Early-Warning Inflation Model

Anticipating CPI Moves with Alternative Data

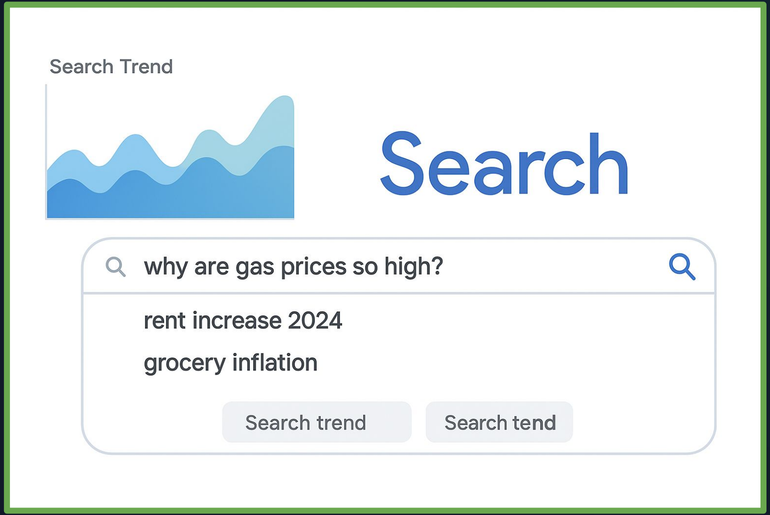
Swayam Singh, Brandon Nunez, Ritz Mahmood, Sameer Yamala

IDSC

Team DataFed

The Story: Why This Matters

- **CPI:** A statistical estimate of the level of prices of goods and services bought for consumption purposes by households.
- **The Lag Problem:** Official CPI data is released with a significant delay, leaving the common decision-makers reacting to the past rather than the present.
- **Real-Time Information Gap:** ***<u>We are in a volatile economy</u>***; markets and consumers experience price changes instantly, but the data doesn't reflect it for weeks.
- **The Opportunity:** Digital traces (Google Search Trends) act as a real-time proxy for consumer anxiety and price sensitivity.
- **Our Goal:** To bridge this gap by converting search volume into a leading indicator for inflation.



Initial Goals & Scope

- **Anticipate CPI:** Build a model that predicts CPI (Consumer Price Index) moves 1-3 months ahead of official releases.
- **Leverage Alternative Data:** Use fast-moving signals like "gas prices" and "rent" search volume instead of just historical econometric data.
- **Prove the "Lead":** Demonstrate statistically significant lead-lag relationships (Cross-Correlation).
- **Outperform Baselines:** Achieve lower RMSE (Root Mean Squared Error) than standard Seasonal-Naïve and ARIMA models.

Data Composition/Cleaning - From Raw CPI + Searches → Clean Monthly Dataset

date	CPI	CPI_MoM	CPI_YoY	rent_trend	gas_prices_trend	food_index_trend
2010-01-31	217.488	0.064873221	2.621111389	55	7	-0.86973471
2010-02-28	217.281	-0.095177665	2.151336358	53	5	-0.841723744
2010-03-31	217.353	0.033136814	2.286171439	55	7	-0.779662252
2010-04-30	217.403	0.023004053	2.206770753	56	7	-0.848690814
2010-05-31	217.29	-0.051977204	2.003548929	59	8	-0.86214279
2010-06-30	217.199	-0.041879516	1.121560594	61	6	-1.064931165
2010-07-31	217.605	0.186925354	1.34077848	57	6	-1.147755293
2010-08-31	217.923	0.146136348	1.15017754	56	5	-1.159222973
2010-09-30	218.275	0.161524942	1.118312247	47	6	-1.000570498
2010-10-31	219.035	0.348184629	1.166695149	47	6	-1.046470875
2010-11-30	219.59	0.253384162	1.084544777	46	7	-0.894579006
2010-12-31	220.472	0.401657635	1.437793022	42	9	-0.935518866
2011-01-31	221.187	0.32430422	1.700783492	62	11	-0.768414742
2011-02-28	221.898	0.321447463	2.124898173	64	20	-0.668367481
2011-03-31	223.046	0.517354821	2.61924151	62	30	-0.678568487
2011-04-30	224.093	0.469409898	3.077234445	61	29	-0.613889048
2011-05-31	224.806	0.318171473	3.458971881	65	28	-0.613317868
2011-06-30	224.806	0	3.502318151	71	18	-0.89327618
2011-07-31	225.395	0.262003683	3.579880977	68	14	-0.887551376
2011-08-31	226.106	0.315446217	3.754996031	66	14	-0.831422296
2011-09-30	226.597	0.217154786	3.812621693	56	12	-0.851676517
2011-10-31	226.75	0.067520753	3.522268131	52	11	-0.818185342
2011-11-30	227.169	0.184785006	3.451432215	50	9	-0.786870877
2011-12-31	227.223	0.023770849	3.062066838	52	9	-0.879906422
2012-01-31	227.842	0.272419605	3.008766338	64	12	-0.763812176
2012-02-29	228.329	0.213744612	2.898178442	61	24	-0.68499044
2012-03-31	228.807	0.209347039	2.582875281	60	24	-0.738810578
2012-04-30	229.187	0.166078835	2.273163374	63	18	-0.655452032
2012-05-31	228.713	-0.206818013	1.737942937	63	15	-0.706874724
2012-06-30	228.524	-0.082636317	1.653870448	66	16	-0.781459376
2012-07-31	228.59	0.028880993	1.41751148	71	13	-0.761502672

Sheet 1 - merged_inflation_data

Merged monthly CPI + Google Trends

merged_inflation_data.csv = cpi_data_only.csv + Google Trends

Monthly FRED CPI data (CPI, CPI_MoM, CPI_YoY) from 2010–2025

```
✓ Successfully fetched Google Trends data for 13 keyword(s)
  Date range: 2010-01-31 00:00:00 to 2025-12-31 00:00:00
Built 'food_index_trend' from 11 components

Merging CPI and Google Trends datasets...
✓ Successfully merged datasets
  Final dataset: 189 months of data
  Date range: 2010-01-31 00:00:00 to 2025-09-30 00:00:00

✓ Added derived CPI and trend features (horizon = 1 months)
  Rows available for modeling after dropping edge NaNs: 188
```

