Sequence Modeling of NBA Prediction-Market Mispricing Following Win-Probability Shocks

Authors:

Aaron McDonald
Sam Burgess
Ethan Aidam

Abstract

In this paper, we evaluate whether the rapid changes in ESPN win probability, specifically during NBA seasonal games create short-lived mispricing in Kalshi prediction markets. Kalshi is a prediction market that allows people to trade anywhere from $0.00-$1.00 contracts which represent the actual probability of an outcome. These contracts are based on events in a wide array of domains (like politics, sports, etc). We focused on sports whereas this project aimed to test whether the result of super rapid in-game changes in ESPN win probability create any sort of short-lived mispricings in Kalshi NBA prediction markets. The final python notebook contains 12,422 event-market observations spanning 46 playoff games and 92 market tickers from April 22 through May 11, 2026.

1. Background and Motivation

Prediction markets efficiently convert relatively dispersed and unstructured beliefs into prices that can be understood as simple probabilities. For instance, in the sports world, in the context of a sports game, these prices update continuously as new data points crystallize within the game. These data points are usually representative of occurrences like injuries, made/missed shots, turnovers, flagrant fouls, substitutions, clock pressure, ejections, and late-game scoring and fastbreak runs. Under these circumstances, if markets respond suboptimally in that they are too slowly or too aggressively according to these in-game events, short-lived inaccuracy can be present within the prediction market pricings (this is referred to as mispricing). This is an important concept for a plethora of professionals like traders seeking to optimize their trading strategy, or market makers managing substantial inventory risk, and/or researchers examining how quickly public information is taken into account when deciding prices. NBA games provide us with a strong setting for our question because within these intense basketball contests, timestamped shocks are constantly generated by a rich play-by-play context in real time. ESPN win probability allows us the ability to externally evaluate the measure to which an in-game event alters the expected outcome. Kalshi prediction-market candles also show how market prices respond given the same events occurred. By aligning both ESPN and Kalshi sources, we can better study if large game-state shocks are eventually followed by certain predictable short-horizon market moves. Our project focuses on whether rapid changes in win probability during current-season NBA games create statistical inefficiencies in Kalshi markets. Our earlier exploratory analysis suggested that Kalshi prices tend to often move less than the corresponding

win-probability shock, and even more so around high-impact plays. The MS4 work extends that idea by building a cleaned event-alignment pipeline and a sequence-ready dataset centered on these shocks. This allows us to move beyond descriptive mispricing labels and test whether pre-event OHLCV patterns, shock size, and market context contain predictive signal for one-minute direction and five-minute-ahead price movement. Ultimately, the project evaluates whether real-time sports prediction markets contain exploitable nonlinear structure after major informational events.

2. Problem Statement

Our project's refined research question is the following: *Given that ESPN win probability changes sporadically throughout an NBA game, can we, based on this, predict the short-horizon Kalshi market response accurately enough to correctly identify immediate directional price movements or overreactions in the market?* In our dataset for this project, each observation represents a distinct moment that happened in an NBA game and this stat is paired with the corresponding related Kalshi market. So one can imagine, in the scenario that ESPN records a major win-probability swing after a key play in a game, we would pair that play with the Kalshi market for that same exact game and then with this relation, examine how the market price moved around that specific moment in the game. For each pair (the play-market observations), we store a short price history window. This window has 20 time steps, using one-minute Kalshi market data from around the event. At each step, we record five standard market variables: opening price, highest price, lowest price, closing price, and trading volume. In total, the final dataset contains 12,422 of these event-market examples. The notebook uses this dataset to answer two main prediction questions. First, can we predict whether the Kalshi price will move down, stay mostly flat, or move up one minute after the event? Second, can we predict the actual Kalshi price one minute after the event? This changes the project from simply labeling mispricing after the fact to testing whether a model can learn patterns in market behavior around major NBA events.

3. Data and Exploratory Analysis

3.1 Data sources

For our data sources we combined two external data sources: Kalshi NBA game markets, including event metadata and 1-minute candlestick OHLCV data and ESPN NBA scoreboard/game-summary endpoints, which provide game schedules, play-by-play context, and win-probability updates.

3.2 Pipeline summary

We parse Kalshi event tickers, match them to ESPN game IDs, identify high-impact plays, fetch market candles around each shock, and convert each event-market pair into a fixed-length sequence. For dataset construction, the notebook uses a shock filter of 0.02 absolute win-probability change, while the stricter overreaction label uses a 0.05 shock threshold plus price-move and gap conditions.

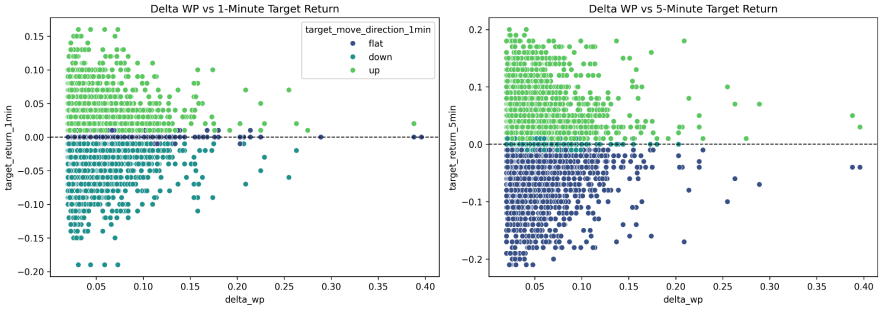
3.3 Final modeling slice

The repository's data/lstm_sequences.json file contains 12,422 saved event-market sequences. These sequences span 46 unique ESPN games and 92 Kalshi market tickers, with game dates from April 22 through May 11, 2026. Every sample has a 20-step sequence and the same ve features: price_close_dollars, volume_fp, price_open_dollars, price_high_dollars, and price_low_dollars. In tensor form, the sequence block is therefore (12,422, 20, 5).

3.4 EDA findings that influenced modeling

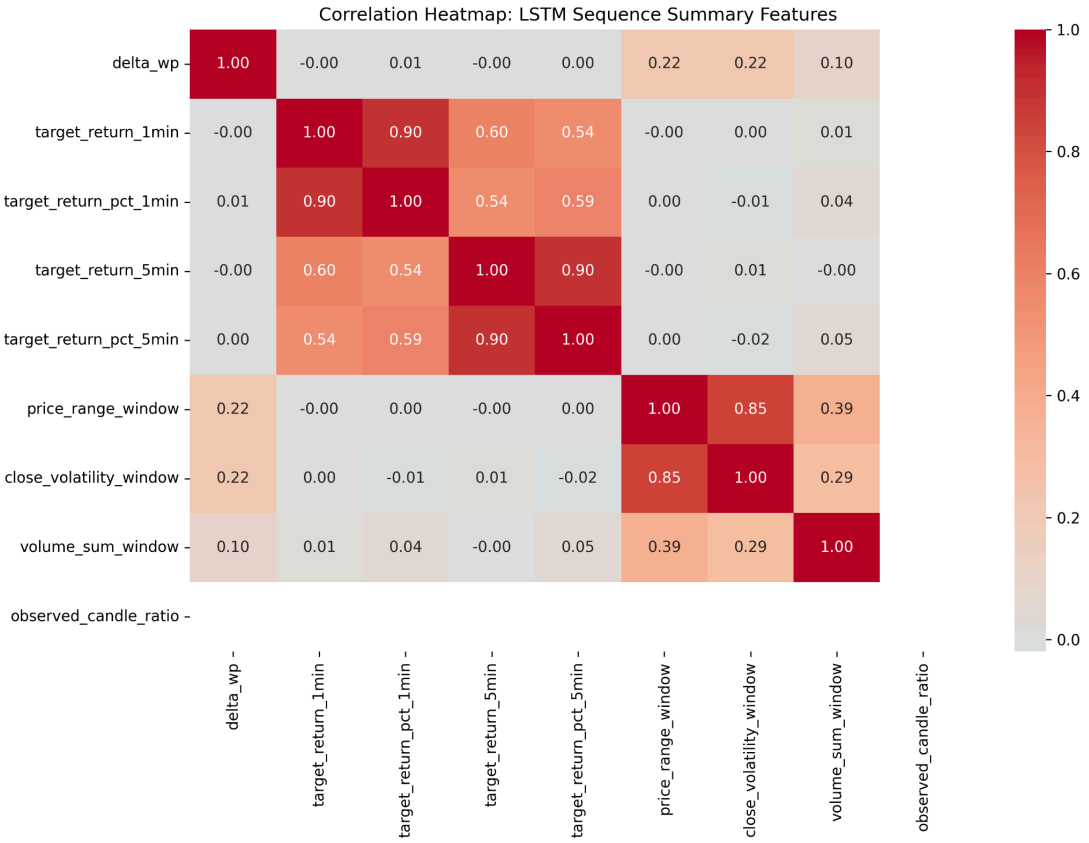
After completing exploratory data analysis we observed that our data quality was sufficiently strong enough for sequence learning: all 12,422 samples were flagged ready for LSTM use, we saw that the median observed-candle ratio was 1.000, and we observed the median target alignment error at both 1 and 5 minutes was 0. We observed an average shock magnitude of 0.0384 and for the median we observed a value of 0.0300. This reinforces that the task is mainly driven by moderate but meaningful in-game information updates. We also see that 1-minute direction labels were reasonably distributed (flat: 5,074; up: 3,734; and down: 3,614).

