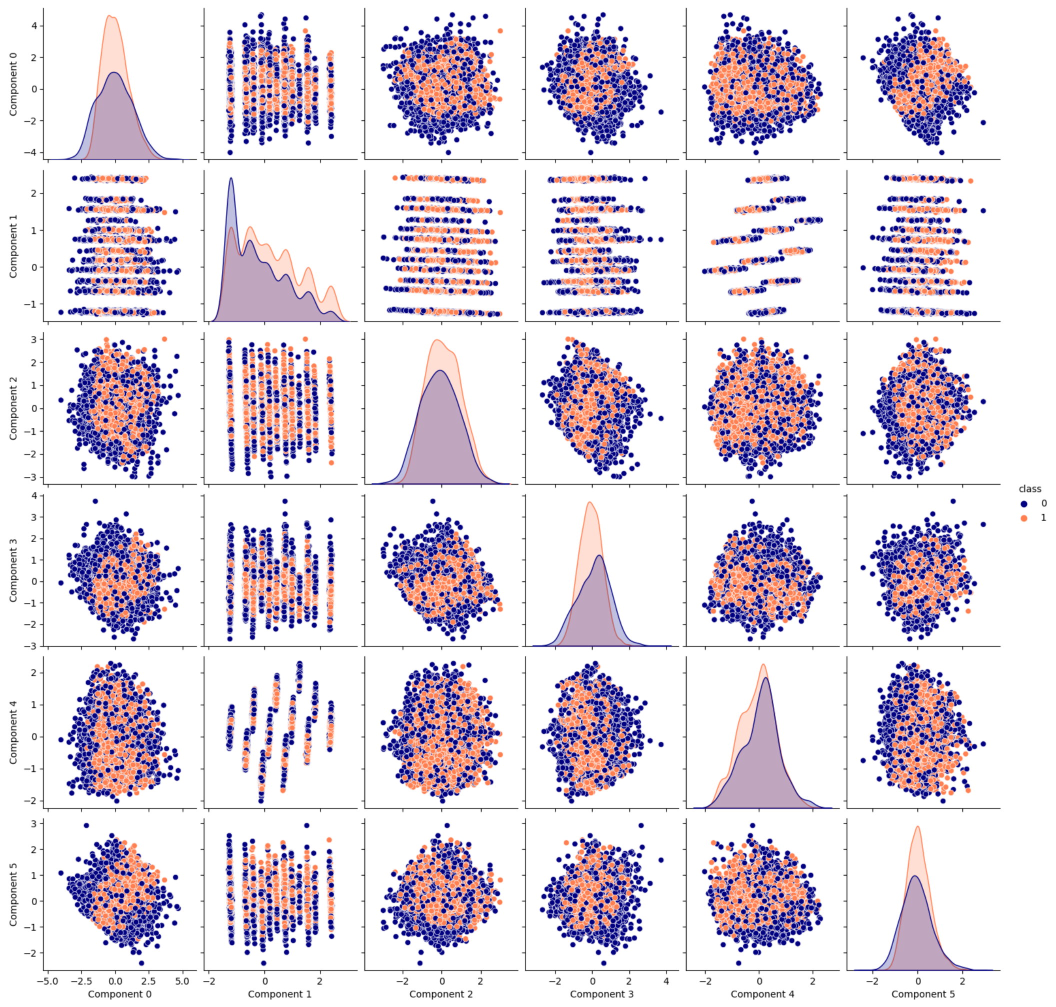

Figure 3: PCA Component Pairwise Distributions

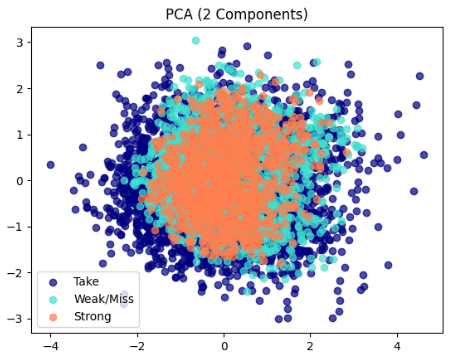

Figure 4: Class Distribution - 2 Component PCA

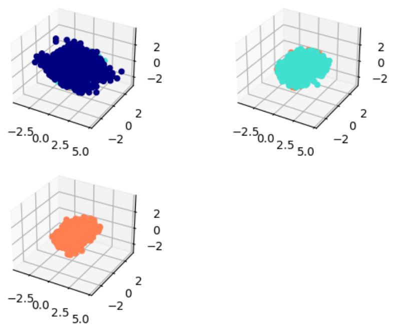

Figure 5: Class Distribution - 3 Component PCA

the explained variance after PCA is effectively the same, there should be little to no accuracy loss.

3.4 Kernelized Logistic Regression

In order to boost the performance of the Logistic Regression classifier, the technique of a kernel trick can be used. By using a kernel function, data that is not linearly separable in it's original feature space can be projected into a space in which it is linearly separable. Essentially, it can turn a linear model such as Logistic Regression into a non-linear classifier, given the correct kernel function [7].

For this problem, the radial basis function (RBF) kernel was chosen due to the spatial representation of the data. The RBF kernel can be used for "donut" shaped distributions [8][9], in which clusters take a circular shape within another circular cluster. Another description would be concentric circles. This may not be the optimal kernel function for this particular data distribution, but for the purpose of this paper, it will suffice.

3.5 Multiclass Classification

As explained before, there are multiple ways to classify a pitch result. For the bulk of this paper, we have explored the classification as a binary problem. However, the result of a pitch can be cut into 3 classes as opposed to 2 (In reality, there are even more ways to classify the result).

We can classify a pitch result as a "Take", "Weak Contact or Miss" (Batter swung and made marginal contact with the ball, or completely missed), or "Strong contact" (Batter swung and made strong contact with the ball). This greatly increases the complexity of the problem, due to the extreme randomness that occurs once a batter has swung. In baseball, the same exact pitch can, and often does, yield vastly different results if a batter chooses to swing.

After running PCA to decompose the dimensionality to 2 components, we can view this problem in a 2D setting (Figure 4). We see almost the exact same distribution as in Figure 1, when viewing the binary class distribution given the x and z coordinates of a pitch. However, the appearance of the "Weak/Miss" labelled points is very marginal, and this is because the cluster of these points is "underneath" the cluster of "Strong" labelled points. In reality, this gives a very homogeneous mixture of points towards the center of the distribution. The classifiers will thus find it difficult to discern points in the center of the cluster as "Weak/Miss" or "Strong". This is better illustrated in Figure 5, where we can see the clusters for "Weak/Miss" and "Strong" are about the same size and spatial distribution.

4 Results

4.1 Initial Results (Without PCA)

Before performing the PCA, we can see how the classifiers performed using precision, recall, and the F-1 Score. The problem is binary, and relatively balanced, so accuracy is also a good measure. The accuracies for K-Nearest Neighbors, Naive Bayes, and Logistic Regression were 0.747, 0.747, and 0.635 respectively.

Table 1: K-Nearest Neighbors

	Take	Swing
Precision	0.749	0.745
Recall	0.675	0.808
F1-Score	0.709	0.775

Table 2: Gaussian Naive Bayes

	Take	Swing
Precision	0.770	0.733
Recall	0.641	0.838
F1-Score	0.699	0.782

Table 3: Logistic Regression

	Take	Swing
Precision	0.619	0.645
Recall	0.532	0.723
F1-Score	0.572	0.682

4.2 Results after PCA

Below are the results after PCA was performed, and the classifiers were trained on the PCA transformed dataset. Because very little of the explained variance was lost, we expected to see very little accuracy loss. The respective accuracies for K-Nearest Neighbors, Gaussian Naive Bayes, and Logistic Regression were 0.750, 0.714, and 0.647.

