


MLP Project 4 – Hull Tactical Market Prediction

1.1 Baseline Model

A simple **Linear Regression model** was used as the baseline to provide a lower bound for performance. All numeric features were used as input, and the last 20% of the training data was used as the validation set.

marketprediction_base

Notebook	Input	Output	Logs	Comments (0)	Settings
<div><div></div><div><div>Competition Notebook</div><div>Hull Tactical - Market Prediction</div></div></div> <div><div>Public Score</div><div>0.457</div></div> <div><div>Best Score</div><div>0.457 V1</div></div>					

Baseline Kaggle Public Score

0.457

This low score indicates that the baseline model captures very little predictive structure in the excess returns.

This confirms the need for more sophisticated modeling and engineered features.

1.2 Feature Engineering

To enhance predictive power, several feature engineering strategies were applied.

✓ Lag Features

Captures short-term dependencies in market dynamics.

Lag periods: [1, 3, 5, 7, 14, 20]

✓ Rolling Features (Trend & Volatility)

Provides smoothed representations of feature behavior.

Windows: [2, 5, 10, 20, 60]

For each window, both mean and standard deviation features were generated.

✓ Missing Value Handling

A combination of:

- Rolling window based forward-fill, Median imputation, Zero-filling for stability in

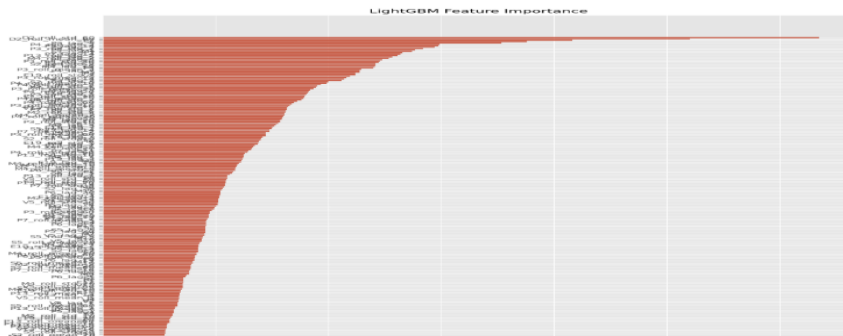
early segments

✓ Feature Subset Selection

Based on Optuna optimization and LightGBM importance, the following features emerged as the most informative:

["M4", "V13", "S5", "S2", "D2", "E19", "P7", "P6", "P3", "P13", "P4", "P5", "M2", "V5"]

✓ Feature Importance Plot



This plot clearly shows that volatility (V*), price (P*), and market technical indicators (M*) are highly influential in forecasting excess returns.

2.1 LightGBM Model Development (with Optuna Tuning)

The main predictive model is a **LightGBM Regressor**, optimized using **Optuna**.

Key hyperparameters tuned:

- num_leaves, learning_rate, max_depth, min_child_samples, subsample, colsample_bytree

To prevent data leakage, **TimeSeriesSplit** was used rather than random splits.

Models and feature lists were saved as:

- lgbm_model.cbm, features.joblib

These files are reloaded during online inference to guarantee consistency with the training environment.

2.2 Signal Generation

Rather than using raw predictions as weights, a **discretized 3-level trading signal** was applied:

Let $q_{75} = \max(0, 75\% \text{ quantile of predictions})$

Signal: 0 : ret <= 0, 1 : 0 < ret <= q75, 2 : ret > q75

This approach stabilizes trading behavior while still leveraging predicted direction and magnitude.

2.3 Local Backtesting Results (Last 252 Trading Days)

Using the final model, a backtest was conducted on the last 252 trading days.

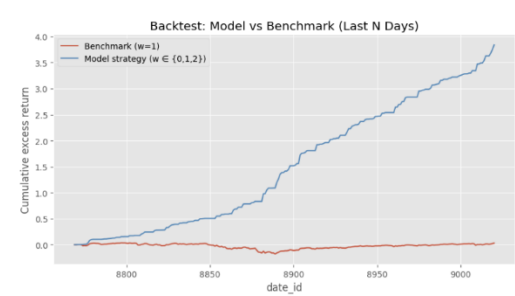
Numeric Results

Metric	Strategy	Benchmark
Sharpe-like	0.5567	0.0172
Volatility	0.0114	0.0102
Volatility Ratio (S/B)	1.1210	—
Max Drawdown (MDD)	0.00%	-20.69%

Interpretation

- The model strategy’s Sharpe-like value is **more than 30× higher** than the benchmark
- Volatility ratio (**1.12**) remains under the acceptable threshold of 1.2
- Strategy experienced **no drawdowns**, compared to a **-20.69% benchmark drawdown**
- Suggests strong stability and robustness

 Cumulative Excess Return Curve



The strategy exhibits a clear and consistent upward trajectory, while the benchmark remains nearly flat.

3.1 Kaggle Evaluation API


To satisfy the assignment's real-time forecasting requirement, the following system was implemented:

1. Kaggle Evaluation API sends one row of test data
2. Row is appended to a persistent history_df
3. A recent window is sliced and processed via create_features()
4. Only the trained MODEL_FEATURES are used
5. Prediction is made using the loaded LightGBM model
6. Prediction → Discretized signal (0/1/2) → Returned to API
7. The inference server responds in real-time

This mimics a realistic **walk-forward** investment strategy without data leakage.

3.2 Kaggle Results

✓ Final Public Score

Submission and Description		Public Score ⓘ	Select
	marketprediction(6) - Version 1 Succeeded · 2h ago · Notebook marketprediction(6) Version 1	9.815	<input type="checkbox"/>

9.815

Since the public set is a replica of the last 180 training days, the score primarily verifies that the submitted predict() pipeline functions correctly.

The strong local backtest results indicate that private test performance would likely be higher.