189 rows, 31 columns

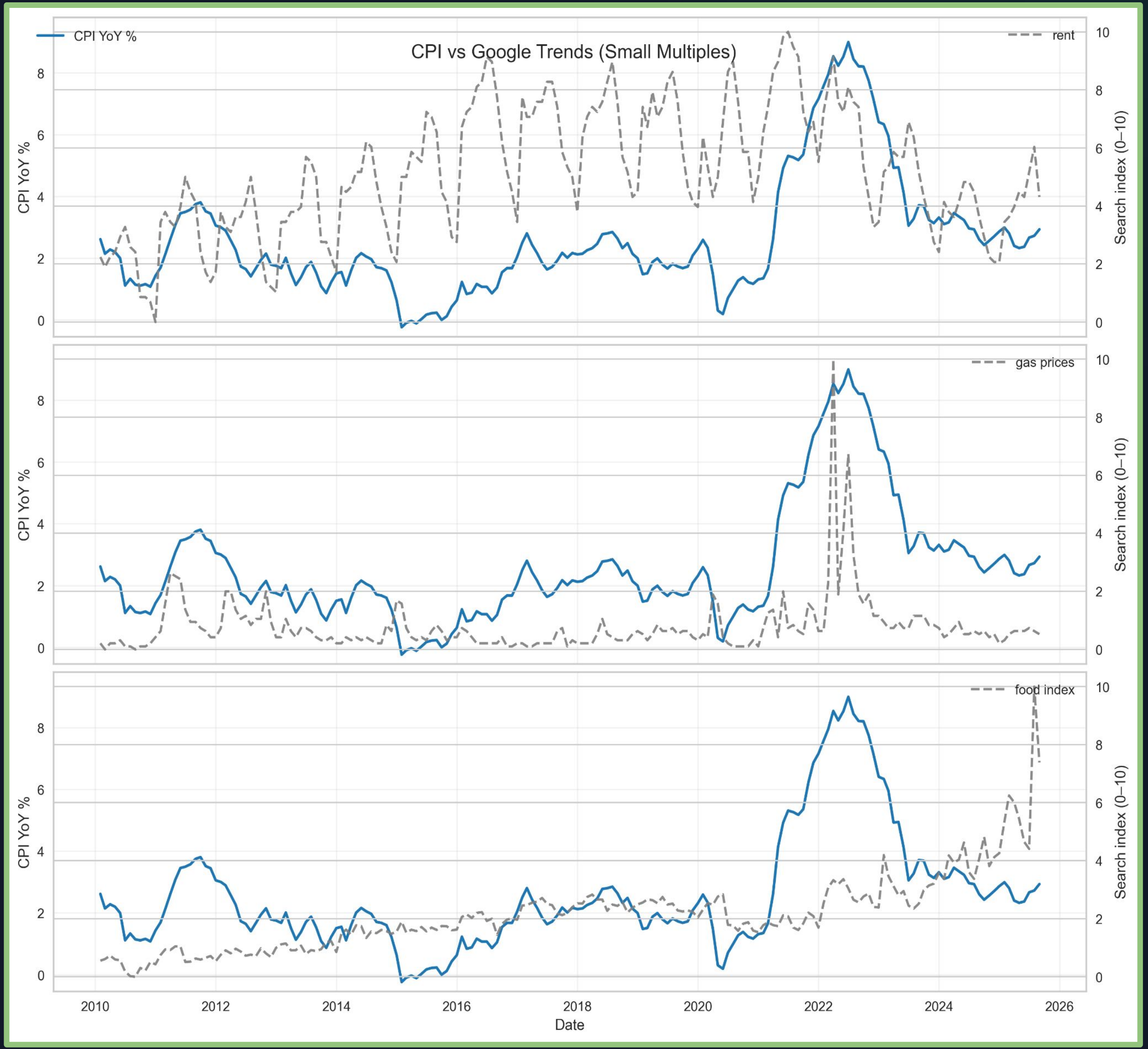
- **Inner join on month; drop months with missing search data to prevent artificial time gaps**

Data Cleaning: EDA Results

Cleaning Process - Rate Limiting: Implemented backoff strategies to fetch keywords without hitting API blocks.

Key Findings from 189 Months of Data

- **Strong Signal:** "Gas prices" - Spikes in gas price searches move closely with CPI YoY, especially during the 2021–22 inflation surge (corr ≈ 0.56)
- **Moderate Signal:** Rent searches change more slowly but still track inflation pressures over time (corr ≈ 0.20–0.28, depending on lead)
- Composite "Food index" - A combined index of grocery-related searches shows a moderate positive link to CPI and captures recent food-cost anxiety (late 2023–25), so we keep it as a supporting signal.



Data Cleaning: EDA Results Extended

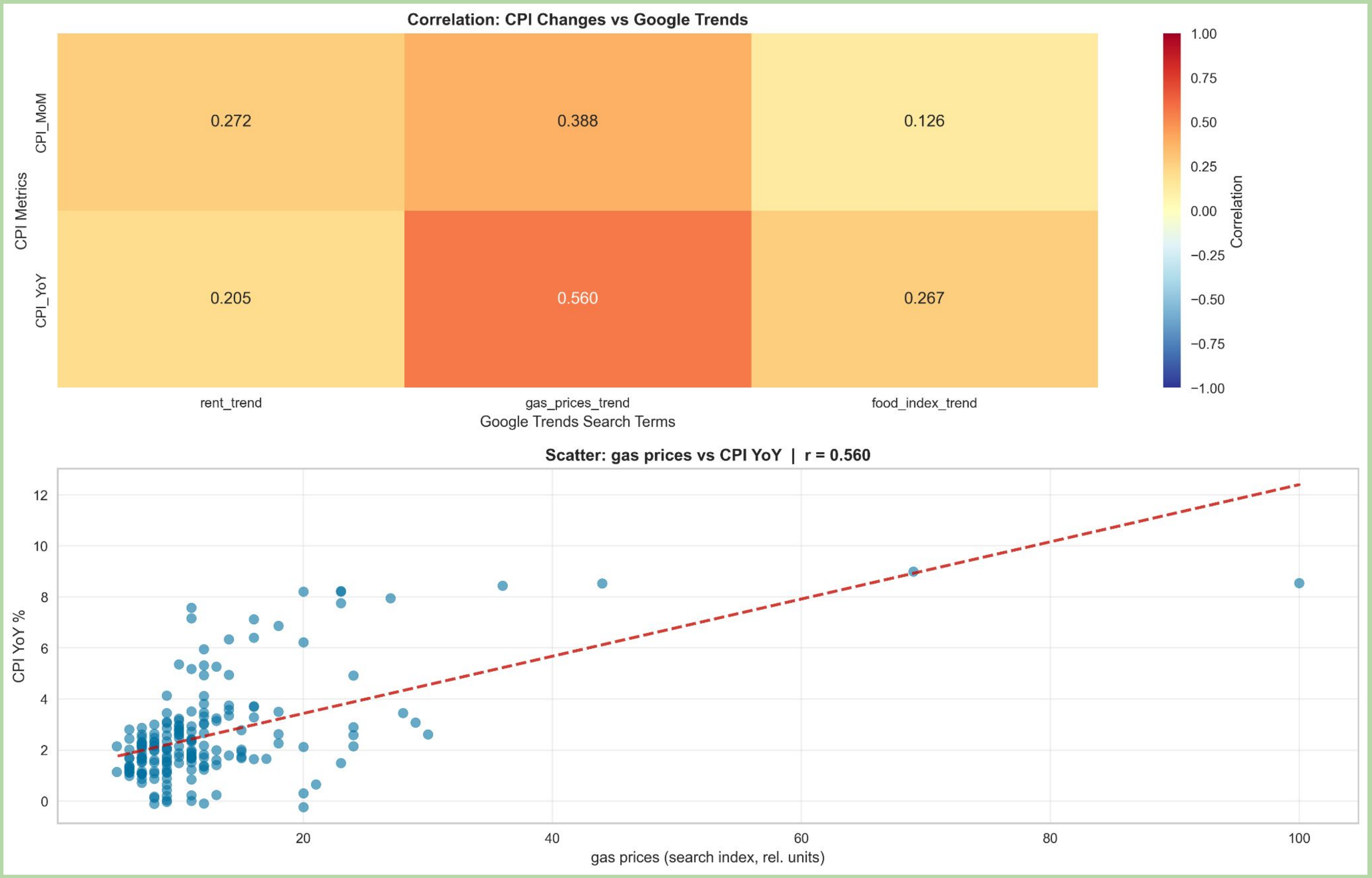
Key Findings in Depth - from 189 Months of Data

(gas prices r ≈ 0.56, rent ≈ 0.27, food index ≈ 0.27).

gas-price searches - strongest indicator

rent-price searches - hinting lead-lag effect

food index - noisy, agg, signal meaningful



Analysis: Preprocessing Strategy

component	explained_variance_ratio	cumulative_explained_variance
1	0.344299175	0.344299175
2	0.168727139	0.513026314
3	0.152996062	0.666022376
4	0.053378719	0.719401095
5	0.042380316	0.761781411
6	0.037723827	0.799505239
7	0.031640296	0.831145535
8	0.027934921	0.859080456
9	0.024480947	0.883561403
10	0.020195856	0.903757259
11	0.019859187	0.923616446
12	0.01598969	0.939606136
13	0.014828972	0.954435108

Feature Engineering

We created lagged versions of our search signals (t-1, t-2, t-3) to explicitly test predictive power at different time horizons.

Scaling (StandardScaler)

Standardized all numeric features (mean 0, std 1) before PCA and modeling (gas, rent, food index) Prevents high-variance series (like gas prices) from dominating the principal components and regression.

Dimensionality Reduction

Used PCA (Principal Component Analysis) to compress 28 raw numeric features into 13 key components, preserving 95% of variance while reducing noise.

Analysis: Model Selection (PyCaret)

```
==============================================================
Running PyCaret regression on PCA features...
==============================================================
                                    Model     MAE     MSE    RMSE      R2  \
lr                      Linear Regression  0.3019  0.1557  0.3878  0.9560
ridge                    Ridge Regression  0.3019  0.1555  0.3874  0.9560
lar                Least Angle Regression  0.3019  0.1557  0.3878  0.9560
br                         Bayesian Ridge  0.3018  0.1555  0.3872  0.9560
huber                     Huber Regressor  0.3065  0.1563  0.3902  0.9554
par          Passive Aggressive Regressor  0.3601  0.2216  0.4641  0.9374
et                  Extra Trees Regressor  0.3855  0.2532  0.4872  0.9275
gbr         Gradient Boosting Regressor  0.4174  0.3497  0.5755  0.8976
rf                Random Forest Regressor  0.4651  0.4542  0.6462  0.8721
ada                    AdaBoost Regressor  0.4828  0.4439  0.6520  0.8697
knn                 K Neighbors Regressor  0.5149  0.5653  0.7195  0.8427
en                            Elastic Net  0.5671  0.6058  0.7566  0.8292
lightgbm  Light Gradient Boosting Machine  0.6135  0.6752  0.8117  0.8056
lasso                    Lasso Regression  0.6521  0.7999  0.8721  0.7728
llar         Lasso Least Angle Regression  0.6521  0.7999  0.8721  0.7728
omp           Orthogonal Matching Pursuit  0.7033  0.8361  0.8995  0.7547
dt                Decision Tree Regressor  0.6393  0.8883  0.9327  0.7237
dummy                     Dummy Regressor  1.3123  3.5661  1.8770 -0.0239
```

Why AutoML?

- Instead of manually tuning one model, we used **PyCaret** to systematically test Linear, Tree-based, and Ensemble methods on the same PCA Features
- **Ranking Metric:** Models were ranked by Out-of-Sample RMSE (Root Mean Squared Error) target CPI_YoY_future_1m.

What Won?

A cluster of simple linear models - Linear Regression, Ridge, and Bayesian Ridge - best scores (RMSE ≈ 0.39, R^2 ≈ 0.96).

Tree-based models like Random Forest and Extra Trees were noticeably worse, and dummy baseline had RMSE ≈ 1.88

- Bayesian Ridge as our "winner" - robust to multicollinearity, but performance is essentially tied with plain OLS

Analysis: Validation Methodology

	A	B	C	D	E
1	date	actual	model	baseline_last_month	baseline_seasonal
2	1/31/20	2.599767977	1.714255225	2.31952747	1.487589358
3	2/29/20	2.34131668	1.648676928	2.599767977	1.518861535
4	3/31/20	1.494039964	2.361566232	2.34131668	1.883186351
5	4/30/20	0.313047294	2.274593147	1.494039964	2.00058347
6	5/31/20	0.198201304	1.682082695	0.313047294	1.795910555
7	6/30/20	0.716656283	1.550926101	0.198201304	1.671194894
8	7/31/20	0.996864763	1.518753571	0.716656283	1.826331335
9	8/31/20	1.28106985	1.571248348	0.996864763	1.737641211
10	9/30/20	1.391022891	1.618970873	1.28106985	1.684497704
11	10/31/20	1.230386343	1.628515378	1.391022891	1.7339737
12	11/30/20	1.175745214	1.692860438	1.230386343	2.092290395
13	12/31/20	1.320419132	1.542614834	1.175745214	2.31952747
14	1/31/21	1.355319978	1.850184072	1.320419132	2.599767977
15	2/28/21	1.667502411	2.058769246	1.355319978	2.34131668
16	3/31/21	2.623645748	2.133892048	1.667502411	1.494039964
17	4/30/21	4.137373453	1.715488463	2.623645748	0.313047294
18	5/31/21	4.926466564	2.508272821	4.137373453	0.198201304
19	6/30/21	5.317418943	1.926540112	4.926466564	0.716656283
20	7/31/21	5.269167647	1.987346156	5.317418943	0.996864763
21	8/31/21	5.181323173	1.908684339	5.269167647	1.28106985
22	9/30/21	5.363523425	1.862195052	5.181323173	1.391022891
23	10/31/21	6.226591221	2.360493692	5.363523425	1.230386343
24	11/30/21	6.86555952	2.248426107	6.226591221	1.175745214
25	12/31/21	7.159457345	1.837100552	6.86555952	1.320419132
26	1/31/22	7.578082463	1.818011542	7.159457345	1.355319978
27	2/28/22	7.949220899	2.647740683	7.578082463	1.667502411
28	3/31/22	8.540780149	6.351655063	7.949220899	2.623645748
29	4/30/22	8.235161744	2.452239029	8.540780149	4.137373453
30	5/31/22	8.530051713	3.552592625	8.235161744	4.926466564
31	6/30/22	8.999298142	4.862764633	8.530051713	5.317418943
32	7/31/22	8.447778207	3.12818355	8.999298142	5.269167647
33	8/31/22	8.21625506	2.452239029	8.447778207	5.181323173
34	9/30/22	8.205751582	2.322315672	8.21625506	5.363523425
35	10/31/22	7.757261471	2.461783534	8.205751582	6.226591221
36	11/30/22	7.131380369	2.098180503	7.757261471	6.86555952
37	12/31/22	6.410831677	2.040913473	7.131380369	7.159457345
38	1/31/23	6.34029631	1.909756879	6.410831677	7.578082463
39	2/28/23	5.957648713	1.778600285	6.34029631	7.949220899
40	3/31/23	4.931348642	1.783372537	5.957648713	8.540780149
41	4/30/23	4.946947488	1.933618142	4.931348642	8.235161744
42	5/31/23	4.125314539	1.835867315	4.946947488	8.530051713

Each row = one month in our test period (2019–2025).

actual = the true CPI year-over-year inflation for that month.

model = our search-trend model's prediction for that same month.

baseline_last_month = "just use last month's CPI YoY again" (naive guess).

baseline_seasonal = "use the CPI YoY from the same month last year" (seasonal guess)

Compare our model vs simple baselines month-by-month

From these columns we compute RMSE and directional accuracy to see if the model actually adds value over "just guess last month / last year"

.

Predictions & Conclusions

Trained model file (saved and reloadable): results/best_pycaret_model_yoy_1m.pkl.
Nowcast regression results: results/nowcast_r3_g0.csv.

