Kelompok 9

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FMIPA

Prediksi arah pergerakan QQQ menggunakan MLP

Tugas Akhir
Sains Data

May 25, 2026

Anggota Kelompok

Mochamad Alfath Aliandi

2406426271

Sofyan Shodiqin

2406423824

Putra Muhammad Rafy

2406412783

Utsman Izzudien

2406432122

Latar Belakang

Indeks QQQ (NASDAQ-100) memiliki volatilitas harian yang tinggi sehingga menyulitkan investor menentukan waktu masuk dan keluar pasar.

Banyak pendekatan prediksi yang ada hanya mengandalkan data harga QQQ itu sendiri, padahal informasi dari aset lain—seperti kurs bitcoin (BTC) atau indeks komoditas—dapat memberikan sinyal tambahan yang berharga.

Penelitian ini bertujuan membangun model Multilayer Perceptron (MLP) untuk memprediksi arah pergerakan harian QQQ dengan memanfaatkan data dari berbagai aset, serta membandingkan kontribusi masing-masing aset terhadap akurasi prediksi.

Fokus utama penelitian ini adalah pada akurasi arah (naik atau turun), karena dalam praktik trading, mengetahui arah pergerakan lebih penting daripada mengetahui besaran perubahan.

Rumusan Masalah

Bagaimana membangun model MLP optimal dengan data terbatas (~985 sampel) tanpa overfitting?

Seberapa akurat model dalam memprediksi magnitude dan arah pergerakan QQQ?

Aset mana yang memberikan informasi independen terbaik untuk prediksi QQQ?

Apakah model lebih baik dari prediksi naive (prediksi 0)?

Tujuan

Merancang pipeline data fitur stasioner (log return) dan memilih aset relevan

Membangun model MLP dengan arsitektur sederhana (32→16 neuron, dropout 0.1)

Mengevaluasi performa menggunakan metrik regresi (MAE, RMSE, R^2) dan arah (Directional Accuracy, uji binomial)

Proses Pengerjaan

01. Download Data
02. Stasioneritas Data Time-Series
03. Feature Engineering
04. Merge Data
05. Preprocessing Data
06. EDA
07. Feature Matrix Building
08. Train/Test Split
09. MLP
10. Hasil
11. Prediksi Harga
12. Kesimpulan

Download Data

```python
import yfinance as yf
import numpy as np
import pandas as pd
import time


class YFDownloader:
    def __init__(self, retries=3, delay=2, timeout=20):
        self.retries = retries
        self.delay   = delay
        self.timeout = timeout

    def detect_type(self, ticker):
        """Deteksi jenis aset dari format ticker."""
        if ticker.startswith("^"):
            return "index"
        elif ticker.endswith("=F"):
            return "commodity"
        elif ticker.endswith("=X"):
            return "forex"
        else:
            return "equity"

    def download(self, tickers, period="3y", interval="1d"):
        """
        Download data untuk daftar ticker.
        Return: dict {ticker: {"type": ..., "data": DataFrame}}
        """
        if isinstance(tickers, str):
            tickers = [tickers]
```

Kode untuk download data

```
=== GC=F (commodity) ===
  Shape: (1006, 15)
                 close         high          low         open  volume  \
Date
2026-05-21  4539.799805  4539.799805  4503.799805  4507.200195     426
2026-05-22  4521.000000  4530.299805  4519.100098  4519.500000     426

                 date       sma_20       ema_20        rsi       macd  \
Date
2026-05-21  2026-05-21  4620.709961  4623.758993  42.480380 -45.049922
2026-05-22  2026-05-22  4610.644971  4613.972422  50.147783 -47.695727

            macd_signal     bb_upper     bb_lower   returns  log_return
Date
2026-05-21   -34.641661  4774.905479  4466.514443  0.001876    0.001874
2026-05-22   -37.252474  4763.190200  4458.099741 -0.004141   -0.004150

=== CL=F (commodity) ===
  Shape: (1006, 15)
                close        high        low       open  volume        date  \
Date
2026-05-21  96.349998  102.660004  95.760002  98.949997  345482  2026-05-21
2026-05-22  96.599998   99.430000  94.730003  98.000000  345482  2026-05-22

                sma_20      ema_20        rsi      macd  macd_signal  \
Date
2026-05-21  100.874500  100.290153  43.728963  1.588486     1.991767
2026-05-22  100.984499   99.938710  37.828464  1.112853     1.815984
```

Contoh sampel data yang didapat

Dalam project ini, kami import data dari yfinance yaitu data historis dalam pasar keuangan yang merangkum pergerakan harga dan volume transaksi per hari, data dicatat selama 1004 hari

Stasioneritas Data Time-Series

Harga mentah (raw prices) bersifat non-stasioner atau memiliki tren (random walk). Jika langsung digunakan, model akan mengalami Covariate Shift (model "kaget" melihat rentang harga baru di masa depan). Untuk menghilangkan drift (tren), mengubah harga menjadi fitur yang kembali ke nilai rata-rata (mean-reverting)/stationer merupakan salah satu solusinya.

