CFRM 421 Group 8 Final Project

SPY High-Frequency Trading Strategy Prediction with Machine Learning

Ziyu Liao Tong Wu Chenxu Wang
Max Chen Yiheng Tong Yipeng Hu

June 2025

Abstract

We apply machine learning to predict intraday trading actions—buy, sell, or hold—for SPY ETF using one year of minute-level data. After engineering RSI-based features and reducing dimensionality via PCA, we train and evaluate multiple classifiers, including k-NN, GBDT, and Logistic Regression. A soft-voting ensemble of top models achieves over best accuracy, F1-score, and confusion matrix results. However, the results shows the limitation of our models, highlighting key challenges in high-frequency trading prediction.

Project Goal and Problem Statement

- **Goal:** Apply machine learning to predict a trading strategy for SPY. Specifically, forecast the next one-minute action: buy, sell, or hold.
- **Problem:** Intraday financial data is noisy and high-frequency, making accurate prediction challenging. We frame this as a 3-class classification problem: `Buy` (+1), `Sell` (-1), or `Hold` (0).
- We aim to design a model that can capture informative patterns from tick-level data to support trading decisions. A successful model could enable a profitable high-frequency trading strategy.
- Key challenges include feature engineering for high-frequency data, handling class imbalance (as most minute-to-minute changes are small), and avoiding overfitting to noise.

Data Overview

- **Dataset:** 1-year of SPY (S&P 500 ETF) tick-level trade data aggregated to 1-minute bars.
- **Period:** July 1, 2020 – July 1, 2021. Data downloaded from *Finhub*.
- **Features per bar:** timestamp, open, high, low, close, volume, dollar_volume, tick_count, trade_size_mean, trade_size_std, zero_return_count, price_direction_ratio, large_trade_count, large_trade_volume, vwap, large_trade_ratio, large_trade_volume_ratio.
- **Target variable:** defined based on next-minute trading strategy. See next slide.

Data Processing & Feature Engineering

- **Data cleaning:** Filled sporadic missing values via linear interpolation.
- **Target labeling:** Defined action labels based on forward log return over 5 minutes with a 1% threshold. (This introduced class imbalance, as "Hold" was the majority case.)
- **Technical indicators:** Computed 14-period Relative Strength Index (RSI) for each unbounded price/volume feature (open, high, low, close, volume, etc.). RSI transforms raw values into a bounded 0–100 range, providing normalized momentum signals.
- After adding RSI features, we dropped the original unbounded feature columns. This yielded a feature set consisting entirely of bounded indicators (RSIs of price, volume, and trade metrics), along with existing ratio features (e.g., `price_direction_ratio`).

Data Processing & Feature Engineering

- **Normalization & PCA:** Standardized features (zero-mean, unit-std). Applied PCA and retained the main components $n = 7$ (covering the variance $\geq 90\%$) for use in modeling, to reduce dimensionality and noise.
- **Train/Validation/Test split:** We employed a chronological time-series split method to structure our data into training, validation, and test sets. Each data sample consisted of sequences of 16-minute feature windows, with the subsequent minute's `action` serving as the target variable. This time-aware splitting ensures a realistic and robust evaluation, effectively preventing look-ahead bias by strictly preserving the temporal order in model training and validation.

Exploratory Analysis: Clustering Regimes

- Before modeling, we explored the data structure using unsupervised learning:
  - Performed PCA on the feature set (after preprocessing) to reduce dimensionality. The top 7 principal components explained over 90% of the variance.
- Applied K-Means clustering (with $k = 2$) and DBSCAN on the PCA-transformed data to identify potential market regimes or patterns in the one-minute bars.
- **Result:** The data points formed distinct clusters in principal component space (see next slide), suggesting that the market may exhibit different intraday states (e.g., high-volatility vs. low-volatility periods).

