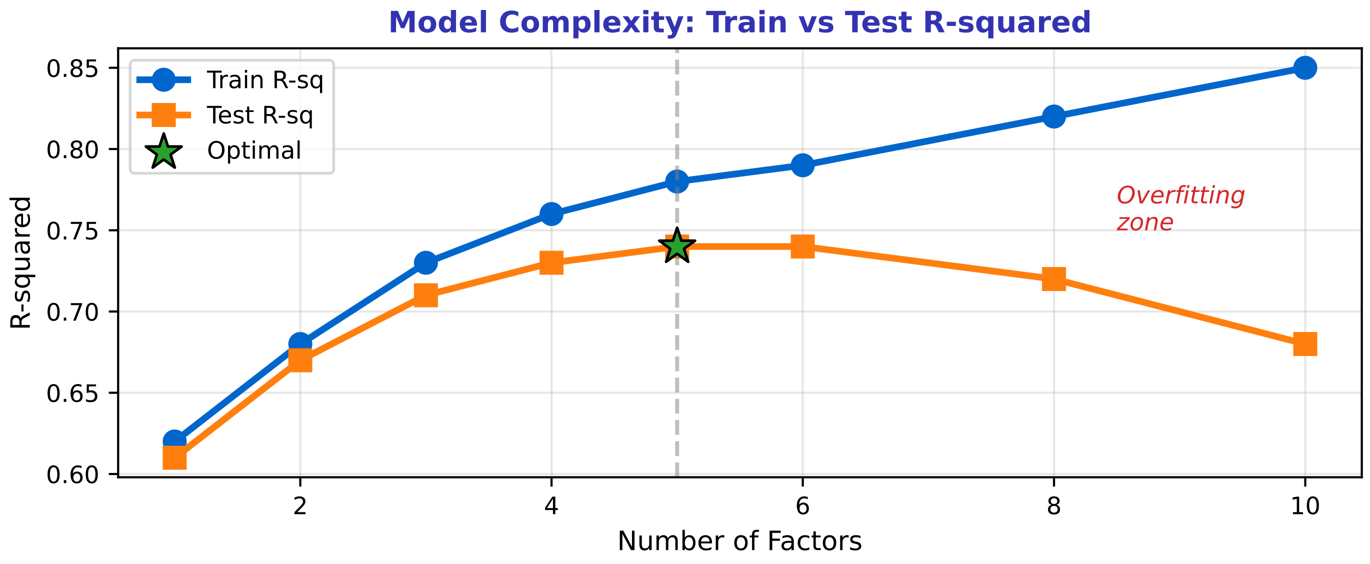
Complete Factor Model Pipeline with sklearn

Pipeline Overview

```
FACTOR MODEL PIPELINE

1. DATA PREPARATION
   +------------------------+
   | Load factor data (FF)  |
   | Load stock returns     |
   | Merge on dates         |
   | Calculate excess ret   |
   +------------------------+
              |
              v
2. FEATURE ENGINEERING
   +------------------------+
   | Scale factors (opt.)   |
   | Handle missing data    |
   | Check collinearity     |
   +------------------------+
              |
              v
3. MODEL FITTING
   +------------------------+
   | Train/Test split       |
   | Fit LinearRegression   |
   | or Ridge/Lasso         |
   +------------------------+
              |
              v
4. EVALUATION
   +------------------------+
   | R-squared              |
   | Alpha significance     |
   | Residual analysis      |
   +------------------------+
              |
              v
5. INTERPRETATION
   +------------------------+
   | Factor exposures       |
   | Risk attribution       |
   | Performance decomp     |
   +------------------------+
```

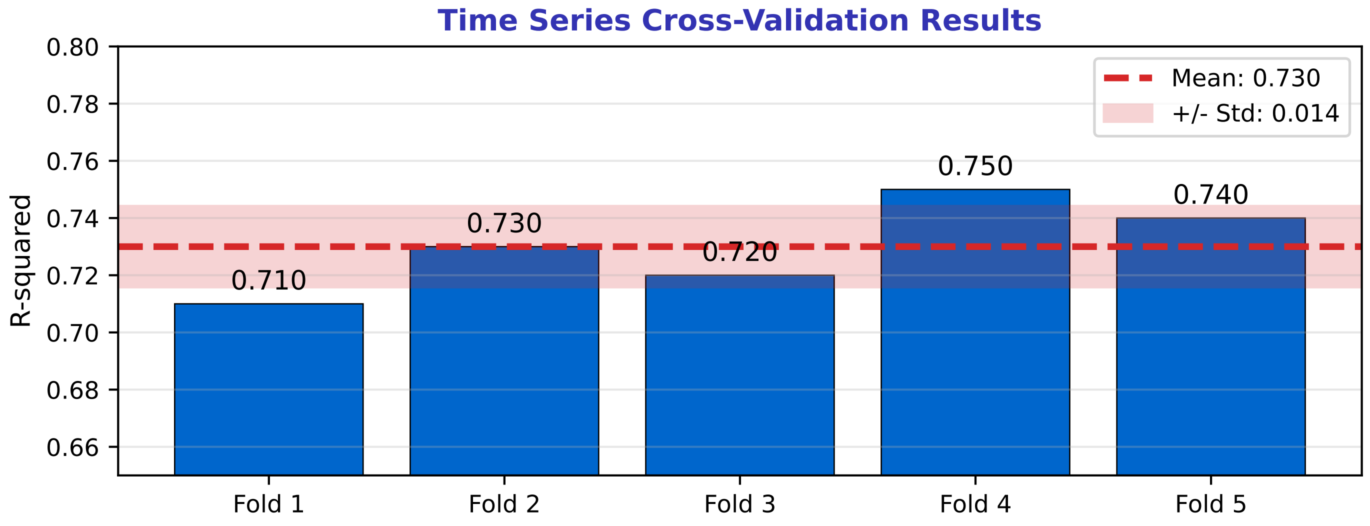
Model Complexity: Train vs Test R-squared



sklearn Pipeline Code

```
Full sklearn Pipeline Code

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, TimeSeriesSplit
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import r2_score, mean_squared_error

# 1. Load and prepare data
stock_ret = pd.read_csv('stock_returns.csv', index_col='date')
ff_factors = pd.read_csv('ff_factors.csv', index_col='date')
data = pd.merge(stock_ret, ff_factors, left_index=True, right_index=True)

# 2. Define features and target
X = data[['Mkt-RF', 'SMB', 'HML', 'MOM']]
y = data['stock_ret'] - data['RF']  # Excess return

# 3. Time series split (NOT random!)
tscv = TimeSeriesSplit(n_splits=5)

# 4. Create pipeline
pipeline = Pipeline([
    ('scaler', StandardScaler()),  # Optional
    ('regressor', Ridge(alpha=0.1))
])

# 5. Cross-validate
scores = []
for train_idx, test_idx in tscv.split(X):
    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

    pipeline.fit(X_train, y_train)
    score = r2_score(y_test, pipeline.predict(X_test))
    scores.append(score)

print(f"CV R-sq: {np.mean(scores):.4f} +/- {np.std(scores):.4f}")
```

Time Series Cross-Validation Results


Time Series Cross-Validation Results
Mean: 0.730
+/- Std: 0.014
R-squared
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5