Modeling NBA 2nd Half Points Totals

Daniel Klausz, Duncan Rawlings, Kento Takeda
Milestone II SIADS 696
Fall 2025

Introduction

We wanted to try predicting whether NBA games would go over or under the second half points total set by sportsbooks. The idea started from a shared curiosity in the NBA that originally got our team connected. After some discussions and discovering one of our team members had experience sports betting professionally and that a lot of the betting volume was placed on second half points total over/under bets, this provided a glimpse that there were potential inefficiencies that could be capitalized on by leveraging a machine learning model using first half data and other contextual information.

We are using supervised learning to predict the actual second half points total for a given game and set of parameters and we are using unsupervised learning to attempt to uncover patterns and groupings in the data without reference to second half totals. Using dimensionality reduction and clustering techniques, we explore natural groupings of games with similar characteristics. This could end up providing insight on positive expectation investments.

If this model successfully predicts second half point totals better than sportsbooks, it could reveal systematic pricing inefficiencies. We need approximately a 52.38% success rate to overcome the standard -110 odds. Any advantage beyond this, even a few percentage points, would yield a profitable betting strategy and demonstrate that second half totals are systematically mispriced. This level of predictive accuracy has real world sports analytics potential. Any edge in modeling game dynamics is valuable, and since NBA teams have substantial resources at their disposal, proven innovations in predictive analytics would be quickly adopted for strategy and player evaluation.

Related work

We looked at the following projects that tried similar things:

Title: NBA Game Outcome Prediction Project
URL: https://www.kaggle.com/code/khushbukakdiya/nba-game-predictions
This project aims to predict game outcomes. Ours differs because we focus on in game second half predictions and specifically on total points instead of outcomes.

Title: NBA Sports Betting Model
URL: https://www.kaggle.com/code/perry613/nba-sports-betting-model

This project is explicitly trying to gain experience with regression models instead of trying to out predict the sportsbooks betting lines. It is similar in that it focuses on over/under points totals but ours is different since we are only looking at in game second half points totals.

Title: NBA_Betting
URL: https://github.com/NBA-Betting/NBA_Betting?tab=readme-ov-file
This project's goal is to create a profitable system for predicting the outcome of games. We have a similar goal except we are focused on second half points totals.

Data Sources

We used multiple endpoints from the official NBA Stats API, primarily through the nba_api Python package. Initially, we were impressed by the utility of the NBA API and its well structured data architecture. The API offers numerous endpoint calls with thoughtfully designed keys for games, players, referees, teams, and other entities, making it straightforward to understand the data relationships and structure our queries.

However, the implementation proved far more challenging than expected. We attempted to optimize our requests using sleep commands and rate limiting strategies, but pulls would still fail seemingly at random, regardless of timing. There was no consistent pattern to the failures, no clear error messages indicating what went wrong, and no obvious way to predict which requests would succeed or fail.

This forced us into a trial and error approach. Rather than trying to perfect a single request strategy, we built a multi pass system that automatically retries failed pulls and saves progress after each successful pass. The script continuously cycles through missing data, retrying requests until every game is successfully retrieved. This ended up being the most practical solution for our needs. Instead of manually monitoring and restarting failed pulls, our script simply persisted until it had complete coverage. Using this approach, we collected seasons 2020-2021 through 2023-2024, covering 7,807 unique games.

Data came from:

- **LeagueGameFinder (nba_api)** - all game IDs and metadata
- **BoxScoreTraditionalV2 (nba_api)** - team and player stats for both first halves and complete games
- **BoxScoreSummaryV2 (nba_api)** - contextual stuff like referees, points by quarter, etc.
- **NBA Betting Data | October 2007 to June 2025 (Kaggle)** - historical betting lines for second half totals

Each of the nba_api endpoints returns multiple tables. Some didn't include the GAME_ID, so we patched that in manually. The data was saved as a CSV after every successful pass to protect against crashes. We also included rest and travel context, referee info, and head to head history. The end result was a clean and unified dataset ready for modeling that, after merging all the data, has no missing games. This is likely

the result of the NBA making its data available as the Kaggle data likely also used the NBA API illustrating the benefits of sound data structure.

Feature engineering

Feature engineering was easily the messiest but most rewarding part of the project. We started simple, but it quickly turned into a monster with over 1,000 columns.

