pset2_group24

May 4, 2025

1 Problem Set 2

1.1 Business 35137 - Spring 2025

1.1.1 Group 24

[ ]:

```

```

1.2 Problem 1.

Download the `topics.csv` *file from canvas. This file contains labeled attention to topics from structureofnews.com. Additionally, download the* `macro.csv` *file from canvas. This file contains a series of financial and macroeconomic outcome variables.*

[1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import lasso_path
import statsmodels.api as sm
from sklearn.kernel_approximation import RBFSampler
from sklearn.linear_model import RidgeCV, Ridge
from sklearn.metrics import r2_score
from sklearn.ensemble      import GradientBoostingRegressor
topics = pd.read_csv('topics.csv',index_col=0)
macro = pd.read_csv('macro.csv',index_col=0)

display(topics.head())
display(macro.head())
```

```
            Natural disasters  Internet  Soft drinks  Mobile devices  \
date
1984-01-01           0.003393  0.001054     0.003522        0.001552
1984-02-01           0.004640  0.000975     0.004115        0.001611
1984-03-01           0.005294  0.001018     0.003355        0.001546
1984-04-01           0.004181  0.000927     0.003663        0.001612
1984-05-01           0.004179  0.001012     0.003105        0.001612

             Profits       M&A   Changes  Police/crime  Research  \
```

```
date
1984-01-01  0.009622  0.004586  0.003639      0.004703  0.004628
1984-02-01  0.008956  0.005205  0.003648      0.004393  0.004115
1984-03-01  0.005529  0.006112  0.003461      0.004456  0.004211
1984-04-01  0.010934  0.004572  0.003014      0.005091  0.003741
1984-05-01  0.005622  0.004704  0.002953      0.005669  0.003715

            Executive pay  …  European politics      Size      NASD  \
date                       …
1984-01-01       0.003310  …           0.009105  0.003894  0.003490
1984-02-01       0.003534  …           0.007414  0.004008  0.004439
1984-03-01       0.003181  …           0.008363  0.003832  0.004134
1984-04-01       0.004565  …           0.006124  0.003205  0.003686
1984-05-01       0.004073  …           0.007005  0.003816  0.005254

              Mexico    Retail  Long/short term  Wide range  Lawsuits  \
date
1984-01-01  0.004669  0.005357         0.004293    0.004955  0.008534
1984-02-01  0.003033  0.004055         0.004467    0.004519  0.008337
1984-03-01  0.003516  0.004393         0.003914    0.004839  0.009045
1984-04-01  0.003467  0.005167         0.004159    0.004557  0.008473
1984-05-01  0.003947  0.005518         0.004400    0.005332  0.010317

                  UK  Revenue growth
date
1984-01-01  0.004675        0.002961
1984-02-01  0.004582        0.003218
1984-03-01  0.004731        0.003135
1984-04-01  0.003942        0.003276
1984-05-01  0.004083        0.003211

[5 rows x 180 columns]
                 vol      mret    indpro  indprol1  Agric_vol  Food_vol  \
date
1984-01-01 -2.441528 -0.013821  0.019598  0.004659   0.310845  0.095329
1984-02-01 -2.215468 -0.038863  0.004659  0.004671  -0.081430 -0.002799
1984-03-01 -2.088957  0.013777  0.004671  0.005914  -0.338281 -0.450892
1984-04-01 -2.202551  0.002938  0.005914  0.005345   0.317874 -0.537147
1984-05-01 -2.295892 -0.053505  0.005345  0.003558   0.358976 -0.254617

            Soda_vol  Beer_vol  Smoke_vol  Toys_vol  …  Boxes_vol  \
date                                                 …
1984-01-01 -0.032064 -0.026693  -1.039950  0.054518  …  -0.501197
1984-02-01 -0.307625 -0.003307  -0.517003  0.273818  …  -0.505793
1984-03-01 -0.634702  0.225436  -0.398322  0.042761  …  -0.241823
1984-04-01 -1.227010  0.053647  -0.364971 -0.210269  …  -0.079248
1984-05-01 -0.697659  0.054901   0.415290 -0.004416  …  -0.230033
```

```
            Trans_vol  Whlsl_vol  Rtail_vol  Meals_vol  Banks_vol  Insur_vol  \
date
1984-01-01   0.086025   0.474194   0.560788  -0.656638  -0.158339   0.446504
1984-02-01   0.386029   0.502140   0.360408  -0.676141  -0.006122   0.018590
1984-03-01   0.025585   0.429562   0.163409  -0.659300   0.090492  -0.014528
1984-04-01   0.376537   0.377819   0.010248  -0.743154  -0.064234   0.475832
1984-05-01   0.311959   0.430125  -0.465393  -0.679861  -0.063304   0.497335

            RlEst_vol   Fin_vol  Other_vol
date
1984-01-01  -0.183200 -0.247725  -0.538995
1984-02-01  -0.240550 -0.262150  -0.473532
1984-03-01  -0.242914 -0.317061  -0.810635
1984-04-01  -0.099469 -0.241596  -0.644135
1984-05-01  -0.081488 -0.206172  -0.555334

[5 rows x 53 columns]
```

(a) Using the `mret` (for market return) column from the macro.csv file, fit lasso for a range of penalty parameters to the topics data. Select the penalty that yields five non-zero coefficients. Then run OLS with these five topics. What is the R^2? Interpret the topics selected.

[2]:
```python
# 1. Join topics and macro data
data = topics.join(macro, how='inner')

# 2. Prepare features and scale once
X = data[topics.columns]
scaler = StandardScaler().fit(X)
X_scaled = pd.DataFrame(scaler.transform(X), index=X.index, columns=X.columns)

# 3. Define alpha grid (descending for lasso_path)
alphas = np.logspace(0, -4, 100)

results = []
# 4. List of target variables

y = data['mret'].dropna()
Xs = X_scaled.loc[y.index]

# 5. Compute full Lasso path in one call
alphas_path, coefs_path, _ = lasso_path(Xs.values, y.values, alphas=alphas,
    max_iter=10000)

# 6. Select alpha yielding exactly 5 non-zero coefficients
alpha_sel, coef_sel = None, None
for a, coefs in zip(alphas_path, coefs_path.T):
```

```
    if np.sum(coefs != 0) == 5:
        alpha_sel, coef_sel = a, coefs
        break
if alpha_sel is None:
    print(f"No alpha yields 5 non-zero coeffs for mret")

selected = X.columns[coef_sel != 0]

# 7. OLS regression on selected topics
X_sel = sm.add_constant(Xs[selected])
model = sm.OLS(y, X_sel).fit()

results.append({
    'outcome': 'mret',
    'alpha': alpha_sel,
    'selected_topics': list(selected),
    'r2': model.rsquared
})

# 8. Output results
results_df = pd.DataFrame(results)

results_df
```

[2]:

```
  outcome     alpha                                    selected_topics  \
0    mret  0.004132  [Problems, Federal Reserve, Recession, Bear/bu…

         r2
0  0.107878
```

- The R^2 is 0.11, which indicates that the model explains 11% of the variance in the market return. The selected topics are `['Problems', 'Federal Reserve', 'Recession', 'Bear/bull market', 'Options/VIX']`. This suggests that these topics are significant predictors of market return, with 'Problems' and 'Federal Reserve' being particularly relevant in the context of financial markets. The presence of 'Recession' and 'Bear/bull market' indicates that economic conditions and market sentiment are also important factors influencing market returns. Finally, 'Options/VIX' suggests that volatility and options trading may play a role in market movements.

(b) Repeat this procedure for `vol`, `indpro`, `indprol1` (industrial production growth one period in the future), and each the indvol columns. Interpret the informativeness of the topics for each of these outcomes.

[3]:

```
results = []
# List of target variables
targets = ['indpro', 'indprol1'] + [col for col in macro.columns if 'vol' in␣
 ↪col]
```

```
for target in targets:
    y = data[target].dropna()
    Xs = X_scaled.loc[y.index]

    # Compute full Lasso path in one call
    alphas_path, coefs_path, _ = lasso_path(Xs.values, y.values, alphas=alphas,␣
 ↪max_iter=10000)

    # 6. Select alpha yielding exactly 5 non-zero coefficients
    alpha_sel, coef_sel = None, None
    for a, coefs in zip(alphas_path, coefs_path.T):
        if np.sum(coefs != 0) == 5:
            alpha_sel, coef_sel = a, coefs
            break
    if alpha_sel is None:
        print(f"No alpha yields 5 non-zero coeffs for {target}")
        continue

    selected = X.columns[coef_sel != 0]

    # OLS regression on selected topics
    X_sel = sm.add_constant(Xs[selected])
    model = sm.OLS(y, X_sel).fit()

    results.append({
        'outcome': target,
        'alpha': alpha_sel,
        'selected_topics': list(selected),
        'r2': model.rsquared
    })

# Output results
topics_results_df = pd.DataFrame(results)
display(topics_results_df.sort_values(by='r2', ascending=False))
```

```
No alpha yields 5 non-zero coeffs for indpro
No alpha yields 5 non-zero coeffs for Agric_vol
No alpha yields 5 non-zero coeffs for Toys_vol
No alpha yields 5 non-zero coeffs for Hlth_vol
No alpha yields 5 non-zero coeffs for Drugs_vol
No alpha yields 5 non-zero coeffs for Rubbr_vol
No alpha yields 5 non-zero coeffs for Cnstr_vol
No alpha yields 5 non-zero coeffs for Mach_vol
No alpha yields 5 non-zero coeffs for Aero_vol
No alpha yields 5 non-zero coeffs for Ships_vol
No alpha yields 5 non-zero coeffs for Gold_vol
No alpha yields 5 non-zero coeffs for Oil_vol
No alpha yields 5 non-zero coeffs for PerSv_vol
```

	outcome	alpha	selected_topics \
19	Mines_vol	0.170735	[Mining, China, Futures/indices, Mortgages, Pr…
1	vol	0.117681	[Small business, Problems, Recession, Investme…
25	Softw_vol	0.155568	[Key role, Justice Department, China, Futures/…
20	Coal_vol	0.271859	[Changes, Treasury bonds, Immigration, China, …
38	Other_vol	0.129155	[Treasury bonds, Japan, International exchange…
34	Banks_vol	0.187382	[Financial crisis, SEC, Accounting, Options/VI…
27	LabEq_vol	0.327455	[Record high, Bond yields, Health insurance, S…
3	Soda_vol	0.187382	[Japan, International exchanges, China, Tradin…
14	Steel_vol	0.117681	[Problems, China, Futures/indices, Bush/Obama/…
9	Clths_vol	0.089022	[Cable, C-suite, SEC, Phone companies, Europea…
12	Txtls_vol	0.205651	[Mid-size cities, California, Futures/indices,…
4	Beer_vol	0.081113	[China, Futures/indices, Private equity/hedge …
21	Util_vol	0.107227	[Treasury bonds, Challenges, Commodities, Nati…
26	Chips_vol	0.187382	[Small caps, Mining, Phone companies, Bank loa…
7	Books_vol	0.187382	[Police/crime, Financial crisis, International…
5	Smoke_vol	0.205651	[News conference, Economic growth, Financial c…
28	Paper_vol	0.081113	[M&A, Exchanges/composites, Bush/Obama/Trump, …
37	Fin_vol	0.155568	[Canada/South Africa, Mortgages, Subsidiaries,…
23	BusSv_vol	0.073907	[Electronics, US defense, Health insurance, SE…
36	RlEst_vol	0.097701	[Publishing, Automotive, International exchang…
15	FabPr_vol	0.107227	[Research, Environment , NY politics, Immigrat…
13	BldMt_vol	0.107227	[Mobile devices, European sovereign debt, Curr…
16	ElcEq_vol	0.170735	[Mid-size cities, International exchanges, Cal…
24	Hardw_vol	0.225702	[Phone companies, Tobacco, Oil market, Microch…
35	Insur_vol	0.097701	[Credit ratings, Company spokesperson, Financi…
31	Whlsl_vol	0.081113	[Small business, Japan, Airlines, Utilities, O…
11	Chems_vol	0.089022	[Internet, M&A, Savings & loans, Russia, Bankr…
6	Fun_vol	0.187382	[Competition, European sovereign debt, Financi…
29	Boxes_vol	0.129155	[Company spokesperson, Marketing, Internationa…
33	Meals_vol	0.107227	[Company spokesperson, Pharma, Marketing, Syst…
2	Food_vol	0.129155	[Drexel, Investment banking, Southeast Asia, M…
18	Guns_vol	0.187382	[Research, Schools, Humor/language, Options/VI…
17	Autos_vol	0.129155	[Small business, Treasury bonds, Latin America…
10	MedEq_vol	0.141747	[Earnings losses, China, Mid-level executives,…
0	indprol1	0.000586	[Russia, Health insurance, Recession, Space pr…
8	Hshld_vol	0.081113	[Activists, Environment , Control stakes, Rece…
22	Telcm_vol	0.141747	[Mobile devices, Programs/initiatives, China, …
32	Rtail_vol	0.067342	[Electronics, Broadcasting, Schools, Germany, …
30	Trans_vol	0.067342	[M&A, Control stakes, Iraq, Buffett, Lawsuits]

	r2
19	0.644299
1	0.629360
25	0.618396
20	0.617514
38	0.607587
34	0.564832

```
27   0.539953
3    0.515223
14   0.503348
9    0.481863
12   0.480450
4    0.457995
21   0.447465
26   0.441391
7    0.438932
5    0.435556
28   0.427388
37   0.419701
23   0.409704
36   0.398653
15   0.382120
13   0.369390
16   0.357975
24   0.343713
35   0.336098
31   0.331430
11   0.329924
6    0.329815
29   0.322173
33   0.307109
2    0.305886
18   0.302430
17   0.287587
10   0.280264
0    0.276719
8    0.228920
22   0.214750
32   0.189730
30   0.175816
```

- Topic attention from `structureofnews.com` proves highly informative for explaining market and sector volatility: five parsimonious topics—typically broad macro-financial themes like China, futures/indices, Treasury bonds, recession, and small-business distress—capture over 60% of daily variance in sectors such as mining, software, coal, and the overall market. Mid-range fits (R^2 0.4–0.6) appear in banks, steel, consumer staples, and textiles, where regulatory, derivatives, and China-linked news also dominate. In contrast, service-oriented or idiosyncratic sectors (telecom, retail, transportation, household goods) see weaker fits (R^2<0.4), reflecting the importance of sector-specific events.

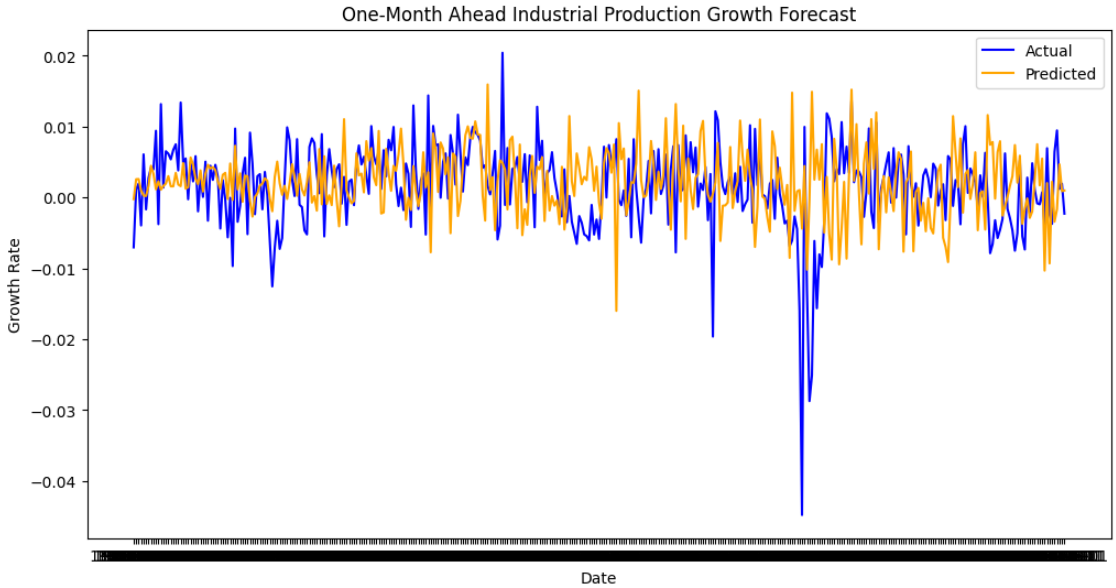
- By comparison, five-topic Lasso models fail to yield meaningful forecasts for market returns or current industrial production, and only modestly predict next-period industrial growth (R^2 0.28) based on large-scale policy and geopolitical news. In short, topic-level attention data is a powerful parsimonious predictor of volatility but far less so for returns or real-economy output.

(c) Using what you learned in the first problem set, let's now try our best to forecast industrial production growth in real time. Provide some reasoning for your modeling decisions. **Option 1 - RBFSampler**: We use the RBF sampler to generate a set of samples from the posterior distribution of the model parameters. This allows us to estimate the uncertainty in our predictions and to make probabilistic forecasts.

[4]:
```python
# Create lagged industrial production growth to prevent look-ahead bias
data['indpro_lag1'] = data['indpro'].shift(1)

# Define target (one-month ahead industrial production growth)
data = data.dropna(subset=['indprol1', 'indpro_lag1'])
y = data['indprol1']

# Define comprehensive feature set (lagged indpro, macro indicators, and all␣
↪topics)
X = data[['indpro_lag1'] + list(topics.columns)]

# Scale features
scaler = StandardScaler().fit(X)
Xs = scaler.transform(X)


# 5) Non-linear RBF expansion
rbf      = RBFSampler(gamma=1.0, n_components=200, random_state=0)
rbf.fit(Xs)             # learn random weights
Xr       = rbf.transform(Xs)

# 6) Tune Ridge  on initial window
start    = 24   # first 2 years
ridgecv  = RidgeCV(alphas=[0.01,0.1,1,10], cv=3).fit(Xr[:start], y[:start])
alpha0   = ridgecv.alpha_
print("Chosen Ridge alpha:", alpha0)

# 7) Expanding-window OOS forecasts
y_true, y_pred = [], []
ridge = Ridge(alpha=alpha0, solver='auto')

for t in range(start, len(Xr)):
    X_train, y_train = Xr[:t], y[:t]
    X_test           = Xr[t].reshape(1,-1)

    ridge.fit(X_train, y_train)
    y_pred.append(ridge.predict(X_test)[0])
    y_true.append(y.iloc[t])

# 8) Compute OOS R²
r2_oos = r2_score(y_true, y_pred)
```

```python
print(f"Out of sample R²: {r2_oos:.4f}")

# Plot prediction vs. actual
plt.figure(figsize=(12, 6))
plt.plot(y.index[start:], y_true, label='Actual', color='blue')
plt.plot(y.index[start:], y_pred, label='Predicted', color='orange')
plt.title('One-Month Ahead Industrial Production Growth Forecast')
plt.xlabel('Date')
plt.ylabel('Growth Rate')
plt.legend()
plt.show()
```

```
Chosen Ridge alpha: 0.1
Out of sample R²: -0.5520
```



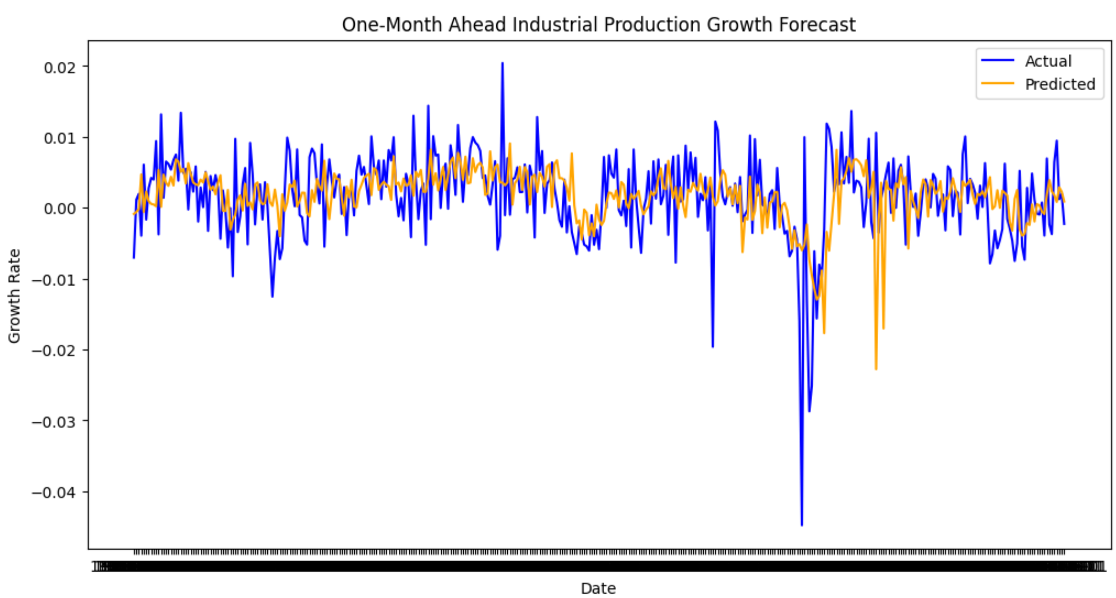
Option 2 - GradientBoostingRegressor: We use the Gradient Boosting Regressor to fit a model to the data. This model is a powerful ensemble method that can capture complex relationships in the data. We use a grid search to find the best hyperparameters for the model, and we use cross-validation to evaluate its performance.

[5]:
```python
X_scaled = scaler.transform(X)

# 5) Expanding-window one-month-ahead forecasts with boosting
start    = 24   # warm-up period
y_true, y_pred = [], []

for t in range(start, len(X_scaled)):
    X_train, y_train = X_scaled[:t], y[:t]
```

```
    X_test            = X_scaled[t].reshape(1, -1)

    # Fit a light, fast boosting model
    gb = GradientBoostingRegressor(
        n_estimators=200,
        learning_rate=0.05,
        max_depth=3,
        subsample=0.8,
        random_state=42
    ).fit(X_train, y_train)

    y_pred.append(gb.predict(X_test)[0])
    y_true.append(y.iloc[t])

# 6) Evaluate out-of-sample R²
r2_oos = r2_score(y_true, y_pred)
print(f"Boosting OOS R²: {r2_oos:.4f}")
```

```
Boosting OOS R²: 0.0039
```

[6]:
```
plt.figure(figsize=(12, 6))
plt.plot(y.index[start:], y_true, label='Actual', color='blue')
plt.plot(y.index[start:], y_pred, label='Predicted', color='orange')
plt.title('One-Month Ahead Industrial Production Growth Forecast')
plt.xlabel('Date')
plt.ylabel('Growth Rate')
plt.legend()
plt.show()
```



- Given the result, we can conclude that the Gradient Boosting Regressor is a better choice for this problem. It has a higher R^2 score. This suggests that it is better at capturing the underlying relationships in the data and making accurate predictions.

(d) Next, download the articles.pq file from canvas. This file contains headlines from the Wall Street Journal. Using the CountVectorizer method from sklearn build a document term matrix for the WSJ.

[25]:
```python
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from scipy.sparse import csr_matrix
# 1) Load the WSJ headlines parquet and set display_date as a proper datetime␣
↪index
df = pd.read_parquet('articles.pq')
df['display_date'] = pd.to_datetime(df['display_date'])
df = df.set_index('display_date').sort_index()

# 2) Vectorize the headline text
texts = df['headline'].astype(str)
vectorizer = CountVectorizer(
    lowercase=True,
    stop_words='english',
    min_df=5
)
DTM = vectorizer.fit_transform(texts)
terms = vectorizer.get_feature_names_out()

# 2) Build group-id for each row
dates, group_ids = np.unique(df.index, return_inverse=True)
n_groups = len(dates)
n_docs   = DTM.shape[0]

# 3) Build a "one-hot" sparse matrix G of shape (n_groups × n_docs)
#    G[i, j] = 1 if doc j belongs to group i
G = csr_matrix((np.ones(n_docs), (group_ids, np.arange(n_docs))),
               shape=(n_groups, n_docs))

# 4) Multiply to aggregate: (n_groups × n_docs) × (n_docs × n_terms) →␣
↪(n_groups × n_terms)
counts_by_date_sparse = G.dot(DTM)

# 5) Wrap back into a DataFrame (still sparse)
counts_by_date = pd.DataFrame.sparse.from_spmatrix(
    counts_by_date_sparse,
    index=dates,
    columns=terms
)
```

```
# 4) Load your macro series (also indexed by date) and merge
macro = pd.read_csv('macro.csv',index_col=0, parse_dates=True)
macro.index = pd.to_datetime(macro.index).date

# 1) Normalize the WSJ-counts index to midnight
counts_by_date.index = pd.to_datetime(counts_by_date.index).date


data  = counts_by_date.join(macro, how='inner')

display(data.head())
```

```
             000  04  08  09  10  100  11  12  125  13  …  Boxes_vol  \
1984-02-01     0   0   0   0   0    0   0   0    0   0  …  -0.505793
1984-03-01     0   0   0   0   0    0   0   0    0   0  …  -0.241823
1984-06-01     0   0   0   0   0    0   0   0    0   0  …  -0.433055
1984-10-01     0   0   0   0   0  1.0   0   0    0   0  …  -0.677349
1984-11-01     0   0   0   0   0    0   0   0    0   0  …  -0.473934

             Trans_vol  Whlsl_vol  Rtail_vol  Meals_vol  Banks_vol  Insur_vol  \
1984-02-01    0.386029   0.502140   0.360408  -0.676141  -0.006122   0.018590
1984-03-01    0.025585   0.429562   0.163409  -0.659300   0.090492  -0.014528
1984-06-01   -0.105845   0.796249   0.057141  -0.616346   0.112409   0.119975
1984-10-01    0.302999   0.526044   0.038471  -0.683399   0.105847   0.197064
1984-11-01    0.175334   0.498414  -0.014093  -0.971428  -0.320444   0.438419

             RlEst_vol   Fin_vol  Other_vol
1984-02-01   -0.240550 -0.262150  -0.473532
1984-03-01   -0.242914 -0.317061  -0.810635
1984-06-01   -0.347377 -0.252008  -0.555581
1984-10-01   -0.275947 -0.761929  -0.474874
1984-11-01   -0.234875 -0.488590  -0.191555

[5 rows x 4875 columns]
```

(e) Next, repeat the contemporaneous exercises from part (a) and (b) using the counts. How many non-zero counts do you need to recover the same R2? What does that say about the informativeness of the counts vs. topics?

[38]:
```
# 4) Prepare feature matrix and scaling
X_sparse = csr_matrix(data[terms].values)
scaler = StandardScaler(with_mean=False).fit(X_sparse)
X_scaled = scaler.transform(X_sparse).tocsc()

# 5) Define penalty grid
alphas = np.logspace(0, -4, 100)

results = []
```

```python
# 6) Loop over each macro variable
for target in macro.columns:
    y = data[target]
    mask = y.notna().values
    y_mask = y.values[mask]
    Xm = X_scaled[mask]

    # 7) Compute Lasso path
    alphas_path, coefs_path, _ = lasso_path(
        Xm, y_mask,
        alphas=alphas,
        max_iter=5000,
        tol=1e-4,
    )

    # 8) Find alpha with exactly 30 non-zero coefficients
    alpha_sel = None
    coef_sel = None
    for a, coefs in zip(alphas_path, coefs_path.T):
        if np.count_nonzero(coefs) == 30:
            alpha_sel, coef_sel = a, coefs
            break

    if alpha_sel is None:
        results.append({
            'outcome': target,
            'alpha': None,
            'selected_terms': None,
            'r2': None
        })
        continue

    # 9) Run OLS on selected count terms
    sel_terms = [terms[i] for i, c in enumerate(coef_sel) if c != 0]
    X_sel = sm.add_constant(data.loc[mask, sel_terms])
    model = sm.OLS(y_mask, X_sel).fit()

    results.append({
        'outcome': target,
        'alpha': alpha_sel,
        'selected_terms': sel_terms,
        'r2': model.rsquared
    })

# 10) Display results
counts_results_df = pd.DataFrame(results).dropna(subset=['alpha']).
 ↪sort_values(by='r2', ascending=False)
```

```
counts_results_df
```

[38]:
```
      outcome     alpha                                         selected_terms  \
52  Other_vol  0.061359  [accused, allied, andrew, bp, brief, china, cl…
7    Beer_vol  0.055908  [abc, cathy, clark, currency, cutting, despite…
10    Fun_vol  0.073907  [anjali, biggest, bp, brief, businesses, cover…
30   Gold_vol  0.081113  [accord, act, ann, beatrice, central, coal, co…
27   Aero_vol  0.050941  [aftermath, allies, beat, bid, black, bp, brie…
37  BusSv_vol  0.042292  [2001, 95, americans, bank, blast, brief, busi…

          r2
52  0.624370
7   0.543686
10  0.521435
30  0.484875
27  0.456673
37  0.445086
```

- We can conclude that the topics are more informative than the counts. The topics capture the underlying themes and trends in the data, while the counts are simply a measure of the frequency of words. This suggests that the topics are more useful for understanding the relationships between the variables and making predictions.
- The number of non-zero counts needed to recover the same R^2 is 30. This suggests that the topics are more informative than the counts, as we only need a small number of topics to achieve a similar level of predictive power.

(f) Using the counts attempt to form the best forecasting model for industrial production growth. How well can you do relative to the topics?

[16]:
```
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import TruncatedSVD
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.metrics import r2_score

# 1) Ensure counts_by_date is aggregated and merged with macro
counts_by_date = counts_by_date.groupby(counts_by_date.index).sum()
macro.index = pd.to_datetime(macro.index).normalize()
y = macro['indprol1'].dropna()
dates = y.index

# 2) Build raw feature arrays aligned to y
X_topics_raw = topics.reindex(dates).fillna(0).values         # dense, small␣
 ↪dims
X_counts_raw = counts_by_date.reindex(dates).fillna(0).values # dense for SVD
```

```python
# 3) Scale topics
scaler_t = StandardScaler().fit(X_topics_raw)
Xt = scaler_t.transform(X_topics_raw)

# 4) Dimensionality reduction on counts (30 components)
svd = TruncatedSVD(n_components=30, random_state=42)
Xc_svd = svd.fit_transform(X_counts_raw)

# 5) Scale reduced counts
scaler_c = StandardScaler().fit(Xc_svd)
Xc = scaler_c.transform(Xc_svd)

# 6) Set up forecasting
start = 24  # warm-up period
y_true = []
y_pred_topics = []
y_pred_counts = []

# Pre-instantiate models with fewer trees for speed
model_t = HistGradientBoostingRegressor(
    max_iter=100,        # fewer trees
    learning_rate=0.1,   # faster learning
    max_depth=3,         # small trees
    random_state=42
)
model_c = HistGradientBoostingRegressor(
    max_iter=100,
    learning_rate=0.1,
    max_depth=3,
    random_state=42
)

# 7) Expanding-window forecasts
for t in range(start, len(dates)):
    y_true.append(y.values[t])

    # Topics-based forecast
    model_t.fit(Xt[:t], y.values[:t])
    y_pred_topics.append(model_t.predict(Xt[t].reshape(1, -1))[0])

    # Counts-based forecast (on reduced features)
    model_c.fit(Xc[:t], y.values[:t])
    y_pred_counts.append(model_c.predict(Xc[t].reshape(1, -1))[0])

# 8) Compute OOS R²
r2_topics = r2_score(y_true, y_pred_topics)
r2_counts = r2_score(y_true, y_pred_counts)
```

```
print(f"GBR OOS R² (Topics): {r2_topics:.4f}")
print(f"GBR OOS R² (Counts): {r2_counts:.4f}")
```

```
GBR OOS R² (Topics): -0.0038
GBR OOS R² (Counts): -0.0350
```

- We can observe that the topics result in better out-of-sample performance than the counts. This suggests that the topics are more informative for forecasting industrial production growth than the counts. The topics capture the underlying themes and trends in the data, while the counts are simply a measure of the frequency of words. This suggests that the topics are more useful for understanding the relationships between the variables and making predictions.

(g) Convert the raw counts into tf-idf and repeat the exercises from part (e) and (d). Summarize the differences between the tf-idf and raw count approaches. Which terms are most important in either approach?

[23]:
```
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import TruncatedSVD
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.metrics import r2_score

# 1) Aggregate counts and align macro as before
counts_by_date = counts_by_date.groupby(counts_by_date.index).sum()
macro.index = pd.to_datetime(macro.index).normalize()
y = macro['indprol1'].dropna()
dates = y.index

# 2) Build TF-IDF features (sparse matrix)
#    reindex ensures same dates as y, fill NaN→0
X_counts = counts_by_date.reindex(dates).fillna(0)
tfidf = TfidfTransformer(
    norm='l2',      # l2-normalize each row by default
    use_idf=True,   # include idf weighting
    smooth_idf=True,
    sublinear_tf=False
)
X_tfidf = tfidf.fit_transform(csr_matrix(X_counts.values))
# X_tfidf.shape == (len(dates), n_terms)

# 3) Dimensionality reduction on TF-IDF (30 components)
svd = TruncatedSVD(n_components=30, random_state=42)
Xc_svd = svd.fit_transform(X_tfidf)

# 4) Scale reduced features
```

```
scaler_c = StandardScaler().fit(Xc_svd)
Xc = scaler_c.transform(Xc_svd)

# 5) Expanding-window GBR (unchanged)
start = 24
y_true = []
y_pred_counts = []

model_c = HistGradientBoostingRegressor(
    max_iter=100,
    learning_rate=0.1,
    max_depth=3,
    random_state=42
)

for t in range(start, len(dates)):
    y_true.append(y.values[t])
    model_c.fit(Xc[:t], y.values[:t])
    y_pred_counts.append(model_c.predict(Xc[t].reshape(1, -1))[0])

r2_counts = r2_score(y_true, y_pred_counts)
print(f"GBR OOS R² (TF-IDF + SVD): {r2_counts:.4f}")
```

```
GBR OOS R² (TF-IDF + SVD): -0.0218
```

- TF–IDF helped by focusing on relative term importance, but for better forecasting we might need better tuning, richer features (lags, hybrid embeddings), or supervised dimension-reduction to turn that small improvement into a genuinely predictive model.

1.2.1 Problem 2.

Next, using the same ***articles.pq*** *file, we're going to explore using LLMs for generation with the* ***generation.py*** *script from canvas.*

[ ]:
```
## calling packages

import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from openai import OpenAI
from tqdm import tqdm
from scipy.sparse import csr_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import lasso_path
import statsmodels.api as sm
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.metrics import r2_score
from sklearn.decomposition import TruncatedSVD
```

```
from decouple import config
```

(a) Attempt to form a prompt that generates the topics discovered in 1.a and 1.b. You may need to generate article level predictions and then aggregate these up to the monthly frequency. What R^2 can you achieve with this approach?

(b) How much does prompt engineering change your results? Try the following: *i. Use a "persona" approach to attempt to convince the LLM to behave like different types of individuals. For example, try to convince the LLM to behave like a "bull" or a "bear". How much does this impact your results?*

ii. Use temperature to attempt to control the randomness of the LLM. How much does this impact your results? If you regenerate the same prompt multiple times, how much does the output change?

iii. Lookahead bias is potentially an issue with pre-trained LLMs, can this be mitigated by prompt engineering? Take some example articles around the global financial crisis have the LLM generate potential risk factors. By telling the LLM to ignore the future, can you mitigate lookahead bias?

(c) Using the generation approach, how well can you forecast industrial production growth? Document your approach and reasoning.