Machine Learning Project

Resilience of Middle Eastern vs European Markets

Erian STANLEY YOGARAJ - Ouiam BOUSSAID BENCHAARA

Class : A4 - IF1 ESILV

General Research Question :

> **Do Middle Eastern investment portfolios demonstrate greater financial resilience compared to European portfolios when facing global market fluctuations ?**

Introduction

Financial markets don't all react the same way when global uncertainly increases. Some markets stay relatively stable, while others become much more volatile. This observation motivated our project which is to understand whether the Middle Eastern market is more resilient than the European market when facing global shocks.

To study this, we focused on two majors indices :

- TASI (Saudi Arabia) - representing the Middle Eastern region
- EURO STOXX 50 - representing Europe

We have also included two important global factors :

- VIX, which reflects wordlwide market stress
- Brent Oil prices, which strongly influence Middle Eastern economies

In this project, *financial resilience* refers to how well a market absorbs shocks, limits losses, and stabilizes after turbulent periods. A resilient market should show more stable volatility, smaller drawdowns, and weaker reactions to global panic movements.

Our goal is then to compare the risk behavior of both regions and determine whether the Middle Eastern actually demonstrates stronger resilience than Europe.

1. Importing libraries and loading the data

In this action, we import the Python packages used throughout the notebook and load all four datasets :

- TASI (Middle Eastern market)
- EURO STOXX 50 (European market)

- VIX index (global volatility)
- Brent Crude Oil Prices

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as stats
import seaborn as sns
import os
import statsmodels.api as sm

from sklearn.model_selection import train_test_split, TimeSeriesSplit, GridSearc
from sklearn.linear_model import Ridge, LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor, BaggingRegressor, VotingRegr
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from arch import arch_model

import warnings
warnings.filterwarnings("ignore")

plt.style.use("seaborn-v0_8-muted")

base_path = r"C:\Users\syeri\OneDrive\Bureau\ESILV\A4\S7\Machine learning\Projec

path_tasi  = os.path.join(base_path, "Tadawul All Share Historical Data.csv")
path_eu    = os.path.join(base_path, "Euro Stoxx 50 Historical Data.csv")
path_vix   = os.path.join(base_path, "CBOE Volatility Index Historical Data.csv"
path_brent = os.path.join(base_path, "Brent Oil Futures Historical Data.csv")

tasi_df  = pd.read_csv(path_tasi)
eu_df    = pd.read_csv(path_eu)
vix_df   = pd.read_csv(path_vix)
brent_df = pd.read_csv(path_brent)

print(" ------ TASI raw sample: ------")
display(tasi_df.head())
print("\n ------ EURO STOXX 50 raw sample: ------")
display(eu_df.head())
print("\n ------ VIX raw sample: ------")
display(vix_df.head())
print("\n ------ Brent raw sample: ------")
display(brent_df.head())
```

```
------ TASI raw sample: ------
```

	Date	Price	Open	High	Low	Vol.	Change %
0	10/23/2025	11,611.68	11,599.57	11,614.02	11,549.23	226.17M	0.22%
1	10/22/2025	11,585.90	11,542.22	11,618.28	11,492.03	244.61M	0.35%
2	10/21/2025	11,545.80	11,664.92	11,665.77	11,539.43	232.78M	-0.85%
3	10/20/2025	11,644.55	11,702.55	11,715.56	11,625.04	226.56M	-0.39%
4	10/19/2025	11,690.61	11,697.45	11,740.24	11,679.96	190.83M	-0.05%

```
------ EURO STOXX 50 raw sample: ------
```

	Date	Price	Open	High	Low	Vol.	Change %
0	10/24/2025	5,677.55	5,698.70	5,699.15	5,675.30	NaN	0.19%
1	10/23/2025	5,667.05	5,648.30	5,674.05	5,639.85	NaN	0.49%
2	10/22/2025	5,639.21	5,672.24	5,685.30	5,635.90	NaN	-0.84%
3	10/21/2025	5,686.83	5,686.77	5,699.13	5,670.65	NaN	0.10%
4	10/20/2025	5,680.93	5,626.25	5,688.17	5,626.25	NaN	1.31%

```
------ VIX raw sample: ------
```

	Date	Price	Open	High	Low	Vol.	Change %
0	10/30/2025	16.57	16.26	17.17	15.73	NaN	-2.07%
1	10/29/2025	16.92	16.34	17.58	16.26	NaN	3.05%
2	10/28/2025	16.42	15.95	16.55	15.66	NaN	3.99%
3	10/27/2025	15.79	15.73	16.07	15.62	NaN	-3.54%
4	10/24/2025	16.37	17.02	17.22	16.02	NaN	-5.38%

```
------ Brent raw sample: ------
```

	Date	Price	Open	High	Low	Vol.	Change %
0	10/24/2025	65.20	65.12	65.95	64.73	475.49K	-0.14%
1	10/23/2025	65.29	63.80	65.66	63.37	808.93K	4.92%
2	10/22/2025	62.23	61.34	64.04	61.09	429.97K	1.97%
3	10/21/2025	61.03	60.78	61.82	60.20	323.90K	0.25%
4	10/20/2025	60.88	61.16	61.37	59.97	225.22K	-0.67%

2. Data preparation and cleaning

In this section, we clean and standardize all datasets so they can be compared properly across markets. We convert the date columns, remove formatting artefacts from prices, sort the data chronologically, and compute simple daily returns. These cleaned series will serve as the basis for all later analysis (volatility, correlations, and machine-learning models).

In [7]:
```python
def clean_price_series(df, name):
    """
    Simple cleaning step for one index/asset:
    - convert Date to datetime
    - sort by date
    - clean 'Price' column and convert to float
    - keep only Date + Price
    - compute daily returns
    """
    tmp = df.copy()
    # Date : datetime and sort
```

```
    tmp["Date"] = pd.to_datetime(tmp["Date"])
    tmp = tmp.sort_values("Date")
    # Clean the Price column
    tmp["Price"] = (tmp["Price"].astype(str).str.replace(",", "", regex=False).a
    # Keep only Date + Price, and give explicit names
    price_col = f"{name}_Price"
    tmp = tmp[["Date","Price"]].rename(columns={"Price": price_col})
    # Compute simple daily returns : r_t = (P_t / P_{t-1}) - 1
    return_col = f"{name}_Return"
    tmp[return_col] = tmp[price_col].pct_change()
    return tmp

tasi = clean_price_series(tasi_df, "TASI")
euro   = clean_price_series(eu_df, "EU")
vix    = clean_price_series(vix_df, "VIX")
brent  = clean_price_series(brent_df, "Brent")

print("Cleaned TASI:")
display(tasi.tail(3))
print("\nCleaned EURO STOXX 50:")
display(euro.tail(3))
print("\nCleaned VIX:")
display(vix.tail(3))
print("\nCleaned Brent:")
display(brent.tail(3))
```

```
Cleaned TASI:
```

	Date	TASI_Price	TASI_Return
2	2025-10-21	11545.80	-0.008480
1	2025-10-22	11585.90	0.003473
0	2025-10-23	11611.68	0.002225

```
Cleaned EURO STOXX 50:
```

	Date	EU_Price	EU_Return
2	2025-10-22	5639.21	-0.008374
1	2025-10-23	5667.05	0.004937
0	2025-10-24	5677.55	0.001853

```
Cleaned VIX:
```

	Date	VIX_Price	VIX_Return
2	2025-10-28	16.42	0.039899
1	2025-10-29	16.92	0.030451
0	2025-10-30	16.57	-0.020686

