CAPM Web Application

Vipin Kumar

13 Aug 2023

Introduction to the Capital Asset Pricing Model (CAPM) with Python

The Capital Asset Pricing Model (CAPM) is one of the most widely used formulas in finance.

We'll first look at the theory and intuition behind CAPM and then we'll review how to calculate it with Python, both for an individual stock and a portfolio of stocks.

Then, we created a web application using the Python library *Streamlit*, which aimed at performing CAPM calculations for different stocks from a market index, *NIFTY50*. The purpose of this application is to empower investors with the ability to make well-informed decisions by analyzing the risk and return associated with their investments.

Understanding the CAPM

The Capital Asset Pricing Model (CAPM) describes the relationship between the expected return of assets and the systematic risk of the market.
CAPM indicates that the expected return of an asset is equal to the risk-free return plus a risk premium. The assumption of CAPM is that investors are rational and want to maximize return and reduce risk as much as possible. The goal of CAPM is thus to calculate what return an investor can expect to make for a given risk premium over the risk-free rate.

Formula:

$$r_i = r_f + \beta_i (r_m - r_f)$$

Where:

- r_i = Expected return of the asset
- r_f = Risk-free rate of return
- β_i = Beta between the stock and the market
- r_m = Expected return of the market
- $(r_m - r_f)$ = Risk premium

Key Components

Risk-Free Asset Return (r_f) -

The theoretical return of an investment with zero risk, such as Bank Fixed Deposit (FD), Public Provident Fund (PPF), National Pension Scheme (NPS), and Gold are some risk-free assets in India.

Market Portfolio Return (r_m) -

The market portfolio return refers to the overall rate of return earned by an investment portfolio that consists of all available assets in the market, weighted according to their market values. The market portfolio return serves as a benchmark against which the performance of individual investments or portfolios can be compared. If an investment or portfolio earns a return higher than the market portfolio's return, it's considered to have outperformed the market, while if it earns a lower return, it's considered to have underperformed.

Market Risk Premium ($r_m - r_f$) -

The market risk premium is a fundamental concept in finance that represents the additional return that investors expect to receive for holding a risky investment (such as stocks) over a risk-free investment (such as government bonds).

Beta (β_i) -

Beta (β) is a measure of a financial asset's sensitivity to market movements. It quantifies the relationship between the price movements of the asset (such as stock) and the overall market's movements. In other words, beta indicates how much an asset's returns tend to move in response to changes in the broader market.

- $\beta = 0$: means returns are unrelated to market movements (risk-free asset).
- $\beta = 1$: means returns move in line with the market (market-average risk).
- $\beta > 1$: means returns are more volatile than the market (higher than average risk).
- $\beta < 1$: means returns are less volatile than the market (lower than average risk).
- $\beta < 0$: means returns move in the opposite direction of the market

Expected Return (r_i) -

Expected return refers to the anticipated average return that an investor can reasonably expect to earn from holding a financial asset or investment over a specific period of time.

Example of CAPM

Below is an example of the CAPM formula for ITC stock, where we'll make the following (made-up) assumptions:

- The risk-free rate (r_f) is 6.8%
- The market portfolio return (r_m) is 15.0348%
- The Beta (B_i) of ITC is 2.92 (more volatile than the market as whole)

Now, we can calculate the CAPM of ITC stock as -

$$r_i = 0.068 + 2.92(0.150348 - 0.068) = 0.1156 = 11.56\%$$

This formula suggests that an investor should expect a return of 11.56% to compensate for the additional risk they're taking. The risk premium of ITC stock is 4.76%, means that investors are willing to accept a 4.76% higher return on their investment in ITC stock to compensate them for the additional risk of investing in the stock.

Statistics Formulas Used

Min-Max Normalization -

Min-max normalization, also known as feature scaling, is a data preprocessing technique commonly used in machine learning and statistics. This technique scales the values of a feature to a range between 0 and 1. This is done by subtracting the minimum value of the feature from each value and then dividing it by the range of the feature.

Formula:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Where:

$X_{normalized}$ = normalized value
X = original stock price
X_{min} = minimum stock price for the given period
X_{max} = maximum stock price for the given period

Daily Return -

Daily return refers to the percentage change in the value of a financial asset, such as a stock or an index, over a single trading day. It's a measure of how much an asset's price has moved up or down in a single day's trading session.

Formula:

$$Daily\ Return\ =\ \frac{Today\ Value - Yesterday\ Value}{Yesterday\ Value} \times 100$$

The formula measures the percentage change in the asset's price from the previous trading day to the current trading day. If the result is positive, it indicates a price increase (positive return), and if the result is negative, it indicates a price decrease (negative return). Daily returns are commonly used to analyze short-term price movements, assess market volatility, and track the performance of individual assets or investment portfolios.

Covariance -

It is a measure of the relationship between two random variables such that a change in one variable is equal to a change in another variable.

Formula:

$$cov(x,\ y)\ =\ \frac{\sum((x - x_{mean})(y - y_{mean}))}{n - 1}$$

Where:

cov(x, y) = covariance between variable x and y
x = data value of x
y = data value of y
x_{mean} = mean of x
y_{mean} = mean of y
n = number of data values

If the covariance between two variables is positive, it means that they tend to increase or decrease together, indicating a positive relationship. If the covariance is negative, it means that as one variable tends to increase, the other tends to decrease, indicating a negative relationship. A covariance of zero suggests that there is no linear relationship between the variables.

Variance -

It is used to understand the variability and distribution of data. A high variance indicates that the data points are more spread out from the mean, while a low variance indicates that the data points are closer to the mean. It's important to note that variance is a non-negative value; it's always greater than or equal to zero.

Formula:

$$Var = \frac{\sum(x - x_{mean})^2}{n - 1}$$

x = value of one observation
x_{mean} = mean value of all observations
n = no. of observations

Beta Value -

Beta value measures the stock's volatility in relation to the overall market. A beta greater than 1 indicates higher volatility than the market, while a beta less than 1 suggests lower volatility.

Formula:

$$Beta = \frac{Covariance(stock\_return, market\_return)}{Variance(market\_return)}$$

Portfolio Weight -

A portfolio is a collection of financial assets (such as stocks, bonds, or other investments) that an investor holds. The portfolio weights determine the proportion of the total investment allocated to each individual asset within the portfolio. Portfolio weights are determined based on the allocation of funds to different assets within a portfolio. The method used to find portfolio weights is the *Equal Weights* method.

Formula:

$$Equal\ Weight\ for\ each\ asset = \frac{1}{number\ of\ assets}$$

Assign an equal portion of the total investment to each asset in the portfolio. For example, if you have 4 assets, each asset might receive a portfolio weight of 0.25 (25%).

Capital Asset Pricing Model with Python

Importing Libraries: First, import the necessary libraries that necessary for analysis. These include NumPy for numerical calculations, Pandas for data manipulation, Plotly Express for interactive visualizations, Streamlit for creating the web application, yfinance for fetching historical stock data, and datetime for handling date-related operations.

```
import numpy as np
import pandas as pd
import plotly.express as px
import streamlit as st
import yfinance as yf
import datetime as dt
```

Configuring Streamlit: Configure the appearance of Streamlit web application by setting the page title, icon, and layout to 'wide'.

```
st.set_page_config(page_title="Market Insight Analysis",
page_icon="chart_with_upwards_trend", layout='wide')
```

Adding Title: Adding a title to my web application to introduce the project

```
st.title("Capital Asset Pricing Model")
```

User Inputs: Created two columns to display user inputs. In the first column (col1), users are allowed to select the stocks they want to analyze from a multi-select dropdown. In the second column (col2), users can specify the number of years of historical data they want to consider.

```
col1, col2 = st.columns([3, 1])
with col1:
    stock_list = st.multiselect("Choose your stocks", ('ADANIENT.NS',
    'ADANIPORTS.NS', 'APOLLOHOSP.NS', 'ASIANPAINT.NS', 'AXISBANK.NS',
    'BAJAJ-AUTO.NS', 'BAJFINANCE.NS', 'BAJAJFINSV.NS', 'BPCL.NS',
    'BHARTIARTL.NS', 'BRITANNIA.NS', 'CIPLA.NS', 'COALINDIA.NS',
    'DIVISLAB.NS', 'DRREDDY.NS', 'EICHERMOT.NS', 'GRASIM.NS',
    'HCLTECH.NS', 'HDFCBANK.NS', 'HDFCLIFE.NS', 'HEROMOTOCO.NS',
    'HINDALCO.NS', 'HINDUNILVR.NS', 'HDFC.NS', 'ICICIBANK.NS',
    'ITC.NS', 'INDUSINDBK.NS', 'INFY.NS', 'JSWSTEEL.NS',
    'KOTAKBANK.NS', 'LT.NS', 'M&M.NS', 'MARUTI.NS', 'NTPC.NS',
    'NESTLEIND.NS', 'ONGC.NS', 'POWERGRID.NS', 'RELIANCE.NS',
    'SBILIFE.NS', 'SBIN.NS', 'SUNPHARMA.NS', 'TCS.NS',
    'TATACONSUM.NS', 'TATAMOTORS.NS', 'TATASTEEL.NS', 'TECHM.NS',
    'TITAN.NS', 'UPL.NS', 'ULTRACEMCO.NS', 'WIPRO.NS'),
    ['ICICIBANK.NS', 'ITC.NS', 'INDUSINDBK.NS', 'INFY.NS',
    'JSWSTEEL.NS'])
with col2:
    years = st.number_input("Number of years", 1, 10)
```

Data Retrieval and Preprocessing:

Calculate the start and end dates for the historical data using the user-provided number of years.
Fetch historical data for the NIFTY 50 index and reset its index to have the 'Date' as a column.
Create a DataFrame with 'Date' and 'Close' columns for the NIFTY 50 index.
Fetch close price data for selected stocks and reset their index.
Merge the stock data with the NIFTY 50 data to create a final DataFrame for analysis.

```
start_date = dt.date(dt.date.today().year - years,
dt.date.today().month, dt.date.today().day)
end_date = dt.date.today()

nifty50_data = yf.download('^NSEI', start=start_date, end=end_date)
nifty50_data.reset_index(inplace=True)
data1 = nifty50_data[['Date', 'Close']]
nifty50 = data1.copy()
nifty50.rename(columns={'Close': 'Nifty50'}, inplace=True)
```

```
stock_df = pd.DataFrame()
for stock in stock_list:
    stocks_data = yf.download(stock, period=f'{years}y')
    stock_df[f'{stock}'] = stocks_data['Close']
stock_df.reset_index(inplace=True)
stocks_df = pd.merge(stock_df, nifty50, on='Date', how='inner')
```

Displaying Data: Displaying the head and tail of the stock data using Streamlit's st.dataframe() function within two columns (col3 and col4).

```
col3, col4 = st.columns([1, 1])
with col3:
    st.markdown("### Stocks Data Head")
    st.dataframe(stocks_df.head(), use_container_width=True)
with col4:
    st.markdown("### Stocks Data Tail")
    st.dataframe(stocks_df.tail(), use_container_width=True)
```

Creating Interactive Plots:

Define a function plot() to create an interactive line chart using Plotly Express for the selected stocks' prices over time.
Define another function normalize() to perform min-max normalization on the stock prices.
Display two line charts using the plot() function, one for actual stock prices and another for normalized stock prices.

```
def plot(df):
    fig1 = px.line()
    for i in df.columns[1:]:
        fig1.add_scatter(x=df['Date'], y=df[i], name=i)
        fig1.update_layout(width=700, margin=dict(l=20, r=20, t=50,
        b=20), legend=dict(orientation='h', yanchor='bottom', y=1.02,
        xanchor='right',x=1))
    return fig1
```

```
def normalize(df2):
    for i in df.columns[1:]:
        min_val = df[i].min()
        max_val = df[i].max()
        df[i] = (df[i] - min_val) / (max_val - min_val)
    return df

col5, col6 = st.columns([1, 1])
with col5:
    st.markdown('### Stock Price Trends over Time')
    st.plotly_chart(plot(stocks_df), use_container_width=True)
with col6:
    st.markdown('### Normalized Stock Price Trends over Time')
    st.plotly_chart(plot(normalize(stocks_df)),
use_container_width=True)
```

Calculating Daily Returns:

Define a function stocks_daily_return() to calculate the daily return of each asset.
Calculate daily returns for stocks and the NIFTY 50 index and drop any rows with NaN values.

```
def stocks_daily_return(df):
    df_daily_return = df.copy()
    for i in df.columns[1:]:
        df_daily_return[i] = ((df[i] - df[i].shift(1)) /
        df[i].shift(1)) * 100
    return df_daily_return

daily_return = stocks_daily_return(stocks_df).dropna()
```

Calculating CAPM:

Calculate the market return, covariance matrix, market variance, beta values, and expected returns using the CAPM formula.
Display daily return and market return data using the st.dataframe() function within columns col7, col8.

```
# Calculate Daily Return for stocks and market(NIFTY50) over time
daily_return = stocks_daily_return(stocks_df).dropna()

# Calculate Market Return
market_return = daily_return['Nifty50']

# Calculate Covariance for each stock
cov_matrix = daily_return.iloc[:,1:].cov() # Covariance matrix

# Calculate Variance for NIFTY50
market_variance = np.var(market_return)

# To store beta values for each stock
beta_value = {}

# Calculating Beta values for each stock
for column in daily_return.iloc[:,1:].columns:
    covariance = cov_matrix[column]['Nifty50']
    beta = covariance / market_variance
    beta_value[column] = beta

# Calculating market portfolio return (rm)
market_portfolio_return = market_return.mean() * 252

# Considering Risk-Free Rate of Return (rf) as 0
risk_free_rate = 0.068

# Calculate Expected Return (ri) using CAPM
expected_return = {}
for stock, beta in beta_value.items():
    expected_returns = risk_free_rate + beta *
    (market_portfolio_return - risk_free_rate)
    expected_return[stock] = expected_returns

# Converting dict to dataframes
beta_df = pd.DataFrame.from_dict(beta_value, orient='index',
columns=['Beta Value'])
beta_df.reset_index(inplace=True)
beta_df.rename(columns={'index': 'Stocks'}, inplace=True)
```

```
expected_return_df = pd.DataFrame.from_dict(expected_return,
orient='index', columns=['Expected Return (in %)'])
expected_return_df.reset_index(inplace=True)
expected_return_df.rename(columns={'index': 'Stocks'}, inplace=True)

col7, col8 = st.columns([1, 1])
with col7:
    data2 = daily_return.iloc[:, :-1]
    st.markdown("### Daily Returns for Different Stocks")
    st.dataframe(data2.sort_values(by='Date',
ascending=False).head(), use_container_width=True)
with col8:
    st.markdown("### Market Returns for NIFTY50")
    st.dataframe(daily_return[['Date',
'Nifty50']].sort_values(by='Date', ascending=False).head(),
use_container_width=True)
```

Creating Bar Charts for Returns:

Define functions plot2() and plot3() to create interactive bar charts using Plotly Express for daily return trends and market return trends. Display the bar charts for a selected stock and the market return within columns col9 and col10.

```
def plot2(df, col):
    color ='#0072BB'
    fig2 = px.bar(df, x=df['Date'], y=df[col],
    color_discrete_sequence=[color])
    fig2.update_layout(width=700, margin=dict(l=20, r=20, t=50,
    b=20), xaxis_title='', yaxis_title='')
    return fig2

def plot3(df):
    color = '#004B8D'
    fig3 = px.bar(df, x=df['Date'], y=df['Nifty50'],
    color_discrete_sequence=[color])
    fig3.update_layout(height=500, width=700, margin=dict(l=20, r=20,
    t=50, b=20), xaxis_title='', yaxis_title='')
    return fig3
```

```
col9, col10 = st.columns([1, 1])
with col9:
    st.markdown("### Daily Return Trends")
    stock_lst = st.selectbox("Choose a stock", (stock_list))
    st.plotly_chart(plot2(daily_return.iloc[:, :-1], stock_lst),
use_container_width=True)
with col10:
    st.markdown("### Market Return Trends")
    st.plotly_chart(plot3(daily_return), use_container_width=True)
```

Displaying Beta Values and Expected Returns:

Display calculated beta values and expected returns in separate columns (col11 and col12) using the st.dataframe() function.

```
col11, col12 = st.columns([1, 1])
with col11:
    st.markdown("### Calculated Beta Value")
    st.dataframe(beta_df, use_container_width=True)
with col12:
    st.markdown("### Calculated Expected Returns using CAPM")
    st.dataframe(expected_return_df, use_container_width=True)
```

Creating Bar Charts for Beta and Expected Returns:

Define functions plot4() and plot5() to create bar charts for beta values and expected returns using Plotly Express. Display these bar charts within columns col13 and col14.

```
def plot4(df):
    fig4 = px.bar(x=df['Stocks'], y=df['Beta Value'])
    fig4.update_layout(width=700, xaxis_title='', yaxis_title='')
    return fig4

def plot5(df):
   fig5 = px.bar(x=df['Stocks'], y=df['Expected Return (in %)'],
   color_discrete_sequence=[color])
   fig5.update_layout(width=700, xaxis_title='', yaxis_title='')
   return fig5
```

```
col13, col14 = st.columns([1, 1])
with col13:
    st.markdown("### Beta Values for Different Stocks and Market")
    st.plotly_chart(plot4(beta_df), use_container_width=True)
with col14:
    st.markdown("### Expected Returns for Different Stocks and
Market")
    st.plotly_chart(plot5(expected_return_df),
use_container_width=True)
```

Calculating Expected Return for Portfolio:

Calculate the expected return for a portfolio using the weights of selected stocks and their respective expected returns.
Display the conclusion about the expected return based on CAPM for the portfolio.

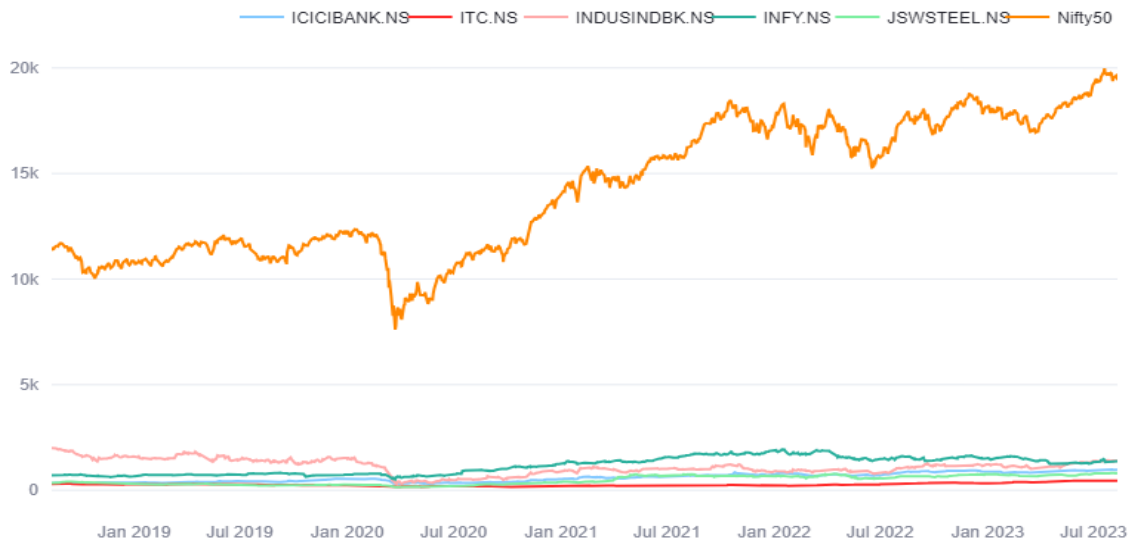
```
n = len(stocks_df.columns) - 1
portfolio_weights = 1/n * np.ones(n)
er_portfolio = sum(np.array(list(expected_return.values())) *
portfolio_weights)

st.subheader(f'Conclusion: The Expected Return Based on CAPM for the
portfolio is roughly {round(er_portfolio, 2)}%')
```

Error Handling: In case of any issues, Display a message asking the user to select valid stocks or refresh the page.

```
except:
    st.write("Please select valid stock or refresh the web page...")
```

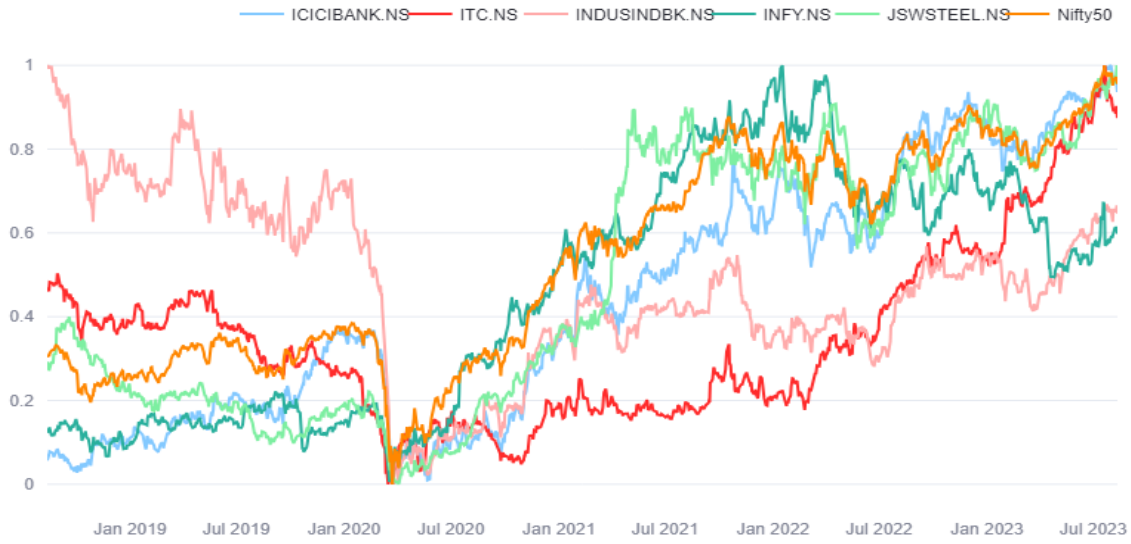
Visualizing Insights

Stock Prices Over Time -



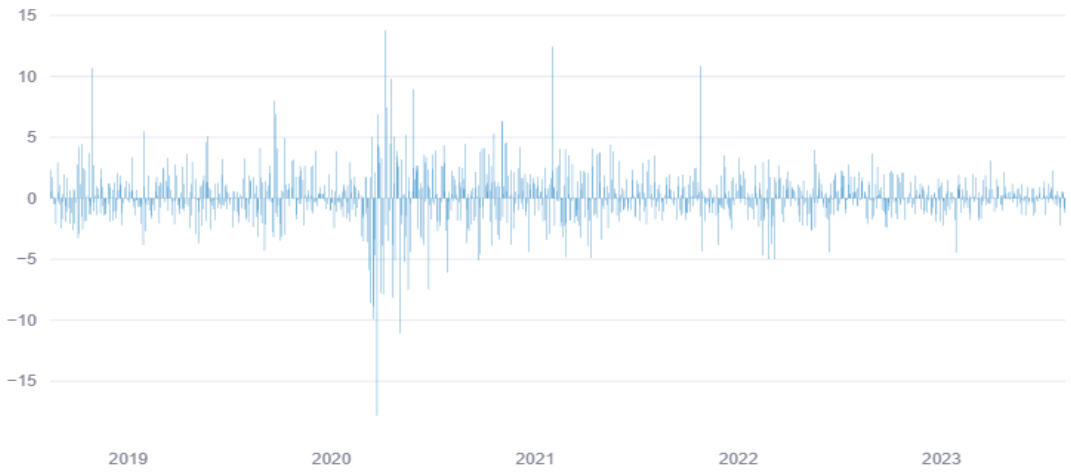
This line chart shows how the prices of stock and the market have changed over the given time period. It gives an overview of the price trends for each stock and the market index.

Normalized Stock Price Trends over Time -



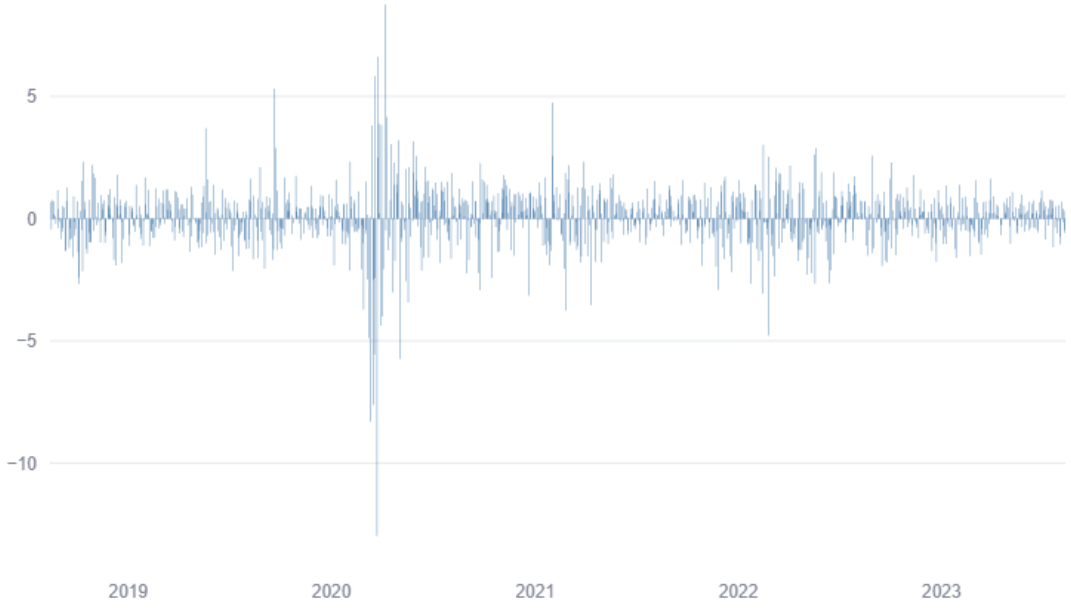
This chart will show how the stock price evolves over time on a scale of 0 to 1, allowing you to compare its performance regardless of the original price range.

Daily Returns Over Time -



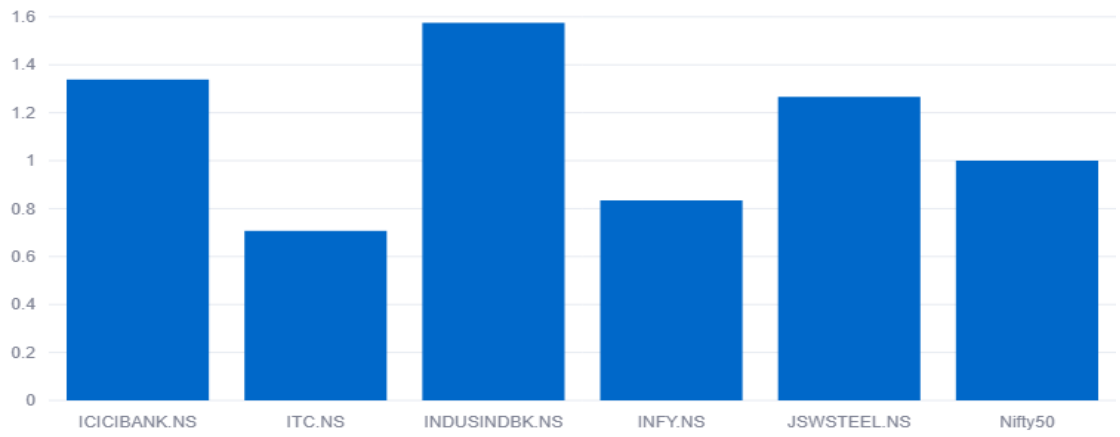
This line chart illustrates the daily returns of stocks over time. It shows how the returns of each stock and the market have fluctuated(in %) on a daily basis.

Market Return (Nifty50) Over Time -



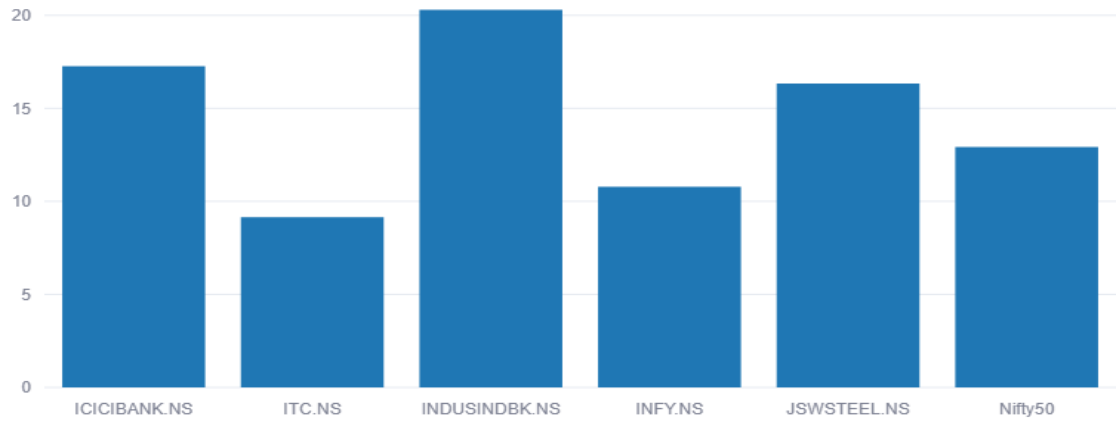
This line chart focuses specifically on the market return (Nifty50) over time. It visualizes how the market has performed and provides insights into its volatility and trends.

Beta Values for Different Stocks -



This bar chart compares the beta values of each stock. Beta measures the sensitivity of a stock's returns to the market returns. A beta greater than 1 indicates the stock is more volatile than the market, while a beta less than 1 indicates lower volatility.

Expected Returns for Different Stocks -



This bar chart compares the expected returns (in percentage) of each stock. Expected return represents the projected return an investor can expect from holding stock. It helps in comparing potential gains from different investment options.

Limitations

It is important to note that the CAPM is only a theoretical model and it is not always accurate in predicting the actual return of a stock. The actual return of a stock can be affected by a number of factors, such as unexpected events, changes in investor sentiment, and changes in the market environment. As a result, it is important to use the CAPM as a guide, but not as a definitive prediction of the actual return of a stock. Moreover, criticisms exist regarding its simplicity and inability to fully elucidate real-world asset return variations.

Conclusion

Despite its limitations, CAPM remains a fundamental pillar of modern finance and investment theory. It forms the basis for comprehending the intricate interplay between risk and returns, thereby offering a framework for estimating required rates of return for diverse assets within the context of portfolio construction.

The code provided presents a comprehensive analysis using the Capital Asset Pricing Model (CAPM) for selected stocks' historical data. Users can choose specific stocks and define the historical timeframe they wish to analyze. The line charts vividly illustrate trends in stock prices over time, offering insights into their performance. Normalized trends provide a relative perspective on stock price movements. Daily return analysis showcases the volatility and growth patterns of each stock and how they compare to the overall market return (NIFTY 50). The calculated beta values indicate each stock's market sensitivity, helping users gauge risk. Furthermore, the CAPM-based expected returns offer a valuable lens to assess investment opportunities. Concluding with the expected return of a selected stock portfolio, the code equips users to make informed investment decisions. Notably, error handling ensures a smooth experience even when issues arise. In essence, this code-based analysis delivers a powerful toolset for comprehending stock performance, risk, and potential returns, thereby aiding investment decision-making processes.