Untuk setiap aset yang digunakan, akan diekstrak fitur stationary untuk mengatasi masalah covariate shift (distribusi fitur train dan test berbeda) dengan:

Fitur	Rumus	Sifat
log_return	$\ln(P_t/P_{t-1})$	Rata-rata ≈ 0, tidak drift
rsi_norm	RSI / 100	Selalu dalam $[0, 1]$
bb_pct	$(P_t - BB_{lower})/(BB_{upper} - BB_{lower})$	Terbatas $[0, 1]$, mean-reverting
close_vs_sma	$(P_t/\text{SMA}_{20}) - 1$	Mean-reverting

Feature Engineering & Merge Data

```python
import pandas as pd

# —— Load CSV hasil Step 1 ——
qqq       = pd.read_csv("data_QQQ.csv",     index_col=0, parse_dates=Tru
alumunium = pd.read_csv("data_ALI.csv",     index_col=0, parse_dates=Tru
palladium = pd.read_csv("data_PA.csv",      index_col=0, parse_dates=Tru
copper    = pd.read_csv("data_HG.csv",      index_col=0, parse_dates=Tru
silver    = pd.read_csv("data_SI.csv",      index_col=0, parse_dates=Tru
platinum  = pd.read_csv("data_PL.csv",      index_col=0, parse_dates=Tru
lithium   = pd.read_csv("data_LIT.csv",     index_col=0, parse_dates=Tru
uranium   = pd.read_csv("data_URA.csv",     index_col=0, parse_dates=Tru
btc       = pd.read_csv("data_BTC-USD.csv", index_col=0, parse_dates=Tru
vix       = pd.read_csv("data_VIX.csv",     index_col=0, parse_dates=Tru


def prep(df, prefix):
    """
    Ubah raw OHLCV + indikator → 4 fitur stasioner.
    Harga mentah dibuang karena bersifat non-stasioner (tren).
    """
    close           = df['close']
    df['rsi_norm']  = df['rsi'] / 100
    band_width      = df['bb_upper'] - df['bb_lower'] + 1e-8   # epsilon
    df['bb_pct']    = (close - df['bb_lower']) / band_width
    df['close_vs_sma'] = (close / df['sma_20']) - 1

    cols = ['log_return', 'rsi_norm', 'bb_pct', 'close_vs_sma']
    return df[cols].rename(columns=lambda c: f"{prefix}_{c}")


dfs = [
    prep(qqq,       'qqq'),
    prep(vix,       'vix'),
    prep(alumunium, 'alumunium'),
    prep(palladium, 'palladium'),
    prep(copper,    'copper'),
    prep(silver,    'silver'),
    prep(platinum,  'platinum'),
    prep(lithium,   'lithium'),
    prep(uranium,   'uranium'),
    prep(btc,       'btc'),
```

Dipilih fitur yang stasioner agar dapat ditinjau oleh model

```python
dfs = [
    prep(qqq,       'qqq'),
    prep(vix,       'vix'),
    prep(alumunium, 'alumunium'),
    prep(palladium, 'palladium'),
    prep(copper,    'copper'),
    prep(silver,    'silver'),
    prep(platinum,  'platinum'),
    prep(lithium,   'lithium'),
    prep(uranium,   'uranium'),
    prep(btc,       'btc'),
]
```

komoditas tersebut dipilih karena menjadi penggerak **teknologi** . Sedangkan **BTC** , karena merupakan **mata uang digital** yang sering digunakan untuk transaksi. Ticker **QQQ** yang akan diprediksi.

```
Shape setelah merge : (1003, 10)
Missing values      : 1

Sample data:
```

Date	qqq_log_return	vix_log_return	alumunium_log_return	palladium_log_
2022-05-24	-0.021491	0.033492	-0.020232	
2022-05-25	0.013898	-0.037362	-0.009086	
2022-05-26	0.027330	-0.031146	-0.005368	

log return dipilih karena **hasilnya lebih baik dibandingkan** dengan **fitur stasioner lainnya**

Preprocessing Data

```
# ── 1. Hapus duplikat index ──────────────────────────────
n_dup = df.index.duplicated().sum()
df = df[~df.index.duplicated(keep='first')]
df = df.sort_index()
print(f"[1] Duplikat dihapus   : {n_dup} baris → shape: {df.shape}")
```

Memeriksa dan menghapus baris yang memiliki tanggal/waktu kembar, lalu mengurutkan data berdasarkan tanggal dari masa lalu ke masa depan.

```
# ── 2. Filter hanya hari kerja (weekday) ─────────────────
before = len(df)
df = df[df.index.dayofweek < 5]
print(f"[2] Weekend dihapus    : {before - len(df)} baris → shape: {df.shape}")
```

Filter Hari Kerja (Weekday Filter): Data disinkronkan hanya untuk hari Senin–Jumat untuk menyesuaikan kalender bursa saham (ekuitas), mengingat aset kripto (BTC) diperdagangkan 24/7.

```
# ── 3. Ganti Inf dengan NaN ──────────────────────────────
n_inf = np.isinf(df.values).sum()
df.replace([np.inf, -np.inf], np.nan, inplace=True)
print(f"[3] Inf → NaN          : {n_inf} nilai dikonversi")
```

Mendeteksi nilai tak hingga (inf atau -inf) yang biasanya muncul akibat pembagian dengan angka nol pada kalkulasi indikator teknikal, lalu mengubahnya menjadi nilai kosong (NaN).

```
# ── 4. Laporan missing values per kolom ──────────────────
missing = df.isna().sum()
missing_cols = missing[missing > 0]
print(f"[4] Kolom dengan NaN   : {len(missing_cols)} kolom")
if len(missing_cols) > 0:
    print(missing_cols.to_string())

# ── 5. Drop baris NaN (akibat indikator warm-up & Inf) ──
before = len(df)
df = df.dropna()
print(f"[5] Baris NaN dihapus  : {before - len(df)} baris → shape: {df.shape}")
```

Menghitung total nilai kosong per kolom, lalu menghapus secara permanen baris data yang tidak lengkap tersebut.