We built features on multiple levels:

- **First Half Stats -** team pace, efficiency, turnovers, rebounding, assists
- **Second Half Derived Stats -** computed by subtracting first half numbers from full game totals
- **Player Level Rolling Windows -** 7, 14, and 30 day averages for top rotation players, tracking minutes, shooting percentages, and contribution metrics
- **Team Level Rolling Windows -** team shooting percentages, assist rates, turnover trends, win rates, etc., over the same time windows
- **Contextual Features -** rest days, back to backs, travel streaks, home/away stretches, and referee IDs
- **Opponent and Head to Head Stats -** records versus that opponent, point differentials in last meeting, days since last matchup

The feature engineering process was messy but productive. We kept brainstorming new angles like asking questions about rest advantages, shooting variance, home stand length, etc. Each idea led to more features, and the code ballooned accordingly. Then we realized we needed all the same stats for the opposing team as well, which essentially doubled our feature set and added another layer of nested loops. At one point, we had loops within loops for time windows, player groups, team stats, and opponent stats, which made the code complex and slow. We realized too late that many of these loops should have been refactored into cleaner functions. If we redid this project, we would definitely restructure the code from the start. Still, this version gave us a comprehensive dataset that reflects what bettors and sportsbooks actually consider when setting lines.

Part A. Supervised Learning

Methods description

We implemented a supervised regression framework to predict second half point totals using three seasons (2020-2021 through 2022-2023) for training, with the first half of 2023-2024 held out for testing. We employed walk forward prediction to respect temporal ordering and prevent data leakage. Features were engineered from prior games only, excluding player names to force generalization. Models requiring feature scaling (Elastic Net, Neural Networks) used StandardScaler fitted on training data only.

We tested five diverse model families:

1. **XGBoost (Gradient Boosted Trees) -** Chosen as our primary model since it's widely regarded as one of the most effective gradient boosting methods for high dimensional tabular data. Unlike a single decision tree, gradient boosting combines many weak trees sequentially to reduce bias and improve predictive accuracy. XGBoost's ability to capture complex non-linear relationships made it well suited for our 1300+ features.
2. **LightGBM (Alternative Gradient Boosting) -** Tested as a secondary gradient boosting framework to verify that XGBoost was the right choice. Both XGBoost and LightGBM extend traditional tree based methods, but LightGBM grows trees leaf wise (splitting the leaf with the largest gain) rather than level wise. Comparing the two allowed us to test whether XGBoost's structure offered a meaningful advantage for our data.
3. **Random Forest (Bootstrap Aggregation) -** Used as a non-gradient tree ensemble to evaluate whether sequential boosting meaningfully outperforms independent tree averaging. Random Forests improve on single trees by training many models on bootstrapped samples and averaging their results, which reduces variance and overfitting without relying on gradient updates.
4. **Multi-Layer Perceptron (Neural Network) -** Included as a non-tree, non-linear benchmark to explore a fundamentally different modeling mechanism. Neural networks use continuous activation functions and backpropagation to approximate complex functions. We tested both narrow (256 neurons) and wide (1024 neurons) architectures to see whether smooth non-linear transformations could outperform tree based methods.
5. **Elastic Net (Regularized Linear Regression) -** Served as a simple probabilistic baseline combining L1 and L2 regularization. This model helped determine whether the relationships in our data were mostly linear or required more expressive methods. A weaker performance from Elastic Net would confirm the need for more complex models like XGBoost.

XGBoost was manually tuned through iterative adjustments to tree depth, learning rate, and the number of estimators, using light regularization to balance fit and generalization. The final configuration achieved strong performance with parameters such as n_estimators=1800, learning_rate=0.018, and max_depth=8, along with moderate penalties (reg_alpha=0.2, reg_lambda=0.6) and early_stopping_rounds=120 to prevent overfitting.

Initial testing showed that the Vegas line alone outperformed our base model, so we incorporated it as a feature and trained all models to predict the residuals between actual outcomes and Vegas predictions. Using the Vegas line as a residual consistent improved model performance.

We applied what we learned from XGBoost's tuning process to set comparable parameter ranges for LightGBM and Random Forest, allowing a fair evaluation under similar conditions. However, when these models, along with Elastic Net and the neural networks, consistently performed worse than XGBoost on the residual task, we did not feel further tuning was a worthwhile investment. This confirmed XGBoost as the most effective and efficient model for our problem.

