FIN41660 - Crypto Forecasting (OLS with BTC Benchmark)

This notebook loads your **2 years of daily crypto prices** (BTC, ADA, ATOM, DOGE, DOT), builds **log returns**, and fits **OLS** with optional **cross-crypto benchmark lags** (e.g., BTC lags as exogenous predictors for altcoins). It performs **walk-forward** (expanding) out-of-sample evaluation and saves **metrics, features, predictions, and plots** for the report.

Where to put the CSVs

- **Option A (recommended):** Keep your current absolute Windows paths (the defaults below).
- **Option B:** Upload the files into a local `./data/` folder next to this notebook and set `USE_ABSOLUTE_PATHS = False`.

Files expected (either absolute or under `./data/`):

- `bitcoin_data.csv`, `cardano_data.csv`, `cosmos_data.csv`, `dogecoin_data.csv`, `polkadot_data.csv`

> The loader is **robust to your header format** (dates in a column named *Price*, extra *Ticker/Date* lines).

In [1]:
```
%pip install -q pandas numpy statsmodels arch matplotlib seaborn
```

```
Note: you may need to restart the kernel to use updated packages.
WARNING: Ignoring invalid distribution ~tatsmodels (C:\Users\khain\anaconda3\Lib
\site-packages)
WARNING: Ignoring invalid distribution ~tatsmodels (C:\Users\khain\anaconda3\Lib
\site-packages)
WARNING: Ignoring invalid distribution ~tatsmodels (C:\Users\khain\anaconda3\Lib
\site-packages)
```

In [8]:
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA  # optional baseline
from arch import arch_model                    # optional baseline
from IPython.display import display

sns.set(style='whitegrid', context='talk')
plt.rcParams['figure.figsize'] = (12, 5)
```

In [9]:
```
# === Configuration: choose A or B ===
USE_ABSOLUTE_PATHS = True   # A) keep your Windows absolute paths
# USE_ABSOLUTE_PATHS = False  # B) if you upload CSVs into ./data/ next to this

if USE_ABSOLUTE_PATHS:
```

```
    PATHS = {
        'bitcoin':  r"C:\Users\khain\Documents\Documents\UCD\Financial Econometr
        'cardano':  r"C:\Users\khain\Documents\Documents\UCD\Financial Econometr
        'cosmos':   r"C:\Users\khain\Documents\Documents\UCD\Financial Econometr
        'dogecoin': r"C:\Users\khain\Documents\Documents\UCD\Financial Econometr
        'polkadot': r"C:\Users\khain\Documents\Documents\UCD\Financial Econometr
    }
else:
    PATHS = {
        'bitcoin':  r'./data/bitcoin_data.csv',
        'cardano':  r'./data/cardano_data.csv',
        'cosmos':   r'./data/cosmos_data.csv',
        'dogecoin': r'./data/dogecoin_data.csv',
        'polkadot': r'./data/polkadot_data.csv',
    }

ALTCOINS  = ['cardano','cosmos','dogecoin','polkadot']
BENCHMARK = 'bitcoin'    # or 'ethereum' if you have it
EXOG_LAGS = [1,2,5,10]   # benchmark lagged returns used in OLS

OUT_FIG = os.path.join('report','outputs','figures')
OUT_CSV = os.path.join('report','outputs','csv')
os.makedirs(OUT_FIG, exist_ok=True)
os.makedirs(OUT_CSV, exist_ok=True)

# Quick file existence check
for k, p in PATHS.items():
    print(f"{k:9s} -> {'FOUND' if os.path.exists(p) else 'MISSING'}: {p}")
```

```
bitcoin   -> FOUND: C:\Users\khain\Documents\Documents\UCD\Financial Econometrics
\Group FInal Project\data\bitcoin_data.csv
cardano   -> FOUND: C:\Users\khain\Documents\Documents\UCD\Financial Econometrics
\Group FInal Project\data\cardano_data.csv
cosmos    -> FOUND: C:\Users\khain\Documents\Documents\UCD\Financial Econometrics
\Group FInal Project\data\cosmos_data.csv
dogecoin  -> FOUND: C:\Users\khain\Documents\Documents\UCD\Financial Econometrics
\Group FInal Project\data\dogecoin_data.csv
polkadot  -> FOUND: C:\Users\khain\Documents\Documents\UCD\Financial Econometrics
\Group FInal Project\data\polkadot_data.csv
```

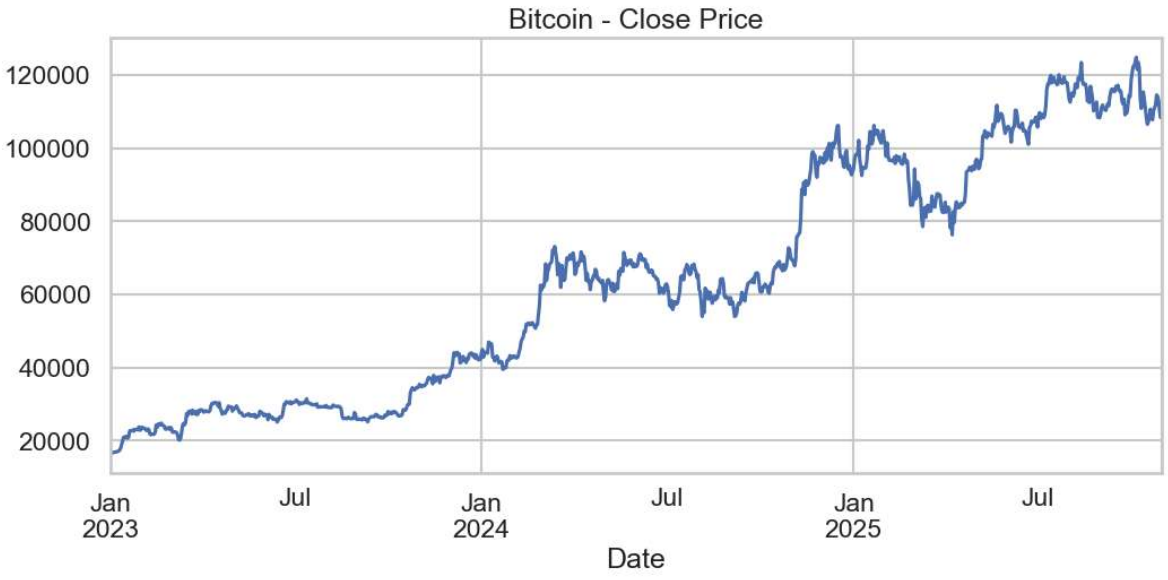
In [10]:

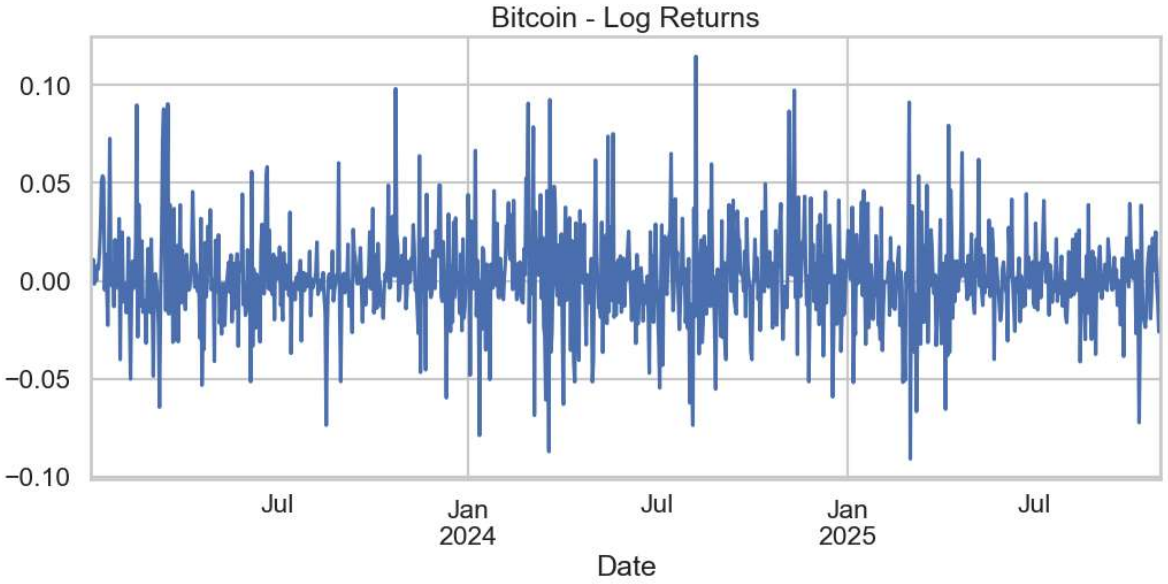
```
# --- Robust CSV loader (handles your header quirks) ---
def _detect_and_clean_header(df_raw: pd.DataFrame) -> pd.DataFrame:
    df = df_raw.copy()
    first_col = df.columns[0]
    # Drop metadata lines 'Ticker,...' and 'Date,,,,,'
    drop_mask = df[first_col].astype(str).str.upper().isin(['TICKER','DATE'])
    df = df.loc[~drop_mask].copy()
    # Map 'Price' -> 'Date' if dates live there
    if first_col.lower() == 'price':
        df.rename(columns={first_col: 'Date'}, inplace=True)
    return df

def _resolve_cols(df: pd.DataFrame):
    date_candidates  = ['Date','date','Time','time','Timestamp','timestamp']
    close_candidates = ['Close','Adj Close','close','adj_close']
    date_col  = next((c for c in date_candidates  if c in df.columns), None)
    close_col = next((c for c in close_candidates if c in df.columns), None)
    if not date_col:  raise ValueError(f'No date column in {list(df.columns)}')
    if not close_col: raise ValueError(f'No close column in {list(df.columns)}')
    return date_col, close_col
```

```
def load_price_series(csv_path: str, dayfirst: bool = True) -> pd.DataFrame:
    df_raw = pd.read_csv(csv_path, dtype=str)
    df = _detect_and_clean_header(df_raw)
    date_col, close_col = _resolve_cols(df)
    df = df[[date_col, close_col]].copy()
    df[date_col] = pd.to_datetime(df[date_col], dayfirst=dayfirst, errors='coerc
    df[close_col] = (df[close_col].astype(str)
                                  .str.replace(',', '', regex=False)
                                  .str.replace(' ', '', regex=False))
    df[close_col] = pd.to_numeric(df[close_col], errors='coerce')
    df = df.dropna(subset=[date_col, close_col]).sort_values(date_col).set_index
    df.rename(columns={close_col: 'Close'}, inplace=True)
    return df

def compute_log_returns(price_df: pd.DataFrame) -> pd.DataFrame:
    df = price_df.copy()
    df['log_price'] = np.log(df['Close'])
    df['ret'] = df['log_price'].diff()
    return df.dropna()[['Close','log_price','ret']]
```

In [5]:
```
# --- Load all series & preview ---
prices, returns = {}, {}
for sym, path in PATHS.items():
    prices[sym]  = load_price_series(path)
    returns[sym] = compute_log_returns(prices[sym])
    print(sym, len(returns[sym]), 'observations')

# Quick charts for BTC
prices['bitcoin']['Close'].plot(title='Bitcoin - Close Price'); plt.show()
returns['bitcoin']['ret'].plot(title='Bitcoin - Log Returns'); plt.show()
```

```
bitcoin 1034 observations
cardano 407 observations
cosmos 407 observations
dogecoin 407 observations
polkadot 407 observations
```





In [6]:
```python
# --- OLS forecaster with optional benchmark (exogenous) lags ---
class OLSForecaster:
    def __init__(self, lag_list=None, exog_lag_list=None, use_newey_west=True, h
        self.lag_list = lag_list or [1,2,5,10]
        self.exog_lag_list = exog_lag_list
        self.use_newey_west = use_newey_west
        self.hac_maxlags = hac_maxlags
        self.model, self.columns_ = None, None

    @staticmethod
    def _metrics(actual: pd.Series, pred: pd.Series):
        resid = actual - pred
        rmse = float(np.sqrt(np.mean(resid**2)))
        mae  = float(np.mean(np.abs(resid)))
        mape = float(np.mean(np.abs(resid / (actual.replace(0, np.nan))))) * 100
        return {'RMSE': rmse, 'MAE': mae, 'MAPE_%': mape}

    def make_features(self, ret_df: pd.DataFrame, benchmark_ret: pd.Series | Non
        df = ret_df.copy()
        for L in self.lag_list:
            df[f'lag_{L}'] = df['ret'].shift(L)
        if (benchmark_ret is not None) and self.exog_lag_list:
            b = benchmark_ret.copy()
            df = df.join(b.rename('bench_ret'), how='left')
            for L in self.exog_lag_list:
                df[f'bench_lag_{L}'] = df['bench_ret'].shift(L)
        return df.dropna()

    def fit(self, feature_df: pd.DataFrame):
        y = feature_df['ret']
        X = feature_df[[c for c in feature_df.columns if c.startswith('lag_') or
        Xc = sm.add_constant(X)
        if self.use_newey_west:
            self.model = sm.OLS(y, Xc).fit(cov_type='HAC', cov_kwds={'maxlags':
        else:
            self.model = sm.OLS(y, Xc).fit()
        self.columns_ = Xc.columns
        return self.model

    def rolling_forecast(self, feature_df: pd.DataFrame, initial_train_frac: flo
        n = len(feature_df); n_train = int(np.floor(n * initial_train_frac))
```

```
        if n_train < 20: raise ValueError('Training window too small.')
        y = feature_df['ret']
        X = feature_df[[c for c in feature_df.columns if c.startswith('lag_') or
        preds, idxs = [], []
        for t in range(n_train, n):
            y_tr, X_tr = y.iloc[:t], X.iloc[:t]
            X_te = X.iloc[t:t+1]
            Xc_tr = sm.add_constant(X_tr)
            if self.use_newey_west:
                m = sm.OLS(y_tr, Xc_tr).fit(cov_type='HAC', cov_kwds={'maxlags':
            else:
                m = sm.OLS(y_tr, Xc_tr).fit()
            Xc_te = sm.add_constant(X_te).reindex(columns=Xc_tr.columns, fill_va
            preds.append(float(m.predict(Xc_te).iloc[0]))
            idxs.append(y.index[t])
        preds_df = pd.DataFrame({'ret': y.iloc[n_train:].values, 'pred_ret': pre
        preds_df['resid'] = preds_df['ret'] - preds_df['pred_ret']
        return preds_df, self._metrics(preds_df['ret'], preds_df['pred_ret'])
```

In [15]:

```
# --- Helpers: save CSVs & figures; run one symbol with benchmark ---

def save_df(df: pd.DataFrame, name: str):
    path = os.path.join(OUT_CSV, f'{name}.csv')
    df.to_csv(path, index=True)
    return path

def plot_series(series: pd.Series, title: str, ylabel: str, fname: str):
    ax = series.plot(title=title)
    ax.set_ylabel(ylabel); ax.set_xlabel('Date')
    plt.tight_layout()
    path = os.path.join(OUT_FIG, f'{fname}.png')
    plt.savefig(path, dpi=150); plt.show()
    return path

def plot_actual_vs_pred(preds_df: pd.DataFrame, title: str, fname: str):
    ax = preds_df[['ret','pred_ret']].plot(title=title)
    ax.set_ylabel('log return'); ax.set_xlabel('Date')
    plt.tight_layout()
    path = os.path.join(OUT_FIG, f'{fname}.png')
    plt.savefig(path, dpi=150); plt.show()
    return path

def plot_resid(resid: pd.Series, title: str, fname: str):
    fig, axes = plt.subplots(1,2, figsize=(14,5))
    resid.plot(ax=axes[0]); axes[0].set_title(title); axes[0].set_xlabel('Date')
    sns.histplot(resid, ax=axes[1], kde=True, color='steelblue'); axes[1].set_ti
    axes[1].set_xlabel('Residual'); axes[1].set_ylabel('Count')
    plt.tight_layout()
    path = os.path.join(OUT_FIG, f'{fname}.png')
    plt.savefig(path, dpi=150); plt.show()
    return path

def run_ols_with_benchmark(symbol: str, benchmark: str, exog_lags=None, use_newe
    if exog_lags is None: exog_lags = [1,2,5,10]
    ret_df = returns[symbol][['ret']].copy()
    bench_ret = None
    if (benchmark in returns) and (benchmark != symbol):
        aligned = ret_df.join(returns[benchmark][['ret']].rename(columns={'ret':
        ret_for_feats = pd.DataFrame({'ret': aligned['ret']}, index=aligned.inde
        bench_ret = aligned['bench_ret']
```

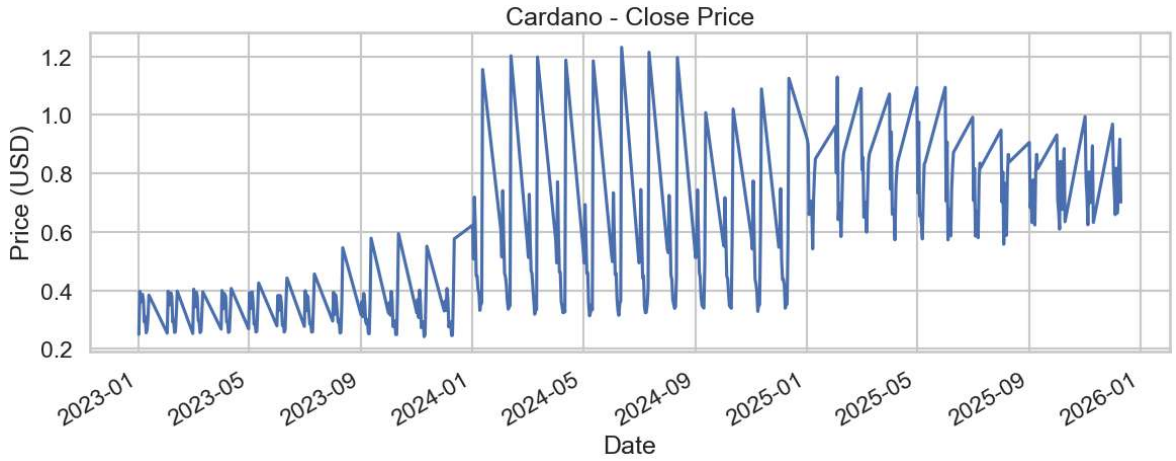
```
    else:
        ret_for_feats = ret_df
    ols = OLSForecaster(exog_lag_list=exog_lags, use_newey_west=use_newey)
    feats = ols.make_features(ret_for_feats, benchmark_ret=bench_ret)
    model = ols.fit(feats)
    preds_df, metrics = ols.rolling_forecast(feats)
```

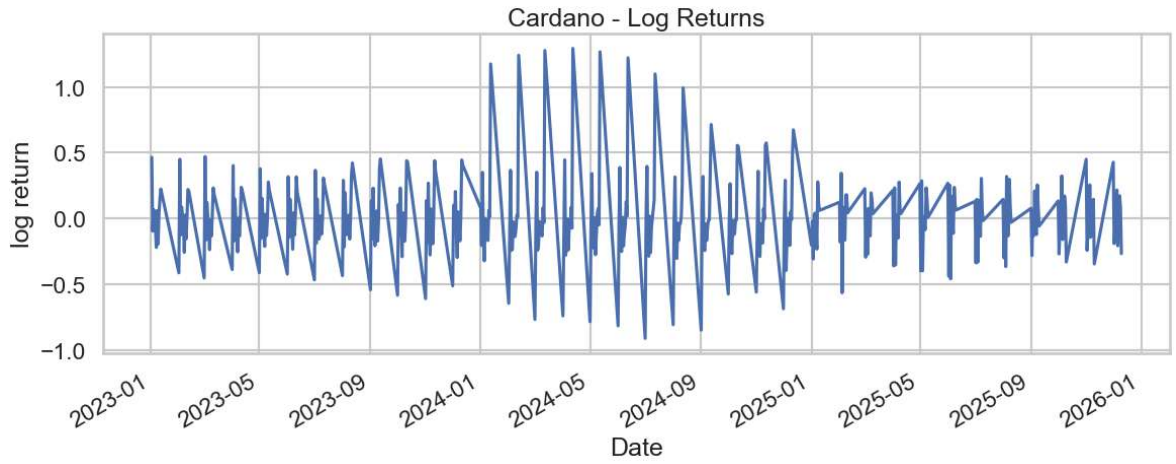
In [18]:

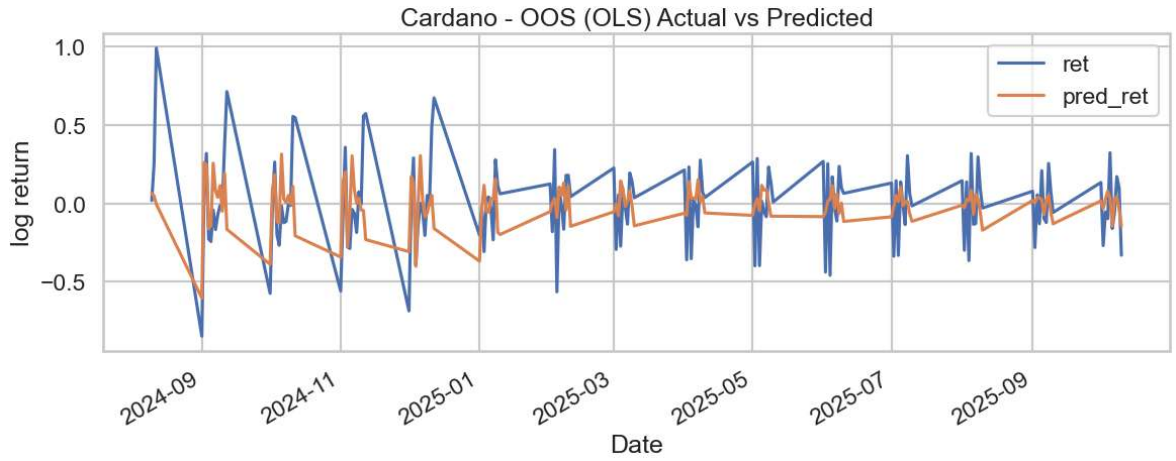
```
def run_ols_with_benchmark(symbol: str, benchmark: str, exog_lags=None, use_newe
    if exog_lags is None:
        exog_lags = [1,2,5,10]