Preprocessing Data

```python
# ── 6. Clip outlier ekstrem per kolom (±5σ) ──────────────────
# Fitur non-bounded (log_return, close_vs_sma) mungkin punya nilai ekstrem
# saat crash pasar. Kita clip pada ±5 standar deviasi.
log_ret_cols  = [c for c in df.columns if 'log_return' in c]
cvs_cols      = [c for c in df.columns if 'close_vs_sma' in c]
clip_cols     = log_ret_cols + cvs_cols

n_clipped = 0
for col in clip_cols:
    mu, sigma = df[col].mean(), df[col].std()
    lo, hi    = mu - 5 * sigma, mu + 5 * sigma
    before_clip = ((df[col] < lo) | (df[col] > hi)).sum()
    df[col]   = df[col].clip(lower=lo, upper=hi)
    n_clipped += before_clip

print(f"[6] Outlier di-clip    : {n_clipped} nilai (±5σ pada log_return & close_vs_sma)")
```

Mendeteksi fitur log return dan jarak harga ke Moving Average yang melompat ekstrem melebihi +-5 standar deviasi saat terjadi crash pasar, lalu memotong nilainya agar kembali ke batas maksimal/minimal tersebut.

```python
# ── 7. Validasi range fitur bounded ──────────────────────────
rsi_cols = [c for c in df.columns if 'rsi_norm' in c]
bbp_cols = [c for c in df.columns if 'bb_pct' in c]

rsi_bad = sum((df[c] < 0).sum() + (df[c] > 1).sum() for c in rsi_cols)
bbp_out = sum((df[c] < -0.5).sum() + (df[c] > 1.5).sum() for c in bbp_cols)
print(f"[7] RSI di luar [0,1]  : {rsi_bad} nilai")
print(f"    bb_pct di luar [-0.5, 1.5]: {bbp_out} nilai (wajar untuk gap ekstrem)")

print(f"\n[✓] Shape akhir preprocessing: {df.shape}")
print(f"    Rentang tanggal: {df.index[0].date()} → {df.index[-1].date()}")
```

Mengecek kualitas nilai pada indikator teknikal yang memiliki batasan alami, seperti RSI (harus di rentang 0-1) dan Bollinger Bands %B.

Preprocessing Data

```
============================================================
PREPROCESSING PIPELINE
============================================================

[0] Shape awal          : (1003, 40)
[1] Duplikat dihapus    : 0 baris → shape: (1003, 40)
[2] Weekend dihapus     : 0 baris → shape: (1003, 40)
[3] Inf → NaN           : 0 nilai dikonversi
[4] Kolom dengan NaN    : 21 kolom
qqq_rsi_norm          13
qqq_bb_pct            18
vix_rsi_norm          13
vix_bb_pct            18
alumunium_rsi_norm    13
alumunium_bb_pct      18
palladium_rsi_norm    13
palladium_bb_pct      18
copper_rsi_norm       13
copper_bb_pct         18
silver_rsi_norm       13
silver_bb_pct         18
platinum_rsi_norm     13
platinum_bb_pct       18
lithium_rsi_norm      13
lithium_bb_pct        18
uranium_rsi_norm      13
uranium_bb_pct        18
btc_log_return         1
btc_rsi_norm           9
btc_bb_pct            13
[5] Baris NaN dihapus   : 18 baris → shape: (985, 40)
[6] Outlier di-clip     : 28 nilai (±5σ pada log_return & close_vs_sma)
[7] RSI di luar [0,1]   : 0 nilai
    bb_pct di luar [-0.5, 1.5]: 0 nilai (wajar untuk gap ekstrem)

[✓] Shape akhir preprocessing: (985, 40)
    Rentang tanggal: 2022-06-21 → 2026-05-22
```

Pada tahap preprocessing, integritas data diperiksa dengan melihat kemunculan nilai kosong (missing value) yang disebabkan oleh perbedaan kalender atau hari libur perdagangan dari berbagai instrumen lintas aset yang digabungkan.

Hasil pengecekan menunjukkan terdapat 18 baris data yang tidak lengkap, sehingga ke-18 baris tersebut langsung dihapus secara permanen menggunakan fungsi .dropna().

Langkah ini sangat krusial untuk menjamin keandalan kualitas data (data integrity) serta mencegah terjadinya eror komputasi seperti NaN atau infinity pada matriks pembobot ketika model MLP Neural Network dilatih.

Preprocessing Data

```
# ── Ringkasan statistik deskriptif fitur ──────────────────
print("\nStatistik deskriptif fitur:")
df.describe().round(4)
```

Statistik deskriptif fitur:

	qqq_log_return	qqq_rsi_norm	qqq_bb_pct	qqq_close_vs_sma	vix_log_return	vix_rsi_norm	vix_bb_pct	vix_close_vs_sma	alumunium_log_return	alumunium_rsi_norm	...	lithium_bb_pct	lithium_
count	984.0000	984.0000	984.0000	984.0000	984.0000	984.0000	984.0000	984.0000	984.0000	984.0000	...	984.0000	
mean	0.0009	0.5676	0.6079	0.0091	-0.0009	0.4801	0.4880	-0.0031	0.0004	0.5127	...	0.5104	
std	0.0131	0.1633	0.3118	0.0307	0.0704	0.1368	0.3008	0.1360	0.0146	0.1630	...	0.3217	
min	-0.0641	0.1370	-0.3453	-0.1117	-0.3665	0.0801	-0.1890	-0.2974	-0.0729	0.0152	...	-0.2569	
25%	-0.0059	0.4489	0.3820	-0.0083	-0.0390	0.3914	0.2608	-0.0891	-0.0085	0.4040	...	0.2506	
50%	0.0012	0.5762	0.6867	0.0115	-0.0059	0.4888	0.4305	-0.0207	-0.0001	0.5203	...	0.5038	
75%	0.0083	0.6906	0.8469	0.0290	0.0322	0.5676	0.6902	0.0600	0.0091	0.6258	...	0.7771	
max	0.0681	0.9778	1.2385	0.0975	0.3653	0.9108	1.4903	0.7274	0.0669	0.9632	...	1.3082	

