Instructions

- Homework 3 is due December 14th at 16:00 Chicago Time.
  - We will not accept any submissions past 16:00:00, even if they are only one second late.
- You **must** upload the following files to the class Canvas:
  - `LASTNAME_FIRSTNAME.pdf`
  - `LASTNAME_FIRSTNAME.ipynb`
- Your code notebook **must** be runnable using my environment outlines in class 1 (Python 3.14, and the `requirements.txt`).
- You **must** use this template file and fill out your solutions for the written portion.
- Please note that your last name and first name should match what you appear on Canvas as.
- Include code snippets where required, as well as math and equations.
- Be *concise* where possible, all of the homework probelms can be answered in a few lines of math, code, and words.

Problem 1: Ridge Regression and Stability

Generate a dataset with $n = 100$ observations based on the true model:

$$y = 1 + x_1 + x_2 + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 1)$.

Additionally, make $x_1, x_2 \sim \mathcal{N}(0, 1)$ highly correlated, with $\rho = 0.99999$. You can do this via the following code snippet:

```
import numpy as np
np.random.seed(1)

mean = [0, 0]
cov = [[1, 0.99999], [0.99999, 1]]
x1, x2 = np.random.multivariate_normal(mean, cov, 100).T
epsilon = np.random.normal(0, 1, 100)
```

Problem 1.1: OLS Inference

Fit an OLS model to your generated data using `statsmodels`.

- Report the p-values and confidence intervals for $\hat{\beta}_1$ and $\hat{\beta}_2$.
- Discuss the results. Are the coefficients close to the true values (1 and 1)? Are they statistically significant?

Answer:

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	0.9940	0.105	9.483	0.000	0.786	1.202
X1	24.1321	24.274	0.994	0.323	-24.046	72.310
X2	-22.1958	24.257	-0.915	0.362	-70.340	25.948

The coefficients are far from the true values of 1 and 1, and neither coefficient is statistically significant (high p-values). This is likely due to the high multicollinearity between x_1 and x_2, which inflates the variance of the coefficient estimates.

Problem 1.2: Stability

Now, re-generate your data using `np.random.seed(42)` and fit the OLS model again.

- Report the new estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$.
- Compare these estimates to the previous ones. What do you notice?

Answer:

	coef	std err	t	P> |t|	[0.025	0.975]
const	1.0928	0.108	10.115	0.000	0.878	1.307
X1	39.3968	24.079	1.636	0.105	-8.393	87.187
X2	-37.5875	24.083	-1.561	0.122	-85.386	10.211

Comparing to the previous estimates, we see that the coefficients for $\hat{\beta}_1$ and $\hat{\beta}_2$ have changed significantly.

Problem 1.3 Variance via Simulation

To better understand this phenomenon, run a simulation (note, you will need to unset the random seed for this part):

- Run a simulation loop 1000 times. In each iteration, generate a new dataset $(x1, x2, \epsilon)$ and fit an OLS model.
- Store the estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$.

Plot 2 histograms showing the distribution of the estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$. Additionally, report the volatility (standard deviation) of your $\hat{\beta}_1$ and $\hat{\beta}_2$ estimates across the simulations.

Answer:

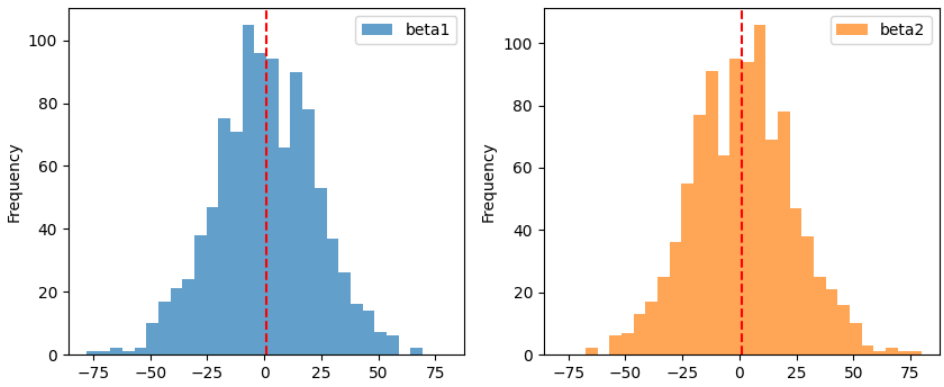

Figure 1: Histograms of OLS estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$ across 1000 simulations. The red dashed lines indicate the true value of 1.

We also see the standard deviations of the estimates:
beta1 22.069571
beta2 22.069610

Problem 1.4: Ridge to the Rescue

Now, fit a Ridge Regression model using `statsmodels` (you can use `fit_regularized` with `L1_wt=0` for Ridge).

- Use three different values for alpha (lambda): 0.01, 0.1, and 1.
- For each alpha, report the coefficients $\hat{\beta}_1$ and $\hat{\beta}_2$.
- Discuss how the coefficients change with different alpha values.

Answer:

alpha	const	X1	X2
0.01	0.902489	0.906318	0.944797
0.1	0.824985	0.877538	0.880740
1	0.441079	0.587218	0.587153

The coefficients for both $\hat{\beta}_1$ and $\hat{\beta}_2$ decrease as the alpha value increases.

Note: We first see that the corrleated features have approximately equal contributions to the outcome. This is because by ridge regression, numerically computing the inverse of $X'X + \alpha I$ is more stable than that of $X'X$ alone.

Note2: With small alpha, the Ridge estimates are closer to our population model. This is because as $\alpha \to 0$, the ridge estimates converge to what we expect from OLS. Check **Ridgeless regression** for more details. (It is possible to derive the closed form solution and see this property mathematically.)

Problem 1.5: Ridge Simulation

Next, you should repeat the simulation from subproblem 1.3, but this time use a Ridge model with a fixed alpha of 0.01.

- Run the simulation loop 1000 times, generating new datasets and fitting the Ridge model each time.
- Store the estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$.

Plot 2 histograms showing the distribution of the Ridge estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$. Additionally, report the volatility (standard deviation) of your Ridge $\hat{\beta}_1$ and $\hat{\beta}_2$ estimates across the simulations. Compare these volatilities to those obtained from the OLS simulation in subproblem 1.3.

Answer:

The volatilities of the Ridge estimates are:
beta1 0.054180
beta2 0.054505
which are lower than the OLS volatilities from subproblem 1.3 (around 22).

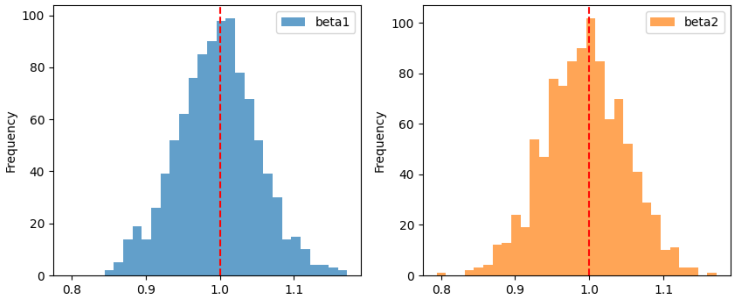

Figure 2: Histograms of Ridge estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$

Problem 2: Lasso Regression and Sparsity

This problem explores Lasso regression for a sparse model.

Generate a dataset with $n = 100$ observations and $p = 50$ features. The true model depends on only the first 3 features, while the remaining 47 features are irrelevant.

$$y = 5x_1 - 2x_2 + 3x_3 + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 1)$ and all $x_j \sim \mathcal{N}(0, 1)$.

Use the following code snippet to generate your data:

```
import numpy as np
np.random.seed(42)

n_samples = 100
n_features = 50

X = np.random.normal(0, 1, (n_samples, n_features))
true_beta = np.zeros(n_features)
true_beta[:3] = [5, -2, 3]

y = np.dot(X, true_beta) + np.random.normal(0, 1, n_samples)
```

Problem 2.1: Naive OLS

Fit an OLS model using all 50 features.

- Look at the coefficients for the 47 "noise" features (x_4 through x_{50}). Are they exactly zero (you don't need to report them all)?
- How many of these noise features have p-values < 0.05 (i.e., appear statistically significant purely by chance)?

Answer:
Non-zero coefficients (OLS): 47
Number of significant coefficients (OLS): 4

Problem 2.2: Lasso for Feature Selection

Now, fit a Lasso model using `statsmodels`. You can do this using `fit_regularized` with `L1_wt=1` (which specifies pure Lasso). Use an alpha of 0.1, 1.0, and 100.

- Report the estimated coefficients.
- How many coefficients are estimated to be exactly zero?
- Did the Lasso model correctly identify the 3 relevant features (x_1, x_2, x_3) while suppressing the noise?
- Which alpha value would you recommend?

Answer:
Number of non-zero coefficients (Lasso, alpha=0.1): 20
Number of non-zero coefficients (Lasso, alpha=1): 3
Number of non-zero coefficients (Lasso, alpha=100): 0

The estimated coefficients are omitted for $\alpha = 0.1$. with $\alpha = 1$, LASSO correctly identify the 3 relevant features while suppressing the noise, giving the fitted model as:

$$\hat{y} = 3.7783x_1 - 0.9403x_2 + 2.0392x_3$$

As such, I would recommend $\alpha = 1$.

Problem 2.3: The Regularization Path

The sparsity of the model depends heavily on the strength of α.

- Create a list of alphas: `[0.001, 0.01, 0.1, 0.5, 1.0, 5.0, 10, 100, 1000]`.
- Loop through these alphas, fitting a Lasso model for each.
- For each alpha, store the values of all coefficients.

Plot a graph with $\log(\alpha)$ on the x-axis and the coefficients on the y-axis (you can plot each coefficient as a separate line). Discuss how the coefficients change as alpha increases, and what the bias looks like (how far are the first 3 coefficients from their true values?).

Hint: It might be useful to plot the irrelevant coefficients in gray, and the relevant ones in different colors.

Answer:

The coefficients for the relevant features (x_1, x_2, x_3) decrease in magnitude as alpha increases, indicating increased bias. The irrelevant features' coefficients shrink to zero more quickly, demonstrating Lasso's ability to perform feature selection. As alpha becomes very large, all coefficients approach zero.

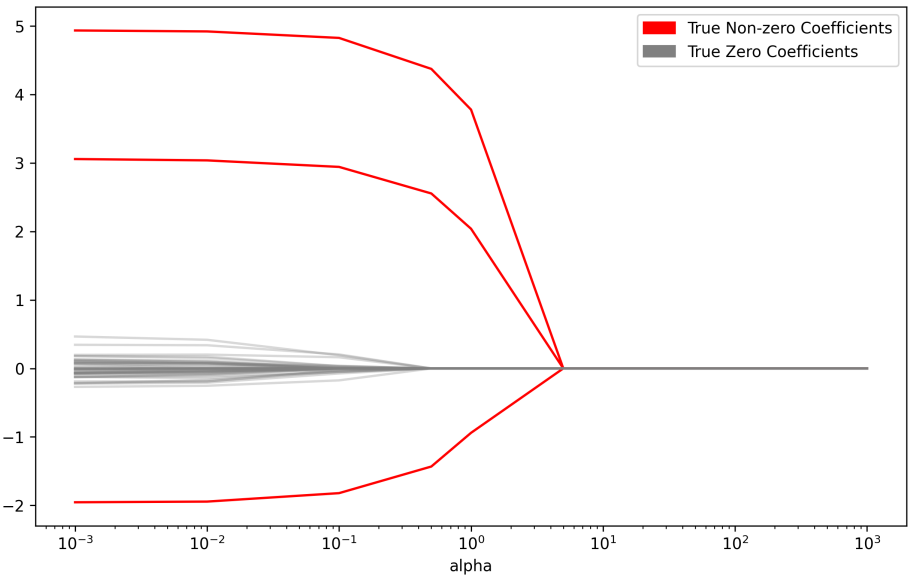

Figure 3: Lasso Regularization Path: Coefficients vs. $\log(\alpha)$

Problem 3: Time Series and Time-Varying Heteroskedasticity

Generate a dataset with $n = 1000$ observations. The data follows an AR(1) process:

$$y_t = 0.5y_{t-1} + \epsilon_t$$

The error term ϵ_t follows an ARCH(1) process plus a regime:

$$\epsilon_t = \sigma_t z_t, \quad z_t \sim \mathcal{N}(0,1)$$

$$\sigma_t^2 = 1 + 0.5\epsilon_{t-1}^2 + 10 \cdot \mathbb{I}_{\text{HighRegime}}$$

Use the following code snippet to generate your data:

```python
import numpy as np
import pandas as pd

np.random.seed(100)
n = 1000

indicator = np.zeros(n)
for i in range(0, n, 200):
    indicator[i:i+100] = 1

epsilon = np.zeros(n)
sigma2 = np.zeros(n)
epsilon[0] = np.random.normal(0, 1)

for t in range(1, n):
    sigma2[t] = 1 + 0.5 * (epsilon[t-1]**2) + 10 * indicator[t]
    epsilon[t] = np.random.normal(0, np.sqrt(sigma2[t]))
```

```
18
19 y = np.zeros(n)
20 phi = 0.5
21 for t in range(1, n):
22     y[t] = phi * y[t-1] + epsilon[t]
```

Problem 3.1: The AR(1) Model

First, fit an AR(1) model to the generated series y_t.

$$y_t = \phi y_{t-1} + \epsilon_t$$

Extract the residuals ϵ_t from this model.

Create two plots:

- Residuals vs. fitted values.
- Residuals over time.

What do you notice about the first graph vs. the second graph?

Answer:

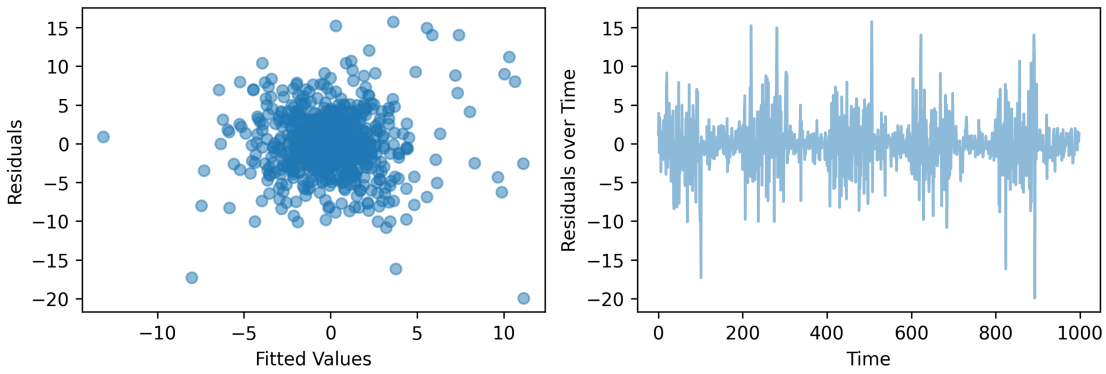

Figure 4: AR(1) Model Residuals

The first graph (Residuals vs. Fitted Values) shows no clear pattern. However, the second graph (Residuals over Time) reveals periods of high and low volatility, suggesting time-varying heteroskedasticity (volatility clustering) in the residuals.

Problem 3.2: ACF

Plot the Autocorrelation Function (ACF) of the squared residuals $\hat{\epsilon}_t^2$.

What does the ACF plot suggest about the independence of the squared residuals?

Answer:

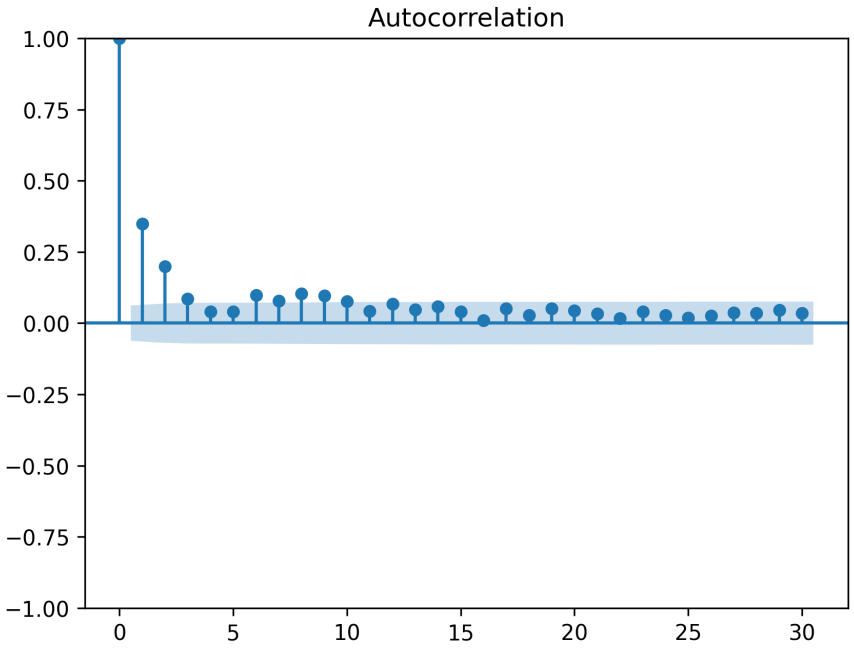

Figure 5: ACF of Squared Residuals

The ACF plot shows significant autocorrelations at early lags, indicating that the squared residuals are not independent.

Problem 3.3: Modeling Variance (ARCH + Regime)

Now, we will model the variance of these residuals explicitly. We hypothesize that the variance σ_t^2 depends on the previous squared residual (ARCH effect) and the regime indicator.

- Construct the squared residuals $\hat{\epsilon}_t^2$.
- Construct a lagged feature $\hat{\epsilon}_{t-1}^2$.
- Run a linear regression of $\hat{\epsilon}_t^2$ on:

$$\sigma_t^2 = \beta_0 + \beta_1 \hat{\epsilon}_{t-1}^2 + \beta_2 \mathbb{I}_{\text{HighRegime}}$$

- Report the coefficients, compare them to the true values ($\beta_0 = 1$, $\beta_1 = 0.5$, $\beta_2 = 10$), and discuss their statistical significance.

Answer:

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	2.9868	1.233	2.422	0.016	0.567	5.406
resid2_lag1	0.2994	0.030	9.927	0.000	0.240	0.359
indicator	11.3243	1.795	6.307	0.000	7.801	14.848

The estimated coefficients are roughly similar to the true values, but with some deviations. This is because the linear model fits the residual distribution (which is conditional to the data $T \leq 1000$) rather than the true distribution of the error term.

Problem 3.4: Regime-Dependent Forecasting

Suppose we are at the end of the series ($T = 1000$) and we want to predict the range of movement for the next step ($T = 1001$). Assume the last observed residual was $\hat{\epsilon}_{1000} = 2$.

Calculate the predicted variance $\hat{\sigma}^2_{1001}$ and the resulting 95% Confidence Interval width ($\pm 1.96\hat{\sigma}_{1001}$) for the error term under two scenarios:

- **Scenario A (Low Regime):** Assume the indicator for $T = 1001$ is 0.
- **Scenario B (High Regime):** Assume the indicator for $T = 1001$ is 1.

(Use the coefficients you estimated in 3.3).

What do you observe about your confidence intervals in the two scenarios?

Answer:

	Lower Bound	Upper Bound
Low Regime	-4.009417	4.009417
High Regime	-7.718727	7.718727

In the Low Regime, the width of the confidence interval is narrower, indicating lower uncertainty in the prediction. In contrast, the High Regime results in a much wider confidence interval, reflecting increased uncertainty due to higher volatility.