PAOM: Pitch Arsenal Optimization Model

Rating MLB Pitch Arsenals and Recommending Evidence-Based Changes

Ben Gratz | Washington University in St. Louis | B.A. Economics & Computer Science, 2027

2020–2025 MLB Seasons | Data: Baseball Savant Statcast (via pybaseball), supplemental Kaggle arsenal-evolution dataset

Live dashboard: PAOM Dashboard · Streamlit

4,381 real historical arsenal-change events | **6** MLB seasons | **78.9%** ranking concordance on genuinely held-out data

PAOM: Pitch Arsenal Optimization Model

Pitch-quality metrics like Stuff+ and pitch-level xwOBA answer a narrow question well: how good is this pitch, right now, in isolation? What they don't answer is what a pitcher should actually do about it. Should a reliever with a below-average slider drop it? Should a starter with an elite changeup throw it more? These are the questions pitching coaches and front offices face and answering them requires something existing metrics don't provide: real evidence about what happens when a pitcher makes a real change.

PAOM has two connected parts. The first is a scoring system — a single 0–100 rating of how good a pitcher's current arsenal is, built from four components (Effectiveness, Movement, Command, and Velocity). The second, and the more novel piece of this project, is a recommendation engine that answers the "what should they do" question directly: for any candidate change to a pitcher's arsenal, it finds similar historical pitchers who made that same kind of change and predicts the outcome based on what actually happened to them, rather than applying a fixed rule to every pitcher alike. Both are built into a full interactive dashboard, linked above, where either can be explored directly for any real MLB pitcher.

Part 1: PAOM Score

Data

PAOM Score is built from Baseball Savant Statcast data, pulled via the pybaseball library, covering real MLB pitchers across the 2020–2025 seasons — hundreds of thousands of individually-tracked pitches per season, aggregated to the pitcher-by-pitch-type level for scoring (so, for instance, a starter's slider and his fastball are each scored as their own separate row, not blended together).

The Four Components

PAOM Score combines four independently-computed, independently-validated components, each already scaled to a 0–100 percentile before being combined. A score of 75 means this pitch (or pitcher) is better than 75% of everyone else who qualifies — 50 is always exactly league-average, 100 is the very best in the league, on that specific dimension.

Component	What it measures	Scored At
Effectiveness	Real pitch-quality, modeled against outcomes	Pitch type
Movement	How much distinct movement-shape territory the whole arsenal covers	Pitcher
Command	How consistently a pitch is located where aimed	Pitch type
Velocity	Velocity level and range across the arsenal	Pitcher

That last column reflects a structural distinction worth understanding up front. Effectiveness and Command are both computed per pitch type. A pitcher throwing four qualifying pitches gets four separate Effectiveness scores and four separate Command scores, one per pitch, before being usage-weighted into a single real pitcher-level number for PAOM Score itself. Movement and Velocity, by contrast, are inherently pitcher-level only: both measure something about the whole arsenal working together — how much distinct territory the pitches cover together and how wide a velocity range spans across them — so neither can be meaningfully computed for a single pitch in isolation. This is also the real reason the dashboard's own Pitch Comparison tool can show a specific pitch's real Effectiveness and Command scores side by side, but has no per-pitch Movement or Velocity number to show — those two genuinely don't exist at that level of detail.

Effectiveness

Effectiveness is the only component in PAOM built as a real, supervised prediction against an actual outcome. The other three, described below, are built from purely descriptive statistical techniques that never try to predict anything. Effectiveness starts from xwOBA ("expected weighted on-base average"), a well-established measure of how much offensive damage a pitch tends to allow, based on the quality of contact and outcomes it produces. Lower xwOBA is better for a pitcher.

A Ridge regression (a standard statistical method for finding how several factors combine to predict an outcome, with a built-in safeguard against overfitting to noise in the data) is fit to predict a pitch's real xwOBA from four real, pitch-level rate stats:

$$xwOBA \approx \beta_0 + \beta_1 \cdot csw\_rate + \beta_2 \cdot chase\_rate + \beta_3 \cdot hard\_hit\_rate + \beta_4 \cdot gb\_rate$$

csw_rate is how often a pitch results in either a called strike or a swing-and-miss; chase_rate is how often a batter swings at it even though it's outside the strike zone; hard_hit_rate is how often contact against it is hit at 95+ mph; gb_rate is how often it produces a ground ball (launch angle below 10 degrees). The regression learns, from real league-wide data, how much each of these four factors matters for predicting outcomes, rather than assuming they all matter equally.

Because this is a regression fit against a held-out prediction target, it has something the other three components don't: a real R^2 value, measuring how much of the real variation in xwOBA this model actually explains. A five-fold cross-validated run on the 2025 season (checking the model's own predictions against data it wasn't trained on, five separate times) produced an average R^2 of 0.775 — meaning these four real rate stats together explain roughly 78% of the real variation in a pitch's actual xwOBA, a strong result for four inputs this simple. This model isn't trying to reconstruct xwOBA exactly, as xwOBA still carries genuine variation these four rate stats can't fully capture such as batted-ball luck, defense, and other factors. A real R^2 of 0.775 means the model captures most of the underlying signal in xwOBA, while eliminating much of the noise xwOBA can report in order to give us an idea of how effective a pitch truly is. A separate run on 2024 data produced an almost identical 0.780, evidence this isn't a one-season fluke.

The fitted regression also shows which of the four factors mattered most, and every one came back correctly signed, matching baseball intuition. Each of the four components in PAOM is refit on every season's own data, not computed once and reused, so the relative importance of these four factors shifts year to year. Converting each season's coefficients to relative importance (normalized so the four sum to 100%, the same convention used for PAOM Score's own component weights) across the full 2020-2025 history:

Feature	2020	2021	2022	2023	2024	2025
csw_rate	32.4%	34.1%	31.6%	33.5%	34.5%	31.5%
hard_hit_rate	26.9%	30.5%	31.2%	30.7%	29.5%	33.0%
chase_rate	23.2%	21.9%	22.7%	20.3%	23.3%	21.9%
gb_rate	17.5%	13.5%	14.5%	15.5%	12.7%	13.6%

csw_rate leads in five of the six seasons on record, with 2025 as the one exception, where hard_hit_rate edges narrowly ahead instead. chase_rate stays the most stable of the four across the whole history, never straying far from 21-23%, while gb_rate has drifted downward from 17.5% in 2020 to roughly 13-14% in recent seasons. Cross-validated R^2 has stayed in a tight band across the same six years: 0.765, 0.795, 0.784, 0.766, 0.780, and 0.775 respectively, acting as evidence that this predictive strength is a stable property of the underlying relationship, not a one-season result. The numbers shown throughout the rest of this section reflect the 2025 season specifically, and should be read as a snapshot of that season's fit rather than a fixed formula applied unchanged across years.

The literal effect of a realistic 10-percentage-point change in each rate stat on predicted xwOBA, using the 2025 coefficients specifically:

Feature	Per +10pp Change in Predicted xwOBA
hard_hit_rate	+0.0024 (worse)
csw_rate	−0.0023 (better)
chase_rate	−0.0016 (better)
gb_rate	−0.0010 (better)

Hard-hit rate and CSW rate carry the most weight in 2025 — together accounting for roughly two-thirds of the model's total reliance, each individually about twice as influential as ground-ball rate, the smallest of the four. The per-10pp column demonstrates modest predicted magnitudes in concrete terms: even a real, meaningful 10-point swing in a pitcher's single most influential rate stat moves predicted xwOBA by only about two-thousandths — a small number in isolation, and a preview of a pattern that shows up again throughout the recommendation engine in Part 2.

A pitch's raw effectiveness index is simply the negative of its predicted xwOBA (since lower predicted damage should mean a higher score) converted to a 0–100 percentile within the eligible population. The real entry cutoffs (a minimum of 60 real pitches and 40 real swings before a pitch type is even scored) weren't picked as round-number guesses: a first sweep tested candidate pitch-count floors one at a time, and a more thorough follow-up sweep tested pitch count and swing count together, on a full grid — which is how it was discovered that the swing-count floor only actually mattered once the pitch-count floor had already moved. Testing them one at a time alone would have missed that.

