Weekly NFL Fantasy Point Projection using PySpark

Ryan Quinlan
College of Computing & Informatics, Drexel University

Abstract—A player's fantasy points are a deterministic function of a game's box score, so projecting them before kickoff must rely only on prior information. This project builds a leakage-free weekly fantasy-point projection for the four skill positions (QB, RB, WR, TE), implemented end-to-end in PySpark over nine seasons (2016–2024) of NFLverse data. Every feature is derived from strictly prior games: rolling measures of a player's form and usage, and an engineered opponent-strength feature built from play-by-play expected points added (EPA) allowed, split into pass and run defense, rolled leakage-free, and joined position-appropriately. Three models were compared on a season-based split (train < 2023, validate 2023, test 2024): linear regression, gradient-boosted trees, and random forest. All three converge to $R^2 \approx 0.355$ and RMSE ≈ 6.5 on the held-out 2024 season, indicating a largely linear relationship; the engineered matchup feature ranks fourth of 43 by importance. Measured by within-week ranking, the metric a fantasy user cares about, the model attains a mean Spearman correlation of 0.617 and correctly identifies about seven of the top ten players per position.

1 INTRODUCTION

This project builds a machine-learning model to project how many fantasy points an NFL player will score in an upcoming game, using PySpark for distributed data processing, feature engineering, and modeling. It focuses on the four main skill positions: quarterback (QB), running back (RB), wide receiver (WR), and tight end (TE). These account for most fantasy scoring and behave consistently enough week to week to be predictable. Kickers and team defenses are excluded: they are added and dropped constantly during the season and are far noisier to model.

A single constraint shapes the entire design. A player's fantasy points for a game are a fixed linear function of that game's box score, verified to reconstruct exactly for all 49,161 player-games. Predicting points from the same game's statistics would therefore be circular. The real task is to project a player's points *before* the game is played, using only information available beforehand: recent form, role in the offense, and the strength of the defense being faced. Every feature is built from strictly prior games, where window frames end at the game before kickoff, so the model never sees information that did not exist at prediction time.

2 DATA SOURCE AND INGESTION

Data comes from the nflverse project [1], an openly licensed (CC-BY 4.0) community source of NFL data. Rather than a packaged wrapper, the pipeline pulls raw season Parquet files directly from the nflverse-data GitHub releases: play-by-play, weekly player statistics, rosters, snap counts, schedules, and team information, covering the 2016–2024 seasons.

The ingestion layer solves a practical problem: nflverse sometimes writes the same column with different numeric types across seasons (an integer one year, a decimal the next), which breaks a multi-season load. The loader detects these type conflicts and harmonizes them by promoting to double where lossless, falling back to string otherwise. It then writes a single season-partitioned Parquet "lake." Downstream reads are then clean and fast, with no per-season schema merging. All ingestion, feature engineering, modeling, and evaluation are performed in PySpark; visualization uses matplotlib on small aggregated results collected to the driver, as Spark has no native plotting.

3 EXPLORATORY DATA ANALYSIS

Exploration focused on de-risking the modeling rather than producing decoration: each analysis answers a question that informs a downstream choice.

3.1 Target distribution

Weekly PPR (Points Per Reception) points are zero-inflated and right-skewed at every position (Figure 1). QB output is bimodal, a spike at zero for non-starters and early exits, then a bell-shaped hump around 15 points for players who start and finish. RB, WR, and TE show a large mass near zero decaying into a long right tail of boom games. The descending means track the fantasy hierarchy exactly: QB 14.0, RB 9.0, WR 8.6, TE 6.2. This zero-inflation and heavy tail are the fundamental reasons single-week error stays high and R^2 is capped.