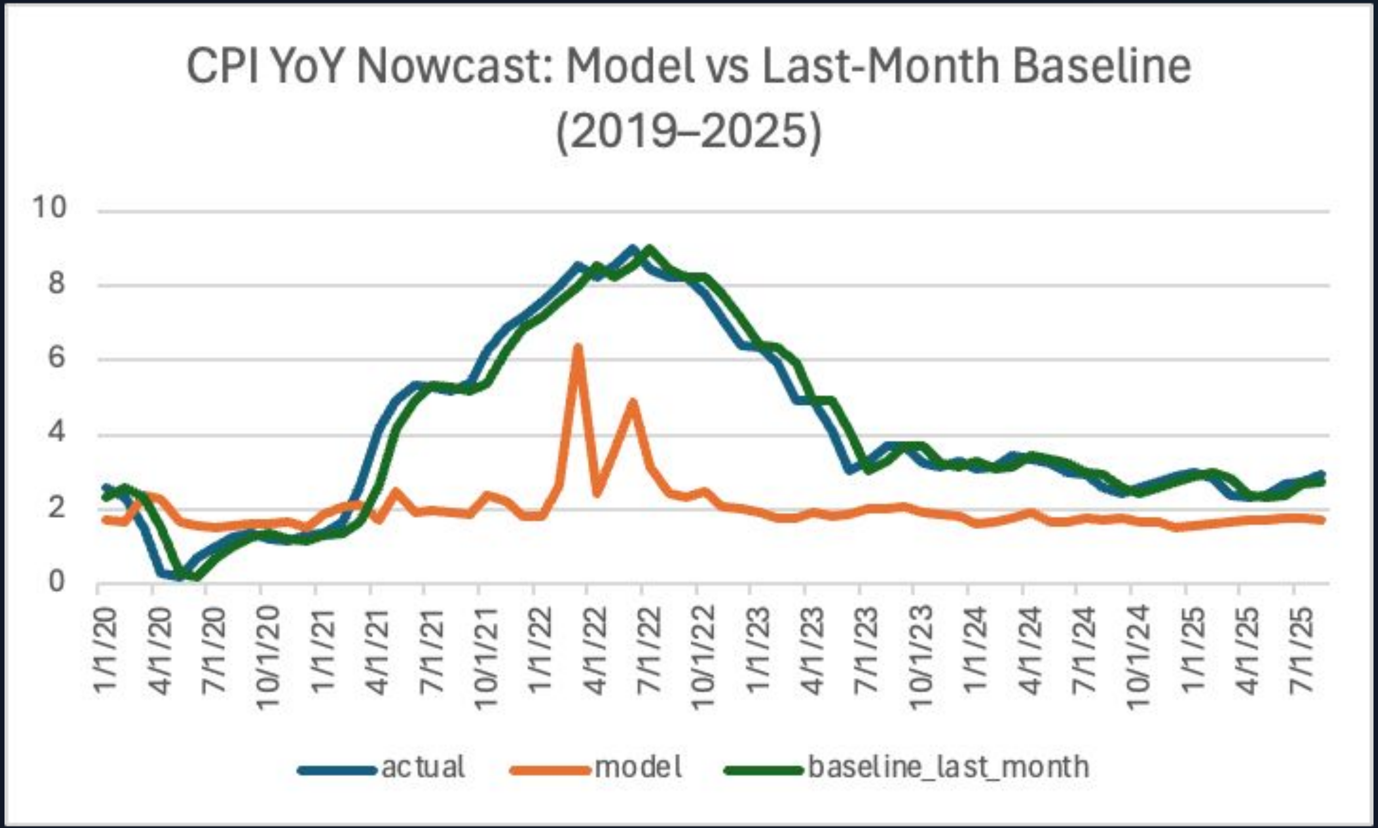
- **Strongest signal:** Google searches for "gas prices" have the highest link with inflation (correlation r ≈ 0.56), moving almost in real time with CPI YoY
- **Lagged Response:** "Rent" searches line up best when shifted ~3 months earlier, matching slower lease-renewal cycles.
- **Model performance:** With PCA + PyCaret, simple linear models (especially Bayesian Ridge) predict next-month CPI YoY with low cross-validated error (RMSE ≈ 0.39, R^2 ≈ 0.96).
- **RMSE:** Nowcast takeaway: In a 2019–2025 backtest, our search-based regression beats a seasonal baseline on error and direction (~55–60% right), but the "last-month CPI" rule is still very hard to beat..

Visualizing the Result

The Lead-Lag Relationship

Note how the green line (Search Trends) hugs the blue line (Official CPI) very closely. There is a timing gap between the two lines, which shows that search behavior reacts earlier than reported inflation, creating a 1–2 month window for policymakers and businesses to prepare for upcoming inflation changes.

The key thing to look at is how the predicted line follows the same general movement as the actual CPI, especially the timing of peaks and drops. Even though the prediction isn't perfect in terms of magnitude, the model still captures the direction and turning points reasonably well.



Roadblocks & Pivots

⚠ Signal Noise

Roadblock: "Food prices" keywords were extremely noisy and had zero correlation with actual food inflation.

Pivot: We dropped individual food items (eggs, milk) and tested aggregate terms like "grocery bill," to focus on the stronger Gas/Rent signals.

🚫 Rate Limits

Roadblock: Google Trends API aggressively blocks frequent requests, which stopped our data pipeline.

Pivot: We added backoff and caching so we could collect 5 years of data without getting blocked.

Next Steps

Based on what we learned, we see three clear next steps:

1

Incorporate weekly gasoline price data for higher-frequency validation.

2

Build a "Walk-Forward" simulation to stress-test the model on 2024 data.

3

Deploy as a live API that updates predictions every morning.

Thank you! Questions?