Financial Risk Analysis - Apple and S&P 500

The Scenario

ABC company is about to buy shares in Apple and S&P 500 with the target to obtain optimum profit in the space of two years. But they are curious how the two companies have been performing for the past three years.

So, the manager approaches you as the Data Analyst in the ABC company; please can we know how the Apple and S&P 500 stock market have been for the past three years? We would like to know if the risk is minimal or unaffordable for us.

In Python, we can use libraries such as pandas for data manipulation, yfinance for fetching historical stock prices, and matplotlib for visualization. For more advanced financial analysis, we can use numpy for mathematical operations and pyfolio for perfomance analysis.

In [ ]:

```

```

In [1]:

```
#install Libraries
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
import numpy as np
import pyfolio
%matplotlib inline
```

In [2]:

```
#download data using yfinance
symbol = "AAPL"
start_date = "2020-01-01"
end_date = "2023-12-31"
data = yf.download(symbol,start=start_date,end=end_date)
```

```
[*********************100%***********************]  1 of 1 completed
```

In [3]:

```
#view data
data.head()
```

Out[3]:

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	74.059998	75.150002	73.797501	75.087502	73.059425	135480400
2020-01-03	74.287498	75.144997	74.125000	74.357498	72.349136	146322800
2020-01-06	73.447502	74.989998	73.187500	74.949997	72.925636	118387200
2020-01-07	74.959999	75.224998	74.370003	74.597504	72.582657	108872000
2020-01-08	74.290001	76.110001	74.290001	75.797501	73.750244	132079200

In [4]:

```
#Calculate the percentage change in the 'Adj Close' column of the DataFrame
# and assign the result to a new column named 'Returns'.
data['Returns']= data['Adj Close'].pct_change()
data.head()
```

Out[4]:

Date	Open	High	Low	Close	Adj Close	Volume	Returns
2020-01-02	74.059998	75.150002	73.797501	75.087502	73.059425	135480400	NaN
2020-01-03	74.287498	75.144997	74.125000	74.357498	72.349136	146322800	-0.009722
2020-01-06	73.447502	74.989998	73.187500	74.949997	72.925636	118387200	0.007968
2020-01-07	74.959999	75.224998	74.370003	74.597504	72.582657	108872000	-0.004703
2020-01-08	74.290001	76.110001	74.290001	75.797501	73.750244	132079200	0.016086

In [5]:

```
# Calculate the standard deviation of the returns in the DataFrame `data`
volatility = np.std(data['Returns'])
volatility
```

Out[5]:

```
0.021135399087142986
```

In [21]:
```
# Download historical market data for the S&P 500 index (^GSPC) using Yahoo
# for the specified start and end dates.

Market_data = yf.download("^GSPC", start= start_date, end = end_date)

# Join the 'Adj Close' column of the market data to the existing DataFrame `
# The added column is suffixed with "_Market" to differentiate it from exist

data = data.join(Market_data["Adj Close"],on = data.index, rsuffix="_Market"

# Calculate the percentage change in the adjusted close prices of the S&P 50
# to obtain the returns of the market, and assign the result to a new column

returns_market = Market_data["Adj Close"].pct_change()
data["Returns_Market"] = returns_market

data = data.loc[:,~data.columns.duplicated()]

#data["Returns_Market"] = data["Adj Close_Market"].pct_change()

data.head()
```

```
[*********************100%***********************]  1 of 1 completed
```

Out[21]:

Date	Open	High	Low	Close	Adj Close	Volume	Returns	Clos
2020-01-02	74.059998	75.150002	73.797501	75.087502	73.059425	135480400	NaN	32
2020-01-03	74.287498	75.144997	74.125000	74.357498	72.349136	146322800	-0.009722	32
2020-01-06	73.447502	74.989998	73.187500	74.949997	72.925636	118387200	0.007968	32
2020-01-07	74.959999	75.224998	74.370003	74.597504	72.582657	108872000	-0.004703	32
2020-01-08	74.290001	76.110001	74.290001	75.797501	73.750244	132079200	0.016086	32

In [12]:
```
market_data.head()
```

Out[12]:

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	3244.669922	3258.139893	3235.530029	3257.850098	3257.850098	3459930000
2020-01-03	3226.360107	3246.149902	3222.340088	3234.850098	3234.850098	3484700000
2020-01-06	3217.550049	3246.840088	3214.639893	3246.280029	3246.280029	3702460000
2020-01-07	3241.860107	3244.909912	3232.429932	3237.179932	3237.179932	3435910000
2020-01-08	3238.590088	3267.070068	3236.669922	3253.050049	3253.050049	3726840000

In [22]:

```
cov_matrix = np.cov(data["Returns"].dropna(),data["Returns_Market"].dropna()
beta = cov_matrix[0,1]/cov_matrix[1,1]

beta
```

Out[22]:

```
1.1896770019660008
```

In [25]:

```
sharpe_ratio =data["Returns"].mean()/volatility
sharpe_ratio
```

Out[25]:

```
0.0561412252535486
```

In [26]:

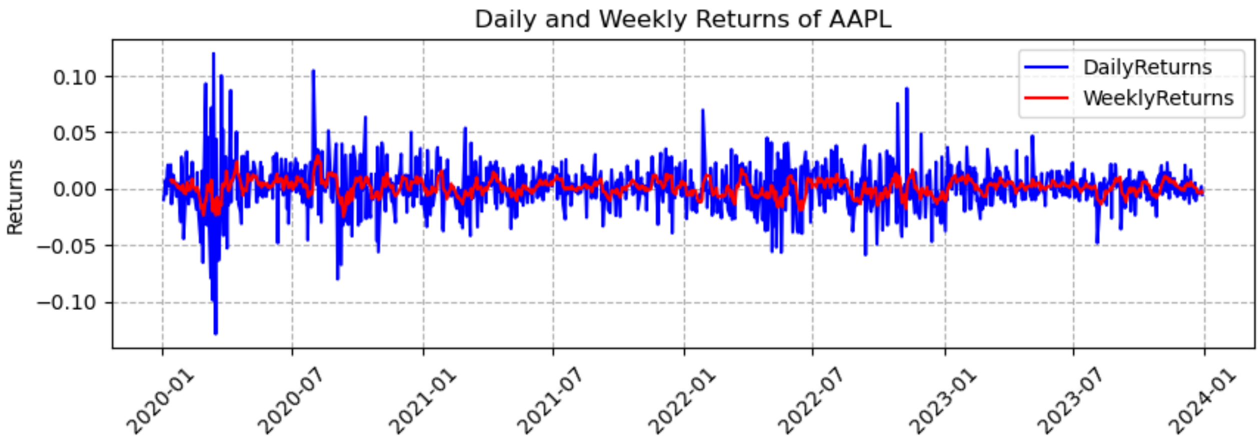
```
data["WeeklyReturns"]= data["Returns"].rolling(7).mean()
data["MonthlyReturns"]= data["Returns"].rolling(30).mean()
data.head()
```

Out[26]:

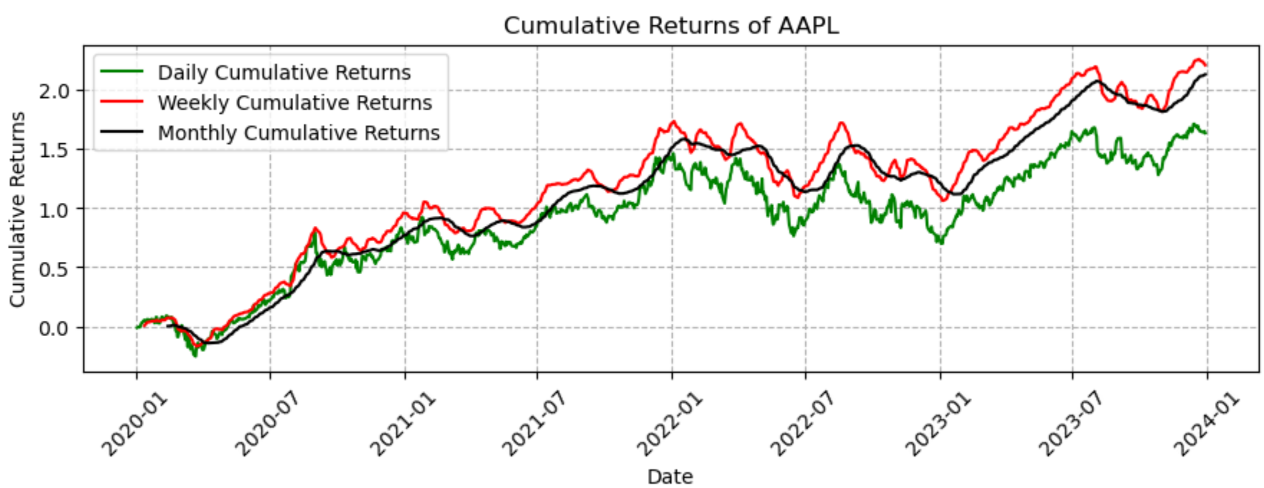
Date	Open	High	Low	Close	Adj Close	Volume	Returns	Clos
2020-01-02	74.059998	75.150002	73.797501	75.087502	73.059425	135480400	NaN	32
2020-01-03	74.287498	75.144997	74.125000	74.357498	72.349136	146322800	-0.009722	32
2020-01-06	73.447502	74.989998	73.187500	74.949997	72.925636	118387200	0.007968	32
2020-01-07	74.959999	75.224998	74.370003	74.597504	72.582657	108872000	-0.004703	32
2020-01-08	74.290001	76.110001	74.290001	75.797501	73.750244	132079200	0.016086	32

In [33]:
```
# visualization
plt.figure(figsize=(10,6))
#plot returns

plt.subplot(2,1,1)
plt.plot(data.index,data["Returns"],label="DailyReturns",color="blue")
plt.plot(data.index,data["WeeklyReturns"],label="WeeklyReturns",color="red")
plt.title("Daily and Weekly Returns of {}".format(symbol))
plt.ylabel("Returns")
plt.grid(linestyle="--")
plt.xticks(rotation=45)
plt.legend()
```

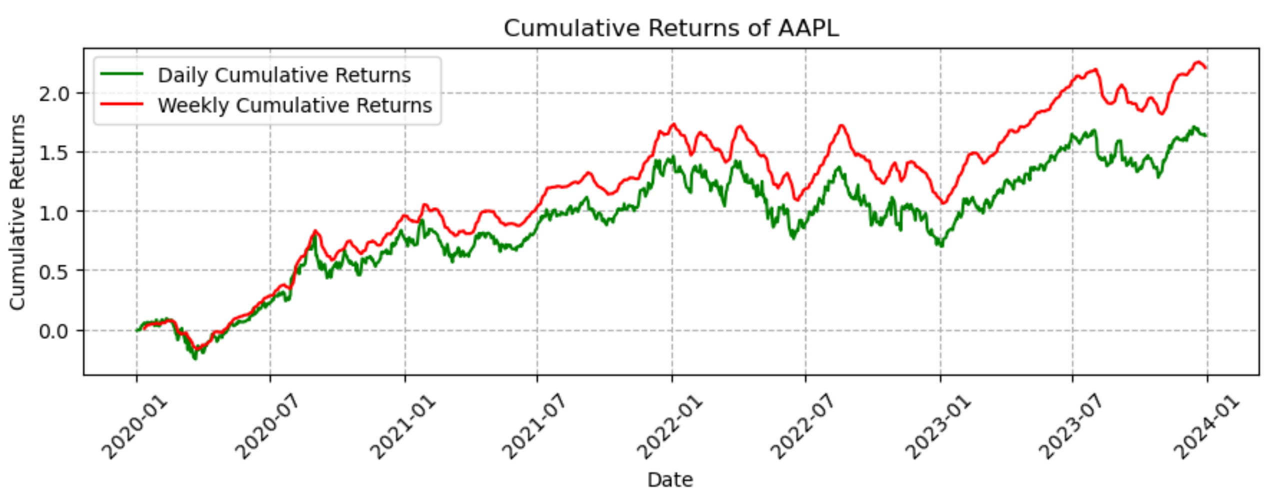
Out[33]:
```
<matplotlib.legend.Legend at 0x16aa22150>
```



In [36]:
```
plt.figure(figsize=(9, 6))

plt.subplot(2, 1, 2)

cumulative_returns = (1 + data["Returns"]).cumprod() - 1

cumulative_returns_w = (1 + data["WeeklyReturns"]).cumprod() - 1

cumulative_returns_m = (1 + data["MonthlyReturns"]).cumprod() - 1

plt.plot(data.index, cumulative_returns, label="Daily Cumulative Returns", c

plt.plot(data.index, cumulative_returns_w, label="Weekly Cumulative Returns"
plt.plot(data.index, cumulative_returns_m, label="Monthly Cumulative Returns

plt.title("Cumulative Returns of {}".format(symbol))

plt.xlabel("Date")

plt.ylabel("Cumulative Returns")

plt.legend()

plt.grid(linestyle='--')

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()
```



In [37]:
```
plt.figure(figsize=(9, 6))

plt.subplot(2, 1, 2)

cumulative_returns = (1 + data["Returns"]).cumprod() - 1

cumulative_returns_w = (1 + data["WeeklyReturns"]).cumprod() - 1

cumulative_returns_m = (1 + data["MonthlyReturns"]).cumprod() - 1

plt.plot(data.index, cumulative_returns, label="Daily Cumulative Returns", c

plt.plot(data.index, cumulative_returns_w, label="Weekly Cumulative Returns"
#plt.plot(data.index, cumulative_returns_m, label="Monthly Cumulative Return

plt.title("Cumulative Returns of {}".format(symbol))

plt.xlabel("Date")

plt.ylabel("Cumulative Returns")

plt.legend()

plt.grid(linestyle='--')

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()
```



In [41]:
```
#Calculate VaR

VaR = np.percentile(data["Returns"].dropna(), 5)

#Calculate Alpha I

model = np.polyfit(data["Returns_Market"].dropna(), data["Returns"].dropna()
alpha =model[1]

# Calculate Treynor Ratio

risk_free_rate = 0.02 # Assume a risk-free rate

treynor_ratio = (data["Returns"].mean() - risk_free_rate) / beta

# Calculate Maximum Drawdown

cumulative_returns = (1 + data["Returns"]).cumprod()

peak = cumulative_returns.cummax()
drawdown = (cumulative_returns - peak) / peak
max_drawdown = abs(drawdown.min())

#Print Results

print("Volatility:", volatility)

print("Beta:", beta)

print("Sharpe Ratio:", sharpe_ratio)

# Print Additional Results

print("VaR at 95% Confidence Level:", VaR)

print("Alpha:", alpha)

print("Treynor Ratio:", treynor_ratio)

print("Maximum Drawdown:", max_drawdown)
```

```
Volatility: 0.021135399087142986
Beta: 1.1896770019660008
Sharpe Ratio: 0.0561412252535486
VaR at 95% Confidence Level: -0.03240541240696362
Alpha: 0.0006095819016819145
Treynor Ratio: -0.01581389971222014
Maximum Drawdown: 0.3142726401615783
```

Financial Risk Analysis

Volatility (0.0211): Indicates the degree of variation of a trading price series. A lower volatility suggests a more stable investment.

• Beta (1.1897): Reflects the stock's sensitivity to market movements. A beta above 1 suggests the stock is more volatile than the market.

Sharpe Ratio (0.0561): Measures the risk-adjusted return. A positive Sharpe ratio indicates a potentially favorable risk- return profile.

• VaR at 95% Confidence Level (-0.0324): Represents the maximum expected loss with a 95% confidence. A negative value suggests a potential loss, emphasizing risk.

Alpha (0.0006): Indicates the excess return over the benchmark. Positive alpha implies the investment outperforms expectations.

• Treynor Ratio (-0.0158): Measures the excess return per unit of systematic risk. A negative value may suggest an underperformance compared to the market.

Maximum Drawdown (0.3143): Represents the largest peak-to-trough decline in the investment's value. A lower drawdown is generally preferred.

Advice for the Investor

Considering the positive Sharpe ratio and alpha, the investment shows potential for positive risk-adjusted returns. However,the high volatility, beta, and negative Treynor ratio indicate higher risk and sensitivity to market movements. Investors should carefully assess their risk tolerance and consider diversification strategies. It's crucial to monitor market conditions and stay informed about company developments. Consulting with a financial advisor is recommended for a more personalized assessment based on the company's financial goals and risk tolerance.

One may ask: are there no threshold to consider for these parameters? My response would be: No and Yes!

• It is NO because the thresholds are functions of several factors such as your risk tolerance, investment goals, and market conditions. What I consider to be highly risky might be tolerable for you!

• It is yes because when you compare with other companies portfolio you might be able to draw boundaries

Summary

This project fetches historical stock prices for Apple (AAPL) and the S&P 500 (^GSPC) using yfinance, calculates daily returns, and then computes volatility, beta, and the Sharpe ratio etc. Finally, it visualizes the daily returns and cumulative returns

In [ ]:

```

```