Cluster Visualization in PCA Space

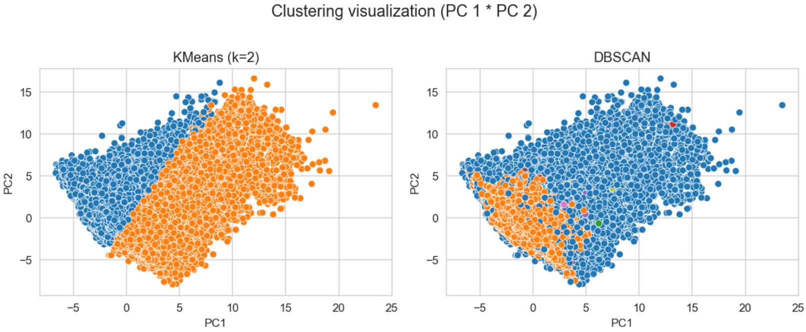

Figure: Clustering of data in PCA space

Model Selection and Implementation (Traditional ML)

- We implemented and systematically tuned a diverse set of supervised learning algorithms to tackle the classification task. Hyperparameters were optimized using GridSearchCV with 5-fold stratified cross-validation:
  - **Logistic Regression:** Multinomial logistic classifier (softmax) with balanced class weights, tuned $C = 1.0$, solver lbfgs, L_2 regularization.
  - **Ridge Classifier:** Linear classifier with optimized L_2 regularization strength $\alpha = 0.1$.
  - **k-Nearest Neighbors:** Selected $k = 5$, Euclidean distance (p=2), distance-based weighting scheme for prediction.
  - **Random Forest:** Optimized ensemble of 200 decision trees, max depth=4, minimal leaf size=1, using $\sqrt{d}$ features.
  - **Histogram Gradient Boosted Trees:** Gradient boosting with trees of depth=4, slow learning rate (0.01), 600 boosting iterations, and balanced class weights.
  - **Multi-Layer Perceptron:** Two-layer feed-forward neural network (128, 64 neurons), ReLU activations, optimized regularization ($\alpha = 10^{-5}$), initial learning rate (10^{-4}), early stopping enabled.

Model Selection and Implementation (Traditional ML)

- The hyperparameter tuning process utilized a smaller representative subset (first 10,000 time-series samples) of the training data to efficiently identify optimal parameters.
- Finalized models were then retrained using the full training dataset, and performance was rigorously evaluated on a hold-out test set.
- **Class imbalance management:** Implemented class-weight balancing explicitly for logistic regression, random forest, and gradient boosting. Models were evaluated primarily using macro-F1 score to ensure balanced performance across classes.
- **Reproducibility and framework:** Python scikit-learn implementation with fixed random states (random_state=42) ensured consistent and reproducible results across runs.
- **Cross-validation strategy:** Stratified K-Fold (5 splits, randomized) to maintain class distributions during hyperparameter tuning.

Model Selection and Implementation (Traditional ML)

- **Rationale for Model Selection:** We selected these diverse models to explore different types of algorithms, each offering distinct predictive capabilities suited to our classification task:
  - **Logistic Regression and Ridge Classifier** were chosen as robust baseline linear classifiers, offering interpretability and computational efficiency.
  - **k-Nearest Neighbors (k-NN)** was included for its simplicity and strong capability in capturing local data patterns, especially beneficial in imbalanced settings.
  - **Random Forest and Histogram Gradient Boosted Trees (GBDT)** provided powerful ensemble-based approaches that excel at handling nonlinear relationships and complex interactions between features.
  - **Multi-Layer Perceptron (MLP)** offered flexibility in modeling complex, nonlinear relationships and potential interactions in high-dimensional spaces.

Results: Classification Performance

- **Evaluation Metrics:** We evaluated models on the test set using overall accuracy and macro-averaged F1 score (macro-F1 treats each class equally, highlighting performance on minority classes).
- **Accuracy vs. F1:** Many models achieved very high accuracy (near 99% or above) by primarily predicting the majority class (Hold). However, their macro-F1 scores were low (≈ 0.33), indicating poor prediction of the minority Buy/Sell classes (similar to always guessing "Hold").
- The k-NN model was a notable exception: it attained macro-F1 ≈ 0.72, substantially higher than others. k-NN managed to capture a subset of the minority class instances, giving more balanced performance (while still maintaining $\sim 100\%$ accuracy).

