ISYE 6767: Project 2 Report

Data Analysis and Machine Learning for Stock Price Prediction

Arjo Bhattacharya
Georgia Institute of Technology

Fall 2025

ISYE 6767 – Design and Implementation of Computational Finance Models

1 Data Collection and Preprocessing (5%)

1.1 Data Source and Collection

Daily OHLCV data for the period **January 1, 2000** to **November 12, 2021** was obtained from **Yahoo Finance** using the `yfinance` Python package. The implementation followed the project specification: all data was programmatically downloaded inside the Python script, ensuring full reproducibility.

- **Small universe:** 10 tickers from `tickers.csv`: `FIX, TSLA, CNP, DLTR, WMS, HAS, HIBB, RHI, TGT, WBA`. Data for `HIBB` and `WBA` could not be reliably downloaded (delisting / data feed issues), so the effective small universe consists of **8 stocks**.
- **Large universe:** The larger list from the assignment (e.g., large-cap technology, financial, energy, consumer, and industrial stocks) was used to construct a **58-stock** universe. All tickers with sufficient data in the specified window were kept; others were automatically skipped.

All downloaded data frames were stored with standardized column names `[date, open, high, low, close, volume, adj_close]` and sorted in ascending date order.

1.2 Data Issues and Resolution

The data preprocessing logic is encapsulated in a dedicated `DataPreprocessor` class, which implements the following steps:

Issue 1: Variable Start Dates. Not all stocks have observations from 2000-01-01. For example, `TSLA` starts at IPO date (June 2010) and `WMS` starts later in 2014. Rather than imposing an artificial common start date, the models were trained on the longest available history for each stock. This is consistent with realistic practice and avoids losing useful data.

Issue 2: Missing Values. Missing values arise from holidays, partial trading days, or data glitches. The following procedure was used:

- Data sorted by `date`.

- Price columns `[open, high, low, close]` were forward-filled using `ffill`, never back-filled, to avoid using future information.
- `volume` was forward-filled, and any residual `NaN` volumes were set to zero.

Using only forward-fill ensures that the training and test sets do not suffer from look-ahead bias.

Issue 3: Outliers. Extreme outliers were detected and handled via a conservative $\mathbf{5\sigma}$ **z-score** filter:

- For each of `open, high, low, close`, compute sample mean and standard deviation.
- Values beyond $\mu \pm 5\sigma$ were labelled as potential glitches, replaced by `NaN`, and then forward/backward filled.

A 5σ threshold avoids removing legitimate large moves while still cleaning absurd spikes.

Issue 4: Data Integrity. To ensure internal consistency, the following corrections were enforced row by row:

$$\texttt{high} = \max(\texttt{open}, \texttt{high}, \texttt{low}, \texttt{close}), \quad \texttt{low} = \min(\texttt{open}, \texttt{high}, \texttt{low}, \texttt{close}).$$

This guarantees $\texttt{low} \leq \texttt{open}, \texttt{close} \leq \texttt{high}$ for all days.

1.3 Data Split and Normalization

After feature construction (Section 2), the final dataset for each stock was:

- **Chronologically split**: first **60%** of observations used for training, remaining **40%** for testing; no shuffling to respect time order.
- **Input scaling**: all numeric feature columns (excluding prices, returns, volume, raw date) were standardized using `StandardScaler`. The scaler was fit *only on the training set* and then applied to the test set to avoid data leakage.
- Rows with any missing feature or target values were dropped before splitting.

This pipeline ensures appropriate handling of missing values, outliers, and scaling, fully aligned with the data-preprocessing expectations in the project description. :contentReference[oaicite:0]index=0

2 Feature Engineering (5%)

Feature construction is encapsulated in the `FeatureEngineer` class. Starting from a clean OHLCV panel, the following categories of features were created.

