My Journey into Machine Learning for Horse Racing: Skills, Lessons, and Capabilities

This document details my self-guided journey into building high-quality machine learning pipelines, from data ingestion to model deployment, in the specific domain of horse racing. Through deep experimentation, debugging, and model architecture design, I've developed working systems using both **XGBoost** and **Transformer-based neural networks** for structured predictive modeling.

What I Can Do

- Build **complete machine learning pipelines** for tabular and structured data
- Engineer **domain-aware features** with racing expertise
- Understand and prevent **data leakage** — particularly temporal and target leakage
- Use **race-relative modeling** instead of flat binary classification
- Build and extend **deep learning models** using Transformers
- Manage datasets, feature stores, and model evaluation metrics
- Apply **NLP** techniques to leverage domain text data (commentary, expert summaries)

⚙ What I've Built

XGBoost-Based Model (Binary Classification)

- Optimized for `log_loss`
- Used `GroupShuffleSplit` by `race_id` to preserve race integrity during cross-validation
- Avoided race leakage via careful group-wise splitting
- Simulated real-world betting conditions with **per-race softmax normalization**
- Used in historical inference across a full year of data (~13,000 bets)

Result Snapshots:

- **Without RPR (leakage-free):**
    - Accuracy: ~88%
    - F1 Score: ~0.008
    - Log Loss: ~0.36
    - ROI: -14.18%

    - Win Rate: 18.53%
    - Total Profit: -£1864.99 over 13,155 bets

- **With suspected leakage (pre-cleanup RPR):**

    - Unrealistically high prediction confidence
    - Flagged due to suspiciously low log-loss and high hit rate
    - Led to investigation and eventual resolution

---

2. Leakage Investigation and Resolution

- Detected that Racing Post Ratings (RPRs) might be updated post-race
- Investigated by scraping and comparing JSON files before and after race day
- Validated that historical data contained **updated/retrospective RPRs**, making it a leak
- Built tooling to:
    - Compare pre-race and post-race RPR values
    - Create a safe, pre-race-only feature vault
- Built markdown reports and audit trail to communicate this issue

---

3. Transformer-Based Architecture (Race-Relative Modeling)

- Designed a race-aware system where each race is modeled as a **set of runners**
- Input shape: `[batch_size, num_runners, feature_dim]`
- Used **embedding layers** for:
    - Trainer ID
    - Jockey ID
    - Course
- Combined with **numeric features** (e.g., RPR, OR, draw, age, TS)
- Planned use of **SetTransformers** or attention layers to:
    - Learn intra-race dynamics
    - Output softmax-normalized win probabilities for each runner
- Designed custom loss: **per-race softmax + log loss**
- Envisioned extensions for ROI, ranking, uncertainty

Feature Engineering & Data Insight

Extracted & Modeled Features:

- `or`, `rpr`, `draw`, `age`, `ts`, `lbs`, `last_run`
- `trainer_id`, `jockey_id`, `trainer_rtf`, `going`, `field_size`, `race_class`, `type`
- 14-day trainer and jockey form: win %, profit, run count
- Historical win/place data by:
    - Course

    - Distance
    - Going

Pipeline Infrastructure:

- Parsed and flattened nested JSON data for daily inference
- Built daily ingestion logic for creating clean, race-wise dataframes
- Extracted course-level profiles for embedding or enrichment
- Stored engineered features with flag-based activation for future use

---

Advanced Areas of Exploration

- **NLP Embedding** of Racing Post commentary and spotlights
- Plans for embedding sentiment or BERT-style summaries into runner vectors
- Course-level win bias and draw bias
- Wind surgery history and flags
- Composite engineered features (e.g., “fitness score”, “risk index”)

What I’ve Learned

- Leakage isn’t just about target columns — **features can silently leak** if updated post-outcome
- **Racing data is temporal, relational, and domain-rich** — not just a tabular problem
- **Modeling races as a unit**, not individual runners, dramatically improves realism
- Attention-based models can extract inter-runner signals **no tree-based model can see**
- Documentation, reproducibility, and modularity are as important as code quality

---

What I Can Contribute

- Design and implementation of **leak-free, race-aware ML systems**
- Domain-expert feature engineering pipelines
- Evaluation tools that **simulate real-world betting environments**
- Transformer architectures for structured multi-entity modeling
- Cross-discipline thinking: NLP, tabular, time-series, and sports analytics

---

Summary

I’ve built models that think like bettors.
I’ve debugged leaks like a researcher.
And I’ve architected pipelines like a production engineer.

All without formal credentials — but with working systems to show for it.