Andrew Doyle

August 15, 2017

Capstone Report

**Definition**

**Section 1 - Overview**

**Domain Background**

For my Capstone project I will be applying several different unsupervised learning algorithms to Major League Baseball data, specifically clustering algorithms, in order to properly classify each pitch into its type (i.e. Four-Seam Fastball, Curveball, etc.). There are several reasons I chose the domain of baseball for my final project. First and foremost, I am a big fan of baseball, particularly the analytical and quantitative side of the game. It was this interest that lead me to the field of machine learning in the first place, and even helped me learn some machine learning and statistical software (Marchi & Albert, 2013). Statistical analysis is in the DNA of the game of baseball, as game data has been collected for as long as the game has been around. Because of the vast amount of data, the field lends itself very well toward machine learning.

Secondly, with the advent of the Statcast system, recently collected data can offer insights into the game that were previous unavailable. Per the website Baseball Savant (<https://baseballsavant.mlb.com/about>), Statcast

is a state-of-the-art tracking technology, capable of measuring previously unquantifiable aspects of the game. Set up in all 30 Major League ballparks, Statcast collects data using a series of high-resolution optical cameras along with radar equipment. The technology precisely tracks the location and movements of the ball and every player on the field, resulting in an unparalleled amount of information covering everything from the pitcher to the batter to baserunners and defensive players.

Finally, the datasets for MLB data are both large and free to use by anyone, making it very conducive toward this type of analysis. Machine learning has been applied to baseball data both publicly among the baseball community for pleasure and privately among professional teams with the goal of gaining a competitive advantage. An example of using this field of research on a Statcast dataset would be using a Random Forest classifier to predict whether a batted ball will be a hit or an out (<http://www.hardballtimes.com/using-statcast-data-to-predict-hits/>).

**Datasets and Inputs**

Statcast tracks many different features of each pitch thrown in a major league baseball game, with all data being free to use and publicly available via [www.BaseballSavant.com](http://www.baseballsavant.com). The impetus for the project came from watching the game - pitch recognition is a vital aspect of being a successful hitter. This recognition is based on some combination of the velocity of the pitch, the spin direction and rate, and horizontal and vertical movement (known as the break).

The specific data set I will be using will be every pitch thrown by Chris Sale, starting pitcher for the Boston Red Sox. Chris Sale has been one of the very best pitchers in baseball throughout his entire career, and this year has arguably been the best of his career and one of the best statistical years by a starting pitcher in Red Sox franchise history. He is on pace to set career highs in Wins Above Replacement, Strikeouts and Strikeouts per Nine Innings Pitched, and is already in the discussion for not only the American League Cy Young Award but the Most Valuable Player Award as well (http://www.fangraphs.com/blogs/chris-sale-for-mvp/).

**Section 2 – Problem Statement**

The problem I will be attempting to solve is “can unsupervised Machine Learning algorithms correctly classify pitches thrown by Major League Pitchers into their proper pitch type?” Using Statcast data, consisting of perceived velocity, spin rate, horizontal movement, and vertical movement I will apply unsupervised clustering algorithms in my effort to accomplish this. Statcast tracks the above features for every pitch that is thrown in major league baseball, as well as the result of each pitch and numerous other metrics. The metric I will be attempting to identify is the resulting label of the pitch type.

**Section 3 – Metrics**

Because the label of each pitch is already known, I can use the pitch type as a way to evaluate the accuracy of each cluster created, as well as gain any insights into the pitch repertoire of this particular player.

**Analysis**

**Section 4 – Data Exploration**

The first thing to do is to download Chris Sale’s dataset from Baseball Savant, which can be found here: ([Chris Sale Data](https://baseballsavant.mlb.com/statcast_search?hfPT=&hfAB=&hfBBT=&hfPR=&hfZ=&stadium=&hfBBL=&hfNewZones=&hfGT=R%7C&hfC=&hfSea=2017%7C&hfSit=&player_type=pitcher&hfOuts=&opponent=&pitcher_throws=&batter_stands=&hfSA=&game_date_gt=&game_date_lt=&player_lookup%5B%5D=519242&team=&position=&hfRO=&home_road=&hfFlag=&metric_1=&hfInn=&min_pitches=0&min_results=0&group_by=name-event&sort_col=pitches&player_event_sort=api_p_release_speed&sort_order=desc&min_abs=0%23results)). I will save the file as ‘sale\_pitch\_data.csv’. Once the file is downloaded it can be loaded into the iPython file the we can explore what we have.

