Week 5 Assignment:
Prompt-Assisted Feature Engineering for Portfolio Optimization

MSc Banking and Finance – FinTech Course

Due Date: Tuesday, October 23, 2025

Abstract

This assignment tests your ability to use AI prompts to discover meaningful features, implement Ridge & Lasso regularization, evaluate feature importance, and apply findings to real portfolio optimization. Total Points: 100. Submission: Jupyter Notebook + PDF Report + GitHub Repository.

Assignment Overview

This assignment evaluates your ability to:

1. Use **AI prompts** to discover meaningful features
2. Implement **Ridge & Lasso** regularization
3. Evaluate **feature importance** and model performance
4. Apply findings to **real portfolio optimization**

Core Task: Create 3 new features using prompt engineering, implement them in Python, and prove they improve model performance.

Contents

1 Assignment Tasks

1.1 Task 1: Prompt Engineering for Feature Discovery (25 points)

Task 1.1: Generate Features Using AI Prompts (10 points)

Create **3 novel features** using AI assistance (ChatGPT, Claude, or Gemini). Your prompts should explore:

- Market microstructure (e.g., bid-ask patterns, volume dynamics)
- Behavioral finance signals (e.g., sentiment, herding indicators)
- Cross-asset relationships (e.g., correlations, spreads, ratios)
- Regime detection (e.g., bull/bear, high/low volatility)

Example Prompt Template:

Given cryptocurrency data with the following available features:

- *Return forecasts (ARIMA)*
- *Volatility forecasts (GARCH)*
- *Historical prices*
- *Trading volumes*

Suggest 3 features that capture [SPECIFIC PATTERN] and explain:

1. *The financial intuition behind each feature*
2. *The mathematical formula to calculate it*
3. *Why it might predict portfolio returns*

Deliverable 1.1:

- Document your **3 prompts** (exact text sent to AI)
- Show **AI responses** for each prompt

Task 1.2: Feature Justification (15 points)

For each of your 3 features, provide:

1. **Financial Intuition (5 points per feature)**
   - What market behavior does this feature capture?
   - Why should it predict portfolio returns?
   - What's the economic theory behind it?
2. **Mathematical Specification (5 points per feature)**
   - Write the formula clearly
   - Define all variables
   - Explain any transformations (logs, differences, ratios)
3. **Expected Impact (5 points per feature)**
   - Will it improve Lasso or Ridge more? Why?
   - What's your hypothesis about its importance?

Example Format:

Feature 1: Bitcoin-Ethereum Volatility Ratio

Financial Intuition: During market stress, BTC and ETH volatilities diverge. BTC (digital gold) becomes less volatile, while ETH (DeFi platform) experiences higher volatility. This ratio signals flight-to-quality dynamics.

Formula:

$$\text{vol\_ratio} = \frac{\sigma_{\text{BTC},t}}{\sigma_{\text{ETH},t}} \tag{1}$$

Where:

- $\sigma_{\text{BTC},t}$ = GARCH volatility forecast for BTC at time t
- $\sigma_{\text{ETH},t}$ = GARCH volatility forecast for ETH at time t

Expected Impact: Should improve Lasso performance by capturing regime changes. Hypothesis: Non-zero coefficient when vol_ratio < 0.8 (stress periods) or > 1.2 (calm periods).

1.2 Task 2: Implementation & Testing (35 points)

Task 2.1: Feature Implementation (10 points)

Implement your 3 features in Python. Code must:

- ✓ Be well-commented
- ✓ Handle edge cases (NaN, inf, division by zero)
- ✓ Use vectorized operations (no slow loops)
- ✓ Include unit tests (assert statements)

```python
def calculate_feature_1(btc_vol, eth_vol):
    """
    Calculate BTC-ETH volatility ratio.

    Parameters:
    -----------
    btc_vol : array-like
        BTC GARCH volatility forecasts
    eth_vol : array-like
        ETH GARCH volatility forecasts

    Returns:
    --------
    vol_ratio : array-like
        Ratio of BTC to ETH volatility
    """
    # Handle division by zero
    eth_vol_safe = np.where(eth_vol == 0, 1e-10, eth_vol)
    vol_ratio = btc_vol / eth_vol_safe

    # Sanity check
    assert np.all(np.isfinite(vol_ratio)), "Non-finite values detected"
    assert np.all(vol_ratio > 0), "Negative ratios detected"

    return vol_ratio

# Test your function
btc_test = np.array([0.04, 0.05, 0.03])
eth_test = np.array([0.05, 0.04, 0.05])
result = calculate_feature_1(btc_test, eth_test)
print(f"Test passed: {result}")  # [0.8, 1.25, 0.6]
```

Task 2.2: Baseline Performance (10 points)

First, establish baseline performance **without** your new features:

1. Train Ridge and Lasso on **original features only** (from Week 5 code)
2. Use 5-fold cross-validation
3. Record: R^2, MSE, number of features selected (Lasso)

```python
# Baseline evaluation
baseline_ridge_r2 = ...
baseline_lasso_r2 = ...
baseline_lasso_n_features = ...
```

Task 2.3: Enhanced Model Performance (15 points)

Now add your 3 features and re-evaluate:

1. Combine original + your 3 new features
2. Re-train Ridge and Lasso with same CV setup
3. Record same metrics
4. **Prove improvement** using statistical tests

Deliverable 2.3: Comparison Table:

Metric	Baseline Ridge	Enhanced Ridge	Baseline Lasso	Enhanced Lasso
CV R^2	0.72	**0.78**	0.70	**0.81**
CV MSE	0.00045	**0.00038**	0.00048	**0.00035**
Features	20	23	8	**11**
Improvement	–	+8.3%	–	+15.7%

Table 1: Model Performance Comparison

Statistical Significance Test:

```python
from scipy import stats

# Paired t-test on CV fold scores
baseline_scores = [0.70, 0.68, 0.72, 0.71, 0.69]
enhanced_scores = [0.79, 0.81, 0.82, 0.78, 0.85]

t_stat, p_value = stats.ttest_rel(enhanced_scores, baseline_scores)
print(f"t-statistic: {t_stat:.3f}, p-value: {p_value:.4f}")

if p_value < 0.05:
    print("Success: Improvement is statistically significant!")
```

1.3 Task 3: Feature Importance Analysis (20 points)

Task 3.1: Lasso Coefficient Analysis (10 points)

1. **Rank features** by absolute Lasso coefficients
2. **Identify** which of your 3 features were selected (non-zero)
3. **Compare** to original features

Deliverable 3.1:

```python
# Feature importance
feature_importance = pd.DataFrame({
    'feature': feature_names,
    'lasso_coef': np.abs(lasso_model.coef_),
    'is_new': ['Yes' if f in new_features else 'No'
               for f in feature_names]
}).sort_values('lasso_coef', ascending=False)

print(feature_importance.head(10))
```

Visualization Required:

```
# Bar plot of top 10 features
plt.figure(figsize=(10, 6))
top10 = feature_importance.head(10)
colors = ['green' if x == 'Yes' else 'blue' for x in top10['is_new']]
plt.barh(top10['feature'], top10['lasso_coef'], color=colors)
plt.xlabel('Absolute Lasso Coefficient')
plt.title('Top 10 Important Features (Green = New Features)')
plt.tight_layout()
```

Task 3.2: Ablation Study (10 points)

Test each new feature individually:

```
# Test Feature 1 alone
X_with_f1 = np.column_stack([X_baseline, feature_1])
score_f1 = cross_val_score(Lasso(), X_with_f1, y, cv=5).mean()

# Test Feature 2 alone
X_with_f2 = np.column_stack([X_baseline, feature_2])
score_f2 = cross_val_score(Lasso(), X_with_f2, y, cv=5).mean()

# Test Feature 3 alone
X_with_f3 = np.column_stack([X_baseline, feature_3])
score_f3 = cross_val_score(Lasso(), X_with_f3, y, cv=5).mean()

# Rank by individual contribution
```

Deliverable 3.2:

- Table showing each feature's **individual contribution**
- Identify which feature is **most valuable**
- Discuss **interactions** between features (if any)

1.4 Task 4: Portfolio Optimization & Business Insights (20 points)

Task 4.1: Optimal Portfolio Weights (10 points)

Use your enhanced Lasso model to:

1. Predict expected returns for each asset
2. Calculate optimal portfolio weights
3. Compare to **baseline** (equal-weighted) portfolio

```
# Your enhanced model predictions
expected_returns = {
    'BTC': lasso_predict_btc,
    'ETH': lasso_predict_eth,
    'DOGE': lasso_predict_doge
}

# Optimal weights (simple softmax approach)
```

```
weights_optimized = calculate_softmax_weights(expected_returns)

# Baseline: equal weighted
weights_baseline = {'BTC': 0.33, 'ETH': 0.33, 'DOGE': 0.34}
```

Task 4.2: Backtest & Performance (10 points)

Simulate portfolio performance over the test period:

1. Calculate daily portfolio returns for both strategies
2. Compute: Sharpe ratio, max drawdown, cumulative return
3. Visualize cumulative returns

Deliverable 4.2:

Metric	Baseline (Equal Weight)	Optimized (ML-Enhanced)
Annual Return	12.3%	**18.7%**
Annual Volatility	35.2%	**31.8%**
Sharpe Ratio	0.35	**0.59**
Max Drawdown	-28.1%	**-22.4%**

Table 2: Portfolio Performance Comparison

```
# Cumulative return plot
plt.figure(figsize=(12, 6))
plt.plot(dates, cumret_baseline, label='Baseline', alpha=0.7)
plt.plot(dates, cumret_optimized, label='ML-Enhanced', linewidth=2)
plt.xlabel('Date')
plt.ylabel('Cumulative Return')
plt.title('Portfolio Performance Comparison')
plt.legend()
plt.grid(True, alpha=0.3)
```

2 Evaluation Criteria

2.1 Grading Rubric

Component	Points	Criteria
Task 1: Prompt Engineering	25	Quality of prompts (5), Financial intuition (10), Math clarity (10)
Task 2: Implementation	35	Code quality (10), Baseline (10), Improvement proof (15)
Task 3: Analysis	20	Lasso importance (10), Ablation study (10)
Task 4: Portfolio Insights	20	Optimal weights (10), Backtest results (10)
TOTAL	**100**	

Table 3: Grading Breakdown

2.2 Bonus Points (Optional)

- Implement **ElasticNet** and compare to Ridge/Lasso
- **Hyparameter-tunig** visualization

3 Submission Requirements

3.1 Deliverables:

- **Jupyter Notebook** (.ipynb)
- **PDF Report** (1-3 pages)
- **Code Repository** (GitHub)

4 Recommended AI Prompts for Inspiration:

Prompt 1 – Technical Analysis:

> *I have cryptocurrency data with ARIMA return forecasts and GARCH volatility forecasts. Suggest 3 technical indicators that could predict portfolio returns, focusing on momentum and mean-reversion signals. Provide formulas suitable for Python implementation.*

Prompt 2 – Risk Metrics:

> *Given forecasted volatility for BTC, ETH, and DOGE, what are 3 risk-based features that capture portfolio diversification benefits? Include correlation-based and tail-risk measures.*

Prompt 3 – Market Regime:

> *Design 3 features that identify different market regimes (bull/bear, high/low volatility, trending/ranging) using only return and volatility forecasts. Explain the threshold logic for each.*

Remember

> *"Feature engineering is the art of asking the right questions about your data."*
>
> The best features come from:
>
> 1. **Deep understanding** of the domain (behavioral finance, market microstructure)
> 2. **Creative thinking** (AI prompts help, but you must curate)
> 3. **Rigorous testing** (cross-validation never lies)
> 4. **Iteration** (first attempt rarely succeeds)