2.1 Feature Categories

The target variable is the **direction of next-day return**:

$$\texttt{future\_return}_t = \frac{\texttt{close}_{t+1} - \texttt{close}_t}{\texttt{close}_t}, \quad \texttt{target}_t = \mathbb{1}\{\texttt{future\_return}_t > 0\}.$$

The feature set is organized into six main groups:

1. **Returns and Trend Measures.**
   - Simple and log returns: `returns`, `log_returns`.
   - **Moving averages**: SMA and EMA with windows $\{5, 10, 20, 50, 100, 200\}$.
   - Relative distance to SMA: `close_to_sma_w` = `close`/`sma_w` − 1.
   - Crossover indicators: $\mathbb{1}\{\texttt{sma\_5} > \texttt{sma\_20}\}$, $\mathbb{1}\{\texttt{sma\_20} > \texttt{sma\_50}\}$.

2. **Momentum Indicators.** Using the `ta` library:
   - RSI(7) and RSI(14).
   - MACD line, MACD signal, MACD histogram.
   - Stochastic oscillator (`stoch_k`, `stoch_d`).
   - Rate-of-change (ROC) over 5 and 10 days.

3. **Volatility Indicators.**
   - Bollinger Bands (20-day, 2σ): upper, lower, mid, band width, and %B.
   - Average True Range (ATR-14).
   - 20-day rolling volatility of returns annualized by $\sqrt{252}$.

4. **Volume-Based Indicators.**
   - 20-day volume SMA and `volume_ratio` = `volume`/`volume_sma_20`.
   - On-Balance Volume (OBV).
   - Volume-Price Trend (VPT).

5. **Price Pattern Features.**
   - Candlestick body and body percentage: `body` = `close` − `open`, `body_pct` = `body`/`open`.
   - Upper and lower shadows.
   - Daily range: (`high` − `low`)/`low`.
   - Gap between current open and previous close.

6. **Lagged Features.** To capture autocorrelation and short-term memory:
   - Lags of `returns`, `volume_ratio`, and `rsi_14` at lags 1, 2, 3, and 5 days.

Together, these features cover trend, momentum, volatility, volume confirmation, price patterns, and short-term history, and serve as a rich input space for binary direction prediction.

2.2 Feature Effectiveness

Empirically, the feature set produced a number of models that exceeded the benchmark accuracy and precision thresholds (Accuracy ≥ 0.5014, Precision ≥ 0.5141). :contentReference[oaicite:1]index=1 Key observations:

- **Moving averages** and price-to-SMA ratios provided robust trend-following signals and helped Logistic Regression achieve above-benchmark accuracy on stocks like CNP, FIX, and TSLA.
- **RSI and ROC** were particularly informative around local reversals and overbought/oversold regimes.
- **Volume-based features** improved precision by filtering out unconfirmed breakouts.
- **Lagged features** (especially 1-day and 5-day lags of returns and RSI) captured short-term continuation / mean-reversion patterns and contributed significantly to model performance.

All features were standardized (zero mean, unit variance) on the training set before model fitting, which improved the numerical stability and convergence of Logistic Regression and reduced feature-scale bias for Random Forest splitting.

3 Model Selection and Hyperparameters (10%)

3.1 Choice of Models

The project requires two classification models with binary output (up/down next day). I selected:

Random Forest Classifier.

- Naturally handles non-linear decision boundaries and high-dimensional feature spaces.
- Captures complex interactions among technical indicators.
- Robust to outliers and noise due to bagging and feature randomness.
- Provides feature importance measures for interpretability.

Logistic Regression.

- Strong, well-understood linear baseline for classification.
- Output probabilities are directly interpretable as estimated rise probabilities.
- L1/L2 regularization allows for embedded feature selection and shrinkage.
- Efficient to train and stable under standardized features.

This pair gives a contrast between a flexible non-linear ensemble and an interpretable linear model.

3.2 Hyperparameter Tuning Strategy

Hyperparameter tuning was performed using **GridSearchCV** with **TimeSeriesSplit** (3 folds) to respect temporal order:

Random Forest Grid.

- `n_estimators`: [100, 200]
- `max_depth`: [5, 10, 20, None]
- `min_samples_split`: [2, 5, 10]
- `min_samples_leaf`: [1, 2, 4]

Logistic Regression Grid.

- `C`: [0.001, 0.01, 0.1, 1, 10]
- `penalty`: [`'l1'`, `'l2'`]
- `solver`: [`'saga'`], which supports both L1 and L2.
- `max_iter`: 1000

For each stock in the small universe, the grid search selected the best hyperparameters based on validation accuracy.

For the larger universe, a fixed, reasonable configuration was used (Random Forest with 100 trees and depth 10; Logistic Regression with $C = 1.0$) in order to control computational cost while still capturing typical model behavior.

3.3 Initial Expectations

Under the weak-form Efficient Market Hypothesis, pure technical-indicator-based models should not produce strong predictive power. Therefore my realistic prior expectations were:

- Out-of-sample accuracies in the **51–55%** range at best, modestly above random guessing.
- Slightly higher **precision** than accuracy if the models lean towards predicting the majority class.
- Logistic Regression to provide a strong baseline, with Random Forest capturing additional non-linear structure for particularly volatile names (e.g., TSLA).
- Substantial friction from transaction costs, spreads, and drawdowns, making it difficult to convert small predictive advantages into reliable trading profits.

4 Model Performance Results

4.1 Small Universe: Prediction Metrics

The small-universe results are summarized in Table 1. The benchmark thresholds from the assignment are accuracy 0.5014 and precision 0.5141. :contentReference[oaicite:2]index=2

Table 1: Model Performance on Small Universe (8 Tickers, 16 Models)

Ticker	Model	Accuracy	Precision	ROC-AUC	Benchmark
FIX	Random Forest	0.4804	0.4222	0.5187	
FIX	Logistic Regression	0.5186	0.5228	0.5111	✓
TSLA	Random Forest	0.5103	0.5266	0.5011	✓
TSLA	Logistic Regression	0.5084	0.5223	0.5340	✓
CNP	Random Forest	0.4715	0.5069	0.4992	
CNP	Logistic Regression	0.5304	0.5304	0.5000	✓
DLTR	Random Forest	0.4899	0.6379	0.5015	
DLTR	Logistic Regression	0.4908	0.5074	0.4732	
WMS	Random Forest	0.4863	0.6395	0.5383	
WMS	Logistic Regression	0.4497	0.0000	0.5000	
HAS	Random Forest	0.4649	0.4366	0.5017	
HAS	Logistic Regression	0.4781	0.5438	0.5083	
RHI	Random Forest	0.4771	0.5750	0.5157	
RHI	Logistic Regression	0.5007	0.5630	0.5167	
TGT	Random Forest	0.4899	0.5197	0.4980	
TGT	Logistic Regression	0.4932	0.5402	0.5059	
Benchmark Threshold		**0.5014**	**0.5141**	—	—
Models Meeting Both		—	—	—	**4/16 (25%)**

Benchmark Analysis.

- **4 out of 16** models (**25%**) meet both benchmark criteria:
  1. CNP – Logistic Regression
  2. FIX – Logistic Regression
  3. TSLA – Logistic Regression
  4. TSLA – Random Forest
- The best accuracy is achieved by **CNP – Logistic Regression** (53.04%).
- **Logistic Regression dominates**: 3 of the 4 qualifying models use LR.

Aggregate Statistics. Using the 16 models in the small universe:

Table 2: Aggregate Prediction Statistics (Small Universe)

Metric	Random Forest	Logistic Regression	Overall
Mean Accuracy	0.4838	0.4962	0.4900
Mean Precision	0.5331	0.4662	0.4997
Mean ROC-AUC	0.5093	0.5062	0.5077
Models Meeting Both	1/8 (12.5%)	3/8 (37.5%)	4/16 (25%)

Key takeaway: **Random Forest** delivers higher average precision, whereas **Logistic Regression** delivers higher average accuracy and more models that satisfy both benchmarks.

4.2 Backtesting Results for the Small Universe

The ML predictions were turned into trading signals and backtested with BackTrader using the following simple strategy:

- If predicted signal = 1 (up), invest 95% of available cash in the stock.
- If predicted signal = 0, close any open position.
- Initial capital: $100,000, with 0.1% transaction cost per trade.

Table 3 summarizes the backtest metrics for all 16 small-universe strategies.

Table 3: Backtest Results (Small Universe, Initial Capital: $100,000)

Ticker	Model	Return (%)	Sharpe	Max DD (%)	Win Rate (%)
FIX	Random Forest	-16.73	-0.821	26.05	41.03
FIX	Logistic Regression	420.58	0.361	47.37	65.93
TSLA	Random Forest	91.25	0.254	50.45	63.01
TSLA	Logistic Regression	229.69	0.318	60.64	65.97
CNP	Random Forest	-14.21	-0.396	28.89	46.19
CNP	Logistic Regression	58.19	0.061	58.02	0.00
DLTR	Random Forest	12.92	-0.650	5.78	54.29
DLTR	Logistic Regression	-40.58	-0.399	54.67	54.05
WMS	Random Forest	37.11	0.288	31.23	66.10
WMS	Logistic Regression	0.00	0.000	0.00	0.00
HAS	Random Forest	-33.74	-0.726	52.15	39.39
HAS	Logistic Regression	-26.22	-0.302	42.21	50.00
RHI	Random Forest	13.15	-0.430	25.88	59.68
RHI	Logistic Regression	-47.64	-0.674	61.14	54.95
TGT	Random Forest	-16.19	-0.374	36.72	59.14
TGT	Logistic Regression	5.56	-0.210	31.68	54.51

Top Performers.

1. **FIX – Logistic Regression**: 420.6% cumulative return, Sharpe 0.36, max drawdown 47.4%, win rate 65.9%.
2. **TSLA – Logistic Regression**: 229.7% return, Sharpe 0.32, max drawdown 60.6%, win rate 66.0%.
3. **TSLA – Random Forest**: 91.3% return, Sharpe 0.25, max drawdown 50.4%, win rate 63.0%.

Notable Pathologies.

- **CNP – Logistic Regression** has 58.2% total return but a 0% win rate with only one trade, illustrating that profits can be dominated by a single episode.
- **WMS – Logistic Regression** effectively never trades (0 trades, 0% return), indicating a degenerate decision boundary.

Table 4: Trading Performance Metrics Summary (All Small-Universe Strategies)

Metric	Best	Worst	Mean	Median
Total Return (%)	420.58	-47.64	42.07	2.78
Sharpe Ratio	0.36	-0.82	-0.23	-0.34
Max Drawdown (%)	0.00	61.14	38.30	39.47
Total Trades	296	0	141.06	140
Win Rate (%)	66.10	0.00	48.39	54.40

Aggregate Backtest Statistics (16 Strategies). Despite some very high individual returns, median return, Sharpe, and drawdowns highlight that risk-adjusted performance is weak and highly stock-dependent.

4.3 Large Universe Results

For the large universe of **58 stocks**, both models were trained with fixed hyperparameters. Key statistics:

- Average Random Forest accuracy: $\approx 49.3\%$.
- Average Logistic Regression accuracy: $\approx 50.3\%$.
- **11 stocks** (18.97%) achieved best accuracy $> 52\%$.
- **1 stock** (META) achieved best accuracy $> 54\%$.

Table 5 lists the **top 10 stocks** ranked by out-of-sample `Best_Accuracy`.

Table 5: Top 10 Large-Universe Stocks by Out-of-Sample Accuracy

Ticker	Best Model	Best Accuracy	RF Accuracy	LR Accuracy
META	Logistic Regression	0.5422	0.5137	0.5422
PYPL	Logistic Regression	0.5391	0.5249	0.5391
UNH	Random Forest	0.5285	0.5285	0.4913
WFC	Logistic Regression	0.5271	0.5007	0.5271
MMM	Logistic Regression	0.5252	0.4875	0.5252
MA	Logistic Regression	0.5250	0.4959	0.5250
T	Random Forest	0.5243	0.5243	0.5087
YUM	Logistic Regression	0.5233	0.4847	0.5233
WMS	Logistic Regression	0.5229	0.4756	0.5229
WMT	Logistic Regression	0.5219	0.4899	0.5219

These 10 stocks are used in Section 5 as the basis for trading-report generation as required by the project.

5 Trading Reports Summary (5%)

QuantStats was used to generate detailed HTML trading reports from the backtested daily returns series. For each strategy, the report includes:

- **Profit and Loss (P&L) curve** and cumulative returns.
- **Sharpe ratio**, volatility, and other risk-adjusted metrics.
- **Maximum drawdown** and drawdown-over-time plots.
- **Win/loss ratio**, hit rate, and trade statistics (number of trades, average win/loss).

The code produces:

- QuantStats reports for all 16 small-universe strategies.
- QuantStats reports for each of the **10 best** large-universe stocks in Table 5, using their best-performing model.

In the submission package, I include **10 representative HTML trading reports**, one for each of the top-10 large-universe stocks. These reports satisfy the requirement of including P&L curves, Sharpe ratios, maximum drawdowns, and win/loss ratios for 10 strategies, and they provide a clear visual assessment of the strategy behavior over time.

6 Conclusion and Model Comparison (10%)

6.1 Effectiveness of the Models

Overall, both models exhibited **limited predictive power**:

- Only **4 out of 16** small-universe stock-model combinations (25%) met both benchmark thresholds of accuracy and precision.
- The overall mean accuracy across all small-universe models is approximately **49.0%**, below the 50% random baseline.
- ROC-AUC values hover around 0.51, indicating near-random separation between up and down days.
- While some strategies (e.g., FIX–LR, TSLA–LR) produced high historical returns, the distribution of Sharpe ratios is skewed negative, with a mean Sharpe of about -0.23 and median around -0.34.
- Maximum drawdowns as high as 60% are not acceptable in a professional trading context.

The models do capture *some* limited structure in the data (especially for specific names like TSLA, FIX, META, PYPL), but the average edge is very small and largely overwhelmed by risk and variance.

6.2 Random Forest vs Logistic Regression

Table 6: Qualitative Comparison of Random Forest and Logistic Regression

Aspect	**Random Forest**	**Logistic Regression**
Average Performance	Higher mean precision (53.3%), lower mean accuracy (48.4%).	Higher mean accuracy (49.6%), lower mean precision (46.6%).
Benchmark Success	1 out of 8 models (12.5%) met both accuracy and precision thresholds.	3 out of 8 models (37.5%) met both thresholds.
Advantages	• Captures non-linear patterns. • Handles feature interactions automatically. • Robust to noisy features. • Can provide feature importance rankings.	• Interpretable coefficients. • Well-calibrated probability outputs. • Computationally efficient. • Works well with standardized features.
Disadvantages	• Computationally heavier per stock. • Predictions are less interpretable. • Only one small-universe RF model met both benchmarks.	• Assumes linear decision boundary. • May underfit complex time-series dynamics. • One pathological stock (WMS) with 0% precision / no trades.
Best Use Case	Volatile, non-linear names such as TSLA and some large-universe stocks (e.g., UNH, T).	Trend-following or smoother names like FIX, CNP, META, PYPL.

Overall recommendation: Within this project's setup, **Logistic Regression** is preferred as a baseline model due to its higher mean accuracy and greater fraction of benchmark-satisfying models, despite lower average precision. Random Forest remains valuable as a complementary non-linear model, particularly for certain tickers, but it did not consistently outperform Logistic Regression in this study.

7 Additional Considerations (10%)

(a) Accuracy, Tradability, and Realism

Did the models achieve high accuracy? No. The best out-of-sample accuracy observed (CNP–LR in the small universe, META–LR in the large universe) is in the 53–54% range, and the mean accuracy is close to 49%. This is only marginally above random guessing and does not constitute a robust statistical edge once transaction costs and slippage are included.

Can we start trading on these models? No, not in their current form. The historical backtests show:

- High dispersion in returns (from 420% to -48%).
- Predominantly negative Sharpe ratios across strategies.
- Extreme drawdowns (up to 60%).
- Some strategies barely trade at all, indicating unstable decision rules.

Any small predictive advantage is likely wiped out by:

- Bid-ask spread and slippage.
- Commissions and fees.
- Execution delays and market impact.