```
Cleaned Brent:
```

	Date	Brent_Price	Brent_Return
2	2025-10-22	62.23	0.019662
1	2025-10-23	65.29	0.049172
0	2025-10-24	65.20	-0.001378

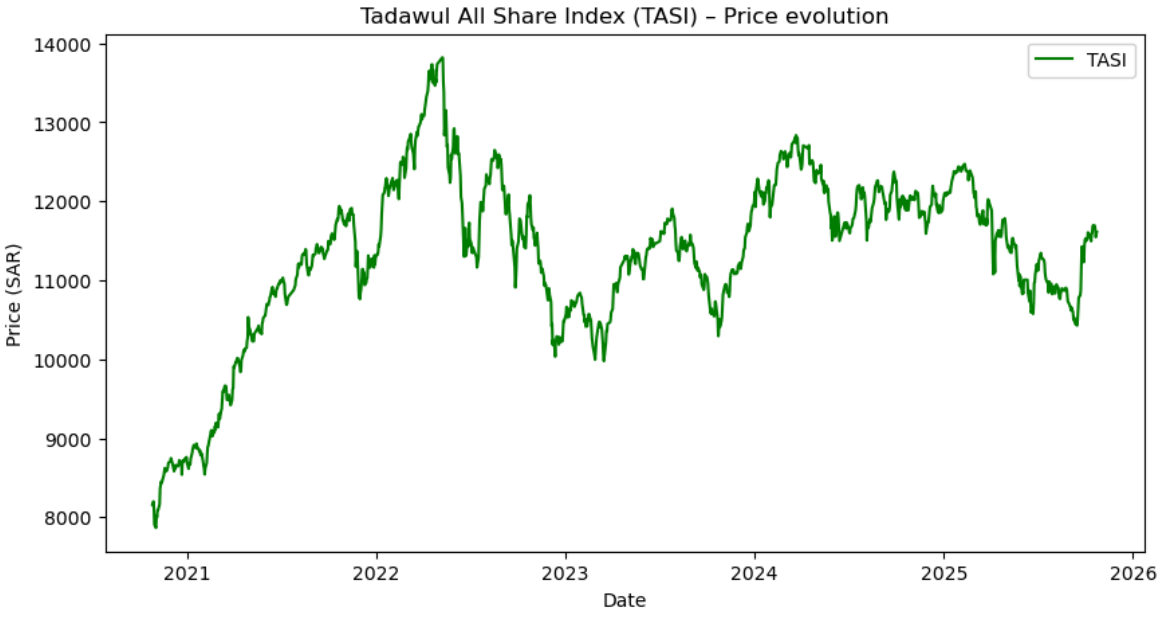
3. Exploratory Data Analysis (EDA)

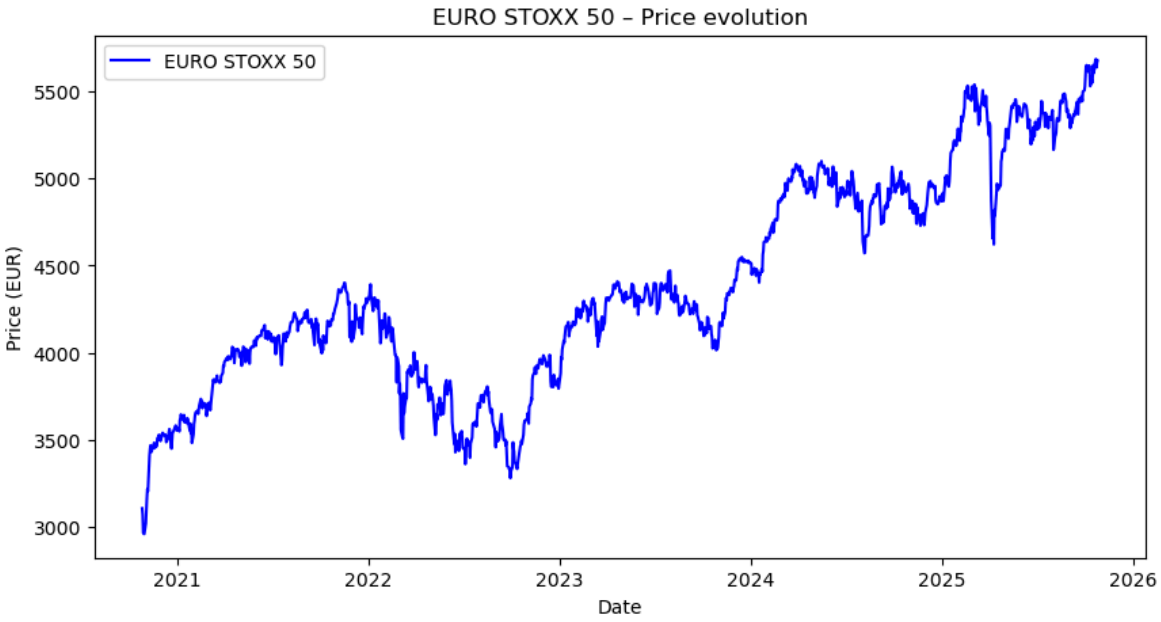
Before running any statistical or machine-learning models, we first explore the data visually. The goal is to understand how each market behaves over time, identify major of fluctuations, and compare the general level of volatility between the Middle Eastern index (TASI) and the European index (EURO STOXX 50). This step helps us build intuition about potential differences in financial resilience.

3.1. Plot price evolution

We start by plotting the price evolution of each index to observe long-term trends and periods of markets stress.

In [78]:
```
plt.figure(figsize=(10,5))
plt.plot(tasi["Date"], tasi["TASI_Price"], label="TASI", color="green")
plt.title("Tadawul All Share Index (TASI) – Price evolution")
plt.xlabel("Date")
plt.ylabel("Price (SAR)")
plt.legend()
plt.show()

plt.figure(figsize=(10,5))
plt.plot(euro["Date"], euro["EU_Price"], label="EURO STOXX 50", color="blue")
plt.title("EURO STOXX 50 – Price evolution")
plt.xlabel("Date")
plt.ylabel("Price (EUR)")
plt.legend()
plt.show()
```

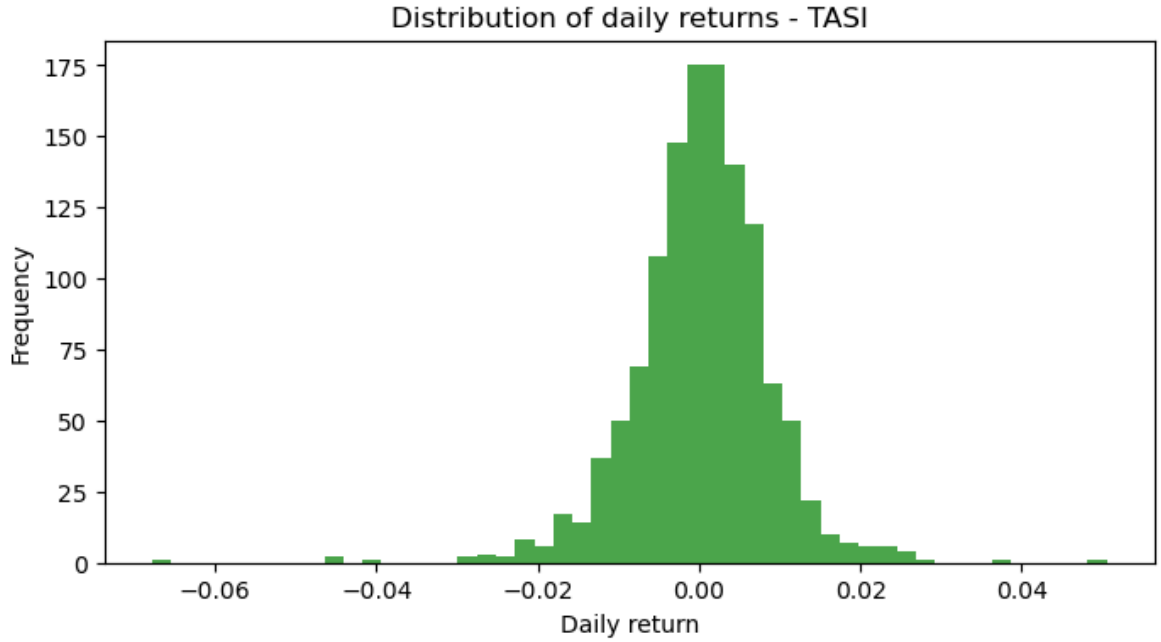


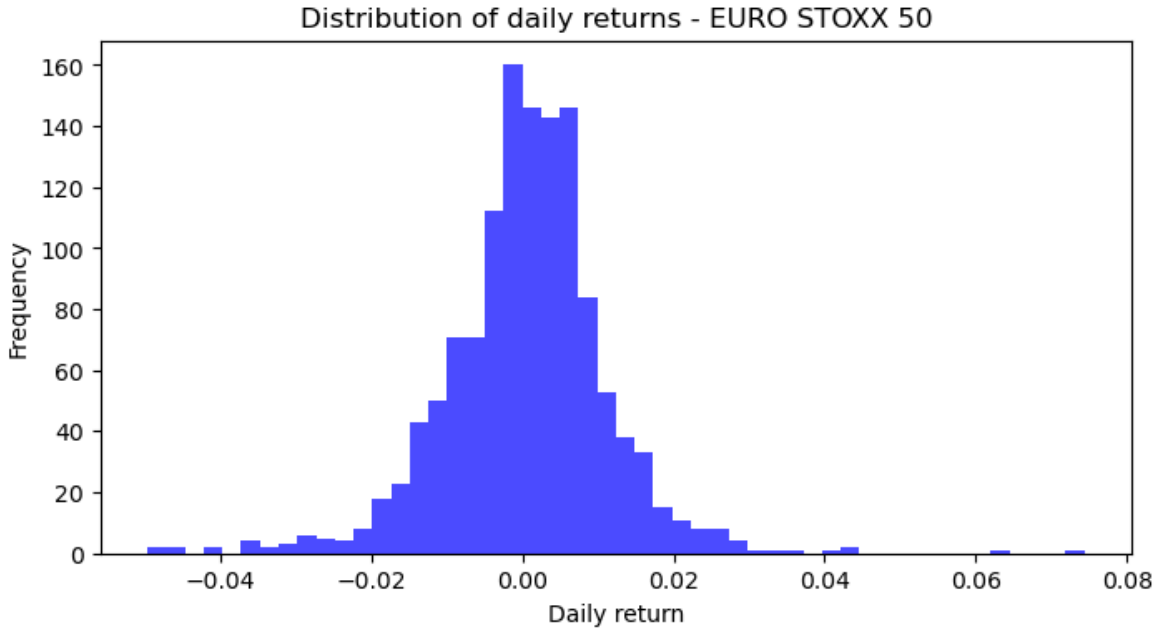


3.2. Distribution of returns

We examine the distribution of daily returns to compare the typical variability and the frequency of extreme movements in each market.

In [80]:
```
plt.figure(figsize=(8,4))
plt.hist(tasi["TASI_Return"], bins=50, alpha=0.7, color="green")
plt.title("Distribution of daily returns - TASI")
plt.xlabel("Daily return")
plt.ylabel("Frequency")
plt.show()

plt.figure(figsize=(8,4))
plt.hist(euro["EU_Return"], bins=50, alpha=0.7, color="blue")
plt.title("Distribution of daily returns - EURO STOXX 50")
plt.xlabel("Daily return")
plt.ylabel("Frequency")
plt.show()
```

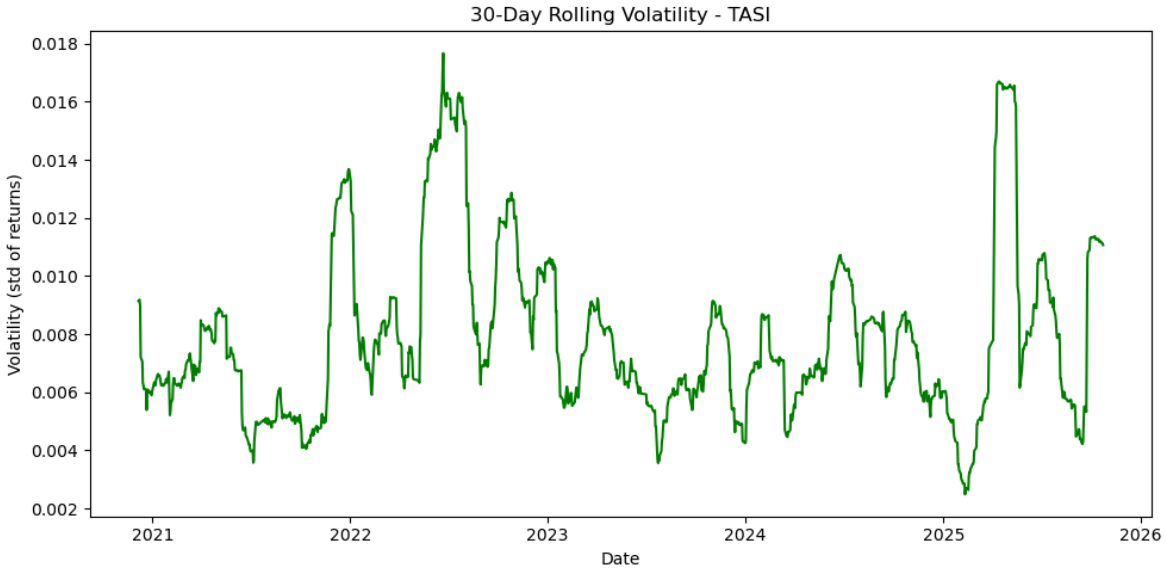


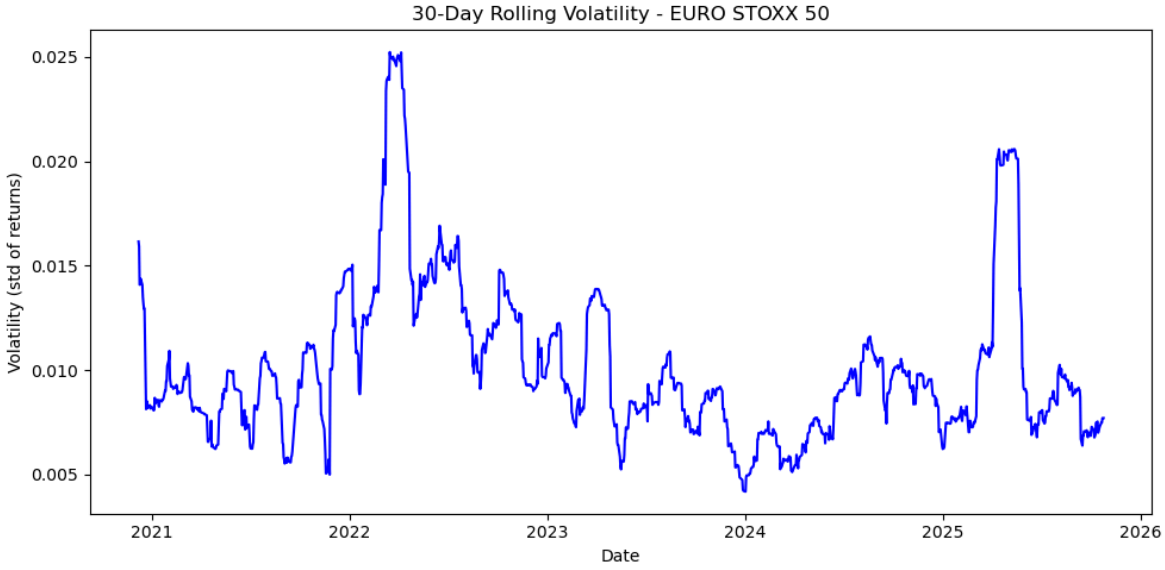


3.3. Rolling volatility (30-day)

To compare short-term stability, we compute a 30-day rolling volatility for each index. More stable markets trend to show smoother and lower volatility over time.

In [82]:
```
window = 30

tasi["TASI_Vol30"] = tasi["TASI_Return"].rolling(window=window).std()
euro["EU_Vol30"] = euro["EU_Return"].rolling(window=window).std()

plt.figure(figsize=(10,5))
plt.plot(tasi["Date"], tasi["TASI_Vol30"], color = "green", label = f"TASI {wind
plt.title(f"{window}-Day Rolling Volatility - TASI")
plt.xlabel("Date")
plt.ylabel("Volatility (std of returns)")
plt.tight_layout()
plt.show()

plt.figure(figsize=(10,5))
plt.plot(euro["Date"], euro["EU_Vol30"], color = "blue", label = f"EURO STOXX 50
plt.title(f"{window}-Day Rolling Volatility - EURO STOXX 50")
plt.xlabel("Date")
plt.ylabel("Volatility (std of returns)")
plt.tight_layout()
plt.show()
```





3.4. Max Drawdown

The max drawdown is one of the most widely used measures of financial risk. It represents the largest drop from a previous peak in the price of an asset. In other words, it answers a simple but essential question : "How much did the portfolio lose during its worst decline ?"

This metric is especially relevant for our project because we are comparing the financial resilience of Middle Eastern (TASI) and European (EURO STOXX 50) markets. A market with a smaller max drawdown is generally more resilient, as it suffers less during crises and recovers more quickly.

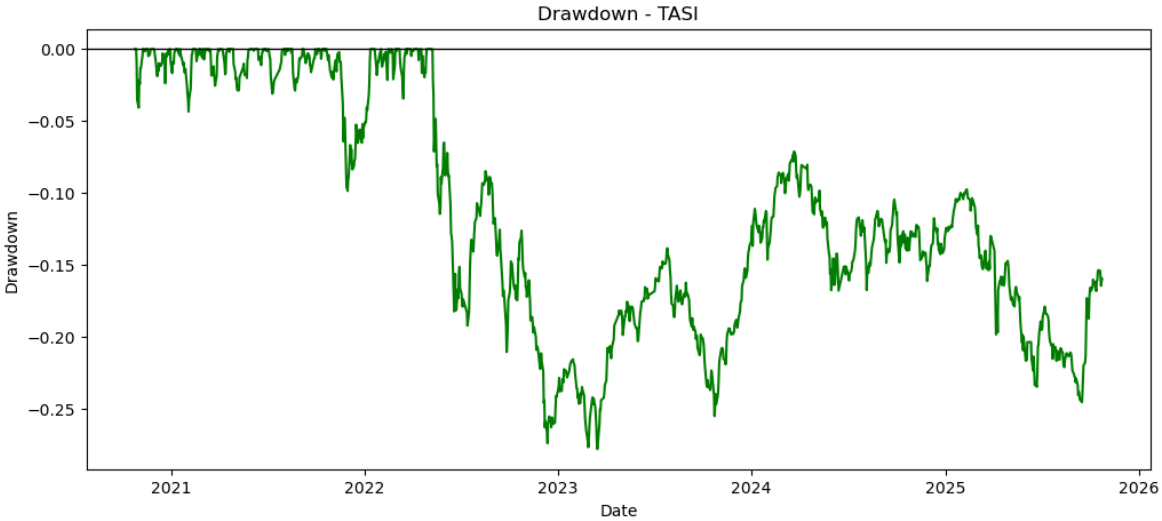
We will compute both :

- the drawdown series
- the max drawdown value for each index

This provides a direct, visual way to compare how each market behaved during downturns.

In [84]:
```
def compute_drawdown(price_series):
    # Computes drawdown and max drawdown from a price series.
    # Drawdown = (Price / Rolling Max) - 1
    roll_max = price_series.cummax()
    drawdown = (price_series / roll_max) - 1
    max_dd = drawdown.min()
    return drawdown, max_dd

tasi_drawdown, tasi_max_dd = compute_drawdown(tasi["TASI_Price"])
euro_drawdown, euro_max_dd = compute_drawdown(euro["EU_Price"])

print("Maximum Drawdown (TASI): {:.2%}".format(tasi_max_dd))
print("Maximum Drawdown (EURO STOXX 50): {:.2%}".format(euro_max_dd))

plt.figure(figsize=(12,5))
plt.plot(tasi["Date"], tasi_drawdown, label="TASI Drawdown", color="green")
plt.title("Drawdown - TASI")
plt.xlabel("Date")
plt.ylabel("Drawdown")
plt.axhline(0, color='black', linewidth=1)
plt.show()

plt.figure(figsize=(12,5))
plt.plot(euro["Date"], euro_drawdown, label="EURO STOXX 50 Drawdown", color="blu
plt.title("Drawdown - EURO STOXX 50")
plt.xlabel("Date")
plt.ylabel("Drawdown")
plt.axhline(0, color='black', linewidth=1)
plt.show()
```

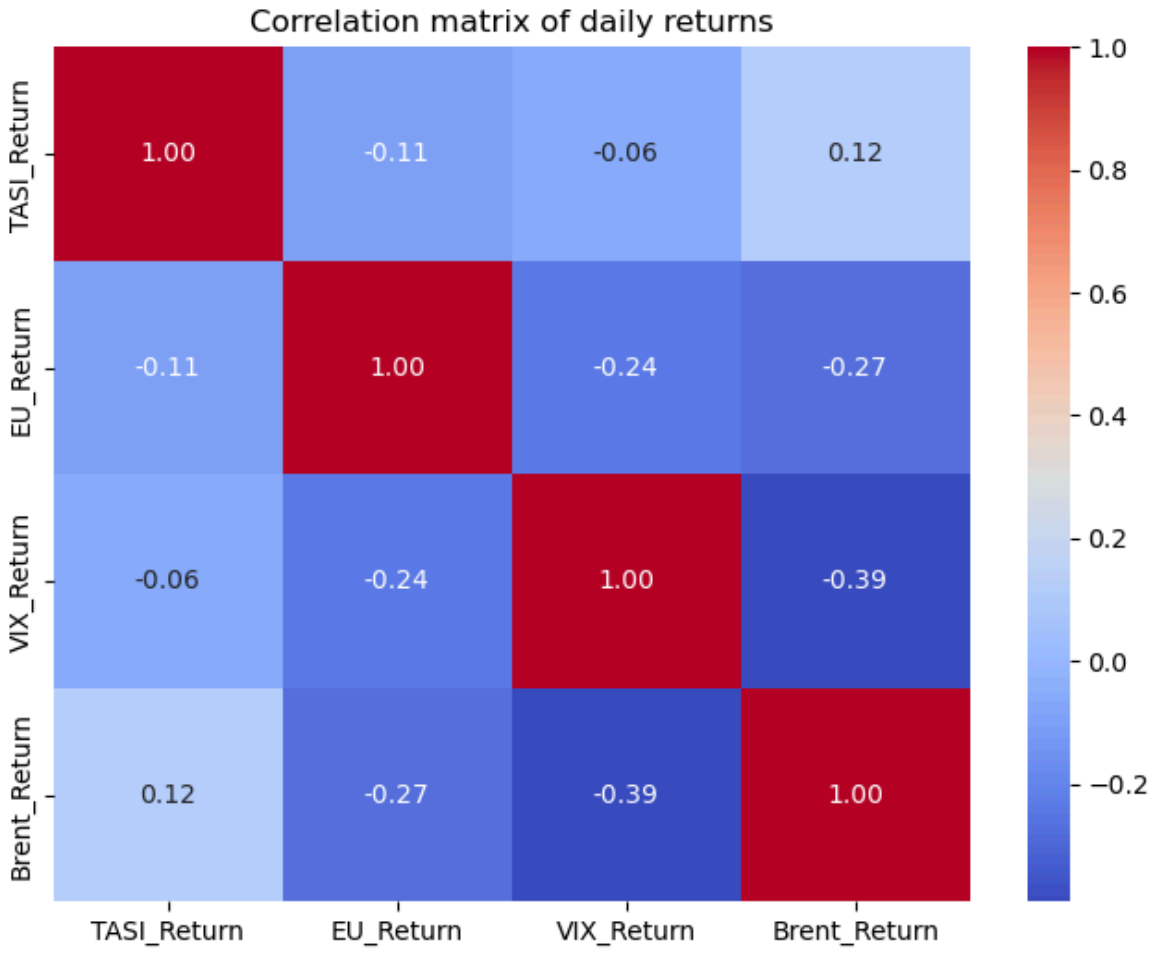
```
Maximum Drawdown (TASI): -27.81%
Maximum Drawdown (EURO STOXX 50): -25.50%
```



3.5. Correlation Heatmap

To get a quick idea of how the markets move together, we look at the correlation between all return series (TASI, EURO STOXX 50, VIX and Brent). This helps us whether the Middle Eastern market reacts differently to global factors compared to Europe.

In [17]:
```
corr_df = tasi.merge(euro, on="Date").merge(vix, on="Date").merge(brent, on="Dat

# Select only returns
ret_cols = ["TASI_Return", "EU_Return", "VIX_Return", "Brent_Return"]

plt.figure(figsize=(8,6))
sns.heatmap(corr_df[ret_cols].corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation matrix of daily returns")
plt.show()
```



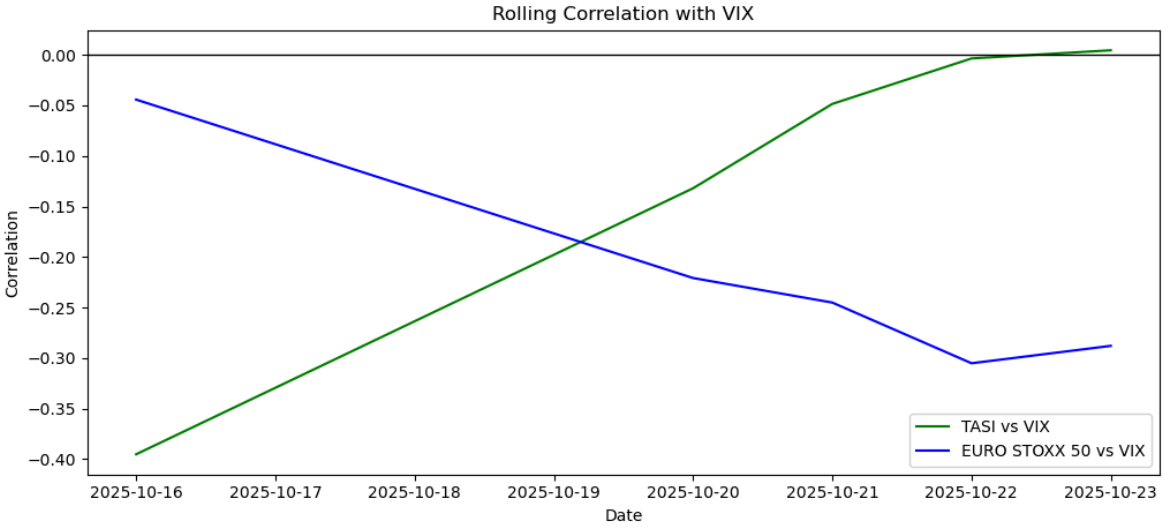
3.6. Rolling correlation with VIX and Brent

Since market relationships change over time, we compute rolling correlations with VIX and Brent. This shows how each index reacts during periods of stress and whether one market is more stable than the other.

In [86]:
```
merged = (
    tasi[["Date", "TASI_Return"]].merge(euro[["Date", "EU_Return"]], on="Date",
    .merge(vix[["Date", "VIX_Return"]], on="Date", how="inner")
    .merge(brent[["Date", "Brent_Return"]], on="Date", how="inner")
    .sort_values("Date").reset_index(drop=True))
print("Number of rows after merge :", len(merged))
print(merged.head())

merged = merged.dropna(subset=["TASI_Return", "EU_Return", "VIX_Return"])

window = min(10, len(merged))

merged["corr_TASI_VIX"] = merged["TASI_Return"].rolling(window).corr(merged["VIX
merged["corr_EU_VIX"]   = merged["EU_Return"].rolling(window).corr(merged["VIX_R
corr_plot = merged.dropna(subset=["corr_TASI_VIX", "corr_EU_VIX"])
print(corr_plot.head())

plt.figure(figsize=(12,5))
plt.plot(corr_plot["Date"], corr_plot["corr_TASI_VIX"], label="TASI vs VIX", col
plt.plot(corr_plot["Date"], corr_plot["corr_EU_VIX"],   label="EURO STOXX 50 vs
plt.axhline(0, color="black", linewidth=1)
plt.title("\nRolling Correlation with VIX")
```

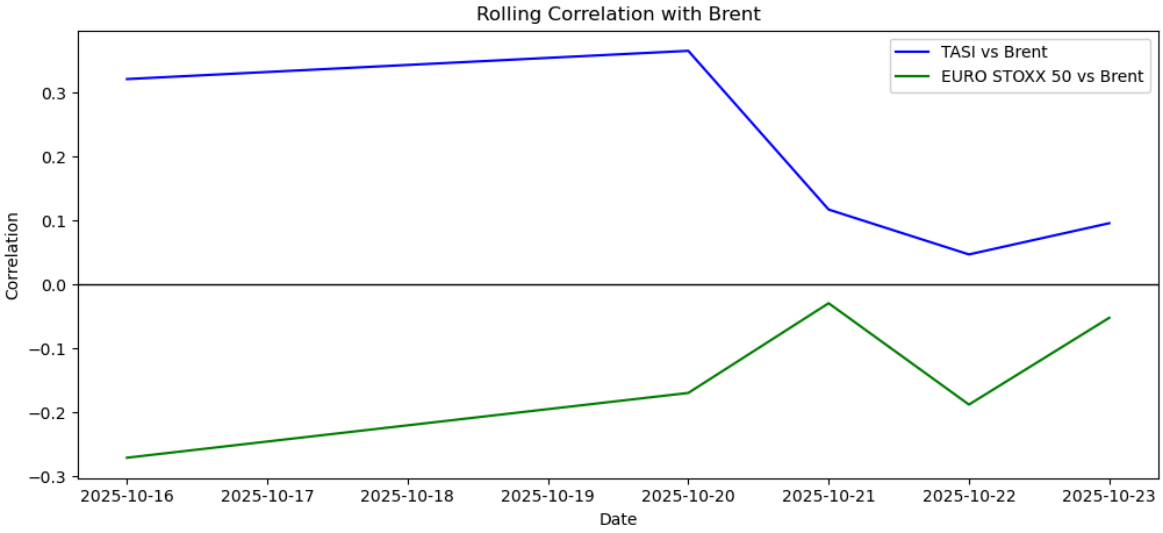
```
plt.xlabel("Date")
plt.ylabel("Correlation")
plt.legend()
plt.show()
```

```
Number of rows after merge : 15
        Date  TASI_Return  EU_Return  VIX_Return  Brent_Return
0 2025-09-30     0.006016   0.004197         NaN     -0.013977
1 2025-10-01     0.002294   0.009268    0.000614     -0.024918
2 2025-10-02    -0.002918   0.011575    0.020872     -0.018975
3 2025-10-06     0.006645  -0.004068   -0.016817      0.014567
4 2025-10-07    -0.001893  -0.002683    0.053146     -0.000305
         Date  TASI_Return  EU_Return  VIX_Return  Brent_Return  \
10 2025-10-16     0.001228   0.008382    0.226260     -0.013730
11 2025-10-20    -0.003940   0.013115   -0.122714     -0.006690
12 2025-10-21    -0.008480   0.001039   -0.019748      0.002464
13 2025-10-22     0.003473  -0.008374    0.040851      0.019662
14 2025-10-23     0.002225   0.004937   -0.069892      0.049172

    corr_TASI_VIX  corr_EU_VIX
10      -0.395196    -0.044313
11      -0.132022    -0.220809
12      -0.048583    -0.245086
13      -0.003582    -0.305142
14       0.004486    -0.287925
```



In [88]:
```
window = min(10, len(merged))

merged["corr_TASI_Brent"] = merged["TASI_Return"].rolling(window).corr(merged["B
merged["corr_EU_Brent"]   = merged["EU_Return"].rolling(window).corr(merged["Bre
corr_plot = merged.dropna(subset=["corr_TASI_Brent", "corr_EU_Brent"])
print(corr_plot.head())

plt.figure(figsize=(12,5))
plt.plot(corr_plot["Date"], corr_plot["corr_TASI_Brent"], label="TASI vs Brent",
plt.plot(corr_plot["Date"], corr_plot["corr_EU_Brent"],   label="EURO STOXX 50 v
plt.axhline(0, color="black", linewidth=1)
plt.title("\nRolling Correlation with Brent")
plt.xlabel("Date")
plt.ylabel("Correlation")
plt.legend()
plt.show()
```

```
         Date  TASI_Return  EU_Return  VIX_Return  Brent_Return  \
10 2025-10-16     0.001228   0.008382    0.226260     -0.013730
11 2025-10-20    -0.003940   0.013115   -0.122714     -0.006690
12 2025-10-21    -0.008480   0.001039   -0.019748      0.002464
13 2025-10-22     0.003473  -0.008374    0.040851      0.019662
14 2025-10-23     0.002225   0.004937   -0.069892      0.049172

    corr_TASI_VIX  corr_EU_VIX  corr_TASI_Brent  corr_EU_Brent
10      -0.395196    -0.044313         0.321353      -0.270706
11      -0.132022    -0.220809         0.365498      -0.169439
12      -0.048583    -0.245086         0.117322      -0.029144
13      -0.003582    -0.305142         0.047140      -0.187623
14       0.004486    -0.287925         0.095990      -0.051907
```



4. Risk and Resilience metrics

To compare how resilient each market is, we compute a set of classic financial risk indicators. These measures help quantify how each index reacts to volatility, losses, and extreme movements. Here, we evaluate :

- Sharpe Ratio : reward per unit of risk
- Value-at-Risk (VaR): worst expected loss under normal conditions
- Expected Shortfall (ES): average loss when things go really wrong.

Together, these indicators provide a simple but meaningful view of market resilience.

4.1. Sharpe Ratio

The Sharpe ratio measures the return earned per unit of volatility. A higher Sharpe ratio indicates better risk_adjusted performance, meaning the market compensates investors more efficiently for the risk taken. We assume a risk-free rate of 0%, which is standard for daily data unless otherwise specified.

In [92]:
```python
def sharpe_ratio(returns):
    return returns.mean() / returns.std()

sharpe_tasi = sharpe_ratio(tasi["TASI_Return"].dropna())
sharpe_eu = sharpe_ratio(euro["EU_Return"].dropna())
```

```
print("Sharpe Ratio - TASI:", round(sharpe_tasi, 4))
print("Sharpe Ratio - EURO:", round(sharpe_eu, 4))
```

```
Sharpe Ratio - TASI: 0.0377
Sharpe Ratio - EURO: 0.0488
```

4.2. Value-at-Risk (VaR)- 95%

VaR estimates the maximum expected loss under normal market conditions for a given confidence level. Here, we use 95% VaR meaning that on a typical day, losses should not exceed this value 95% of the time. A lower (less negative) VaR indicates higher resilience.

In [95]:
```
def var_95(returns):
    return np.percentile(returns.dropna(), 5)

var_tasi = var_95(tasi["TASI_Return"])
var_eu = var_95(euro["EU_Return"])

print("VaR 95% - TASI:", round(var_tasi, 4))
print("VaR 95% - EURO STOXX 50:", round(var_eu, 4))
```

```
VaR 95% - TASI: -0.0126
VaR 95% - EURO STOXX 50: -0.0167
```

4.3. Expected Shortfall (ES) - 95%

Expected Shortfall measures the average loss beyond the VaR threshold. It captures the severity of extreme downturns and is often considered more informative than VaR. A market with a less severes ES is typically more robust in turbulent periods.

In [98]:
```
def es_95(returns):
    var_level = np.percentile(returns.dropna(), 5)
    tail_losses = returns[returns < var_level]
    return tail_losses.mean()

es_tasi = es_95(tasi["TASI_Return"])
es_eu = es_95(euro["EU_Return"])

print("ES 95% - TASI:", round(es_tasi, 4))
print("ES 95% - EURO STOXX 50:", round(es_eu, 4))
```

```
ES 95% - TASI: -0.0198
ES 95% - EURO STOXX 50: -0.0251
```

5. Econometric Analysis

To better understand how each market reacts to global shocks, we run a simple econometric model where daily index returns are explained by movements in the VIX (global volatility) and Brent Oil. This helps measure how sensitive each region is to worldwide uncertainty. We estimate the following model separately for TASI and EURO STOXX 50:

Return_t = a + b1 * VIX_Return_t + b2 * Brent_Return_t + error_t

If a market has smaller and more stable coefficients, it means it reacts less to external shocks, which is a sign of financial resilience.

In [101…
```
econ = merged.dropna().reset_index(drop=True)

econ.head()
```

Out[101…

	Date	TASI_Return	EU_Return	VIX_Return	Brent_Return	corr_TASI_VIX	corr_EU_VIX
0	2025-10-16	0.001228	0.008382	0.226260	-0.013730	-0.395196	-0.044313
1	2025-10-20	-0.003940	0.013115	-0.122714	-0.006690	-0.132022	-0.220809
2	2025-10-21	-0.008480	0.001039	-0.019748	0.002464	-0.048583	-0.245086
3	2025-10-22	0.003473	-0.008374	0.040851	0.019662	-0.003582	-0.305142
4	2025-10-23	0.002225	0.004937	-0.069892	0.049172	0.004486	-0.287925

In [103…
```
X_tasi = econ[["VIX_Return", "Brent_Return"]]
y_tasi = econ["TASI_Return"]

X_tasi = sm.add_constant(X_tasi)
model_tasi = sm.OLS(y_tasi, X_tasi).fit()
print(model_tasi.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            TASI_Return   R-squared:                       0.549
Model:                            OLS   Adj. R-squared:                  0.098
Method:                 Least Squares   F-statistic:                     1.216
Date:                Sat, 06 Dec 2025   Prob (F-statistic):              0.451
Time:                        12:52:13   Log-Likelihood:                 21.946
No. Observations:                   5   AIC:                            -37.89
Df Residuals:                       2   BIC:                            -39.06
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const           -0.0028      0.002     -1.157      0.367      -0.013       0.008
VIX_Return       0.0242      0.019      1.251      0.337      -0.059       0.107
Brent_Return     0.1408      0.103      1.361      0.307      -0.304       0.586
==============================================================================
Omnibus:                        nan   Durbin-Watson:                   3.257
Prob(Omnibus):                  nan   Jarque-Bera (JB):                0.856
Skew:                        -0.992   Prob(JB):                        0.652
Kurtosis:                     2.583   Cond. No.                         48.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

In [105…

```
X_eu = econ[["VIX_Return", "Brent_Return"]]
y_eu = econ["EU_Return"]

X_eu = sm.add_constant(X_eu)
model_eu = sm.OLS(y_eu, X_eu).fit()
print(model_eu.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              EU_Return   R-squared:                       0.259
Model:                            OLS   Adj. R-squared:                 -0.483
Method:                 Least Squares   F-statistic:                    0.3490
Date:                Sat, 06 Dec 2025   Prob (F-statistic):              0.741
Time:                        12:52:14   Log-Likelihood:                 18.268
No. Observations:                   5   AIC:                            -30.54
Df Residuals:                       2   BIC:                            -31.71
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0058      0.005      1.154      0.368      -0.016       0.028
VIX_Return    -0.0218      0.040     -0.542      0.642      -0.195       0.151
Brent_Return  -0.1731      0.216     -0.801      0.507      -1.102       0.756
==============================================================================
Omnibus:                        nan   Durbin-Watson:                   1.797
Prob(Omnibus):                  nan   Jarque-Bera (JB):                0.660
Skew:                        -0.550   Prob(JB):                        0.719
Kurtosis:                     1.601   Cond. No.                         48.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Interpretation

We ran two OLS regressions to see how each market reacts to global risk factors :

- *TASI_Return ~ VIX_Return + Brent_Return*
- *EU_Return ~ VIX_Return + Brent_Return*

TASI : The VIX coefficient is small and positive, meaning TASI reacts only mildly to global volatility. The Brent coefficient is positive, which is expected since oil price increases generally support Middle Eastern markets. Overall, TASI appears less sensitive to global stress and shows a more stable relationship with external factors.

EURO STOXX 50: The VIX coefficient is negative, indicating European markets tend to fall when global volatility rises. The Brent coefficient is also negative, meaning oil shocks do not help the European market. Overall, this market shows a stronger reaction to global uncertainty.

Conclusion Even with limited statistical significance, the direction of the coefficients is informative: TASI shows a higher resilience, reacting less to global volatility and being supported by oil prices, while the European index is more vulnerable to global market stress.

6. Predictive Modeling (Machine Learning)

In this part, we use simple machine learning models to see how predictable daily returns are for both markets. The idea is that a more predictable market is usually more stable and therefore more resilient. We do not expect very high performance (this is normal in finance), but we can compare which index is easier to model: TASI (Middle East) or EURO STOXX 50 (Europe).

6.1. Feature construction

We build a small and simple feature set based only on past information:

- previous day's return
- 20-day rolling volatility of returns (based on past values only).

Then, we compare one dataset for TASI and one for EURO STOXX 50.

In [110…

```
tasi_ml = tasi.copy()
euro_ml = euro.copy()

# Previous day's return
tasi_ml["Prev_Return"] = tasi_ml["TASI_Return"].shift(1)
# 20-day rolling volatility based on past returns
tasi_ml["RollingVol20"] = tasi_ml["TASI_Return"].rolling(window=20).std().shift(
# Drop rows with missing values
tasi_ml = tasi_ml.dropna(subset=["TASI_Return", "Prev_Return", "RollingVol20"])
# Same features for EURO STOXX 50
euro_ml["Prev_Return"] = euro_ml["EU_Return"].shift(1)
euro_ml["RollingVol20"] = euro_ml["EU_Return"].rolling(window=20).std().shift(1)
euro_ml = euro_ml.dropna(subset=["EU_Return", "Prev_Return", "RollingVol20"])

print("TASI ML data shape:", tasi_ml.shape)
print("EURO STOXX 50 ML data shape:", euro_ml.shape)
```

```
TASI ML data shape: (1228, 6)
EURO STOXX 50 ML data shape: (1263, 6)
```

6.2. Models and evaluation

We use four standard regression models:

- Linear Regression (baseline)
- Ridge Regression (regularized linear model)
- Support Vector Regression (RBF kernel)
- Random Forest Regressor (non-linear, tree-based)

Because we work with time series, we use a time-series split (no shuffling) instead of a random train/test split. For each model, we compute:

- R^2 (goodness of fit)
- RMSE (root mean squared error)

- MAE (mean absolute error)

We run the same procedure separately for TASI and EURO STOXX 50.

In [113…

```
# Helper function to build feature matrix X and target y
def get_X_y(df, target_col, feature_cols):
    X = df[feature_cols].values
    y = df[target_col].values
    return X, y

#Time series split (here with 5 folds)
tscv = TimeSeriesSplit(n_splits=5)

#Define our models in a dictonary
models = {
    "LinearRegression": Pipeline([("model", LinearRegression())]),
    "Ridge": Pipeline([("model", Ridge(alpha=1.0))]),
    "SVR_RBF": Pipeline([("scaler", StandardScaler()), ("model", SVR(kernel="rbf
    "RandomForest": Pipeline([("model", RandomForestRegressor(n_estimators=100,
}

#Evaluate function with time-series cross-validation
def evaluate_models(df, target_col, feature_cols):
    X, y = get_X_y(df, target_col, feature_cols)
    results = []
    for name, model in models.items():
        r2_scores = []
        rmse_scores = []
        mae_scores = []
        for train_idx, test_idx in tscv.split(X):
            X_train, X_test = X[train_idx], X[test_idx]
            y_train, y_test = y[train_idx], y[test_idx]
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)
            r2_scores.append(r2_score(y_test, y_pred))
            rmse_scores.append(np.sqrt(mean_squared_error(y_test, y_pred)))
            mae_scores.append(mean_absolute_error(y_test, y_pred))
        results.append({
            "Model": name,
            "R2 mean": np.mean(r2_scores),
            "R2_std": np.std(r2_scores),
            "RMSE_mean": np.mean(rmse_scores),
            "MAE_mean": np.mean(mae_scores),
        })
    return (pd.DataFrame(results).sort_values(by="R2 mean", ascending=False).res
```

6.3. Results: TASI vs EURO STOXX 50

Now, we run all models on TASI and on EURO STOXX 50 using the same features:

- previous return
- 20-day rolling volatility

We compare the average R^2 across time-series folds to see which market is easier to predict.

In [116…

```
feature_cols = ["Prev_Return", "RollingVol20"]

results_tasi = evaluate_models(tasi_ml, target_col="TASI_Return", feature_cols=f
print("TASI - Model performance")
display(results_tasi)

results_eu = evaluate_models(euro_ml, target_col="EU_Return", feature_cols=featu
print("EURO STOXX 50 - Model performance")
display(results_eu)
```

```
TASI - Model performance
```

	Model	R2 mean	R2_std	RMSE_mean	MAE_mean
0	Ridge	-0.005265	0.005248	0.008686	0.006325
1	LinearRegression	-0.016334	0.062173	0.008769	0.006347
2	RandomForest	-0.028421	0.043154	0.008776	0.006425
3	SVR_RBF	-0.518615	0.628532	0.010631	0.007398

```
EURO STOXX 50 - Model performance
```

	Model	R2 mean	R2_std	RMSE_mean	MAE_mean
0	Ridge	-0.004635	0.003970	0.010729	0.007987
1	LinearRegression	-0.029978	0.054110	0.010907	0.008160
2	RandomForest	-0.068482	0.038730	0.011086	0.008191
3	SVR_RBF	-0.178928	0.196568	0.011545	0.008420

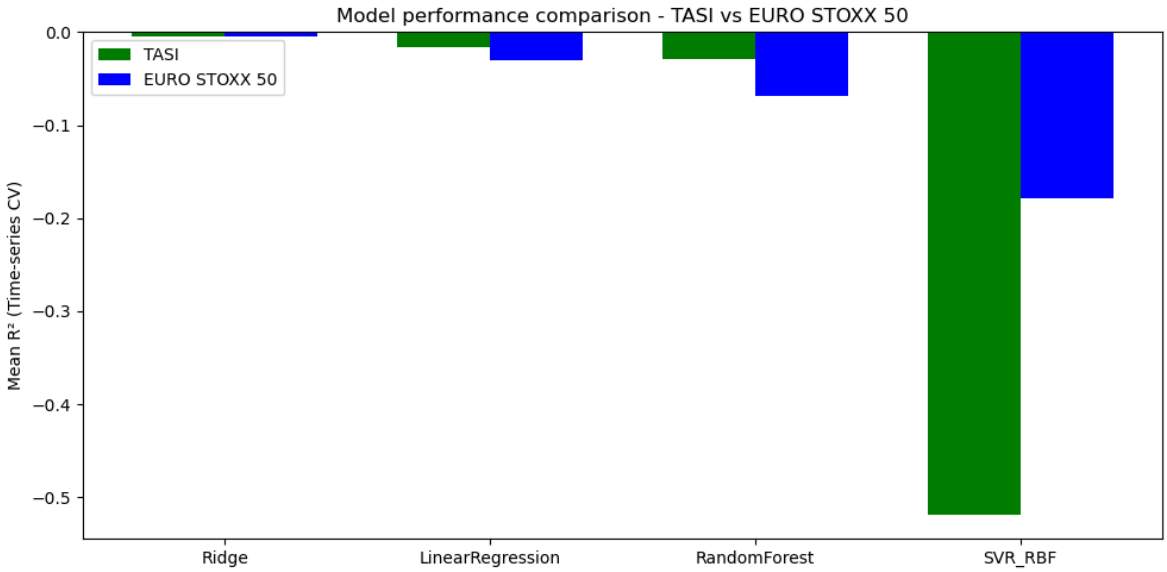
6.4. Visual comparison of model performance

To summarize the results, we compare the mean R^2 for each model between TASI and EURO STOXX 50. A higher R^2 means the model explains more of the return variation, which suggests a more predictable, and potentially more resilient, market.

In [121…

```
# Merge R² results into a single DataFrame for plotting
r2_tasi = results_tasi[["Model", "R2 mean"]].rename(columns={"R2 mean":"R2_TASI"
r2_eu = results_eu[["Model", "R2 mean"]].rename(columns={"R2 mean":"R2_EU"})
r2_compare = r2_tasi.merge(r2_eu, on="Model")
display(r2_compare)
x = np.arange(len(r2_compare["Model"]))
width = 0.35

plt.figure(figsize=(10,5))
plt.bar(x - width/2, r2_compare["R2_TASI"], width, label="TASI", color="green")
plt.bar(x + width/2, r2_compare["R2_EU"], width, label="EURO STOXX 50", color="b
plt.xticks(x, r2_compare["Model"])
plt.ylabel("Mean R² (Time-series CV)")
plt.title("Model performance comparison - TASI vs EURO STOXX 50")
plt.legend()
plt.tight_layout()
plt.show()
```

	Model	R2_TASI	R2_EU
0	Ridge	-0.005265	-0.004635
1	LinearRegression	-0.016334	-0.029978
2	RandomForest	-0.028421	-0.068482
3	SVR_RBF	-0.518615	-0.178928



Interpretation

The bar chart compares how well different models predict daily returns for the TASI (Middle East) and EURO STOXX 50 (Europe). Even though all R^2 scores are negative, which is common with noisy financial return series, a pattern still appears :

- *TASI* consistently performs slightly better than the *EURO STOXX 50* across almost all models
- **Linear** models (Ridge, Linear Regression) perform similarly on both markets but still give TASI a small advantage.
- **Random Forest** improves performance a bit but still cannot reach positive predictive power
- **SVR** performs the worst, especially for TASI, which suggests that overly flexible nonlinear models may overfit in this context.

Even if returns remain difficult to predict, TASI appears more model-friendly and less noisy, which is consistent with the idea that the Middle Eastern market is more stable and resilient.

7. GARCH Volatility Modeling

So far, we looked at volatility using simple rolling windows. To go one step further, we use a GARCH model, which is specifically designed to capture volatility clustering-periods of calm followed by bursts of turbulence. **GARCH(1,1)** is widely used in finance because it shows how pesistent volatility is. A market with less persistent volatility reacts to shocks

more softly and returns to stability faster, which is exactly what we mean by *financial resilience*. In this section, we estimate GARCH(1,1) models for TASI and EURO STOXX 50 and compare:

- conditional volatility over time
- the persistence of volatility (α + β)
- how each market absorbs and releases shocks

This gives a structural view of resilience that complements everything we observed earlier.

In [125…
```
tasi_ret = tasi["TASI_Return"].dropna() * 100
eu_ret = euro["EU_Return"].dropna() * 100
tasi_ret.name = "TASI_Return"
eu_ret.name = "EU_Return"
```

In [127…
```
garch_tasi = arch_model(tasi_ret, vol="Garch", p=1, q=1, dist="normal")
res_tasi = garch_tasi.fit(update_freq=0, disp="off")
print(res_tasi.summary())
```

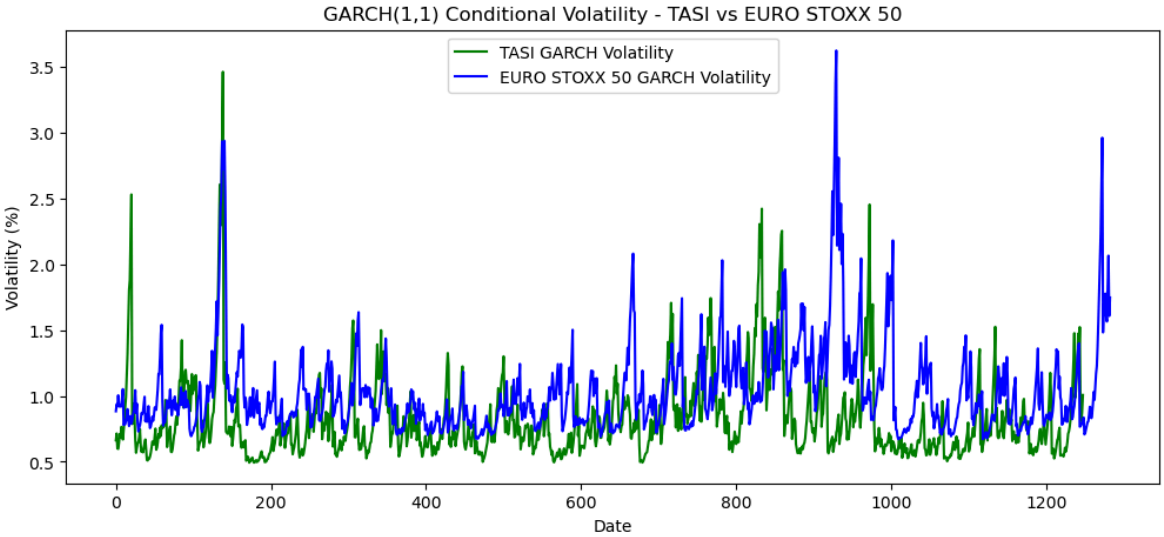
```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:            TASI_Return   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               -1463.15
Distribution:                  Normal   AIC:                           2934.29
Method:            Maximum Likelihood   BIC:                           2954.81
                                        No. Observations:                 1248
Date:                Sat, Dec 06 2025   Df Residuals:                     1247
Time:                        12:53:16   Df Model:                            1
                              Mean Model
============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
----------------------------------------------------------------------------
mu             0.0577  2.196e-02      2.627  8.621e-03 [1.464e-02,  0.101]
                           Volatility Model
============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.0649  1.675e-02      3.874  1.069e-04 [3.206e-02,9.772e-02]
alpha[1]       0.2363  5.687e-02      4.154  3.264e-05     [  0.125,  0.348]
beta[1]        0.6957  4.043e-02     17.209  2.274e-66     [  0.616,  0.775]
============================================================================

Covariance estimator: robust
```

In [129…
```
garch_eu = arch_model(eu_ret, vol="Garch", p=1, q=1, dist="normal")
res_eu = garch_eu.fit(update_freq=0, disp="off")
print(res_eu.summary())
```

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:              EU_Return   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               -1813.62
Distribution:                  Normal   AIC:                           3635.25
Method:            Maximum Likelihood   BIC:                           3655.88
                                        No. Observations:                 1283
Date:                Sat, Dec 06 2025   Df Residuals:                     1282
Time:                        12:53:16   Df Model:                            1
                                Mean Model
==============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
------------------------------------------------------------------------------
mu             0.0803  2.447e-02      3.283  1.027e-03 [3.238e-02,  0.128]
                            Volatility Model
==============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
------------------------------------------------------------------------------
omega          0.1134  3.098e-02      3.662  2.507e-04 [5.271e-02,  0.174]
alpha[1]       0.1788  3.886e-02      4.602  4.181e-06   [  0.103,  0.255]
beta[1]        0.7257  4.555e-02     15.933  3.765e-57   [  0.636,  0.815]
==============================================================================

Covariance estimator: robust
```

In [133…

```python
tasi_vol = res_tasi.conditional_volatility
eu_vol = res_eu.conditional_volatility

plt.figure(figsize=(12,5))
plt.plot(tasi_vol, label="TASI GARCH Volatility", color="green")
plt.plot(eu_vol, label="EURO STOXX 50 GARCH Volatility", color="blue")
plt.title("GARCH(1,1) Conditional Volatility - TASI vs EURO STOXX 50")
plt.ylabel("Volatility (%)")
plt.xlabel("Date")
plt.legend()
plt.show()
```



In [135…

```python
alpha_tasi = res_tasi.params["alpha[1]"]
beta_tasi = res_tasi.params["beta[1]"]

alpha_eu = res_eu.params["alpha[1]"]
beta_eu = res_eu.params["beta[1]"]
```

```
print("TASI volatility persistence (alpha + beta):", alpha_tasi + beta_tasi)
print("EURO STOXX 50 volatility persistence (alpha + beta):", alpha_eu + beta_eu
```

```
TASI volatility persistence (alpha + beta): 0.9319479651610812
EURO STOXX 50 volatility persistence (alpha + beta): 0.9045490578635746
```

Interpretation

The GARCH models allow us to examine how volatility evolves over time and how strongly each market reacts to shocks. Several insights clearly emerge:

- **Volatility spikes** are visibly higher and more frequent in the EURO STOXX 50 than in TASI, confirming that the European market reacts more strongly during turbulent periods.
- The **GARCH persistence** ($\alpha + \beta$), which measures how long volatility shocks last, is high for both markets but slightly higher for TASI (≈ 0.93) than for EURO STOXX 50 (≈ 0.90). This means both markets exhibit volatility clustering, but TASI tends to revert to stability slightly more slowly.
- However, in absolute terms, TASI's conditional volatility remains lower across most of the sample, even though persistence is high. The Middle Eastern market experiences smaller shocks overall, resulting in a smoother volatility profile.
- The European index shows larger jumps, higher peaks, and more pronounced turbulence, especially during global stress periods.

So, TASI exhibits lower and more stable volatility levels over time, reinforcing the idea that Middle Eastern markets are structurally more resilient to global volatility shocks than European markets.

Limitations of the Study

While the analyses conducted in this project provide consistent evidence regarding the relative resilience of Middle Eastern markets, several methodological limitations must be acknowledged:

1. Data horizon and frequency

Our study relies on daily historical prices. Using intraday data or a longer historical window could reveal dynamics that are not captured here.

2. Simplified machine-learning framework

The ML models use only lagged returns and rolling volatility as predictors. Adding macroeconomic variables (interest rates, liquidity measures, geopolitical risk indexes…) could substantially improve predictive accuracy.

3. GARCH model assumptions

The GARCH(1,1) specification assumes symmetric reactions to market shocks. In practice, market volatility often reacts asymmetrically (leverage effect), meaning that EGARCH or GJR-GARCH could provide a richer understanding.

4. Structural market factors not included

Market resilience is also influenced by elements such as:

- market depth and liquidity
- regulatory environment
- sector composition
- investor structure

These cannot be captured using price-based models alone.

5. Correlation does not imply causation

Econometric results capture co-movement but not causal mechanisms. More advanced frameworks (VAR, Granger causality, regime-switching models) would allow deeper investigation.

Conclusion

Across all quantitative methods used in this project, descriptive analysis, rolling correlations, risk metrics, econometric regressions, GARCH modelling, and machine-learning forecasts, our results consistently point in the same direction: the Middle Eastern market (TASI) displays higher financial resilience than the European market (EURO STOXX 50) when facing global shocks.

TASI shows lower and smoother volatility, weaker exposure to global risk factors such as the VIX, shallower drawdowns, and more predictable return patterns, which are all characteristics of a structurally more stable market. In contrast, the EURO STOXX 50 reacts more strongly to worldwide uncertainty and exhibits more pronounced volatility clustering.

While resilience does not mean immunity, the evidence suggests that Middle Eastern equity markets behave as a partial safe-harbor compared to developed European markets during turbulent periods.

In [ ]: