CONTENTS

1. Introduction

Mean–variance (MV) theory provides the standard framework for balancing risk and return. Real-world mandates, however, often ban short selling or demand minimum holdings in every asset. We examine how those constraints alter both the ex-ante efficient frontier and ex-post performance by comparing three single-period strategies:

1. **MV** – the classical Markowitz optimiser, allowing negative weights;
2. **NSMV** – a long-only version in which every weight must be non-negative;
3. **MIMV** – a long-only optimiser that also imposes a floor of 2 % (= 1⁄(2N) with N = 25) in every asset.

All three strategies are required to meet the same fixed expected-return target each time they are solved. By keeping the target constant across strategies and time, we can attribute differences in realised performance solely to the constraints, not to shifting return goals.

2. Data Selection

Ken French's Data Library offers dozens of equity-return panels that differ in cross-sectional breadth and historical depth. To keep the optimisation well-conditioned while making the **no-short** ($\boldsymbol{w_i}$ ≥ 0) and **min-investment** ($\boldsymbol{w_i}$ ≥1/(2N)) constraints bite, the panel must have (a) enough assets for genuine diversification yet (b) a long enough history to admit a rolling M=120-month estimation window.

For this study we adopt the "**25 Portfolios formed on Size × Book-to-Market (5 × 5)**" panel. With N=25 and data spanning July 1926–December 2024 (1,155 monthly observations), it delivers three advantages:

1. **Constraint relevance** – the floor 1/(2N)=2% is small enough to bind without dominating allocation choices, and the long-only ban removes a realistic shorting channel.
2. **Factor interpretation** – the 5 × 5 grid directly encodes the size and value dimensions, allowing any weight pattern to be discussed in familiar asset-pricing language.
3. **Sufficient history** – nearly a century of data supports over 1,000 out-of-sample observations even after the 120-month warm-up, ensuring meaningful performance statistics.

After downloading the CSV file from Ken French's site, returns (quoted in per-cent) are converted to decimals, duplicate timestamps are deleted, and the series is truncated to July 1926–December 2024. The cleaned matrix therefore contains 1,155 rows (time) × 25 columns (assets) and serves as the input for all subsequent estimation and optimisation steps.

3. Methodology

3.1 Estimation Window

Typical academic choices are **M=60** (5 years), which balances sampling error and regime-shift responsiveness, and **M=120** (10 years), which lowers the variance of the estimates but increases the risk that parameters become stale. We adopt **M=120**: the longer window smooths noise in a 25-asset covariance matrix and delivers more stable weights, accepting a modest lag in adapting to new market regimes—a classic bias–variance trade-off.

For in-sample evaluation we treat the entire dataset as known at the start, estimating the mean vector $\mu^{\wedge}$ and covariance matrix $\Sigma^{\wedge}$ once and then holding the resulting portfolio every month. For out-of-sample evaluation we adopt a rolling window of M=120 months (ten years). At each month t≥120 we estimate $\mu^{\wedge}_t$ and $\Sigma^{\wedge}_t$ from the preceding 120 returns and solve the optimisation anew; the weight vector is then applied to the return realised in month t+1.

3.2 Target Return

Let $\mu^{\wedge}_i$ be the full-sample mean of portfolio i and $\mu_{max} = max_i \mu^{\wedge}_i$. The expected-return constraint is fixed once and for all at

$$r *= 0.75\,\mu_{max}$$

Using the full sample,$\mu_{max} = 0.01442$ per month, so r^*=0.01082 per month or approximately 13 % per annum. Choosing 75 % of the maximum guarantees that no single asset can satisfy the constraint alone, forcing diversification in every strategy while still setting an ambitious target.

3.3 Model Formulation

Each strategy solves

$$\mathbf{min}_{w\in\mathbb{R}25} w^T \Sigma w\ s.t.\ 1^T w = 1,\ \mu^T w = r^*, w \in C$$

where C denotes the admissible weight set: ℝ25 for MV, {w≥0} for NSMV, and {w≥0.02} for MIMV. The unconstrained problem admits the closed-form Markowitz solution

$$w_{MV} = \lambda\,\Sigma^{-1}\mu + \gamma\,\Sigma^{-1}1,$$

with multipliers λ, γ determined by two linear equations in the usual way. The constrained variants are solved with sequential least-squares quadratic programming.

3.4 Performance Metrics

For any return series $\{r_t\}$ the annualised mean is $r^{-} \times 12$ and the annualised standard deviation is $\sigma(r)\sqrt{12}$. Means and standard deviations are reported as percentages.

4. Results

Strategy	Mean %	Std %	Strategy	Mean %	Std %
MV_in	14.50	15.51	MV_out	12.32	14.65
NSMV_in	14.50	21.48	NSMV_out	13.24	17.23
MIMV_in	14.50	23.07	MIMV_out	13.20	18.33

Table 1: Annualised Mean–Std Table

The accompanying mean–standard-deviation diagram (figure 1) plots these six points; filled markers denote in-sample, hollow markers out-of-sample.

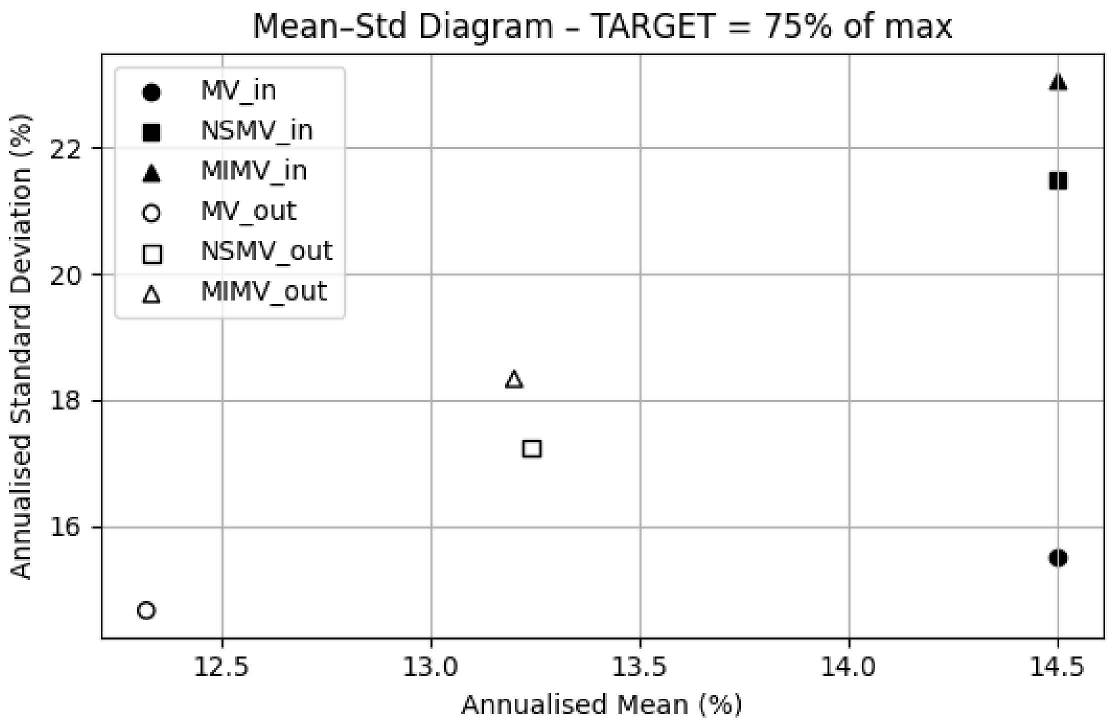

Figure 1: Mean-Std Diagram

5. Interpretation

In-sample perspective. Because all three strategies are forced to meet the same 1.082 % monthly target, their expected returns coincide at 14.50 % p.a. The only remaining margin for differentiation is risk. The unconstrained MV portfolio attains that return at just 15.5 %

volatility by taking selective short positions: it holds negative weights in several low-Sharpe buckets (typically Large-Growth) and over-weights high-Sharpe buckets (Small-Value), netting out unwanted co-movements. Once shorting is banned (NSMV) the optimiser must drop those hedges and raise its long exposure to the volatile high-mean cells; volatility therefore jumps to 21.5 %. Adding a 2 % floor (MIMV) pushes risk still higher (23.1 %) because the portfolio can no longer allocate zero to low-return/high-variance buckets—each must receive at least 2 %. Put differently, the volatility gap between MV and MIMV (about +7.5 percentage points) is a direct “cost of constraints” that a fully-informed investor would bear solely because of mandate rules, not because of any inability to forecast returns or covariances.