Confidence and Shrinkage — a Concept Used Throughout PAOM

Before going further, it's worth explaining a concept that reappears in every single PAOM component: confidence, and what it does to a score.

A pitcher who's thrown a slider only 65 times this season hasn't shown enough real data to be fully trusted — that small of a sample could easily reflect a lucky or unlucky stretch, not their underlying talent. But simply throwing out a pitch's score for having "too little data" would lose useful information too. PAOM's answer is a middle ground called shrinkage: every raw score is blended toward the league average, with the blend controlled by a real confidence number (0–100) based on how much data backs that score:

$$trusted\_score = raw\_score \times (confidence \,/\, 100) + league\_average \times (1 - confidence \,/\, 100)$$

At 0% confidence, the score shown is just the league average — the raw number is fully distrusted. At 100% confidence, the raw score is shown exactly as computed, with no blending at all. In between, it's a real, proportional mix of the two. Every component described below reports its own trusted version of its score using exactly this formula. It's the trusted version, not the raw one, that actually feeds into PAOM Score. Effectiveness's own confidence formula is:

$$confidence = swings \,/\, (swings + 200)$$

a tested form (chosen after directly comparing it against a simpler alternative on real validation results) that rises quickly at first and then more gradually, matching how statistical uncertainty actually shrinks as more real data accumulates. In practice, across the full real 2025 population, the average Effectiveness confidence came out to 32.85 (33.30 in 2024) — meaning most real pitch-type rows are still meaningfully blended toward league average, not fully trusted at face value, since a great many pitch types in a single season simply don't clear 200 real swings by a wide margin.

Movement

Movement measures how much distinct movement-shape territory a pitcher's whole arsenal covers — the intuition being that an arsenal with real variety is harder for a hitter to prepare for than one where every pitch looks and moves roughly the same.

Five real geometric features feed into Movement: hull area (imagine plotting every pitch type's average movement on a two-dimensional chart, then drawing the smallest possible shape that contains every point. Hull area is how much space that shape covers; a bigger area means a more spread-out, varied arsenal), horizontal coverage and vertical coverage, average nearest-neighbor distance (how far apart, on average, each pitch type sits from whichever other pitch type is closest to it in movement), and fastball-relative break (how differentiated the arsenal's other pitches are specifically from the pitcher's own fastball, since that's the pitch a hitter is calibrated to expect by default).

These five features are combined into one Movement score using PCA (principal component analysis) — a statistical technique that looks at how several related measurements move together across the whole population of pitchers, and finds a single combined axis. This axis, called the first principal component or PC1, captures as much of that real, shared variation as possible, rather than just averaging the five features arbitrarily or assuming they all matter equally. The alternative (simply averaging the five features together) assumes each one is equally important and fully independent of the others, but that's not how these measurements behave in real data. An arsenal with more hull area also tends to have wider horizontal and vertical coverage, since all three are really different lenses on the same underlying "how much space does this arsenal cover" question. Averaging them as if they were five separate, unrelated signals would effectively double- or triple-count whatever they already share in common. PCA solves this directly: rather than assuming a weighting scheme in advance, it looks at how the five features co-vary across the population and finds the one combined direction that captures their shared pattern. This lets the data determine how much each feature should matter, rather than an arbitrary assumption. How much of the total spread across all five original features that one new axis alone accounts for is reported as a percentage. PC1 explaining, say, 60% of variance means that single combined axis captures 60% of how pitchers actually differ from each other across these five measurements together, with the rest being either noise or genuinely separate information one axis can't summarize. This project treats 40% as the minimum worth trusting for any PCA-based component. Below that, the underlying features don't share a strong enough common signal to justify collapsing them into one number, and doing so anyway would risk losing distinct information rather than capturing a genuine shared pattern. Getting Movement's own combination method right took several design attempts, detailed directly below.

An Engineering Story: Movement's Design Attempts

Four of Movement's five raw geometric features were found to correlate 0.55–0.69 with a pitcher's raw pitch count — a meaningful confound, since a pitcher who simply throws more distinct pitch types will mechanically have more room to spread out across movement space, independent of whether that spread reflects any real quality.

Feature	**Correlation w/ Pitch Count (Before)**	**Correlation w/ Pitch Count (After Dampening)**
hull_area	0.563	0.263
horizontal_coverage	0.711	0.375
vertical_coverage	0.620	0.301
fastball_relative_break	0.229	0.094
avg_nn_distance	−0.099	−0.058

The per-feature breakdown tells its own story worth noting: horizontal_coverage carried the worst real confound by far (0.711) and, even after dampening, still ends with the highest residual correlation of the five (0.375) — the single feature this correction worked hardest on, and least completely for. fastball_relative_break started with the mildest real confound (0.229) and ends with the lowest residual (0.094). avg_nn_distance is the outlier of the group: it was already very weakly, and even slightly negatively, related to pitch count before any correction at all (−0.099) — a hint that this specific feature was never meaningfully confounded with arsenal size to begin with, unlike the other four.

Fixing this took several real, distinct attempts, each correcting one real problem while creating another:

- Orthogonalize, then combine. Each feature's own statistical relationship with pitch count was removed first (a technique called orthogonalization), then the five corrected features were combined directly. Combining them afterward re-introduced the same imbalance the correction was meant to fix.
- Group by concept, combine separately. The five features were split into two conceptual groups — coverage-related (hull area, horizontal and vertical coverage) and distinctness-related (fastball-relative break, nearest-neighbor distance) — and each group combined on its own. This reintroduced redundancy between features that still shared the same underlying dependency on pitch count within each group.
- Reverse the order: rebalance first, then orthogonalize. Correcting the relative balance between features before removing the pitch-count relationship reintroduced the same redundancy the reordering was meant to avoid.
- Correct everything simultaneously. A single, combined approach touching both the pitch-count correction and the feature balancing at once diluted the pitch-count correction while destabilizing one of the five features entirely.

The real insight came from recognizing a consistent pattern across every one of these attempts: whichever features were most correlated with each other kept losing their individual credit to whichever one the model happened to favor, regardless of how the feature set was grouped or restructured or which correction was applied first. That consistency was itself the finding: these features are correlated because they reflect a shared underlying fact about what makes a real MLB arsenal good, not because of a fixable measurement artifact. The final design stopped trying to force artificial separation between them — each feature is individually dampened against pitch count by a tested factor (found by sweeping a range of candidate values and checking which one actually produced the most predictive Movement score against real outcomes, not a round-number guess), then combined via one single, unified PCA that lets the real, shared axis emerge directly rather than being forced apart.

A run on 2024 data confirms the final design works as intended. PC1 explains 59.3% of the total variance across the five features, comfortably above the 40% threshold this project treats as the minimum for a shared axis. The per-feature loadings confirm no single feature dominates or falls out entirely, direct confirmation the attempts above converged on a shared signal rather than an artifact of however the features happened to be combined. Each of PAOM's four components is refit fresh on every season's own data, so the loadings and variance explained were pulled across the full 2020-2025 history to check whether this pattern holds up over time, not just in the one season shown throughout the rest of this document:

Feature	**2020**	**2021**	**2022**	**2023**	**2024**	**2025**
hull_area	0.449	0.448	0.440	0.447	0.448	0.435
horizontal_coverage	0.379	0.352	0.368	0.382	0.347	0.379
vertical_coverage	0.398	0.380	0.354	0.330	0.380	0.346
fastball_relative_break	0.502	0.519	0.527	0.526	0.522	0.531
avg_nn_distance	0.494	0.511	0.518	0.518	0.512	0.516
PC1 variance explained	71.2%	62.9%	59.9%	59.6%	59.3%	57.1%

fastball_relative_break and avg_nn_distance lead every single year, not just 2024. This is confirmation of a structural pattern rather than a one-season result. A separate trend is worth flagging: PC1's own variance explained has declined every year since 2020, from 71.2% down to 57.1%. It has stayed comfortably above the 40% threshold throughout, but the shared signal across the five features has measurably weakened over time. This could be an area to further investigate as the constructions of MLB arsenals continue to evolve.

fastball_relative_break and avg_nn_distance contribute the most to the shared axis, with hull_area close behind; horizontal_coverage and vertical_coverage contribute somewhat less individually, though still meaningfully. No feature approaches zero, and none approaches 1.0 alone — a real, balanced contribution across all five, not one or two features quietly carrying the whole score. Two further real checks validate the result directly: after dampening, Movement's correlation with raw pitch count dropped to just 0.05, and its correlation with Effectiveness was only

0.03. These are both appropriately low and serve as confirmation that Movement captures something genuinely distinct from either how many pitches a pitcher throws or how well any one of them individually performs.

Movement's own confidence formula is based on arsenal breadth, not sample size, since Movement measures coverage across pitch types rather than repetitions of any single one:

$$confidence = min(n\_pitch\_types / 4,\ 1) \times 100$$

A pitcher with four or more qualifying pitch types reaches full, 100% confidence in their Movement score; fewer real pitch types means the arsenal's own coverage can't be measured with as much certainty, regardless of how many times any single pitch has been thrown.

Command

Command measures how consistently a pitcher locates a given pitch type — a separate skill from Effectiveness (does the pitch work) and Movement (does the arsenal have enough shape variety). Command is not built from PCA the way Movement and Velocity are — it uses a different technique (a clustering model, described below), so it has no PC1 or explained-variance figure of its own; the real validation numbers that exist for it instead measure how often its own real design choices actually engage in practice.

The naive approach (simply measuring how spread out a pitch's real locations are around one single average point) has a real problem: a pitcher who deliberately works a pitch to two different real spots (backdoor to same-side batters, backfoot to opposite-side batters, for example) would be scored as having bad command, when they're actually showing precise command of two separate targets. Command fixes this with a Gaussian mixture model, a statistical technique that looks for distinct clusters where a pitch actually lands, rather than assuming there's only ever one intended target. Critically, this clustering runs separately within each handedness split — same-handed and opposite-handed batters are never pooled together for this step — and the model automatically decides, separately for each pitcher, pitch type, and handedness split, whether one cluster or two fits the location data better. Dispersion is then measured as each pitch's distance to its own assigned cluster's center, so a two-target pattern scores as precise, not scattered.

Within a given handedness split, the model only ever chooses between one cluster and two; it never tests three or more. This is a more defensible design choice than it might first sound, specifically because of the handedness split described above: the single most common, deliberate reason a pitcher would target multiple distinct real locations with one pitch (a different spot for same-handed versus opposite-handed batters) is already captured by splitting on handedness itself, before the clustering step ever runs. The within-split 1-versus-2 decision is really asking a narrower question: does this pitcher deliberately diversify further within just one handedness split alone? A pitcher who works three or more distinct real spots within a single handedness split specifically would still be missed, approximated by whichever of one or two clusters the data favors rather than modeled exactly. This is a real gap, but a narrower one than it would be without the handedness split already doing work first. A second refinement works alongside the handedness split: dispersion is compared only against other pitchers throwing that same pitch type, not against the whole league pooled together. Different pitch types naturally carry different baseline levels of spread — a slider might inherently land in a wider range of locations than a fastball, even among pitchers with excellent command, just from the physics of the pitch. Comparing dispersion globally across all pitch types would unfairly penalize a pitcher with great slider command simply because sliders as a class tend to be less tightly located than fastballs. This pitch-type-relative normalization fixes that by scoring a pitcher's dispersion only against other pitchers throwing that same pitch type, falling back to the broader, global comparison only when there aren't enough pitchers throwing that specific pitch type to make a fair within-type comparison.

Each of PAOM's four components is refit on every season's own data, so these two engagement rates were pulled across the full 2020-2025 history to check how consistently they hold up.

Year	Handedness Split Used	Pitch-Type-Relative Normalization Used
2020	1,545 / 1,545 (100%)	1,541 / 1,545 (99.7%)
2021	2,299 / 2,299 (100%)	2,294 / 2,299 (99.8%)

Year	Handedness Split Used	Pitch-Type-Relative Normalization Used
2022	2,314 / 2,314 (100%)	2,312 / 2,314 (99.9%)
2023	2,350 / 2,350 (100%)	2,342 / 2,350 (99.7%)
2024	2,393 / 2,393 (100%)	2,385 / 2,393 (99.7%)
2025	2,540 / 2,540 (100%)	2,529 / 2,540 (99.6%)

The handedness split engages at exactly 100% every year on record, and pitch-type-relative normalization stays in a tight 99.6-99.9% band across the whole history, confirming these aren't rare edge-case corrections but the typical way Command gets computed for nearly every pitcher in the league, every year.

Command's own confidence formula is based on raw pitch count directly, the same kind of sample-size logic Effectiveness uses, though in a simpler, linear form rather than Effectiveness's curve:

$$confidence = min(pitches / 300, 1) \times 100$$

A pitch type reaches full, 100% confidence once it has 300 real qualifying pitches, rising in a straight line rather than the faster-then-slower curve Effectiveness's own formula uses.

There is an honest limitation of this metric: it measures dispersion around where pitches actually landed, not around where they were actually intended to go. Data on catcher glove placement or a pitcher's own intended target for a given pitch isn't available in this dataset, so a pitch that misses its target by six inches is indistinguishable, in this data, from a pitch that was thrown exactly where intended but simply landed in a spot that happens to be six inches from the arsenal's other real pitches. Clustering around actual outcomes is a reasonable, defensible proxy for command — but it's a proxy for missed intent, not a direct measurement of it.

Velocity

Velocity combines two measures of a pitcher's arm strength: max velocity (their single hardest pitch) and velocity range (the gap between their hardest and softest qualifying pitch). Both are combined using the same PCA approach described under Movement above. A single-feature version, velocity range alone, was tried first and rejected. It correlated too closely with Movement's own score and behaved unstably across the two validation targets described later in Part 1. The two-feature version tested here was built specifically to see whether adding max velocity as a second, more independent signal would fix that, and it did.

Pulling the per-feature loadings and variance explained across the full 2020-2025 history:

Feature	2020	2021	2022	2023	2024	2025
velocity_range	0.707	0.707	0.707	0.707	0.707	0.707
max_velo	0.707	0.707	0.707	0.707	0.707	0.707
PC1 variance explained	56.6%	59.2%	64.0%	62.6%	63.3%	65.5%

Both loadings land on precisely 0.707 every single year, equal to 1 divided by the square root of 2. This is not a rounding artifact or a one-season coincidence. When exactly two standardized features combine through PCA in perfectly even weight, this is the exact value the loadings take, and it holds across the entire six-year history. Velocity range and max velocity contribute to the combined score in identical, symmetric weight every year, with neither one carrying more influence than the other. Worth noting against Movement's own trend above: Velocity's PC1 variance explained has generally risen since 2020, the opposite direction of Movement's decline over the same period.

Velocity's confidence formula is identical to Movement's, based on arsenal breadth rather than sample size:

$$confidence = min(n\_pitch\_types / 4, 1) \times 100$$

Movement and Velocity share this same arsenal-breadth logic since both measure something about the whole arsenal working together, not any single pitch's own repetition count. Effectiveness and Command, by contrast, are both sample-size-based, though in different forms. Effectiveness's formula is a smooth curve while Command's is a straight line to its saturation point.

Velocity's confidence shrinkage step was tested directly and confirmed necessary, not just assumed. Even with shrinkage applied, Velocity's correlation with raw pitch count in the 2024 run came in at 0.352, meaningfully higher than Movement's post-dampening 0.05. Velocity relies on confidence shrinkage rather than Movement's explicit per-feature dampening step, so some residual relationship with how many pitch types a pitcher throws remains in the trusted score.

Rejected Components

Two additional components were built, tested, and ultimately excluded from the final score, each for a specific, tested reason rather than a stylistic preference.

Tunneling — a well-known baseball concept measuring how similar two pitches look to a hitter early in their flight, even if they end up in very different places — was built across four independent construction attempts: a version based directly on swing-and-miss/chase outcomes, a residualized version (specifically subtracting out each pitch's own individual quality first, to isolate whatever additional value comes from the pairing itself), a version predicted from physical release and movement traits, and a trajectory-based version that solved the actual physics equations for where a pitch is at the exact moment a hitter has to commit to a swing. All four showed essentially zero additional, marginal relationship with real season-level outcomes once Effectiveness and Movement were already accounted for.

An early, simpler version of PAOM's scoring even directly tested whether Effectiveness, a movement-"interaction" concept, and raw movement coverage were secretly measuring the same underlying thing. They weren't — the data below shows all three pairs sitting in a diffuse cloud with no strong linear relationship, confirming they were capturing genuinely separate dimensions, not redundant restatements of each other.

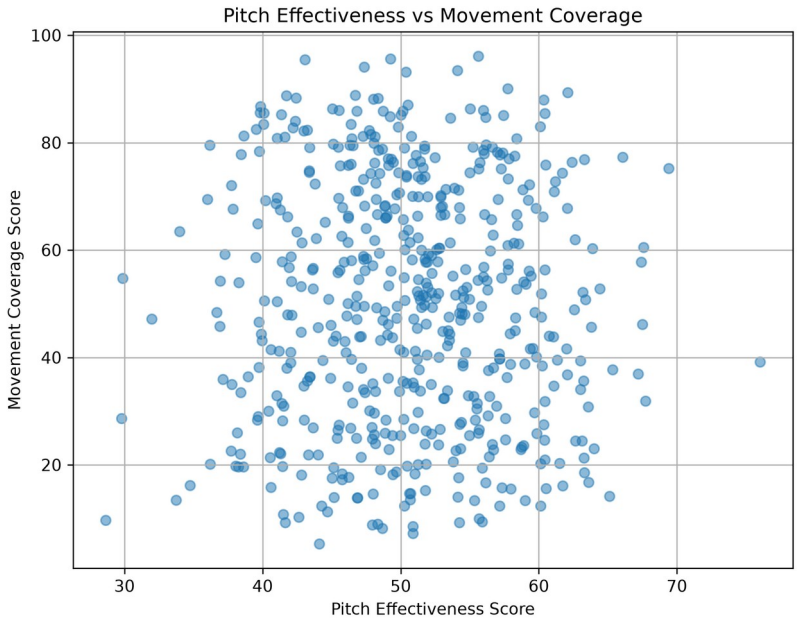

Figure 1. Pitch effectiveness vs. movement coverage (2025) — an early, diffuse relationship confirming these are separate dimensions, not restatements of one another.

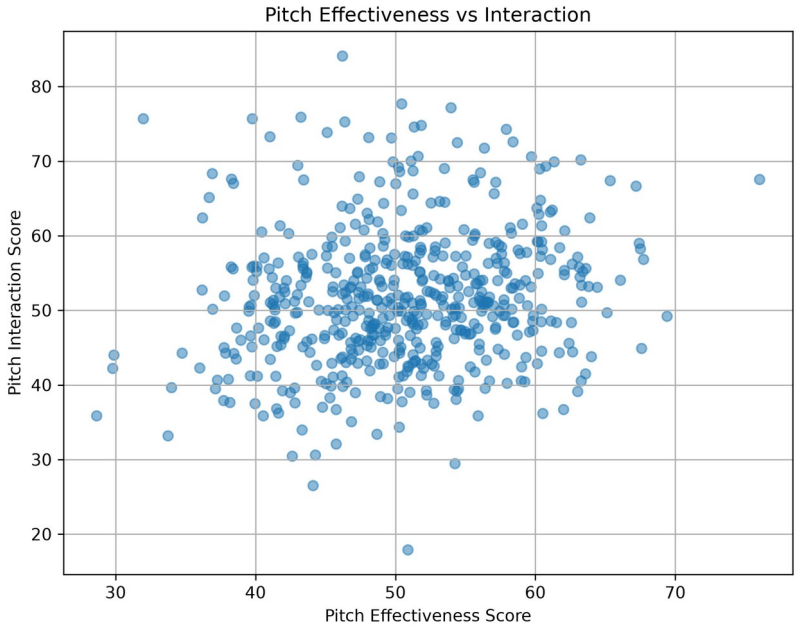

Figure 2. Pitch effectiveness vs. an early pitch-interaction concept (2025) — again no strong linear relationship.

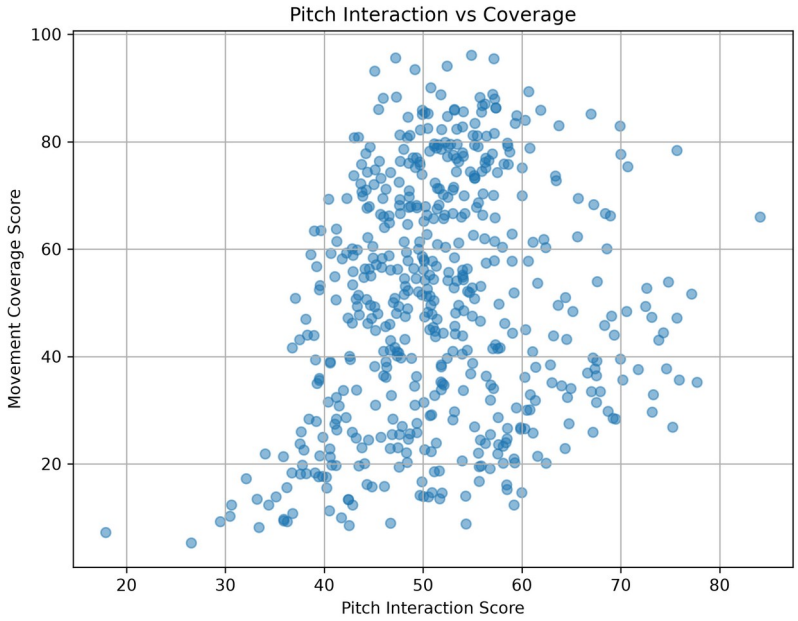

Figure 3. Interaction vs. movement coverage (2025) — a real, honest exception: a funnel shape where low interaction scores rarely pair with high coverage, but high interaction scores can pair with either.

Rather than discard the tunneling concept entirely, it was repositioned as a matching criterion inside the recommendation engine (described in Part 2) and as its own standalone dashboard visualization.

Release Consistency — how tightly a pitcher's release point clusters, both within a single pitch type and across their whole arsenal — showed a small but genuinely stable, correctly-signed relationship, agreeing in direction across both validation targets described below. But its additional contribution to predicting either target, once everything else was already in the model, was close to negligible. It was kept out of PAOM Score itself and instead surfaced as its own dashboard visualization (the release point map).

Combining the Components

The four component percentiles are combined into a single PAOM Score via a weighted average, not an equal split, but a learned weighting reflecting how much each component predicts actual outcomes:

$$PAOM = w\_eff \cdot Effectiveness + w\_mov \cdot Movement + w\_cmd \cdot Command + w\_vel \cdot Velocity$$

The weights themselves come from a Ridge regression predicting an independent validation target (run value per 100 pitches, a standard baseball statistic measuring how many runs a pitcher's performance saved or cost their team) from the four component percentiles, then normalized so the four weights add up to exactly 1 (100%). Each of PAOM's four components is refit on every season's own data. So, this weighted average is recomputed each year rather than fixed once and reused. The learned weights across the full 2020-2025 history are shown below:

Component	2020	2021	2022	2023	2024	2025
Effectiveness	71.2%	58.8%	72.8%	69.4%	65.9%	68.9%
Command	13.2%	19.0%	3.4%	5.2%	20.7%	16.0%
Velocity	6.5%	21.5%	20.9%	15.9%	13.3%	12.7%
Movement	9.1%	0.7%	2.9%	9.4%	0.0%	2.4%

Effectiveness is the largest component in every one of the six years, ranging from 58.8% to 72.8%, confirming its dominant share of PAOM Score is a structural pattern rather than something specific to the current season. Command and Velocity trade the second and third spots across different years, with no fixed rank between them. Movement is consistently the smallest, and in 2024 its weight dropped to exactly 0%, described next.

2024 is the one year on record where a validation safeguard fired. Movement's own coefficient disagreed in sign between the primary and secondary validation targets that year, the only time this happened across the full six-year history. Its raw coefficient in the primary fit also came back negative (−0.022). Per an explicit rule already built into this step, any component whose primary-fit coefficient comes back negative is floored to zero rather than allowed to actively subtract from the score, so Movement's weight for 2024 is exactly 0%, not a small positive number rounded down. This is a concrete demonstration of the sign-agreement check doing its job in a live year: catching a genuinely unstable, likely noise-driven signal and preventing it from shaping that season's scores.

Every component, and the final PAOM Score itself, uses the same trusted (confidence-shrunk) version described earlier, not the raw score. Overall PAOM confidence is the minimum of the four components' own individual confidence values, not an average: a pitcher who has plenty of real data on three components but is still thin on just one gets an overall confidence pulled down to match that single weak link, since the final score is only as trustworthy as its least-supported input. However, the overall PAOM confidence score just shows up on the dashboard. It does not affect the PAOM metric like the individual components' confidence scores do.

Validation

PAOM Score's combination weights were checked against two independent validation targets, to make sure the result wasn't just an artifact of whichever single target it was originally fit to. The first, run value per 100 pitches, is the primary target the weights above are fit to. The second, K% − BB% (strikeout rate minus walk rate), is a widely used measure of a pitcher's underlying skill that Effectiveness's regression never targeted at all. Both were pulled across the full 2020-2025 history, along with the PAOM-eligible population size for each year:

Year	Primary R^2 (Run Value)	Secondary R^2 (K%−BB%)	PAOM-Eligible Pitchers
2020	0.220 (± 0.092)	0.276 (± 0.195)	209
2021	0.286 (± 0.072)	0.463 (± 0.025)	495

Year	Primary R^2 (Run Value)	Secondary R^2 (K%−BB%)	PAOM-Eligible Pitchers
2022	0.351 (± 0.072)	0.488 (± 0.079)	474
2023	0.248 (± 0.067)	0.471 (± 0.074)	467
2024	0.230 (± 0.041)	0.498 (± 0.039)	470
2025	0.253 (± 0.037)	0.451 (± 0.052)	459

2020 stands out as an outlier. This year has the smallest PAOM-eligible population by a wide margin, reflecting the COVID-shortened season, and the secondary target's widest variance, nearly four times any other year's spread. 2020's validation numbers should be read with more caution than the other five years for that reason. Setting 2020 aside, the secondary target's R^2 stays fairly stable in the 0.45-0.50 range across 2021-2025, while the primary target's R^2 moves around more, from 0.230 to 0.351, with no clear trend. All four components agreed in sign across both targets in five of the six years, with 2024 as the one exception described above. This demonstrates that the primary result generally isn't specific to one particular choice of target and shows a concrete case where checking for disagreement actually mattered.

A separate check confirms the four components are worth combining in the first place, not just correctly weighted once combined. No pairwise correlation between any two of the four real component scores exceeded 0.5 in any of the six years, representing evidence that Effectiveness, Movement, Command, and Velocity are genuinely measuring distinct things about a pitcher's arsenal across the whole history, not four redundant views of the same underlying signal.

One disclosed tradeoff: because PAOM Score is a literal weighted average of four already-percentiled components — chosen specifically so a real score can be reconstructed by hand from its four visible parts, rather than hidden inside an opaque model — it no longer carries the strict "50 = exact population median" guarantee the individual components have on their own. That's a deliberately accepted cost of making the final number transparent.

Who Actually Gets Scored

The population that clears every threshold to receive a full PAOM Score is smaller than any individual component's own population. A pitcher can have a valid Movement or Velocity score visible on the dashboard without necessarily having a full PAOM Score at all. For the current season, 2025, Effectiveness scored 459 real pitchers (aggregated up from 1,074 real pitcher-pitch-type rows), while Movement, Velocity, and Command each independently scored a larger population of 722 pitchers apiece. Effectiveness's own real entry requirements (60 pitches and 40 swings, per pitch type) are the strictest of the four, which makes it the real bottleneck. Every one of the 459 real pitchers who cleared Effectiveness also happened to clear the other three, so the full PAOM-eligible population lands at exactly 459. Effectiveness, not any of the other three components, is what determines who receives a complete real score.

These real population sizes aren't fixed; they genuinely vary year to year, since each season has its own pitcher population and its own distribution of who happens to clear each component's thresholds. A direct point of comparison: Movement and Velocity scored 703 real pitchers in the 2024 season, versus 722 in 2025 — a modest year-to-year difference reflecting genuine variation in the underlying population, not a fixed number the model assumes or bakes in.

Part 2: The Historical-Precedent Recommendation Engine

Core Idea

Scoring a pitcher's arsenal is useful on its own, but turning that knowledge into actionable recommendations is what sets this project apart. To do that, I built an engine that recommends arsenal changes based on what similar pitchers

have done in the past. Rather than a hand-written rule like "if effectiveness falls below X, recommend dropping the pitch," the recommendation engine asks a more direct, more honest question: has a real pitcher who looks like this one ever actually made this same kind of change before, and what really happened to them afterward? Every recommendation is built from real historical comps' outcomes, weighted by how similar they are to the target pitcher.

Choosing a Target Metric

Before any of the machinery below was built, a foundational question had to be settled first: what should this engine actually try to predict? WAR was the natural first choice — a single, comprehensive, widely-understood measure of a pitcher's overall value. But testing consistently found it produced a weak, unstable signal, even after feature-engineering improvements to the similarity calculation. This raised a genuine question worth answering directly rather than assuming: was this a problem with WAR specifically, or a more fundamental limitation of the whole modeling approach?

A direct, side-by-side test settled it. The same diagnostic — how much a specific candidate change's own effect (change_specific_coef) mattered relative to simple mean-reversion (mean_reversion_coef) — was run across three candidate targets, on the same stratified sample of ADD events (pitchers adding a new pitch to their arsenal):

Metric	Sign Consistency	Mean-Reversion-to-Change Ratio
WAR	56.3%	5.67×
FIP	58.3%	3.74×
xwOBA-against	83.3%	3.36×

The gap was decisive, not marginal. xwOBA-against showed 83.3% sign consistency — meaning a specific arsenal change's own predicted effect pointed the same direction across nearly every pitch type tested for a given pitcher — versus just 56.3% for WAR, barely better than a coin flip. The mean-reversion-to-change ratio makes the same gap concrete in a different, complementary way: it's simply how many times larger the model's own reliance on a pitcher's past performance (mean-reversion) was, on average, than its reliance on the specific change actually being evaluated. A ratio near 1× would mean the two carry roughly equal real weight; WAR's real 5.67× means mean-reversion swamped the change-specific signal more than five times over, effectively predicting "pitchers who were already good stay good" far more than it was actually evaluating the arsenal change itself. xwOBA-against's 3.36× is still mean-reversion-dominated. This is expected, since a pitcher's recent track record is always genuinely informative, but meaningfully more balanced than either alternative. FIP landed in between on both measures, real evidence this wasn't only a WAR-specific problem: a metric measuring what a single pitch actually does, closer to the change itself, outperforms a metric shaped by an entire season's worth of context. This comparison — not an assumption or a stylistic preference — is why xwOBA-against became this project's default outcome metric throughout, for both PAOM Score's validation and every prediction described below.

Data

The engine is built on a Kaggle-sourced dataset of real pitch-level arsenal evolution, 2020–2025, containing 4,381 historical arsenal-change events across four types: adding a pitch, dropping a pitch, and increasing or decreasing a pitch's usage. Each event is a pitcher's actual decision, with a before-and-after outcome attached. Only true consecutive-season transitions are compared, so a pitcher with an injury or minor-league gap year never gets a change event attributed across a season where a full year of other, unknown things could also explain any difference.

Similarity

To find comparable historical pitchers, two pitchers' similarity is measured as a distance — conceptually the same idea as measuring the distance between two points on a map, just across several real traits at once instead of two:

$$distance = sqrt(\ \Sigma_i\ w_i \cdot (target_i - comp_i)^2\ )$$

Every trait going into this calculation is first standardized (converted to a common, unitless scale) before being compared — otherwise a real difference in, say, velocity (measured in miles per hour) could unfairly dominate a real difference in, say, hull area (measured in square inches), purely because of the units each happens to be measured in, not because it actually matters more.

Five real arsenal-shape features (hull area, nearest-neighbor distance, fastball-relative break, velocity range, max velocity) each carry full weight (1.0) in this calculation; release position and arm angle each carry half weight (0.5). Two additional terms sharpen the comparison further, each weighted at 1.5: pitch-composition overlap (do these two pitchers throw the same kinds of pitches at all) and shared-pitch shape distance (for pitch types both pitchers actually throw, how close in shape are those specific pitches). For candidates other than adding a brand-new pitch, a usage-level term, weighted at 2.0, also compares how closely a comp's own starting usage of that pitch matches the target's, since a pitcher going from 20% to 29% usage is a meaningfully different decision than one going from 37% to 48%, even though the raw change is identical.

The Prediction Regression

For a given candidate change, a weighted linear regression — a regression where more similar real comps count for more in shaping the prediction, and less similar ones count for less — is fit across historical comps, using their before-and-after outcomes:

$$\hat{y}\_after = \beta_0 + \beta_1 \cdot y\_before + \beta_2 \cdot usage\_pct\_delta + \beta_3 \cdot existing\_pitch\_quality$$

An honest pattern worth flagging: even the model's most confidently-recommended candidates typically predict a modest change, not a dramatic one. This is a direct consequence of the regression's own structure, not a hidden weakness. Because a pitcher's own recent performance level (y_before) is a real predictor in every one of these regressions, and because historical outcomes are themselves dominated by mean-reversion — a pitcher who's already pitched well tends to keep pitching close to that level regardless of any single arsenal change — the model's own predicted magnitudes stay small for the same reason. A concrete case: Grayson Rodriguez's recommendation to drop his changeup carries a Strong-evidence tier and 155 real historical comps behind it yet predicts a real xwOBA change of only about −0.0003 — a genuinely tiny number on its own. That's not a sign the recommendation is weak; it's the mechanical result of a model that won't claim a single arsenal tweak can overhaul a pitcher's already-established performance level, only nudge it in a historically supported direction. This is also the direct reason ranking concordance, not raw predicted magnitude, is the more meaningful number throughout this project's own validation, covered fully below.

A second stage, built and validated separately before being integrated into this main regression, predicts a brand-new pitch's own quality directly for a pitcher who's never thrown it — reusing the exact same similarity-weighted comparison group described above rather than a separate model. This stage was originally built as a weighted median (a single, robust "typical value" among the real comps), but data showed that approach barely differentiated between different pitchers adding the very same pitch type — completely different pitchers all got nearly identical predictions, even though their real actual outcomes for that pitch type varied enormously. The fix was a genuine regression instead of a single summary number, letting the prediction shift based on where a specific target pitcher's own traits sit relative to the comps. This redesigned version cleared its own leave-one-out validation (predicting each real historical case while deliberately hiding that case's own real answer from the model first) with a placebo-tested correlation of 0.448 against actual pitch-level outcomes, versus −0.122 for a version tested against deliberately scrambled data.

Swap Predictions

The single most consequential design decision in this project: a pitcher replacing one pitch with another is really one combined decision with one real outcome, not two independent events that happen to occur together. An early approach modeled a swap as the sum of two separate DROP and ADD predictions. A direct check against combined historical outcomes found this badly overstated the real effect, since both independent predictions include the same real mean-reversion pattern, effectively counting that same pull twice.

The fix is a separate model, predicting a swap's combined outcome directly from 455 historical swap-pair events across 328 season transitions:

$$\hat{y}\_after = \beta_0 + \beta_1 \cdot y\_before + \beta_2 \cdot dropped\_usage\_pct\_before$$

with an added weight boost when a real comp's own historical swap matched the exact same (dropped, added) pitch-type pair being evaluated for the target. dropped_usage_pct_before represents how often a given pitch was thrown before it was dropped. This model was validated on unseen 2024-and-later real swap events — meaning it was tested only on events from years it never saw during its own training — and, somewhat surprisingly, outperformed the single-change models on every metric checked: a 36.7% improvement in average prediction error over a naive baseline, and 78.9% ranking concordance (explained fully below in the validation section).

Ranking the Candidates

Every candidate for a given pitcher — NO_CHANGE, every viable ADD/DROP/USAGE_INCREASE/USAGE_DECREASE pitch type, and every real SWAP pair — ends up on one common, real scale: predicted change in xwOBA-against relative to the pitcher's own current real baseline, all computed by whichever of the mechanisms above actually apply to that candidate type. Because lower xwOBA-against is better for a pitcher, every candidate is then sorted by this predicted change in ascending order — the option predicting the largest real improvement (the most negative change) ranks first. NO_CHANGE is included in this exact same ranking as the honest baseline every other option actually has to beat.

Guardrails

Anchor-pitch exclusion. A pitcher's own primary fastball-type pitch (their real, identified anchor pitch) can never be recommended for a full drop — a domain-knowledge decision, not a statistically-derived one, since the model can't directly measure a fastball's real tunneling and setup value for the rest of the arsenal. This was verified directly at full scale across every current recommendation the dashboard has generated: zero cases exist where a pitcher's own anchor pitch was ever recommended for a drop.

High-usage-drop disclosure. A DROP candidate whose current usage exceeds the historical 90th percentile for pitches actually dropped is flagged on the dashboard, not excluded — historical precedent for even higher-usage drops does exist, just rarely.

Minimum resulting arsenal size. No candidate can leave a pitcher with fewer than two qualifying (5%+ usage) pitch types — a DROP that would reduce a pitcher to a single pitch is never recommended, regardless of what the raw numbers say about that specific pitch in isolation.

Confidence Tiers

Every recommendation is assigned a confidence tier based on how many historical events its own prediction is built from — a signal of how much precedent backs a specific suggestion, separate from how good the suggestion itself looks:

Tier	Real historical events
Strong evidence	110+
Moderate evidence	50–110
Limited evidence	Fewer than 50

A recommendation backed by 150 historical pitchers who made a similar change is a much stronger basis for trust than one backed by only 20, even when both predictions point the same direction.

This distinction shows up on the dashboard's Top Recommendations page, which ranks the single best candidate for every pitcher league-wide. Two separate numbers are shown side by side, and they are not the same thing. Predicted

Change is the historical engine's raw, unmodified regression output for that specific candidate — nothing is adjusted. Weighted Score is what the page truly sorts by, and applies a tier-based shrinkage directly tied to a confidence-calibration check described later in this writeup's validation section:

$$weighted\_score = predicted\_change \times tier\_weight$$

with tier_weight set to 1.00 for Strong evidence, 0.85 for Moderate, and 0.70 for Limited — shrinking a candidate's predicted improvement toward zero in proportion to how little evidence backs it, before anything gets ranked. These specific weights were deliberately kept modest rather than dramatic, matching how modest the real, underlying calibration effect actually was (a ~10% relative MAE difference between the lowest- and highest-evidence bins, detailed in validation below) — not an arbitrary or heavier-handed discount invented for the ranking alone. In practice, this means a large predicted improvement resting on thin evidence won't automatically outrank a more modest, better-supported one. The raw Predicted Change number is always shown alongside the Weighted Score, never hidden — only re-ordered.

Confidence Calibration

A direct check on whether the confidence tiers actually mean anything: real historical events were binned by how much evidence backed them, then compared against their own walk-forward-validated error. The lowest-evidence bin showed a real MAE of 0.0266; the highest-evidence bin showed 0.0240. Predictions backed by more historical evidence really were somewhat more accurate on genuinely unseen data. The difference is modest, roughly 10%. A strong-evidence recommendation is more trustworthy than a limited-evidence one, but not vastly more so.

Validation

Every layer of this project was checked against real, held-out data, not just fit to the data it was built from.

Placebo Tests

A placebo test here means the same technique used in a real clinical drug trial, applied to data instead of medicine. Take a real feature, deliberately scramble which value goes with which outcome (breaking the true pairing while keeping the exact same range of values), and see whether the model still "works" on this now-meaningless, shuffled version. If it does just as well on scrambled data as on the real thing, that's a sign the model was never picking up on anything real in the first place — just an artifact of scale or regression mechanics. If the scrambled version collapses to noticeably worse performance, that's evidence the original feature was doing genuine work.