The first thing to do is view the data as is, prior to any processing. Analyzing the data shows us that the dataset has 2626 samples with 78 features each. Not all 78 features will be relevant to this analysis however, and some fields do not populate at all. For this particular analysis I will be using a subset of those 78 features, specifically 'pitch\_type', the type of pitch, 'effective\_speed', the perceived velocity of the pitch (in miles-per-hour), 'release\_spin\_rate', the rotational spin rate of the pitch, 'release\_extension', the amount of extension the pitcher achieves on the pitch, 'pfx\_x', the horizontal movement of the pitch, 'pfx\_z', the vertical movement of the pitch, 'release\_pos\_x', the x-coordinate of the release point of the pitch, 'release\_pos\_y', the y-coordinate of the release point of the pitch, and 'release\_pos\_z', the z-coordinate of the release point of the pitch. I will name the subset of the original data, which consists only of the above reference fields, as ‘df\_pitch’.

Now that I have the subset of data which I will use in the clustering algorithm, it needs to be processed before any further analysis can be done. The first step in this is to remove any null values in the original data, which exist throughout the dataset. Once those indices are removed, the data type of each feature needs to be examined to ensure that the processing will work correctly. Examining the data type of each feature gives me the following:

pitch\_type object

effective\_speed object

release\_spin\_rate object

release\_extension float64

pfx\_x float64

pfx\_z float64

release\_pos\_x float64

release\_pos\_y float64

release\_pos\_z float64

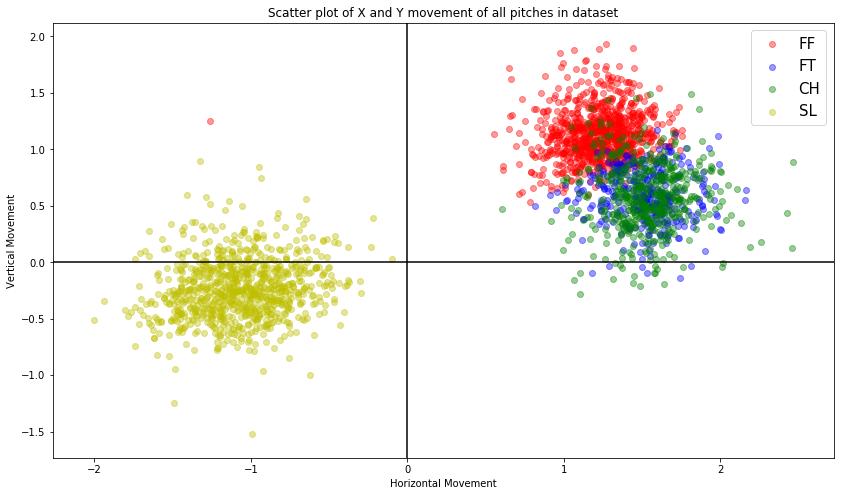
dtype: object

A few of these features need to be converted to a different data type, in this case, I want to convert all numerical data to a float data type.

Once the null values have been removed and all numerical features are stored as floats, I will calculate 2 new features based on the existing data. Those 2 features are the ‘break’, which is the magnitude of the movement vector, and the ‘angle’, which is the angle the pitch moves at relative to the release point of the pitcher.

The ‘break’ can be calculated by using the Pythagorean theorem on the horizontal and vertical movement variables for each pitch. Similarly, the angle each pitch breaks at can be calculated by using trigonometric functions based on the same variables.

Calculation of the angle requires careful consideration however. Normally angle is calculated with respect to the positive portion of the x-axis. This becomes a problem however when a particular pitch type exhibits positive vertical movement in some instances and negative vertical movement in other instances. Because of this issue, I want to make sure that the axis which the angle is calculated relative to does not pass through a group of pitches, thereby skewing the calculation of the angle, which in turn would throw off any clustering algorithms performed in the data. To illustrate this point, I will plot the horizontal and vertical movement of each pitch in the dataset:

