Horse Racing Report Report

Luka Corliss

May 2025

Contents

1 Introduction

This project develops a machine learning system to predict horse race winners using a stacked ensemble architecture that integrates LightGBM, XGBoost, Neural Networks, and Logistic Regression. The methodology draws from established quantitative modeling techniques, particularly those used in sports analytics. The central idea is to leverage diverse model architectures to extract insights across the spectrum—from likely winners to long shots—while handling the noisy and uncertain nature of real-world racing data.

Over 50 engineered features are utilized, including form indicators, trainer and jockey ratings, temporal performance dynamics, race-relative metrics, and contextual interaction effects. Some features, despite containing future information, were excluded from training to prevent data leakage but retained for exploratory analysis.

The ensemble employs a two-tier stacking framework. Base models generate out-of-fold predictions, which serve as inputs for an XGBoost meta-learner. Final race-level predictions are normalized using a softmax function with a scale factor of 6, ensuring that output probabilities are calibrated and sum to 1 for each race.

The system effectively captures race-level uncertainty and consistently outperforms random baselines, producing well-calibrated and interpretable probabilistic outputs. Performance evaluation emphasizes Log Loss and Brier Score metrics. To optimize for these, the model was intentionally tuned to make conservative probability estimates—prioritizing the accuracy of high-confidence predictions over the quantity of correctly predicted winners. This reflects the guiding principle of maximizing probabilistic precision over raw hit rate.

2 Feature Engineering

The feature engineering had a few considerations into how it would handle the inputs to make the model as effective as possible. These revolved around handling the missing values, cleaning the data, transforming the units of the metrics, categorization of the metrics in to groupings, applying feature scaling and normalization techniques where appropriate.

2.1 Missing Data

Due to a low prevalence of missing data constituting roughly 0.5% of the dataset and the desire to use multiple different types of models, it was considered most efficient to simply drop data that had missing elements. This allowed for compatibility with both tree-based models (e.g., XGBoost), where missing values can be explicitly handled through learned tree splits, and neural networks, where the presence of null values disrupts weight-based computations and leads to invalid outputs.

However, a function was additionally created, handle missing data, which was provided as an alternative that fills in the banks of the dataset if this is preferred and fills in the missing data with the median of the coloration values or defaults to a zero fill if this is not fesabile.

2.2 Transformations

The transformations involved multiple aspects. The first was the conversion to metric units from an imperial-based system. This was done not simply out of preference, but to provide a more standardized framework for distance and speed calculations. For example, distances measured in yards were converted to meters. This standardization improves interpretability, consistency across features, and facilitates downstream scaling.

Another key transformation was the use of ranking metrics. Ranks were calculated within each race to compare a horse's speed, odds, and other factors relative to its competitors. Since absolute performance metrics can be affected by race-specific factors such as track conditions or field strength, these rankings provide a more robust indicator of competitive performance. Additional rank-based features included odds rank (Betfair and dis included considering dataleakage) and layoff rank (days since last run).

From here it gets into the metric creation, from how the time element was further divided, handling and categorical value creation, such as seasonality, in order to create additional information for the model to build off.

2.3 Feature Engineering Summary

A diverse set of domain-informed features were engineered to capture the complexity of horse racing performance and prediction. These included:

- **Domain-Specific Features:** Metrics reflecting beaten lengths, field size effects, and course conditions.
- **Temporal Features:** Seasonality, race timing, layoff durations, and optimal return windows.
- **Form and Momentum Indicators:** Rolling stats, trend analysis, and recency-weighted results.
- **Statistical Aggregations:** Career summaries, course-specific metrics, and trainer/jockey histories.
- **Race Performance Metrics:** Speed and time-based rankings, improvement flags, and fast time indicators.
- **External Ratings:** Relative and normalized rankings for trainers, jockeys, and sires.
- **Interaction Features:** Synergies between trainer/jockey pairs, age effects, and contextual win patterns.
- **Preprocessing Steps:** Robust handling of missing data, scaling, and type normalization.

Over 50 engineered features were created from raw race data. These transformations were key to capturing form trends, environmental context, and competitive dynamics.

