Advanced Machine Learning Strategy for Predicting Profitable Outcomes in Cryptocurrency Gambling

I. Strategic Overview and Executive Recommendations

1.1 Project Mandate

The primary mandate of this initiative is to develop and deploy a state-of-the-art machine learning system capable of accurately predicting profitable games on the Rugs.fun platform. A "profitable" game is defined as one achieving a multiplier greater than 4.8x. The system must not only achieve high predictive accuracy but also provide deep, interpretable insights into the underlying behavioral dynamics, particularly the identified "SOL whale effect," which has demonstrated a significant preliminary correlation with profitable outcomes. The ultimate goal is to create a production-ready system that enhances strategic decision-making, optimizes platform engagement, and provides a sustainable competitive advantage through superior market intelligence.

1.2 Core Challenges

The successful execution of this mandate requires navigating a series of complex technical challenges inherent to the dataset and the problem domain. These challenges form the foundation upon which our strategic recommendations are built:

- **Temporal Data Dependency:** The game data is a time series, where each game is a sequential event. This temporal nature means that observations are not independent, a critical consideration that invalidates standard cross-validation techniques and necessitates time-aware modeling to prevent data leakage from the future into the past.[1]

- **Class Imbalance:** Profitable games represent approximately 17% of the dataset. This significant class imbalance can bias standard classification algorithms towards the majority (non-profitable) class, leading to poor performance in identifying the very outcomes of interest.[2]
- **Mixed Data Types:** The 29-feature dataset contains a combination of numerical (e.g., finalTick, volatility) and categorical (e.g., is_mixed_mode, is_sol_dominated) features. The chosen model architecture must effectively handle this mixed-type data without extensive manual preprocessing.[4]
- **Dual Objective: Accuracy and Interpretability:** The project requires a dual optimization. It is not sufficient to merely predict profitable games; the system must also explain *why* a prediction was made. This necessitates a model architecture and a corresponding explainability framework that can deconstruct predictions into understandable, feature-based drivers to illuminate the mechanics of whale behavior.[6]

1.3 Executive Summary of Recommendations

To address these challenges and achieve the project's objectives, a multi-faceted strategy is recommended, integrating best-in-class techniques from feature engineering to production deployment.

- **Primary Model Architecture:** The core predictive engine will be a **CatBoost Gradient Boosted Decision Tree (GBDT)**. This choice is predicated on its state-of-the-art performance on structured tabular data, its superior native handling of categorical features which simplifies the pipeline, and its exceptional synergy with modern explainability frameworks.[8]
- **Feature Engineering and Selection Strategy:** An advanced, hybrid feature strategy will be implemented. This begins with leveraging CatBoost's embedded feature importance to prune the initial 29 features. Subsequently, new, high-impact features will be engineered, including non-linear polynomial and interaction terms to capture complex dynamics, and domain-specific features that model temporal concepts like "Whale Momentum" and "Market Fear." Finally, UMAP will be used not just for dimensionality reduction but as a sophisticated feature generation tool to create latent behavioral archetypes.[10]

- **Validation and Evaluation Protocol:** A rigorous **Walk-Forward Cross-Validation** methodology will be the cornerstone of our validation framework. This time-aware approach, which uses an expanding window of past data to predict future data, is essential for obtaining a realistic estimate of model performance and preventing data leakage.[1] The primary evaluation metric will be the
  F1-Score for the positive (profitable) class, supplemented by the **Precision-Recall Area Under Curve (PR-AUC)**, as both are robust to class imbalance.
- **Hyperparameter Optimization:** We will employ **Bayesian Optimization via the Optuna framework** for hyperparameter tuning. This will be configured as a multi-objective optimization problem, simultaneously seeking to maximize the F1-Score while minimizing model inference latency, ensuring the final model is both accurate and production-ready.[17]
- **Model Explainability Framework: SHAP (SHapley Additive exPlanations)** will be the primary tool for interpreting the CatBoost model. SHAP will provide both global feature importance rankings and local, per-game prediction explanations, allowing for a granular analysis of the factors, especially whale behavior, that drive profitability.[7] This will be augmented with an anomaly detection system built on SHAP values to identify novel whale strategies.
- **Production Deployment Strategy:** A containerized (Docker) microservice architecture using a high-performance framework like FastAPI is proposed for serving the model. This will be deployed on a scalable platform like Kubernetes and will be supported by a comprehensive MLOps pipeline for monitoring concept drift and automating model retraining to ensure sustained performance over time.[20]

1.4 Expected Business Impact

The successful implementation of this strategic plan is projected to yield significant business value. The primary impact will be the ability to more accurately identify potentially profitable games in real-time, enabling the development of automated trading or betting strategies with a higher expected return on investment. Beyond direct profitability, the explainability framework will generate profound business intelligence. By deconstructing the "SOL whale effect" and other predictive patterns, the platform can gain a deeper understanding of its most valuable user segments, inform new feature development, enhance user experience, and implement more sophisticated risk management protocols. This transforms the machine learning system from a simple predictive tool into a strategic asset for continuous learning and platform optimization.

II. Advanced Feature Engineering and Selection: Maximizing Signal from Noise

The existing 29-feature set provides a strong foundation, particularly with the discovery of the "SOL whale effect." However, to achieve state-of-the-art performance, it is imperative to move beyond this baseline. This section details a comprehensive strategy to refine the current feature set, generate new high-impact variables that capture non-linear and temporal dynamics, and select the optimal subset for model training.

2.1 A Hybrid Strategy for Optimal Feature Subset Selection

With a moderately-sized feature space of 29, selecting the most predictive and least redundant subset is crucial for model performance, interpretability, and computational efficiency. A multi-stage, hybrid feature selection strategy is recommended, combining the strengths of different methodologies to achieve a result superior to any single approach.[10]

- **Methodology Overview:** The selection process will unfold in two primary stages:
  1. **Stage 1: Embedded Method for Initial Pruning:** The first stage will utilize an **Embedded** feature selection method. These methods perform feature selection as an integral part of the model training process.[10] We will train an initial CatBoost or XGBoost model on the full feature set. The resulting feature importance scores, which are derived from the features' contributions to reducing the loss function during tree construction, will be used to perform an initial, aggressive pruning of the least important features. This approach is computationally efficient and inherently considers feature interactions, as the importance of a feature is evaluated in the context of the entire model.[10]
  2. **Stage 2: Wrapper Method for Final Refinement:** The reduced feature set

from Stage 1 will then be subjected to a more rigorous **Wrapper** method. Wrapper methods evaluate subsets of features by training and testing a specific model, using its performance as the selection criterion.[26] We recommend
Recursive Feature Elimination (RFE). RFE works by recursively training the model, ranking features by importance, and eliminating the least important one until the desired number of features is reached.[27] This ensures the final feature subset is optimized directly for the predictive performance of our chosen model architecture.

- **Justification for the Hybrid Approach:** This hybrid strategy is superior to relying on a single method. **Filter methods**, such as those based on correlation or Chi-square tests, are computationally cheap but evaluate features in isolation, potentially discarding variables that are only powerful when combined with others.[10] Given the complex, interactive nature of gambling dynamics, this is a significant risk. Wrapper methods are powerful but can be computationally prohibitive on a large feature set.[29] Our hybrid approach uses the efficient Embedded method to narrow the search space, then applies the more powerful Wrapper method for final, performance-driven selection, yielding an optimal balance of rigor and efficiency.

Feature Selection Method	Key Techniques	Computational Cost	Considers Interactions?	Model-Agnostic/Specific	Recommended Use Case in this Project
Filter	Information Gain, Chi-Square, Fisher's Score [10]	Low	No	Agnostic	Not recommended as primary method; may miss key interactions.
Wrapper	RFE, Forward/Backward Selection [26]	High	Yes	Specific	**Stage 2:** Final refinement on a pruned feature set for maximum performance.

Embedded	L1 (Lasso), Tree-based Importance [10]	Medium	Yes	Specific	**Stage 1:** Efficient initial pruning of the full feature set.