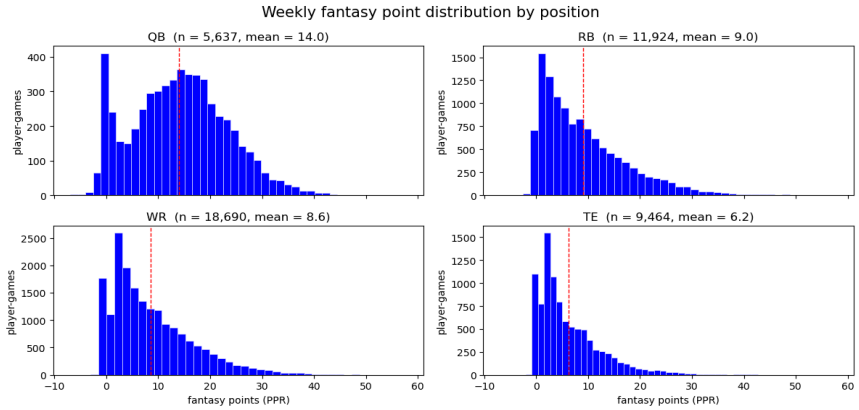

Figure 1: Weekly PPR fantasy point distribution by position (2016–2024).

3.2 Predictive value of prior output

Plotting a player's prior five-game mean against their actual output (Figure 2) gives a clear positive relationship (Pearson r = 0.545) with wide vertical scatter. The scatter is the story: two players with identical recent form routinely post very different single-week results, and a flat row of points sits along zero so established players who were scratched, game-planned out, or exited early. Recent form is the strongest single signal, but explains only part of the variance, setting a realistic ceiling for this project.

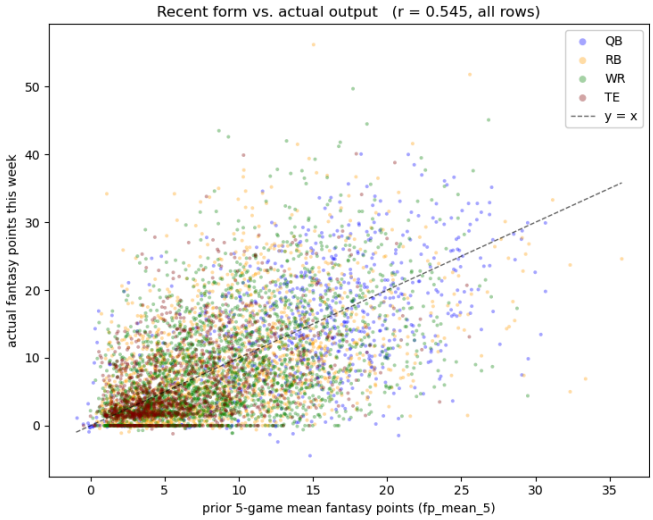

Figure 2: Prior five-game mean vs. actual weekly output (r = 0.545).

Splitting the autocorrelation by position (Figure 3) confirms two things: a five-game average predicts better than a single prior game everywhere, averaging cancels week-to-week noise and RB is the most predictable position (fp_mean_5 r = 0.525), TE the least (0.464), foreshadowing the model's later accuracy by position.

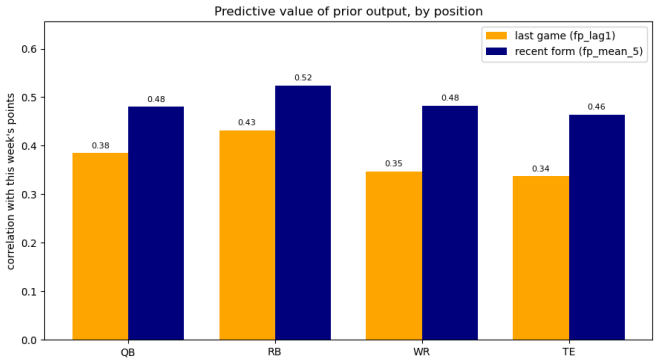

Figure 3: Correlation of prior output with current-week points, by position.

4 DERIVING OPPONENT STRENGTH

Opponent strength is the main modeling contribution of this project. It is constructed from play-by-play data by aggregating each defense's expected points added (EPA) allowed per play [2], split into pass defense and run defense. "EPA allowed" measures the expected points opposing offenses generated against a defense: positive values mean the defense gave up efficient plays (weak), negative values mean it suppressed them (strong).

Ratings are aggregated to the game level first and only then rolled over prior games, so a high-play shootout does not outweigh a low-play grind. League-wide, pass EPA allowed (mean +0.028, std 0.317) sits above run EPA allowed (mean -0.050, std 0.214) thus passing is the more efficient attack (Figure 4). Critically, when single-game noise is averaged to the season-team level, the spread shrinks from 0.317 to 0.104 but does not collapse. That persistence is evidence that defensive quality is a real, stable property rather than randomness, which makes a rolling defensive rating usable as a feature.

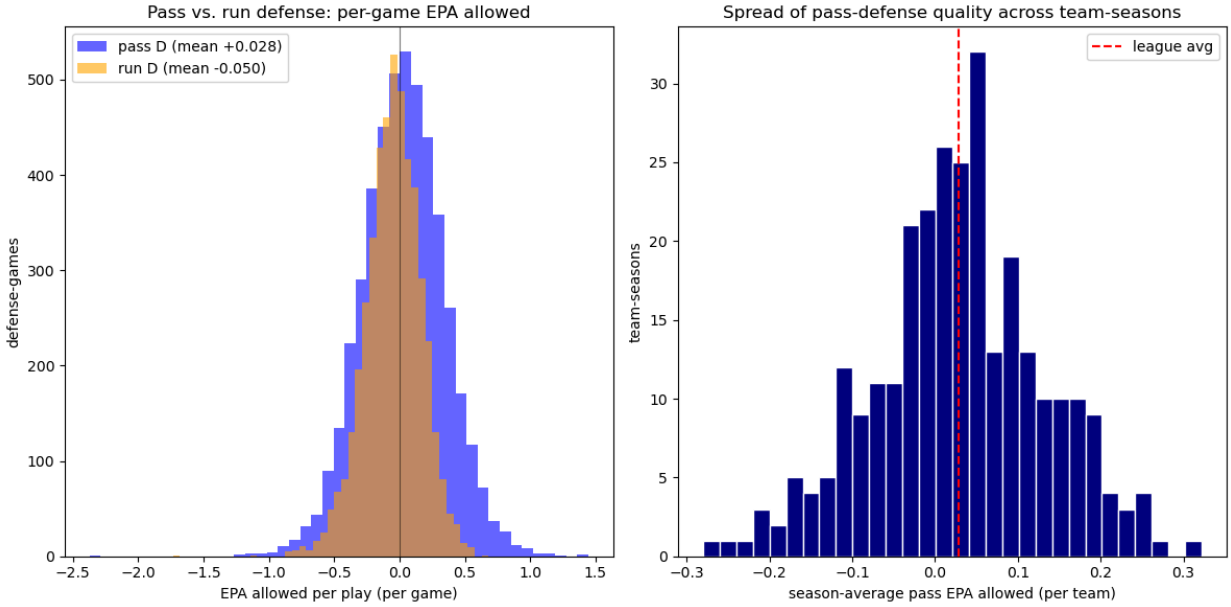

Figure 4: Defensive EPA allowed: per-game pass vs. run distributions (left) and the persistent spread of pass-defense quality across team-seasons (right).

To keep the feature leakage-free, a defense's rating for a given game is computed only from that defense's prior games, using the same window discipline as the player features. The rating is joined onto each player-week by opponent and applied position-appropriately: passing-game players (QB, WR, TE) face the opponent's pass-defense rating, running backs the run-defense rating. The join preserves the row count exactly (41,459 before and after), confirming no fan-out.

5 FEATURE ENGINEERING AND SELECTION

The modeling table has one row per player-season-week. Features fall into three groups, all derived from strictly prior games plus pre-game context:

- **Rolling target history:** 3 and 5-game means, standard deviations, and maxima of prior fantasy output, plus career mean, games played, and last-game value.
- **Rolling usage:** 3- and 5-game means of volume and efficiency signals (carries, targets, receptions, attempts, target share, air-yards share, WOPR, passing/rushing/receiving yards and EPA).
- **Matchup and context:** the position-appropriate opponent defensive rating over 3-game, 5-game, and season-to-date windows, plus home/away, week number, and early-season flags.

