ISLP (Ch. 2+) → Trading/Research Exercises Using the Alpaca API

Goal: map each relevant chapter idea to a concrete, testable exercise that uses Alpaca market data (and optional Alpaca paper trading for execution). Chapters that don't map cleanly to Alpaca US equities/ETFs data are explicitly skipped.

Alpaca data you'll use in most exercises

Data type	Typical Alpaca endpoint/object	Notes
Daily/Minute OHLCV	Stocks bars (v2)	Backtests, features, labels
Trades	Stocks trades (v2)	Microstructure, slippage proxies
Quotes (NBBO)	Stocks quotes (v2)	Spread, mid-price, liquidity
Snapshots	Stocks snapshots	Latest state, quick screening
Paper trading (optional)	Trading API (paper)	Order simulation + logs

Chapter 2 — Statistical Learning (basics, KNN, train/test, bias–variance)

Exercise 2.1 — Build a 'market direction' dataset from Alpaca bars

- Pull SPY (or a basket of liquid ETFs) daily bars from Alpaca; compute daily return r_t.
- Label y_t = 1{r_{t+1} > 0}. Build features from lagged returns, rolling vol, volume change, intraday range.
- Split by time (no shuffling): train 2009–2018, validate 2019–2021, test 2022–present.
- Deliverable: a clean feature/label pipeline + leakage checks.

Note: Purely an analysis exercise (no trading).

Exercise 2.2 — KNN baseline vs. 'always up' / 'random' baselines

- Train KNN on the dataset above; tune k via walk-forward validation.
- Report accuracy, balanced accuracy, log loss; compare to baselines.
- Hypothesis test: does KNN accuracy exceed 50% on the test set? Use a binomial test (n = #test days).

Note: This teaches evaluation + basic hypothesis testing using Alpaca-derived labels.

Exercise 2.3 — Bias–variance demo with lookback length

- Vary feature lookback windows (e.g., 2, 5, 10, 20 days) and model flexibility (k in KNN).
- Plot training vs. test error vs. complexity; show where overfitting begins.

Note: Use only Alpaca data; keep design time-series aware.

Chapter 3 — Linear Regression (inference, interactions, diagnostics)

Exercise 3.1 — Predict next-day return and test if the model is useful

- Using Alpaca bars for SPY, predict r_{t+1} from lagged returns, vol, volume, range.
- Run OLS with HAC/Newey–West standard errors (time-series noise).
- Hypothesis test: H0: all slope coefficients = 0. Evaluate out-of-sample R^2 and significance.

Note: Expect small signal—this is the point.

Exercise 3.2 — Linear ‘factor’ model using ETF proxies

- Use ETFs as factor proxies (e.g., market=SPY, size=IWM, value=IWD, momentum=MTUM).
- For each stock in a universe, regress stock returns on factor returns to estimate exposures (betas).
- Trading overlay (optional): form a market-neutral portfolio that targets a factor (e.g., momentum) using estimated betas.

Note: Uses Alpaca price data only; treat as a toy factor lab.

Chapter 4 — Classification (logistic regression, LDA/QDA, Naive Bayes)

Exercise 4.1 — Logistic regression for Up/Down with proper calibration

- Train logistic regression on the Chapter 2 dataset.
- Evaluate ROC-AUC, calibration curve, Brier score; choose a probability threshold by maximizing expected value net of costs.

Note: Turn prediction into a decision rule.

Exercise 4.2 — ‘Cost-sensitive’ classifier for trading decisions

- Define payoff: +1 for correct long, -1 for wrong long, 0 for no-trade; include transaction cost c per trade.
- Train models and pick threshold that maximizes expected utility on validation.
- Backtest decision rule on test period; compare trade frequency vs. net return.

Note: Explicitly forces you to map classification metrics to trading P&L.;

Exercise 4.3 — LDA/QDA vs logistic under regime shift

- Train on a calm regime, test on a volatile regime (e.g., 2017–2019 train, 2020 test).
- Measure performance decay; do a stability test: difference in accuracy/ROC across regimes.

Note: Highlights distribution shift using Alpaca time splits.

Chapter 5 — Resampling Methods (CV, bootstrap)

Exercise 5.1 — Time-series cross-validation for strategy selection

- Implement expanding-window CV (walk-forward) for any model-based strategy above.
- Show that random K-fold CV is optimistic vs. walk-forward CV (quantify gap).

Note: Core best practice for finance.

Exercise 5.2 — Bootstrap Sharpe and max drawdown confidence intervals

- From a backtest return series, compute Sharpe and max drawdown.
- Use block bootstrap (e.g., stationary bootstrap) to respect autocorrelation.
- Report 95% CI; test H0: Sharpe ≤ 0.

Note: All inputs come from Alpaca-derived backtest returns.

Chapter 6 — Linear Model Selection & Regularization (ridge/lasso/PCR/PLS)

Exercise 6.1 — Feature explosion then Lasso selection

- Create a large feature set: multiple lags, rolling stats, interactions (vol×momentum), calendar dummies.
- Fit Lasso/ElasticNet with walk-forward CV; inspect selected features and stability over time.
- Trading rule: trade only when predicted return exceeds a cost-adjusted threshold; compare to unregularized OLS.

  Note: The point is controlling overfit.

Exercise 6.2 — PCA for 'synthetic factors' then regression

- Build a panel of returns for 30–100 liquid ETFs (Alpaca bars).
- Run PCA on returns; use top PCs as factors to predict SPY volatility/returns.
- Hypothesis: PCs improve forecast vs. raw lags; validate with walk-forward CV.

  Note: This bridges linear methods + finance factor intuition.

Chapter 7 — Moving Beyond Linearity (splines, GAMs, local regression)

Exercise 7.1 — Nonlinear effect of volatility on momentum

- Hypothesis: momentum works differently in high-vol vs low-vol states.
- Fit a GAM/spline model: expected next-day return ~ spline(volatility) + momentum + interaction.
- Backtest a volatility-conditioned momentum strategy (position size or on/off switch).

  Note: Nonlinearity should show up as shape differences, not magic alpha.

Exercise 7.2 — Local regression for intraday mean reversion (minute bars)

- Pull minute bars for a high-volume ETF (e.g., SPY) from Alpaca.
- Model short-horizon return as a smooth function of last-k minute move and spread proxy (from quotes).
- Backtest a conservative mean-reversion rule with strict trade limits.

  Note: Careful: microstructure + costs dominate at short horizons.

Chapter 8 — Tree-Based Methods (RF/GBM)

Exercise 8.1 — Random forest classifier with feature importance sanity checks

- Train RF to predict Up/Down using the same features as earlier.
- Check stability of feature importances across time folds; flag if importances jump wildly.
- Trading overlay: probability-threshold strategy; compare turnover and drawdowns vs logistic.

  Note: If feature importance is unstable, your 'edge' is likely noise.

Exercise 8.2 — Gradient boosting on cross-sectional momentum

- Universe: top 200 liquid stocks by dollar volume (Alpaca snapshots).
- At each month-end, build cross-sectional features (12-1 momentum, vol, liquidity proxies) from Alpaca bars.
- Train boosted model to predict next-month return rank; form long-short decile portfolio; evaluate with transaction costs.

  Note: This is closer to realistic quant equity research.

Chapter 9 — Support Vector Machines

Exercise 9.1 — SVM vs logistic under class imbalance

- Define ‘big move’ days: y_t = 1{|r_{t+1}| > threshold}. Rare-event classification.
- Train SVM with different kernels; compare to logistic with class weights.
- Hypothesis test: does the model meaningfully improve precision@k over baseline? Use permutation test on labels.

  Note: Connects margin methods to rare-event prediction.

Chapter 10 — Deep Learning (optional if you have GPU/time)

Exercise 10.1 — Sequence model for volatility forecasting (not return prediction)

- Use Alpaca bars to build sequences of returns/realized vol.
- Train a small RNN/1D-CNN to forecast next-week realized volatility for SPY/QQQ.
- Evaluate with walk-forward; use forecast only for volatility targeting (risk control), not directional bets.

  Note: Deep learning is more defensible for volatility than for direction.

Chapter 11 — Survival Analysis (SKIP for Alpaca)

Why skipped

- Chapter 11 focuses on time-to-event with censoring (e.g., patient survival).
- You *could* manufacture ‘time to hit barrier’ labels, but it’s not a clean match to the chapter’s core censoring setup.

  Note: Better to apply Ch. 11 to truly censored datasets rather than force it.

Chapter 12 — Unsupervised Learning (PCA, clustering)

Exercise 12.1 — Regime discovery via clustering

- Compute daily features for SPY: return, realized vol, skew proxy, volume, intraday range.
- Run K-means / Gaussian mixture to cluster days into regimes.
- Backtest a simple strategy that changes leverage or stops trading in the highest-risk cluster.

  Note: Unsupervised learning as risk management.

Exercise 12.2 — Hierarchical clustering for portfolio construction

- Universe: sector ETFs (XLK, XLF, XLE, etc.). Build correlation matrix from Alpaca returns.
- Perform hierarchical clustering and build a ‘cluster-balanced’ portfolio (equal weight within clusters).
- Evaluate diversification stability and drawdowns vs equal-weight naive.

  Note: A practical, transparent use of clustering.

Chapter 13 — Multiple Testing (p-hacking control)

Exercise 13.1 — Multiple-hypothesis screening of signals

- Generate many candidate signals (e.g., 200 variations of momentum/mean-reversion rules).
- For each signal, compute out-of-sample t-stat of mean daily return across walk-forward folds.
- Apply Benjamini–Hochberg FDR control to decide which signals ‘survive’.

  Note: Stops you from fooling yourself when you test lots of ideas.

Exercise 13.2 — Permutation-based p-values for strategy returns

- For a chosen strategy, compute a test statistic (Sharpe or mean return).
- Randomly permute returns in blocks (or permute signals) to estimate the null distribution.
- Report empirical p-value; compare to classical t-test assumptions.

Note: Finance returns violate iid assumptions—resampling is your friend.

Implementation checklist (so you don’t accidentally invalidate results)

- Always split by time (train/validate/test); never shuffle time-series data.
- Use realistic trading frictions: at least commissions (if any), bid–ask spread proxy, and slippage.
- Avoid look-ahead: indicators must use only data available at decision time.
- Prefer walk-forward CV for tuning and model selection.
- Keep a simple baseline (buy & hold, moving average, random/no-skill classifier).
- Log everything (data pull parameters, universe, rebalance dates, orders, fills).

Suggested minimal project structure

```
1) data.py (Alpaca fetch + caching) 2) features.py 3) models.py 4) backtest.py 5) eval.py 6) notebooks/
```

Book reference: This mapping is based on the chapter organization and topics in *An Introduction to Statistical Learning, With Applications in Python (ISLP)*.