8 rows × 40 columns

Ringkasan statistik deskriptif dari setiap fitur yang terdiri dari banyaknya:
data, mean, standar deviasi, nilai minimum, Quartil 1, Quartil 2 (Median), Quartil 3, dan nilai Maksimum

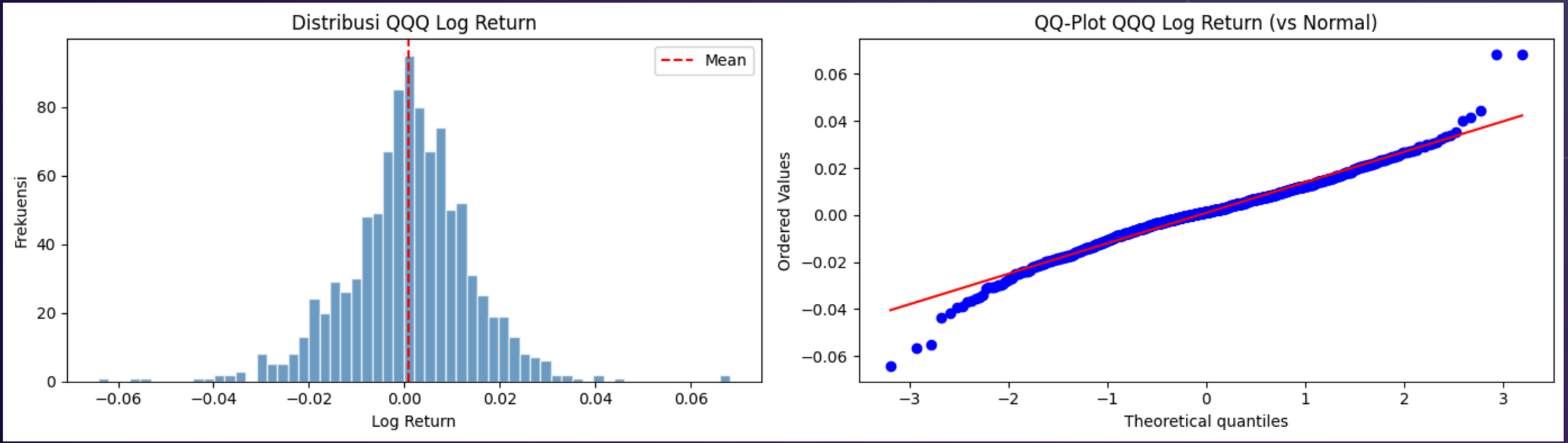
Preprocessing Data

```python
# ── Uji Stasionaritas dengan ADF Test ──────────────────────
# ADF (Augmented Dickey-Fuller): H0 = terdapat unit root (non-stasioner)
# Jika p-value < 0.05 → tolak H0 → seri stasioner ✓

from statsmodels.tsa.stattools import adfuller

print("=" * 55)
print("ADF STATIONARITY TEST  (p < 0.05 = stasioner ✓)")
print("=" * 55)
print(f"{'Kolom':<30} {'p-value':>10}  {'Status':>12}")
print("-" * 55)

# Uji semua log_return dan satu sampel fitur lainnya
test_cols = (log_ret_cols +
             [c for c in df.columns if 'rsi_norm' in c][:3] +
             [c for c in df.columns if 'bb_pct' in c][:3])

for col in test_cols:
    series = df[col].dropna()
    try:
        adf_stat, p_val, *_ = adfuller(series, autolag='AIC')
        status = "✓ stasioner" if p_val < 0.05 else "✗ non-stasioner"
        print(f"{col:<30} {p_val:>10.4f}  {status:>12}")
    except Exception as e:
        print(f"{col:<30}  Error: {e}")
```

```
=======================================================
ADF STATIONARITY TEST  (p < 0.05 = stasioner ✓)
=======================================================
Kolom                                p-value        Status
-------------------------------------------------------
qqq_log_return                        0.0000   ✓ stasioner
vix_log_return                        0.0000   ✓ stasioner
alumunium_log_return                  0.0000   ✓ stasioner
palladium_log_return                  0.0000   ✓ stasioner
copper_log_return                     0.0000   ✓ stasioner
silver_log_return                     0.0000   ✓ stasioner
platinum_log_return                   0.0000   ✓ stasioner
lithium_log_return                    0.0000   ✓ stasioner
uranium_log_return                    0.0000   ✓ stasioner
btc_log_return                        0.0000   ✓ stasioner
qqq_rsi_norm                          0.0000   ✓ stasioner
vix_rsi_norm                          0.0000   ✓ stasioner
alumunium_rsi_norm                    0.0000   ✓ stasioner
qqq_bb_pct                            0.0000   ✓ stasioner
vix_bb_pct                            0.0000   ✓ stasioner
alumunium_bb_pct                      0.0000   ✓ stasioner
```

Memastikan seluruh fitur runtun waktu (time series) sudah bersifat stasioner (tidak memiliki tren random walk) sebelum masuk ke tahap pemodelan. Langkah ini wajib dilakukan untuk menghindari regresi lancung (spurious regression).

Exploratory Data Analysis (EDA)



Membuat grafik Histogram, Density Plot, dan Q-Q Plot untuk memeriksa persebaran nilai log return harian dari indeks teknologi QQQ.

Memeriksa normalitas data. Hasil grafik menunjukkan bentuk yang simetris di bagian tengah, namun memiliki karakteristik fat-tails (ujung distribusi lebih tebal/ekstrem) yang mengonfirmasi adanya volatilitas tinggi khas data keuangan.

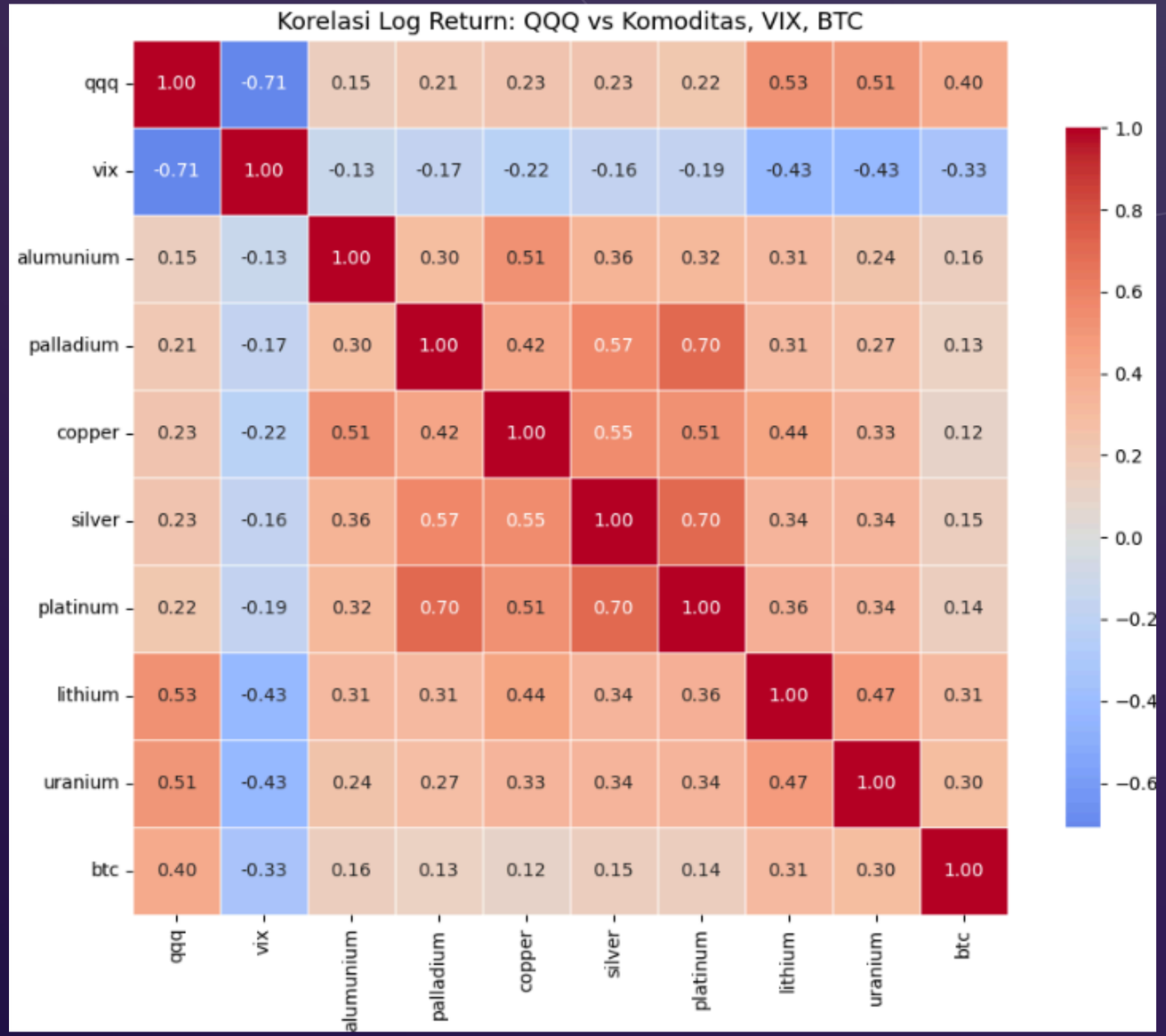
Exploratory Data Analysis (EDA)

Aset yang Dipilih (Informasi Independen & Baru)

- Copper (0.23): Membawa informasi independen meski korelasi rendah.
- Lithium (0.53) & Uranium (0.51): Memberikan sinyal dari sektor energi bersih.
- Bitcoin (0.40): Indikator aset risk-on (bergerak searah saham teknologi).

Aset yang Dieliminasi (Redundan & Sinyal Lemah)

- VIX (-0.71): Redundan (informasinya sudah tercermin di QQQ).
- Logam Lain (Al, Pd, Ag, Pt) (0.14 - 0.23): Dieliminasi karena sinyal terlalu lemah.



Feature Matrix Building & Train/Test Split

```
# ── Buat lag features tanpa fragmentasi DataFrame ──────────────
lag_dict = {}
for asset in assets:
    for feat in features:
        col = f"{asset}_{feat}"
        if col not in df.columns:
            print(f"  ⚠ Kolom tidak ditemukan: {col}")
            continue
        for lag in range(1, lookback + 1):
            lag_dict[f"{col}_lag{lag}"] = df[col].shift(lag)

df_lagged = pd.concat([df, pd.DataFrame(lag_dict, index=df.index)], axis=1)
```

Membuat fitur historis 1 hingga 3 hari ke belakang (Lag 1, 2, 3) untuk 5 aset utama (qqq, copper, lithium, uranium, btc). untuk memberikan konteks memori sekuensial kepada model MLP dasar agar mampu memahami pola pergerakan harga beberapa hari ke belakang.

```
# ── Target: QQQ return BESOK ──────────────────────────────────
df_lagged['target'] = df_lagged['qqq_log_return'].shift(-1)

# ── Drop NaN (dari shift) ─────────────────────────────────────
df_lagged = df_lagged.dropna()
```

Menggeser variabel target (qqq_log_return) ke atas sebanyak 1 hari (shift(-1)), lalu membuang baris kosong (NaN) sisa pergeseran data. untuk menyelaraskan sinyal indikator lintas aset hari ini (t) untuk memprediksi arah return indeks QQQ esok hari (t+1).