Table 4: K-Nearest Neighbors

	Take	Swing
Precision	0.749	0.745
Recall	0.675	0.808
F1-Score	0.709	0.775

Table 5: Gaussian Naive Bayes

	Take	Swing
Precision	0.770	0.733
Recall	0.641	0.838
F1-Score	0.699	0.782

Table 6: Logistic Regression

	Take	Swing
Precision	0.619	0.645
Recall	0.532	0.723
F1-Score	0.572	0.682

4.3 Storage Space Reduction

As mentioned, an enticing benefit of doing PCA was to reduce model size in order to reduce storage space, for when this project is scaled up on an order of thousands. The model file size was measured for each classifier, before and after the PCA transformation.

Table 7: File Size Reduction

Classifier File Size	No PCA (Bytes)	PCA (Bytes)
K-Nearest Neighbors	715684	557854
Gaussian Naive Bayes	1010	940
Logistic Regression	949	927

Table 8: Logistic Regression

	Take	Swing
Precision	0.759	0.715
Recall	0.644	0.814
F1-Score	0.697	0.761

4.4 Results after Kernelization

An RBF kernel was applied to the Logistic Regression classifier in order to allow it to classify non-linearly separable data. This classifier was also trained on the PCA transformed data. The accuracy was boosted from 0.647 without the kernelization to 0.733, which is as expected. We can see in the above Table 8, the precision, recall, and F1-Score

5 Discussion

5.1 Evaluation of Results

The initial results were fairly encouraging, at least for the K-Nearest Neighbors and Naive Bayes classifiers. Because the problem is binary, and the dataset is relatively balanced, the accuracy is a fair measure of performance, and both classifiers had a solid accuracy of 0.747. While this is nowhere close to perfect, for a preliminary, not super-tuned model, this can provide some value. We also see that the precision, recall, and F1-Scores (Tables 1 and 2) all hover around the 0.6-0.8 range, which indicates that the accuracy is most likely an appropriate measure [6].

As expected, the Logistic Regression classifier performed much worse compared to the other two classifiers. The accuracy of 0.635 was more than 10% lower, and the precision, recall and F1-Scores were in the 0.5-0.7 range (Table 3). This is providing very little informational value, as a classifier with no knowledge would theoretically achieve an accuracy of 0.5.

However, once we apply the RBF kernel (after PCA), the kernelized logistic regression yielded much more promising results, as the accuracy jumped 8.6 points from 0.647 to 0.733. This is very similar to the performance of the other two classification algorithms, and the same is true for the precision, recall, and F1-scores (Table 8). Again, this is completely as expected, Logistic Regression as a linear classifier was completely unsuited for this problem, but once the kernel trick was applied, it was more appropriate.

The most important results were after the PCA had be done. Because the cumulative explained variance of the 6 components was almost 100%, it was expected that there would be very little accuracy loss, and there was. In fact, for the k-Nearest Neighbors algorithm, the accuracy increased marginally from 0.747 to 0.750. The accuracy for Gaussian Naive Bayes did decrease by a small amount from 0.747 to 0.714, but for both algorithms, the precision, recall, and F1-score remained stable in the 0.6 to 0.8 range (Tables 4 and 5). Logistic Regression also actually saw a small increase in accuracy, from 0.635 to 0.647, which is rather surprising, as the dimensionality reduction would reduce the chance of the data being linearly separable.

The sizes of the models saw a reduction, but it was not extremely significant for Naive Bayes and Logistic Regression. As can be seen in Table 7, the Naive Bayes classifier after PCA was only 7% smaller than the original classifier, and the Logistic Regression classifier saw an even smaller reduction of 2%. One possible explanation for this is that the dimensionality reduction simply wasn't enough, as it only went from 8 features to 6 components. However, the reduction for K-Nearest Neighbors was massive in comparison, with a 22% reduction in file size. Of course, the file size for K-Nearest Neighbors was still hundreds of times larger than the other two algorithms. It is possible that the great reduction is because K-Nearest Neighbors does not truly create a model, but simply measures distances between the test input and the training data. So, when dimensionality is reduced, the distances decrease significantly, and so does the "model" size.

With these results in mind, we know that the PCA-adjusted K-Nearest Neighbors classifier performed the best, but the Naive Bayes classifier without PCA performed almost just as well but at a fraction of the file size. When scaling this problem to the thousands, this probably makes the Gaussian Naive Bayes classifier without PCA the best choice for this specific project.

5.2 Possible Improvements and Extensions

Although 75% accuracy is a respectable performance for a task that deals with great randomness, it is definitely possible for improvements. It would be interesting to explore how the Gaussian Naive Bayes classifier can be improved upon. Gaussian Naive Bayes does make the assumption that all features follow the Gaussian distribution, and although this is mostly true, there are certainly features that may not be Gaussian. It is possible to transform the distribution of the features to be Gaussian, which may boost the performance of the Naive Bayes classifier.

On the subject of Naive Bayes, which deals with prior and posterior knowledge, it might be beneficial to incorporate information about the previous pitch a batter has seen into the model. In baseball, sequencing pitches can be extremely important, as the pitches a batter has seen greatly affects their mindset and ability to hit new pitches.

Even though Logistic Regression did not give extremely useful present results, the boost in accuracy after using the kernel trick was very promising. The RBF kernel clearly provided some value, but it is most likely not the optimal kernel for this specific problem. If more time was spent into tuning the parameters for the RBF kernel, or even creating a new, clever kernel that properly fits this set, a Kernelized Logistic Regression classifier could produce a good predictor. Furthermore, a Kernelized SVM classifier would also be an interesting avenue, as it may be more suited for a classification problem like this, and there is a great deal of literature concerning Kernelized SVMs.

Of course, this project must be scaled to every player in professional baseball. Meaning, a model must be built for each player, and trained on that specific player's tendencies. It is entirely possible that K-Nearest Neighbors is the best classifier for one player, but Naive Bayes performs better for another. So, when going about a problem of this scope, it is probably better to find the classifier that performs well on the highest amount of players, instead of testing on one individual, and immediately building iterations of the model best fit to that individual.

6 References

1. Baseball savant: Trending MLB players, Statcast and Visualizations. baseballsavant.com. (n.d.). Retrieved from https://baseballsavant.mlb.com/

2. Acquiring and analyzing baseball data. Root Mean Squared Musings. (n.d.). Retrieved from https://billpetti.github.io/baseballr/

3. Kramer, O. (2013). K-Nearest Neighbors. In Dimensionality reduction with unsupervised nearest neighbors (pp. 13–23). essay, Springer Berlin Heidelberg.

4. Salih, D. M., & Esmail, M. F. (2019). Bayesian and naive Bayesian decision boundaries for multidimensional cases. 2019 International Conference on Computing and Information Science and Technology and Their Applications (ICCISTA). https://doi.org/10.1109/iccista.2019.8830658

5. Lei, D., Tang, J., Li, Z., & Wu, Y. (2019). Using low-rank approximations to speed up kernel logistic regression algorithm. IEEE Access, 7, 84242–84252. https://doi.org/10.1109/access.2019.2924542

6. HAN, J., PEI, J., & KAMBER, M. (2012). Data Mining: Concepts and Techniques. Morgan Kaufmann.

7. Rahayu, S. P., Purnami, S. W., & Embong, A. (2008). Applying kernel logistic regression in data mining to classify credit risk. 2008 International Symposium on Information Technology. https://doi.org/10.1109/itsim.2008.4631725

8. Kuo, B.-C., Ho, H.-H., Li, C.-H., Hung, C.-C., & Taur, J.-S. (2014). A kernel-based feature selection method for SVM with RBF Kernel for Hyperspectral Image Classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 7(1), 317–326. https://doi.org/10.1109/jstars.2013.2262926

9. Larsen, A. B., Vestergaard, J. S., & Larsen, R. (2014). HEP-2 cell classification using shape index histograms with donut-shaped spatial pooling. IEEE Transactions on Medical Imaging, 33(7), 1573–1580. https://doi.org/10.1109/tmi.2014.2318434