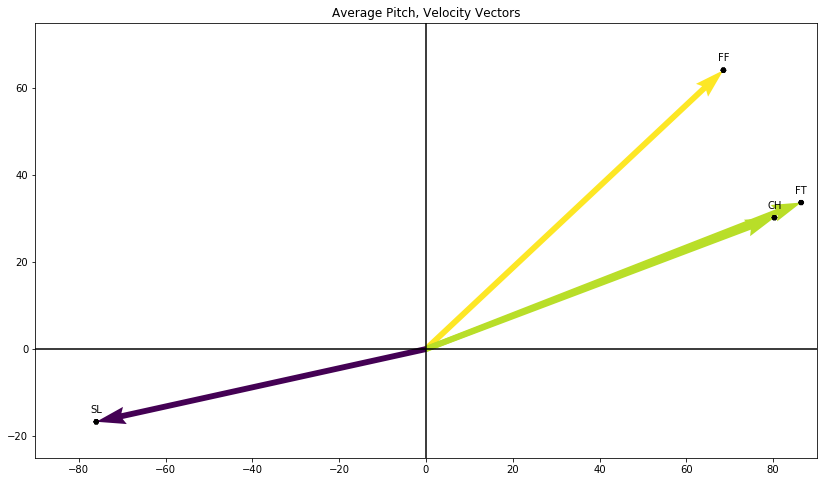


Using the positive portion of the x-axis as the starting point for calculating the angle would result in some pitches being classified with an extremely low angle (<15°) and others being classified with an extremely high angle (>345°), when these pitches are very similar in terms of their horizontal and vertical movement. For this reason I will use the negative portion of the y-axis as my line of demarcation from which angle will be calculated.

To illustrate how the angle is calculated, let’s suppose Chris Sale threw a changeup that exhibited no vertical break, and positive horizontal break. This pitch would thus like on x-axis, and it’s break angle would be calculated at 90°. Similarly if Chris Sale threw a slider that exhibited no vertical break, but had negative horizontal break, the break angle would be 270°.

Now that the dataset has been properly formatted and holds all features we are going to use, I will select a set of sample points which we will look at in the future once the cluster analysis has been completed. I am primarily interested in how each pitch type is classified and whether the clustering algorithms can properly label each pitch type. I will thereby randomly select 4 indices, each one classified as a different pitch type.

Now that the samples have been chosen, I want to explore each type of pitch a little more to get an idea of how they vary from one another. By looking at the mean values for the 4 types of pitches Chris Sale thrown, namely 4-seam fastballs (“FF”), 2-seam fastballs (“FT”), changeups (“CH”) and sliders (“SL”), we can view the features for the average pitch of each type. In addition to calculating these mean values, I also will plot a vector for each average pitch so we can see how the mean values of each pitch differ from one another.



These vectors are merely to give an idea of what the average fastball, changeup and slider thrown by Chris Sale does. The length of the vector represents the velocity and the angle of the vector is the angle the pitch typically breaks at. There are a few things we can infer about each pitch based on the graph.

The first is that it looks like both types of fastballs and the changeup move “up”, but this doesn’t mean the ball is rising, in the traditional sense. The horizontal and vertical movement of each pitch recorded is done so relative to a pitch thrown with no spin whatsoever and which is only affected by gravity (http://www.fangraphs.com/library/pitch-type-abbreviations-classifications/). In this case, the 4-seam fastball, 2-seam fastball and changeup all have “positive” vertical movement in the sense that they do not exhibit the expected amount of drop due to gravity.

The second takeaway from this plot is the similarities and differences between each type of pitch. The 4-seam has the highest average velocity and tends to have positive vertical and horizontal break. The 2-seam fastball and changeup are very similar in terms of both break and velocity, however the 2-seam is typically thrown harder. The slider Chris Sale throws is clearly the most unique of the pitch-mix: it is both the pitch with the lowest average velocity and the only one with “negative” movement.

**Section 5 – Feature Relevance**

Now that we have explored the pitch dataset a bit, we can start to look closer at the different features to see how relevant they are to one another. The first step in this process is to run a supervised regression algorithm on the dataset with a feature removed to view how well that feature can be predicted by the rest of the dataset. First, I will attempt to predict the angle of break using a regressed decision tree.

When the angle is attempted to be predicted, the dataset does so with a correlation coefficient of >.99, meaning that more than 99% of the variation in the angle can be determined by the rest of the dataset. The angle can be predicted with incredible accuracy – which should be expected, as the angle was calculated using the information we already had. So let’s try to predict another feature which wasn’t calculated, effective\_speed. Running the same test but on the effective speed of each pitch produces a correlation coefficient of >.83, again a very high number, which means the dataset can predict the velocity of each pitch with a high degree of accuracy.

Continuing with our exploration of the features in the dataset, I will create a scatter plot of each feature paired with every other feature to show how the related to one another, as well as how each feature is distributed.

**Section 6 - Algorithms and Techniques**

As indicated in the introduction I will be performing 2 types of cluster analysis on the dataset, KMeans and Gaussian Mixture. Each has it’s own pros and cons.

