

Lesson 24: Factor Models

Data Science with Python – BSc Course

45 Minutes

The Problem: CAPM uses only market beta, but stocks also respond to size, value, and momentum. How do we capture multiple sources of systematic risk?

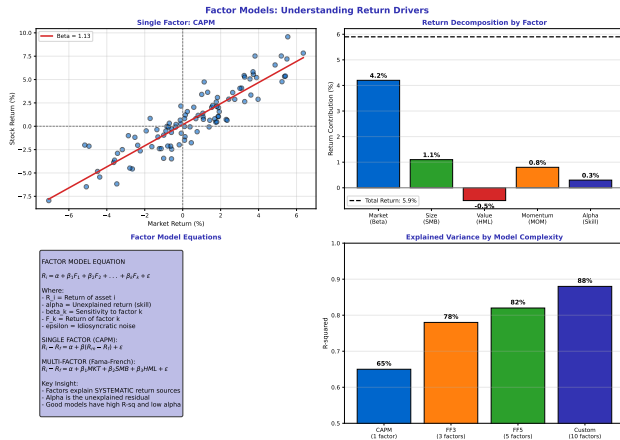
After this lesson, you will be able to:

- Build multi-factor regression models
- Understand Fama-French factors (SMB, HML)
- Interpret factor loadings and alpha
- Create complete ML pipelines with sklearn

Finance Application: Decomposing returns into systematic factors for attribution

From Single to Multiple Risk Sources

- CAPM: $R_i - R_f = \alpha + \beta_M(R_M - R_f)$ – only market risk
- Multi-factor: Add size, value, momentum as additional explanatory variables



Key insight: Different stocks have different exposures to different risk factors

The Classic Three-Factor Model

- **SMB** (Small Minus Big): Small caps outperform large caps historically
- **HML** (High Minus Low): Value stocks outperform growth stocks

Factor Definitions

FAMA-FRENCH THREE-FACTOR MODEL (1993)

- MKT (Market)**
 - Market return minus risk-free rate
 - $R_M - R_f$
 - Captures broad market exposure
- SMB (Small Minus Big)**
 - Small cap return minus large cap return
 - Size effect: small stocks outperform
 - Based on market capitalization
- HML (High Minus Low)**
 - Value stocks minus growth stocks
 - Value effect: high B/M outperform
 - Based on Book-to-Market ratio

FAMA-FRENCH FIVE-FACTOR MODEL (2015)

- RMW (Robust Minus Weak)**
 - Profitable firms minus unprofitable
 - Profitability effect
- OMA (Conservative Minus Aggressive)**
 - Low investment minus high investment
 - Investment effect

Factor Statistics

FACTOR STATISTICS (Monthly)

Factor	Mean	Std	Sharpe (Ann.)
MKT	0.44%	4.15%	0.37
SMB	0.17%	3.02%	0.43
HML	0.20%	3.17%	0.31

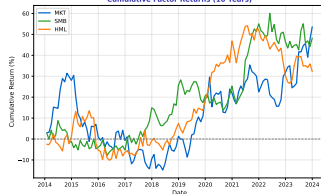
CORRELATION MATRIX

	MKT	SMB	HML
MKT	1.00	0.10	-0.11
SMB	0.10	1.00	0.13
HML	-0.11	0.13	1.00

Key: Low correlations = good diversification

Fama-French Factor Model

Cumulative Factor Returns (10 Years)



Python Implementation

```

Loading Fama-French data with pandas
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression

# Load Fama-French factors (from Ken French website)
# Format: Date, MKT-RF, SMB, HML, RF (in %)
ff_data = pd.read_csv('F:\Research\ff_data\Factors.csv',
                      skiprows=1)
ff_data['Date'] = pd.to_datetime(ff_data['Date'],
                                format='%m/%d/%Y')

# Load stock returns
stock_returns = pd.read_csv('stock_returns.csv')

# Merge and prepare for regression
merged = pd.merge(stock_returns, ff_data, on='Date')
merged['excess_return'] = merged['return'] - merged['RF']

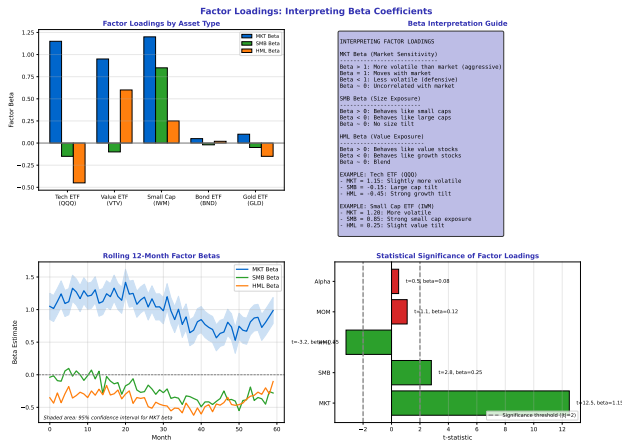
# Fit F3 model
X = merged[['MKT-RF', 'SMB', 'HML']]
y = merged['excess_return']
model = LinearRegression()
model.fit(X, y)

print(f'Alpha: {model.intercept_: .4f}')
print(f'MKT beta: {model.coef_[0]: .3f}')
print(f'SMB beta: {model.coef_[1]: .3f}')
print(f'HML beta: {model.coef_[2]: .3f}')
    
```

Nobel Prize (2013): Fama showed these factors explain returns better than CAPM alone

How Much Exposure Does a Stock Have?

- Each stock has different sensitivities to each factor
- Loading = regression coefficient on that factor

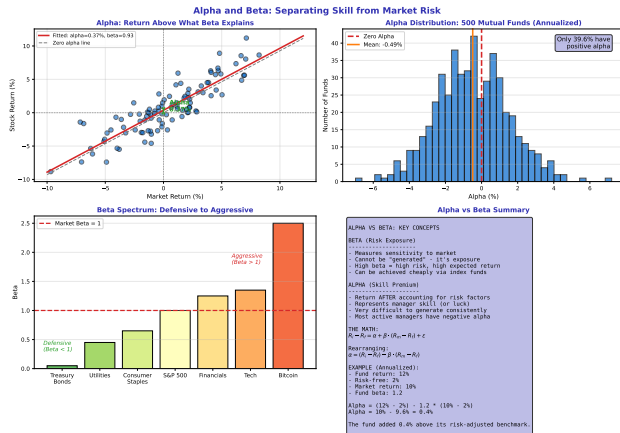


Example: TSLA has high market beta but negative HML (growth stock, not value)

Alpha and Beta Decomposition

Separating Skill from Risk Exposure

- α : Return not explained by factors (manager skill or mispricing)
- Multi-factor alpha is “purer” than CAPM alpha



Industry standard: Report alpha after controlling for Fama-French factors

Multi-Factor Regression

Implementation in Python

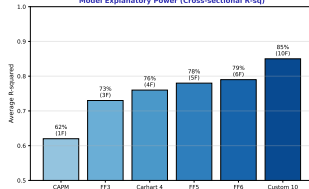
- Same sklearn API: `LinearRegression().fit(X, y)` where X has multiple columns
- Each coefficient is a factor loading

Multi-Factor Models: Comprehensive Risk Decomposition

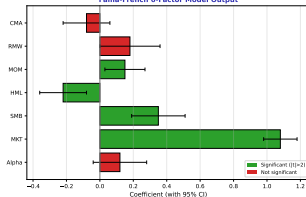
The Factor Zoo

THE FACTOR ZOO	
Academic Factors (Well-Established)	
Market (MKT)	: Equity premium
Size (SMB)	: Small cap premium
Value (HML)	: Value vs growth
Momentum (MOM)	: Winners keep winning
Profitability	: Profitable > unprofitable
Investment	: Conservative > aggressive
Quality	: Low debt, stable earnings
Industry Factors (Practitioner)	
BAB	: Betting Against Beta
QMJ	: Quality Minus Junk
LID	: Liquidity risk premium
VOL	: Volatility factor
TERM	: Term structure
CREDIT	: Credit spread
THE PROBLEM: Factor proliferation	
- 400+ published factors	
- Many don't replicate	
- Data mining concerns	
- Use skeptically!	

Model Explanatory Power (Cross-sectional R-sq)



Fama-French 6-Factor Model Output



Python Implementation

```
Multi-Factor Model in sklearn
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import statsmodels.api as sm

# Prepare factor data
factors = ['MKT', 'SMB', 'HML', 'MOM', 'RMW', 'CMA']
X = ff_data[factors]
y = stock_returns

# Option 1: sklearn (simple)
model = LinearRegression()
model.fit(X, y)
print('Alpha: {model.intercept:.4f}')
print('Beta: {dict(zip(factors, model.coef_))}')
print('R-squared: {r2_score(y, model.predict(X)):.4f}')

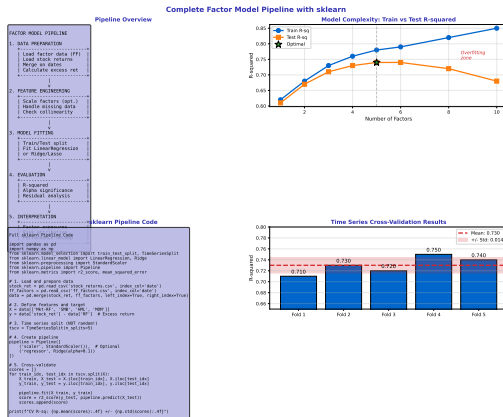
# Option 2: statsmodels (with t-stats)
X_sm = sm.add_constant(X) # Add intercept
model_sm = sm.OLS(y, X_sm).fit()
print(model_sm.summary())

# Key outputs from statsmodels:
# - Coefficients with standard errors
# - t-statistics and p-values
# - R-squared and Adjusted R-squared
# - F-statistic for overall significance
```

X matrix columns: [Mkt-RF, SMB, HML] – intercept is alpha

Combining Preprocessing and Modeling

- Pipeline([('scaler', StandardScaler()), ('reg', Ridge())])
- Prevents data leakage: scaler fits only on training data



Best practice: Always use pipelines for reproducible workflows

Saving and Loading Trained Models

- `joblib.dump(model, 'model.pkl')` – save to disk
- `model = joblib.load('model.pkl')` – reload for production

Model Persistence: Save, Load, and Deploy

Why Model Persistence?

WHY SAVE MODELS?

1. **REPRODUCIBILITY**
 - Same predictions every time
 - Audit trail for compliance
 - Version control models
2. **DEPLOYMENT**
 - Train once, predict many times
 - No need to re-fit on server
 - Faster predictions
3. **SHARING**
 - Share models with team
 - Model registry (MFlow)
 - API serving

COMMON FORMATS:

pickle / joblib

- EASY, ONLY
- Works with sklearn
- Python version sensitive
- Security risk (arbitrary code)

ONNX

- Framework agnostic
- Production ready
- More complex setup

JSON (coefficients only)

- EASIEST
- HUMAN READABLE

pickle & joblib Code

```
Save and Loading sklearn Models
import pickle
import joblib
from sklearn.linear_model import Ridge

# =====
# METHOD 1: pickle
# =====
# Save
with open('model.pkl', 'wb') as f:
    pickle.dump(model, f)

# Load
with open('model.pkl', 'rb') as f:
    loaded_model = pickle.load(f)

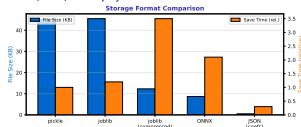
# =====
# METHOD 2: joblib (recommended for sklearn)
# =====
# Save (uncompressed)
joblib.dump(model, 'model.joblib')

# Save (compressed - smaller files)
joblib.dump(model, 'model.joblib.gz', compress=3)

# Load
loaded_model = joblib.load('model.joblib')

# =====
# Use loaded model
# =====
predictions = loaded_model.predict(X_new)

# Verify it's the same
assert np.allclose(model.coef_, loaded_model.coef_)
```



Best Practices

MODEL PERSISTENCE BEST PRACTICES

1. **VERSION EVERYTHING**
 - model v1.0.0 2024-01-15.joblib
 - Include version number
 - Include training date
 - Track with git/DVC
2. **SAVE METADATA**

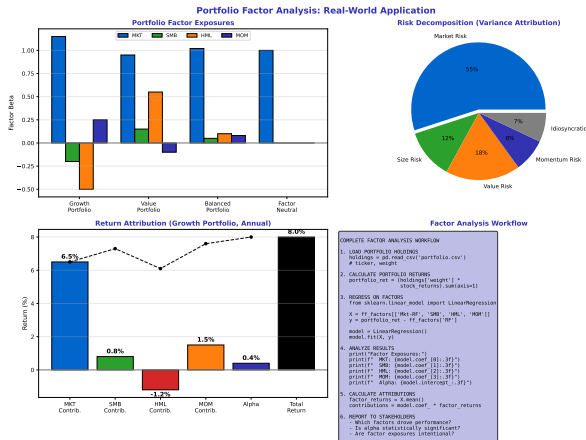
```
{
  "model_type": "Ridge",
  "features": ["WGT", "SHR", "HMR"],
  "train_r2": 0.72,
  "train_data": "2024-01-15",
  "sklearn_version": "1.3.0",
  "python_version": "3.10.12"
}
```
3. **SAVE PREPROCESSING TOO**
 - StandardScaler parameters
 - Feature names and order
 - Or use Pipeline (saves everything)
4. **SECURITY**
 - Never load untrusted pickles!
 - Pickle can execute arbitrary code
 - Use joblib with trusted sources only
5. **TESTING**

```
# After loading
assert model.feature_names_in_ == expected_features
assert model.predict(X_test[1]).shape == (1,)
```

Deployment: Train once, deploy saved model to production

Where Did Your Returns Come From?

- Decompose portfolio returns into factor contributions
- Shows whether performance came from market timing, factor bets, or alpha



Risk management: Understand your factor exposures before they surprise you

Hands-On Exercise (25 min)

Task: Build a Fama-French Factor Model

- 1 Download Fama-French 3-factor data from Kenneth French's website
- 2 Merge with your stock returns (AAPL or similar)
- 3 Fit multi-factor regression: stock returns vs [Mkt-RF, SMB, HML]
- 4 Interpret: What are the factor loadings? Is there alpha?
- 5 Compare R^2 to single-factor CAPM model

Deliverable: Table of factor loadings + comparison of R^2 values.

Extension: Add momentum (UMD) as a fourth factor – does R^2 improve?

Problem Solved: We can now decompose stock returns into multiple systematic factors, giving better risk attribution than CAPM alone.

Key Takeaways:

- Fama-French: Market + SMB (size) + HML (value)
- Factor loadings = regression coefficients on each factor
- Alpha after factors = true outperformance
- sklearn pipelines ensure reproducible workflows

Next Lesson: Classification (L25) – predicting categories instead of numbers

Memory: SMB = Small Minus Big (size), HML = High Minus Low (value)