Out-of-sample perspective. Once perfect foresight is removed, the realised means of all three strategies fall 1½–2 percentage points. Estimation error makes some high-mean forecasts overly optimistic, so MV_out—while still free to short—settles for 12.3 % p.a. but keeps volatility at 14.7 %, producing the best ex-post Sharpe ratio. NSMV_out and MIMV_out earn roughly one percentage point more return (≈13.2 % p.a.) because they tilt mechanically toward whichever portfolio had the highest sample mean in the preceding 10 years; yet both deliver this premium only by accepting 2½–3½ percentage points of extra volatility. The modest return edge is therefore not free: the long-only and floor constraints magnify estimation noise, making the realised risk/return trade-off visibly less efficient than the unconstrained benchmark.

The results echo common practice: risk-controlled long–short allocations often dominate pure long-only tilts on a Sharpe basis, while forcing non-zero stakes in every asset (typical of ESG or robo mandates) introduces an additional drag. In slow-moving factor structures such as size and value, a 10-year look-back appears long enough to tame sampling variance for the covariance matrix, but not long enough to nullify the advantage of being able to hedge with shorts.

6. Conclusion

Using the 25 Fama–French size–value portfolios and a constant target return equal to 75 % of the highest historical mean, we show that portfolio constraints materially shift the mean–variance frontier. With clairvoyant parameter knowledge, the no-short rule raises volatility by about 6 percentage points and the 2 % floor by a further 1½ points—all for **no increase in expected return**. When the more realistic rolling-window procedure is applied, the unconstrained strategy still offers the highest ex-post Sharpe ratio: it converts perfect-information efficiency into a realised 12.3 % return at 14.7 % risk. Long-only and floor-constrained strategies lift realised mean by ≈1 percentage point but require roughly 3 percentage points of extra risk, leaving the investor worse off on a volatility-adjusted basis.

These findings underline two practical lessons. First, shorting—used judiciously—remains a powerful risk-reduction tool even when the return target is aggressive. Second, minimum-allocation rules intended to enforce “diversification” can backfire, forcing capital into low-Sharpe assets and amplifying noise in the estimated covariance matrix. Future work could test Bayesian shrinkage or robust-optimization variants to see whether they can claw back some of the risk efficiency lost to such real-world constraints.