K-means clustering identifies a specified number of centroids that cluster the data points together. These centroids aim to minimize the 'inertia' of each cluster. This algorithm works well for very large sample sizes, however because it is iterative until a minimum is calculated it can take time to run. It scales well to extremely large samples.

GMM clustering is a clustering algorithm used when you are trying to estimate covariance in the data. It is a faster algorithm, however if the sample is too small covariance can be difficult to calculate.

The clustering method varies between the two however. K-means attempts to separate the data into groups of equal variance, whereas GMM will create clusters of covariance.

The goal of this project is to see how each algorithm describes the dataset and then compare to the actual pitch labeling. Because this is exploratory in nature, I will perform both types of clustering on the dataset and see what each says about the data.

**Methodology**

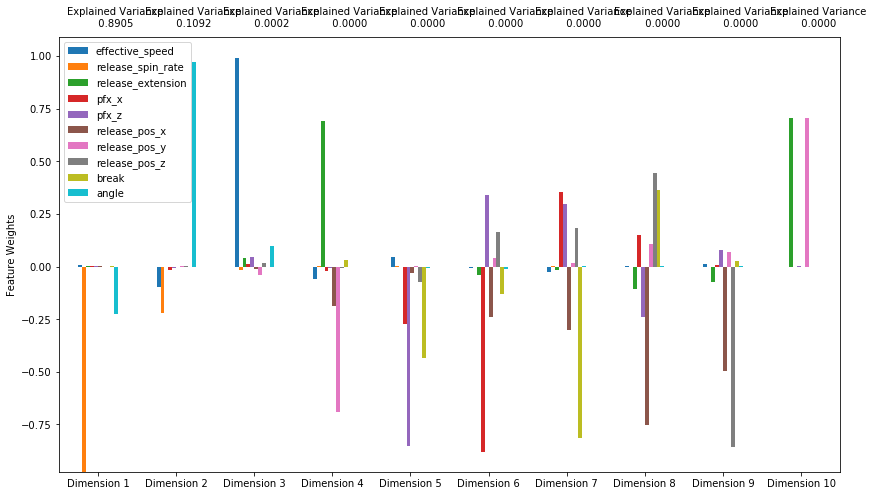
**Section 7 – Data Preprocessing**

Now that we have analyzed the data closely and viewed the features, we can start the process of performing the clustering. Prior to doing this the dataset needs to be preprocessed so it is in a format that the clustering algorithms can use.

The primary step in preprocessing is reducing the dimensionality of the dataset through principal component analysis. Principal component analysis, per scikit-learn, is

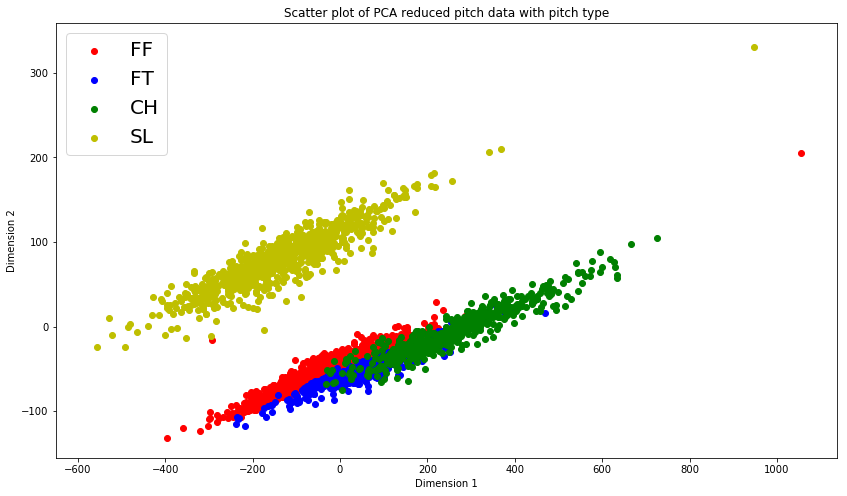
used to decompose a multivariate dataset in a set of successive orthogonal components that explain a maximum amount of the variance. In scikit-learn, PCA is implemented as a transformer object that learns n components in its fit method, and can be used on new data to project it on these components (<http://scikit-learn.org/stable/modules/decomposition.html#pca>).

Once we reduce the dataset to 2 dimensions, we can continue to explore the data. The first thing to do is view how much variance is explained by each dimension calculated, which is shown below.



We can then calculate the total explained variance of the first two dimensions, which is the dataset we will perform the clustering on. That total variance is >.99, meaning nearly the entirety of the variance in the dataset is explained by the first 2 dimensions.

Now that PCA has been performed on the data, we can construct a scatter plot to see where each pitch lies relative to the first 2 dimensions.

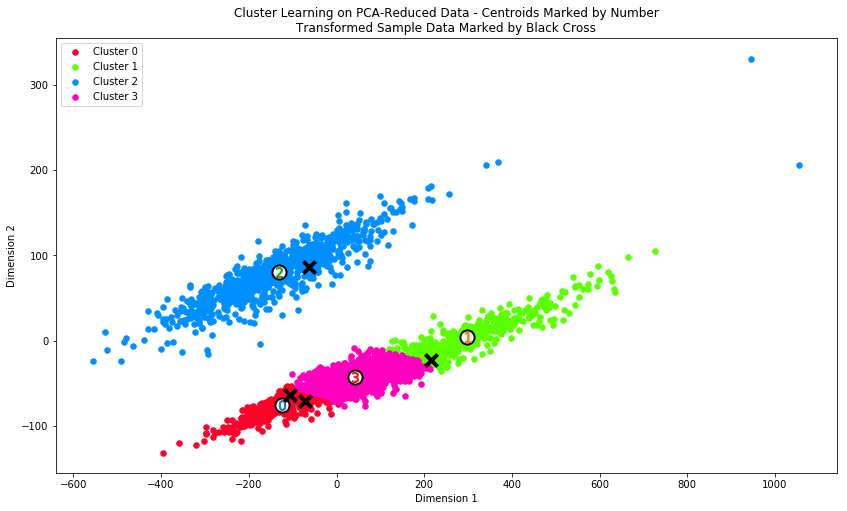


This illustrates how the 4 pitch types are clustered together when the features are reduced to 2 dimensions. We will refer to this plot once we perform our cluster analysis to see how well the algorithms label each pitch.

**Section 8 – Implementation**

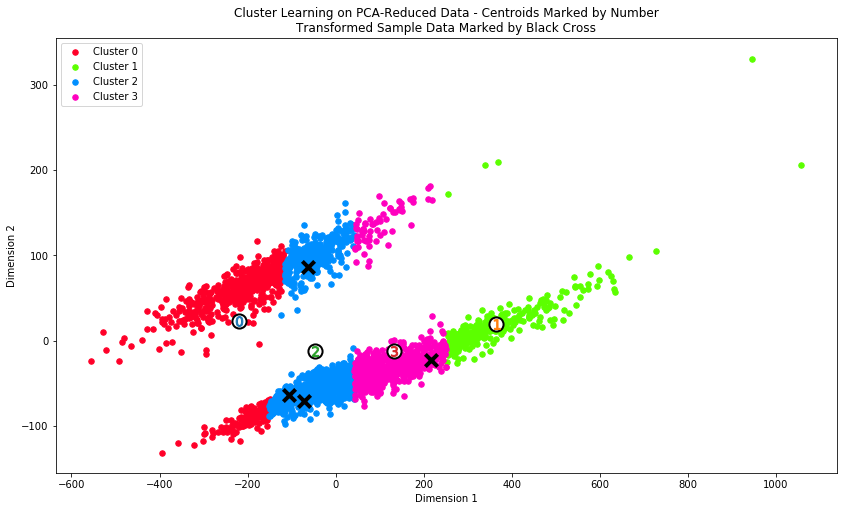
Now that the dataset has been preprocessed and examined, it is ready for the clustering algorithms. I will first run each clustering algorithm on the dataset with 4 clusters, as we already know there are 4 pitch types.

Running a Gaussian Mixture model on the reduced data produces the following plot:



Here the 4 clusters are identified along with their centers, as well as the 3 sample points identified earlier. This clustering algorithm does a decent job at labelling each pitch, with a few noted exceptions. Sliders and changeups are classified very well, however the algorithm does not differentiate between the 2 fastball types well.

Running a KMeans algorithm on the data produces the following plot:



As illustrated, KMeans does not perform as well as the GMM clustering did. The pitches are not differentiated properly, with sliders being classified into the same cluster as 4-seam fastballs, 2-seam fastballs and changeups.

For the next step in the analysis, I won’t provide the algorithms with the number of clusters ahead of time. Instead, I’ll will iterate the each model with a different number of clusters and calculate the Silhouette score that results. The Silhouette score, as defined by scikit-learn, is

composed of two scores:

a: The mean distance between a sample and all other points in the same class.

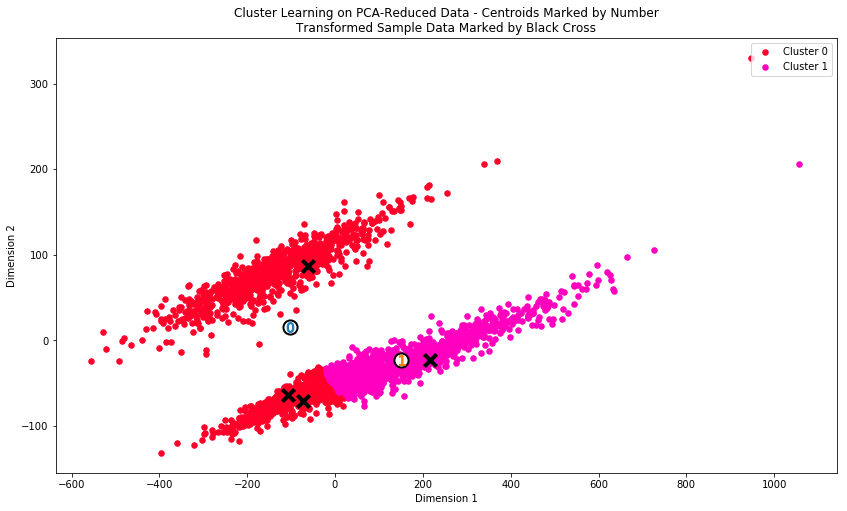
b: The mean distance between a sample and all other points in the next nearest cluster.

The Silhouette Coefficient s for a single sample is then given as:

s = \frac{b - a}{max(a, b)}

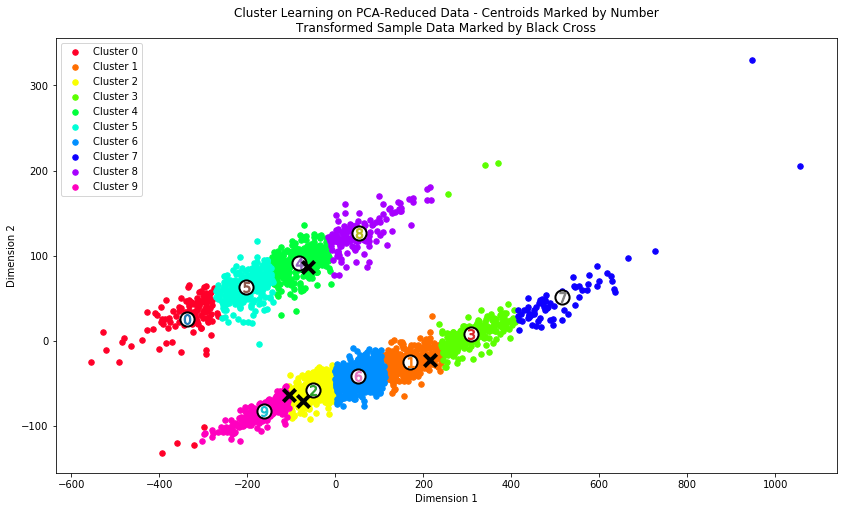
This coefficient for a data point measures how similar it is to its assigned cluster from -1 (dissimilar) to 1 (similar). Calculating the mean silhouette coefficient provides for a simple scoring method of a given clustering. The closer the silhouette score is to 1, the more representative each cluster is of each data point.

First I will calculate the number of clusters with the maximum silhouette score for the Gaussian Mixture model, then I will do the same for the KMeans model. For the GMM, the optimal number of clusters calculated is 2, with a silhouette score of .449. With the number of clusters equal to 2, the GMM clustering produces the following plot:



While the silhouette score is better, this doesn’t perform as well at labeling each pitch type as the model did when 4 clusters is used.

After calculating the optimal silhouette score for the KMeans algorithm, the results are 10 clusters, with a score of .508. That model produces the following plot:



Now this is interesting! According to the KMeans clustering algorithm, Chris Sale doesn’t have a 4-pitch mix, he actually has a 10-pitch mix! Is it possible Chris Sale throws 10 different types of pitches?

**Results**

**Section 9 – Model Evaluation and Validation**

According to the results of the KMeans clustering algorithm, Chris Sale may actually throw 10 different “types” of pitches, despite there being only 4 classifications in the Statcast data. It certainly seems that our clustering algorithms don’t label each pitch accurately. But is there anyway we can validate the labelling done by Statcast? Is it possible that the labelling isn’t fully representative of the differences in the pitches Sale throws? To do this, we can hear from look no further than a story by sportswriter Brian MacPherson, from a story back in July of 2017:

Sale didn’t want to talk much about how he goes about his task after Saturday’s game.

“I’m just trying to go out there and win games,” Sale said. “However that shakes out, it does.”

But others in the Red Sox clubhouse were happy to do it for him.

What makes Sale exceptional is the way **he expands a three-pitch repertoire into an eight- or nine-pitch repertoire.**

**“He’ll add and subtract with everything, basically,” Pomeranz said. “He has three really good pitches, and through changing speeds, he turns them into even more than that. It’s all about getting them off-balance. They never know what to expect because everything is jumping all over the scale of speeds.”**

**Sale doesn’t just throw a fastball. Sale throws several fastballs — a two-seam fastball he calls a batting-practice fastball for an early-count strike; a two-seam fastball in the low-90s to get weak contact; and a four-seam fastball he can ride up to 99 when he needs to, either up and away from a lefty or in on the hands of a righty.**

**Sale doesn’t just throw a changeup. Sale throws two changeups, and he has two different changeup grips.**

**Sale doesn’t even throw just a slider. Sale throws a get-me-over slider in the mid-70s; a back-door slider around 80 that elicits weak contact; and a wipeout slider around 80 with which he gets most of his swings and misses.**

**“He’s giving you different looks within all of the pitches themselves,” Bannister said.**

Just adding to and subtracting from a fastball is rare ability, to be able to pitch with the fastball at several different speeds within the same outing.

A first-inning strikeout of Troy Tulowitzki saw Sale throw a two-seam fastball 92 mph and get a foul ball on it — and then come back with a four-seam fastball at 96 to which Tulowitzki had no chance to catch up.

Bannister pitched for five seasons in the major leagues but has been around the game his entire life. He said he’s seen only two other pitchers who can vary speeds on their fastballs to the degree Sale does — Detroit’s Justin Verlander and Arizona’s Zack Greinke.

“They can comfortably pitch 92-95 and then reach the upper 90s at will,” he said. “It’s a special ability. Most people try to do that, but you just can’t.”

“He can be throwing 93 — and then here comes 98,” Pomeranz said. “That’s what makes him even more special. He’ll throw a slider at 76 and then he’ll throw one at 82 and then he’ll throw a changeup a little slower and then he’ll throw a BP fastball — and if you start sitting on soft, all of a sudden, here comes 99.”

The featured pitch for Sale on Saturday was that slider. Sale threw nearly 50 sliders against the Blue Jays, ranging from 75-83 mph, and he elicited nine swings and misses. It wasn’t until the fourth inning that a Toronto hitter put a Sale slider in play.

One of his most impressive sequences came when he struck out Toronto second baseman Ryan Goins on three pitches, all sliders. He threw one at 79 for a called strike. He threw one at 75 that got a foul ball. He finished Goins off with a slider at 80 in the dirt.

“It was running like this, all the way across,” said Pomeranz, holding his hands about two feet apart.

Source: http://www.telegram.com/sports/20170701/red-sox-chris-sale-excels-at-turning-three-pitches-into-arsenal

According to Sale’s teammates, due to the way he can vary the speed and break of each “type” of pitch he throws, there is enough variation that he may as well be throwing 10 different pitches!

So we now have 2 sources indicating that Chris Sale actually throws 9 or 10 pitches as oppose to the 4 types of pitches Statcast classifies. If we are to take this as true, we can use the optimized KMeans model with 10 clusters as truly representative of Sale’s pitch repertoire. Let’s explore that model a little further.

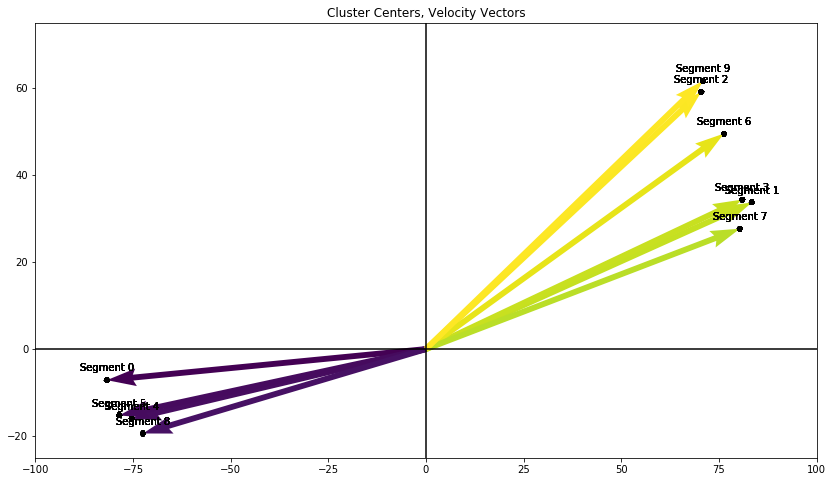
By looking at the centers of all 10 clusters, we can see what Chris Sale’s true pitch mix looks like. After converting the reduced data back to the original features, a subset of the 10 centers look are the following:



For convenience, I’ve sorted by angle from smallest to largest to illustrate how Chris Sale varies his pitches. If we are to separate his pitch type into (FF/FT/CH) and (SL), we can see that his pitches really operate on a scale more than anything. Segments 7, 1, 3, 6, 2 and 9 represent the scale for that first grouping of pitches. This can be interpreted as a scale going from a true changeup (85 mph at 109°) to a true 4-seam fastball (94 mph at 130°). The pitches between those 2 extremes are stop on the scale, with 2-seam fastballs, harder changeups and “batting-practice” 4-seam fastballs making up that group.

The next group of cluster centers are Chris Sale’s sliders. Again, this can be interpreted as a scale, with a harder, tighter slider representing Segment 0 (82 mph at 275°) and Segment 8 representing a softer slider with more break (75 mph at 285°).

To illustrate these 10 different pitches even further, let’s plot the velocity vectors again, however instead of plotting the average velocity and break angle for each of Sale’s 4 pitch types, I will plot the velocity and break angle of the 10 cluster centers.



**Solution Statement**

The feature I will be trying to identify is pitch type, which is identified along with each other feature that Statcast tracks.

In order to accomplish this I will use several different clustering algorithms on the pitch data. Clustering “is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis” (<https://en.wikipedia.org/wiki/Cluster_analysis>). I’ve decided to use clustering as a way of labeling each pitch thrown because the data lends itself well to this sort of analysis, namely that these large datasets exist with labels attached to each record that contains multiple features for the clustering algorithms to use.

Specifically, the two clustering algorithms I will use are K-Means and a Gaussian Mixture model. I will review each individually in detail.

Per the scikit-learn site (<http://scikit-learn.org/stable/modules/clustering.html#k-means>), the K-Means clustering algorithm

clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares. This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.

The k-means algorithm divides a set of N samples X into K disjoint clusters C, each described by the mean of the samples in the cluster. The means are commonly called the cluster “centroids”; note that they are not, in general, points from X, although they live in the same space. The K-means algorithm aims to choose centroids that minimise the inertia, or within-cluster sum of squared criterion:



A Gaussian Mixture model

is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

**Benchmark Model**

In this case, the benchmark model is the actual pitch type. The results are recorded for every pitch thrown, so I can easily separate the data set into the features and the variable I am attempting to predict. I can then use that variable as a method for evaluating the classification made by the algorithm.

In the researching this project, the only other possible benchmark was a blog post conducting the same sort of cluster analysis on pitch data (<https://baseballwithr.wordpress.com/2015/02/22/pitch-classification-with-k-means-clustering/>). This analysis would not be identical however, as the author uses different software and a different dataset. It can be used as a reference however when conducting the analysis, even if the work done is slightly different.

**Evaluation Metrics**

Because the label of each pitch is already known, I can use the pitch type as a way to evaluate the accuracy of each cluster created.

**Project Design**

The initial step in the process is to download the data set. The data set will consist of a single major league pitcher and every pitch that pitcher have thrown in the 2016 and 2017 seasons, along with all of the features measured by Statcast. The reason for using a single pitcher is because no two pitchers are exactly alike. The curveball of one pitcher may be significantly different, in terms of velocity and movement, from that of another pitcher. The pitch repertoire of a single pitcher will be much more consistent.

Once the dataset is complete the data needs to be cleaned and preprocessed. This consists of extracting only those features which are relevant to the clustering algorithm. Those features are pitch type (fastball, curveball, etc.), which is the label that will be used to evaluate each sluter, release speed (in miles-per-hour), the horizontal and vertical movement of the pitch, and the spin rate. Once the relevant features have been extracted the values need to be converted to a format that can be used by the algorithm. This is composed of 2 steps.

The first step is to view how each feature is distributed. If any particular feature is heavily skewed or includes outliers those will need to be addressed. The second step is to identify whether principal component analysis (PCA) is needed in order to perform the analysis. PCA is used to reduce the dimensionality of the dataset by weighing each feature by its variance.

Once the data has been preprocessed, the clusters can then be created. After the clusters have been created, the labels created by the cluster can be compared to the actual label of each pitch to view the accuracy of the labeling. Any pitches which are mislabeled will be inspected further for the purposes of edification.