Chapter 3: Financial Feature Engineering – How to Research Alpha Factors

1. Introduction: What is Feature Engineering and Why Does it Matter?

Feature engineering is the process of transforming raw financial data into meaningful variables (called “features” or “factors”) that can be used as inputs for machine learning models. In trading, these features are often called **alpha factors**—signals that might help predict future price movements or returns.

Why is this important?
No matter how advanced your machine learning model is, it can only learn from the information you give it. If your features are weak or irrelevant, your model will not discover profitable patterns. Powerful, well-designed features are the secret sauce behind most successful trading strategies.

Analogy:
Think of features as clues in a detective story. The more relevant and insightful the clues, the more likely you are to solve the mystery (predict price movements).

2. What Makes a Good Alpha Factor?

A good alpha factor is a feature that helps you predict future returns better than random guessing. But not all features are created equal. A strong factor should be:

- **Predictive:** It should have some relationship to future price changes.
- **Robust:** It should work across different stocks, time periods, and market conditions.
- **Unique:** It shouldn’t just duplicate information from other features.
- **Economically sensible:** There should be a logical reason why it might work (not just a statistical fluke).

Example:
A simple alpha factor is “momentum”—the idea that stocks that have gone up recently tend to keep going up for a while.

3. Types of Financial Features (Alpha Factors)

Feature engineering in finance draws on decades of academic research as well as practical experience from traders. Here are the main types of alpha factors you’ll encounter:

A. Price-Based Features (Technical Indicators)

These are calculated from historical prices and volumes. They try to capture trends, reversals, volatility, and other price behaviors.

Common examples:

- **Moving Averages:** The average price over a set period (e.g., 20 days). Used to spot trends and smooth out noise.
- **Momentum:** Measures the speed and direction of price changes (e.g., return over the last 10 days).
- **Volatility:** How much the price fluctuates (e.g., standard deviation of daily returns over 30 days).
- **Relative Strength Index (RSI):** Indicates if a stock is overbought or oversold.
- **MACD (Moving Average Convergence Divergence):** Captures shifts in momentum.

Why use them?
Many traders believe that price patterns repeat due to human psychology and market structure.

B. Volume-Based Features

These use trading volume data to spot unusual activity or confirm price moves.

Examples:

- **Volume spikes:** Sudden increases in trading volume can signal big news or a change in trend.
- **On-Balance Volume (OBV):** Combines price and volume to show whether money is flowing into or out of a stock.

C. Fundamental Features

These are based on company financials—earnings, book value, debt, and more.

Examples:

- **Earnings Yield:** Earnings per share divided by price.
- **Book-to-Market Ratio:** Book value per share divided by price.
- **Debt-to-Equity Ratio:** How much debt a company has compared to its equity.
- **Profit Margin:** Net income divided by revenue.

Why use them?
Strong fundamentals may indicate undervalued or overvalued stocks.

D. Alternative Data Features

These use non-traditional data sources, such as:

- **Sentiment Scores:** Derived from news headlines, analyst reports, or social media posts using natural language processing (NLP).
- **Event Flags:** Mark days with earnings announcements, product launches, or regulatory changes.
- **Web Search Trends:** Spikes in Google searches for a company can signal rising interest.

4. The Feature Engineering Process: Step by Step

Let’s walk through how you’d create and test new alpha factors for your trading strategy.

Step 1: Generate Ideas

Start with a hypothesis about what might predict returns. This could be based on academic research, market intuition, or observation.

Example hypotheses:

- “Stocks with high momentum over the last month will outperform next month.”
- “Companies with positive earnings surprises will see their stock prices rise.”

Step 2: Calculate the Feature

Use your raw data (prices, volumes, fundamentals, etc.) to compute the feature for each stock and date.

Example:
To calculate 10-day momentum for Apple, subtract the closing price 10 days ago from today’s closing price.

Step 3: Align and Lag the Feature

To avoid look-ahead bias, always use only data that would have been available at the time. Lag your features so that today’s prediction only uses information up to yesterday.

Step 4: Standardize or Normalize

Features are often on different scales (e.g., price in dollars, volume in millions). Standardizing (subtracting the mean and dividing by the standard deviation) or normalizing (scaling between 0 and 1) helps models learn better.

Step 5: Test Predictive Power

Before using a feature in your model, check if it actually predicts returns. This is called **factor testing** or **factor research**.

How to test:

- **Correlation:** Does the feature correlate with future returns?
- **Quantile analysis:** Sort stocks by feature value, group into “buckets,” and see if top buckets outperform bottom buckets.
- **Backtesting:** Simulate a simple strategy based on the feature and see if it makes money.

Step 6: Combine Features

Most successful strategies use several features together. You can combine them in a model, or even create new features by blending existing ones (e.g., “momentum times volatility”).

5. Examples of Classic Alpha Factors

Let’s look at some of the most famous and widely used alpha factors in finance.

A. Momentum

Definition:
Stocks that have performed well in the recent past tend to keep performing well for a while.

How to compute:
Calculate the return over the last 12 months (excluding the most recent month to avoid short-term reversal).

Why it works:
Behavioral finance suggests that investors underreact to news, causing trends to persist.

B. Value

Definition:
Stocks that are cheap relative to fundamentals (like earnings or book value) tend to outperform expensive stocks.

How to compute:
Use ratios like P/E (price-to-earnings), P/B (price-to-book), or earnings yield.

Why it works:
Markets sometimes overreact to bad news, making good companies temporarily cheap.

C. Size

Definition:
Smaller companies (by market capitalization) often outperform larger companies over the long run.

How to compute:
Rank stocks by their total market value.

Why it works:
Small companies may be riskier but also have more growth potential.

D. Low Volatility

Definition:
Stocks with lower price volatility tend to outperform more volatile stocks.

How to compute:
Calculate the standard deviation of daily returns over the past month or year.

Why it works:
Investors often overlook boring, stable stocks, but these can deliver steady returns.

6. Advanced Feature Engineering Techniques

As you gain experience, you can create more sophisticated features using advanced techniques.

A. Rolling Windows

Instead of using a fixed period, you can calculate features over rolling windows (e.g., 20-day moving average, recalculated each day).

Why use them?
Rolling features adapt to changing market conditions and provide up-to-date information.

B. Cross-Sectional Features

Compare a stock's feature value to that of its peers on the same day.

Example:
Is Apple's momentum higher than the average momentum of all tech stocks today? This helps spot relative strength.

C. Feature Transformations

You can apply mathematical functions to features to make them more informative.

Examples:

- Take the log of a ratio to reduce skewness.
- Square a feature to emphasize large values.

D. Feature Interactions

Combine two or more features to capture more complex relationships.

Example:
Multiply momentum by volatility to find stocks with strong and stable trends.

E. Dimensionality Reduction

If you have many features, techniques like Principal Component Analysis (PCA) can help reduce noise and focus on the most important information.

7. Evaluating and Selecting Features

Not all features are useful. You need to evaluate and select the best ones for your model.

A. Avoiding Overfitting

If you use too many features, your model may “memorize” the training data and fail on new data. This is called overfitting.

How to avoid:

- Use only features that make economic sense.
- Test features on out-of-sample data (data the model hasn’t seen before).
- Use regularization techniques in your models to penalize complexity.

B. Feature Importance

Some machine learning models (like random forests) can tell you which features are most important for predictions.

How to use:
After training your model, look at the feature importances and consider dropping features that contribute little.

C. Correlation and Redundancy

Highly correlated features (like 10-day and 11-day moving averages) don’t add much new information. Try to use features that are as independent as possible.

8. Practical Example: Building and Testing a Simple Alpha Factor

Check colab link

Step 1: Get Data

Step 2: Calculate a Feature

Compute the 10-day momentum:

python

```
import pandas as pd

import yfinance as yf



data = yf.download('RELIANCE.NS', start='2019-01-01', end='2025-04-01')

data['Momentum_10'] = data['Close'] - data['Close'].shift(10)
```

Step 3: Lag the Feature

Shift the feature so that today's prediction only uses information up to yesterday:

Step 4: Standardize

Subtract the mean and divide by the standard deviation:

Step 5: Test Predictive Power

Check if high momentum predicts higher returns:

If the correlation is positive, your feature has some predictive power!

9. Common Pitfalls in Feature Engineering

A. Look-Ahead Bias

Never use future information when creating features. Always lag features so they only use data available at the prediction time.

B. Data Snooping

Testing too many features on the same data can lead to false discoveries. Always validate your findings on new, unseen data.

C. Overfitting

Using too many or too complex features can cause your model to fit noise instead of real patterns. Keep things as simple as possible.

D. Ignoring Transaction Costs

A feature may look great in backtests but fail in real trading if it leads to too many trades (and thus high costs).

10. Combining Features: Multi-Factor Models

Most professional strategies use several alpha factors together. You can combine them in a linear model, a decision tree, or a more advanced ML algorithm.

Example:

- Use momentum, value, and volatility features together.
- Train a model to predict next-day returns based on all features.

Why combine?
Different features capture different aspects of market behavior. Combining them can make your model more robust and less sensitive to changing conditions.

11. Feature Engineering for Machine Learning Models

When using ML models like random forests, gradient boosting, or neural networks, feature engineering is even more important.

- **ML models can find complex patterns,** but only if you give them informative features.
- **Feature selection and importance:** Use model outputs to refine your features.
- **Iterative process:** You'll often go back and create new features based on what your model learns.

12. Key Takeaways

- **Feature engineering is the heart of ML for trading:** The quality of your features usually matters more than the complexity of your model.
- **Alpha factors should be predictive, robust, and make economic sense.**
- **Test features carefully:** Use correlation, quantile analysis, and backtesting.
- **Avoid common pitfalls:** Look-ahead bias, data snooping, and overfitting can ruin results.
- **Combine features for stronger models:** Multi-factor strategies are more robust.
- **Keep learning:** Feature engineering is as much an art as a science. The best traders are always searching for new, better alpha factors.