Steps to make the analysis more realistic. To move toward a realistic, tradable strategy, the following improvements are needed:

- **Richer features**: Incorporate fundamental data, macro indicators, sentiment data (news, social media), and cross-asset signals rather than relying solely on technical indicators.
- **Cost modeling**: Explicitly model variable spreads, slippage, and fees; apply these in backtests.
- **Risk management**: Introduce stop-losses, take-profit rules, portfolio-level risk limits, and position sizing based on volatility or risk budgets.
- **Portfolio construction**: Trade baskets of stocks with diversification constraints instead of single-name strategies.
- **Regime awareness**: Condition predictions on volatility regimes (e.g., via VIX) and market states (bull/bear, high/low liquidity).
- **Walk-forward retraining**: Periodically retrain models (e.g., monthly) using expanding or rolling windows, and evaluate in a fully walk-forward manner.

(b) Overfitting and Mitigation

Is the model possibly overfitted? The evidence suggests that **severe overfitting is unlikely**; if anything, the models may be underfitting or facing a truly weak signal:

- Mean test accuracy is $\approx 49\%$, not excessively high.
- Many strategies produce negative Sharpe ratios and large drawdowns.
- ROC-AUC values are close to 0.50, indicating only weak separation ability.
- Performance is mediocre across many different stocks, not just a cherry-picked subset.

These are more consistent with models struggling to extract signal from noisy data rather than overfitting and then collapsing out-of-sample.

Overfitting mitigation steps taken. The following explicit steps were implemented to mitigate overfitting:

1. **Time-respecting splits:** All train/test splits and cross-validation folds were chronological (TimeSeriesSplit), with training strictly in the past and validation/testing in the future.
2. **Regularization:** Logistic Regression used L1/L2 penalties with a grid over `C` values, while Random Forest tuned max depth, minimum samples per split, and leaf size.
3. **Standardization:** Features were standardized using parameters from the training set only.
4. **Large test fraction:** 40% of the available history per stock was held out as a final test set.
5. **Feature curation:** Non-stationary cumulative features (e.g., raw OBV/VPT) were not used as direct predictive targets; instead, the focus was on incremental/ratio-based indicators.

Additional mitigation strategies (future work). To further guard against overfitting and better understand model behavior:

- **Feature selection**: Use LASSO, mutual information, or permutation importance to prune redundant/irrelevant features.
- **Ensembles and stacking**: Combine Logistic Regression and Random Forest predictions in an ensemble.
- **Learning curves**: Plot training vs. validation performance as a function of sample size to diagnose bias–variance trade-offs.
- **Out-of-time validation**: Test models on post-2021 data (e.g., 2022–2024) as a strictly out-of-sample period.

Conclusion on Overfitting. Given the low test accuracies, near-random ROC-AUC, and negative risk-adjusted returns, the model family here is **not** obviously overfitted in a classical sense. Instead, it appears that **technical indicators alone do not provide a strong enough signal** to support robust, profitable trading strategies in an efficient equity market.

8 Final Conclusion

This project implemented a complete pipeline for data-driven trading strategy design:

- Robust data collection, cleaning, and normalization from Yahoo Finance.
- Rich feature engineering using trend, momentum, volatility, volume, pattern, and lagged indicators.
- Two ML models (Random Forest and Logistic Regression) tuned via time-series cross-validation.
- BackTrader integration for turning signals into trades, with QuantStats reports for detailed performance analysis.
- Extension from a small universe to a larger 58-stock universe, identifying the 10 most predictable names.

Key findings:

- Only **25%** of small-universe models met both the benchmark accuracy and precision thresholds.
- Mean small-universe accuracy is $\approx 49.0\%$, below random baseline.
- Logistic Regression outperformed Random Forest in terms of benchmark success (3 vs. 1 models).
- Some strategies produced attractive historical returns (e.g., FIX–LR), but with high drawdowns and weak Sharpe ratios.
- Large-universe results confirm that only a small fraction of stocks exhibit modestly better-than-random predictability.

Practical implication: In their current form, these models are **not suitable for live trading**. They do, however, serve as an excellent educational example of a full ML-in-finance workflow, highlighting:

- The importance of careful data preprocessing and leakage avoidance.
- The gap between statistical significance and economic significance.
- The substantial additional work required (risk management, cost modeling, regime detection, alternative data) to move from a predictive model to a robust trading strategy.

Future work should focus on combining technical indicators with fundamentals, macro data, sentiment, and more sophisticated model architectures, while adhering to strict out-of-sample and out-of-time evaluation standards.