    ret_df = returns[symbol][['ret']].copy()
    bench_ret = None

    if (benchmark in returns) and (benchmark != symbol):
        aligned = ret_df.join(returns[benchmark][['ret']].rename(columns={'ret':
        ret_for_feats = pd.DataFrame({'ret': aligned['ret']}, index=aligned.inde
        bench_ret = aligned['bench_ret']
    else:
        ret_for_feats = ret_df

    ols = OLSForecaster(exog_lag_list=exog_lags, use_newey_west=use_newey)
    feats = ols.make_features(ret_for_feats, benchmark_ret=bench_ret)
    model = ols.fit(feats)
    preds_df, metrics = ols.rolling_forecast(feats)

    # ----------------------------
    # Save outputs (now 'symbol' is defined in this scope)
    # ----------------------------
    save_df(prices[symbol],  f"{symbol}_price")
    save_df(returns[symbol], f"{symbol}_returns")
    save_df(feats,           f"{symbol}_ols_features")
    save_df(preds_df,        f"{symbol}_ols_oos")

    plot_series(prices[symbol]["Close"], f"{symbol.capitalize()} - Close Price",
    plot_series(returns[symbol]["ret"],  f"{symbol.capitalize()} - Log Returns",
    plot_actual_vs_pred(preds_df,        f"{symbol.capitalize()} - OOS (OLS) Act
    plot_resid(preds_df["resid"],        f"{symbol.capitalize()} - OOS Residuals

    print(f"[{symbol.upper()}] In-sample OLS summary:")
    print(model.summary())

    print("[{}] OOS metrics with benchmark={}, lags={}: {}".format(
        symbol.upper(), benchmark, exog_lags, metrics
    ))

    return model, feats, preds_df, metrics
```

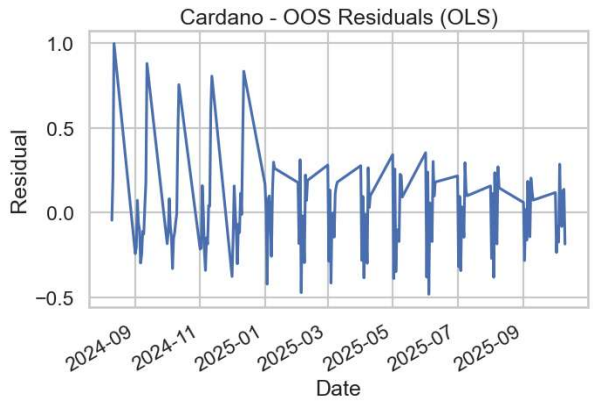
In [20]:

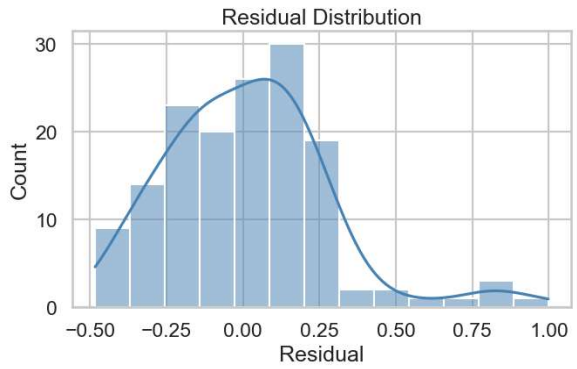
```
# --- Run OLS with BTC benchmark for all altcoins & summarize ---
summary_rows = []
for sym in ALTCOINS:
    model, feats, preds, metr = run_ols_with_benchmark(sym, BENCHMARK, exog_lags
    row = {'symbol': sym, **metr}
    summary_rows.append(row)
summary_df = pd.DataFrame(summary_rows).set_index('symbol')
display(summary_df.style.format({'RMSE':'{:.6f}','MAE':'{:.6f}','MAPE_%':'{:.3f}
```


Cardano - Close Price
Price (USD)
1.2
1.0
0.8
0.6
0.4
0.2
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
2025-01
2025-05
2025-09
2026-01
Date

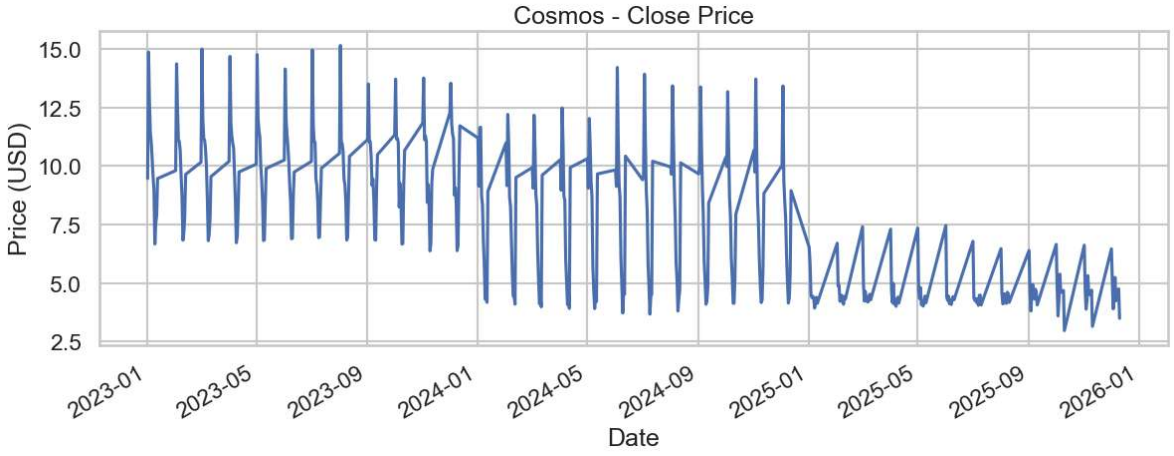

Cardano - Log Returns
log return
1.0
0.5
0.0
−0.5
−1.0
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
2025-01
2025-05
2025-09
2026-01
Date

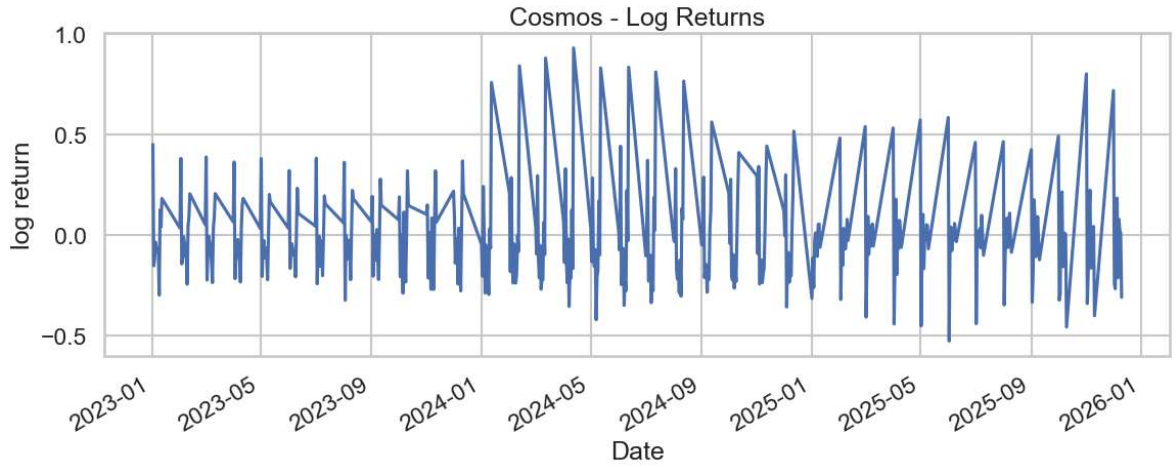

Cardano - OOS (OLS) Actual vs Predicted
log return
1.0
0.5
0.0
−0.5
ret
pred_ret
2024-09
2024-11
2025-01
2025-03
2025-05
2025-07
2025-09
Date

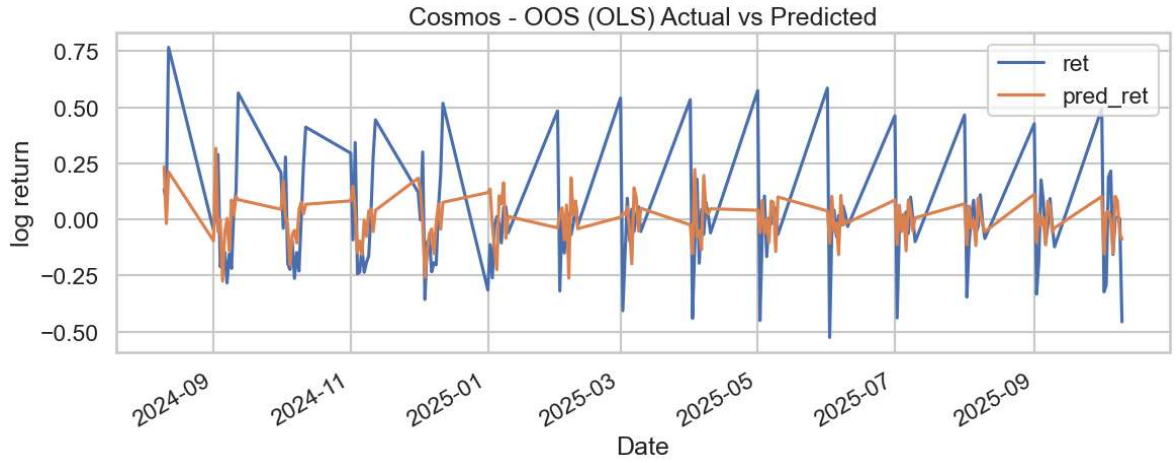

Cardano - OOS Residuals (OLS)
Residual
1.0
0.5
0.0
−0.5
2024-09
2024-11
2025-01
2025-03
2025-05
2025-07
2025-09
Date

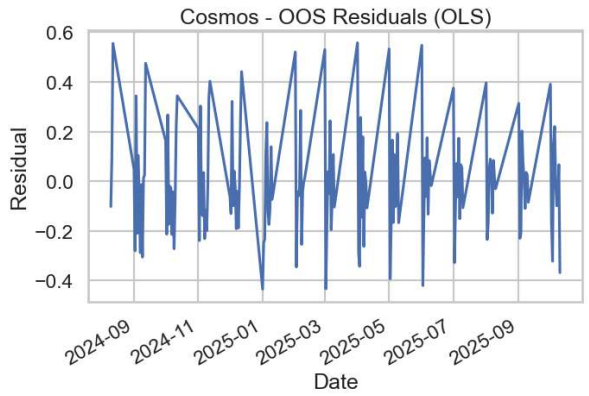

Residual Distribution
Count
30
20
10
0
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Residual

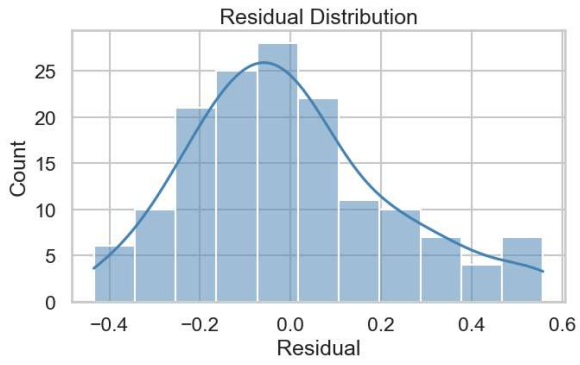
```
[CARDANO] In-sample OLS summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                    ret   R-squared:                       0.187
Model:                            OLS   Adj. R-squared:                  0.169
Method:                 Least Squares   F-statistic:                     13.94
Date:                Fri, 12 Dec 2025   Prob (F-statistic):           9.93e-18
Time:                        23:29:04   Log-Likelihood:                -54.280
No. Observations:                 377   AIC:                             126.6
Df Residuals:                     368   BIC:                             162.0
Df Model:                           8
Covariance Type:                  HAC
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0072      0.013      0.564      0.572      -0.018       0.032
lag_1         -0.4048      0.045     -9.035      0.000      -0.493      -0.317
lag_2         -0.2068      0.035     -5.897      0.000      -0.275      -0.138
lag_5         -0.1553      0.034     -4.609      0.000      -0.221      -0.089
lag_10        -0.0340      0.046     -0.740      0.459      -0.124       0.056
bench_lag_1   -0.0949      0.629     -0.151      0.880      -1.327       1.137
bench_lag_2   -0.2034      0.482     -0.422      0.673      -1.148       0.741
bench_lag_5   -0.5600      0.519     -1.078      0.281      -1.578       0.458
bench_lag_10  -0.7833      0.596     -1.315      0.189      -1.951       0.384
==============================================================================
Omnibus:                      159.458   Durbin-Watson:                   2.065
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              651.977
Skew:                           1.854   Prob(JB):                    2.66e-142
Kurtosis:                       8.268   Cond. No.                         40.0
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC) using
5 lags and without small sample correction
[CARDANO] OOS metrics with benchmark=bitcoin, lags=[1, 2, 5, 10]: {'RMSE': 0.2626
799623387049, 'MAE': 0.200082542933478, 'MAPE_%': 791.8410691930272}
```



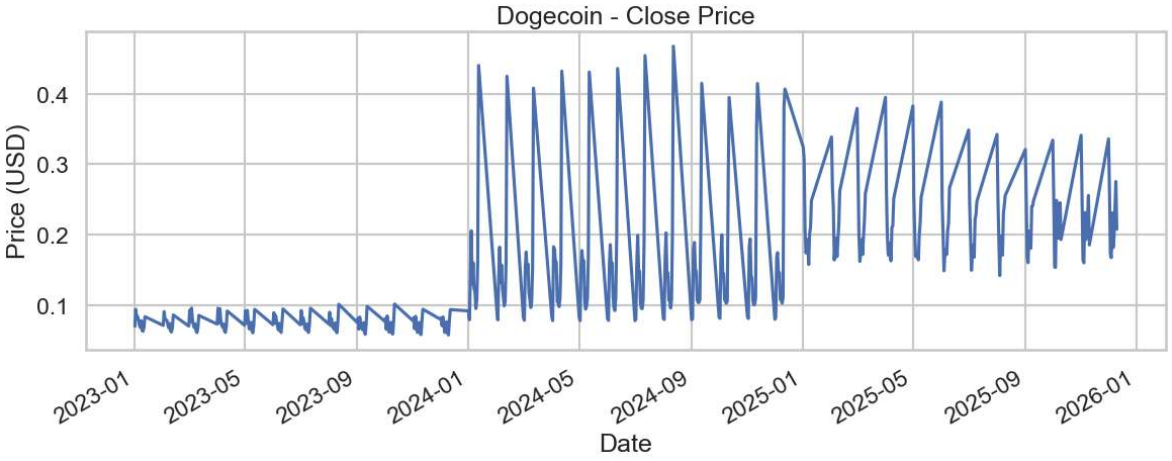

Cosmos - Log Returns
log return
1.0
0.5
0.0
−0.5
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
2025-01
2025-05
2025-09
2026-01
Date

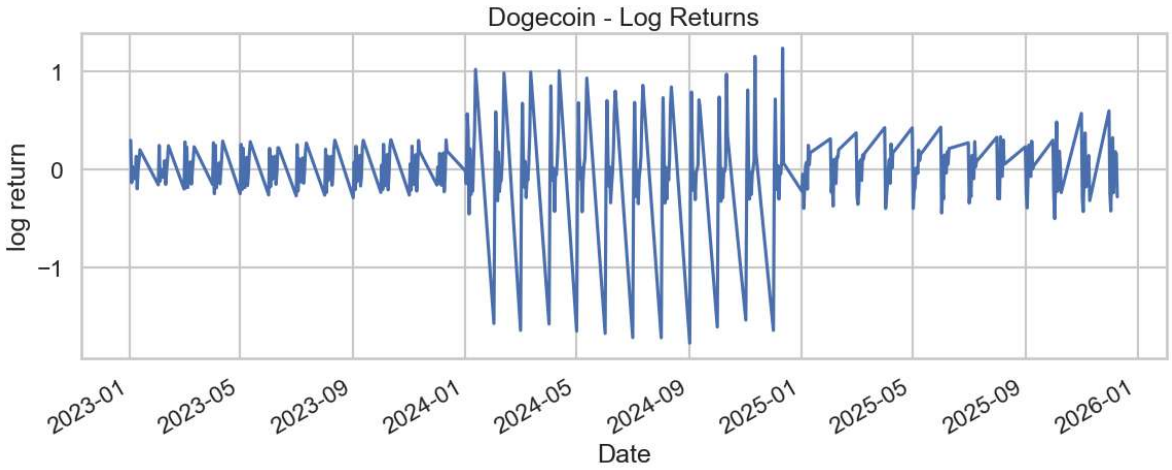

Cosmos - OOS (OLS) Actual vs Predicted
log return
0.75
0.50
0.25
0.00
−0.25
−0.50
ret
pred_ret
2024-09
2024-11
2025-01
2025-03
2025-05
2025-07
2025-09
Date

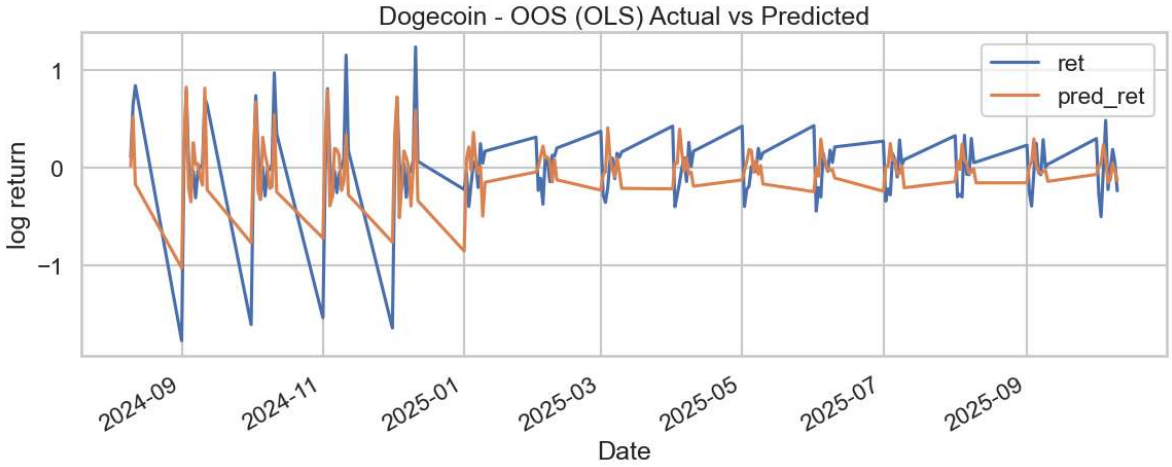

Cosmos - OOS Residuals (OLS)
Residual
0.6
0.4
0.2
0.0
−0.2
−0.4
2024-09
2024-11
2025-01
2025-03
2025-05
2025-07
2025-09
Date

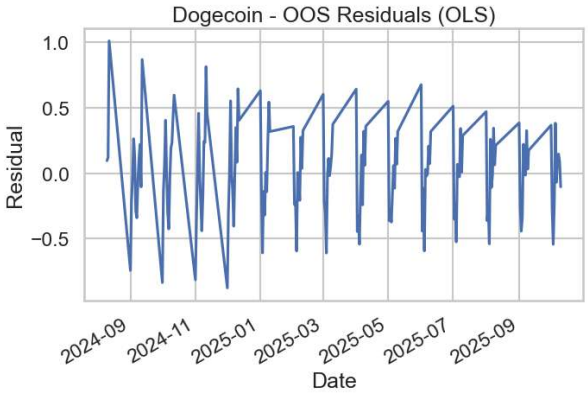

Residual Distribution
Count
25
20
15
10
5
0
−0.4
−0.2
0.0
0.2
0.4
0.6
Residual

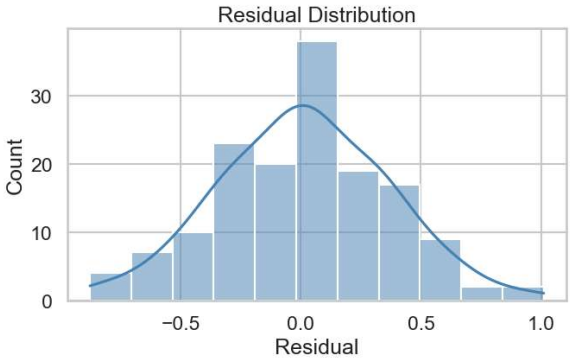
```
[COSMOS] In-sample OLS summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                    ret   R-squared:                       0.148
Model:                            OLS   Adj. R-squared:                  0.130
Method:                 Least Squares   F-statistic:                     13.14
Date:                Fri, 12 Dec 2025   Prob (F-statistic):           1.04e-16
Time:                        23:29:11   Log-Likelihood:                 45.737
No. Observations:                 377   AIC:                            -73.47
Df Residuals:                     368   BIC:                            -38.08
Df Model:                           8
Covariance Type:                  HAC
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          -0.0002      0.009     -0.026      0.979      -0.017       0.017
lag_1          -0.0889      0.039     -2.299      0.022      -0.165      -0.013
lag_2           0.1216      0.040      3.075      0.002       0.044       0.199
lag_5          -0.2522      0.052     -4.869      0.000      -0.354      -0.151
lag_10          0.2226      0.070      3.171      0.002       0.085       0.360
bench_lag_1    -0.4909      0.459     -1.070      0.284      -1.390       0.408
bench_lag_2     0.1787      0.421      0.424      0.672      -0.647       1.005
bench_lag_5    -0.1655      0.387     -0.428      0.669      -0.923       0.592
bench_lag_10   -0.8528      0.496     -1.718      0.086      -1.826       0.120
==============================================================================
Omnibus:                       74.967   Durbin-Watson:                   2.136
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              137.100
Skew:                           1.105   Prob(JB):                     1.69e-30
Kurtosis:                       4.961   Cond. No.                         39.9
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC) using
5 lags and without small sample correction
[COSMOS] OOS metrics with benchmark=bitcoin, lags=[1, 2, 5, 10]: {'RMSE': 0.22186
625323184947, 'MAE': 0.17422928518531744, 'MAPE_%': 438.5417402837172}
```



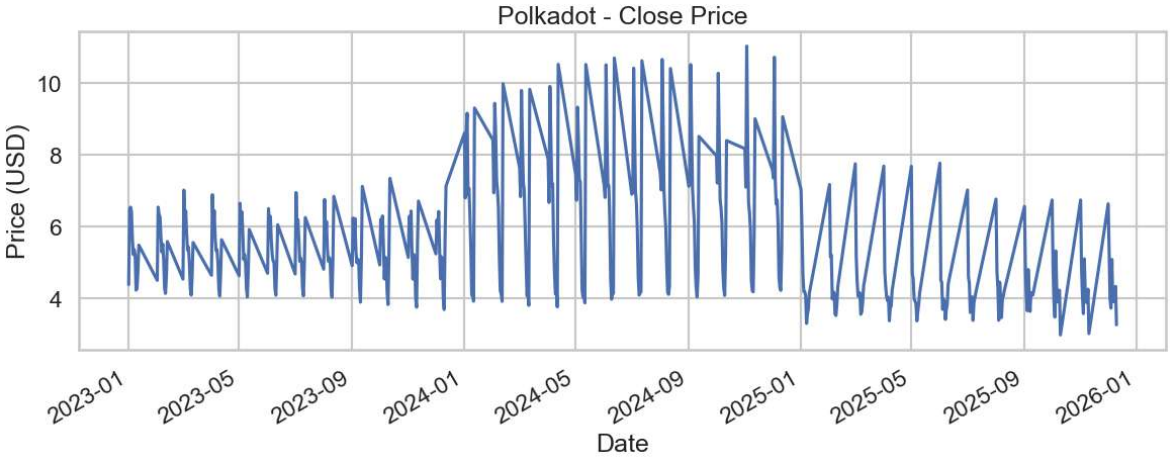

Dogecoin - Log Returns
log return