Feature Matrix Building & Train/Test Split

```
# ── Train / Test split (kronologis) ──────────────────────────
split_idx = int(len(df_lagged) * 0.8)
train     = df_lagged.iloc[:split_idx]
test      = df_lagged.iloc[split_idx:]

lag_cols             = [c for c in df_lagged.columns if '_lag' in c]
X_train, y_train     = train[lag_cols], train['target']
X_test,  y_test      = test[lag_cols],  test['target']

# ── StandardScaler fit hanya pada train ──────────────────────
scaler           = StandardScaler()
X_train_scaled   = scaler.fit_transform(X_train)
X_test_scaled    = scaler.transform(X_test)
```

Membagi data secara kronologis menjadi 80% Train dan 20% Test tanpa pengacakan (shuffle=False), lalu dinormalisasi dengan StandardScaler. untuk mencegah kebocoran data masa depan (data leakage) dan menyamakan skala input (Mean = 0, Std = 1) agar proses pembaruan bobot Neural Network berjalan stabil

```
=== Leakage Check ===
  Kolom lag (5 pertama): ['qqq_log_return_lag1', 'qqq_log_return_lag2', 'qqq_log_return_lag3', 'copper_log_return_lag1', 'copper_log_return_lag2']
  qqq_log_return tanpa lag ada di X? False ← harus False!
```

MLP

15 Fitur

Setiap neuron input menerima satu nilai fitur. Di sini 15 fitur = log return dari 5 aset × 3 lag hari. Tidak ada komputasi di sini, hanya "pintu masuk" data.

Dense (32, ReLU)

- Dense = fully connected: setiap neuron terhubung ke semua 15 input
- Terjadi operasi: z = W·x + b, lalu diaktivasi: output = max(0, z)
- ReLU membuang nilai negatif → membuat jaringan bisa belajar pola non-linear
- Layer ini "mendeteksi" pola dasar dari kombinasi fitur

Dropout (0,1)

- Bukan layer komputasi, tapi regularisasi
- Saat training, 10% neuron dimatikan secara acak tiap batch
- Efeknya: model tidak bergantung pada neuron tertentu → lebih general, tidak hafal data training (overfitting)
- Saat inference/prediksi, semua neuron aktif kembali

Dense (16, ReLU)

MLP

Dropout (0,1)

Dense (16, ReLU)

- Menerima representasi dari layer 1, lalu memperhalus/mengabstraksi lebih lanjut
- Neuron lebih sedikit (16 < 32) → model dipaksa mengompresi informasi ke yang paling penting
- Pola arsitektur funnel (mengecil) ini umum untuk regresi

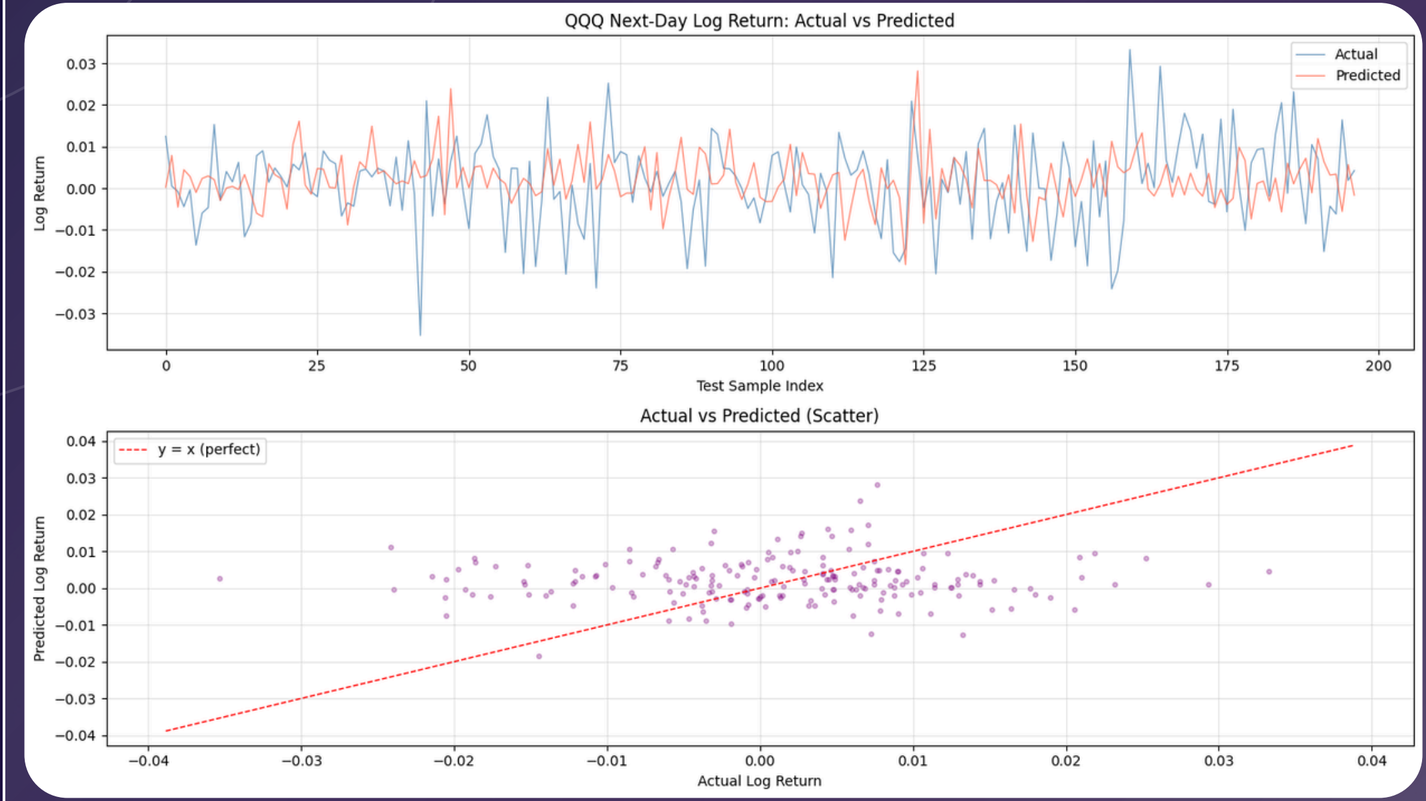
Dense (1)

- Menghasilkan satu angka kontinu = prediksi log return besok
- Tidak pakai aktivasi (linear) karena ini regresi, bukan klasifikasi
- Kalau pakai sigmoid/softmax, outputnya terbatas rentangnya — tidak cocok untuk nilai return yang bisa positif/negatif bebas

Loss : MSE

Error dikuadratkan agar kesalahan besar dihukum lebih berat. Adam mengoptimasi bobot W agar MSE ini minimal lewat backpropagation.

Hasil



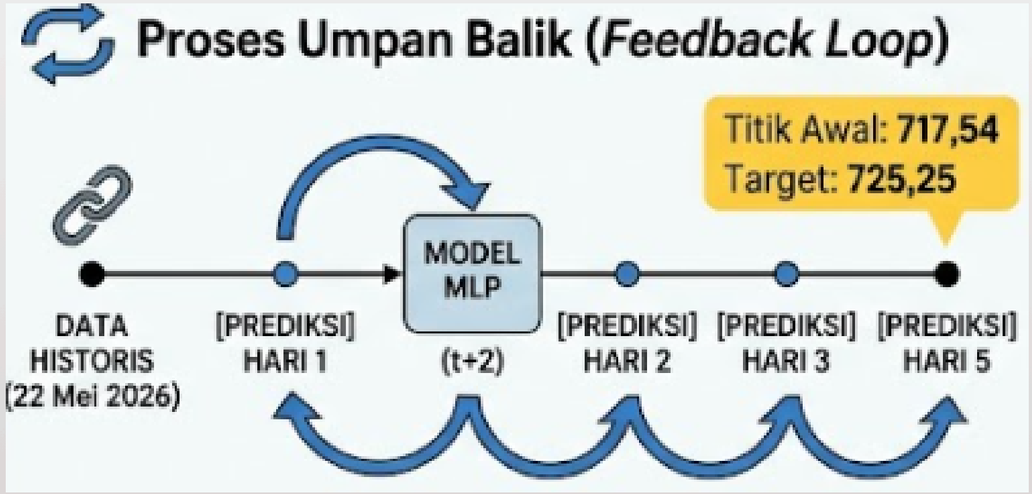
Diperoleh hasil prediksi pada gambar
di samping. dengan directional accuracy sekitar 56.85%.

```
==================================
EVALUASI MODEL
==================================
R²                    : -0.2935
MAE                   : 0.009613
RMSE                  : 0.012097
Directional Accuracy  : 56.85%
Binomial p-value      : 0.0318
```

Prediksi Harga 5 Hari kedepan

Mekanisme Iteratif

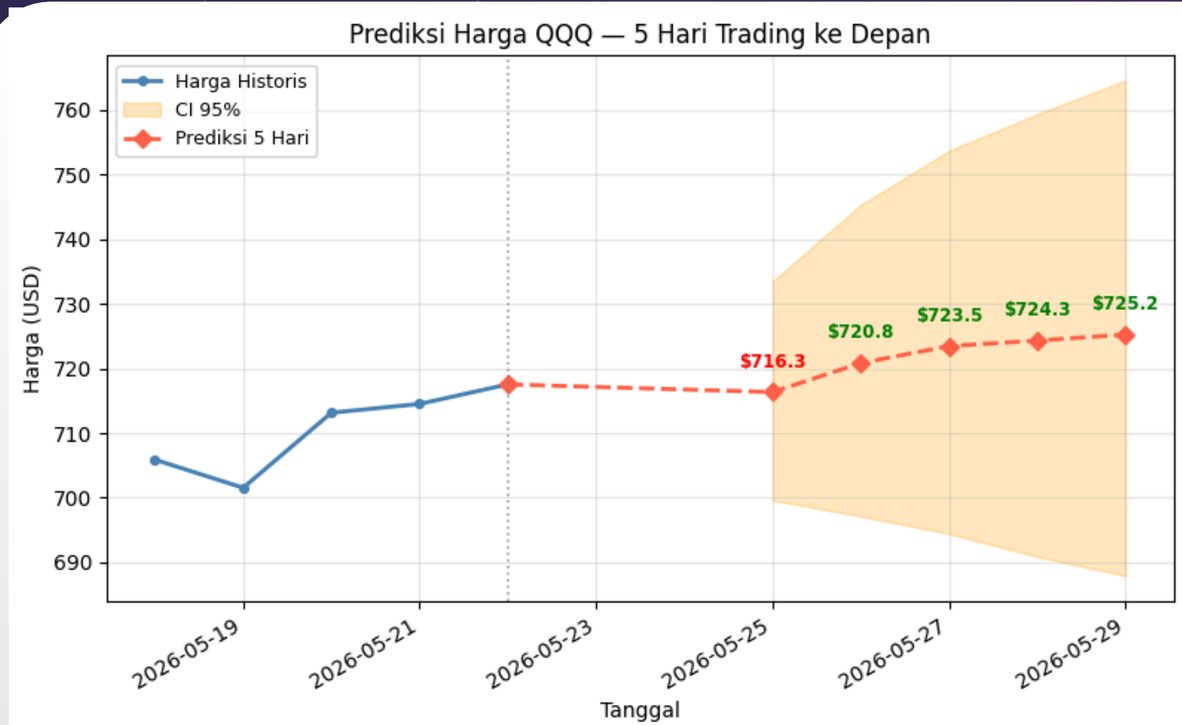
Karena model MLP dasar hanya mampu memprediksi satu hari ke depan (t+1), kami menggunakan metode berantai (**Recursive Forecasting**) untuk memproyeksikan prediksi 5 hari kedepan