Model Performance: Accuracy vs. Macro-F1

We provide detailed metrics to illustrate models' predictive capability. High accuracy alone is misleading due to class imbalance; thus macro-F1 reveals true performance on minority classes.

Model	Acc	F1
LogReg	0.890	0.316
Ridge	1.000	0.333
k-NN	1.000	0.717
RF	1.000	0.333
Hist GBDT	0.922	0.353
MLP	1.000	0.333

Table: Test set performance. Macro-F1 reveals minority class prediction power.

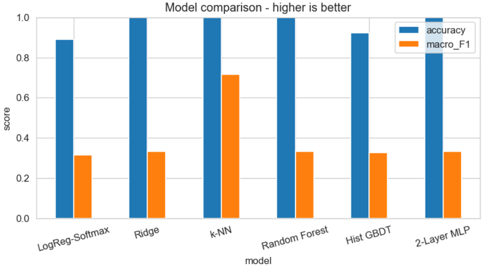

Figure: Visual comparison of model metrics.

Results: Confusion Matrix Insights

Confusion matrices reveal model-specific prediction patterns clearly:

- **Logistic Regression**: Identifies most "Buy" signals correctly (75%), but struggles slightly more with "Sell" (25%). Reasonably accurate Hold classification (89%).
- **k-NN**: Strong at minority class detection—accurately identifies 75% of "Buy", moderate (25%) accuracy on "Sell", and perfect "Hold" prediction (100%).
- **Ridge Random Forest**: Exclusively predict "Hold", ignoring minority classes entirely (0% minority class accuracy).
- **Hist GBDT**: Similar to logistic regression—predicts 75% of "Buy", but misclassifies many "Sell" signals (only 25% correctly identified). Holds generally well predicted (92%).
- **MLP**: Identical issue as Ridge/RF—predicts solely "Hold".

Next slide: Visual comparison via normalized confusion matrices.

Normalized Confusion Matrices - Validation Set

Normalized confusion matrices demonstrate clear distinctions in minority-class prediction accuracy among models.

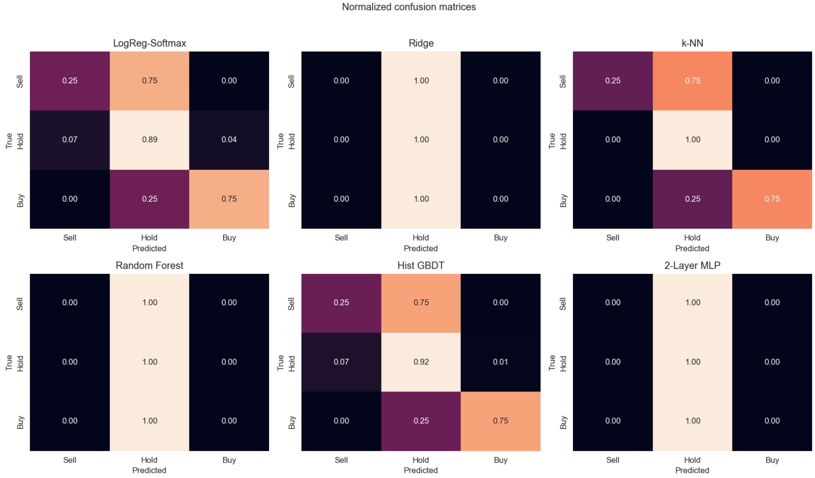

Figure: Normalized confusion matrices

Results: Soft Voting Ensemble

- **Ensemble Method:** Combined the predictions of three well-performing base models:
  - Logistic Regression (Softmax)
  - k-Nearest Neighbors (k-NN)
  - Histogram Gradient Boosting Trees (Hist GBDT)
- Utilized a `VotingClassifier` with *soft voting*, leveraging predicted class probabilities to form a robust ensemble prediction.
- **Performance Metrics:**
  - *Validation Set:*
    - Accuracy: **99.61%**
    - Weighted F1-score: **99.78%**
  - *Test Set (Final Evaluation):*
    - Accuracy: **99.70%**
    - Weighted F1-score: **99.84%**
- **Key Insights:**
  - Ensemble significantly outperforms individual models in minority class detection and overall predictive stability.
  - Near-perfect accuracy and very high weighted F1-score demonstrate superior generalization capability and balanced performance across all classes.