Supervised Evaluation

Evaluation Metrics

- **Mean Absolute Error (MAE)** - Primary metric. Interpretable in points, treats all errors equally, robust to outliers. Critical for betting applications where consistent accuracy matters more than occasional large misses.
- **Root Mean Squared Error (RMSE)** - Secondary metric. Penalizes large errors, useful for detecting catastrophic predictions.
- **R^2 (Coefficient of Determination)** - Normalized variance explained. Allows baseline comparison and indicates predictability ceiling.
- **We prioritized MAE** - reducing it by even 0.1 points across thousands of games translates to profitability given the 52.38% required breakeven percentage needed to overcome -110 betting odds.

Overall Results

Models evaluated on held out 2023-2024 half season test set:

Model	MAE	RMSE	R^2	Δ MAE vs Vegas	Δ RMSE vs Vegas
Vegas Line	9.738	12.618	0.124	0.000	0.000
XGBoost	**9.735**	**12.601**	**0.127**	**-0.004**	**-0.017**
LightGBM	9.745	12.611	0.125	+0.007	-0.007
Random Forest	9.747	12.620	0.124	+0.008	+0.002
Neural Network	9.806	12.722	0.110	+0.067	+0.103
XGBoost (No Residual)	10.610	13.458	0.004	+0.872	+0.840
Elastic Net	10.868	13.880	-0.060	+1.130	+1.262

Key Findings - Including the Vegas line as a residual target improved all models. XGBoost without residuals performed far worse, confirming its value. Among residual based models, tree methods clustered within 0.012 MAE, suggesting a performance ceiling. Neural networks underperformed by ~0.07 MAE, and Elastic Net's negative R^2 confirmed strong non-linearity. XGBoost slightly beat the Vegas baseline, but gains were marginal.

Feature Importance

With over 1,300 features, we grouped them into categories to understand which types of information drive predictions. We conducted this analysis on our best performing model which was XGBoost trained with Vegas residuals.

Using XGBoost's gain metric (average improvement in accuracy from each feature across all splits):

Feature Category	Cumulative Gain	% of Total
30 Day Rolling Stats	282,434	31.7%
14 Day Rolling Stats	253,681	28.5%
7 Day Rolling Stats	181,484	20.4%
Opponent Features	131,889	14.8%
Halftime Stats	28,095	3.2%
Schedule Features	2,972	0.3%
Referee IDs	2,202	0.2%

Rolling statistics across all time windows account for over 80% of feature importance, with opponent features contributing another 15%. Surprisingly, the halftime statistics which we thought was going to give us an edge only accounts for 3.2%.

Ablation

We retrained the XGBoost residual model after removing the top 5 feature groups by importance. Results are ordered by performance degradation, calculated as the percentage increase in MAE relative to the full model:

Feature Group Removed	Test MAE	Δ MAE	% Degradation
None (Baseline)	9.735	0.000	0.00
7-Day Rolling Stats	10.027	+0.292	+3.00%
Halftime Stats	9.968	+0.233	+2.40%
Opponent Features	9.939	+0.205	+2.10%
30-Day Rolling Stats	9.925	+0.190	+1.96%
14-Day Rolling Stats	9.870	+0.135	+1.39%

Key Findings:

The 7 day rolling window proves most critical with 3.0% degradation, reflecting a team's most recent performance, injuries, and lineup changes that are more diluted in the other rolling averages. Notably, halftime statistics show a striking discrepancy. Despite accounting for only 3.2% of feature gain, their removal causes 2.4% performance degradation, the second largest impact. This suggests halftime stats contribute little on average but are essential in abnormal game conditions like unexpected player absences (injuries, foul outs, ejections), unusual pace, or extreme shooting variance.

No single feature group causes more than 3% degradation, indicating predictive power is distributed across many features rather than concentrated in a few. Historical team performance being the strongest predictor explains why we barely outperform Vegas, even when training on residuals. Sportsbooks have been refining their models for decades with teams of experienced data scientists and likely have access to proprietary information. By incorporating the Vegas line as a feature, we leverage their expertise, achieving marginal improvement by identifying small systematic pricing inefficiencies. However, halftime scores alone are insufficient for a robust betting strategy. With more time, we could explore using the model to identify specific edge cases where our predictions consistently diverge from Vegas in profitable ways.