1
0
−1
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
2025-01
2025-05
2025-09
2026-01
Date

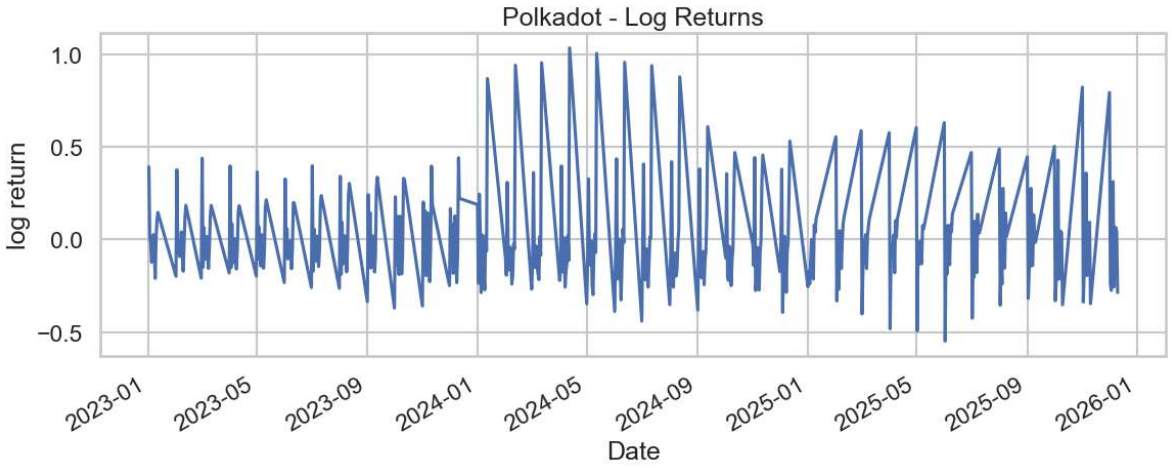

Dogecoin - OOS (OLS) Actual vs Predicted
ret
pred_ret
log return
1
0
−1
2024-09
2024-11
2025-01
2025-03
2025-05
2025-07
2025-09
Date

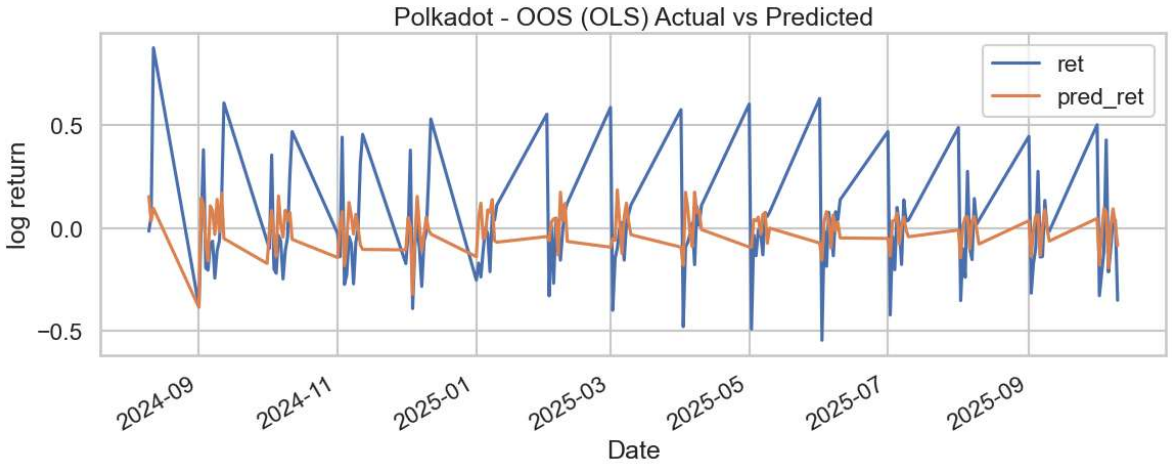

Dogecoin - OOS Residuals (OLS)
Residual
1.0
0.5
0.0
−0.5
2024-09
2024-11
2025-01
2025-03
2025-05
2025-07
2025-09
Date

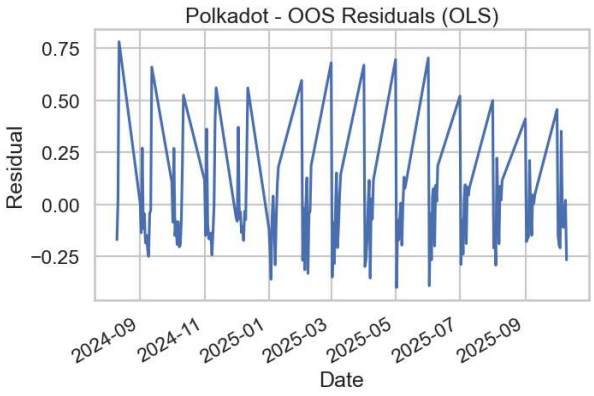

Residual Distribution
Count
30
20
10
0
−0.5
0.0
0.5
1.0
Residual

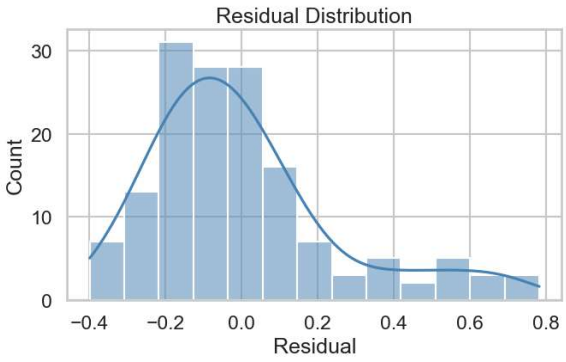
```
[DOGECOIN] In-sample OLS summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                    ret   R-squared:                       0.330
Model:                            OLS   Adj. R-squared:                  0.316
Method:                 Least Squares   F-statistic:                     23.09
Date:                Fri, 12 Dec 2025   Prob (F-statistic):           1.30e-28
Time:                        23:29:17   Log-Likelihood:                -118.67
No. Observations:                 377   AIC:                             255.3
Df Residuals:                     368   BIC:                             290.7
Df Model:                           8
Covariance Type:                  HAC
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0080      0.011      0.745      0.456      -0.013       0.029
lag_1         -0.2069      0.039     -5.281      0.000      -0.284      -0.130
lag_2         -0.4133      0.034    -12.334      0.000      -0.479      -0.348
lag_5         -0.0819      0.032     -2.544      0.011      -0.145      -0.019
lag_10        -0.3166      0.055     -5.717      0.000      -0.425      -0.208
bench_lag_1   -0.1198      0.702     -0.171      0.864      -1.495       1.255
bench_lag_2    0.1130      0.666      0.170      0.865      -1.192       1.418
bench_lag_5   -1.2380      0.836     -1.481      0.139      -2.876       0.400
bench_lag_10   0.3423      0.710      0.482      0.630      -1.050       1.734
==============================================================================
Omnibus:                       25.862   Durbin-Watson:                   2.425
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               88.638
Skew:                          -0.113   Prob(JB):                     5.66e-20
Kurtosis:                       5.365   Cond. No.                         40.1
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC) using
5 lags and without small sample correction
[DOGECOIN] OOS metrics with benchmark=bitcoin, lags=[1, 2, 5, 10]: {'RMSE': 0.353
72090253210453, 'MAE': 0.2777894641801047, 'MAPE_%': 439.77689773526134}
```




Polkadot - Log Returns
log return
1.0
0.5
0.0
−0.5
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09
2025-01
2025-05
2025-09
2026-01
Date


Polkadot - OOS (OLS) Actual vs Predicted
ret
pred_ret
log return
0.5
0.0
−0.5
2024-09
2024-11
2025-01
2025-03
2025-05
2025-07
2025-09
Date


Polkadot - OOS Residuals (OLS)
Residual
0.75
0.50
0.25
0.00
−0.25
2024-09
2024-11
2025-01
2025-03
2025-05
2025-07
2025-09
Date


Residual Distribution
Count
30
20
10
0
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
Residual

