ABHINAV KADAM – theabhinavkadam222@gmail.com

Capital asset pricing model

"Capital Asset Pricing Model (CAPM)," and it is a web application built using the Streamlit library in Python. The application allows users to analyze and visualize financial data for selected stocks, calculate their beta values, and estimate expected returns using the Capital Asset Pricing Model (CAPM).

Libraries Which used

- Importing libraries
- Streamlit
- Pandas
- Yfinance
- Pandas_datareader.data
- Datetime
- Capm_functions

1) Streamlit: =

- Implementing machine learning models and data visualization.
- Using Streamlit, a popular framework for creating interactive web applications.
- Streamlining the process of building and deploying machine learning applications.
- Providing a user-friendly interface for development.

2) Pandas

- used for working with data sets. It has functions for analysing, cleaning, exploring, and manipulating data

3) yfinance

- library offers Python users a seamless way to retrieve stock data from Yahoo Finance

4) pandas_datareader.data

- library uses multiple web sources to import data into a pandas dataframe

5) datetime library

- a versatile tool that allows programmers to manipulate dates and times in their code

6) capm functions

- calculates the expected rate of return for an asset or investment

Let's go through each library and explain its role in the CAPM application:

1. streamlit as st:

- Streamlit is the primary library used for building the web application interface. It provides a simple and efficient way to create interactive web applications using Python.
- In the CAPM application, Streamlit is used for the following tasks: Creating the main application layout and title (st.set_page_config(), st.title())
- Displaying user input widgets for selecting stocks and specifying the number of years (st.multiselect(), st.number_input())
- Displaying the resulting data and visualizations (st.dataframe(), st.plotly_chart())
- Rendering Markdown text for section headers and descriptions (st.markdown())
- Handling error messages (st.write())

2)pandas as pd:

- Pandas is a powerful data manipulation and analysis library in Python.
- In the CAPM application, Pandas is used for the following tasks: Creating and manipulating DataFrames to store the retrieved stock price data and S&P 500 index data
- Merging the stock data and S&P 500 index data into a single DataFrame (pd.merge())
- Handling data types and date formatting within the DataFrames
- Displaying the head and tail of the merged DataFrame

3) yfinance as yf:

- yfinance is a Python library that provides access to stock data from Yahoo Finance.
- In the CAPM application, yfinance is used to retrieve historical stock price data for the selected stocks (yf.download())

4) pandas_datareader.data as web: pandas_datareader is a Python library that provides a consistent way to download data from various sources.

- The data module from pandas_datareader is used in the CAPM application to retrieve the S&P 500 index data from the Federal Reserve Economic Data (FRED) API (web.DataReader())

5) datetime: datetime is a built-in Python module for working with dates and times.

- In the CAPM application, datetime is used to calculate the start and end dates for retrieving the stock and index data based on the number of years specified by the user

6)capm_functions: capm_functions is a custom Python module defined in the capm_functions.py file.

- This module contains several utility functions used throughout the CAPM application:
- interactive_plot(): Creates an interactive line plot using Plotly Express to visualize stock prices over time
- normalize(): Normalizes the stock prices by dividing each price by the initial price
- daily_return(): Calculates the daily returns for each stock and the S&P 500 index

- calculate_beta(): Computes the beta value for a given stock by performing a linear regression between its daily returns and the S&P 500 daily returns

The CAPM application can efficiently retrieve and process stock data, create an interactive user interface, visualize data, perform calculations like beta estimation and expected return estimation, and present the results to the user in a user-friendly manner.

CAPM Application Working

User Input:

- The application prompts users to select up to four stocks from a predefined list (TSLA, AAPL, NFLX, MSFT, MGM, AMZN, NVDA, GOOGL).
- Users specify the number of years for which they want to analyze stock data.

Data Retrieval:

- Historical stock price data for selected stocks is fetched from Yahoo Finance using the yfinance library.
- Historical S&P 500 index data is retrieved from the Federal Reserve Economic Data (FRED) API using the pandas_datareader library.
- Data retrieved covers the specified period based on user input.

Data Preprocessing:

- Stock price data and S&P 500 index data are merged into a single DataFrame, aligning dates.
- Missing or misaligned data points are handled via an inner join on the date column.

Data Visualization:

- The application displays the initial and final rows of the merged DataFrame for user inspection.
- It generates two interactive line charts using Plotly Express:
  1. Shows stock prices over time for selected stocks and the S&P 500 index.
  2. Displays normalized stock prices to visualize relative performance.

Daily Returns Calculation:

- Daily returns for each stock and the S&P 500 index are calculated using the \`daily_return\` function from \`capm_functions.py\`.
- Daily return is computed as the percentage change in price between consecutive trading days.

Beta Calculation:

- Beta values for each selected stock are calculated by performing linear regression between the stock's daily returns and the S&P 500 daily returns.
- The \`calculate_beta\` function from \`capm_functions.py\` is used, utilizing NumPy's \`polyfit\` function to find the slope (beta) and intercept of the regression line.

- Calculated beta values are stored in a dictionary for further use.

Expected Return Estimation:

- Using calculated beta values and assuming a risk-free rate of 0, the application estimates expected returns for each stock based on the CAPM formula:
  Expected Return = Risk-free Rate + Beta × (Market Return - Risk-free Rate)
- Market return is calculated as the mean annualized return of the S&P 500 index during the specified period.

Results Display:

- The application presents calculated beta values and estimated expected returns in separate tables using Streamlit's data display functionality.
- Beta values are presented with corresponding stock names and numerical values.
- Expected returns are shown as percentages alongside corresponding stock names.

Error Handling:

- The application catches exceptions during data retrieval or processing steps, displaying error messages to users and prompting them to provide valid input.

Modular Approach:

- The application follows a modular approach with `capm_functions.py` containing reusable functions for tasks such as plotting, normalization, daily returns calculation, and beta computation.
- This separation of concerns promotes code organization and maintainability.

Technologies Used:

- Python libraries and frameworks utilized include Streamlit for the user interface, Pandas for data manipulation, yfinance and Pandas Datareader for data retrieval, NumPy for numerical computations, and Plotly Express for interactive data visualization.

Conclusion:

- By following these steps, the CAPM application offers a user-friendly interface for analyzing and applying the Capital Asset Pricing Model to a selected set of stocks. Users can visualize stock performance, calculate beta values, and estimate expected returns based on the CAPM formula.

Key Features

1. Stock Selection:

- Users can select up to four stocks from a predefined list (TSLA, AAPL, NFLX, MSFT, MGM, AMZN, NVDA, GOOGL).
- They can specify the number of years for which they want to analyze the data.

2. Data Retrieval:

- The application retrieves historical stock price data from Yahoo Finance and S&P 500 index data from the Federal Reserve Economic Data (FRED) API for the specified period.

3. Data Visualization:

- Interactive line charts using Plotly Express visualize the stock prices over time, both in their original form and after normalization.

4. Daily Returns Calculation:

- The application computes the daily returns for each selected stock and the S&P 500 index.

5. Beta Calculation:

- Beta values for each selected stock are calculated via linear regression between the stock's daily returns and the S&P 500 daily returns.

6. Expected Return Estimation:

- Utilizing calculated beta values and assuming a risk-free rate (set to 0 in the code), the application estimates the expected returns for each stock based on the Capital Asset Pricing Model (CAPM) formula.

7. Results Display:

- The application presents the calculated beta values and estimated expected returns in a tabular format for easy analysis.

ABHINAV KADAM – theabhinavkadam222@gmail.com

Project Report

Introduction

The Capital Asset Pricing Model (CAPM) is a widely used model in finance for determining the expected return of an asset based on its risk. This model provides a framework for estimating the appropriate required rate of return for an investment, considering its systematic risk relative to the overall market. The CAPM is based on the fundamental principle that investors should be compensated for taking on additional risk, and it establishes a relationship between risk and expected return.

The primary objective of this project is to develop a user-friendly web application that allows users to analyze and apply the CAPM to a selected set of stocks. By providing an interactive interface, users can visualize stock price data, calculate beta values, and estimate expected returns based on the CAPM formula.

Project Description

The CAPM application is built using Python and leverages several libraries and frameworks, including Streamlit, Pandas, yfinance, Pandas Datareader, NumPy, and Plotly Express. The application allows users to select up to four stocks from a predefined list (TSLA, AAPL, NFLX, MSFT, MGM, AMZN, NVDA, GOOGL) and specify the number of years for which they want to analyze the data.

The application retrieves historical stock price data from Yahoo Finance using the yfinance library and obtains S&P 500 index data from the Federal Reserve Economic Data (FRED) API using the Pandas Datareader library. The retrieved data is then preprocessed and merged into a single DataFrame for further analysis.

Key Features

1. Interactive Data Visualization: The application provides interactive line charts using Plotly Express to visualize the stock prices over time, both in their original form and after normalization. This allows users to observe the price trends and relative performance of the selected stocks.

2. Daily Returns Calculation: The application calculates the daily returns for each selected stock and the S&P 500 index. Daily returns provide insights into the short-term performance and volatility of the stocks.

3. Beta Calculation: The application computes the beta value for each selected stock by performing a linear regression between the stock's daily returns and the S&P 500 daily returns. Beta is a measure of systematic risk and is a crucial component of the CAPM formula.

4. Expected Return Estimation: Using the calculated beta values and assuming a risk-free rate (set to 0 in the provided code), the application estimates the expected returns for each stock based on the CAPM formula: `Expected Return = Risk-free Rate + Beta × (Market Return - Risk-free Rate)`. The market return is calculated as the mean annualized return of the S&P 500 index during the specified period.

5. Tabular Result Display: The application presents the calculated beta values and estimated expected returns in a tabular format for easy analysis and comparison.

Implementation Details

The CAPM application follows a modular approach, separating the core functionality into different components. The main application code (`capm_base_code`) handles user input, data retrieval, preprocessing, and result display. The utility functions required for various calculations and visualizations are defined in a separate module (`capm_functions.py`).

The application utilizes several Python libraries and frameworks:

- Streamlit: Used for building the web application interface, displaying user input widgets, and rendering data visualizations and results.
- Pandas: Employed for data manipulation and analysis, including creating and manipulating DataFrames, merging data, and handling data types and date formatting.
- yfinance: Utilized for retrieving historical stock price data from Yahoo Finance.
- Pandas Datareader: Used for downloading S&P 500 index data from the Federal Reserve Economic Data (FRED) API.
- NumPy: Leveraged for numerical computations, particularly for the linear regression used in beta calculation.
- Plotly Express: Employed for creating interactive line charts to visualize stock price data.

Results and Analysis

The CAPM application provides users with valuable insights into the risk and expected returns of selected stocks. By calculating beta values, users can assess the systematic risk of each stock relative to the overall market. The estimated expected returns, derived from the CAPM formula, offer a quantitative basis for evaluating investment opportunities and making informed decisions.

The interactive visualizations allow users to observe stock price trends and relative performance over time, enabling a deeper understanding of the stocks' behavior. The tabular presentation of beta values and expected returns facilitates easy comparison and analysis across different stocks.

Challenges and Limitations

During the development of the CAPM application, several challenges were encountered:

1. Data Retrieval and Preprocessing: Ensuring consistent and reliable data retrieval from multiple sources (Yahoo Finance and FRED API) and handling potential data discrepancies or missing values required careful data preprocessing techniques.

2. User Interface Design: Creating a user-friendly and intuitive interface that effectively communicates the application's functionality and presents the results in a clear and organized manner required careful consideration of user experience principles.

3. Performance Optimization: As the application handles financial data and performs calculations, optimizing the computational efficiency and minimizing latency was crucial for providing a smooth user experience.

Additionally, it is important to note that the CAPM has its own limitations and assumptions, such as the assumption of a perfectly efficient market, the use of a constant risk-free rate, and the potential oversimplification of risk factors. These limitations should be considered when interpreting and applying the results obtained from the application.

Future Enhancements

While the current CAPM application provides a solid foundation for analyzing and applying the Capital Asset Pricing Model, several enhancements can be explored to further improve its functionality and usefulness:

1. Real-time Data Updates: Incorporating real-time data updates for stock prices and market indices would enhance the application's relevance and accuracy.

2. Expanded Stock Selection: Allowing users to input their own stock tickers or select from a broader range of stocks and indices could increase the application's versatility and appeal to a wider audience.

3. Customizable Input Parameters: Enabling users to input their own risk-free rate and market return assumptions could provide more personalized and tailored results.

4. Portfolio Optimization: Extending the application to incorporate portfolio optimization techniques, such as the Markowitz model, would allow users to construct optimal portfolios based on risk and return preferences.

5. Advanced Risk Analysis: Integrating additional risk analysis techniques beyond CAPM, such as the Fama-French three-factor model or the Arbitrage Pricing Theory (APT), could provide a more comprehensive risk assessment.

6. User Account Management: Implementing user account management features could enable users to save their analyses, track their investment portfolios, and access historical data and results.

Possible Questions and Answers

1. Q: What is the Capital Asset Pricing Model (CAPM), and why is it important?

A: The CAPM is a model used in finance to determine the expected return of an asset based on its risk compared to the overall market. It's important because it helps investors assess the risk and return trade-off of investments.

2. Q: How does the CAPM application retrieve data?

A: The application fetches historical stock prices from Yahoo Finance and S&P 500 index data from the Federal Reserve Economic Data (FRED) API.

3. Q: What functions does the `capm_functions.py` module contain?

A: It contains functions for plotting, normalization, daily returns calculation, and beta computation.

4. Q: How does the application calculate beta values?

A: It calculates beta values by performing linear regression between the stock's daily returns and the S&P 500 daily returns.

5. Q: What formula does the application use to estimate expected returns based on CAPM?

A: It uses: `Expected Return = Risk-free Rate + Beta × (Market Return - Risk-free Rate)`, assuming a risk-free rate of 0.

6. Q: What are some challenges of the CAPM application?

A: Challenges include ensuring data consistency, handling missing values, and optimizing computational efficiency.

7. Q: Can you suggest potential enhancements for the CAPM application?

A: Enhancements could include real-time data updates, more stock options, customizable input parameters, and integration of advanced risk analysis models.

8. Q: What is the role of the Streamlit library in the CAPM application?

A: Streamlit is used to create the web interface, making it easy to build interactive web applications with Python.

9. Q: How does the application handle errors?

A: It catches exceptions during data retrieval or processing and displays error messages to the user.

10. Q: Why is modular design important in the CAPM application?

A: Modular design promotes code organization, maintainability, and reusability, making it easier to understand and extend the application.