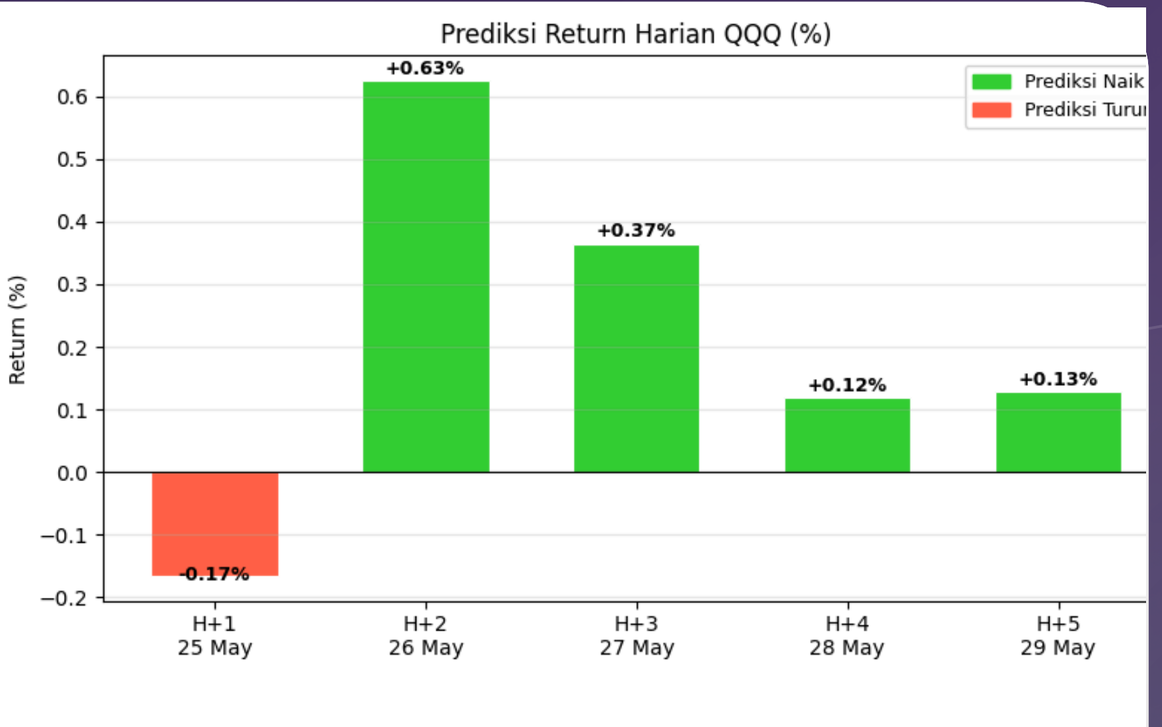


Risiko dan Akumulasi Error

- Compounding Errors: Setiap deviasi kecil pada prediksi Hari ke-1 akan terbawa dan melipatgandakan bias pada Hari ke-2, dan seterusnya.
- Pelebaran Confidence Interval: Secara visual, pita ketidakpastian (95% CI) melebar untuk mencerminkan risiko yang tumbuh eksponensial.
- Asumsi Input Statis: Model mengasumsikan returnaset lain (BTC, Komoditas) bernilai nol saat rekursi, yang menambah margin kesalahan riil

Prediksi Harga 5 Hari kedepan





Tangal	Pred Log Return	Return %	Harga Predksi	CI Lover (95%)	CI Upper (95%)	Arah
2026-05-25	-0.00166	-0.17%	$716.35	$699.62	$733.48	Turun
2026-05-26	+0.00624	+0.63%	$720.83	$697.14	$745.33	✅ Naik
2026-05-27	+0.00364	+0.37%	$723.46	$694.45	$733.69	✅ Naik
2026-05-28	+0.00118	+0.12%	$724.32	$690.88	$759.38	✅ Naik
2026-05-29	+0.00128	+0.13%	$725.25	$687.92	$764.61	✅ Naik

Kesimpulan

Model MLP dengan arsitektur sederhana (32→16 neuron) mampu memprediksi arah pergerakan harian QQQ menggunakan sinyal lintas aset. Fitur log return terbukti lebih efektif dibanding fitur mentah. Copper, Lithium, dan Uranium dipilih karena memberikan informasi independen yang tidak redundan terhadap QQQ. Model melampaui naive baseline pada metrik directional accuracy, menunjukkan potensi penggunaan cross-asset signal dalam prediksi indeks teknologi.

Kelompok 9

Terima Kasih