Four predictors inside the recommendation engine were checked this way, not just diagnostic experiments run on the side. New-pitch distance applies only to ADD events: it measures how far the pitch being added actually sits, in movement, from everything already in the pitcher's arsenal. So, a genuinely new pitch shape and a redundant near-copy of an existing pitch get treated differently. Arsenal synergy also applies only to ADD events: it measures whether a pitcher's other, already-established pitches got more swings and misses after the addition, not the new pitch itself. This tests whether a change makes the rest of the arsenal play up. This is a separate, distinct feature from the Tunneling component discussed earlier, which was tested for PAOM Score specifically and excluded from it. Arsenal synergy lives inside the recommendation engine instead. Both of these features are used only when there's enough real data in a given comparison group to support them, falling back to the two core predictors (y_before and $dropped\_usage\_pct\_before$) otherwise, so neither is present in every single prediction. Stage 1 pitch-quality prediction is the second-stage model, already introduced above, that predicts a brand-new pitch's own quality for a pitcher who's never thrown it. Swap usage-magnitude term is the dropped-pitch's own usage share before the swap, $dropped\_usage\_pct\_before$.

Feature	Real result	Placebo result
New-pitch distance	87.5% sign consistency	62.5%

Feature	Real result	Placebo result
Arsenal synergy (as a predictor)	100% sign consistency	58.3%
Stage 1 pitch-quality prediction	0.448 real correlation	−0.122
Swap usage-magnitude term	0.0227 MAE	0.0235 MAE

("Sign consistency" here means: across many real, different test cases, did this feature's real effect consistently point the same direction, rather than flipping back and forth unpredictably? "MAE," mean absolute error, means simply: on average, how far off was the prediction from what really happened — a smaller number is better.)

Walk-Forward Validation

The strongest test applied in this project: models were trained only on real events through a cutoff year, then tested only on real events from years never seen during that training — a true test of predicting the future, not just explaining the past.

Model	Real MAE improvement over baseline	Ranking concordance
Single-change (ADD/DROP/USAGE)	~5–9%	64–71%
Swap predictions	~37%	78.9%

Ranking concordance answers a specific, concrete question: if you give the model two different real possible changes a pitcher could make, how often does it correctly identify which one actually turned out better in reality? A coin flip would get this right 50% of the time by pure chance — every model here does meaningfully better than that, especially the swap model. In practical baseball terms: a coach using this tool to choose between two real candidate changes for a pitcher, and picking whichever one the model ranks higher, would land on the actually-better real choice roughly 64–71% of the time for a single change, and closer to 79% of the time for a swap — a meaningful edge over a coin flip for the specific question "which of these options is better," even before knowing exactly how much better.

The MAE improvement numbers answer a different, harder question: not which option is better, but exactly how many real points of xwOBA a specific change is likely to move the needle. Here the honest picture is more modest — the single-change models' predictions landed only about 5–9% closer to real outcomes than simply assuming no change at all would have. That's a genuine improvement, not nothing, but a coach reading a specific predicted number should treat it as a rough, directional signal, not a confident point estimate. The model is far more sure a change will help than it is about precisely how much. The consistent finding across this whole project: this system is considerably more trustworthy for "which option should I pick" than for "exactly how much will this help," and that's exactly why the dashboard's own ranking, not the raw predicted number, is the primary signal throughout.

An Engineering Story: The NO_CHANGE Bias Investigation

A broad, unsupervised sweep across 75 real, randomly-sampled pitchers surfaced something suspicious: the model's own "make no change" baseline never ranked first as the best recommendation — not once, in any of the 75 real cases. For something that should occasionally be the correct choice for at least a few pitchers, a 0% win rate demanded real investigation.

The cause turned out to be a structural mismatch: the NO_CHANGE option's own model is trained on every consecutive pitcher-season pair in the entire dataset, while every other candidate's model is trained only on pitchers who made a qualifying arsenal change and have a measurable follow-up season to check the result against. That's a genuine selection effect: a pitcher who got meaningfully worse is more likely to lose their role — and therefore be missing from an event-based comparison group entirely — than to simply vanish from the broader, unrestricted population NO_CHANGE draws from.

This was tested directly: the outcome distributions of both populations were compared head to head, and the event-based population showed a meaningfully better average follow-up outcome. A second, more careful check ruled out an alternative explanation — that pitchers who make changes are simply already better pitchers to begin with — by comparing each population's real baseline, before-the-fact outcomes too. Those baselines were not meaningfully different between the two groups; the real gap only appeared afterward, confirming survivorship, not pre-existing quality, as the actual mechanism. NO_CHANGE was rebuilt on the same similarity-weighted architecture as every other candidate, pulling its own comps from that same event-based population (real pitchers who made some qualifying change and have a real follow-up season) rather than the broader, unrestricted population it drew from before. It's evaluated at zero usage change instead of some nonzero amount, the same way ADD or DROP get evaluated at their own usage change, just set to zero here. This directly closes the population mismatch described above. A pitcher who got meaningfully worse and lost their role is now missing from NO_CHANGE's own comparison group too, the same way they were always missing from every other candidate's. The same standard now applies consistently across every option, not a special exception carved out for one of them.

Results

Real Top 20 PAOM Score Pitcher-Seasons, 2020-2025

Pulling the top 20 pitcher-seasons from the full 2,574-row population, across all six years rather than one per year, shows some real concentration worth pointing out.

#	Pitcher	Season	PAOM	Effectiveness	Movement	Command	Velocity
1	deGrom, Jacob	2022	97.2	98.7	39.9	99.2	99.4
2	Wheeler, Zack	2021	96.6	98.4	84.4	89.3	98.4
3	Yamamoto, Yoshinobu	2025	95.8	94.3	94.3	100.0	98.5
4	Nola, Aaron	2020	95.5	100.0	88.0	91.4	64.6
5	deGrom, Jacob	2021	94.6	99.2	48.5	99.4	79.4
6	Sale, Chris	2024	94.1	100.0	40.9	77.0	91.7
7	Cole, Gerrit	2021	94.1	97.8	96.2	79.0	97.4
8	Burnes, Corbin	2021	93.4	99.4	41.6	72.7	97.2
9	Crismatt, Nabil	2022	93.3	99.2	96.4	61.0	77.6
10	Cole, Gerrit	2022	93.3	94.7	78.1	53.8	96.6
11	Morton, Charlie	2021	93.1	97.2	77.6	79.6	94.3
12	McClanahan, Shane	2022	93.0	94.9	90.1	41.1	95.1
13	Cease, Dylan	2022	92.9	98.9	80.4	15.0	86.3
14	López, Pablo	2021	92.8	99.0	40.4	94.3	76.4
15	Ohtani, Shohei	2022	92.6	93.5	73.5	45.4	100.0
16	Smith, Cade	2024	92.6	96.2	81.4	92.6	75.1
17	Leiter Jr., Mark	2022	92.3	96.8	91.8	30.4	86.5

#	Pitcher	Season	PAOM	Effectiveness	Movement	Command	Velocity
18	Wheeler, Zack	2025	92.1	93.2	78.0	90.6	90.2
19	Rodón, Carlos	2022	91.8	95.4	81.2	73.6	83.8
20	deGrom, Jacob	2020	91.8	99.5	34.4	96.7	77.5

Jacob deGrom appears three times in the top 20, in 2020, 2021, and 2022, never finishing below 91.8. 2022 alone accounts for eight of the twenty slots, a lopsided share for one season out of six. Dylan Cease's 2022 finish stands out for a different reason: he lands at #13 with a Command score of just 15.0, among the lowest in the entire real population, propped up almost entirely by a 98.9 Effectiveness score. This is a concrete, extreme case of the same pattern noted above with deGrom's 2022 and Sale's 2024 seasons: Effectiveness's dominant weight can carry a pitcher into the top tier even when another component is severely below average.