Table 1: A comparison of feature selection methodologies, justifying the recommended hybrid approach that combines the strengths of Embedded and Wrapper methods for this project.

2.2 Capturing Non-Linear Dynamics: Polynomial and Interaction Features

Financial and behavioral systems are rarely linear. The relationship between a feature like sol_total_invested and game profitability is unlikely to be a straight line. Polynomial and interaction features are essential for allowing the model to capture these more complex, non-linear relationships.[11]

- **Polynomial Features:** To capture non-linear effects such as diminishing returns, we will generate second-degree polynomial features for key continuous variables. For instance, the impact of the fifth SOL invested by a whale might be marginally less than the first; a quadratic term can model this curvature.[32]
    - **Target Features:** finalTick, volatility, sol_total_invested, sol_whale_concentration.
- **Interaction Terms:** This is arguably the most critical area for feature engineering. The predictive power of a feature like has_real_sol_whale is likely magnified or dampened by the context of other game variables. We will explicitly create interaction terms to test specific hypotheses about whale behavior.[33]
    - **Hypothesis 1: Whale behavior is context-dependent.**
        - sol_whales * volatility: Do whales in highly volatile games have a different impact than those in stable games?
        - sol_early_whales * finalTick: Does the presence of early whales correlate with shorter, more profitable games?
    - **Hypothesis 2: Market sentiment amplifies whale impact.**
        - sol_whale_concentration * buy_sell_ratio: Does high whale concentration have a greater effect when the overall market is bullish (high buy-to-sell ratio)?
        - sol_to_free_ratio * trades_per_player: Does a high ratio of serious (SOL) to

unserious (FREE) players, combined with high activity, signal a more predictable environment?

- **Implementation:** These new features will be generated using the sklearn.preprocessing.PolynomialFeatures utility before the feature selection process, allowing the hybrid selection strategy to determine their ultimate value to the model.

2.3 Domain-Specific Feature Generation

To gain a true predictive edge, we must create features that encapsulate domain-specific knowledge from both quantitative finance and gambling analytics. The current features describe the state of a single game; the following proposals introduce features that capture temporal dynamics and player psychology across games.[34]

- **Whale Momentum and Herding Indicators:** These features aim to model the temporal behavior of whales.
  - sol_whale_count_MA_3: A 3-game moving average of the number of SOL whales. A rising value indicates potential herding behavior.
  - sol_whale_concentration_change: The percentage change in whale investment concentration from the previous game. A sharp increase could signal a coordinated move.
- **Market "Fear & Greed" Index:** A composite score to quantify market sentiment within a game.
  - **Formula:** A weighted average of normalized buy_sell_ratio, volatility, and trades_per_player. A high value indicates a "greedy," frenzied, and potentially over-extended market, while a low value indicates "fear" or caution.
- **Player Sophistication Proxy:**
  - sol_to_free_ratio_MA_10: A 10-game moving average of the ratio of SOL to FREE players. A sustained increase in this ratio might indicate a shift from a casual player base to a more professional or "sharp" one, potentially altering game dynamics.
- **"Chasing Losses" Behavioral Flag:** Drawing from responsible gambling analytics, this feature aims to identify potentially irrational behavior from known whale actors.[36]
  - **Implementation:** For identified whale wallets, track their profit and loss (PnL) over the last N games. A new binary feature, whale_is_chasing, would be

flagged as '1' if a whale significantly increases their investment size immediately following a substantial loss. This could be a powerful predictor of high-risk, high-volatility games.
- **Volatility of Volatility (VIX-like feature):** Inspired by the CBOE Volatility Index (VIX) in traditional finance, this feature measures market stability.[37]
  - **Implementation:** Calculate the standard deviation of the volatility feature over the last 10-20 games. A high value indicates an unstable and unpredictable market environment, while a low value suggests stability.

2.4 Dimensionality Reduction for Latent Feature Creation

The existing features describing player dynamics (Categories B, C, D) form a high-dimensional space. While each feature is informative, their combined interaction describes latent "behavioral archetypes" that are difficult for a tree-based model to capture directly. By using non-linear dimensionality reduction, we can project this space into a lower-dimensional representation, effectively creating powerful, information-dense summary features.

- **Technique Comparison and Recommendation:**
  - **Principal Component Analysis (PCA):** Unsuitable for this task. PCA is a linear technique that assumes independent observations and struggles with non-linear relationships and temporal data.[38]
  - **t-Distributed Stochastic Neighbor Embedding (t-SNE):** Primarily a visualization tool. It excels at preserving local structure but is computationally intensive and does not preserve global structure, meaning the distances between resulting clusters are not meaningful.[12]
  - **Uniform Manifold Approximation and Projection (UMAP):** The highly recommended approach. UMAP is a modern manifold learning technique that is significantly faster than t-SNE, preserves both local and global data structure, and can be integrated directly into scikit-learn pipelines for feature extraction.[14]
- Advanced Application: UMAP as a Feature Generation Engine:
  The common application of dimensionality reduction is to reduce the number of features to prevent overfitting. A more sophisticated application, and the one we propose here, is to use it as a feature generation tool. GBDT models build splits one feature at a time and can struggle to learn complex relationships that involve many features simultaneously. By applying UMAP to a specific, conceptually

related group of features (e.g., all 20+ features related to player behavior and investment), we can create a small number (e.g., 2-3) of new, dense features. These UMAP-generated features represent a projection of the high-dimensional "behavior space" onto its underlying manifold. A single UMAP feature might represent a meaningful axis, such as an axis from "cautious, low-volume play" to "aggressive, whale-dominated play." Feeding these powerful, pre-summarized latent features into the CatBoost model allows it to make more informed splits with less complexity, potentially boosting accuracy and improving interpretability by simplifying the final tree structures.14

- **Proposed Implementation:**
  1. Isolate all player-centric features from categories B, C, and D.
  2. Standardize these features.
  3. Fit a UMAP transformer to this subset, configured to output 2 or 3 components.
  4. Append these new features (e.g., umap_behavior_1, umap_behavior_2) to the main dataset before it is passed to the model training and selection pipeline.

III. Model Architecture and Selection: Balancing Performance and Interpretability

The choice of model architecture is the most critical decision in this project. It directly influences predictive accuracy, training time, deployment complexity, and—crucially—our ability to interpret the results and understand whale behavior. The recommendation is a pragmatic, performance-driven approach that prioritizes the strengths of different model families for this specific tabular, time-series problem.

3.1 Primary Recommendation: Gradient Boosted Decision Trees (GBDTs)

For structured, tabular datasets with mixed data types, GBDT models are the undisputed state-of-the-art. Recent empirical studies consistently show that well-tuned tree-based models like XGBoost, LightGBM, and CatBoost outperform deep learning models on the vast majority of tabular data problems.[45]

- **Justification for GBDT Dominance:**

    - **Exceptional Performance:** GBDTs excel at learning complex, non-linear decision boundaries and feature interactions directly from tabular data.
    - **Robustness and Efficiency:** They are generally faster to train than deep learning models, are robust to outliers, and can handle missing values natively. They do not require the extensive feature scaling and normalization that neural networks depend on.
    - **Superior Interpretability:** The primary advantage for this project is their interpretability. Tree-based models are inherently more transparent than neural networks, and their synergy with explanation frameworks like SHAP is unparalleled, allowing for precise, feature-level attribution of predictions.[7]
- Candidate Model Comparison and Recommendation:
  While several excellent GBDT libraries exist, CatBoost is the primary recommendation for this project due to its specific advantages in handling the given dataset.

