15.437 Options and Futures: Optional Project

Group 20

Anna Wang (@mit.edu MIT ID:)
Cathy Jin (cathyj@mit.edu MIT ID: 929907410)
Ke Zhang (kezhang@mit.edu MIT ID: 957125607)

Literature Review

Prior studies have explored the informational content of trader positions and inventory dynamics in shaping commodity price behavior. Dedi and Mandilaras (2022) examine the predictive content of trader positioning in the Brent oil futures market using vector autoregressions and Markov-switching models. They find that while managed money and producers adjust their positions in response to oil price shocks, there is limited evidence that positions themselves forecast price movements. These results underscore the importance of modeling assumptions and raise questions about whether trader positions contain forward-looking information or simply reflect contemporaneous market sentiment.

In parallel, substantial research emphasizes the role of inventory data as a determinant of commodity price dynamics. Indriawan et al. (2021) find that weekly U.S. Energy Information Administration (EIA) inventory announcements significantly shift intraday return predictability in crude oil markets, especially during high-volatility periods—highlighting inventory releases as key informational events. For refined products such as gasoline, Borenstein and Shepard (2002) and Pindyck (2004) argue that firms strategically use inventories to buffer supply-demand imbalances, which in turn influences both the speed and magnitude of price adjustments. Extending this line of inquiry, Kuper (2012) demonstrates that gasoline prices exhibit asymmetric responses to changes in the net marginal convenience yield, particularly under tight inventory conditions when storage or distribution frictions become binding.

Motivated by these findings, our study explores whether trader positioning data—either in isolation or interacted with inventory levels—can help forecast returns across different commodity markets and time horizons. By examining both crude oil and gasoline, we assess whether predictive relationships vary systematically with the storability and structural characteristics of the underlying asset.

Data Collection

This study examines trader behavior and price predictability in two highly liquid U.S. energy markets: **WTI Crude Oil (NYMEX: CL)** and **RBOB Gasoline (NYMEX: RB)**. The trader positioning data is sourced from the *Disaggregated Commitments of Traders (COT) – Futures Only* reports, published weekly by the U.S. Commodity Futures Trading Commission (CFTC).[1] These reports provide long and short open interest for five trader categories: Producer/Merchant, Swap Dealer, Managed Money, Other Reportables, and Non-Reportables. We extract and organize these reports into structured weekly time series, beginning in July 2009 for WTI (658 observations) and June 2006 for RBOB (787 observations), with both series extending through April 2025.

Futures prices are collected from Bloomberg using the continuous front-month contracts—*CL1 Comdty* for WTI and *XB1 Comdty* for RBOB. These series reflect daily settlement prices quoted by the exchange. Each weekly COT report is matched to its corresponding front-month futures price on the same COT report date.

---

[1] CFTC (Commodity Futures Trading Commission). (n.d.). Disaggregated Futures Only Reports. Retrieved April 16, 2025, from https://publicreporting.cftc.gov/stories/s/Disaggregated-Futures/ubmb-6exi/

Data Preparation

To avoid distortions related to contract expiration, we exclude rollover weeks from the return calculation. For WTI, contracts expire three business days prior to the 25th of the month preceding delivery.[2] For RBOB Gasoline, expiration occurs on the last business day of the month before delivery.[3] We identify all dates within five days of contract expiry and exclude them. These exclusions keep returns based on stable front-month pricing, avoiding the volatility that come with expiry.

Each trader group's net position is first expressed in contract counts (long – short) and then transformed into a 52-week z-score—subtracting the rolling mean and dividing by the rolling standard deviation for that group. This standardization strips out slow-moving level shifts and places all five trader categories on a common scale. The resulting dataset therefore contains weekly front-month simple returns and five z-scored positioning variables, which serve as the predictors in the linear-regression tests reported below.

---

[2] CME Group. (n.d.). *Crude Oil Futures Contract Specs*. Retrieved April 16, 2025, from https://www.cmegroup.com/markets/energy/crude-oil/light-sweet-crude.contractSpecs.html

[3] CME Group. (n.d.). *RBOB Gasoline Futures Contract Specs*. Retrieved April 16, 2025, fromhttps://www.cmegroup.com/markets/energy/refined-products/rbob-gasoline.contractSpecs.html

Linear Regression Model

We evaluate whether weekly trader positioning has explanatory power for near-term price dynamics by estimating linear regressions in which the dependent variable is the simple return on the front-month futures contract for WTI crude oil or RBOB gasoline. The five variables are 52-week z-scores of net positions for the CFTC trader groups—Producer/Merchant, Swap Dealer, Managed Money, Other Reportables, and Non-Reportables. In symbols, for a horizon of h=1,2,3 weeks,

$$r_{t+h} = \beta_0 + \beta_1 \,\mathrm{PMPU}_t^z + \beta_2 \,\mathrm{Swap}_t^z + \beta_3 \,\mathrm{MM}_t^z + \beta_4 \,\mathrm{Other}_t^z + \beta_5 \,\mathrm{NonRep}_t^z + \varepsilon_{t+h}.$$

We estimated the equation separately for WTI and RBOB using weekly observations from 2006 to 2025.

Regression Performance and Findings

The results are nearly flat. The goodness-of-fit metrics show that the weekly COT positioning data add very little explanatory power for short-horizon return forecasts in these two petroleum markets. As shown in the tables below, the highest R^2 we obtain is 1.4% for gasoline at the two-week horizon, while all WTI horizons remain below 1%.

However, RBOB gasoline offers one marginal signal. The Producer/Merchant's net position carries a coefficient of -0.0108 and a p-value of 0.048—just inside the 5 % threshold—for two-weeks returns, while every other trader group remains far from significance. The result suggests that commercial hedgers provide a small but detectable signal in the gasoline market, whereas the other trader groups do not. The negative sign implies that when commercials build a larger long position, the front-month price tends to fall over the next two weeks.

Table 1: Goodness-of-fit for WTI crude: R^2 by forecast horizon

Horizon	R^2
$t+1$	0.0072
$t+2$	0.0093
$t+3$	0.0058

Table 2: Goodness-of-fit for RBOB gasoline: R^2 by forecast horizon

Horizon	R^2
$t+1$	0.0125
$t+2$	0.0140
$t+3$	0.0106

Table 3: Coefficient estimates and p-values for the best-performing WTI model (t + 2 horizon)

Predictor	**Beta**	***p*-value**
Managed Money (MM)	0.0043	0.1457
Swap Dealer (Swap)	0.0054	0.1822
Producer / Merchant (Prod)	0.0042	0.3336
Other Reportables (Other)	0.0006	0.8747
Non-Reportables (NonRep)	0.0025	0.3043

Table 4: Coefficient estimates and p-values for the best-performing RBOB model (t + 2 horizon)

Predictor	**Beta**	***p*-value**
Managed Money (MM)	−0.0054	0.3298
Swap Dealer (Swap)	0.0004	0.8901
Producer / Merchant (Prod)	**-0.0108**	**0.0478**
Other Reportables (Other)	−0.0013	0.7151
Non-Reportables (NonRep)	−0.0033	0.3003

Adjusting for Actual Release Date

To better align the information set available to market participants, we re-estimate the predictive regressions using a modified timing convention: instead of matching each COT report date, we shift the positioning data forward by three calendar days to align with the actual release date of the report.

The results remain broadly consistent with our previous analysis. The overall predictive power remains weak, with R^2 values still below 1.4% across all horizons. For both WTI and RBOB, the models yield modest improvements in fit, but the changes are not statistically meaningful.

Table 5: Goodness-of-fit for WTI crude with release date: R^2 by forecast horizon

Horizon	R^2
t + 1	0.0058
t + 2	0.0103
t + 3	0.0086

Table 6: Goodness-of-fit for RBOB gasoline with release date: R^2 by forecast horizon

Horizon	R^2
t + 1	0.0092
t + 2	0.0123
t + 3	0.0096

Table 7: Coefficient estimates and p-values for the best-performing WTI model with release date (t+2 horizon)

Predictor	**Beta**	**p-value**
Managed Money (MM)	0.0045	0.0611
Swap Dealer (Swap)	0.0031	0.3527
Producer / Merchant (Prod)	0.0014	0.6965
Oter Reportables (Other)	-0.0006	0.8412
Non-Reportables (NonRep)	0.0008	0.7103

Table 8: Coefficient estimates and p-values for the best-performing RBOB model with release date (t+2 horizon)

Predictor	Beta	p-value
Managed Money (MM)	-0.0013	0.7974
Swap Dealer (Swap)	0.001	0.6804
Producer / Merchant (Prod)	-0.0067	0.1716
Oter Reportables (Other)	0.0004	0.8887
Non-Reportables (NonRep)	-0.0017	0.5525

As shown in Table 5-8, the best horizon for WTI remains at t+2, with R^2 improving marginally to 1.03%. For RBOB, t+2 also remains the best horizon with a near-identical R^2 of 1.23%.

No trader group reaches statistical significance at conventional thresholds. The previously significant signal from Producer/Merchants in the gasoline market (RBOB) disappears once the release timing is accounted for, suggesting the predictive content of that signal may have relied on information not yet available to the broader market. In contrast, the Managed Money position in WTI now carries the largest coefficient (0.0045) with a p-value of 0.0611—just above the 5% threshold.

It seems like adjusting for the actual release date of the COT reports does not materially change our conclusions. The COT data exhibits limited predictive power for short-horizon futures returns, and any signals observed in the raw report-date analysis appear even weaker once we align with realistic market information availability.

Storability, Inventory, and Return Predictability

Based on our model, there is some evidence suggesting that return predictability may vary with the storability of the underlying asset. We observe that RBOB gasoline, which is less storable, shows consistently higher explanatory power across all horizons, with a peak R^2 of 1.4% at the two-week horizon. WTI crude oil, which is relatively more storable, shows lower explanatory power, with R^2 values below 1%.

We then pulled the U.S. Petroleum and Other Liquids monthly Inventory data from the U.S. Energy Information Administration, using the “Crude Oil Closing Inventory” for crude and the “Total Motor Gasoline Closing Inventory” for gasoline.

To align with the monthly frequency of the EIA inventory data, we aggregate weekly asset returns into month-end returns. Simultaneously, we construct a monthly trader positioning signal by summing the weekly net-position Z-scores within each month. These monthly signals are then merged with the corresponding month-end EIA inventory levels, from which we compute a 12-month rolling Z-score to proxy market storability or tightness.

We estimate a series of predictive regressions to assess the explanatory power of inventory levels alone, trader positions alone, their combination, and models incorporating interaction terms. To further capture potential nonlinear relationships, we also implement a Random Forest regressor on the same feature set and compare model performance using R^2 and feature importance metrics. Based on goodness-of-fit statistics in Table 9, we could observe that random forests capture more information in return predictions and achieve higher R^2 due to its non-linear characteristics.