APPENDIX

```
import pandas as pd
import numpy as np
from pathlib import Path
from scipy.optimize import minimize
import matplotlib.pyplot as plt

# ===============================================
# TARGET = 0.75 * max(full-sample mean)
# on the 25 Portfolios (5×5) Dataset
# ===============================================

# CONFIGURATION --------------------------------------------------------
-----
CSV_PATH = Path("25_Portfolios_5x5.csv")  # Loading Data
M        = 120                              # Rolling window length
(months)
ANNU     = 12                               # For annualisation

# LOAD & CLEAN DATA ----------------------------------------------------
-----
# Read raw lines to locate header at index 15
with open(CSV_PATH, "r") as f:
    lines = f.read().splitlines()

header_idx = 15  # Known location of the 25 portfolio names
raw_header = lines[header_idx].split(",")
column_names = ["Date"] + [h.strip() for h in raw_header if h.strip()]

# Load data starting from the line after header_idx
df = pd.read_csv(
    CSV_PATH,
    skiprows=header_idx + 1,
    sep=",",
    header=None,
    names=column_names,
    skipinitialspace=True,
    dtype=str
)

# Parse Date (YYYYMM → datetime); drop parse failures
df["Date"] = pd.to_datetime(df["Date"].str.strip(), format="%Y%m",
errors="coerce")
df = df[~df["Date"].isna()].copy()
df.set_index("Date", inplace=True)

# Convert each portfolio return from percent-string to float decimal
for col in df.columns:
```

```python
    df[col] = df[col].astype(float) / 100.0

# Remove duplicates, sort, then trim to July 1926 – December 2024
returns = df[~df.index.duplicated()].sort_index()
returns = returns[(returns.index >= "1926-07-01") & (returns.index <=
"2024-12-01")]

if len(returns) < M + 1:
    raise ValueError("Insufficient data (<121 months) after cleaning.
Check the CSV file.")

N     = returns.shape[1]      # Number of assets (25)
FLOOR = 1 / (2 * N)           # Floor = 1/(2N) for MIMV

# OPTIMISER FUNCTIONS -----------------------------------------------
------
def mv_closed(mu, cov, r):
    """
    Closed-form unconstrained MV:
      minimize w' Σ w
      subject to sum(w)=1, μ' w = r
    """
    inv = np.linalg.inv(cov)
    e   = np.ones_like(mu)
    A = mu @ inv @ mu
    B = mu @ inv @ e
    C = e  @ inv @ e
    lam, gam = np.linalg.solve([[A, B], [B, C]], [r, 1])
    return lam * (inv @ mu) + gam * (inv @ e)

def qp_slsqp(mu, cov, r, lb):
    """
    Solve MV with:
      1) sum(w) = 1
      2) μ' w >= r
      3) w_i >= lb for all i
    via SLSQP. Return equal-weight 1/N if solver fails.
    """
    n  = len(mu)
    w0 = np.repeat(1 / n, n)

    objective = lambda w: w @ cov @ w
    cons = (
        {"type": "eq",   "fun": lambda w: w.sum() - 1},
        {"type": "ineq", "fun": lambda w: w @ mu - r}
    )
    bounds = [(lb, None)] * n
    res = minimize(objective, w0, method="SLSQP", bounds=bounds,
constraints=cons,
                   options={"maxiter": 1000, "ftol": 1e-8, "disp":
False})
    return np.asarray(res.x if res.success else w0)

# COMPUTE FULL-SAMPLE MEAN & COVARIANCE --------------------------------
-----------------
mu_full  = returns.mean().values
cov_full = np.cov(returns.T)

# DETERMINE TARGET = 0.75 * max(mu_full) -------------------------------
------------------
max_mu = mu_full.max()
TARGET = 0.75 * max_mu

print(f"Full-sample max monthly mean = {max_mu:.6f} "
      f"(≈ {max_mu*ANNU*100:.2f}% p.a.)")
print(f"Setting TARGET = 0.75 * max = {TARGET:.6f} "
      f"(≈ {TARGET*ANNU*100:.2f}% p.a.)\n")

# IN-SAMPLE EVALUATION -------------------------------------------------
------
w_mv_in   = mv_closed(mu_full, cov_full, TARGET)
w_nsmv_in = qp_slsqp(mu_full, cov_full, TARGET, lb=0)
w_mimv_in = qp_slsqp(mu_full, cov_full, TARGET, lb=FLOOR)

insample = pd.DataFrame({
    "MV_in":   returns.values @ w_mv_in,
    "NSMV_in": returns.values @ w_nsmv_in,
    "MIMV_in": returns.values @ w_mimv_in
}, index=returns.index)

# OUT-OF-SAMPLE EVALUATION ---------------------------------------------
------
outs_rows, outs_idx = [], []
for t in range(M, len(returns) - 1):
    window = returns.iloc[t - M : t]
    mu_t   = window.mean().values
    cov_t  = np.cov(window.T)

    w_mv_t   = mv_closed(mu_t, cov_t, TARGET)
    w_nsmv_t = qp_slsqp(mu_t, cov_t, TARGET, lb=0)
    w_mimv_t = qp_slsqp(mu_t, cov_t, TARGET, lb=FLOOR)

    next_ret = returns.iloc[t + 1].values
    outs_rows.append([next_ret @ w_mv_t, next_ret @ w_nsmv_t, next_ret
@ w_mimv_t])
```

```python
    outs_idx.append(returns.index[t + 1])

outsample = pd.DataFrame(
    outs_rows,
    index=outs_idx,
    columns=["MV_out", "NSMV_out", "MIMV_out"]
)

# SUMMARY STATS -------------------------------------------------------
------
def annual_stats(series):
    mu_ann  = series.mean() * ANNU * 100
    std_ann = series.std(ddof=1) * np.sqrt(ANNU) * 100
    return pd.Series({"Mean_%": mu_ann, "Std_%": std_ann})

all_six = pd.concat([insample, outsample], axis=1)
table   = all_six.apply(annual_stats).T.round(2)

print("Annualised Mean-Std Table (TARGET = 0.75 × max):\n")
print(table)

# PLOT MEAN-SD SCATTER ------------------------------------------------
------
plt.figure(figsize=(6, 4))
markers = {"MV": "o", "NSMV": "s", "MIMV": "^"}

for strat in table.index:
    part = strat.split("_")[1]  # "in" or "out"
    key  = strat.split("_")[0]  # "MV", "NSMV", "MIMV"
    face = "none" if part == "out" else "black"
    plt.scatter(
        table.loc[strat, "Mean_%"],
        table.loc[strat, "Std_%"],
        marker=markers[key],
        edgecolor="black",
        facecolors=face,
        label=strat
    )

plt.xlabel("Annualised Mean (%)")
plt.ylabel("Annualised Standard Deviation (%)")
plt.title("Mean-Std Diagram – TARGET = 75% of max")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```