Model	Key Algorithm	Categorical Handling	Speed	Overfitting Risk	Key Advantage for this Project
XGBoost	Level-wise (horizontal) tree growth [48]	Requires manual one-hot or label encoding.	Fast, but slower than LightGBM.	Medium	Industry standard, robust, large community.
LightGBM	Leaf-wise (vertical) tree growth [48]	Native support, but less sophisticated than CatBoost.	Fastest	Higher on small datasets due to leaf-wise growth.	Excellent for rapid experimentation.
CatBoost	Symmetric (balanced) tree growth [8]	**Ordered Boosting** and permutation-based target encoding.[8]	Slower training, but very fast prediction.	Low, due to symmetric trees and ordered boosting.	**Superior native handling of categorical features, preventing target leakage and simplifying**

					the pipeline.

Table 2: A comparative analysis of leading GBDT libraries. CatBoost's advanced categorical feature handling makes it the optimal choice for the mixed-type dataset in this project.

The key differentiator is CatBoost's sophisticated handling of categorical features. It employs a technique called "ordered boosting," which uses a random permutation of the training data to calculate target statistics, thereby preventing the target leakage that can plague standard target encoding methods.[8] This is highly relevant for our dataset, which contains numerous binary and multi-class categorical flags (`is_mixed_mode`, `is_sol_dominated`, etc.). This native capability eliminates the need for manual one-hot encoding, which can lead to a high-dimensional, sparse feature space that is less efficient for tree-based models.

3.2 Specialized Configuration for Imbalanced Data

With only 17% of games being profitable, the model must be explicitly instructed to pay more attention to this minority class. Failure to do so will result in a model with high accuracy but zero practical value, as it will simply learn to predict the majority "non-profitable" class most of the time.

- **Recommended Technique: Cost-Sensitive Learning:** This approach is preferred over data resampling techniques like SMOTE or random undersampling. Resampling alters the original data distribution and, critically for this project, can destroy the temporal dependencies between sequential games.[2] Cost-sensitive learning addresses the imbalance at the algorithm level by modifying the loss function.
- **Implementation in CatBoost:** The scale_pos_weight hyperparameter is the most direct way to implement cost-sensitive learning for binary classification. It increases the penalty for misclassifying the positive class.
  - Calculation: The standard starting point for this value is the ratio of the number of negative samples to positive samples. Given the dataset statistics,

    this is calculated as:
    scale_pos_weight=count(positive class)count(negative class)≈17%83%≈4.88
  - This value will be used as the central point in our hyperparameter search space, allowing the optimization process to find the precise optimal weight.[49]

3.3 Secondary Recommendation: Deep Learning for Sequential Analysis

While GBDTs are the primary recommendation, it would be imprudent to ignore the potential of deep learning models, which are specifically designed to learn patterns from sequential data.[50] An LSTM or Transformer model will serve as a powerful performance benchmark, ensuring that we are not leaving predictive power on the table by overlooking complex temporal patterns that hand-engineered features might miss.

- **Data Preparation for DL Models:** Deep learning models for sequences require the input data to be shaped into a 3D tensor of (samples, timesteps, features). This will be achieved by creating sliding windows from the game data. For example, to predict the outcome of game N, the input to the model will be the feature vectors for games [N-10, N-9,..., N-1], where the window size (10 in this case) is a hyperparameter.
- **Proposed Architecture: A Lightweight Time-Series Transformer:** While LSTMs are a valid choice, a modern Transformer-based architecture is recommended as it has shown superior performance in capturing long-range dependencies through its self-attention mechanism.[53] A full-blown Transformer is unnecessary; a lightweight, encoder-only architecture is sufficient and more efficient for this classification task.[55]
  1. **Input Embeddings:** A crucial step for handling mixed data types in a neural network. Categorical features will be passed through an embedding layer to create dense vector representations. Numerical features will be passed through a separate linear layer or simply concatenated.[4]
  2. **Positional Encoding:** Since the self-attention mechanism is permutation-invariant, positional encodings must be added to the input embeddings to inform the model of the sequence order of the games.[57]
  3. **Transformer Encoder:** A stack of 1-2 Transformer encoder blocks, each containing a multi-head self-attention layer and a position-wise feed-forward network. This is the core of the model where temporal patterns are learned.[59]
  4. **Classification Head:** The output from the Transformer encoder is pooled

(e.g., via global average pooling) and passed to a final dense layer with a sigmoid activation function to produce the binary prediction.

3.4 The Ultimate Predictor: A Stacked Ensemble

The debate between GBDTs and Deep Learning for tabular data often presents a false dichotomy. The most sophisticated approach recognizes that these model families capture different types of information. GBDTs excel at navigating the intricate, non-linear decision boundaries within a static feature vector for a single game. In contrast, DL models like Transformers excel at automatically learning temporal representations and dependencies across a sequence of games.

To harness the strengths of both, the ultimate recommendation is to build a **stacked ensemble**.[61]

- **Methodology:**
  1. **Level 0 Models:** Train the fully optimized CatBoost GBDT model and the lightweight Transformer model independently using our walk-forward cross-validation framework.
  2. **Generate Meta-Features:** For each fold in the cross-validation, generate the out-of-sample predictions from both the CatBoost and Transformer models. These predictions become the new input features for the next level.
  3. **Level 1 Model (Meta-Learner):** Train a simple meta-model, such as a Logistic Regression classifier, on these generated predictions. The meta-learner's task is to learn the optimal weights to assign to the GBDT and Transformer predictions to produce the final, most accurate output.
- **Strategic Advantage:** This approach allows each model to specialize. The GBDT focuses on the rich, intra-game features, while the Transformer focuses on inter-game temporal patterns. The meta-learner then acts as a final arbiter, intelligently blending their outputs to achieve a level of performance that is often unattainable by either model alone.

IV. A Rigorous Optimization and Validation Framework

A sophisticated model is only as good as the process used to tune and validate it. For a time-sensitive, imbalanced classification problem, adhering to a rigorous optimization and validation framework is not optional; it is essential for building a model that is robust, generalizable, and trustworthy. This section details the recommended protocols.

4.1 Advanced Hyperparameter Optimization Strategy

The process of finding the optimal set of hyperparameters is critical for maximizing model performance. Traditional methods like Grid Search are computationally infeasible and inefficient for modern models with large parameter spaces. We recommend a more intelligent, automated approach.

- **Technique Comparison and Recommendation:**
  - **Grid Search:** Exhaustively searches all possible combinations. Its computational cost grows exponentially with the number of parameters, making it impractical.[63]
  - **Random Search:** A step up from Grid Search, it randomly samples a fixed number of combinations. While more efficient, it provides no guarantee of finding an optimal region in the search space.[64]
  - **Bayesian Optimization:** The recommended approach. It treats hyperparameter tuning as an optimization problem, building a probabilistic surrogate model of the objective function (e.g., F1-score vs. hyperparameters). It uses past evaluation results to make informed decisions about which hyperparameters to try next, converging on the optimal solution far more efficiently than random or grid search.[63]
  - **Hyperband:** A bandit-based method that is also highly efficient, particularly for deep learning models. It works by adaptively allocating resources and using early stopping to prune unpromising trials.[67]
- Recommended Tool: Optuna:
  Optuna is a state-of-the-art Python framework for hyperparameter optimization. It implements advanced algorithms, including Tree-structured Parzen Estimators (TPE), a highly effective form of Bayesian optimization. Its key advantages for this project are its efficiency, ease of use, and, most importantly, its native support for multi-objective optimization.17
- Multi-Objective Optimization for Accuracy and Latency:
  This project has two competing objectives: maximizing predictive performance

and ensuring the model is fast enough for real-time inference. Optimizing for a single metric like F1-score might yield a highly complex model that is too slow for production. Multi-objective optimization addresses this directly.18

- **Defined Objectives:**
  1. **Objective 1 (Maximize):** F1-Score on the positive class, evaluated via walk-forward cross-validation.
  2. **Objective 2 (Minimize):** Average inference latency per prediction.
- **Outcome (Pareto Front):** Instead of a single "best" model, this process yields a set of non-dominated solutions known as the **Pareto front**. Each point on this front represents an optimal trade-off where one objective cannot be improved without worsening the other. This provides the project team with a menu of high-performing models, allowing for an informed, data-driven decision. For example, a model with a 0.87 F1-score and 5ms latency might be chosen over a model with a 0.88 F1-score and 25ms latency.[18]

Hyperparameter	Optuna Sampler	Suggested Range	Justification
iterations	trial.suggest_int	500 - 2000	Number of boosting rounds (trees). More trees can improve accuracy but increase training time and risk of overfitting.
learning_rate	trial.suggest_loguniform	1e-3 - 1e-1	Controls the step size at each iteration. A log-uniform search is standard for this parameter.
depth	trial.suggest_int	4 - 10	Maximum depth of the trees. Deeper trees capture more complex interactions but can overfit.
l2_leaf_reg	trial.suggest_loguniform	1 - 10	L2 regularization term on weights. Helps prevent overfitting.
scale_pos_weight	trial.suggest_float	3.0 - 7.0	The key parameter for handling class imbalance. The search is centered

			around the calculated ratio of ~4.88.