Table 9: R^2 for linear models and Random Forest by forecast horizon

Horizon	WTI Crude (Linear)	WTI Crude (RF)	Gasoline (Linear)	Gasoline (RF)
$t+1$	0.0688	0.5166	0.1747	0.5618
$t+2$	0.0654	0.5025	0.0596	0.5336
$t+3$	0.1025	0.5281	0.0998	0.5275

For WTI crude oil, we find that inventory levels alone exhibit minimal predictive power for future returns across all horizons. Incorporating the five trader positioning categories yields only marginal improvements in explanatory strength. However, once we introduce interaction terms allowing the slope of each position signal to vary with inventory levels, the model's explanatory power approximately doubles. Several interaction terms emerge as statistically significant. Notably, as presented in Table 10, Other Reportables × Inventory at t + 1, Swap Dealers × Inventory at t + 2 and t + 3, and Non-Reportables × Inventory at t + 3 contribute meaningfully to return predictability. The negative coefficients on these interaction terms suggest that when inventories are relatively tight—i.e., below their 12-month average—increase in the respective trader's net long position leads to a stronger positive return impact. The Random Forest features importance in Table 11 confirms that the Other Reportables × Storage interaction is the most important predictor, highlighting the role of storage constraints in amplifying the price impact of speculative positioning.

Table 10: Linear Regressions for WTI Crude Oil Models

Predictor	$t+1$ **Beta (p-value)**	$t+2$ **Beta (p-value)**	$t+3$ **Beta (p-value)**
MM_NetPos	−0.0002 (0.976)	−0.0098 (0.169)	−0.0005 (0.948)
Swap_NetPos	−0.0070 (0.426)	0.0096 (0.198)	−0.0035 (0.654)
Prod_NetPos	−0.0017 (0.794)	−0.0034 (0.704)	0.0104 (0.242)
Other_NetPos	−0.0014 (0.821)	−0.0004 (0.961)	0.0069 (0.191)
NonRep_NetPos	−0.0011 (0.848)	−0.0036 (0.710)	0.0129 (0.078)
Inventory	−0.0036 (0.555)	−0.0005 (0.930)	0.0044 (0.450)
MM_NetPos × Inventory	−0.0001 (0.984)	0.0041 (0.490)	−0.0019 (0.749)
Swap_NetPos × Inventory	−0.0057 (0.353)	−0.0191 **(0.016)**	−0.0152 **(0.004)**
Prod_NetPos × Inventory	−0.0026 (0.762)	0.0052 (0.377)	−0.0018 (0.751)
Other_NetPos × Inventory	−0.0182 **(0.000)**	−0.0121 (0.066)	0.0048 (0.246)
NonRep_NetPos × Inventory	−0.0031 (0.444)	−0.0084 (0.222)	−0.0099 **(0.025)**

Table 11: Random Forest Top 3 Important Features for WTI Crude Oil

Horizon	**Top 1 Feature**	**Top 2 Feature**	**Top 3 Feature**
$t+1$	Other_NetPos × Inventory (0.215)	Swap_NetPos (0.102)	Prod_NetPos (0.088)
$t+2$	Swap_NetPos × Inventory (0.175)	NonRep_NetPos (0.146)	Other_NetPos × Inventory (0.135)
$t+3$	MM_NetPos × Inventory (0.143)	NonRep_NetPos (0.142)	Swap_NetPos × Inventory (0.116)

The prominence of interaction terms in the WTI crude oil models indicates that the predictive power of trader positions is significantly influenced by current inventory levels. This reflects the globally integrated nature of crude oil markets, where trader behaviors are closely tied to macroeconomic indicators and global supply-demand balances.

In contrast, gasoline models show that inventory levels alone are the most significant predictors. Gasoline markets are more regionally fragmented and heavily influenced by localized factors such as seasonal demand fluctuations, refinery maintenance

schedules, and transportation constraints. Consequently, inventory levels directly reflect the immediate supply-demand balance in specific regions, making them a more reliable predictor of short-term price movements.

According to the results, next-month returns exhibit a strong and statistically significant relationship with inventory alone. This suggests that low gasoline inventories are reliably followed by higher futures returns. Adding the five trader positioning categories modestly improves the model fit, primarily due to a positive and significant coefficient on the Producer net position. Including interaction terms provides only a slight additional gain in explanatory power. None of the interaction terms are statistically significant at the t + 1 or t + 2 horizons, though a positive and weakly significant interaction between Swap Dealer positions and inventory emerges at t + 3. Consistent with the linear model findings, the Random Forest model assigns the greatest predictive importance to inventory, followed by the Producer position for t + 1. These results reinforce the idea that, for gasoline, inventory levels themselves are the dominant driver of short-term return variation, with trader flows playing a more limited role.

Table 12: Linear Regressions for RBOB Gasoline Models

Predictor	$t+1$ **Beta (p-value)**	$t+2$ **Beta (p-value)**	$t+3$ **Beta (p-value)**
MM_NetPos	−0.0139 (0.154)	0.0039 (0.745)	−0.0068 (0.538)
Swap_NetPos	−0.0019 (0.771)	0.0029 (0.689)	0.0043 (0.667)
Prod_NetPos	**0.0268 (0.003)**	0.0043 (0.713)	0.0125 (0.244)
Other_NetPos	−0.0008 (0.886)	**-0.0157 (0.018)**	**0.0161 (0.044)**
NonRep_NetPos	−0.0006 (0.923)	−0.0081 (0.238)	−0.0019 (0.795)
Inventory	**0.0366 (0.000)**	**0.0176 (0.033)**	−0.0027 (0.741)
MM_NetPos × Inventory	−0.0173 (0.181)	−0.0091 (0.552)	0.0123 (0.333)
Swap_NetPos × Inventory	−0.0005 (0.939)	−0.0052 (0.557)	**0.0253 (0.019)**
Prod_NetPos × Inventory	−0.0024 (0.813)	0.0028 (0.843)	−0.0171 (0.086)
Other_NetPos × Inventory	0.0094 (0.169)	−0.0092 (0.274)	0.0101 (0.310)
NonRep_NetPos × Inventory	0.0068 (0.299)	−0.0104 (0.126)	0.0030 (0.665)

Table 13: Random Forest Top 3 Important Features for RBOB Gasoline

Horizon	**Top 1 Feature**	**Top 2 Feature**	**Top 3 Feature**
$t+1$	Inventory (0.312)	Prod_NetPos (0.104)	NonRep_NetPos (0.079)
$t+2$	NonRep_NetPos × Inventory (0.133)	Swap_NetPos (0.132)	Inventory (0.128)
$t+3$	Swap_NetPos × Inventory (0.130)	Prod_NetPos (0.112)	Prod_NetPos × Inventory (0.100)

Appendix

Appendix I: Code/Data References:
https://github.com/CocoZhang1025/Futures-COT-Predictability

References

Dedi, V., & Mandilaras, A. (2022). Trader positions and the price of oil in the futures market. *International Review of Economics & Finance*, *82*, 448–460. https://doi.org/10.1016/j.iref.2022.06.018

Indriawan, I., Lien, D., Wen, Z., & Xu, Y. (2021). Intraday return predictability in the crude oil market: The role of EIA inventory announcements. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3822093

Kuper, G. H. (2012). Inventories and upstream gasoline price dynamics. *Energy Economics*, *34*(1), 208–214. https://doi.org/10.1016/j.eneco.2011.08.008

In [2]:
```
import pandas as pd
from pandas.tseries.offsets import BDay
import statsmodels.api as sm
import numpy as np
```

1. Merge COT Data with Futures Return

WTI Crude Oil

In [ ]:
```
# --- Load and clean the COT data ---
df_cot = pd.read_csv("WTI.csv")
df_cot["COT_Date"] = pd.to_datetime(df_cot["Report_Date_as_YYYY_MM_DD"])

# --- Load and clean front-month futures prices (CL1) ---
df_price_raw = pd.read_excel("CL1.xlsx", skiprows=6)
df_price_clean = df_price_raw[["Date", "PX_SETTLE"]].dropna()
df_price_clean.columns = ["Date", "Settle"]
df_price_clean["Date"] = pd.to_datetime(df_price_clean["Date"])

# --- Build expiry calendar for WTI contracts ---
# Note: CME WTI contracts expire 3 business days before the 25th of the month prior to delivery.
def get_expiry_dates(start, end):
    dates = pd.date_range(start, end, freq="MS")
    expiry = [pd.Timestamp(y, m, 25) - BDay(3) for y, m in zip(dates.year, dates.month)]
    expiry_df = pd.DataFrame({"Expiry": expiry})
    expiry_df["YearMonth"] = expiry_df["Expiry"].dt.to_period("M")
    return expiry_df
expiry_calendar = get_expiry_dates(df_price_clean["Date"].min(), df_price_clean["Date"].max())

# --- Assign each price to a contract month and flag rollover weeks ---
df_price_clean["YearMonth"] = df_price_clean["Date"].dt.to_period("M")
df_price_clean = df_price_clean.merge(expiry_calendar, on="YearMonth", how="left")
# Compute days-to-expiry
df_price_clean["DaysToExpiry"] = (df_price_clean["Expiry"] - df_price_clean["Date"]).dt.days
df_price_clean["RolloverRisk"] = df_price_clean["DaysToExpiry"].between(0, 5) # Any price date within 5
calendar days of expiry is flagged

# --- Merge COT data with prices and drop rollover weeks ---
df_merged = pd.merge(df_cot, df_price_clean, left_on="COT_Date", right_on="Date", how="inner")
df_merged_clean = df_merged[~df_merged["RolloverRisk"]].copy()

# --- Calculate weekly returns ---
df_merged_clean = df_merged_clean.sort_values("COT_Date")
df_merged_clean["Settle_t+1"] = df_merged_clean["Settle"].shift(-1)
df_merged_clean["Return"] = (df_merged_clean["Settle_t+1"] - df_merged_clean["Settle"]) / df_merged_clea
n["Settle"]
```

```
/var/folders/q3/lw2ccwxs3gbd91grj537_4y40000gn/T/ipykernel_18229/113116674.py:3: UserWarning: Could not
infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing
is consistent and as-expected, please specify a format.
  df_cot["COT_Date"] = pd.to_datetime(df_cot["Report_Date_as_YYYY_MM_DD"])
```

