MGMT 638

Session 9

Kerry Back

Fall 2025

Agenda

1. data4 files, scripts, and outputs
2. Gu-Kelly-Xiu, 2020
3. Getting data (`create_data4.py`)
4. Training and predicting in rolling windows (`train_predict_data4.py`)
5. Output current model and predictions (`data4_current.xlsx`, `data4_model.pkl`)
6. Analyze historical portfolio returns (`analyze_portfolios.ipynb`)
7. Try to interpret model (`analyze_model_features.ipynb`)

Asset Pricing and Machine Learning

- PDF: *Asset Pricing and Machine Learning*, Gu, Kelly, and Xiu, 2020
- Video: *Asset Pricing and Machine Learning*, Gu, Kelly, and Xiu, 2020
- NotebookLM

data4.parquet: Overview

- Monthly panel dataset for predicting stock returns
- 591,892 stock-month observations (after dropping missing values)
- 4,851 unique tickers (including delisted)
- Date range: February 2011 to November 2025
- Combines price data, fundamentals, and classifications
- Created by `create_data4.py` (template for similar datasets)

data4.parquet: Variables (1/2)

Identifiers and Returns

- `ticker`, `month` – stock identifier and time period
- `return` – monthly return (decimal)
- `momentum` – 12-month return, skipping most recent month
- `lagged_return` – prior month's return

Price Data (end-of-prior-month)

- `close` – closing price
- `marketcap` – market capitalization (millions USD)
- `pb` – price-to-book ratio

data4.parquet: Variables (2/2)

Fundamentals (from 10-K filings)

- `asset_growth` – 1-year percent change in total assets
- `roe` – return on equity
- `gp_to_assets` – gross profit / total assets
- `grossmargin`, `assetturnover`, `leverage`

Classifications

- `sector`, `industry` – company classification
- `size` – Mega/Large/Mid/Small/Micro/Nano-Cap

data4.parquet: Key Methodology

Avoiding Look-Ahead Bias

- Price data (`close`, `marketcap`, `pb`) lagged by 1 month
- Fundamentals available in first full month *after* filing date
- Forward-filled until next filing

Filters Applied

- All tickers (including delisted) to avoid look-ahead bias
- Penny stock filter: `close` $\geq$ \$5.00

Size Categories

- Based on percentile cutoffs within each month
- Consistent distribution: Mega 1.5%, Large 20%, Mid 27%, Small 33%, Micro 15%, Nano 3%

create_data4.py: Overview

- Python script that creates `data4.parquet` from Rice Data Portal
- Queries SEP, DAILY, SF1, and TICKERS tables via API
- Implements proper time-series methodology to avoid look-ahead bias
- Designed as a template for creating similar ML-ready datasets
- Well-documented with 580 lines including extensive comments
- Outputs raw data only (percentile ranking done in training script)

create_data4.py: Configuration

Key parameters at top of script:

- `START_YEAR = 2010` – beginning of date range
- `MINIMUM_PRICE = 5.00` – penny stock filter
- `BATCH_SIZE = 500` – tickers per API query
- Size percentile cutoffs (cumulative from bottom):
  - `NANO_CUTOFF = 3.34`
  - `MICRO_CUTOFF = 18.83`
  - `SMALL_CUTOFF = 51.46`
  - `MID_CUTOFF = 78.60`
  - `LARGE_CUTOFF = 98.53`

train_predict_data4.py: Overview

- Complete pipeline from raw data to portfolio analysis
- Consolidates training, prediction, and portfolio formation
- Well-documented with 315 lines including extensive comments
- Designed as a template for rolling-window ML predictions
- Outputs `data4_portfolios.csv` and `data4_current.xlsx`

Four Main Steps:

1. Create percentile-ranked features
2. Train LightGBM with 12-month rolling windows (`data4_predict.parquet`)
3. Form decile portfolios and analyze (`data4_portfolios.csv`)
4. Predict current month and save model (`data4_current.xlsx`, `data4_model.pkl`)

train_predict_data4.py: Configuration

- `TRAINING_WINDOW = 12` – months in rolling window
- `N_PORTFOLIOS = 10` – number of decile portfolios
- `PARAMS = {...}` – LightGBM hyperparameters (dict)

data4_portfolios.csv: Overview

- Portfolio returns from sorting on LightGBM predictions
- 1,650 rows (165 months $\times$ 10 deciles)
- Date range: February 2012 to October 2025
- Created by `train_predict_data4.py`

Columns:

- `month` – prediction month
- `decile` – portfolio rank (1 $=$ lowest predicted, 10 $=$ highest)
- `return` – average realized return in decile
- `predict` – average predicted return rank in decile

Performance:

- Average monthly spread (D10 - D1): 2.50%

data4_current.xlsx: Overview

- Current month predictions for live trading/analysis
- Created by Step 4 of `train_predict_data4.py`
- Automatically detects current month (e.g., November 2025)
- Trains on last 12 complete months (e.g., Nov 2024 - Oct 2025)

Columns:

- `ticker` – stock ticker symbol
- `predict` – predicted return rank (sorted highest to lowest)
- All features from `data4.parquet` for current month

Use Cases:

- Identify top/bottom predicted stocks for current month
- Analyze characteristics of high vs. low predicted stocks

analyze_portfolios.ipynb: Overview

- Jupyter notebook for analyzing decile portfolio performance
- Reads `data4_portfolios.csv` (165 months × 10 deciles)
- Creates publication-quality visualizations and statistics
- Saves three PNG figures for presentations/papers

Analysis:

1. **Mean returns bar chart** – average monthly return by decile
2. **Sharpe ratios bar chart** – annualized Sharpe ratio by decile
3. **Cumulative returns** – two subplots (linear and log scale)
4. **Summary statistics** – mean, volatility, Sharpe, min, max
5. **Long-short portfolio** – D10 - D1 spread performance

analyze_model_features.ipynb: Overview

- Jupyter notebook for analyzing how the trained model uses features
- Reads `data4_model.pkl` (trained LightGBM model) and `data4_current.xlsx`
- Creates visualizations showing feature importance and linear relationships
- Saves two PNG figures for presentations/papers

Analysis:

1. **Feature importances pie chart** – from LightGBM split gain
2. **Linear regression** – predictions on percentile-ranked features
3. **Coefficient bar chart** – showing linear relationships (positive = green, negative = red)
4. **Comparison table** – feature importance ranks vs coefficient ranks

Interpretation:

LightGBM Hyperparameters for Return Prediction

Tree Structure

- `num_leaves = 31` – max leaves per tree (up to 30 splits/tree)
- `max_depth = 6` – shallow trees prevent overfitting to noise

Learning Parameters

- `learning_rate = 0.05` – moderate learning rate
- `n_estimators = 100` – fixed 100 boosting iterations (trees)

Regularization

- `min_child_samples = 50` – min samples per leaf
- `subsample = 0.8`, `colsample_bytree = 0.8` – sampling
- `reg_alpha = 0.1` (L1), `reg_lambda = 1.0` (L2)

Training

AI Code Editors: VS Code Forks

Cursor

- Fork of VS Code by Anysphere Inc. (San Francisco)
- Launched: March 2023
- First stable release (v1.0): June 2025
- cursor.com

Windsurf

- Fork of VS Code by Codeium
- Launched: November 13, 2024
- Branded as “agentic IDE” with AI Flows
- windsurf.com

Google's Acqui-Hire of Windsurf

Timeline

- OpenAI offered $3 billion to acquire Windsurf
- Exclusivity period expired: July 11, 2025
- Google executed $2.4B "reverse acqui-hire" + licensing deal
- CEO Varun Mohan and co-founder Douglas Chen joined Google DeepMind
- Google received non-exclusive license to Windsurf's AI technology

Why OpenAI's Deal Collapsed

- Microsoft (OpenAI's largest investor) has rights to OpenAI's acquired IP
- OpenAI wanted to keep Windsurf tech private from Microsoft
- Microsoft refused; deal collapsed when exclusivity expired
- Cognition later acquired remaining Windsurf assets for $250M

Google Antigravity

Release

- Announced: November 18, 2025 (alongside Gemini 3)
- Free public preview available immediately
- Available for Windows, macOS, and Linux

Features

- Fork of VS Code with agent-first interface
- Powered by Gemini 3 (also supports Claude Sonnet 4.5)
- Autonomous agents for planning, executing, and verifying tasks
- Integrates editor, terminal, and browser

Download

- antigravity.google/download