Table 3: A recommended starting search space for Optuna-based hyperparameter optimization of the CatBoost model. This provides a concrete and actionable configuration for the development team.

4.2 Time-Aware Cross-Validation Protocol

This is the most critical aspect of the validation framework. Using standard cross-validation on time-series data is a fundamental methodological error that leads to **data leakage**. Standard k-fold CV shuffles data, allowing the model to be trained on data from the future to predict the past. This results in overly optimistic performance metrics and a model that will fail catastrophically in a real-world, forward-looking deployment.[73]

- Recommended Strategy: Walk-Forward Validation (also known as Rolling Forecast or Expanding Window CV):
  This method rigorously respects the chronological order of the data. The dataset is split into a series of folds, where each fold consists of a training set containing all data up to a certain point in time, and a test set containing the data immediately following that point.1
  - **Implementation with sklearn.model_selection.TimeSeriesSplit:** This scikit-learn utility automates the process. For a 5-split validation, the process would be:
    - **Fold 1:** Train on games 1-800, Test on games 801-1600.
    - **Fold 2:** Train on games 1-1600, Test on games 1601-2400.
    - **Fold 3:** Train on games 1-2400, Test on games 2401-3200.
    - And so on. The training window expands, always using all available past data to predict the next block of future data.
- **Introducing a Gap:** To more realistically simulate a production environment where there might be a delay between when a game occurs and when its features are available for prediction, a "gap" can be introduced between the training and testing sets. This ensures the model is not tested on data that is immediately adjacent to its training data. Libraries like tscv provide classes like GapRollForward for this purpose.[15]
- **Pipeline Integrity:** It is of paramount importance that any data preprocessing

steps (e.g., feature scaling with StandardScaler, fitting the UMAP transformer) are fit *only* on the training data for each specific fold. The fitted transformer is then used to transform the test data for that fold. Fitting these transformers on the entire dataset before splitting would constitute a major form of data leakage.[73]

4.3 Evaluation Framework and Performance Targets

A robust evaluation framework requires using metrics appropriate for the specific business problem and data characteristics.

- **Metrics for Imbalanced Classification:**
  - **Primary Metric: F1-Score.** The F1-score is the harmonic mean of precision and recall. For this problem, where identifying profitable games (the positive class) is the goal, the F1-score provides a balanced measure of the model's ability to find these rare events without being overly penalized by false positives (which precision controls) or false negatives (which recall controls).
  - **Secondary Metric: Precision-Recall Area Under Curve (PR-AUC).** Unlike the standard ROC-AUC, which can be misleadingly high on imbalanced datasets, the PR-AUC focuses on the trade-off between precision and recall for the positive class. A higher PR-AUC indicates superior performance on the minority class of interest.
  - **Supporting Metrics:** A full **confusion matrix** will be generated for each fold to allow for visual inspection of true positives, false positives, true negatives, and false negatives. Standard **Precision**, **Recall**, and **Overall Accuracy** will also be reported for a complete performance picture.
- **Final Validation on a Hold-Out Set:** The ultimate test of a time-series model is its performance on data it has never seen, which is chronologically in the future. After the best model and hyperparameters have been selected using walk-forward cross-validation, a final model will be trained on the *entire* historical dataset. This model's performance will then be evaluated one time on a completely untouched **hold-out set** consisting of the most recent games (e.g., the last 10% of the data). This simulates true deployment and provides the most reliable estimate of future performance.[78]
- **Performance Targets:** Based on the strong existing signal (+13.0 pp advantage from the has_real_sol_whale feature), the project should aim for ambitious but achievable performance targets on the hold-out set:
  - **F1-Score (Positive Class):** > 0.85

    - **PR-AUC:** > 0.88
    - **Overall Accuracy:** > 80%

V. Unlocking Whale Behavior: A Framework for Model Explainability

A core objective of this project extends beyond prediction to understanding. The machine learning model should serve as a powerful lens through which to analyze and comprehend the complex behaviors of SOL whales. This requires moving the model from a "black box" to a transparent, interpretable system.

5.1 Global and Local Explanations with SHAP

SHAP (SHapley Additive exPlanations) has emerged as the industry standard for explaining the output of complex models like GBDTs. It is based on a solid game-theoretic foundation and provides consistent and locally accurate feature attributions, making it superior to other methods like LIME, which can be less stable.[6]

- **Implementation with CatBoost:** The shap library has optimized TreeExplainers that work efficiently with GBDT models, including CatBoost. The process involves training the final model, initializing the explainer with this model, and then calculating SHAP values for the dataset.[9]
- Answering Key Business Questions with SHAP Plots:
  SHAP provides a suite of visualizations that can translate model predictions into actionable business intelligence.
    - **Global Feature Importance (What matters most overall?):** The SHAP summary bar plot (shap.summary_plot(plot_type="bar")) provides a global ranking of features based on their average impact on model output. This will quantify the importance of the has_real_sol_whale feature relative to all others, including our newly engineered features. It answers the high-level question: "What are the top drivers of profitability across all games?".[9]
    - **Feature Effects and Distribution (How do features work?):** The SHAP beeswarm plot (the default shap.summary_plot()) is more insightful. It shows not only the magnitude of a feature's impact but also its direction. Each dot is

a prediction for a single game. For example, we can visualize that high values of sol_whale_concentration (colored red) correspond to high positive SHAP values, meaning they strongly push the prediction towards "profitable." This plot reveals the distribution and directionality of feature effects.[9]

  - **Feature Interactions (How do features work *together*?):** The SHAP dependence plot (shap.dependence_plot()) is crucial for uncovering complex strategies. By plotting a feature's SHAP value against its actual value and coloring the points by a second feature, we can visualize interaction effects. For instance, plotting sol_whales and coloring by volatility might reveal that the positive impact of having many whales is significantly amplified in high-volatility games. This moves beyond simple feature importance to understanding the *context* in which features are powerful.[19]
  - **Local Prediction Explanations (Why was *this game* profitable?):** The SHAP force plot (shap.force_plot()) provides a complete explanation for a single prediction. It shows the base value (average prediction) and how each feature's value for that specific game pushed the prediction higher (in red) or lower (in blue) to arrive at the final output. This is invaluable for case studies, debugging, and providing transparent reasoning to stakeholders.[9]

5.2 Identifying Anomalous Whale Behavior

The explainability framework can be extended beyond understanding average behavior to actively discovering *unusual* or *novel* behavior. This is achieved by applying anomaly detection techniques not to the input features, but to the model's explanations themselves.

- Methodology: Anomaly Detection on SHAP Values:
  The core idea is that while the model learns the common patterns that lead to profitability, any game that is profitable for a highly unusual reason represents a deviation from these patterns and is therefore an anomaly worth investigating.
  1. **Generate "Explanation Signatures":** For every game in the dataset, calculate its corresponding vector of SHAP values. This vector is a high-dimensional "explanation signature" that describes how the model arrived at its prediction for that game.
  2. **Train an Anomaly Detector:** Train an unsupervised anomaly detection algorithm, such as **Isolation Forest**, on the matrix of these SHAP value vectors. Isolation Forest is well-suited for this task as it is efficient and

   performs well on high-dimensional data without requiring assumptions about the data distribution.[81] It works by building random trees and identifying anomalies as points that are easier to "isolate" from the rest.
3. **Deploy for Discovery:** In production, for each new game, the system will first generate a prediction and then its SHAP value explanation. This explanation vector is then fed into the trained Isolation Forest model.
4. **Flag Anomalous Explanations:** If the Isolation Forest flags the SHAP vector as an anomaly, it triggers an alert. This alert signifies that the *reasoning* behind the model's prediction is unusual, even if the prediction itself was confident.

- From Prediction to Discovery Engine:
  This system elevates the machine learning model from a simple predictive tool to a strategic discovery engine. It automates the process of finding "known unknowns"—instances that conform to the target but for unexpected reasons. For example, the model might confidently predict a game will be profitable, a prediction primarily driven by the has_real_sol_whale feature. This is a normal explanation. However, another game might also be predicted as profitable with high confidence, but the SHAP explanation reveals that the prediction was driven almost entirely by features related to FREE (practice) players and an unusual buy_sell_ratio. The Isolation Forest would flag this explanation as an anomaly. This could alert analysts to a new, emerging phenomenon, such as coordinated activity by practice accounts attempting to manipulate game outcomes, a pattern not yet well-represented in the training data. This creates a powerful, human-in-the-loop feedback system where the model flags novel behaviors for human analysis, which can then inform the next iteration of feature engineering and model training. This is a far more nuanced and powerful approach than simply running outlier detection on the raw input features.83

VI. Production Deployment and Operationalization (MLOps)

A high-performing model is only valuable if it can be reliably and efficiently integrated into a production environment. This requires a robust MLOps (Machine Learning Operations) strategy that addresses low-latency serving, safe deployment, and long-term maintenance through monitoring and automated retraining.

6.1 Real-Time Inference Architecture

The application demands real-time predictions to inform live trading or betting strategies. While the user's mention of "microsecond-latency" is likely aspirational for a typical web stack, we will design a system architected for **low-millisecond latency (e.g., <10 ms)**, which is achievable and meets the practical requirements of real-time applications.[20]

- **Proposed Architecture Blueprint:**
  1. **Model Optimization and Serialization:** The final trained CatBoost model will be serialized. For maximum performance, we can explore converting the model to an optimized inference format like **ONNX (Open Neural Network Exchange)**. ONNX provides a standardized format that can be run by high-performance inference engines.
  2. **High-Performance Serving Framework:** The model will be wrapped in a lightweight, asynchronous Python web framework. **FastAPI** is highly recommended over more traditional frameworks like Flask, as its asynchronous nature is ideal for I/O-bound operations and achieving high throughput with low latency.
  3. **Containerization:** The FastAPI application, along with the model artifacts and all necessary dependencies, will be packaged into a **Docker container**. This ensures a consistent, portable, and isolated environment for deployment.
  4. **Scalable Deployment Platform:** The Docker container will be deployed on a managed, scalable platform. **Kubernetes** (e.g., Google Kubernetes Engine - GKE, Amazon Elastic Kubernetes Service - EKS) is an excellent choice as it provides auto-scaling, self-healing, and robust orchestration capabilities. For simpler use cases, a serverless compute platform like **AWS Lambda** or **Google Cloud Functions** could also be considered, as they automatically manage scaling based on request volume.

6.2 A/B Testing Framework for Model Rollouts

Deploying a new machine learning model directly into production is inherently risky. A new model, even with excellent offline validation metrics, may behave unexpectedly on live data. An A/B testing framework is essential for de-risking deployments and making data-driven decisions about model rollouts.[85]

- **Phased Rollout Methodology:**
  1. **Shadow Deployment:** The new model (Model B) is deployed alongside the existing system (Model A). It receives a copy of live production traffic and makes predictions, but these predictions are only logged and not used to make decisions. This phase is critical for validating the model's technical performance (e.g., latency, error rate, stability) under real-world load without any business impact.
  2. **Canary Release (Live A/B Test):** Once the model is technically validated in shadow mode, we begin a live A/B test. A small fraction of traffic (e.g., 5%) is routed to Model B, while the remaining 95% continues to be served by Model A (the control group).
  3. **Performance Evaluation:** The performance of both models is meticulously tracked and compared on key business metrics (e.g., ROI of strategies based on predictions, actual profitability of predicted games) and statistical metrics (e.g., live F1-score, precision, recall).
  4. **Gradual Rollout:** If Model B demonstrates statistically significant superior performance over Model A, the traffic routed to it is gradually increased (e.g., to 10%, 25%, 50%, and finally 100%). This incremental approach minimizes risk and allows for continuous monitoring as the model is exposed to a larger and more diverse set of data.

6.3 Monitoring, Drift Detection, and Retraining

The cryptocurrency gambling environment is highly dynamic and non-stationary. Player strategies evolve, new whales enter the market, and the platform itself may change. This inevitably leads to **concept drift**, a phenomenon where the statistical relationship between the input features and the target variable changes over time, causing the model's predictive performance to decay.[22] A proactive monitoring and retraining strategy is the only way to ensure the model remains accurate and valuable over the long term.[21]

- Comprehensive Monitoring Dashboard:
  A dedicated monitoring dashboard will be established to track the health of the production model in near real-time. Key components to monitor include:
  - **Data Drift:** Tracking the statistical distributions of key input features (e.g.,

mean and variance of volatility, distribution of sol_whale_concentration). A significant shift in these distributions from the training data indicates that the model is seeing data it was not trained on.
    - **Prediction Drift:** Tracking the distribution of the model's output probabilities. A sudden change—for example, if the model begins predicting "profitable" 50% of the time when it historically predicted it 20% of the time—is a strong signal of potential concept drift or an issue with the input data pipeline.
    - **Model Performance Metrics:** When ground truth becomes available (i.e., when a game concludes), track the model's F1-score, precision, and recall over rolling time windows. A clear downward trend is the most direct evidence of model decay.
- Automated Drift Detection:
  Manual monitoring is insufficient. We will implement automated drift detection using a dedicated Python library like frouros or evidently.ai. These libraries can be configured to run statistical tests (e.g., Kolmogorov-Smirnov test for numerical features, Chi-Square test for categorical features) comparing the distribution of recent production data to a reference window (e.g., the original training data). If the p-value of a test drops below a predefined threshold, an automated alert is triggered.89
- Automated Retraining Strategy:
  The MLOps pipeline will incorporate a robust retraining strategy based on both performance degradation and a regular schedule.
    - **Trigger-Based Retraining:** The drift detection alerts will be configured to automatically trigger a retraining pipeline. If significant data drift is detected or if the live F1-score drops below a critical threshold (e.g., 0.80), the system will automatically retrain the model on the latest available data.
    - **Scheduled Retraining:** In addition to reactive triggers, a periodic retraining schedule (e.g., weekly) will be established. This ensures the model is regularly refreshed with new data, proactively combating gradual concept drift even before it becomes severe enough to trigger an alert. The entire walk-forward validation and hyperparameter optimization process will be codified into this automated pipeline to ensure that each new model version is rigorously tested before being proposed for deployment.

VII. Implementation Plan and Risk Assessment

This final section provides a practical roadmap for executing the proposed strategy, including a project timeline, resource estimates, and a proactive assessment of potential risks and their mitigation strategies.

7.1 Phased Implementation Roadmap

A phased approach is recommended to ensure a structured and manageable development process, with clear milestones and deliverables at each stage.

- **Phase 1: Model Development & Validation (4-6 weeks)**
    - **Week 1-2:** Full Dataset Processing and Advanced Feature Engineering.
        - Execute the feature engineering pipeline on the complete 4,789-game dataset.
        - Generate all proposed polynomial, interaction, and domain-specific features.
        - Implement and apply UMAP for latent feature creation.
    - **Week 3:** GBDT (CatBoost) Model Training and Optimization.
        - Implement the hybrid feature selection strategy (Embedded + RFE).
        - Conduct multi-objective hyperparameter optimization using Optuna and the walk-forward validation framework.
    - **Week 4:** Deep Learning (Transformer) Benchmark Development.
        - Develop the data reshaping pipeline to create sequences.
        - Train and tune the lightweight Transformer model.
    - **Week 5-6:** Final Validation, Stacking, and Explainability Analysis.
        - Perform final validation of the best GBDT and DL models on the hold-out set.
        - Develop and test the stacked ensemble model.
        - Conduct a thorough explainability analysis using SHAP on the final selected model.
        - Prepare a final model performance report.
- **Phase 2: Productionization & MLOps Pipeline (4 weeks)**
    - **Week 7-8:** Real-Time Inference Service Development.
        - Serialize the final model artifact.
        - Develop and test the containerized FastAPI inference service.
        - Conduct load testing to ensure latency targets are met.
    - **Week 9-10:** MLOps Infrastructure Setup.
        - Deploy monitoring dashboards for data, prediction, and concept drift.

        - Configure automated drift detection alerts.
        - Build and test the automated retraining and validation pipeline.
- **Phase 3: Deployment & Live Operations (Ongoing)**
    - **Week 11:** Shadow Deployment.
        - Deploy the new model into the production environment in shadow mode to monitor live performance without business impact.
    - **Week 12 onwards:** A/B Testing and Gradual Rollout.
        - Begin the A/B test with a small percentage of live traffic.
        - Continuously monitor performance and gradually increase traffic to the new model based on data-driven evidence of its superiority.

7.2 Resource Requirements

- **Personnel:**
    - **2-3 Machine Learning Engineers / Data Scientists:** Responsible for feature engineering, model development, optimization, and analysis.
    - **1 MLOps / DevOps Engineer:** Responsible for building and maintaining the production infrastructure, deployment pipelines, and monitoring systems.
- **Infrastructure:**
    - **Cloud Computing Platform:** A major cloud provider (e.g., AWS, GCP, Azure) is required.
    - **GPU Instances:** Necessary for efficient training and tuning of deep learning models and potentially for accelerating GBDT training.
    - **Scalable Compute:** A container orchestration platform like Kubernetes or a serverless platform for hosting the inference service.
    - **MLOps Tooling:** Subscriptions or managed services for experiment tracking (e.g., Neptune.ai, Comet), monitoring, and workflow orchestration.

7.3 Risk Register and Mitigation Strategies

A proactive approach to risk management is crucial for project success. The following table outlines potential risks and their corresponding mitigation strategies.

Risk ID	Risk Description	Likelihood	Impact	Mitigation

				Strategy
TEC-01	**Model Underperformance:** The final model fails to meet the target F1-score (>0.85) on the hold-out set.	Medium	High	1. Revisit feature engineering: explore additional domain-specific features or more complex interactions. 2. Expand hyperparameter search space in Optuna. 3. Explore more complex model architectures (e.g., larger ensembles, deeper networks). 4. Prioritize collection of more training data, especially for the minority class.
TEC-02	**High Inference Latency:** The production model's latency exceeds the <10ms target, making it unsuitable for real-time use.	Medium	High	1. Apply model optimization techniques such as quantization (e.g., converting model weights to 8-bit integers) or pruning. 2. Utilize a high-performance inference server like NVIDIA Triton Inference Server. 3. Vertically scale the serving instances (use

				more powerful CPUs/GPUs). 4. Horizontally scale the service by increasing the number of container replicas.
OPS-01	**Rapid Concept Drift:** The gambling environment evolves so quickly that weekly or monthly retraining is insufficient to maintain model performance.	High	High	1. Increase the frequency of scheduled retraining (e.g., to daily). 2. Lower the sensitivity threshold for automated drift detection triggers. 3. Investigate and develop an online learning component where the model can be updated in near-real-time with new game data (this is a significant architectural change).
SEC-01	**Adversarial Attacks:** Malicious actors identify the key predictive features (e.g., whale metrics) and begin to spoof them to manipulate model predictions.	Low	High	1. Work with the platform engineering team to develop more robust, harder-to-spoof behavioral features. 2. Implement the anomaly detection on SHAP values

				system; this can flag predictions that are driven by manipulated or unusual feature combinations. 3. Maintain a "honeypot" or monitoring system to detect patterns of behavior that seem designed to exploit the model.
BIZ-01	**Interpretability Failure:** SHAP analysis reveals that the model is relying on spurious correlations or uninterpretable features, undermining trust in its insights.	Low	Medium	1. Refine the feature selection process to penalize complexity and prioritize features with clear business meaning. 2. Use SHAP interaction plots to ensure that feature effects align with domain expertise and business logic. 3. If a feature is highly predictive but unexplainable, conduct an ablation study to quantify its impact and make an informed decision on whether to

				include it.

Table 4: A proactive risk register outlining potential technical, operational, security, and business risks alongside their corresponding mitigation strategies.

VIII. Conclusion

This report has outlined a comprehensive, multi-faceted strategy for developing and deploying a high-performance machine learning system to predict profitable games on the Rugs.fun platform. The core of the recommendation is a pragmatic and evidence-based approach that prioritizes techniques proven to be effective for the specific challenges of this problem domain: tabular time-series data with significant class imbalance and a dual requirement for accuracy and interpretability.

The recommended pathway begins with a sophisticated feature engineering process that goes beyond the existing feature set to capture non-linear, temporal, and latent behavioral patterns. This is followed by the selection of a **CatBoost GBDT model**, chosen for its state-of-the-art performance on tabular data and its superior handling of the dataset's mixed data types. The significant class imbalance will be addressed at the algorithmic level using cost-sensitive learning.

To ensure the model is robust and its performance is not overestimated, a strict **Walk-Forward Cross-Validation** protocol is mandated. Model optimization will be handled by a **multi-objective Bayesian optimization** process, intelligently balancing the competing needs of predictive accuracy (F1-Score) and production-level inference speed.

Crucially, this plan moves beyond simple prediction. By integrating **SHAP** for model explainability and augmenting it with an innovative **anomaly detection system based on SHAP values**, the resulting system will not only predict outcomes but will also serve as a powerful discovery engine for understanding novel whale behaviors and market dynamics.

Finally, the report provides a clear blueprint for **productionalization**, detailing a low-latency serving architecture, a safe A/B testing framework for deployment, and a robust MLOps pipeline for monitoring and automated retraining to combat concept drift. By following this strategic roadmap, the project is well-positioned to develop a system that provides a durable, data-driven competitive advantage, enhancing both profitability and strategic insight into the dynamics of the cryptocurrency gambling ecosystem.

Works cited

1. Cross Validation in Time Series. Cross Validation: | by Soumya Shrivastava | Medium, accessed July 12, 2025, https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4
2. An Embedded Feature Selection Method for Imbalanced Data Classification, accessed July 12, 2025, https://www.ieee-jas.net/article/doi/10.1109/JAS.2019.1911447
3. Improved Feature-Selection Method Considering the Imbalance Problem in Text Categorization - PMC, accessed July 12, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC4058251/
4. Handling Mixed Data in Machine Learning: A Comprehensive Guide ..., accessed July 12, 2025, https://medium.com/@paghadalsneh/handling-mixed-data-in-machine-learning-a-comprehensive-guide-89a174482415
5. Handling Machine Learning Categorical Data with Python Tutorial - DataCamp, accessed July 12, 2025, https://www.datacamp.com/tutorial/categorical-data
6. Explainable AI: Intro to LIME & SHAP - Kaggle, accessed July 12, 2025, https://www.kaggle.com/code/khusheekapoor/explainable-ai-intro-to-lime-shap
7. Techniques for Explainable AI: LIME and SHAP - Unnat Bak (Founder @ Revscale, TABS Suite) Growth Hacking and Venture Advisory, accessed July 12, 2025, https://www.unnatbak.com/blog/techniques-for-explainable-ai-lime-and-shap
8. When to Choose CatBoost Over XGBoost or LightGBM - neptune.ai, accessed July 12, 2025, https://neptune.ai/blog/when-to-choose-catboost-over-xgboost-or-lightgbm
9. Catboost tutorial — SHAP latest documentation - Read the Docs, accessed July 12, 2025, https://shap.readthedocs.io/en/latest/example_notebooks/tabular_examples/tree_based_models/Catboost%20tutorial.html
10. Feature Selection Techniques in Machine Learning - GeeksforGeeks, accessed July 12, 2025, https://www.geeksforgeeks.org/machine-learning/feature-selection-techniques-in-machine-learning/
11. Polynomial Regression: How to Use Polynomial Regression for Investment Forecasting - FasterCapital, accessed July 12, 2025, https://fastercapital.com/content/Polynomial-Regression--How-to-Use-Polynomial-Regression-for-Investment-Forecasting.html
12. How to Visualize Your Data with Dimension Reduction Techniques - Voxel51, accessed July 12, 2025,

https://voxel51.com/blog/how-to-visualize-your-data-with-dimension-reduction-techniques
13. CMC | Free Full-Text | Optimizing Forecast Accuracy in Cryptocurrency Markets: Evaluating Feature Selection Techniques for Technical Indicators, accessed July 12, 2025, https://www.techscience.com/cmc/v83n2/60595/html
14. UMAP as a Feature Extraction Technique for Classification — umap ..., accessed July 12, 2025, https://umap-learn.readthedocs.io/en/latest/auto_examples/plot_feature_extraction_classification.html
15. GapRollForward — Time Series Cross-Validation 0.1.3 documentation, accessed July 12, 2025, https://tscv.readthedocs.io/en/latest/tutorial/roll_forward.html
16. 3.1. Cross-validation: evaluating estimator performance — scikit ..., accessed July 12, 2025, https://scikit-learn.org/stable/modules/cross_validation.html#time-series-split
17. Hyperparameter Optimization of LightGBM Model with Optuna | by ..., accessed July 12, 2025, https://medium.com/@hazallgultekin/hyperparameter-optimization-of-lightgbm-model-with-optuna-e5ad9f5688ff
18. Multi-Objective Optimization in Machine Learning: Beyond Single ..., accessed July 12, 2025, https://www.kaggle.com/discussions/general/536057
19. How to Combine Scikit-learn, CatBoost, and SHAP for Explainable ..., accessed July 12, 2025, https://machinelearningmastery.com/how-to-combine-scikit-learn-catboost-and-shap-for-explainable-tree-models/
20. Breaking Down "Real-Time" Machine Learning Systems | FeatureForm, accessed July 12, 2025, https://www.featureform.com/post/breaking-down-real-time-machine-learning-systems-mlops
21. randomtrees.medium.com, accessed July 12, 2025, https://randomtrees.medium.com/mastering-model-retraining-in-mlops-4bb961ee7070#:~:text=Model%20retraining%2C%20in%20essence%2C%20involves,present%20in%20the%20underlying%20data.
22. What is concept drift in ML, and how to detect and address it - Evidently AI, accessed July 12, 2025, https://www.evidentlyai.com/ml-in-production/concept-drift
23. A Feature Selection Method for Multi-Dimension Time-Series Data, accessed July 12, 2025, https://project.inria.fr/aaltd20/files/2020/08/AALTD_20_paper_Kathirgamanathan.pdf
24. Feature Selection: Embedded Methods | by Elli Tzini | Analytics Vidhya | Medium, accessed July 12, 2025, https://medium.com/analytics-vidhya/feature-selection-embedded-methods-a7940036973f
25. FeatureSelection/Data Science Lifecycle - Feature Selection (Filter, Wrapper, Embedded and Hybrid Methods).ipynb at master - GitHub, accessed July 12,

2025, https://github.com/codingnest/FeatureSelection/blob/master/Data%20Science%20Lifecycle%20-%20Feature%20Selection%20(Filter%2C%20Wrapper%2C%20Embedded%20and%20Hybrid%20Methods).ipynb
26. Feature Selection and Engineering for Time Series Data - Bocconi Students Investment Club, accessed July 12, 2025, https://bsic.it/feature-selection-and-engineering-for-time-series-data/
27. Wrapper Methods - Feature Selection - GeeksforGeeks, accessed July 12, 2025, https://www.geeksforgeeks.org/machine-learning/wrapper-methods-feature-selection/
28. Master Feature Selection Part 2: Wrapper Methods | by hassane ..., accessed July 12, 2025, https://medium.com/@Hassane_01/introduction-to-wrapper-methods-in-machine-learning-20be72b3f5c6
29. Wrapper Method — Feature Selection | by Nimisha Singh - AI Mind, accessed July 12, 2025, https://pub.aimind.so/wrapper-method-feature-selection-8dfead854ac5
30. Polynomial Regression: Definition, Formula & Real-World Examples - upGrad, accessed July 12, 2025, https://www.upgrad.com/blog/polynomial-regression/
31. Polynomial Regression for Prediction | by Hey Amit - Medium, accessed July 12, 2025, https://medium.com/@heyamit10/polynomial-regression-for-prediction-07f6e56a46ef
32. Implementation of Polynomial Regression - GeeksforGeeks, accessed July 12, 2025, https://www.geeksforgeeks.org/machine-learning/python-implementation-of-polynomial-regression/
33. Feature Engineering in Machine Learning | by Preeti - Medium, accessed July 12, 2025, https://medium.com/@preeti.rana.ai/the-role-of-feature-engineering-in-machine-learning-d024587e6410
34. Predicting Future Cryptocurrency Prices Using Machine Learning ..., accessed July 12, 2025, https://www.scirp.org/journal/paperinformation?paperid=128965
35. Feature-Driven vs Language-Based AI Online Gambling Addiction ..., accessed July 12, 2025, https://www.preprints.org/manuscript/202506.0883/v1
36. Machine Learning-Based Approach for the Gambling Problem Identification | Vietnam Journal of Computer Science - World Scientific Publishing, accessed July 12, 2025, https://www.worldscientific.com/doi/10.1142/S2196888825500034
37. Feature Engineering for Stock Volatility Prediction - About DolphinDB, accessed July 12, 2025, https://docs.dolphindb.com/en/Tutorials/metacode_derived_features.html
38. Feature selection for time series data - Cross Validated - Stats Stackexchange, accessed July 12, 2025, https://stats.stackexchange.com/questions/144116/feature-selection-for-time-series-data

39. The PCA Trick with Time-Series | Towards Data Science, accessed July 12, 2025, https://towardsdatascience.com/the-pca-trick-with-time-series-d40d48c69d28/
40. PCA vs UMAP vs t-SNE - biostatsquid.com, accessed July 12, 2025, https://biostatsquid.com/pca-umap-tsne-comparison/
41. Introduction to t-SNE: Nonlinear Dimensionality Reduction and Data Visualization, accessed July 12, 2025, https://www.datacamp.com/tutorial/introduction-t-sne
42. Mastering Dimensionality Reduction: A Comprehensive Guide to PCA, t-SNE, and UMAP, accessed July 12, 2025, https://naveen-malla.medium.com/mastering-dimensionality-reduction-a-comprehensive-guide-to-pca-t-sne-and-umap-211ace83c28e
43. Intro to PCA, t-SNE & UMAP - Kaggle, accessed July 12, 2025, https://www.kaggle.com/code/samuelcortinhas/intro-to-pca-t-sne-umap
44. How to Use UMAP — umap 0.5.8 documentation, accessed July 12, 2025, https://umap-learn.readthedocs.io/en/latest/basic_usage.html
45. Is Deep Learning finally better than Decision Trees on Tabular Data? - arXiv, accessed July 12, 2025, https://arxiv.org/html/2402.03970v2
46. Optimizing Model Performance: When to Use Ensemble Methods vs. Deep Learning? | Kaggle, accessed July 12, 2025, https://www.kaggle.com/discussions/questions-and-answers/411552
47. A Closer Look at Deep Learning Methods on Tabular Datasets - arXiv, accessed July 12, 2025, https://arxiv.org/html/2407.00956v2
48. XGBoost vs LightGBM: How Are They Different - neptune.ai, accessed July 12, 2025, https://neptune.ai/blog/xgboost-vs-lightgbm
49. python - catboost classifier for class imbalance? - Stack Overflow, accessed July 12, 2025, https://stackoverflow.com/questions/59746304/catboost-classifier-for-class-imbalance
50. LSTM for Text Classification? - Analytics Vidhya, accessed July 12, 2025, https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/
51. Deep Learning for Time Series Forecasting - Machine Learning Mastery, accessed July 12, 2025, https://machinelearningmastery.com/deep-learning-for-time-series-forecasting/
52. A survey of deep learning applications in cryptocurrency - PMC, accessed July 12, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC10726249/
53. [2502.03383] Transformers and Their Roles as Time Series Foundation Models - arXiv, accessed July 12, 2025, https://arxiv.org/abs/2502.03383
54. Transformer Model in Time-Series Analysis with tsfresh, Mann-Whitney Test and Benjamini-Yekutieli procedure | by 232 | Medium, accessed July 12, 2025, https://medium.com/@adamchen564/transformer-model-in-time-series-analysis-with-tsfresh-mann-whitney-test-and-benjamini-yekutieli-a1812a9648dc
55. Time Series Classification with Transformers - Center for Applied AI - University of Kentucky, accessed July 12, 2025, https://caai.ai.uky.edu/time-series-classification-with-transformers/
56. Time Series Forecasting with a Basic Transformer Model in PyTorch | by Kaan Aslan, accessed July 12, 2025,

https://medium.com/@mkaanaslan99/time-series-forecasting-with-a-basic-transformer-model-in-pytorch-650f116a1018
57. Positional Encoding in Transformer-Based Time Series Models: A Survey - arXiv, accessed July 12, 2025, https://arxiv.org/abs/2502.12370
58. Transformers in Time-series Analysis: A Tutorial, accessed July 12, 2025, https://arxiv.org/abs/2205.01138
59. Transformers in Time Series: A Survey - IJCAI, accessed July 12, 2025, https://www.ijcai.org/proceedings/2023/0759.pdf
60. sgrvinod/a-PyTorch-Tutorial-to-Transformers: Attention Is All You Need - GitHub, accessed July 12, 2025, https://github.com/sgrvinod/a-PyTorch-Tutorial-to-Transformers
61. Ensemble Learning: How Combining Multiple Models Can Improve ..., accessed July 12, 2025, https://medium.com/@data-overload/ensemble-learning-how-combining-multiple-models-can-improve-your-machine-learning-results-7683e4ea6bef
62. Guide to Ensemble Learning (with Python codes)- Analytics Vidhya, accessed July 12, 2025, https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/
63. Hyperparameter Tuning - GeeksforGeeks, accessed July 12, 2025, https://www.geeksforgeeks.org/machine-learning/hyperparameter-tuning/
64. Hyperparameter Tuning in Python: a Complete Guide - neptune.ai, accessed July 12, 2025, https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide
65. How to Optimize Hyperparameter Search Using Bayesian ..., accessed July 12, 2025, https://neptune.ai/blog/how-to-optimize-hyperparameter-search
66. Hyperparameter Tuning With Bayesian Optimization - Comet, accessed July 12, 2025, https://www.comet.com/site/blog/hyperparameter-tuning-with-bayesian-optimization/
67. Hyperband Hyperparameter Optimization | by Hey Amit | Medium, accessed July 12, 2025, https://medium.com/@heyamit10/hyperband-hyperparameter-optimization-d7bd66faa8e8
68. Speed-up hyperparameter tuning in deep learning with Keras ..., accessed July 12, 2025, https://analyticsindiamag.com/ai-trends/speed-up-hyperparameter-tuning-in-deep-learning-with-keras-hyperband-tuner/
69. Optimize LightGBM with Optuna - How to do now ? - Inside Machine Learning, accessed July 12, 2025, https://inside-machinelearning.com/en/optimize-lightgbm-optuna/
70. Multi-Objective - AutoML.org, accessed July 12, 2025, https://www.automl.org/dl-2-0/multi-objective/
71. Multi-Objective Optimization for Deep Learning : A Guide ..., accessed July 12, 2025,

https://www.geeksforgeeks.org/deep-learning/multi-objective-optimization-for-deep-learning-a-guide/
72. Multi-Objective Hyperparameter Optimization in Machine Learning – An Overview - arXiv, accessed July 12, 2025, https://arxiv.org/pdf/2206.07438
73. Stop the spill: The blueprint for eradicating data leakage | by ..., accessed July 12, 2025, https://medium.com/data-science-at-microsoft/stop-the-spill-the-blueprint-for-eradicating-data-leakage-6f924e543a95
74. Data Leakage In Machine Learning: Examples & How to Protect ..., accessed July 12, 2025, https://airbyte.com/data-engineering-resources/what-is-data-leakage
75. Data leakage in pre-trained forecasting models, accessed July 12, 2025, https://cienciadedatos.net/documentos/py63-data-leakage-pre-trained-forecasting-models
76. medium.com, accessed July 12, 2025, https://medium.com/@kylejones_47003/data-leakage-lookahead-bias-and-causality-in-time-series-analytics-76e271ba2f6b#:~:text=Data%20leakage%20occurs%20when%20information,the%20temporal%20nature%20of%20data.
77. Can you explain the process of model validation in time series forecasting? - sunrise classes, accessed July 12, 2025, https://www.sunriseclassesiss.com/post/can-you-explain-the-process-of-model-validation-in-time-series-forecasting
78. Out-of-time validation modeling: DataRobot docs, accessed July 12, 2025, https://docs.datarobot.com/en/docs/modeling/special-workflows/otv.html
79. How do you evaluate the accuracy of a time series model? - Milvus, accessed July 12, 2025, https://milvus.io/ai-quick-reference/how-do-you-evaluate-the-accuracy-of-a-time-series-model
80. cloudera/CML_AMP_Explainability_LIME_SHAP: Learn how to explain ML models using LIME and SHAP. - GitHub, accessed July 12, 2025, https://github.com/cloudera/CML_AMP_Explainability_LIME_SHAP
81. Anomaly Detection in Time Series - neptune.ai, accessed July 12, 2025, https://neptune.ai/blog/anomaly-detection-in-time-series
82. Anomaly Detection Techniques with Python - Codefinity, accessed July 12, 2025, https://codefinity.com/blog/Anomaly-Detection-Techniques-with-Python
83. (PDF) Anomaly Detection in Blockchain Using Machine Learning, accessed July 12, 2025, https://www.researchgate.net/publication/380312005_Anomaly_Detection_in_Blockchain_Using_Machine_Learning
84. Anomaly Detection Crypto: Detecting Anomalies ... - CoinAPI.io Blog, accessed July 12, 2025, https://www.coinapi.io/blog/how-axyon-ai-unlocks-the-power-of-market-data-api-to-run-crypto-anomaly-detection-solution
85. accessed December 31, 1969, https://huyenchip.com/2020/12/27/testing-in-production.html
86. accessed December 31, 1969,

https://www.kdnuggets.com/2022/09/ab-testing-machine-learning-models-production.html
87. accessed December 31, 1969, https://huyenchip.com/2022/01/02/machine-learning-systems-design-a-primer.html
88. accessed December 31, 1969, https://www.kdnuggets.com/2022/11/monitoring-machine-learning-models-production.html
89. Concept drift - — Frouros - Read the Docs, accessed July 12, 2025, https://frouros.readthedocs.io/en/v0.5.1/