Sensitivity Analysis

We tested XGBoost's sensitivity to max_depth by varying it from 4 to 12 while holding all other hyperparameters constant:

max_depth	Test MAE	Δ MAE	% Change
4	9.736	+0.001	+0.013%
6	9.739	+0.004	+0.042%
8	**9.735**	**0.000**	**0.000%**
10	9.749	+0.014	0.144%
12	9.745	+0.010	+0.103%

The model is insensitive to max_depth, with all configurations performing within 0.15% of each other. This aligns with our ablation findings that predictive power is distributed across many features rather than concentrated in complex interactions. Predicting second half totals appears to rely primarily on aggregating historical performance metrics rather than capturing deep relationships between them. Shallow trees can effectively partition the feature space without requiring complex decision boundaries.

Tradeoffs

Model Complexity vs. Interpretability - For a betting application, we prioritize predictive accuracy over interpretability. The extreme of a black box model that

consistently identifies winning opportunities is more than acceptable. However, if we were working for an NBA team, interpretability would be critical for communicating insights to coaches and executives in order for them to make actionable changes.

Feature Engineering Depth vs. Diminishing Returns - Our comprehensive feature engineering created over 1,300 features, but many are derivatives of the same underlying data (rolling averages across different windows, player rotation aggregations). This redundancy likely saved computation time during model training by pre-calculating relationships the model would otherwise need to learn. Our ablation analysis validated this approach. No single feature group caused more than 3% degradation. Given how closely our models approached Vegas performance, we likely are close to a predictive ceiling. Further feature engineering would likely yield little.

Generalization vs. Overfitting - This is a fundamental model tradeoff. We intentionally removed player names to force the model to generalize based on aggregate statistics rather than memorizing individual tendencies. We also struggled to incorporate injury data precisely because we could not generalize well across different player absences. During hyperparameter tuning, we observed large performance drops even with parameters that should have led to overfitting, confirming that regularization (alpha=0.2, lambda=0.6, gamma=0.05) and early stopping were essential.

Failure Analysis

We analyzed our five worst predictions (errors >40 points) and found three distinct failure categories.

Overtime Games - Four of our five worst predictions were overtime games (double or triple overtime), averaging 56.3 points of error. NBA regulation quarters are 12 minutes, so two overtimes add approximately 25% more game time. Our worst prediction was a triple overtime game where the Kings beat the Lakers 141-137, adding 31% more game time that no halftime based model could anticipate.

Extreme Shooting Variance - In December 2021, the Jazz beat the Celtics 137-130 in regulation, with both teams shooting 50% from three-point range, well above typical NBA averages (35-37%). This regulation game nearly matched the scoring of our triple overtime game and produced an error of 45.7 points. The model predicts based on historical shooting percentages but cannot anticipate when teams simultaneously shoot far beyond season norms.

High Variance Close Games with Momentum Swings - Games where teams alternate major scoring runs create volatile patterns that inflate totals beyond what pace and efficiency metrics predict. Many of these games end up becoming OT games. The model assumes relatively consistent play but cannot anticipate dramatic scoring bursts.

Future Improvements - Overtime games represent irreducible noise without real time data. For shooting variance and momentum swings, we could analyze abnormal halftime game states, but there's always a risk of regression to the mean in the second

half. Our model failing in these unpredictable ways provides further evidence that we are approaching a predictive ceiling for this task.

Part B. Unsupervised Learning

Methods description

We explored two different methods, Principal Component Analysis (PCA) and K-Means clustering. PCA is a probabilistic, dimensionality reduction technique that identifies linear combinations of features that explain the most variance, by simplifying high-dimensional data while keeping critical information. We applied PCA to numeric features derived from player, team, and contextual statistics. We chose PCA because our dataset contained many correlated features, reducing dimensionality helps improve clustering performance.

K-Means clustering, a non-probabilistic method was used to segment games Into groups with similar underlying characteristics in the PCA transformed feature space. The number of clusters were selected by multiple values (6, 8, 10, 12) and evaluating them using a silhouette score. We eventually selected k=10 as it scored the highest silhouette score, while maintaining a reasonable clustering size. We chose K-Means because of its simplicity and ability to produce easily interpretable clusters.

By combining PCA and K-Means clustering, we leveraged the strengths of both methods. PCA reduces noise, while K-Means provides clear segmentation to identify patterns and similar games.

Unsupervised Evaluation

To evaluate our unsupervised workflow, we evaluated both the variance retained by PCA and the quality of K-Means clusters. PCA evaluation focused on cumulative explained variance. The first 30 components reached 99 percent of the total variance. This indicates that most information in the original features was retained while reducing dimensionality.

For K-Means, we used two complementary metrics, silhouette score and inertia. Silhouette score measures the degree of cluster separation and cohesion, ranging from -1 to 1. Inertia measures the compactness of clusters, where lower values indicate tighter clusters.

We performed clustering across multiple k values (6, 8, 10, 12) and plotted both metrics to analyze stability and tradeoffs. As shown in Figure 1, silhouette score peaked at k=10. This suggests that this configuration offered the best balance between distinctness and cluster cohesion. Inertia decreases as k values go higher, but flattening at k=10.

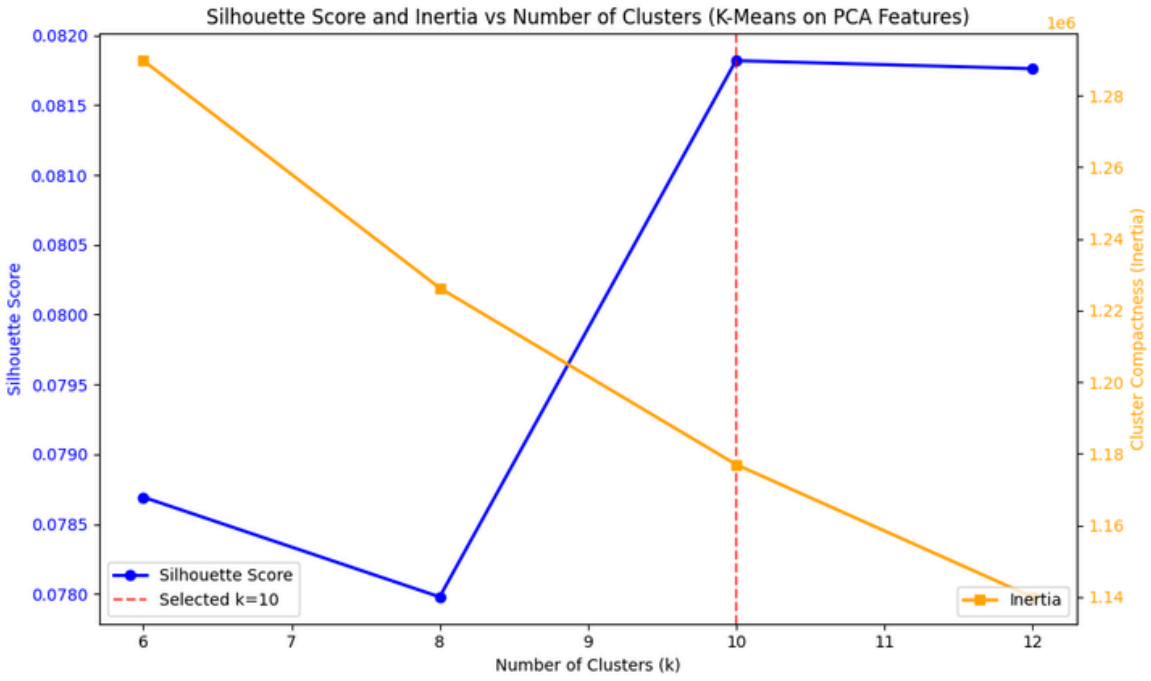

Figure 1. Silhouette Score and Inertia vs Number of Clusters

We summarize the results of our unsupervised learning analysis below. Both PCA and K-Means complement each other. PCA reduces noise, while K-Means segments data into meaningful clusters.

Method	**Evaluation Metric**	**Score / Result**
PCA	Variance retained	99% (30 components)
K-Means (k = 10)	Silhouette Score	0.0818
K-Means (k =10)	Inertia	1.177e6

However, visualizing the clusters in PCA space reveals significant limitations. Figure 2 shows the K-Means cluster assignments projected onto the first two principal components (PC1 and PC2), which together explain only 18.1% of total variance. The low silhouette score (0.082) is evident in the visualization. The clusters exhibit substantial overlap with no clear boundaries separating them. Points from different clusters (indicated by different colors) are intermixed throughout the feature space rather than forming distinct groups.