Real 2025 PAOM Score Leaderboard

Pitcher	PAOM	Effectiveness	Command	Velocity	Movement
Yamamoto, Yoshinobu	95.8	94.3	100.0	98.5	94.3
Wheeler, Zack	92.1	93.2	90.6	90.2	78.0
Smith, Cade	91.3	97.8	92.8	55.3	85.0
Bowlan, Jonathan	90.5	95.0	76.3	90.8	55.1
Detmers, Reid	89.7	93.5	76.0	86.9	88.9
Sale, Chris	89.0	97.2	67.5	76.3	63.0
Wrobleski, Justin	88.8	92.2	82.1	89.8	34.0
Williams, Devin	88.6	98.3	96.5	31.2	63.8
Eovaldi, Nathan	88.2	86.3	95.0	95.4	59.9
Brown, Hunter	88.1	96.7	58.2	81.3	74.9

Real Recommendation Examples

ADD — Bryce Miller: recommended to add a changeup, backed by 95 real historical events (Moderate evidence tier). Top real comps: Janson Junk (2024→2025), Walker Buehler (2020→2021), Kai-Wei Teng (2024→2025).

DROP — Marcus Stroman: recommended to drop his cutter, backed by 66 real historical events (Moderate evidence tier). Top real comps: Chi Chi González (2021→2022), Johnny Cueto (2022→2023), Phil Maton (2021→2022).

SWAP — Spencer Strider: a real, Strong-evidence swap of a slider for a sweeper appears among Strider's own ranked candidates, backed by 454 real historical events. The top real comp, Gavin Williams (2024→2025), made this exact same real swap — a direct, literal precedent, not just a similarly-shaped one.

The Dashboard

Every result in this writeup can be explored directly, for any real MLB pitcher, on the live dashboard: PAOM Dashboard · Streamlit. It's organized into seven real pages, each covering a different way of looking at the same underlying data:

Data Coverage: What's Multi-Year and What's Not

One honest distinction worth being clear about before exploring further: not every part of this project has the same depth of historical coverage. PAOM Score's own four components, and the recommendation engine's underlying training data (the real 4,381 historical arsenal-change events described in Part 2) span all six real seasons, 2020–2025. But several dashboard visualizations remain 2025 season snapshots only, not yet extended across the full history: the Movement, Location, and Release Point maps, and the Similar Pitchers Map's arsenal-shape similarity type (its own on-screen disclosure states this directly). The recommendation engine follows a related pattern worth understanding precisely: its underlying model is trained on the full real 2020–2025 history, but the live, pre-computed recommendations actually shown on the dashboard are generated for 2025 targets specifically, matching PAOM Score's own current-season scope. A pitcher's recommendations reflect their current real arsenal, evaluated against the model's full historical training, not a separate recommendation set for each past season. Extending the remaining single-season visualizations to the same full historical depth is real, disclosed future work, noted again in Limitations below.

Page	What it does
Leaderboard	Year-filterable ranking of every real pitcher-season, with a confidence floor and a "best season only" toggle
Top Recommendations	League-wide, confidence-weighted ranking of the strongest real recommendations across every pitcher, filterable by change type
Pitcher Detail	One pitcher's full PAOM breakdown, real movement and location maps, complete ranked recommendation list with real historical comps, platoon splits, and arsenal evolution over time
Similar Pitchers Map	Three real similarity lenses (mechanical, arsenal shape, release point), plus the recommendation engine's own full similarity ranking
Pitcher Comparison	Two real pitchers' full PAOM breakdowns and entire arsenals, side by side
Pitch Comparison	Two specific real pitches — from different pitchers, or the same pitcher's own two pitches — compared directly, including real platoon splits
Methodology	The full real technical reference behind every number on the dashboard, with plain-language explanations built directly into the page

Putting It to Use

A few concrete examples of how these pages connect to real decisions a team might actually make:

Deciding what a specific pitcher should do next. This is the dashboard's most direct value, and it lives on Pitcher Detail: pull up any real pitcher and get a full, ranked list of real candidate changes — add, drop, increase, decrease, or swap — each with a predicted real outcome, a confidence tier, and the actual real historical pitchers that specific recommendation is built from. A coach doesn't have to take the model's word for it; the comps table shows, by name, which real pitchers made this same change and what happened to them afterward. That's the difference between "the model says so" and "here's the real precedent."

Finding a real-world comp for a pitcher with a thin MLB track record. A scout evaluating a recent waiver claim, a rookie a few starts into their MLB career, or a reliever who just changed roles can center the Similar Pitchers Map on that pitcher and see, by name, which established MLB pitchers they most closely resemble — mechanically, by arsenal shape, or by release point, depending on the question being asked. A pitcher who

mechanically resembles a durable workhorse is a different bet than one who resembles a pitcher who broke down, even at similar raw performance levels — the comp itself is the finding, not just a number.

Deciding which of two overlapping pitches to keep. A pitcher who throws both a slider and a sweeper, say, has two real, potentially redundant offerings. Pitch Comparison lets a coach set both as Side A and Side B for the same pitcher and see their real effectiveness, command, and platoon splits stacked directly against each other — turning the recommendation engine's own DROP logic into something concrete a coach could put in front of the pitcher and actually explain.

Diagnosing what actually drove a real change in performance. Pitcher Detail's Arsenal Evolution section plots a real pitcher's usage, movement, and PAOM Score across every season with real data, tab by tab. When a pitcher's performance shifted meaningfully year over year, a coach can check whether it tracks a real usage change (leaning on a different pitch more), a real shape change (the same pitch actually moving differently), or neither — and the PAOM Score Over Time tab narrows this further by showing which specific component (Effectiveness, Movement, Command, or Velocity) actually moved, rather than leaving "what changed" as a guess.

Checking for a real, visible mechanical tell. A pitching coach concerned a pitcher might be tipping pitches — giving hitters an early, visible cue before the ball is even released — can pull up the Release Point Map on Pitcher Detail and see directly whether different pitch types are actually releasing from meaningfully different points. A real, visible gap between two pitch types' release points is a concrete lead worth investigating further; several pitch types converging tightly on nearly the same point, on the other end of the spectrum, is a real, positive sign of strong tunneling.

Limitations and Future Work

- Modest exact-magnitude accuracy. Across every model in this project, ranking concordance is consistently strong while exact predicted magnitude is only modestly better than a naive baseline. This system is considerably more trustworthy for "which option should I pick" than for "exactly how much will this help," and that distinction is disclosed directly on the dashboard rather than smoothed over.
- No live track record yet. Every validation in this project is a real backtest against historical data. No recommendation has yet been taken by a real decision-maker, followed forward, and checked against what actually happened next.
- Single walk-forward cutoff. Both the single-change and swap models were validated at one train/test split. A more exhaustive validation would repeat this at 2–3 different cutoff points to confirm the result isn't sensitive to exactly where the split falls.
- Arm angle's sparse real coverage. Real arm-angle data exists for only about 20% of pitcher-seasons, so this factor only contributes to the similarity calculation when both sides of a comparison happen to have real data for the relevant season.
- Movement, location, and release data are single-season. Several dashboard visualizations are not yet year-parameterized the way PAOM Score and the recommendation engine's core data are — a real, disclosed gap and the most substantial remaining piece of infrastructure work.

Works Cited

MLB Pitcher Arsenal Evolution (2020–2025). Data collected via pybaseball from MLB Advanced Media Statcast. Kaggle. https://www.kaggle.com/datasets/yasunorim/pitcher-arsenal-evolution-2020-2025.

Baseball Savant. Statcast pitch-level tracking data, 2020–2025 MLB seasons. MLB Advanced Media, LP. https://baseballsavant.mlb.com.

pybaseball. Python package for scraping and retrieving baseball data from Baseball Savant, Baseball Reference, and FanGraphs. Used throughout this project for all raw and season-level Statcast pulls. https://github.com/jldbc/pybaseball.

scikit-learn. Pedregosa, F. et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research 12 (2011): 2825–2830. Used for the Ridge regression underlying Effectiveness, the PCA underlying Movement and Velocity, the Gaussian Mixture Model underlying Command, and the linear regression and nearest-neighbor components of the recommendation engine. https://scikit-learn.org.

pandas and NumPy. Core Python libraries used throughout this project for data aggregation, cleaning, and numerical computation.