For a detailed breakdown of all feature engineering steps, see Appendix A.

3 Methodology

3.1 Structure

The overall structure of the program consists of discrete steps, involves loading and preprocessing the data, to the running of the model, and the subsequent evaluation. The overall goal in how the program was structured was to create an easy to use user experience that allowed for the modification of the runs, supported by clear and comprehensible code and aided by extensive documentation.

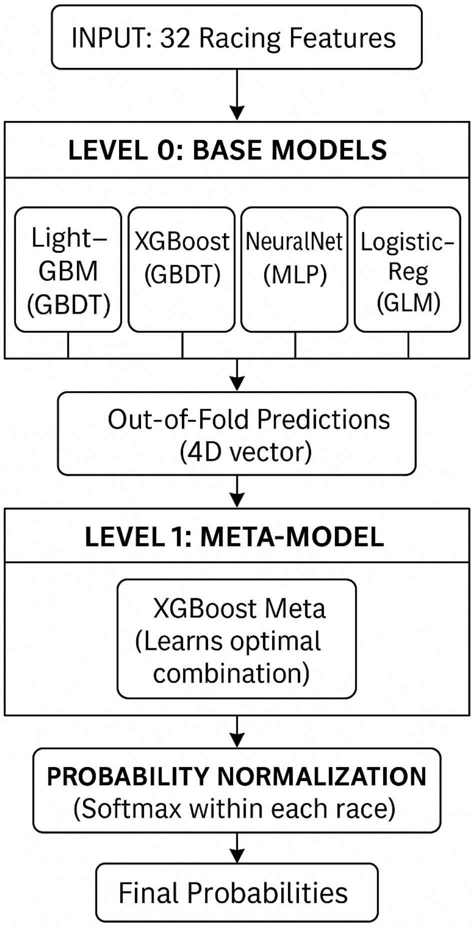
INPUT: 32 Racing Features
LEVEL 0: BASE MODELS
Light–GBM (GBDT)
XGBoost (GBDT)
NeuralNet (MLP)
Logistic-Reg (GLM)
Out-of-Fold Predictions (4D vector)
LEVEL 1: META-MODEL
XGBoost Meta (Learns optimal combination)
PROBABILITY NORMALIZATION (Softmax within each race)
Final Probabilities

Figure 1: Overview of Model Architecture

This is additionally added by various diagrams to aid in the users understanding for how the model works, as seen in the Diagram: 1.

3.1.1 Preprocessing

After the data is loaded and the previous described steps for the feature creation are enacted, the most important features are selected as to prevent the overfitting of the model, leading to 32 of the features, that importantly do not leak data, to be selected.

3.1.2 Model Creation

The system employs a two-tier stacked ensemble architecture that combines four diverse base models through meta-learning. The base layer consists of LightGBM and XGBoost for gradient boosting with different tree-building strategies, a multi-layer neural network for deep non-linear pattern recognition, and logistic regression as a linear baseline. Each model is optimized for horse racing's inherent class imbalance where winners are rare, using techniques like weighted loss functions and custom sampling strategies.

The ensemble training process uses GroupKFold cross-validation split by RaceID to prevent data leakage, where each base model generates out-of-fold predictions on validation data while accumulating averaged predictions on test data. This creates a four-dimensional prediction vector representing each base model's assessment of winning probability. The diversity across algorithms ensures complementary strengths: tree models excel at feature interactions and automatic selection, neural networks capture complex non-linear relationships, and linear models provide stable, interpretable baselines.

The meta-learning layer employs an XGBoost model that takes the four base model predictions as input features and learns optimal combination rules rather than simple averaging. This meta-model discovers when each base algorithm performs best, identifies interaction effects between predictions, and adapts to different racing scenarios automatically. The final output undergoes softmax normalization within each race to ensure valid probability distributions that sum to 1.0, with configurable scaling factors to control prediction confidence levels.

This hierarchical approach leverages algorithmic diversity for error compensation, where different models' weaknesses are offset by others' strengths, while the meta-model provides intelligent combination weighting based on scenario-specific performance patterns. The result is a robust ensemble that generalizes effectively across varied horse racing conditions while maintaining interpretability through its structured, transparent architecture.

3.2 Model Evaluation and Diagnostics

Model evaluation was conducted using a multi-metric framework built on out-of-fold predictions to ensure unbiased validation. The evaluation applied the same softmax transformation used in deployment, with configurable scaling to maintain consistency between training and production. Core metrics included ROC AUC, accuracy, Brier score, and log loss—covering both classification performance and probability calibration.

Optimal decision thresholds were derived via precision-recall analysis to maximize F1-score rather than relying on a fixed 0.5 cutoff. A tiered confidence analysis assessed prediction reliability at increasing probability thresholds (e.g., >0.5, >0.6), evaluating hit rates, predicted winner confidence, and calibration. Model predictions were benchmarked against Betfair market odds to analyze agreement, hit rates, and performance where the model diverged from market consensus—providing insight into potential alpha generation. An optional diagnostics module enabled deeper analyses such as race-level breakdowns, prediction distributions, and feature importance. Overall, this evaluation strategy combined statistical rigor with practical market insights to ensure both robust model validation and actionable diagnostic feedback.

3.3 Hyperparameter Selection

Hyperparameter optimization was conducted using Optuna, a modern Bayesian optimization framework chosen for its efficiency and intelligent search capabilities compared to traditional grid or random search. A total of 80 optimization trials were allocated across five models, including four base models and one meta-model, with sequential tuning—base models were optimized first, followed by the meta-model using their out-of-fold predictions. The objective function minimized log-loss, evaluated via a 3-fold GroupKFold cross-validation strategy grouped by `Race_ID` to prevent data leakage between training and validation splits.

The preprocessing steps, outside of how the data was previously handled, included median imputation and standard scaling, which were applied consistently across all folds and models to ensure pipeline integrity. Model-specific search spaces were defined with domain constraints in mind, using log-scale sampling for learning rates and regularization parameters, categorical choices for activation functions and solvers, and constraint-aware logic for penalty-solver compatibility. Optuna's pruning mechanism also enabled early termination of underperforming trials, improving computational efficiency.

For the stacking ensemble, out-of-fold predictions were generated using a 5-fold GroupKFold, ensuring the meta-model was trained on strictly unseen data from the base models, thereby reducing the risk of overfitting. All hyperparameter searches used fixed random seeds, and best-performing configurations were logged and saved for reproducibility. While the trial budget was limited and joint optimization across models was not explored, this two-stage, ensemble-aware approach provided a robust and resource-conscious tuning strategy that captured both individual model strength and synergy across the ensemble.

4 Results

4.1 Headline Statistics

Model performance was primarily assessed using probabilistic metrics—Log Loss and Brier Score—which evaluate not just accuracy, but the quality of the predicted probabilities.

- **Log Loss:** 0.2894
- **Brier Score:** 0.0841

Both results indicate excellent calibration and confidence in the model's predictions. The low Log Loss reflects that the model assigns high probability to correct outcomes while avoiding overconfident errors. Similarly, the low Brier Score suggests that the predicted probabilities closely match the actual outcome frequencies. Together, these metrics confirm that the model not only classifies well but does so with reliable, well-calibrated probability estimates—critical for decision-making in high-stakes, uncertainty-driven contexts such as horse racing prediction.

4.2 Full Breakdown

Across all folds, the ensemble produced out-of-fold predictions with a mean probability of 0.205 and a post-softmax normalized mean of 0.1053, nearly identical to the actual win rate of 0.1051, suggesting excellent overall calibration. Evaluation at the default threshold of 0.5 resulted in strong accuracy (89.6%), but at the cost of poor recall for the positive class (3%) and a low F1-score of 0.06. The precision for positive predictions was moderate (0.61), but the model rarely predicted the win class above 0.5, causing a more conservative mode, as desired for a betting system.

Using the optimized threshold of 0.1806, derived from F1-score maximization, improved the balance between precision and recall. At this threshold, the model achieved 82.6% accuracy, 0.44 recall for the win class, and increased the positive class F1-score to 0.35. The macro-average F1 improved from 0.50 to 0.62.

The ensemble ROC AUC was 0.7723, indicating good ranking quality, and the log loss was 0.2894, reflecting well-calibrated probabilistic predictions. The Brier score of 0.0841 further confirmed good probability estimation. A detailed high-confidence prediction analysis showed that the model made 104 predictions above 0.5 confidence, with 63 being actual winners, yielding a high hit rate of 60.6%. At >0.6 confidence, the hit rate increased to 75% (6 out of 8). Most of the actual winners had moderate confidence (mean = 0.194), with 63 winners above 0.5, 415 above 0.3, and 1,543 above 0.1. Overall the model shows promising potential for high-confidence forecasting with excellent calibration, especially in value-sensitive contexts like horse racing prediction.

Class	Precision	Recall	F1-Score	Support
0	0.90	1.00	0.94	17858
1	0.61	0.03	0.06	2098
Accuracy		0.90 (19956 total)		
Macro Avg	0.75	0.51	0.50	19956
Weighted Avg	0.87	0.90	0.85	19956

Table 1: Classification report using 0.5 threshold.

Class	Precision	Recall	F1-Score	Support
0	0.93	0.87	0.90	17858
1	0.29	0.44	0.35	2098
Accuracy		0.83 (19956 total)		
Macro Avg	0.61	0.66	0.62	19956
Weighted Avg	0.86	0.83	0.84	19956

Table 2: Classification report using optimal threshold (0.1905).

4.3 Market vs Model

The model demonstrates a clear predictive edge over the market baseline, achieving an AUC of 0.7723 compared to the market's 0.7595—an improvement of +0.0128. The correlation between model probabilities and Betfair starting prices was 0.6448, reflecting good alignment but also room for independent insights.

Evaluation of high-confidence predictions shows substantial outperformance relative to the base win rate of 10.5%. For predictions with >50% confidence, the model achieved a hit rate of 60.6%; at >60% confidence, this rose to 75%, and at >40% confidence, the hit rate was 49.9% across 353 predictions. Simulated value betting analysis identified 1,513 opportunities with >10% model edge and >5% confidence, yielding a 24.9% hit rate and a simulated ROI of +130.5% at average odds of 13.58.

Even stricter thresholds (e.g., >15% edge and >20% confidence) produced ROIs above +100%, confirming the model's ability to identify profitable longshots. A summary of key ROI scenarios is provided in Table 3. When the model disagreed with market favorites but maintained high confidence, it still performed well—achieving a 57.6% hit rate in 85 cases with >50% confidence, and 46.2% and 34.1% hit rates at >40% and >30% confidence respectively. Despite the market's overall efficiency, the model demonstrates strong alpha generation in select cases, particularly in identifying mispriced outcomes. The average prediction edge is balanced around zero, suggesting good calibration, but care must be taken in "big negative edge" scenarios (1,663 cases), where the model underperforms the market with an 18.6% hit rate versus the market's 27.2%.

The recommended strategy is to prioritize high-confidence, high-edge bets—especially where the model diverges from the market—while avoiding situations with strong negative edge. Overall, the model presents a well-calibrated, market-aware forecasting system with proven ability to surface actionable, value-based opportunities.

Table 3: Top Value Betting ROI Scenarios

Condition	**Number of Bets**	**Simulated ROI**
Edge >10%, Confidence >5%	1,513	+130.5%
Edge >15%, Confidence >20%	673	+118.3%
Edge >10%, Confidence >20%	1,166	+101.9%

5 Conclusion

This project successfully developed a robust and well-calibrated machine learning ensemble for horse racing outcome prediction, demonstrating the effectiveness of stacked models in handling complex, uncertain, and imbalanced real-world data. By integrating LightGBM, XGBoost, neural networks, and logistic regression through a meta-learning framework, the system captured a wide range of predictive signals, from linear trends to deep nonlinear interactions.