This further substantiates a fundamental characteristic of the NBA. Games are played at the highest level with billions of dollars collectively at stake, creating strong incentives for teams to converge toward optimal strategies. As teams adopt similar pace and space offenses, three point heavy shot selection, and switching defensive schemes, games become relatively homogeneous despite differences in rosters or

records. The lack of cluster separation aligns with our supervised learning findings that strategic convergence creates fundamental similarity across games that makes meaningful segmentation difficult. While K-Means successfully partitions the data mathematically, these partitions do not represent qualitatively different game types, limiting the interpretability and practical utility of clustering for this problem.

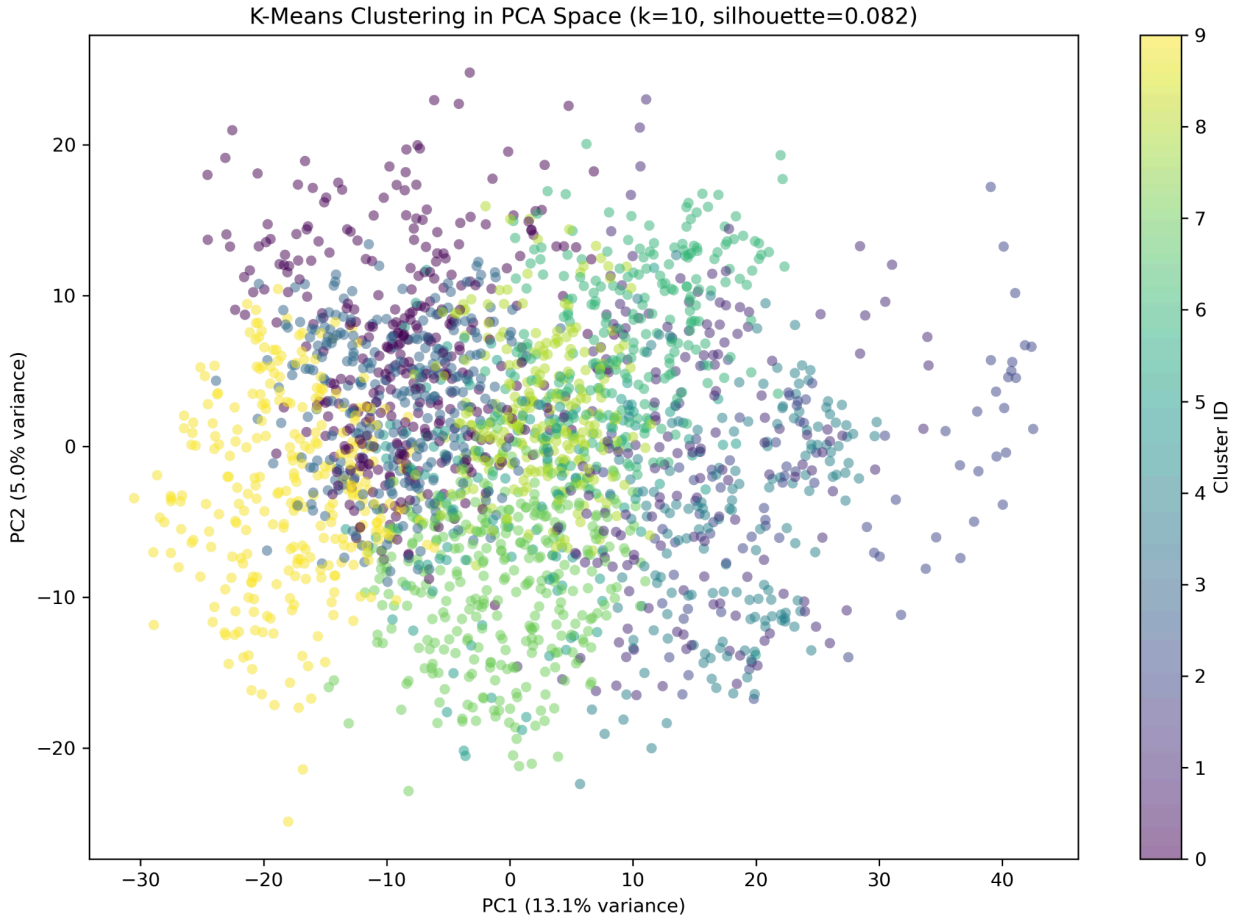

Figure 2. K-Means clustering (k=10) projected onto first two principal components, showing substantial cluster overlap