In [ ]:
```
useful_columns = [
    "COT_Date", "Settle", "Settle_t+1", "Return","Open_Interest_All",
    # Producer/Merchant
    "Prod_Merc_Positions_Long_All", "Prod_Merc_Positions_Short_All", "Prod_Net",
    # Swap Dealer
    "Swap_Positions_Long_All", "Swap__Positions_Short_All", "Swap_Net",
    # Managed Money
    "M_Money_Positions_Long_All", "M_Money_Positions_Short_All", "MM_Net",
    # Other Reportables
    "Other_Rept_Positions_Long_All", "Other_Rept_Positions_Short_All","Other_Net",
    # Non-Reportables
    "NonRept_Positions_Long_All", "NonRept_Positions_Short_All","NonRep_Net"
]
df_merged_clean["MM_Net"] = df_merged_clean["M_Money_Positions_Long_All"] - df_merged_clean["M_Money_Pos
itions_Short_All"]
df_merged_clean["Swap_Net"] = df_merged_clean["Swap_Positions_Long_All"] - df_merged_clean["Swap__Positi
ons_Short_All"]
df_merged_clean["Prod_Net"] = df_merged_clean["Prod_Merc_Positions_Long_All"] - df_merged_clean["Prod_Me
rc_Positions_Short_All"]
df_merged_clean["Other_Net"] = df_merged_clean["Other_Rept_Positions_Long_All"] - df_merged_clean["Other
_Rept_Positions_Short_All"]
df_merged_clean["NonRep_Net"] = df_merged_clean["NonRept_Positions_Long_All"] - df_merged_clean["NonRept
_Positions_Short_All"]

df_WTI_final = df_merged_clean[useful_columns].copy()
```

RBOB Gasoline

In [ ]:
```
# --- Load and prepare RBOB position data ---
df_gas = pd.read_csv("Gasoline.csv")
df_gas["COT_Date"] = pd.to_datetime(df_gas["Report_Date_as_YYYY_MM_DD"])

# --- Load and prepare XB1 front-month futures price data ---
df_xb1_raw = pd.read_excel("XB1.xlsx", skiprows=6)
df_xb1_clean = df_xb1_raw[["Date", "PX_SETTLE"]].dropna()
df_xb1_clean.columns = ["Date", "Settle"]
df_xb1_clean["Date"] = pd.to_datetime(df_xb1_clean["Date"])

# --- Build expiry calendar for RBOB Gasoline ---
# RBOB contracts expire on the last business day of the month before the contract month
def get_rbob_expiry_dates(start, end):
    dates = pd.date_range(start, end, freq="MS")
    expiry = [pd.Timestamp(y, m, 1) - BDay(1) for y, m in zip(dates.year, dates.month)]
    expiry_df = pd.DataFrame({"Expiry": expiry})
    expiry_df["YearMonth"] = expiry_df["Expiry"].dt.to_period("M")
    return expiry_df

rbob_expiry_calendar = get_rbob_expiry_dates(df_xb1_clean["Date"].min(), df_xb1_clean["Date"].max())

# --- Flag rollover risk ---
df_xb1_clean["YearMonth"] = df_xb1_clean["Date"].dt.to_period("M")
df_xb1_clean = df_xb1_clean.merge(rbob_expiry_calendar, on="YearMonth", how="left")
df_xb1_clean["DaysToExpiry"] = (df_xb1_clean["Expiry"] - df_xb1_clean["Date"]).dt.days
df_xb1_clean["RolloverRisk"] = df_xb1_clean["DaysToExpiry"].between(0, 5)

# --- Merge COT data with XB1 price data and exclude rollover weeks ---
df_merged = pd.merge(df_gas, df_xb1_clean, left_on="COT_Date", right_on="Date", how="inner")
df_merged_clean = df_merged[~df_merged["RolloverRisk"]].copy()

# --- Calculate weekly returns ---
df_merged_clean = df_merged_clean.sort_values("COT_Date")
df_merged_clean["Settle_t+1"] = df_merged_clean["Settle"].shift(-1)
df_merged_clean["Return"] = (df_merged_clean["Settle_t+1"] - df_merged_clean["Settle"]) / df_merged_clea
n["Settle"]

# --- Calculate net positions for all five trader types ---
df_merged_clean["MM_Net"] = df_merged_clean["M_Money_Positions_Long_All"] - df_merged_clean["M_Money_Pos
itions_Short_All"]
df_merged_clean["Swap_Net"] = df_merged_clean["Swap_Positions_Long_All"] - df_merged_clean["Swap__Positi
ons_Short_All"]
df_merged_clean["Prod_Net"] = df_merged_clean["Prod_Merc_Positions_Long_All"] - df_merged_clean["Prod_Me
rc_Positions_Short_All"]
df_merged_clean["Other_Net"] = df_merged_clean["Other_Rept_Positions_Long_All"] - df_merged_clean["Other
_Rept_Positions_Short_All"]
df_merged_clean["NonRep_Net"] = df_merged_clean["NonRept_Positions_Long_All"] - df_merged_clean["NonRept
_Positions_Short_All"]

useful_columns = [
    "COT_Date", "Settle", "Settle_t+1", "Return","Open_Interest_All",
    "Prod_Merc_Positions_Long_All", "Prod_Merc_Positions_Short_All", "Prod_Net",
    "Swap_Positions_Long_All", "Swap__Positions_Short_All", "Swap_Net",
    "M_Money_Positions_Long_All", "M_Money_Positions_Short_All", "MM_Net",
    "Other_Rept_Positions_Long_All", "Other_Rept_Positions_Short_All", "Other_Net",
    "NonRept_Positions_Long_All", "NonRept_Positions_Short_All", "NonRep_Net"
]

df_XB_final = df_merged_clean[useful_columns].copy()
df_XB_final.to_excel("XB_merged.xlsx", index=False)
```

```
/var/folders/q3/lw2ccwxs3gbd91grj537_4y40000gn/T/ipykernel_18229/3021542581.py:3: UserWarning: Could not
infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing
is consistent and as-expected, please specify a format.
  df_gas["COT_Date"] = pd.to_datetime(df_gas["Report_Date_as_YYYY_MM_DD"])
```

2. Feature transforms (net positions as % change and z-scores)

In [12]:

In [13]:
```
# --- Convert to % change ---
net_cols = ["MM_Net", "Swap_Net", "Prod_Net", "Other_Net", "NonRep_Net"]

# --- For RBOB ---
for col in net_cols:
    df_XB_final[f"{col}_Chg"] = df_XB_final[col].pct_change()
    df_XB_final[f"{col}_Chg"] =df_XB_final[f"{col}_Chg"].replace([float("inf"), float("-inf")], pd.NA)

# --- For WTI ---
for col in net_cols:
    df_WTI_final[f"{col}_Chg"] = df_WTI_final[col].pct_change()
    df_WTI_final[f"{col}_Chg"] = df_WTI_final[f"{col}_Chg"].replace([float("inf"), float("-inf")], pd.N
A)

df_WTI_final.to_excel("WTI_merged.xlsx", index=False)
df_XB_final.to_excel("XB_merged.xlsx", index=False)
```

In [14]:
```
# --- Convert to z-scores ---

# Rolling Z-scores (52-week window)
window = 52
def add_z(df):
    for col in net_cols:
        df[col+"_z"] = (df[col] - df[col].rolling(window, min_periods=26).mean()) / df[col].rolling(wind
ow, min_periods=26).std()
    return df

df_WTI = add_z(df_WTI_final)
df_XB  = add_z(df_XB_final)

z_cols = [c+"_z" for c in net_cols]
```

3. Regression Model

In [ ]:
```
def get_R2_and_models(df, label):
    R2s = {}
    models = {}
    for h in [1,2,3]:
        tgt = f"Ret_t{h}"
        df[tgt] = df["Return"].shift(-h)
        reg = df[[tgt]+z_cols].dropna()
        X = sm.add_constant(reg[z_cols])
        res = sm.OLS(reg[tgt], X).fit()
        R2s[h] = res.rsquared
        models[h] = res
    return R2s, models

R2_wti, models_wti = get_R2_and_models(df_WTI,"WTI")
R2_xb , models_xb  = get_R2_and_models(df_XB ,"RBOB")

R2_table_WTI = pd.DataFrame({
    "Lag": ["t+1","t+2","t+3"],
    "R2": [round(R2_wti[1],4), round(R2_wti[2],4), round(R2_wti[3],4)]
})
R2_table_XB = pd.DataFrame({
    "Lag": ["t+1","t+2","t+3"],
    "R2": [round(R2_xb[1],4), round(R2_xb[2],4), round(R2_xb[3],4)]
})

# Select best horizon
best_wti_h = max(R2_wti, key=R2_wti.get)
best_xb_h  = max(R2_xb , key=R2_xb.get)

def beta_table(res, label, horizon):
    rows = []
    for var in z_cols:
        rows.append({
            "Predictor": var.replace("_Net_z","").replace("_z",""),
            "Beta": round(res.params[var],4),
            "p_value": round(res.pvalues[var],4)
        })
    return pd.DataFrame(rows)

beta_wti = beta_table(models_wti[best_wti_h], "WTI", best_wti_h)
beta_xb  = beta_table(models_xb[best_xb_h], "RBOB", best_xb_h)
```

In [ ]:
```
R2_table_WTI
```

Out[ ]:

	Lag	R2
0	t+1	0.0072
1	t+2	0.0093
2	t+3	0.0058

In [ ]:
```
R2_table_XB
```

Out[ ]:

	Lag	R2
0	t+1	0.0125
1	t+2	0.0140
2	t+3	0.0106

In [ ]:
```
beta_wti
```

Out[ ]:

	Predictor	Beta	p_value
0	MM	0.0043	0.1457
1	Swap	0.0054	0.1822
2	Prod	0.0042	0.3336
3	Other	0.0006	0.8747
4	NonRep	0.0025	0.3043

In [ ]:
```
beta_xb
```

Out[ ]:

	Predictor	Beta	p_value
0	MM	-0.0054	0.3298
1	Swap	0.0004	0.8901
2	Prod	-0.0108	0.0478
3	Other	-0.0013	0.7151
4	NonRep	-0.0033	0.3003

5. Release Date Regression

In [ ]:
```
from pandas.tseries.holiday import USFederalHolidayCalendar
from pandas.tseries.offsets import CustomBusinessDay
us_bd = CustomBusinessDay(calendar=USFederalHolidayCalendar())
```

In [ ]:
```
df_cot_release = df_cot.copy()
df_cot_release["Release_Date"] = df_cot_release["COT_Date"] + 3 * us_bd

df_merged_release = pd.merge(
    df_cot_release,
    df_price_clean,
    left_on="Release_Date",
    right_on="Date",
    how="inner"
)

df_merged_clean_release = df_merged_release[~df_merged_release["RolloverRisk"]].copy()

df_merged_clean_release = df_merged_clean_release.sort_values("Release_Date")
df_merged_clean_release["Settle_t+1"] = df_merged_clean_release["Settle"].shift(-1)
df_merged_clean_release["Return"] = (
    df_merged_clean_release["Settle_t+1"] - df_merged_clean_release["Settle"]
) / df_merged_clean_release["Settle"]

df_merged_clean_release["MM_Net"] = df_merged_clean_release["M_Money_Positions_Long_All"] - df_merged_cl
ean_release["M_Money_Positions_Short_All"]
df_merged_clean_release["Swap_Net"] = df_merged_clean_release["Swap_Positions_Long_All"] - df_merged_cle
an_release["Swap__Positions_Short_All"]
df_merged_clean_release["Prod_Net"] = df_merged_clean_release["Prod_Merc_Positions_Long_All"] - df_merge
d_clean_release["Prod_Merc_Positions_Short_All"]
df_merged_clean_release["Other_Net"] = df_merged_clean_release["Other_Rept_Positions_Long_All"] - df_mer
ged_clean_release["Other_Rept_Positions_Short_All"]
df_merged_clean_release["NonRep_Net"] = df_merged_clean_release["NonRept_Positions_Long_All"] - df_merge
d_clean_release["NonRept_Positions_Short_All"]

df_WTI_release_final = df_merged_clean_release[useful_columns].copy()
net_cols = ["MM_Net", "Swap_Net", "Prod_Net", "Other_Net", "NonRep_Net"]
df_WTI_release = add_z(df_WTI_release_final)
R2_wti_release, models_wti_release = get_R2_and_models(df_WTI_release, "WTI_Release")

R2_table_WTI_release = pd.DataFrame({
    "Lag": ["t+1", "t+2", "t+3"],
    "R2": [round(R2_wti_release[1],4), round(R2_wti_release[2],4), round(R2_wti_release[3],4)]
})
R2_table_WTI_release
```

```
/var/folders/q3/lw2ccwxs3gbd91grj537_4y40000gn/T/ipykernel_18229/1767991986.py:2: PerformanceWarning: No
n-vectorized DateOffset being applied to Series or DatetimeIndex.
  df_cot_release["Release_Date"] = df_cot_release["COT_Date"] + 3 * us_bd
```

Out[ ]:

	Lag	R2
0	t+1	0.0058
1	t+2	0.0103
2	t+3	0.0086

In [ ]:
```
df_gas_release = df_gas.copy()
df_gas_release["Release_Date"] = df_gas_release["COT_Date"] + 3 * us_bd

df_merged_xb_release = pd.merge(
    df_gas_release,
    df_xb1_clean,
    left_on="Release_Date",
    right_on="Date",
    how="inner"
)

df_merged_xb_clean_release = df_merged_xb_release[~df_merged_xb_release["RolloverRisk"]].copy()

df_merged_xb_clean_release = df_merged_xb_clean_release.sort_values("Release_Date")
df_merged_xb_clean_release["Settle_t+1"] = df_merged_xb_clean_release["Settle"].shift(-1)
df_merged_xb_clean_release["Return"] = (
    df_merged_xb_clean_release["Settle_t+1"] - df_merged_xb_clean_release["Settle"]
) / df_merged_xb_clean_release["Settle"]

df_merged_xb_clean_release["MM_Net"] = df_merged_xb_clean_release["M_Money_Positions_Long_All"] - df_mer
ged_xb_clean_release["M_Money_Positions_Short_All"]
df_merged_xb_clean_release["Swap_Net"] = df_merged_xb_clean_release["Swap_Positions_Long_All"] - df_merg
ed_xb_clean_release["Swap__Positions_Short_All"]
df_merged_xb_clean_release["Prod_Net"] = df_merged_xb_clean_release["Prod_Merc_Positions_Long_All"] - df
_merged_xb_clean_release["Prod_Merc_Positions_Short_All"]
df_merged_xb_clean_release["Other_Net"] = df_merged_xb_clean_release["Other_Rept_Positions_Long_All"] -
df_merged_xb_clean_release["Other_Rept_Positions_Short_All"]
df_merged_xb_clean_release["NonRep_Net"] = df_merged_xb_clean_release["NonRept_Positions_Long_All"] - df
_merged_xb_clean_release["NonRept_Positions_Short_All"]

df_XB_release_final = df_merged_xb_clean_release[useful_columns].copy()

df_XB_release = add_z(df_XB_release_final)

R2_xb_release, models_xb_release = get_R2_and_models(df_XB_release, "RBOB_Release")
R2_table_XB_release = pd.DataFrame({
    "Lag": ["t+1", "t+2", "t+3"],
    "R2": [round(R2_xb_release[1], 4), round(R2_xb_release[2], 4), round(R2_xb_release[3], 4)]
})
R2_table_XB_release
```

```
/var/folders/q3/lw2ccwxs3gbd91grj537_4y40000gn/T/ipykernel_18229/1088181742.py:2: PerformanceWarning: No
n-vectorized DateOffset being applied to Series or DatetimeIndex.
  df_gas_release["Release_Date"] = df_gas_release["COT_Date"] + 3 * us_bd
```

Out[ ]:

	Lag	R2
0	t+1	0.0092
1	t+2	0.0123
2	t+3	0.0096

In [ ]:
```
best_wti_release_h = max(R2_wti_release, key=R2_wti_release.get)
beta_wti_release = beta_table(models_wti_release[best_wti_release_h], "WTI_Release", best_wti_release_h)
beta_wti_release
```

Out[ ]:

	Predictor	Beta	p_value
0	MM	0.0045	0.0611
1	Swap	0.0031	0.3527
2	Prod	0.0014	0.6965
3	Other	-0.0006	0.8412
4	NonRep	0.0008	0.7103

In [ ]:
```
best_xb_release_h = max(R2_xb_release, key=R2_xb_release.get)
beta_xb_release = beta_table(models_xb_release[best_xb_release_h], "RBOB_Release", best_xb_release_h)
beta_xb_release
```

Out[ ]:

	Predictor	Beta	p_value
0	MM	-0.0013	0.7974
1	Swap	0.0010	0.6804
2	Prod	-0.0067	0.1716
3	Other	0.0004	0.8887
4	NonRep	-0.0017	0.5525

In [ ]:

Inventory

In [3]:
```
df_XB  = pd.read_excel("XB_merged.xlsx")
df_WTI = pd.read_excel("WTI_merged.xlsx")
```

In [6]:
```
inv = pd.read_csv("wti_inv.csv", index_col=0)
s = inv.iloc[0]
s.index = pd.to_datetime(s.index, format="%b-%y").to_period("M")
s.name = "Inventory"
inv_wti = s.reset_index().rename(columns={"index":"COT_Date"})
inv_wti["COT_Date"] = inv_wti["COT_Date"].dt.strftime("%Y-%m")

inv_wti
```

Out[6]:

	COT_Date	Inventory
0	2009-07	327.2
1	2009-08	317.5
2	2009-09	317.1
3	2009-10	314.4
4	2009-11	318.8
...	...	...
205	2026-08	414.2
206	2026-09	412.7
207	2026-10	423.4
208	2026-11	422.0
209	2026-12	413.1

210 rows × 2 columns

In [9]:
```
inv_xb = pd.read_csv("xb_inv.csv", index_col=0)
g = inv_xb.iloc[0]
g.index = pd.to_datetime(g.index, format="%b-%y").to_period("M")
g.name = "Inventory"
inv_xb = g.reset_index().rename(columns={"index":"COT_Date"})
inv_xb["COT_Date"] = inv_xb["COT_Date"].dt.strftime("%Y-%m")

inv_xb
```

Out[9]:

	COT_Date	Inventory
0	2006-06	213.3
1	2006-07	208.9
2	2006-08	209.0
3	2006-09	214.1
4	2006-10	204.6
...	...	...
242	2026-08	209.0
243	2026-09	206.8
244	2026-10	202.7
245	2026-11	212.1
246	2026-12	226.8

247 rows × 2 columns

In [10]:
```
df_WTI = pd.read_excel("WTI_merged.xlsx", parse_dates=["COT_Date"])
df_WTI = (
    df_WTI
      .set_index("COT_Date")
      .sort_index()
)

delta_cols = [
    "MM_Net_Chg", "Swap_Net_Chg", "Prod_Net_Chg",
    "Other_Net_Chg", "NonRep_Net_Chg"
]

# Monthly return: compound (1+weekly_return) across all weeks in month, then subtract 1
monthly_ret_WTI = (
    df_WTI["Return"].add(1)
      .resample("M")     # month-end; you can also use "ME" for explicit MonthEnd
      .prod()
      .sub(1)
      .rename("Ret_m")
)

# Monthly net position: sum the weekly Δ across the month
monthly_deltas = (
    df_WTI[delta_cols]
      .resample("M")
      .sum()
      # rename columns to shorter Δ names if you like:
      .rename(columns={
          "MM_Net_Chg":"MM_Δ",
          "Swap_Net_Chg":"Swap_Δ",
          "Prod_Net_Chg":"Prod_Δ",
          "Other_Net_Chg":"Other_Δ",
          "NonRep_Net_Chg":"NonRep_Δ"
      })
)

# Combine into a single monthly DataFrame
df_m_wti = pd.concat([monthly_ret_WTI, monthly_deltas], axis=1).dropna()
df_m_wti.index = df_m_wti.index.strftime("%Y-%m")
df_m_wti
```

```
<ipython-input-10-b2d32325a39c>:16: FutureWarning: 'M' is deprecated and will be removed in a future ver
sion, please use 'ME' instead.
  .resample("M")     # month-end; you can also use "ME" for explicit MonthEnd
<ipython-input-10-b2d32325a39c>:25: FutureWarning: 'M' is deprecated and will be removed in a future ver
sion, please use 'ME' instead.
  .resample("M")
```

Out[10]:

COT_Date	Ret_m	MM_Δ	Swap_Δ	Prod_Δ	Other_Δ	NonRep_Δ
2009-07	0.062323	0.000000	0.000000	0.000000	0.000000	0.000000
2009-08	-0.047186	0.393580	-0.509397	-0.275579	-0.093420	-2.761754
2009-09	0.041587	-0.274340	0.059355	-0.379426	0.364418	8.965208
2009-10	0.123025	0.182130	0.369218	0.783571	-0.622787	-12.040907
2009-11	-0.015452	-0.645380	0.100711	-0.189565	0.438153	1.906155
...	...	...	...	...	...	...
2024-12	0.061624	-0.198787	-0.094822	-0.856296	0.741467	1.772403
2025-01	-0.020875	-0.653224	-0.260872	1.166076	-0.846702	1.938516
2025-02	-0.061073	0.711218	0.479311	1.266449	-0.601381	1.326063
2025-03	0.043071	-0.044444	-0.145631	-0.393052	22.692755	-1.116873
2025-04	-0.163202	0.213276	0.360791	0.444580	1.036219	-0.460434

190 rows × 6 columns

In [11]:
```
df_XB = pd.read_excel("XB_merged.xlsx", parse_dates=["COT_Date"])
df_XB = (
    df_XB
      .set_index("COT_Date")
      .sort_index()
)

delta_cols = [
    "MM_Net_Chg", "Swap_Net_Chg", "Prod_Net_Chg",
    "Other_Net_Chg", "NonRep_Net_Chg"
]

# Monthly return: compound (1+weekly_return) across all weeks in month, then subtract 1
monthly_ret_XB = (
    df_XB["Return"].add(1)
      .resample("M")     # month-end; you can also use "ME" for explicit MonthEnd
      .prod()
      .sub(1)
      .rename("Ret_m")
)

# Monthly net position: sum the weekly Δ across the month
monthly_deltas = (
    df_XB[delta_cols]
      .resample("M")
      .sum()
      # rename columns to shorter Δ names if you like:
      .rename(columns={
          "MM_Net_Chg":"MM_Δ",
          "Swap_Net_Chg":"Swap_Δ",
          "Prod_Net_Chg":"Prod_Δ",
          "Other_Net_Chg":"Other_Δ",
          "NonRep_Net_Chg":"NonRep_Δ"
      })
)

# Combine into a single monthly DataFrame
df_m_XB = pd.concat([monthly_ret_XB, monthly_deltas], axis=1).dropna()
df_m_XB.index = df_m_XB.index.strftime("%Y-%m")
df_m_XB
```

```
<ipython-input-11-a94e08d6d561>:16: FutureWarning: 'M' is deprecated and will be removed in a future ver
sion, please use 'ME' instead.
  .resample("M")     # month-end; you can also use "ME" for explicit MonthEnd
<ipython-input-11-a94e08d6d561>:25: FutureWarning: 'M' is deprecated and will be removed in a future ver
sion, please use 'ME' instead.
  .resample("M")
```

Out[11]:

COT_Date	Ret_m	MM_Δ	Swap_Δ	Prod_Δ	Other_Δ	NonRep_Δ
2006-06	0.098388	0.085394	-0.015535	-0.059429	-0.720183	-3.724444
2006-07	-0.005116	-0.305180	0.123082	0.025504	2.092574	-0.708249
2006-08	-0.290760	0.540467	0.521055	0.513852	0.461198	-1.960889
2006-09	-0.109652	-0.105147	0.112198	0.110988	0.354401	6.659403
2006-10	0.025032	-0.786523	0.136267	-0.070082	0.784841	1.261165
...	...	...	...	...	...	...
2024-12	0.032715	-0.169931	1.748347	0.060756	-1.390064	0.706988
2025-01	0.035725	-0.095527	0.551764	0.055910	-0.024717	5.311146
2025-02	0.045355	-0.094025	0.370914	0.035155	1.573635	0.497210
2025-03	0.049357	2.686955	0.724052	0.014863	-0.101103	-1.568445
2025-04	-0.135114	-0.097621	0.139376	0.033349	0.447806	10.800775

227 rows × 6 columns

In [12]:
```
# wti combined df
inv_wti = inv_wti.set_index("COT_Date")
df_m_wti.index.name = "COT_Date"
df_combined_wti = df_m_wti.join(inv_wti, how="inner")
df_combined_wti
```

Out[12]:

COT_Date	Ret_m	MM_Δ	Swap_Δ	Prod_Δ	Other_Δ	NonRep_Δ	Inventory
2009-07	0.062323	0.000000	0.000000	0.000000	0.000000	0.000000	327.2
2009-08	-0.047186	0.393580	-0.509397	-0.275579	-0.093420	-2.761754	317.5
2009-09	0.041587	-0.274340	0.059355	-0.379426	0.364418	8.965208	317.1
2009-10	0.123025	0.182130	0.369218	0.783571	-0.622787	-12.040907	314.4
2009-11	-0.015452	-0.645380	0.100711	-0.189565	0.438153	1.906155	318.8
...	...	...	...	...	...	...	...
2024-12	0.061624	-0.198787	-0.094822	-0.856296	0.741467	1.772403	413.7
2025-01	-0.020875	-0.653224	-0.260872	1.166076	-0.846702	1.938516	418.8
2025-02	-0.061073	0.711218	0.479311	1.266449	-0.601381	1.326063	435.2
2025-03	0.043071	-0.044444	-0.145631	-0.393052	22.692755	-1.116873	441.9
2025-04	-0.163202	0.213276	0.360791	0.444580	1.036219	-0.460434	453.1

190 rows × 7 columns

In [13]:
```
# gasoline combined df
inv_xb = inv_xb.set_index("COT_Date")
df_m_XB.index.name = "COT_Date"
df_combined_XB = df_m_XB.join(inv_xb, how="inner")
df_combined_XB
```

Out[13]:

COT_Date	Ret_m	MM_Δ	Swap_Δ	Prod_Δ	Other_Δ	NonRep_Δ	Inventory
2006-06	0.098388	0.085394	-0.015535	-0.059429	-0.720183	-3.724444	213.3
2006-07	-0.005116	-0.305180	0.123082	0.025504	2.092574	-0.708249	208.9
2006-08	-0.290760	0.540467	0.521055	0.513852	0.461198	-1.960889	209.0
2006-09	-0.109652	-0.105147	0.112198	0.110988	0.354401	6.659403	214.1
2006-10	0.025032	-0.786523	0.136267	-0.070082	0.784841	1.261165	204.6
...	...	...	...	...	...	...	...
2024-12	0.032715	-0.169931	1.748347	0.060756	-1.390064	0.706988	238.6
2025-01	0.035725	-0.095527	0.551764	0.055910	-0.024717	5.311146	251.1
2025-02	0.045355	-0.094025	0.370914	0.035155	1.573635	0.497210	241.1
2025-03	0.049357	2.686955	0.724052	0.014863	-0.101103	-1.568445	237.7
2025-04	-0.135114	-0.097621	0.139376	0.033349	0.447806	10.800775	229.9

227 rows × 7 columns

In [15]:
```
# rolling z-scores over 12 months
net_cols = ["MM_Δ","Swap_Δ","Prod_Δ","Other_Δ","NonRep_Δ","Inventory"]

window = 12

def add_z(df):
    for col in net_cols:
        m = df[col].rolling(window, min_periods=6).mean()
        s = df[col].rolling(window, min_periods=6).std()
        df[col + "_z"] = (df[col] - m) / s
    return df

df_combined_wti = add_z(df_combined_wti)
df_combined_wti
```

Out[15]:

COT_Date	Ret_m	MM_Δ	Swap_Δ	Prod_Δ	Other_Δ	NonRep_Δ	Inventory	MM_Δ_z	Swap_Δ_z	Prod_Δ_z	Other_Δ_z	NonRep_Δ
2009-07	0.062323	0.000000	0.000000	0.000000	0.000000	0.000000	327.2	NaN	NaN	NaN	NaN	Na
2009-08	-0.047186	0.393580	-0.509397	-0.275579	-0.093420	-2.761754	317.5	NaN	NaN	NaN	NaN	Na
2009-09	0.041587	-0.274340	0.059355	-0.379426	0.364418	8.965208	317.1	NaN	NaN	NaN	NaN	Na
2009-10	0.123025	0.182130	0.369218	0.783571	-0.622787	-12.040907	314.4	NaN	NaN	NaN	NaN	Na
2009-11	-0.015452	-0.645380	0.100711	-0.189565	0.438153	1.906155	318.8	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	
2024-12	0.061624	-0.198787	-0.094822	-0.856296	0.741467	1.772403	413.7	-0.823959	-0.467756	0.088563	0.867864	-0.2590
2025-01	-0.020875	-0.653224	-0.260872	1.166076	-0.846702	1.938516	418.8	-1.872985	-0.851003	0.349923	-1.485423	-0.2620
2025-02	-0.061073	0.711218	0.479311	1.266449	-0.601381	1.326063	435.2	1.619082	0.710775	0.359887	-0.978182	-0.2627
2025-03	0.043071	-0.044444	-0.145631	-0.393052	22.692755	-1.116873	441.9	-0.345920	-0.939470	0.130160	3.158208	-0.2764
2025-04	-0.163202	0.213276	0.360791	0.444580	1.036219	-0.460434	453.1	0.461143	0.702844	0.163597	-0.165490	-0.2775

190 rows × 13 columns

In [17]:
```
df_combined_XB = add_z(df_combined_XB)
df_combined_XB
```

Out[17]:

COT_Date	Ret_m	MM_Δ	Swap_Δ	Prod_Δ	Other_Δ	NonRep_Δ	Inventory	MM_Δ_z	Swap_Δ_z	Prod_Δ_z	Other_Δ_z	NonRep_Δ
2006-06	0.098388	0.085394	-0.015535	-0.059429	-0.720183	-3.724444	213.3	NaN	NaN	NaN	NaN	Na
2006-07	-0.005116	-0.305180	0.123082	0.025504	2.092574	-0.708249	208.9	NaN	NaN	NaN	NaN	Na
2006-08	-0.290760	0.540467	0.521055	0.513852	0.461198	-1.960889	209.0	NaN	NaN	NaN	NaN	Na
2006-09	-0.109652	-0.105147	0.112198	0.110988	0.354401	6.659403	214.1	NaN	NaN	NaN	NaN	Na
2006-10	0.025032	-0.786523	0.136267	-0.070082	0.784841	1.261165	204.6	NaN	NaN	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	
2024-12	0.032715	-0.169931	1.748347	0.060756	-1.390064	0.706988	238.6	-0.593561	2.250355	0.158338	0.329734	0.27792
2025-01	0.035725	-0.095527	0.551764	0.055910	-0.024717	5.311146	251.1	-0.457544	0.533880	0.131740	0.482008	0.54087
2025-02	0.045355	-0.094025	0.370914	0.035155	1.573635	0.497210	241.1	-0.451023	0.239565	0.034428	0.620810	0.23215
2025-03	0.049357	2.686955	0.724052	0.014863	-0.101103	-1.568445	237.7	2.427694	0.645634	-0.001496	0.475035	0.11038
2025-04	-0.135114	-0.097621	0.139376	0.033349	0.447806	10.800775	229.9	-0.504047	-0.111129	0.072624	0.407459	0.81791

227 rows × 13 columns

In [24]:

```python
# Specifycolumn names
signal_cols  = ["MM_Δ_z", "Swap_Δ_z", "Prod_Δ_z", "Other_Δ_z", "NonRep_Δ_z"]
storage_col  = "Inventory_z"
horizons     = [1, 2, 3]

def run_horizons(df, label):
    """
    For each horizon h, regress Ret_m+h on:
      1) Inventory_z only
      2) signals + Inventory_z
      3) signals + Inventory_z + signal×Inventory interactions
    Prints R² and coefficient tables for each.
    Returns a dict of fitted models.
    """
    models = {}
    for h in horizons:
        # 2) Define the dependent variable: h-months-ahead return
        y = df["Ret_m"].shift(-h)

        # 3) Build a working DataFrame with signals, storage, and y
        data = df[signal_cols + [storage_col]].copy()
        data["y"] = y
        data = data.dropna()

        # 4a) Inventory only
        X1 = sm.add_constant(data[[storage_col]])
        m1 = sm.OLS(data["y"], X1).fit(cov_type="HAC", cov_kwds={"maxlags":1})
        print(f"\n--- {label}  t+{h} months: Inventory only (R² = {m1.rsquared:.4f}) ---")
        print(m1.summary().tables[1])

        # 4b) Signals + Inventory
        X2 = sm.add_constant(data[signal_cols + [storage_col]])
        m2 = sm.OLS(data["y"], X2).fit(cov_type="HAC", cov_kwds={"maxlags":1})
        print(f"\n--- {label}  t+{h} months: Signals + Inventory (R² = {m2.rsquared:.4f}) ---")
        print(m2.summary().tables[1])

        # 4c) Add interaction terms: signal_z × Inventory_z
        for col in signal_cols:
            data[f"{col}_x_{storage_col}"] = data[col] * data[storage_col]
        inter_cols = [f"{col}_x_{storage_col}" for col in signal_cols]

        X3 = sm.add_constant(data[signal_cols + [storage_col] + inter_cols])
        m3 = sm.OLS(data["y"], X3).fit(cov_type="HAC", cov_kwds={"maxlags":1})
        print(f"\n--- {label}  t+{h} months: Interactions (R² = {m3.rsquared:.4f}) ---")
        print(m3.summary().tables[1])

        # 5) Store models
        models[h] = {"inv": m1, "sig+stor": m2, "int": m3}

    return models
```

In [25]:
```
models_wti = run_horizons(df_combined_wti, "WTI Crude")
models_wti
```

--- WTI Crude t+1 months: Inventory only (R² = 0.0002) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0039	0.008	0.519	0.604	−0.011	0.019
Inventory_z	−0.0011	0.008	−0.149	0.882	−0.016	0.014

--- WTI Crude t+1 months: Signals + Inventory (R² = 0.0147) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0030	0.007	0.399	0.690	−0.012	0.017
MM_Δ_z	−0.0002	0.006	−0.033	0.974	−0.012	0.011
Swap_Δ_z	−0.0094	0.008	−1.167	0.243	−0.025	0.006
Prod_Δ_z	−0.0029	0.008	−0.354	0.723	−0.019	0.013
Other_Δ_z	−0.0050	0.008	−0.666	0.505	−0.020	0.010
NonRep_Δ_z	−0.0031	0.006	−0.492	0.623	−0.015	0.009
Inventory_z	−0.0013	0.007	−0.192	0.848	−0.015	0.012

--- WTI Crude t+1 months: Interactions (R² = 0.0688) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0024	0.007	0.339	0.734	−0.011	0.016
MM_Δ_z	−0.0002	0.006	−0.030	0.976	−0.013	0.012
Swap_Δ_z	−0.0070	0.009	−0.796	0.426	−0.024	0.010
Prod_Δ_z	−0.0017	0.007	−0.261	0.794	−0.015	0.011
Other_Δ_z	−0.0014	0.006	−0.227	0.821	−0.014	0.011
NonRep_Δ_z	−0.0011	0.006	−0.192	0.848	−0.013	0.010
Inventory_z	−0.0036	0.006	−0.590	0.555	−0.015	0.008
MM_Δ_z_x_Inventory_z	−0.0001	0.006	−0.021	0.984	−0.011	0.011
Swap_Δ_z_x_Inventory_z	−0.0057	0.006	−0.929	0.353	−0.018	0.006
Prod_Δ_z_x_Inventory_z	−0.0026	0.008	−0.303	0.762	−0.019	0.014
Other_Δ_z_x_Inventory_z	−0.0182	0.005	−3.585	0.000	−0.028	−0.008
NonRep_Δ_z_x_Inventory_z	−0.0031	0.004	−0.765	0.444	−0.011	0.005

--- WTI Crude t+2 months: Inventory only (R² = 0.0003) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0039	0.008	0.495	0.621	−0.011	0.019
Inventory_z	0.0015	0.007	0.218	0.827	−0.012	0.015

--- WTI Crude t+2 months: Signals + Inventory (R² = 0.0058) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0032	0.008	0.402	0.688	−0.012	0.019
MM_Δ_z	−0.0046	0.007	−0.700	0.484	−0.017	0.008
Swap_Δ_z	0.0003	0.008	0.040	0.968	−0.016	0.016
Prod_Δ_z	−0.0007	0.008	−0.090	0.928	−0.017	0.015
Other_Δ_z	−0.0032	0.008	−0.416	0.677	−0.018	0.012
NonRep_Δ_z	−0.0055	0.010	−0.560	0.575	−0.025	0.014
Inventory_z	0.0013	0.007	0.184	0.854	−0.012	0.015

--- WTI Crude t+2 months: Interactions (R² = 0.0654) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0037	0.008	0.492	0.623	−0.011	0.019
MM_Δ_z	−0.0098	0.007	−1.377	0.169	−0.024	0.004
Swap_Δ_z	0.0096	0.007	1.289	0.198	−0.005	0.024
Prod_Δ_z	−0.0034	0.009	−0.380	0.704	−0.021	0.014
Other_Δ_z	−0.0004	0.007	−0.049	0.961	−0.015	0.014
NonRep_Δ_z	−0.0036	0.010	−0.371	0.710	−0.023	0.015
Inventory_z	−0.0005	0.006	−0.088	0.930	−0.012	0.011
MM_Δ_z_x_Inventory_z	0.0041	0.006	0.690	0.490	−0.008	0.016
Swap_Δ_z_x_Inventory_z	−0.0191	0.008	−2.419	0.016	−0.035	−0.004
Prod_Δ_z_x_Inventory_z	0.0052	0.006	0.883	0.377	−0.006	0.017
Other_Δ_z_x_Inventory_z	−0.0121	0.007	−1.835	0.066	−0.025	0.001
NonRep_Δ_z_x_Inventory_z	−0.0084	0.007	−1.220	0.222	−0.022	0.005

--- WTI Crude t+3 months: Inventory only (R² = 0.0015) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0034	0.008	0.424	0.672	−0.012	0.019
Inventory_z	0.0033	0.006	0.523	0.601	−0.009	0.016

--- WTI Crude t+3 months: Signals + Inventory (R² = 0.0385) ---

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0041      0.008      0.514      0.607      -0.012       0.020
MM_Δ_z         0.0042      0.007      0.647      0.518      -0.009       0.017
Swap_Δ_z      -0.0105      0.008     -1.274      0.203      -0.027       0.006
Prod_Δ_z       0.0093      0.008      1.213      0.225      -0.006       0.024
Other_Δ_z      0.0058      0.007      0.887      0.375      -0.007       0.019
NonRep_Δ_z     0.0141      0.007      1.912      0.056      -0.000       0.029
Inventory_z    0.0049      0.006      0.758      0.449      -0.008       0.017
==============================================================================

--- WTI Crude  t+3 months: Interactions (R² = 0.1025) ---
==========================================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                      0.0046      0.008      0.582      0.561      -0.011       0.020
MM_Δ_z                    -0.0005      0.008     -0.065      0.948      -0.015       0.014
Swap_Δ_z                  -0.0035      0.008     -0.448      0.654      -0.019       0.012
Prod_Δ_z                   0.0104      0.009      1.169      0.242      -0.007       0.028
Other_Δ_z                  0.0069      0.005      1.307      0.191      -0.003       0.017
NonRep_Δ_z                 0.0129      0.007      1.765      0.078      -0.001       0.027
Inventory_z                0.0044      0.006      0.756      0.450      -0.007       0.016
MM_Δ_z_x_Inventory_z      -0.0019      0.006     -0.320      0.749      -0.013       0.009
Swap_Δ_z_x_Inventory_z    -0.0152      0.005     -2.884      0.004      -0.025      -0.005
Prod_Δ_z_x_Inventory_z    -0.0018      0.006     -0.317      0.751      -0.013       0.009
Other_Δ_z_x_Inventory_z    0.0048      0.004      1.161      0.246      -0.003       0.013
NonRep_Δ_z_x_Inventory_z  -0.0099      0.004     -2.238      0.025      -0.019      -0.001
==========================================================================================
```

Out[25]:
```
{1: {'inv': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66c88e650>,
  'sig+stor': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66c886510>,
  'int': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66c8a3650>},
 2: {'inv': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66c8cc310>,
  'sig+stor': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66c89f750>,
  'int': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66f2cce10>},
 3: {'inv': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66cc6ee90>,
  'sig+stor': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66c8734d0>,
  'int': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66c8f9650>}}
```

In [30]:
```
models_xb = run_horizons(df_combined_XB, "GBOB Gasoline")
models_xb
```

--- GBOB Gasoline t+1 months: Inventory only (R² = 0.1125) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0067	0.007	0.934	0.351	−0.007	0.021
Inventory_z	0.0370	0.007	5.223	0.000	0.023	0.051

--- GBOB Gasoline t+1 months: Signals + Inventory (R² = 0.1419) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0072	0.007	1.021	0.307	−0.007	0.021
MM_Δ_z	−0.0121	0.010	−1.213	0.225	−0.032	0.007
Swap_Δ_z	−0.0010	0.007	−0.144	0.885	−0.014	0.012
Prod_Δ_z	0.0255	0.009	2.788	0.005	0.008	0.043
Other_Δ_z	−0.0023	0.006	−0.420	0.675	−0.013	0.009
NonRep_Δ_z	−0.0015	0.007	−0.222	0.824	−0.015	0.012
Inventory_z	0.0350	0.007	4.743	0.000	0.021	0.049

--- GBOB Gasoline t+1 months: Interactions (R² = 0.1747) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0061	0.007	0.848	0.396	−0.008	0.020
MM_Δ_z	−0.0139	0.010	−1.424	0.154	−0.033	0.005
Swap_Δ_z	−0.0019	0.007	−0.291	0.771	−0.015	0.011
Prod_Δ_z	0.0268	0.009	2.979	0.003	0.009	0.044
Other_Δ_z	−0.0008	0.006	−0.143	0.886	−0.012	0.011
NonRep_Δ_z	−0.0006	0.006	−0.097	0.923	−0.013	0.012
Inventory_z	0.0366	0.007	4.901	0.000	0.022	0.051
MM_Δ_z_x_Inventory_z	−0.0173	0.013	−1.337	0.181	−0.043	0.008
Swap_Δ_z_x_Inventory_z	−0.0005	0.007	−0.077	0.939	−0.014	0.013
Prod_Δ_z_x_Inventory_z	−0.0024	0.010	−0.236	0.813	−0.022	0.017
Other_Δ_z_x_Inventory_z	0.0094	0.007	1.377	0.169	−0.004	0.023
NonRep_Δ_z_x_Inventory_z	0.0068	0.007	1.038	0.299	−0.006	0.020

--- GBOB Gasoline t+2 months: Inventory only (R² = 0.0209) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0078	0.008	1.023	0.306	−0.007	0.023
Inventory_z	0.0160	0.009	1.865	0.062	−0.001	0.033

--- GBOB Gasoline t+2 months: Signals + Inventory (R² = 0.0407) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0071	0.008	0.924	0.355	−0.008	0.022
MM_Δ_z	0.0073	0.011	0.635	0.526	−0.015	0.030
Swap_Δ_z	0.0031	0.008	0.407	0.684	−0.012	0.018
Prod_Δ_z	0.0028	0.011	0.249	0.803	−0.019	0.025
Other_Δ_z	−0.0124	0.006	−2.028	0.043	−0.024	−0.000
NonRep_Δ_z	−0.0052	0.007	−0.755	0.450	−0.019	0.008
Inventory_z	0.0169	0.008	2.177	0.030	0.002	0.032

--- GBOB Gasoline t+2 months: Interactions (R² = 0.0596) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0090	0.007	1.207	0.227	−0.006	0.024
MM_Δ_z	0.0039	0.012	0.325	0.745	−0.020	0.028
Swap_Δ_z	0.0029	0.007	0.401	0.689	−0.011	0.017
Prod_Δ_z	0.0043	0.012	0.368	0.713	−0.019	0.027
Other_Δ_z	−0.0157	0.007	−2.368	0.018	−0.029	−0.003
NonRep_Δ_z	−0.0081	0.007	−1.179	0.238	−0.022	0.005
Inventory_z	0.0176	0.008	2.132	0.033	0.001	0.034
MM_Δ_z_x_Inventory_z	−0.0091	0.015	−0.595	0.552	−0.039	0.021
Swap_Δ_z_x_Inventory_z	−0.0052	0.009	−0.588	0.557	−0.023	0.012
Prod_Δ_z_x_Inventory_z	0.0028	0.014	0.199	0.843	−0.025	0.030
Other_Δ_z_x_Inventory_z	−0.0092	0.008	−1.094	0.274	−0.026	0.007
NonRep_Δ_z_x_Inventory_z	−0.0104	0.007	−1.531	0.126	−0.024	0.003

--- GBOB Gasoline t+3 months: Inventory only (R² = 0.0000) ---

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0082	0.008	1.049	0.294	−0.007	0.024
Inventory_z	−0.0007	0.008	−0.087	0.931	−0.017	0.015

--- GBOB Gasoline t+3 months: Signals + Inventory (R² = 0.0289) ---

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0099      0.008      1.265      0.206      -0.005       0.025
MM_Δ_z        -0.0108      0.012     -0.902      0.367      -0.034       0.013
Swap_Δ_z       0.0044      0.011      0.381      0.703      -0.018       0.027
Prod_Δ_z       0.0152      0.011      1.321      0.186      -0.007       0.038
Other_Δ_z      0.0139      0.008      1.817      0.069      -0.001       0.029
NonRep_Δ_z    -0.0024      0.007     -0.347      0.728      -0.016       0.011
Inventory_z   -0.0051      0.009     -0.589      0.556      -0.022       0.012
==============================================================================

--- GBOB Gasoline  t+3 months: Interactions (R² = 0.0998) ---
==========================================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                      0.0050      0.008      0.623      0.533      -0.011       0.021
MM_Δ_z                    -0.0068      0.011     -0.616      0.538      -0.029       0.015
Swap_Δ_z                   0.0043      0.010      0.431      0.667      -0.015       0.024
Prod_Δ_z                   0.0125      0.011      1.165      0.244      -0.009       0.034
Other_Δ_z                  0.0161      0.008      2.011      0.044       0.000       0.032
NonRep_Δ_z                -0.0019      0.007     -0.260      0.795      -0.016       0.012
Inventory_z               -0.0027      0.008     -0.330      0.741      -0.019       0.014
MM_Δ_z_x_Inventory_z       0.0123      0.013      0.968      0.333      -0.013       0.037
Swap_Δ_z_x_Inventory_z     0.0253      0.011      2.341      0.019       0.004       0.046
Prod_Δ_z_x_Inventory_z    -0.0171      0.010     -1.716      0.086      -0.037       0.002
Other_Δ_z_x_Inventory_z    0.0101      0.010      1.015      0.310      -0.009       0.029
NonRep_Δ_z_x_Inventory_z   0.0030      0.007      0.434      0.665      -0.011       0.017
==========================================================================================
```

Out[30]:
```
{1: {'inv': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66c8ed250>,
  'sig+stor': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66b66d910>,
  'int': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66ac575d0>},
 2: {'inv': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66afa6290>,
  'sig+stor': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66b633d10>,
  'int': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66af4add0>},
 3: {'inv': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66afd1310>,
  'sig+stor': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66ac4e5d0>,
  'int': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66ac52610>}}
```

In [26]:

```python
from sklearn.linear_model import LassoCV
from sklearn.preprocessing import StandardScaler

def feature_select_and_fit(df, label, h, p_thresh=0.05):
    """
    For horizon h, build the full interaction design matrix,
    then apply:
      a) backward-elimination by p-value
      b) LassoCV selection
    Print R² and coeff table for each.
    Return both fitted models.
    """
    # dependent: h-ahead return
    y = df["Ret_m"].shift(-h)

    # build base data
    data = df[signal_cols + [storage_col]].copy()
    data["y"] = y
    data = data.dropna()

    # build interaction terms
    for col in signal_cols:
        data[f"{col}_x_{storage_col}"] = data[col] * data[storage_col]
    inter_cols = [f"{c}_x_{storage_col}" for c in signal_cols]

    # full X matrix
    X_full = sm.add_constant(data[signal_cols + [storage_col] + inter_cols])
    y_full = data["y"]

    # --- a) Backward-elimination by p-value ----------------------------
    X_be = X_full.copy()
    while True:
        m = sm.OLS(y_full, X_be).fit(cov_type="HAC", cov_kwds={"maxlags":1})
        pvals = m.pvalues.drop("const")
        worst_p = pvals.max()
        if worst_p > p_thresh:
            X_be = X_be.drop(columns=[pvals.idxmax()])
        else:
            break
    be_model = sm.OLS(y_full, X_be).fit(cov_type="HAC", cov_kwds={"maxlags":1})

    # --- b) LassoCV selection ------------------------------------------
    # scale features (exclude constant)
    X_noc = X_full.drop(columns="const")
    scaler = StandardScaler()
    Xs = scaler.fit_transform(X_noc)
    lasso = LassoCV(cv=5).fit(Xs, y_full)
    coef = pd.Series(lasso.coef_, index=X_noc.columns)
    selected = coef[coef != 0].index.tolist()
    X_lasso = sm.add_constant(X_noc[selected])
    lasso_model = sm.OLS(y_full, X_lasso).fit(cov_type="HAC", cov_kwds={"maxlags":1})

    # --- Print results -------------------------------------------------
    print(f"\n=== {label} t+{h}-month Feature Selection ===")

    print(f"\n-- Backward Elimination (p>{p_thresh} removed) R²={be_model.rsquared:.4f} --")
    print("Features:", X_be.columns.tolist())
    print(be_model.summary().tables[1])

    print(f"\n-- LassoCV Selection R²={lasso_model.rsquared:.4f} --")
    print("Features:", ["const"] + selected)
    print(lasso_model.summary().tables[1])

    return {"backward": be_model, "lasso": lasso_model}
```

In [27]:
```
models_wti_fs = {h: feature_select_and_fit(df_combined_wti, "WTI Crude",    h) for h in horizons}
models_wti_fs
```

```
=== WTI Crude t+1-month Feature Selection ===

-- Backward Elimination (p>0.05 removed) R²=0.0469 --
Features: ['const', 'Other_Δ_z_x_Inventory_z']
==============================================================================================
                               coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                        0.0028      0.008      0.373      0.709      -0.012       0.018
Other_Δ_z_x_Inventory_z     -0.0164      0.005     -3.194      0.001      -0.027      -0.006
==============================================================================================

-- LassoCV Selection R²=0.0577 --
Features: ['const', 'Swap_Δ_z', 'Other_Δ_z_x_Inventory_z']
==============================================================================================
                               coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                        0.0022      0.007      0.295      0.768      -0.012       0.017
Swap_Δ_z                    -0.0102      0.008     -1.313      0.189      -0.025       0.005
Other_Δ_z_x_Inventory_z     -0.0165      0.005     -3.138      0.002      -0.027      -0.006
==============================================================================================

=== WTI Crude t+2-month Feature Selection ===

-- Backward Elimination (p>0.05 removed) R²=0.0000 --
Features: ['const']
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0041      0.008      0.517      0.605      -0.011       0.020
==============================================================================

-- LassoCV Selection R²=0.0000 --
Features: ['const']
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0041      0.008      0.517      0.605      -0.011       0.020
==============================================================================

=== WTI Crude t+3-month Feature Selection ===

-- Backward Elimination (p>0.05 removed) R²=0.0829 --
Features: ['const', 'NonRep_Δ_z', 'Swap_Δ_z_x_Inventory_z', 'NonRep_Δ_z_x_Inventory_z']
==============================================================================================
                               coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                        0.0044      0.008      0.587      0.557      -0.010       0.019
NonRep_Δ_z                   0.0140      0.007      2.081      0.037       0.001       0.027
Swap_Δ_z_x_Inventory_z      -0.0174      0.004     -4.191      0.000      -0.025      -0.009
NonRep_Δ_z_x_Inventory_z    -0.0101      0.004     -2.547      0.011      -0.018      -0.002
==============================================================================================

-- LassoCV Selection R²=0.0829 --
Features: ['const', 'NonRep_Δ_z', 'Swap_Δ_z_x_Inventory_z', 'NonRep_Δ_z_x_Inventory_z']
==============================================================================================
                               coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                        0.0044      0.008      0.587      0.557      -0.010       0.019
NonRep_Δ_z                   0.0140      0.007      2.081      0.037       0.001       0.027
Swap_Δ_z_x_Inventory_z      -0.0174      0.004     -4.191      0.000      -0.025      -0.009
NonRep_Δ_z_x_Inventory_z    -0.0101      0.004     -2.547      0.011      -0.018      -0.002
==============================================================================================
```