Feature selection uses an **allowlist**, not a denylist. The raw player statistics table carries roughly twenty same-week box-score columns; forgetting to exclude even one would silently reintroduce leakage. The allowlist admits only columns derived from prior games, plus the explicitly-added matchup features and safe context so 43 features total. A complete-case check confirms zero rows are lost to nulls.

A correlation scan (Figure 5) serves as an empirical leakage check: the strongest feature-target correlation is 0.545 (fp_mean_5), nowhere near the ≈1.0 a leaked same-week column would produce. The heatmap also reveals heavy multicollinearity among the rolling means which explains why the tree ensembles fail to beat a linear model.

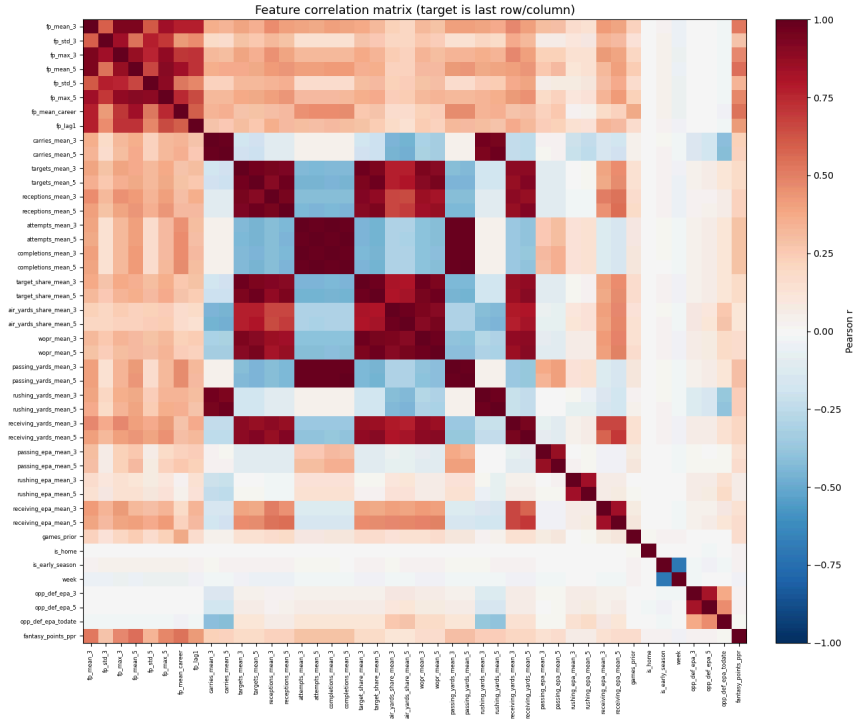

Figure 5: Feature correlation matrix (target is the final row/column). No pre-game feature approaches the correlation a leaked column would show.

Coverage analysis (Figure 6) quantifies the cost of requiring at least three prior games before a player is modeled: 90.7% of skill player-weeks are retained overall (41,459 of 45,715). Losses concentrate in early weeks and in 2016, the first season, which has no prior-year history (69.8% retained versus 91-94% thereafter).

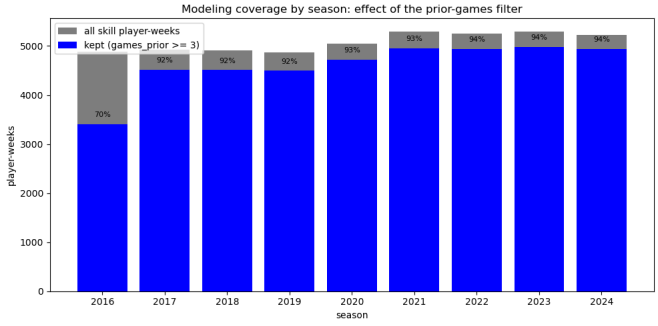

Figure 6: Modeling coverage by season before and after the prior-games filter.