Sensitivity analysis on the number of clusters showed that silhouette scores were relatively stable between 10 and 12 clusters. Excluding rolling averages (7, 14, 30) reduced interpretability and cluster cohesion.

Discussion

What we learned from Part A

The biggest surprise was how much context mattered. Rest days and travel streaks ended up more predictive than some first half stats. We also learned that model

performance doesn't need to be massive to find positive expected value bets. In theory 52.38% success rate would be breakeven but given the kickbacks that bettors get from sportsbooks, often referred to as rakeback, these bets become profitable albeit a very small edge that would incur a lot of variance. With good line shopping across multiple sportsbooks bettors can find even more edge available. The hardest challenge was data wrangling. The modeling part was straightforward compared to the weeks we spent getting clean, complete API data.

What we learned from Part B

We liked how the clustering revealed intuitive game types and patterns in the data. We were surprised that even with complex features and PCA preprocessing, silhouette scores were relatively low. This shows how challenging it is to capture the patterns that sportsbooks use. The anomaly detection angle could definitely be refined into a screening tool for live bets. If we had more time, we'd bring in real time line movement data from sportsbooks and run PCA on that in sync with first half stats. Challenges included feature engineering for rolling statistics and selecting appropriate k-values for clustering.

Ethical Considerations

For Part A, the biggest ethical concern is the connection to sports betting. While this project is academic, models like this could be misused for gambling promotion or misrepresenting a sports betting model as a way to get variance free money, which is far from the case. Responsible betting requires knowledge and understanding of bankroll requirements and inherent variance in the game. Our stance is to focus on analytics, market dynamics, and model interpretability instead of betting solicitation. It is also important to acknowledge the potential social implications of predictive sports models. Even if they are used responsibly, models that identify inefficiencies in betting markets can indirectly influence how sportsbooks adjust their lines or how bettors behave. If we share this widely without context, such insights could amplify risky gambling behavior or create unfair information advantages. To avoid this, our project avoids releasing predictive outputs that could be repurposed for betting.

For Part B, clustering can unintentionally label teams or players in biased ways such as underperformers or volatile, so appropriate context and language should be used in public communication. Also, scraping or API overuse should respect rate limits and fair use terms. Another ethical point relates to interpretation and downstream use of unsupervised results. Clusters and groupings can easily be misinterpreted as objective truths if presented without explanation. This runs the risks of reinforcing judgments or stigmatizing individuals based on patterns that may come from hard schedules and injuries. To address this, results should be taken as exploratory insights meant to generate hypotheses, not as definitive evaluations of performance. Presenting with careful visualizations and transparent explanation of feature selection and clustering logic help ensure that results are communicated responsibly and avoid misrepresentation.

Statement of Work

Daniel Klausz:

- Pulled and merged data from nba_api
- Performed feature engineering and data cleaning
- Supervised and unsupervised learning
- Wrote and edited Final Report

Kento Takeda:

- Researched existing projects/papers to identify key trends and considerations
- Scheduled and organized team meetings
- Created visualizations
- Managed the project and submitted group work
- Wrote and edited Final Report

Duncan Rawlings:

- Organized, wrote, and edited Final Report
- Collected historical betting data
- Exploratory data analysis
- Wrote project proposal

References

Charest, J., Samuels, C.H., Bastien, C.H., Lawson, D., & Grandner, M.A. (2021). Impacts of travel distance and travel direction on back-to-back games in the National Basketball Association. *Journal of Clinical Sleep Medicine*, *17*(11), 2269-2274. https://jcsm.aasm.org/doi/10.5664/jcsm.9446

Schuhmann, J. (2021, August 23). *How rest, road trips and other factors will impact 2021-2022 schedule*. NBA.com. https://www.nba.com/news/how-rest-road-trips-and-other-factors-played-out-in-2021-22-schedule