```
[POLKADOT] In-sample OLS summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                    ret   R-squared:                       0.109
Model:                            OLS   Adj. R-squared:                  0.090
Method:                 Least Squares   F-statistic:                     9.140
Date:                Fri, 12 Dec 2025   Prob (F-statistic):           1.75e-11
Time:                        23:29:24   Log-Likelihood:                 7.6969
No. Observations:                 377   AIC:                             2.606
Df Residuals:                     368   BIC:                             38.00
Df Model:                           8
Covariance Type:                  HAC
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0018      0.011      0.171      0.864      -0.019       0.023
lag_1         -0.2746      0.037     -7.362      0.000      -0.348      -0.201
lag_2         -0.1305      0.031     -4.197      0.000      -0.192      -0.070
lag_5         -0.1254      0.032     -3.912      0.000      -0.188      -0.063
lag_10         0.1029      0.073      1.401      0.161      -0.041       0.247
bench_lag_1   -0.3678      0.489     -0.752      0.452      -1.326       0.591
bench_lag_2   -0.1901      0.434     -0.438      0.661      -1.041       0.660
bench_lag_5   -0.4078      0.446     -0.915      0.360      -1.281       0.466
bench_lag_10  -0.7739      0.526     -1.471      0.141      -1.805       0.258
==============================================================================
Omnibus:                      132.119   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              363.705
Skew:                           1.681   Prob(JB):                     1.05e-79
Kurtosis:                       6.442   Cond. No.                         40.0
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity and autocorrelation robust (HAC) using
5 lags and without small sample correction
[POLKADOT] OOS metrics with benchmark=bitcoin, lags=[1, 2, 5, 10]: {'RMSE': 0.247
89745454273035, 'MAE': 0.1840184120323511, 'MAPE_%': 1008.5748211114552}
```

symbol	RMSE	MAE	MAPE_%
cardano	0.262680	0.200083	791.841
cosmos	0.221866	0.174229	438.542
dogecoin	0.353721	0.277789	439.777
polkadot	0.247897	0.184018	1008.575

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: