MS6621

Market Risk

Navigating Market Risk:Analysis of the NIFTY Financial Services Index

The NIFTY Financial Services Index is a sectoral index that includes stocks from the financial services sector. This index represents the performance of Indian financial services companies, including banks, financial institutions, housing finance companies, insurance companies, and other financial services providers. The selection of securities in the NIFTY Financial Services Index is based on their market capitalization and liquidity.

Nifty Financial Services Index is computed using free float market capitalization method, wherein the level of the index reflects the total free float market value of all the stocks in the index relative to particular base market capitalization value. Nifty Financial Services

Index can be used for a variety of purposes such as benchmarking fund portfolios, launching of index funds, ETFs and structured products

The official NSE India website provides detailed information about the NIFTY Financial Services Index and its constituent securities.

2. Market Data Collection

Securities Included:

HDFC Bank Ltd.

ICICI Bank Ltd.

Kotak Mahindra Bank Ltd.

State Bank of India

Axis Bank Ltd.

Bajaj Finserv Ltd.

Housing Development Finance Corporation Ltd.

HDFC Life Insurance Company Ltd.

ICICI Prudential Life Insurance Company Ltd.

SBI Life Insurance Company Ltd.

ICICI Lombard General Insurance Company Ltd.

General Insurance Corporation of India

SBI Cards and Payment Services Ltd.

Bajaj Finance Ltd.

Mahindra & Mahindra Financial Services Ltd.

Shriram Transport Finance Company Ltd.

Cholamandalam Investment and Finance Company Ltd.

SBI Funds Management Pvt. Ltd.

Aditya Birla Capital Ltd.

Indiabulls Housing Finance Ltd.

ind_niftyfinancelist.csv

Data Collection: From November 20, 2023, to May 18, 2024, market data was gathered for the companies that make up the NIFTY Financial Services Index. There are enough observations from this time period to support a thorough statistical analysis.

Files of each security:

SHRIRAMFIN.csv

ICICIGI.csv,

IDFC.csv,

PFC.csv

CHOLAFIN.csv

KOTAKBANK.csv

RECLTD.csv

HDFCAMC.csv

ICICIPRULI.csv

LICHSGFIN.csv

AXISBANK.csv

BAJFINANCE.csv

SBIN.csv

BAJAJFINSV.csv

SBILIFE.csv

ICICIBANK.csv

HDFCLIFE.csv

SBICARD.csv

MUTHOOTFIN.csv

HDFCBANK.csv

Calculation of Daily Returns: The percentage change in each security's closing price was used to compute the daily returns.

Returns.csv

3. Returns and Covariance Matrix Calculation

The percentage change in the closing prices was utilised to calculate the daily returns. The average of the daily returns was then used to get the mean returns for each security. The daily returns were also used to generate the covariance matrix, which shows how closely the returns of several securities move together.

The average return and the covariance matrix—two crucial components for portfolio optimization—were then computed using these returns.

```python
import numpy as np
import pandas as pd
import scipy.optimize as sco
from scipy.optimize import minimize

# Load data
data = pd.read_csv('returns.csv')

# Calculate daily returns
returns = data

# Calculate mean returns and covariance matrix
mean_returns = returns.mean()
cov_matrix = returns.cov()
def portfolio_total_returns(weights, mean_returns, cov_matrix):
    returns = np.sum(weights*mean_returns)*122
    std_deviation = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights))) * np.sqrt(122)
    return returns, std_deviation

def minimize_volatility(mean_returns, cov_matrix):
    num_assets = len(mean_returns)
    args = (mean_returns, cov_matrix)
    constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
    bounds = tuple((0.01,0.33) for asset in range(num_assets)) #the capping limit for FMCG index is 33%
    result = minimize(lambda weights: portfolio_total_returns(weights, mean_returns, cov_matrix)[1],
                      num_assets*[1./num_assets],
                      method = 'SLSQP',
                      bounds = bounds,
                      tol = 1e-6,
                      constraints = constraints)
    return result
```

```python
mean_returns
```

```
AXISBANK       0.001296
BAJFINANCE    -0.000266
BAJAJFINSV     0.000046
CHOLAFIN       0.001261
HDFCAMC        0.002362
HDFCBANK      -0.000118
HDFCLIFE      -0.001016
ICICIBANK      0.001751
ICICIGI        0.001314
ICICIPRULI     0.000714
IDFC          -0.000100
KOTAKBANK     -0.000215
LICHSGFIN      0.002842
MUTHOOTFIN     0.002228
PFC            0.003401
RECLTD         0.004165
SBICARD       -0.000234
SBILIFE        0.000407
SHRIRAMFIN     0.001620
SBIN           0.003203
dtype: float64
```

```
In [4]: cov_matrix
```

Out[4]:

	AXISBANK	BAJFINANCE	BAJAJFINSV	CHOLAFIN	HDFCAMC	HDFCBANK	HDFCLIFE	ICICIBANK	ICICIGI	ICICIPRULI	IDFC	KOTAKBA
AXISBANK	0.000206	6.884059e-05	0.000050	0.000009	0.000051	0.000061	0.000049	0.000094	-1.164506e-05	0.000028	0.000070	1.782798
BAJFINANCE	0.000069	2.518042e-04	0.000142	0.000108	0.000027	0.000046	0.000063	0.000051	8.353603e-07	0.000030	0.000039	4.856765
BAJAJFINSV	0.000050	1.417458e-04	0.000158	0.000094	0.000027	0.000044	0.000053	0.000044	8.944398e-06	0.000045	0.000033	1.398368
CHOLAFIN	0.000009	1.080998e-04	0.000094	0.000414	0.000075	0.000061	0.000038	0.000015	5.171649e-06	0.000056	0.000118	1.989910
HDFCAMC	0.000051	2.675710e-05	0.000027	0.000075	0.000226	0.000043	0.000068	0.000017	4.034239e-05	0.000038	0.000073	4.391360
HDFCBANK	0.000061	4.561732e-05	0.000044	0.000061	0.000043	0.000193	0.000022	0.000058	-2.334238e-05	0.000038	0.000086	8.597330
HDFCLIFE	0.000049	6.262919e-05	0.000053	0.000038	0.000068	0.000022	0.000201	0.000004	7.730536e-05	0.000126	0.000058	3.976275
ICICIBANK	0.000094	5.064572e-05	0.000044	0.000015	0.000017	0.000058	0.000004	0.000145	-1.005245e-05	-0.000015	0.000024	7.103867
ICICIGI	-0.000012	8.353603e-07	0.000009	0.000005	0.000040	-0.000023	0.000077	-0.000010	2.295205e-04	0.000079	0.000001	-4.9726
ICICIPRULI	0.000028	2.992974e-05	0.000045	0.000056	0.000038	0.000038	0.000126	-0.000015	7.850458e-05	0.000280	0.000060	2.882536
IDFC	0.000070	3.859651e-05	0.000033	0.000118	0.000073	0.000086	0.000058	0.000024	1.030512e-06	0.000060	0.000245	5.308126
KOTAKBANK	0.000018	4.856765e-05	0.000014	0.000020	0.000004	0.000086	0.000004	0.000071	-4.972604e-07	0.000029	0.000053	2.613242
LICHSGFIN	0.000064	4.788455e-05	0.000057	0.000086	0.000090	0.000077	0.000061	0.000036	5.936839e-05	0.000101	0.000126	5.717144

4. Portfolio Variance

The portfolio variance is a measure of the risk associated with the portfolio. It is calculated using the weights of the individual securities and the covariance matrix.The objective was to find the portfolio weights that minimize the variance (risk) of the portfolio. The optimization was carried out using the Sequential Least Squares Programming (SLSQP) method, subject to the constraints that the weights must sum to 1 (fully invested portfolio) and must be between 0 and 1 (no short selling).

```
min_var_port = minimize_volatility(mean_returns, cov_matrix)
# print(min_var_port)
min_var_weights = min_var_port['x']
print("Minimum Variance Portfolio Weights:", min_var_weights)

weights_df = pd.DataFrame(min_var_weights, index=returns.columns, columns=['Weight'])
print(weights_df)
```

```
Minimum Variance Portfolio Weights: [0.02458541 0.01       0.07599533 0.01       0.07335682 0.0298696
 0.0395934  0.16947309 0.13205968 0.02746635 0.01       0.10314602
 0.01       0.01       0.01       0.01       0.15617213 0.02080676
 0.01       0.06747542]
```

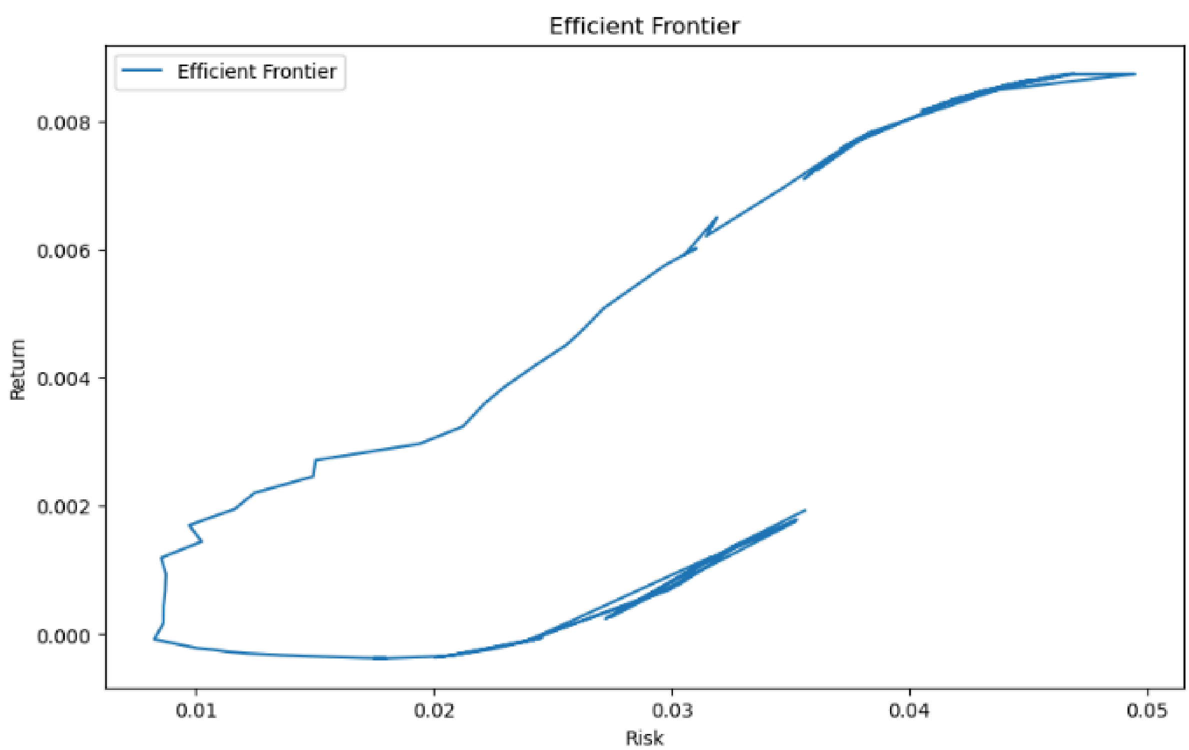
	Weight
AXISBANK	0.024585
BAJFINANCE	0.010000
BAJAJFINSV	0.075995
CHOLAFIN	0.010000
HDFCAMC	0.073357
HDFCBANK	0.029870
HDFCLIFE	0.039593
ICICIBANK	0.169473
ICICIGI	0.132060
ICICIPRULI	0.027466
IDFC	0.010000
KOTAKBANK	0.103146
LICHSGFIN	0.010000
MUTHOOTFIN	0.010000
PFC	0.010000
RECLTD	0.010000
SBICARD	0.156172
SBILIFE	0.020807
SHRIRAMFIN	0.010000
SBIN	0.067475

5.Efficient Frontier:

The efficient frontier was plotted to show the set of optimal portfolios that offer the highest expected return for a given level of risk. The minimum variance portfolio lies on this frontier, representing the portfolio with the lowest risk for a given level of expected return.

```python
def std_deviation(weights):
    return np.dot(weights.T, np.dot(cov_matrix, weights))
def generate_efficient_frontier(mean_returns, cov_matrix, num_portfolios = 100):
    portfolio_returns = []
    portfolio_risk = []
    max_returns = mean_returns.max()
    min_returns = mean_returns.min()
    target_returns = np.linspace(min_returns-0.01, max_returns+0.01, num_portfolios)
    for target in target_returns:
        constraints = [
            {'type': 'eq', 'fun': lambda x: np.sum(x) - 1},  # Sum of weights = 1
            {'type': 'eq', 'fun': lambda x: np.dot(x, mean_returns) - target}  # Portfolio expected return = target_return
        ]
        bounds = [(0.01, 0.33) for i in range(len(mean_returns))]
        guess_weights = np.random.rand(len(mean_returns))
        results = minimize(std_deviation, guess_weights, method='SLSQP', bounds=bounds, constraints=constraints)
        weights = results.x
        returns = np.dot(weights, mean_returns)
        risk = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))
        portfolio_returns.append(returns)
        portfolio_risk.append(risk)
    return np.array(portfolio_returns), np.array(portfolio_risk)

portfolio_returns, portfolio_risks = generate_efficient_frontier(mean_returns, cov_matrix)
plt.figure(figsize=(10, 6))
plt.plot(portfolio_risks, portfolio_returns, label='Efficient Frontier')
plt.xlabel('Risk')
plt.ylabel('Return')
plt.title('Efficient Frontier')
plt.legend()
plt.show()
```



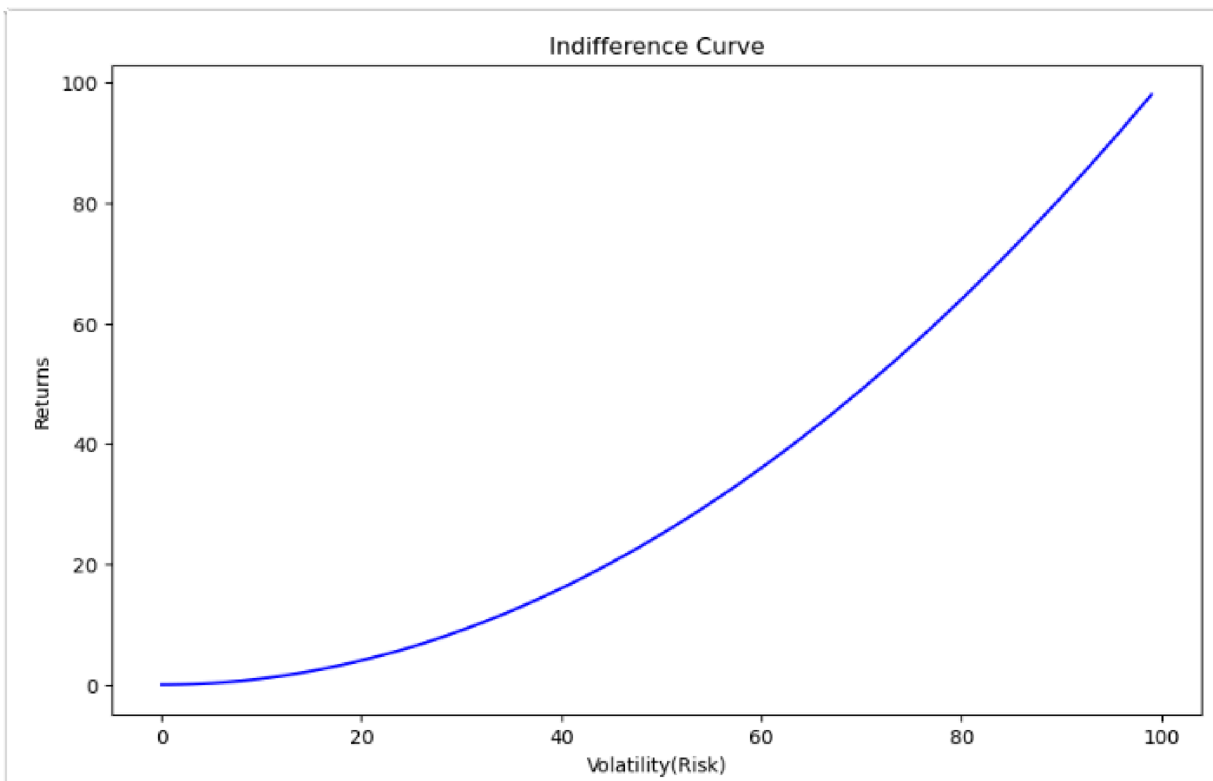
6)Indifference Curve:

The indifference curve represents the investor's preference for risk and return. For constructing the Markowitz Market Index, the investor's risk aversion coefficient was assumed to be moderate, indicating a balanced preference for risk and return.

```
In [36]: def generate_indifference_curve(mean_returns, cov_matrix, c1,c2):
             num = len(mean_returns)
             weights = np.random.rand(num)
             portfolio_returns = np.dot(weights, mean_returns)
             portfolio_risks =np.dot(weights.T, np.dot(cov_matrix, weights))
             value = c1*portfolio_returns + c2*portfolio_risks**2

             return value, portfolio_returns, portfolio_risks
         c1 = 0.4
         c2 = 0.01
         indifference_value, indifference_returns, indifference_risks = generate_indifference_curve(mean_returns, cov_matrix, c1, c2)

         #plotting the indifference curve
         X = np.arange(0, max(indifference_risks, 100))
         Y = c1* indifference_returns + c2*X**2
         plt.figure(figsize=(10,6))
         plt.title('Indifference Curve')
         plt.xlabel('Volatility(Risk)')
         plt.ylabel('Returns')
         plt.plot(X,Y, 'b-')
         plt.show()
```



7. Comparison with Actual Index Weights

The theoretical weights obtained from the optimization were compared with the actual weights of the NIFTY Financial Services Index. This comparison helps in understanding the difference between a theoretically optimal portfolio and the actual market index.

Index weights:

Company's Nam	Weight (%)
Axis Bank Ltd.	9.65
Bajaj Finance Ltd.	4.87
Bajaj Finserv Ltd.	2.41
Cholamandalam In	1.52
HDFC Asset Manag	1.04
HDFC Bank Ltd.	30.61
HDFC Life Insuranc	1.49
ICICI Bank Ltd.	21.63
ICICI Lombard Gen	1.05
ICICI Prudential Lif	0.59
IDFC Ltd.	0.39
Kotak Mahindra Ba	6.73
LIC Housing Financ	0.98
Muthoot Finance L	0.51
Power Finance Cor	1.91
REC Limited	1.92
SBI Cards and Payn	0.60
SBI Life Insurance (	1.73
Shriram Finance Lt	1.68
State Bank of India	8.70

Optimised Weights:

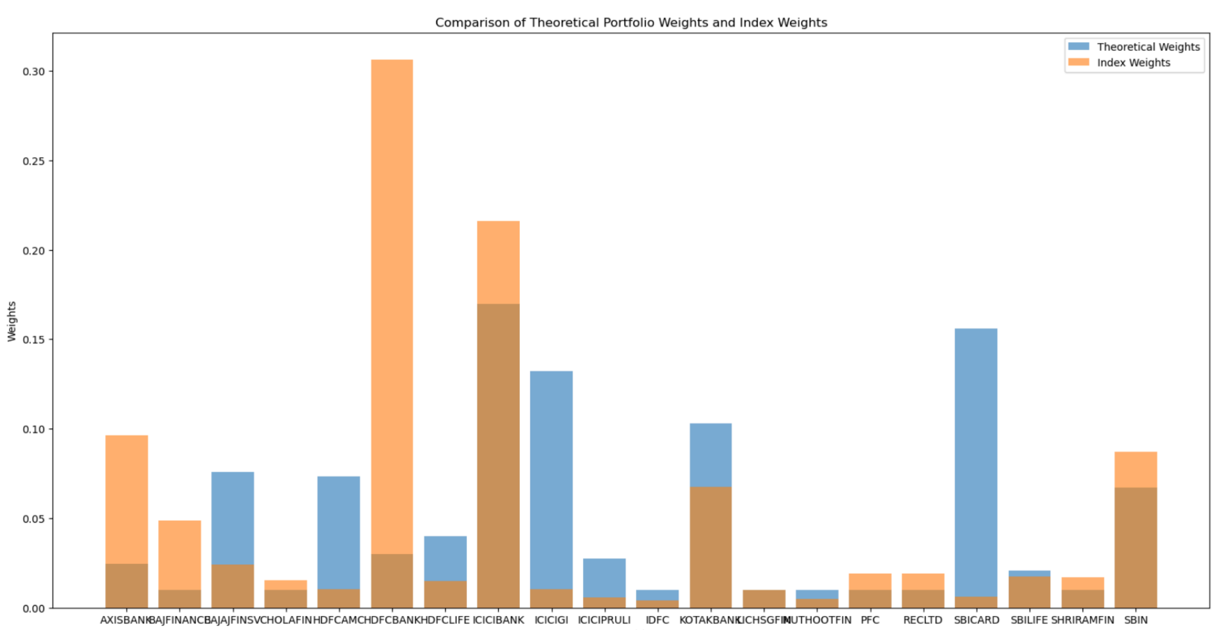
	Weight
AXISBANK	0.024585
BAJFINANCE	0.010000
BAJAJFINSV	0.075995
CHOLAFIN	0.010000
HDFCAMC	0.073357
HDFCBANK	0.029870
HDFCLIFE	0.039593
ICICIBANK	0.169473
ICICIGI	0.132060
ICICIPRULI	0.027466
IDFC	0.010000
KOTAKBANK	0.103146
LICHSGFIN	0.010000
MUTHOOTFIN	0.010000
PFC	0.010000
RECLTD	0.010000
SBICARD	0.156172
SBILIFE	0.020807
SHRIRAMFIN	0.010000
SBIN	0.067475

```
min_var_weights =[0.02456664, 0.01,       0.07602875, 0.01,       0.07325801, 0.02990561,
 0.03975534, 0.16958284, 0.13207463, 0.02752538, 0.01,       0.10293874,
 0.01,       0.01,       0.01,       0.01,       0.15616696, 0.02091523,
 0.01,       0.06728187]
```

```
import matplotlib.pyplot as plt

# Actual weights (example data)
actual_weights = [0.0965,
0.0487,
0.0241,
0.0152,
0.0104,
0.3061,
0.0149,
0.2163,
0.0105,
0.0059,
0.0039,
0.0673,
0.0098,
0.0051,
0.0191,
0.0192,
0.006,
0.0173,
0.0168,
0.087]  # Example values
securities = data.columns

# Plot comparison
plt.figure(figsize=(20, 10))
plt.bar(securities, min_var_weights, alpha=0.6, label='Theoretical Weights')
plt.bar(securities, actual_weights, alpha=0.6, label='Index Weights')
plt.xlabel('Securities')
plt.ylabel('Weights')
plt.title('Comparison of Theoretical Portfolio Weights and Index Weights')
plt.legend()
plt.show()
```



8. Final Thoughts

The method of creating a minimum variance portfolio for the NIFTY Financial Services Index is illustrated in this report. Through the utilisation of historical data, computation of returns and covariances, and optimisation of weights, we were able to effectively identify a portfolio that minimised risk. Compared to the real index weights, the comparison sheds light on how theoretical models are actually applied in the field of portfolio management.

Assumptions

- The historical price data accurately represents the future performance of the securities.
- The covariance matrix remains stable over time.
- No transaction costs or taxes are considered in the optimization.

Excel files of calculations:

Minimum variance portfolio.xlsx

References

ind_Nifty_Financial_Services.pdf (niftyindices.com)

https://www.nseindia.com/