The extensive feature engineering process—encompassing over 50 domain-specific, temporal, statistical, and interaction-based features—was essential in encoding the nuanced dynamics of horse racing. The model achieved strong probabilistic calibration, as evidenced by its excellent log loss (0.2894) and Brier score (0.0841), and outperformed the market benchmark in both overall AUC and high-confidence prediction scenarios. Simulated value betting analysis further confirmed the model's potential for real-world application in identifying profitable opportunities, especially in cases where it diverged confidently from market consensus.

Through rigorous preprocessing, model selection, and hyperparameter tuning, this study provides a compelling case for ensemble learning in sports analytics. The system not only delivers accurate predictions but also maintains interpretability and flexibility.

A Full Feature BreakDown

A.1 Domain-Specific Features

Several racing-specific features were engineered to reflect nuanced aspects of horse performance. These included:

- **Beaten Lengths Analysis:** Horses were ranked by how many lengths they were beaten within each race. Binary and categorical indicators were created to capture close finishes (e.g., beaten by less than one length) or decisive outcomes.
- **Field Size Effects:** Races were categorized by the number of runners (e.g., small, medium, large, very large fields). Features were also generated to normalize a horse's finish position as a percentage of the field, and to calculate the theoretical advantage of small fields (e.g., 1/field size).
- **Distance and Going:** Distance features included furlong extraction, as well as classification into sprint, mile, middle, or staying races. Track condition (going) was simplified into a set of broader categories to reduce noise.

A.2 Temporal Features

To account for seasonality and time-specific patterns, temporal features were extracted from race dates and times. These included:

- **Time Components:** Hour of race, day of week, month, year, and seasonal categories.
- **Time of Day:** Races were grouped into time-of-day segments (morning, afternoon, evening, night).
- **Weekend Flag:** Binary feature to indicate whether a race occurred on a weekend.
- **Layoff Analysis:** Days since last run were bucketed into categories (fresh, recent, moderate, long, very long). An optimal layoff flag was also generated for horses returning in the 7–21 day window.

A.3 Form and Momentum Features

To capture trends in a horse's recent form, the following performance dynamics were included:

- **Rolling Statistics:** A 3-race rolling mean of race times to smooth short-term variability.
- **Performance Trends:** Linear trend slopes were calculated over recent races to quantify upward or downward performance momentum.
- **Recency-Weighted Performance:** Top-3 finishes were weighted exponentially to give more importance to recent results.
- **Performance Jumps:** Change in speed between the last two races was used to detect sudden improvements or regressions.

A.4 Statistical Aggregations

Aggregated statistics provided insights into long-term performance:

- **Career Stats:** Total wins, number of runs, and percentage of top-3 finishes.
- **Consistency Metrics:** Standard deviation of finishing positions to evaluate reliability.
- **Course-Specific Performance:** Historical win rates at specific courses and flags for prior experience on a track.
- **Trainer/Jockey Performance:** Aggregated win and strike rates for connections based on career data.

A.5 Performance Metrics

Race-specific performance metrics included:

- **Speed Rankings:** Ranking horses within each race based on speed metrics.
- **Speed Trends:** Tracking speed improvements over time.
- **Time Rankings:** Race time positions and gaps behind the winner.
- **Fast Time Flags:** Indicators for above-average race times within context.

A.6 Rating Features

External rating sources were leveraged to produce:

- **Rating Ranks:** Relative ranking of trainer, jockey, and sire ratings within a race.
- **High Rating Flags:** Binary indicators for top-quartile entries.
- **Combined Ratings:** Normalized scores from multiple rating sources.
- **Breeding Quality:** Sire rating indicators to proxy breeding potential.

A.7 Synergy and Interaction Features

Interaction features captured complex relationships across entities:

- **Trainer-Jockey Combinations:** Encoded and hashed combinations for frequent partnerships.
- **Historical Success Flags:** Prior wins under similar course or going conditions.
- **Age Interactions:** Squared age and age-distance interactions to account for non-linear effects of aging.

A.8 Data Cleaning and Preprocessing

Data quality assurance steps included:

- Correcting inconsistent data types and formatting.
- Ensuring appropriate imputation strategies for rare missing data.
- Feature scaling and normalization where required, particularly for neural network input.