Out[27]:
```
{1: {'backward': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66c911910>,
  'lasso': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66c8a6b90>},
 2: {'backward': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66b5bea90>,
  'lasso': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66bc99050>},
 3: {'backward': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66b631c50>,
  'lasso': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66bc30c10>}}
```

In [31]:
```
models_xb_fs = {h: feature_select_and_fit(df_combined_XB, "GBOB Gasoline",    h) for h in horizons}
models_xb_fs
```

```
=== GBOB Gasoline t+1-month Feature Selection ===

-- Backward Elimination (p>0.05 removed) R²=0.1535 --
Features: ['const', 'Prod_Δ_z', 'Inventory_z', 'MM_Δ_z_x_Inventory_z']
================================================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                    0.0070      0.007      1.013      0.311      -0.007       0.021
Prod_Δ_z                 0.0182      0.006      2.982      0.003       0.006       0.030
Inventory_z              0.0367      0.007      5.064      0.000       0.023       0.051
MM_Δ_z_x_Inventory_z    -0.0173      0.008     -2.095      0.036      -0.033      -0.001
================================================================================================

-- LassoCV Selection R²=0.1743 --
Features: ['const', 'MM_Δ_z', 'Prod_Δ_z', 'Inventory_z', 'MM_Δ_z_x_Inventory_z', 'Prod_Δ_z_x_Inventory_
z', 'Other_Δ_z_x_Inventory_z', 'NonRep_Δ_z_x_Inventory_z']
================================================================================================
                                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                          0.0061      0.007      0.853      0.393      -0.008       0.020
MM_Δ_z                        -0.0132      0.009     -1.446      0.148      -0.031       0.005
Prod_Δ_z                       0.0262      0.008      3.115      0.002       0.010       0.043
Inventory_z                    0.0361      0.007      4.959      0.000       0.022       0.050
MM_Δ_z_x_Inventory_z          -0.0168      0.012     -1.384      0.166      -0.041       0.007
Prod_Δ_z_x_Inventory_z        -0.0026      0.009     -0.285      0.776      -0.021       0.015
Other_Δ_z_x_Inventory_z        0.0096      0.006      1.510      0.131      -0.003       0.022
NonRep_Δ_z_x_Inventory_z       0.0070      0.007      1.053      0.292      -0.006       0.020
================================================================================================

=== GBOB Gasoline t+2-month Feature Selection ===

-- Backward Elimination (p>0.05 removed) R²=0.0332 --
Features: ['const', 'Other_Δ_z', 'Inventory_z']
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0069      0.008      0.897      0.370      -0.008       0.022
Other_Δ_z       -0.0125      0.006     -2.072      0.038      -0.024      -0.001
Inventory_z      0.0177      0.008      2.098      0.036       0.001       0.034
================================================================================

-- LassoCV Selection R²=0.0209 --
Features: ['const', 'Inventory_z']
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0078      0.008      1.023      0.306      -0.007       0.023
Inventory_z      0.0160      0.009      1.865      0.062      -0.001       0.033
================================================================================

=== GBOB Gasoline t+3-month Feature Selection ===

-- Backward Elimination (p>0.05 removed) R²=0.0571 --
Features: ['const', 'Swap_Δ_z_x_Inventory_z']
================================================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                    0.0031      0.008      0.378      0.705      -0.013       0.019
Swap_Δ_z_x_Inventory_z   0.0243      0.011      2.155      0.031       0.002       0.046
================================================================================================

-- LassoCV Selection R²=0.0571 --
Features: ['const', 'Swap_Δ_z_x_Inventory_z']
================================================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                    0.0031      0.008      0.378      0.705      -0.013       0.019
Swap_Δ_z_x_Inventory_z   0.0243      0.011      2.155      0.031       0.002       0.046
================================================================================================
```

Out[31]:
```
{1: {'backward': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66ac56250>,
  'lasso': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66c91ce50>},
 2: {'backward': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66ac45350>,
  'lasso': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66ac7a850>},
 3: {'backward': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66ac4e290>,
  'lasso': <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7bf66ac4fe50>}}
```

In [ ]:

In [28]:

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score


def run_random_forest(df, label):
    """
    For each horizon h, fit a RandomForestRegressor to predict Ret_m+h from:
      • signal_cols
      • storage_col
      • all interactions signal×storage
    Prints in-sample R² and feature importances.
    Returns a dict of fitted RF models.
    """
    rf_models = {}
    for h in horizons:
        # Build dependent variable y = Ret_m shifted by −h
        y = df["Ret_m"].shift(-h)

        # Build feature DataFrame with signals, storage, and interactions
        X = df[signal_cols + [storage_col]].copy()
        for col in signal_cols:
            X[f"{col}_x_{storage_col}"] = X[col] * X[storage_col]
        X["y"] = y
        X = X.dropna()

        # Separate features and target
        y_train = X.pop("y")
        X_train = X

        # Fit Random Forest
        rf = RandomForestRegressor(
            n_estimators=200,
            min_samples_leaf=5,
            random_state=0,
            n_jobs=-1
        )
        rf.fit(X_train, y_train)

        # In-sample R²
        y_pred = rf.predict(X_train)
        r2 = r2_score(y_train, y_pred)
        print(f"\n--- {label} t+{h} months Random Forest − R² = {r2:.4f} ---")

        # Feature importances
        imps = pd.Series(rf.feature_importances_, index=X_train.columns)
        imps = imps.sort_values(ascending=False)
        print(imps)

        # Store the model
        rf_models[h] = rf

    return rf_models
```

In [29]:
```
models_wti_rf = run_random_forest(df_combined_wti, "WTI Crude")
models_wti_rf
```

```
--- WTI Crude t+1 months Random Forest — R² = 0.5166 ---
Other_Δ_z_x_Inventory_z     0.215297
Swap_Δ_z                    0.101875
Prod_Δ_z                    0.087671
NonRep_Δ_z_x_Inventory_z    0.086360
Swap_Δ_z_x_Inventory_z      0.085893
NonRep_Δ_z                  0.080904
MM_Δ_z_x_Inventory_z        0.076902
MM_Δ_z                      0.074376
Prod_Δ_z_x_Inventory_z      0.070159
Inventory_z                 0.066438
Other_Δ_z                   0.054126
dtype: float64

--- WTI Crude t+2 months Random Forest — R² = 0.5025 ---
Swap_Δ_z_x_Inventory_z      0.174551
NonRep_Δ_z                  0.145911
Other_Δ_z_x_Inventory_z     0.135240
NonRep_Δ_z_x_Inventory_z    0.116181
Prod_Δ_z                    0.075735
Swap_Δ_z                    0.075071
Prod_Δ_z_x_Inventory_z      0.073427
MM_Δ_z                      0.060914
Other_Δ_z                   0.058872
MM_Δ_z_x_Inventory_z        0.045151
Inventory_z                 0.038947
dtype: float64

--- WTI Crude t+3 months Random Forest — R² = 0.5281 ---
MM_Δ_z_x_Inventory_z        0.142565
NonRep_Δ_z                  0.141931
Swap_Δ_z_x_Inventory_z      0.116028
Swap_Δ_z                    0.101313
Prod_Δ_z                    0.078894
Prod_Δ_z_x_Inventory_z      0.078077
Other_Δ_z_x_Inventory_z     0.073722
NonRep_Δ_z_x_Inventory_z    0.069756
Inventory_z                 0.067505
MM_Δ_z                      0.065829
Other_Δ_z                   0.064380
dtype: float64
```

Out[29]:
```
{1: RandomForestRegressor(min_samples_leaf=5, n_estimators=200, n_jobs=-1,
                          random_state=0),
 2: RandomForestRegressor(min_samples_leaf=5, n_estimators=200, n_jobs=-1,
                          random_state=0),
 3: RandomForestRegressor(min_samples_leaf=5, n_estimators=200, n_jobs=-1,
                          random_state=0)}
```

In [32]:
```
models_xb_rf = run_random_forest(df_combined_XB, "GBOB Gasoline")
models_xb_rf
```

```
--- GBOB Gasoline t+1 months Random Forest — R² = 0.5618 ---
Inventory_z                 0.312087
Prod_Δ_z                    0.103692
NonRep_Δ_z                  0.079382
MM_Δ_z_x_Inventory_z        0.077818
NonRep_Δ_z_x_Inventory_z    0.070164
Prod_Δ_z_x_Inventory_z      0.068516
Other_Δ_z_x_Inventory_z     0.064146
Swap_Δ_z_x_Inventory_z      0.062511
Other_Δ_z                   0.058552
Swap_Δ_z                    0.056317
MM_Δ_z                      0.046815
dtype: float64

--- GBOB Gasoline t+2 months Random Forest — R² = 0.5336 ---
NonRep_Δ_z_x_Inventory_z    0.132866
Swap_Δ_z                    0.131815
Inventory_z                 0.128174
Other_Δ_z                   0.093101
Prod_Δ_z                    0.091564
NonRep_Δ_z                  0.087023
Prod_Δ_z_x_Inventory_z      0.080788
MM_Δ_z                      0.078610
Swap_Δ_z_x_Inventory_z      0.062027
MM_Δ_z_x_Inventory_z        0.057883
Other_Δ_z_x_Inventory_z     0.056149
dtype: float64

--- GBOB Gasoline t+3 months Random Forest — R² = 0.5275 ---
Swap_Δ_z_x_Inventory_z      0.129668
Prod_Δ_z                    0.112027
Prod_Δ_z_x_Inventory_z      0.100295
Other_Δ_z                   0.096522
Other_Δ_z_x_Inventory_z     0.091350
Inventory_z                 0.089114
MM_Δ_z                      0.087043
MM_Δ_z_x_Inventory_z        0.078099
NonRep_Δ_z_x_Inventory_z    0.075682
Swap_Δ_z                    0.072136
NonRep_Δ_z                  0.068064
dtype: float64
```

Out[32]:
```
{1: RandomForestRegressor(min_samples_leaf=5, n_estimators=200, n_jobs=-1,
                          random_state=0),
 2: RandomForestRegressor(min_samples_leaf=5, n_estimators=200, n_jobs=-1,
                          random_state=0),
 3: RandomForestRegressor(min_samples_leaf=5, n_estimators=200, n_jobs=-1,
                          random_state=0)}
```