Lab Report #3: Predictive Regression
Course: RSM338H1S, Winter 2026

Instructions:

- This assignment may be completed in **groups of up to 3 students**. Group members may be from either section. If you work in a group, submit one report with all names listed.
- Submit your lab report as a **PDF** to Crowdmark via Quercus. Export your Jupyter Notebook to PDF before uploading. Only one group member should upload the submission, being sure to select their group mates at the time of submission.
- There is no page limit, but be concise. A good report is thorough but not padded.

Marking:

- **75%** — Coding and results (correct implementation, complete answers to all parts, appropriate choice of methods, accurate numerical output, properly formatted tables and figures)
- **25%** — Overall quality (clear and professional writing, thoughtful interpretation of results, demonstrated understanding of the underlying concepts, logical flow and narrative structure)

Writing Expectations: Your report should read as a **coherent narrative**, not just code with scattered comments. Use section headers to indicate which problem you're working on. Before each code block, briefly explain what you are about to do and why. After results appear, interpret what you see. A reader should be able to understand your analysis even if they skipped the code cells.

You may use AI coding assistants (ChatGPT, Copilot, Claude, etc.) to help write code, but you must be able to explain what every line does. The text you write around the code is what demonstrates your understanding. You are ultimately responsible for your own work. **If you use an AI tool, you must disclose this in a note at the end of your report. Mention which tool you used, which tasks you asked it to complete, and discuss your (dis)satisfaction with its assistance.**

Assignment: You are a quantitative researcher at a systematic macro hedge fund. The CIO has asked you to evaluate whether macroeconomic and valuation variables can predict aggregate stock market returns. This is one of the oldest questions in empirical finance—and one of the most controversial. Your task is to replicate and extend the classic Welch and Goyal (2008) analysis, which famously showed that most predictors fail out-of-sample. The fund wants to know: Can modern machine learning techniques (regularization, cross-validation) do better than simple OLS? Your findings will inform whether to incorporate these signals into the fund's tactical allocation model.

Data Preparation

Data: The file `PredictorData2024.xlsx` contains the data for the Welch and Goyal (RFS 2008) paper, updated to the end of 2024. The data is from Professor Amit Goyal's website. We will be using only the monthly data for this assignment.

Before running any regressions, you must prepare the data. The raw Excel file contains many variables; you will construct the predictors used in the Welch-Goyal analysis.

(a) **Load the data.** Read the "Monthly" sheet from `PredictorData2024.xlsx` into a pandas DataFrame.

(b) **Parse dates.** The date column (`yyyymm`) is in YYYYMM format (e.g., 192701 for January 1927). Convert this to a proper datetime index.

(c) **Construct derived variables.** Using the raw columns in the dataset, construct the following predictors:

Variable	Formula	Description
`ExRet`	`CRSP_SPvw` − `Rfree`	Excess market return
`d/p`	$\ln($`D12`$) - \ln($`Index`$)$	Log dividend-price ratio
`d/y`	$\ln($`D12`$) - \ln($`Index`$_{t-1})$	Log dividend yield
`e/p`	$\ln($`E12`$) - \ln($`Index`$)$	Log earnings-price ratio
`d/e`	$\ln($`D12`$) - \ln($`E12`$)$	Log dividend-earnings ratio
`tms`	`lty` − `tbl`	Term spread
`dfr`	`corpr` − `ltr`	Default return spread
`dfy`	`BAA` − `AAA`	Default yield spread

The variables `svar`, `lty`, `ltr`, `tbl`, `b/m`, and `ntis` are already in the raw data and do not need to be constructed.

(d) **Lag inflation.** The inflation variable (`infl`) in the raw data is already lagged by one month relative to when it would have been known. For predictive regressions, lag it one additional month so that the inflation value used to predict month t returns is from month $t - 2$.

(e) **Filter the sample.** Use data from December 1926 onward (i.e., `index >= '1926-12-01'`).

(f) **Verification checkpoint.** After completing your data preparation, check that your values match approximately:

- `d/y` for January 1950: approximately −2.68
- `ExRet` for January 1950: approximately 0.0188 (1.88%)
- `tms` for January 1950: approximately 0.0108

If your values differ substantially, review your variable construction.

Problem 1: OLS Predictive Regressions

Consider the following predictive regression:

$$r_t = \alpha + \beta x_{t-1} + \epsilon_t, \quad t = 1927/1, \ldots, 2024/12,$$

where r_t is the excess (simple arithmetic) return of the value-weighted market portfolio (in excess of risk-free rate), and x_{t-1} is a lagged predictive variable. Notice the timing: you use last month's predictor to forecast this month's return.

Part (a): In-sample fit

Start by fitting this regression on the **entire sample at once**—just a single OLS regression per predictor, using all available data. This is the **in-sample** R^2:

$$R^2_{IS} = 1 - \frac{\sum_{t=1}^{T}(r_t - \hat{r}_t)^2}{\sum_{t=1}^{T}(r_t - \bar{r})^2}$$

where $\hat{r}_t$ are the fitted values and $\bar{r}$ is the sample mean of returns.

Run this regression for each of the 14 predictors: d/e, $svar$, dfr, lty, ltr, $infl$, tms, tbl, dfy, d/p, d/y, e/p, b/m, and $ntis$. Present the in-sample R^2 values in a table.

Why in-sample R^2 can be misleading

The in-sample R^2 tells you how well the model fits data it was *already trained on*—it's like grading a student on the practice exam they studied from. A model can achieve a high in-sample R^2 by fitting noise rather than genuine signal, especially when the true predictive relationship is weak (as it tends to be for stock returns).

The real question is: *if you had been using this model in real time, making forecasts month by month as new data arrived, how well would you have done?* To answer this, we use **out-of-sample** evaluation.

The expanding window

Here is how out-of-sample evaluation works. You simulate what a real-time forecaster would have experienced, starting with an initial training window of 200 months:

- **Month 201:** Estimate α and β using only months 1–200. Forecast $\hat{r}_{201} = \hat{\alpha} + \hat{\beta}\, x_{200}$. Record the forecast error $r_{201} - \hat{r}_{201}$.
- **Month 202:** Re-estimate α and β using months 1–201 (one more observation than before). Forecast $\hat{r}_{202}$. Record the error.
- **Continue** through the end of the sample (month T).

At every step, you re-estimate the model from scratch using all data available up to that point. The training window "expands" by one month each time. You never use any future data—only information that would have been available to a real-time forecaster.

You also need a **benchmark** to compare against. The simplest benchmark forecast is the expanding-window historical mean: at month t, just predict $\bar{r}_t$, the average of all past returns from months 1 through $t-1$. This benchmark also updates every month.

The **out-of-sample** R^2 compares your model's forecast errors to the benchmark's:

$$R^2_{OOS} = 1 - \frac{\sum_{t\in \text{OOS}}(r_t - \hat{r}_t)^2}{\sum_{t\in \text{OOS}}(r_t - \bar{r}_t)^2}$$

- $R^2_{OOS} > 0$: your model beats the historical mean benchmark.
- $R^2_{OOS} = 0$: your model is no better than just guessing the average.
- $R^2_{OOS} < 0$: your model is *worse* than the benchmark—it destroys value.

Because the regression is re-estimated at every month, the coefficients $\hat{\alpha}$ and $\hat{\beta}$ will move around over time. Early on, when the training window is short (200 months), estimates are noisy and can shift substantially with each new observation. As the window grows, the estimates stabilize. The OOS R^2 averages over all of these forecasts—some made with short training histories, some with long ones—giving you an overall measure of real-time predictive performance.

Certainty equivalent value (ΔCEV)

R^2_{OOS} is a statistical measure, but we also want to know: *would these forecasts have made an investor any money?*

ΔCEV answers this question. Imagine a mean-variance investor who, each month, decides how much to allocate to the stock market based on a return forecast. If the forecast says returns will be high, the investor puts more money in stocks; if it says returns will be low, the investor pulls back. ΔCEV measures how much better off (in annualized percentage points) this investor would be using the model's forecast compared to just using the historical average.

$\Delta\text{CEV} > 0$ means the model adds value; $\Delta\text{CEV} < 0$ means it destroys value.

Here is how to compute it. At each month t in the OOS period, you already have the model forecast $\hat{r}_t$, the benchmark forecast $\bar{r}_t$, and the expanding-window variance $\hat{\sigma}^2_t = \text{Var}(r_1, \ldots, r_{t-1})$. All three update every month.

Model portfolio weight and return:

$$w_t^{\text{model}} = \frac{1}{\gamma} \cdot \frac{\hat{r}_t}{\hat{\sigma}^2_t}, \quad \text{clipped to } [0, 1.5] \qquad\qquad r_{p,t}^{\text{model}} = w_t^{\text{model}} \cdot r_t$$

Benchmark portfolio weight and return:

$$w_t^{\text{bench}} = \frac{1}{\gamma} \cdot \frac{\bar{r}_t}{\hat{\sigma}_t^2}, \quad \text{clipped to } [0, 1.5] \qquad\qquad r_{p,t}^{\text{bench}} = w_t^{\text{bench}} \cdot r_t$$

Certainty equivalents (computed once, over the full OOS period):

$$\text{CE}_{\text{model}} = \overline{r_p^{\text{model}}} - \frac{\gamma}{2} \cdot \text{Var}(r_p^{\text{model}}), \qquad \text{CE}_{\text{bench}} = \overline{r_p^{\text{bench}}} - \frac{\gamma}{2} \cdot \text{Var}(r_p^{\text{bench}})$$

where $\overline{r_p}$ and $\text{Var}(r_p)$ are the mean and variance of portfolio returns across all OOS months.

The gain:

$$\Delta\text{CEV} = (\text{CE}_{\text{model}} - \text{CE}_{\text{bench}}) \times 12 \times 100$$

The $\times 12$ annualizes (returns are monthly) and $\times 100$ converts to percentage points.

Part (b): Out-of-sample evaluation

Now go back to each of your 14 single-predictor regressions and compute the out-of-sample R^2 and ΔCEV using the expanding-window procedure described above. Use an initial window of 200 months and $\gamma = 2.5$.

Add the OOS R^2 and ΔCEV columns to your table from part (a) so you can compare in-sample and out-of-sample performance side by side. Which predictors looked promising in-sample but fail out-of-sample?

Part (c): Multivariate regression

Now include all the predictors simultaneously in a single regression. Drop d/e and tms to avoid multicollinearity (they are exact linear combinations of other predictors: $d/e = d/p - e/p$ and $tms = lty - tbl$). Report the in-sample R^2, out-of-sample R^2, and ΔCEV. How does the multivariate model compare to the best single-predictor model?

Problem 2: Ridge and Lasso Regression

In Problem 1, you dropped d/e and tms from the multivariate regression to avoid multicollinearity. Regularized regression offers an alternative: instead of manually removing predictors, we add a penalty term that shrinks coefficients toward zero, which also stabilizes the estimates.

Ridge regression minimizes:

$$\sum_t (r_t - \alpha - \mathbf{x}'_{t-1}\boldsymbol{\beta})^2 + \lambda \sum_j \beta_j^2$$

Lasso regression minimizes:

$$\sum_t (r_t - \alpha - \mathbf{x}'_{t-1}\boldsymbol{\beta})^2 + \lambda \sum_j |\beta_j|$$

The parameter $\lambda \geq 0$ controls how much shrinkage to apply. When $\lambda = 0$, both reduce to OLS. Larger λ means more shrinkage. Ridge shrinks all coefficients toward zero but keeps them nonzero; Lasso can shrink some coefficients to *exactly* zero, effectively dropping those predictors.

Selecting λ with cross-validation

Rather than choosing λ by hand, we let cross-validation pick the best value from a set of candidates. `sklearn`'s `RidgeCV` and `LassoCV` automate this: they try different λ values, evaluate each one using cross-validation, select the best, and refit on all the training data.

There is one subtlety. By default, `sklearn`'s cross-validation randomly shuffles data into folds. For time series, this is a problem: a fold might train on 1950s data and validate on 1930s data, letting the future leak into the past. To prevent this, pass `cv=TimeSeriesSplit(n_splits=5)` (from `sklearn.model_selection`). This splits the training data into 5 time-ordered folds, always training on earlier months and validating on later months.

For `RidgeCV`, supply the candidate grid `alphas=[0.001, 0.01, 0.1, 1, 10, 100]`. For `LassoCV`, you can omit the `alphas` parameter—it computes its own λ path automatically.

Putting it together

The **outer** expanding-window loop is exactly the same as Problem 1. At each OOS month t, you take data through month $t-1$ and:

1. Run `RidgeCV` or `LassoCV` on that data. Internally, this splits the training data into time-ordered folds, tries different λ values, picks the best one, and refits on all of months 1 through $t-1$.
2. Forecast $\hat{r}_t$.
3. Compute portfolio weights and returns, just as in Problem 1.

Part (a)

Using *all* 14 predictors (including d/e and tms), estimate both Ridge and Lasso regressions with cross-validated λ. Use the same expanding window as Problem 1. For each method, report:

- The cross-validated $\hat{\lambda}$ selected at the final month of the OOS period
- Out-of-sample R^2
- ΔCEV

Part (b)

Which predictors does Lasso shrink to exactly zero (at the final OOS month)? Interpret this result in terms of the signal-to-noise ratio of these predictors. How does it relate to what you found in Problem 1?

Part (c)

Compare the out-of-sample performance of Ridge and Lasso to the OLS multivariate regression from Problem 1(c). Does regularization help? Given that these predictors are publicly available and well known, why might we expect this result?

Problem 3: Elastic Net and Summary

Elastic Net combines Ridge and Lasso by minimizing:

$$\sum_t (r_t - \alpha - \mathbf{x}'_{t-1}\boldsymbol{\beta})^2 + \lambda \left[\rho \sum_j |\beta_j| + (1-\rho) \sum_j \beta_j^2 \right]$$

where $\rho \in [0, 1]$ controls the mix between Lasso ($\rho = 1$) and Ridge ($\rho = 0$). (`sklearn` calls this parameter `l1_ratio`.) You can use `ElasticNetCV` with `cv=TimeSeriesSplit(n_splits=5)`, the same as in Problem 2.

Part (a)

Using cross-validation to select both λ and ρ, compute the out-of-sample R^2 and ΔCEV.

Part (b)

Summarize your findings across all methods in a single table: OLS single-predictor (best performing), OLS multivariate, Ridge (CV), Lasso (CV), and Elastic Net (CV). Include both OOS R^2 and ΔCEV for each.

Part (c)

Which approach works best for predicting excess stock returns? What does this tell you about the predictability of returns and the value of regularization in this setting?