6 MODELING AND EVALUATION

Data is split **by season, never randomly**: a random split would let future games inform predictions about the past. Seasons before 2023 form the training set (31,542 rows), 2023 is validation (4,981 rows), and 2024 is held out for testing (4,936 rows). Three models are compared: linear regression as the baseline, and two ensembles being gradient-boosted trees and random forest to look for non-linear interactions.

6.1 Held-out results

All three models converge to essentially the same held-out performance (Table 1) at roughly 0.355 R^2 and 6.5 RMSE. The ensembles do not beat the linear baseline. Gradient-boosted trees fit the training data better (train R^2 0.411), but that advantage does not survive to the test season (0.332), a classic overfitting scenario. Random Forest, using bagging rather than boosting, controls that gap and lands level with linear regression. The relationship is predominantly linear and additive, so the simplest model is the right choice for parsimony and generalization.

Model	RMSE	R^2	Spearman
Linear Regression	6.490	0.355	0.617
Gradient-Boosted Trees	6.602	0.332	0.603
Random Forest	6.489	0.355	—

Table 1: Model performance on the 2024 held-out season.

6.2 Feature importance

Gradient-boosted-tree importances confirm recent form dominates: fp_mean_5 and fp_mean_career are strongest by a wide margin. The engineered matchup features nonetheless earn their place as all three rank in the top eleven of 43, with the season-to-date opponent rating (opp_def_epa_todate) ranking fourth overall. The ordering within the three is informative: the longest, least-noisy window ranks highest (4th), then 5-game (8th), then 3-game (11th), mirroring the finding that longer averaging separates true defensive quality from single-game noise.

6.3 Within-week ranking

RMSE penalizes missing an exact point total, but a fantasy user needs the correct ordering within a week, not exact totals. Mean within-week Spearman rank correlation captures this: linear regression scores 0.617 and gradient-boosted trees 0.603 on 2024 so the model ranks players within a week far more reliably than the modest R^2 alone suggests.

7 RESULTS: PROJECTED VS. ACTUAL

Ranking the model's predicted per-game output within each position and comparing against the actual top-10 finishers (minimum eight games) shows the model correctly places, on average, 7.25 of the top 10 per position (Figure 7): 7/10 QB, 9/10 RB, 6/10 WR, 7/10 TE. Running back is the most volume-driven, stable position and is predicted best. Wide receiver is the most volatile and is hardest, exactly as the EDA anticipated.

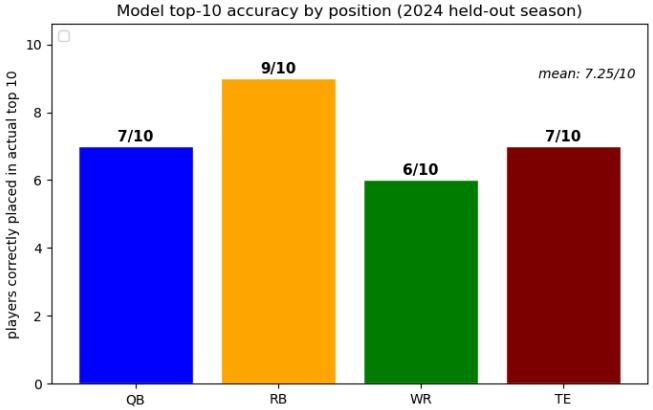

Figure 7: Actual top-10 finishers per position placed in the model's own top 10.

Plotting projected against actual per-game output for every player (Figure 8) shows both the model's strength and its failure mode. Season-averaged predictions correlate strongly with actual output (r = 0.86 QB, 0.93 RB, 0.89 WR, 0.92 TE), far higher than the weekly R^2, because averaging a season cancels most week-to-week noise. The two measures are the same model viewed at different time scales.