Figure 1. Delta win probability versus short-horizon Kalshi target returns.



We saw that this made direction classification a more stable benchmark task than the much rarer overreaction label. Overreactions were extremely sparse, with an overreaction_1min rate of only 0.17%. Larger shock quartiles showed larger mean absolute 5-minute target moves, increasing from 0.0313 in the lowest quartile to 0.0438 in the highest quartile, with higher volatility as well. Correlations between scalar summaries and 5-minute target returns were weak, motivating nonlinear baselines and full sequence models.

Figure 2. Correlation heatmap of sequence summary features and target returns.



4. Methods and Models

4.1 Final modeling pipeline

Our final modeling pipeline has five stages:

1. match Kalshi NBA events to ESPN game IDs using ticker-derived date and team tricodes;
2. extract high-impact ESPN plays by delta win-probability shocks;
3. fetch Kalshi 1-minute candles for each market around the event window;
4. build fixed-length 20-step sequences and attach 1-minute market candles;
5. train shared-split c and sequence models.

4.2 Training Data Details

All benchmark models use group-aware train/test splitting by espn_game_id to reduce leakage across plays from the same game. The holdout size is 20%, and the random state is fixed at 42. The baseline feature set contains nine event-level and pre-event variables: delta_wp, home_win_prob, event_reference_price, pre_price_return_10min, pre_price_return_pct_10min, price_range_window, close_volatility_window, volume_sum_window, and pre_observed_candle_ratio_10min.

4.3 Classification and Regression Tasks

We split up our predictions into two tasks: classification and regression. For the classification task we aim to predict the direction of price change 1 minute post win probability shock event. The regression task aims to predict the price 1 minute after a shock event. The classification models are evaluated based on their accuracy and balanced accuracy in predicting the 1 minute pricing direction. Our Regression models are evaluated using MAE, RMSE, and R^2 from their accuracy on the true predicted values.

4.4 Baseline Models

For 1-minute direction, the repository trains a standardized multinomial logistic regression baseline with class balancing and max_iter=1000, then a HistGradientBoostingClassifier with 200 boosting iterations, learning rate 0.05, and L2 regularization 0.01. For 1-minute price regression, it trains a standardized linear regression and a HistGradientBoostingRegressor with 300 iterations, learning rate 0.05, and the same regularization.

