**Summary (abstract)**

I built a season-aware ML stack that converts public performance, biometric, and market data into a dollar-scaled forecast of free-agent contracts, normalising each deal by the 25 %/30 %/35 % CBA max tier (*AAV ÷ Max-Cap Tier*) so rookies and ten-year veterans sit on one axis. The final CatBoost model—tuned with Optuna inside a five-fold, forward-chaining split—lands at RMSE 0 .138 (≈ $21.4 M), MAE 0 .088 (≈ $13.6 M), and R² 0 .622 on the 2025 hold-out season; those errors under-cut the league’s $13.9 M mean salary and sit only 1.7 × above the $8.0 M median [Basketball-Reference.com](https://www.basketball-reference.com/contracts/players.html?utm_source=chatgpt.com) while explaining roughly 62 % of contract variance. Although still shy of the 0.76–0.97 R² corridor reported by recent Random-Forest and RFAR ensembles [NHSJS](https://nhsjs.com/2025/computational-analysis-of-nba-players-with-machine-and-deep-learning/?utm_source=chatgpt.com)[ijcsm.researchcommons.org](https://ijcsm.researchcommons.org/ijcsm/vol6/iss3/1/?utm_source=chatgpt.com), the model raises our own benchmark by six points and delivers a reproducible baseline for live roster calculus.

1 Data Assembly & Feature Set

1.1 Primary sources

* Spotrac – contracts, taxes, cap space, max/min-tier tables, 2010-25.
* NBA API – box, advanced, Synergy play-types, hustle/defence dashboards, 1996-25.
* Basketball-Reference – VORP, BPM, Win Shares, plus 2025-26 salary aggregates: mean $13.90 M, median $7.97 M [Basketball-Reference.com](https://www.basketball-reference.com/contracts/players.html?utm_source=chatgpt.com).
* Kaggle injury log (pieced together from 2 sources in Kaggle with real time update injury source DAG from nba.com)– 1951-2025 events [Kaggle](https://www.kaggle.com/competitions/nba-players-salaries?utm_source=chatgpt.com).
* Wikipedia for player nicknames so we didn’t miss any by their nicknames (nah’shon hyland for example)

1.2 Market-tier encoding

Markets are bucketed by Nielsen reach and historical spend: *Big* (NYK, BKN, LAL, LAC, GSW, BOS, CHI, MIA, PHI, DAL) vs *Small* (remaining 20). The one-hot flag captures the documented spending premium in big hubs [Bryant Digital Repository](https://digitalcommons.bryant.edu/cgi/viewcontent.cgi?article=1039&context=honors_mathematics&utm_source=chatgpt.com).

1.3 Target

AAV\_pctMaxCap = AAV / MaxCap\_serviceTier; set up to have AAV to the max cap based on the players experience set to the max for each player at 25 %/30 %/35 % tiers, anchored to the changing cap by season/ Retrieved from the maximum salary table in Spotrac and set up to join the players by experience (0-6/7-10/10+) and season so we standardize the salary cap and different sized contracts of players.

A close-up of a graph

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

2 Modelling Protocol

* Pipeline ColumnTransformer → OneHot / Ordinal / Scaler → CatBoostRegressor; artefacts versioned in MLflow.
* Validation Five-fold forward-chaining (t-1 seasons train, t test) blocks look-ahead leakage.
* Hyper-search Optuna (100 trials, TPE) on depth, LR, L2, subsample.

| **Model** | **RMSE** | **MAE** | **R²** |
| --- | --- | --- | --- |
| CatBoost (final) | 0.138 | 0.088 | 0.622 |
| LightGBM | 0.146 | 0.089 | 0.583 |
| XGBoost | 0.153 | 0.091 | 0.542 |
| Random Forest | 0.165 | 0.099 | 0.463 |
| ElasticNet | 0.168 | 0.108 | 0.445 |
| Mean target | 0.184 | 0.123 | 0.000 |

Appendix:

Data sources:

Spotrac: taxes, extensions, player contracts, minimum/maximum contract thresholds

* We got the contracts, the minimum to filter out two way players and we got the maximum to set up as our denominator for AAV (y variable,) so we could standardize to the max the players based on their years can get:
  + 25% for 0-6, 30% for 7-9, and 25% for 10 – 25
  + Two way players would be their own analysis or Bayesian hierarchical modelling so the partial pooling can hierarchy (by season/league/etc.) could lend the mean towards those with less volume and more volume.
  + Rookies should be the ending of a college modelling experiment so we don’t loop them in with the 0-6 year players as much. It could include past players first year in the nba so we can try to predict the possibility of rookie of the year (and the top finishers.)
* Nba api for all the basic stats
  + Basic stats:
    - Career game log
  + Used heavily throughout for canonical player and team identity resolution:
    - commonallplayers: Retrieves player directory information (names, IDs) per season—used to build normalized player lookups.
    - commonteamyears: Retrieves team metadata, used to construct team directory (full name, nickname, abbreviations).
    - playercareerstats / commonplayerinfo: Used as fallbacks to infer a player’s team for a season or current team when other mappings are missing.
    - Bulk lookup pipeline (\_fetch\_player\_team\_mappings\_from\_api, load\_player\_season\_team\_map, etc.): Efficiently builds/refreshes season/player→team mappings and caches them in DuckDB + parquet.
    - b. Wikipedia (Nickname Data)
    - Scrapes (or regex-parses if BeautifulSoup is unavailable) the “List of nicknames in basketball” page to extract player nickname associations.
    - Used to augment the player directory so that alias/nickname-based resolution (e.g., “Greek Freak” → Giannis) is possible.
* Nba efficiency metrics:
  + nba\_api.stats.endpoints.playerestimatedmetrics.PlayerEstimatedMetrics.
* Basketball reference for advanced metrics
* 1951-2023 Kaggle injury data + 2016 – 2025 injury data + (new injury scraping from NBA.com so I can create my own new rows, DAG ready to fill in the season with it automated to the months of the year.)
* Defensive metrics:
  + LeagueDashPlayerStats: advanced per-game stats (e.g., defensive rating, counting stats, usage, etc.). This is the “ADV” block in your merge. You’re filtering to NBA teams when enforce\_nba\_only is on, deduping per (PLAYER\_ID, season), normalizing IDs, and applying optional minutes filters.
  + LeagueHustleStatsPlayer: “Hustle” stats such as deflections, contested shots, loose ball recoveries, screen assists, etc. These effort/intangible defensive metrics started being systematically collected around the 2016–17 season (hustle award introduced then), so missing values before that are expected coverage gaps, not bugs. You’re also merging in D\_FG\_PCT from:
  + LeagueDashPtDefend: provides defensive field goal percentage (“PTDEF”) tied to on-ball defense.
* Playtypes:
  + nba\_api.stats.endpoints.synergyplaytypes

A screenshot of a computer

AI-generated content may be incorrect.

**\*\*preprocessing\*\*A screenshot of a graph

AI-generated content may be incorrect.**

A close-up of a graph

AI-generated content may be incorrect.

A close-up of a graph

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.

**Key findings summary:**  
The target variable AAV\_PCT\_OF\_MAX is highly right‐skewed (skewness 2.34, kurtosis 5.45) with a median of 0.067 and IQR [0.033–0.203], and about 10% of values flagged as potential outliers. Data quality is excellent—97.4% complete, 100% unique rows, and only 44 columns with >10% outliers. Feature importance analysis highlights contract‐usage metrics (e.g. MIN\_RANK, PTS\_RANK, USG%) as strongest predictors. Among feature categories, General (e.g. PLAYER\_POSS r = 0.617), Scoring (PTS r = 0.618), and Usage (USG%, TRUE\_USAGE% r = 0.617) show the highest correlations with our target.

React Site Started:

A screenshot of a computer

AI-generated content may be incorrect.