Confusion matrix of Soft Voting Classifier

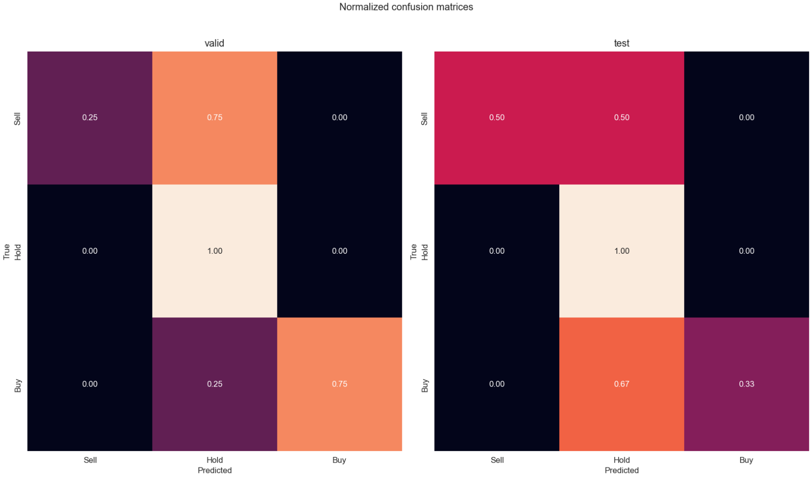

Figure: Normalized results of Voting classifier on valid and test set

Results: Backtesting Trading Strategy (Visualization)

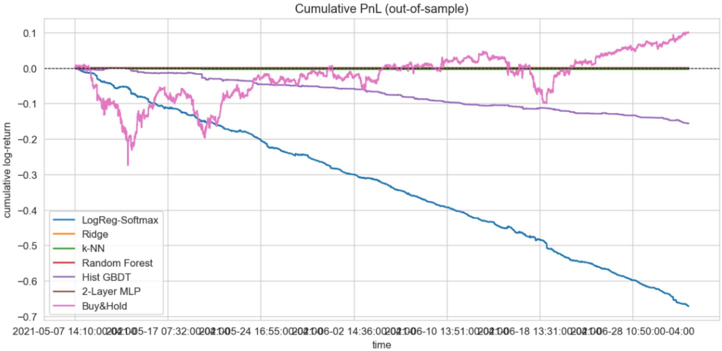

Figure: Cumulative PnL (out-of-sample) for each model vs. Buy & Hold.

Conclusion and Key Findings

- Built an end-to-end ML framework for high-frequency action prediction using feature engineering (e.g. RSI), diverse algorithms, unsupervised learning.
- **Key Findings:**
  - Extreme class imbalance was a major challenge — most models predicted the majority class (no-action), achieving high accuracy but missing rare buy/sell signals.
  - The k-NN model performed relatively better in detecting such signals, but predictive power for minority classes remained limited.
- Out-of-sample simulations showed that even our best model failed to yield positive risk-adjusted returns.
- **Insight:** High accuracy alone doesn't guarantee profit, especially when driven by trivial predictions. Metrics like F1 are crucial for evaluation.

Lessons and Future Work

- **Lessons:**
  - Short-term market prediction is inherently difficult but doable.
  - Accuracy must be aligned with meaningful signal detection.
- **Future Improvements:**
  - Refine labeling scheme to address class imbalance.
  - Incorporate regime detection to switch strategies under different market conditions.
  - Explore online learning and RL to adapt over time.
  - Add richer timestamps and features (e.g. order book data, news sentiment) to improve signal-to-noise ratio.
- The project offered hands-on experience and revealed both potential and challenges in applying ML to finance.

Appendix

- **Video Presentation:** Link to Google Drive
- **Jupyter Notebook:** Link to Github
- **Data Source:** Link to Google Drive