4.5 Sequence Models

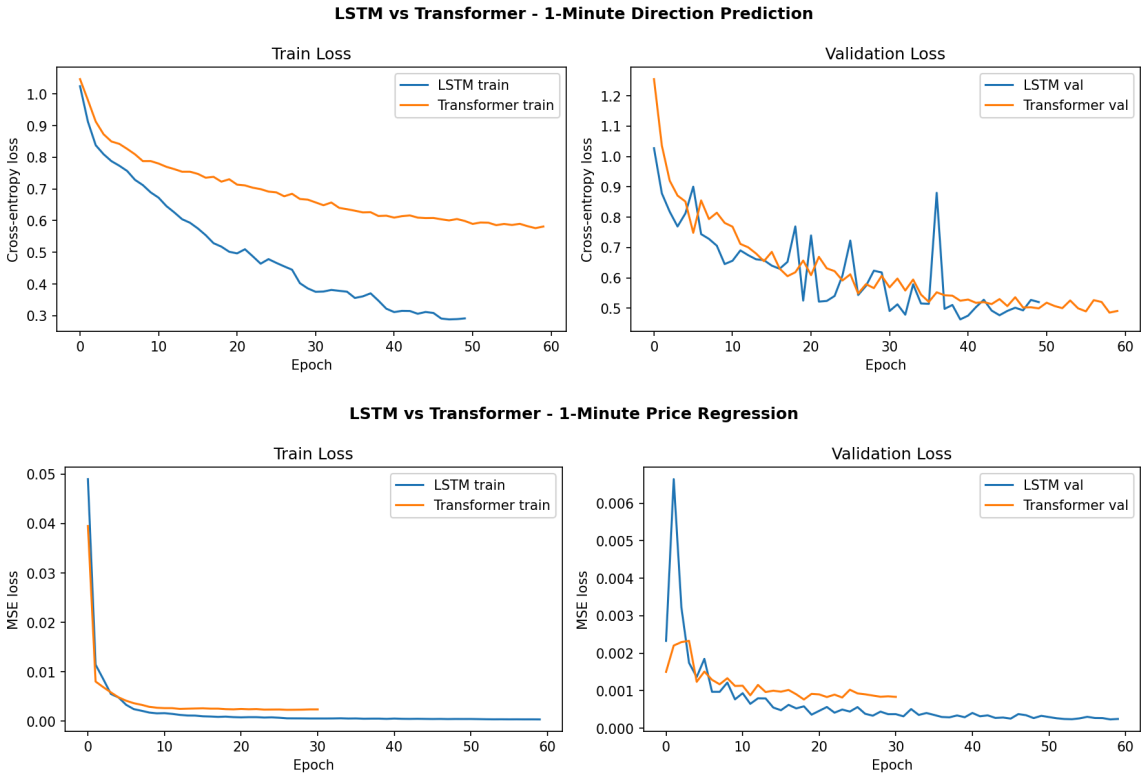
We define two sequence modeling architectures for both the direction classification and price regression tasks. The LSTM direction model uses a two-layer LSTM with hidden size 64, dropout 0.3, batch normalization on the final hidden state, and a small dense head. The Transformer direction model uses a 32-dimensional embedding, 4 attention heads, 2 encoder layers, feedforward width 64, and dropout 0.2. Their price-regression counterparts reuse the same backbone designs with scalar outputs. All models are trained with batch size 32 and evaluated with batch size 64 using the Adam optimizer. The LSTM models use a learning rate of 1×10^{-3} with no weight decay, while the Transformer models use a learning rate of 5×10^{-4} with weight decay 1×10^{-4}. Training is capped at 60 epochs with ReduceLROnPlateau scheduling and early stopping (patience 10 for LSTM, 12 for Transformer).

Table 1. Training configuration and runtime summary for sequence models.

Task	Model	Train BS	Eval BS	LR	WD	Max Ep	Ep Run	Patience	Time (s)
1-min direction	LSTM	32	64	1e-3	0	60	50	10	53.59
1-min direction	Transformer	32	64	5e-4	1e-4	60	60	12	80.71
1-min price	LSTM	32	64	1e-3	0	60	60	10	65.77
1-min price	Transformer	32	64	5e-4	1e-4	60	31	12	41.36

For the 1-minute direction task, the LSTM converges faster and achieves lower training loss (≈0.29 vs ≈0.58) and slightly better validation stability (≈0.50 vs ≈0.49–0.52), completing in 53.6s over 50 epochs compared to the Transformer's full 60-epoch run (80.7s). For the 1-minute price regression task, the LSTM again attains lower training and validation MSE (≈0.0003 vs ≈0.0008 validation) and trains longer (60 epochs) but still more efficiently overall (65.8s vs 41.4s due to earlier stopping of the Transformer at 31 epochs). Across both tasks, the LSTM demonstrates faster convergence, lower final loss, and more stable validation behavior, while the Transformer shows slower but steady improvement with stronger regularization effects.

Figure 3. Training and validation loss curves for LSTM and Transformer models.



5. Results

5.1 Direction Classification

Logistic regression reaches 0.446 accuracy and 0.427 balanced accuracy, which indicates that a purely linear boundary on engineered pre-event summaries does capture some structure but misses a large amount of it. HistGradientBoosting improves to 0.484 accuracy and 0.476 balanced accuracy on the same grouped split. That gap is modest in raw accuracy terms but meaningful substantively: it shows that the market response depends on interactions and threshold behavior rather than on one smooth linear rule.

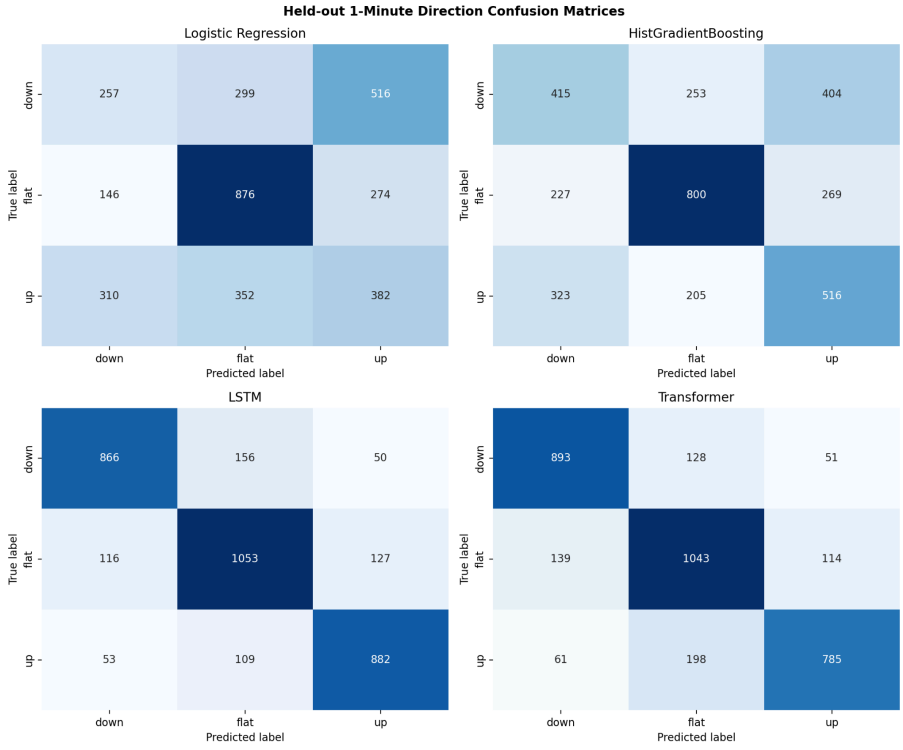
The sequence results are much larger, with the LSTM slightly outperforming the Transformer on both accuracy and balanced accuracy. Interpreted mechanically, this suggests that the full local OHLCV path contains a great deal of information about the immediate next move.

Table 2. One-minute direction classification performance.

Model	Accuracy	Balanced Accuracy
Logistic Regression	0.4440	0.4272
HistGradientBoosting Classifier	0.5073	0.4996
LSTM Classifier	0.8209	0.8217
Transformer Classifier	0.7975	0.7966

The confusion matrix reflects these accuracies as there is an overrepresentation of flat predictions for the logistic regression and HistGradientBoosting models. The sequence based models however show density in the accurate predictions for the direction of price move.

Figure 4: Confusion Matrix of predicted vs true label for each model



5.2 Price Regression

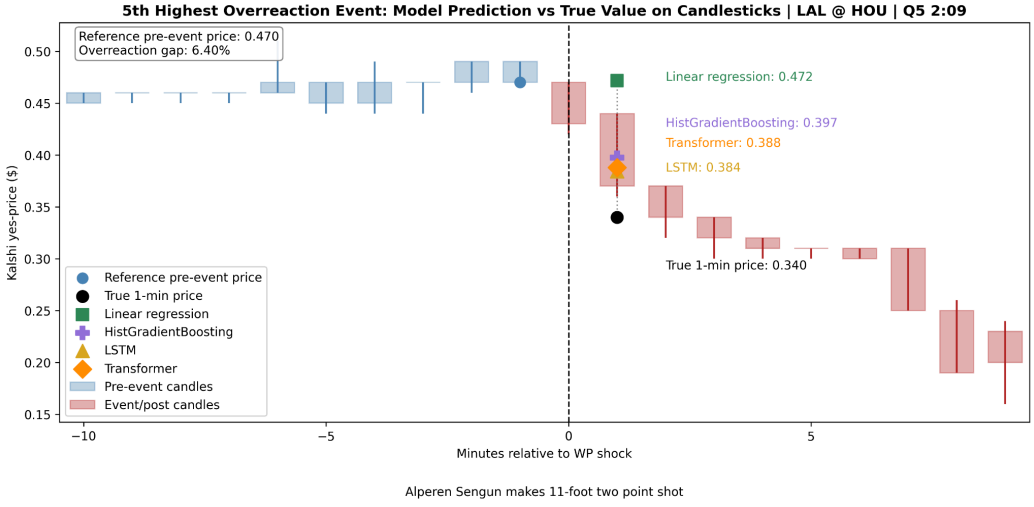
The price task tells a slightly different story from the direction task. Even the linear regression baseline is already very strong, with RMSE 0.0340 and R2 = 0.970. The boosted tree regressor does not improve on that baseline, implying that nonlinear scalar interactions are less useful for point prediction than they are for discrete direction classification. The sequence regressors again post very strong numbers, especially the LSTM at RMSE 0.0153.

Table 3. One-minute price regression performance.

Model	MAE	MSE	RMSE	R^2
Linear Regression	0.0227	0.0012	0.0340	0.9703
Hist Gradient Boosting	0.0278	0.0014	0.0379	0.9630
LSTM	0.0111	0.0002	0.0153	0.9939
Transformer	0.0214	0.0008	0.0277	0.9802

We evaluate the regression models on a specific example overreaction event to see how their predicted prices compare to true prices 1-minute post event. Figure 5 illustrates this high-impact win-probability shock and the resulting sharp decline in Kalshi market price from roughly 0.47 to 0.34 immediately after the event. Again, the predictions reflect the differences in RMSE as the baseline models such as linear regression and HistGradientBoosting underreact to this change, staying closer to the pre-event price and failing to capture the magnitude of the drop. In contrast, the sequence models, particularly the LSTM, produce predictions much closer to the true post-event price, indicating a better understanding of the temporal dynamics.

Figure 5. Example high-overreaction event with model predictions.



6. Conclusions

In conclusion, we have shown that machine learning models can be used to model NBA prediction markets. We've shown this by connecting win probability shocks to Kalshi market data by exporting win probability data using a 10-minute lookback and 10-minute look forward window. We gathered our data on win probability from ESPN and our data on market price from Kalshi. Our dataset includes 12,422 win probability shock event-market sequences. We classified events to be a win probability shock where the probability of teams winning jumped by at least 2%. Our exploratory data analysis found that the price 1 minute after a probability shock was reasonably distributed across moving 'up', 'down', and 'flat'. We went on to fit several models for both classification and regression. We fit logistic regression, HistGradient Boosting, LSTM, and transformer for time series models. The LSTM and transformer models heavily outperformed the other models, suggesting a nonlinear temporal relationship between win probability shocks and Kalshi market price jumps. Furthermore, the same models that perform better at classification, were also better at price prediction.

7. Broader Impact

With such models, the door is opened to further market research and identifying pricing inconsistencies. Practical implementation is likely limited to identifying market behavior upon arrival of new information that has significant impact on game outcomes. However, with the emotional gambling nature of such events, using these models to improve market quality is implausible. It is more likely implemented in speculative or gambling strategies, though ill-advised. However, trying to excessively execute such strategies risks acting on noise as opposed to signal. Going forward, the models discussed in this paper have significance that could extend to stochastic markets beyond just Kalshi. Perhaps, headline news could act to financial markets similarly to how a major steal impacts Kalshi markets for an NBA game. In such cases, the models discussed above could lead to research on more complex trading. Anyhow, future use of such models should be limited under consideration of responsibility of usage and broader fairness.

References

[1] ESPN. *NBA Scoreboard and Summary Endpoints*. Accessed May 2026 via ESPN public NBA API endpoints.

[2] GitHub Copilot CLI. Generative-AI assistance was used for debugging, restructuring, and modeling support while preparing the notebook. Accessed May 2026.

[3] Kalshi. *Kalshi Trade API Documentation*. Accessed May 2026. https://docs.kalshi.com/

[4] Kalshi Python Sync client. Package used for authenticated Kalshi access in the project environment. Repository/package accessed May 2026.

[5] Pedregosa, F. et al. *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12:2825--2830, 2011. Project documentation accessed May 2026 at https://scikit-learn.org/