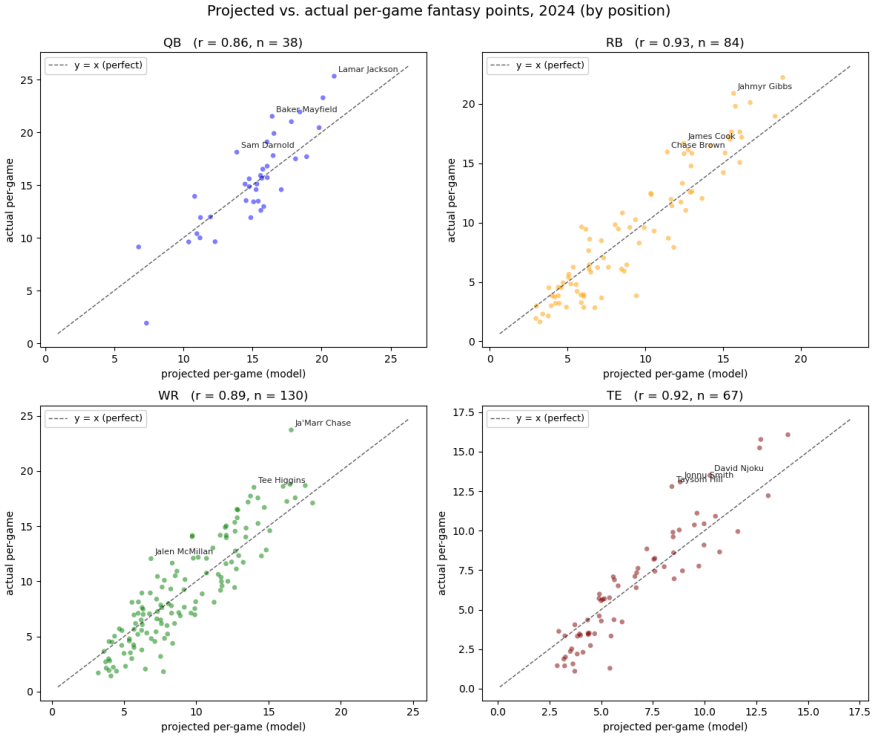

Figure 8: Projected vs. actual per-game fantasy points, 2024, by position. Points above y = x are under-predicted; the shallow tilt is regression-to-the-mean compression.

The clouds tilt slightly flatter than the diagonal: the highest-actual players (Lamar Jackson, Jahmyr Gibbs, Ja'Marr Chase) sit above the line, meaning the model under-predicts stars. This is regression to the mean and a model trained to predict the conditional average will not forecast a career year. This is why RMSE stays around 6.5 even though ranking is strong. The most visible outlier, Ja'Marr Chase, was projected near 16.6 per game and finished near 23.7. Such misses cluster into two structural blind spots: breakout seasons prior form cannot anticipate, and rookies with little history.

8 LIMITATIONS AND FUTURE WORK

- **No injury or availability signal.** The model cannot foresee a lost season; per-game evaluation with a games-played floor mitigates but does not remove this.
- **Rookies and role changes.** Players with little history are hard to project (rookies in particular). A draft capital or snap-share trajectory would help.
- **Compression.** The model under-projects elite ceilings; quantile modeling could better capture upside.
- **Unified matchup column.** A richer position-by-defense interaction may add signal for pass-catchers specifically.

9 CONCLUSION

This project delivers a leakage-free weekly fantasy projection pipeline built end-to-end in PySpark. Its central contribution is a matchup-aware opponent-strength feature from play-by-play EPA, leakage-free and applied position-appropriately, that proves a real top-ranked signal on top of form and usage. Across a linear model and two ensembles, held-out performance converges at $R^2 \approx 0.355$ and RMSE $\approx$ 6.5, showing the relationship is largely linear. Measured by within-week ranking, the model attains a Spearman correlation of 0.617 and identifies about seven of the top ten players per position, with errors falling exactly where the data says they must: unforeseeable breakouts and players without prior history.

REFERENCES

[1] nflverse. nflverse-data: NFL data (play-by-play, player stats, schedules). https://github.com/nflverse/nflverse-data. CC-BY 4.0. Accessed 2025.

[2] B. Burke. Expected Points and Expected Points Added explained. Advanced Football Analytics. https://www.advancedfootballanalytics.com.