Problem 1

For the first part, I firstly define two variables using the datetime module. The first variable is called "current_date" and it represents the current date which I set as March 3rd, 2023. The second variable is called "expiration_date" and it represents the date when something is going to expire, which I set as March 17th, 2023. Next, I calculate the difference between the two dates by subtracting the "current_date" from the "expiration_date". Then, I use the ".days" attribute to extract the number of days between the two dates from the time delta object. This gives me the number of calendar days between the current date and the expiration date. After that, I calculate the "time to maturity" (TTM) by dividing the number of days between the two dates by 365, which is the number of days in a year. This gives me the time to maturity in years.

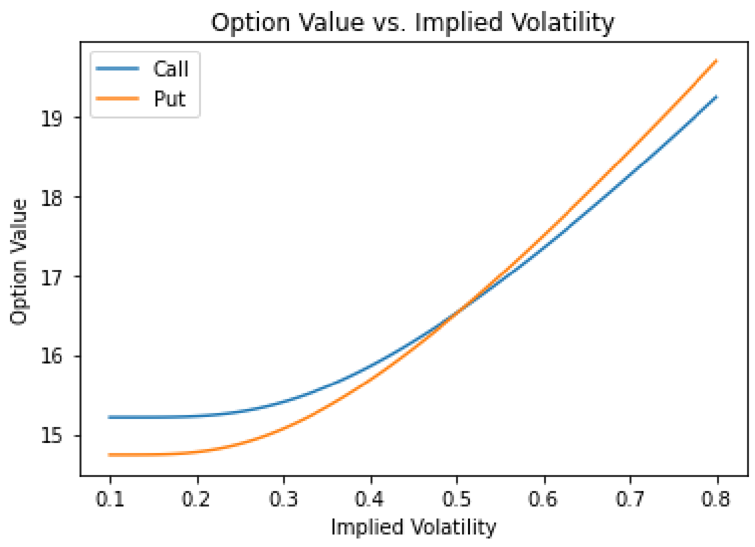
- Time to maturity (calendar days): 0.038356164383561646

For the second part, First, I define several parameters that are needed for calculating the call and put values of an option. S represents the underlying asset price, PK is the strike price of a put option, CK is the strike price of a call option, r is the risk-free interest rate, T is the time to maturity of the option, and q is the dividend yield.

Next, I define a range of implied volatilities that I will use to calculate the call and put values. Implied volatility is the estimated volatility of the underlying asset that is implied by the market price of an option.

Then, I define two functions for calculating the call and put values of an option using the Black-Scholes-Merton (BSM) model. To calculate the call value, I first calculate the two parameters d1 and d2 using the input parameters and the formula for the BSM model. I then use these parameters to calculate the call value using the formula for the BSM call option price.To calculate the put value, I follow a similar process but use a different formula for the BSM put option price.

Finally, I calculate the call and put values for each implied volatility in the range defined earlier using the previously defined functions and store these values in the variables call_values and put_values, respectively. This allows me to see how the call and put values change as the implied volatility of the underlying asset changes.



The graphs show the relationship between the option value and the implied volatility of a call and a put option, as calculated using the Black-Scholes-Merton (BSM) model. As the implied volatility increases, the option value also increases, indicating that higher volatility leads to higher option prices.

The supply and demand of an option can affect its implied volatility. When there is high demand for an option, its price increases, indicating that market participants are willing to pay more to acquire the option. This can be because they believe that the underlying asset's price will move significantly in the future, and the option provides them with a way to profit from this expected move.

Problem 2

First, I have defined some variables:

- The current AAPL stock price, which is 151.03.
- Today's date, which is March 3, 2023.
- The risk-free rate, which is 0.0425.
- The continuous coupon, which is 0.0053.

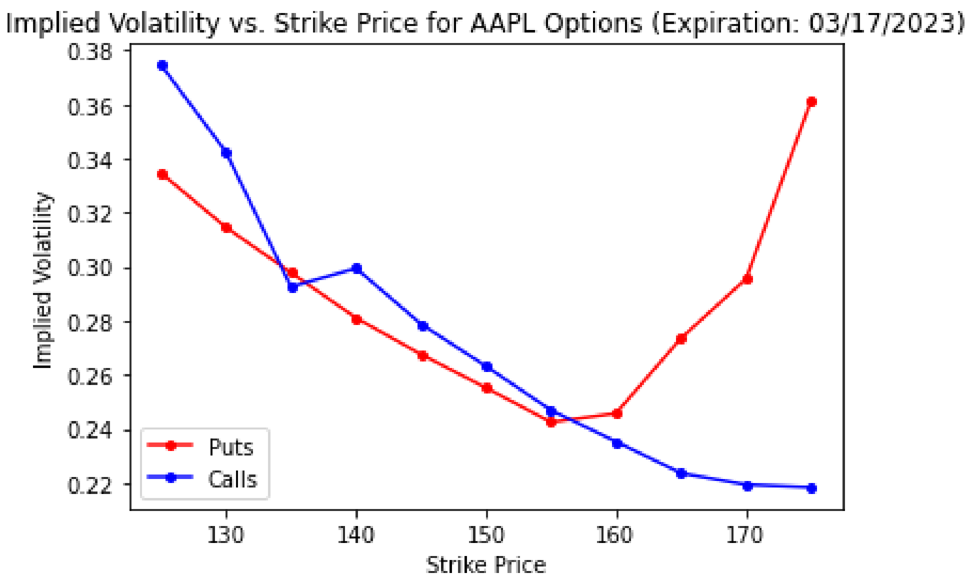
Next, I have defined a function called "black_scholes" that uses the Black-Scholes formula for European options. This function takes in the stock price (S), strike price (K), time to expiration (T), risk-free rate (r), continuous dividend yield (q), implied volatility (sigma), and option type (Call or Put).

Inside this function, I have calculated two values, d1 and d2, using the given formula. Then, depending on the option type, I have calculated the option price using the appropriate formula. Finally, the function returns the calculated option price.

I have also defined a second function called "implied_volatility" which takes in the same parameters as the "black_scholes" function, except for market price, which is the observed market price of the option.

Inside this function, I have defined an objective function which calculates the difference between the calculated option price using the "black_scholes" function and the observed market price. The function then uses the Brent method to find the implied volatility which makes the difference between the calculated option price and the observed market price equal to zero.

Lastly, I have a code block that uses the "implied_volatility" function to calculate the implied volatility for each option. It first converts the expiration dates in a pandas dataframe to datetime format, then calculates the time to expiration in years (T). Finally, it applies the "implied_volatility" function to each row of the dataframe using the "apply" method and stores the results in a new column called "Implied_Volatility".



The graphs depict the relationship between strike price and implied volatility for both put and call options. As strike price increases, the implied volatility for call options generally decreases in a linear fashion, while the implied volatility for put options first decreases and then increases.

The shape of these graphs can be attributed to several market dynamics. Firstly, the linear decrease in implied volatility for call options with increasing strike price can be explained by the fact that as the strike price becomes more and more out-of-the-money, the likelihood of the option being exercised decreases, leading to a decrease in implied volatility.

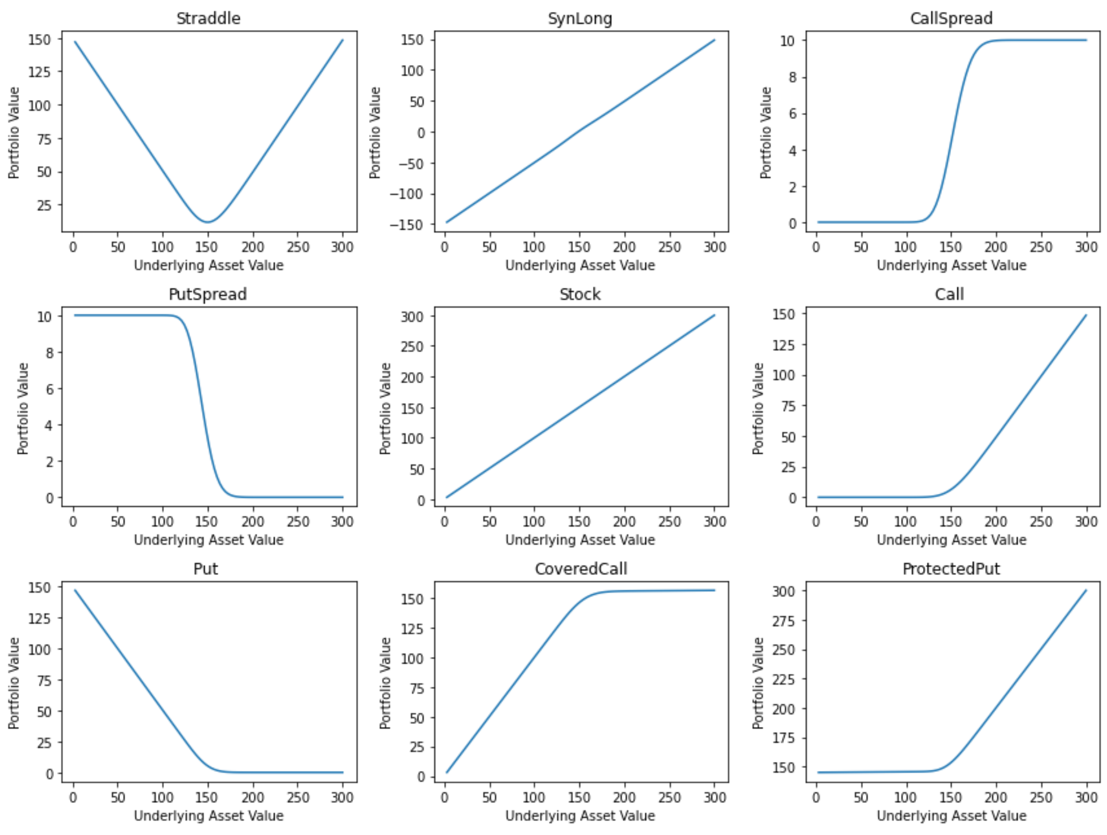
On the other hand, the U-shaped curve for put options can be explained by the presence of factors such as hedging demand and market sentiment. As the strike price becomes increasingly out-of-the-money, the hedging demand for put options increases, leading to an increase in implied volatility.

As the strike price becomes very high, the likelihood of the option being exercised increases, leading to an increase in implied volatility. This is because if the stock price were to fall below the higher strike price, the put option would become more valuable.

Problem 3

First, I calculate the time to expiration (T) for each option in the "port" DataFrame based on the difference between the option's expiration date and the current date. The code also calculates the implied volatility for each option using the Black-Scholes formula and a function named "implied_volatility" which takes in several parameters including the stock price, strike price, risk-free rate, and dividend.

Next, I define a function named "portfolio_value" that takes in several parameters including the "port" DataFrame, a range of underlying stock prices, the risk-free rate, the dividend, the current date, and an optional parameter named "d_ahead". This function calculates the total value of the portfolio for each underlying stock price in the range by iterating through each row in the "port" DataFrame and adding up the value of each stock holding and option holding using the Black-Scholes formula and the previously calculated implied volatility for each option. The function then returns a list of these portfolio values.



1. Straddle: A straddle involves buying both a call option and a put option with the same strike price and expiration date. The graph of a straddle will typically look like a "V" shape, with the maximum profit occurring at the strike price of the options. As the underlying asset price moves away from the strike price, the profit potential of the straddle decreases.
2. Synthetic Long: A synthetic long involves buying a call option and selling a put option with the same strike price and expiration date. The graph of a synthetic long will typically look like an upward sloping line, with the maximum profit occurring as the underlying asset price increases.
3. Call Spread: A call spread involves buying a call option at a lower strike price and selling a call option at a higher strike price with the same expiration date. The graph of a call spread will typically look like a "V" shape, with the maximum profit occurring at the lower strike price of the long call option.
4. Put Spread: A put spread involves buying a put option at a higher strike price and selling a put option at a lower strike price with the same expiration date. The graph of a put spread will typically look like a "V" shape, with the maximum profit occurring at the higher strike price of the long put option.
5. Stock: The graph of a stock will typically look like an upward sloping line, with the profit potential increasing as the stock price increases.
6. Call: The graph of a call option will typically look like an upward sloping line, with the maximum profit occurring as the underlying asset price increases above the strike price of the call option.
7. Put: The graph of a put option will typically look like a downward sloping line, with the maximum profit occurring as the underlying asset price decreases below the strike price of the put option.
8. Covered Call: A covered call involves buying a stock and selling a call option with a strike price above the stock price and the same expiration date. The graph of a covered call will typically look like an upward sloping line up to the strike price of the call option, with the maximum profit occurring at the strike price of the call option.
9. Protected Put: A protected put involves buying a put option and buying enough stock to cover the put option's strike price. The graph of a protected put will typically look like an upward sloping line up to the strike price of the put option, with the maximum profit occurring at the strike price of the put option.

First, I calculate the log returns of the AAPL stock and store it in the variable aapl_returns. This is done by taking the natural logarithm of the current AAPL stock price and subtracting the natural logarithm of the previous AAPL stock price.

Next, I demean the series so that the mean of the aapl_returns series is 0. This is done by subtracting the mean of the aapl_returns series from itself.

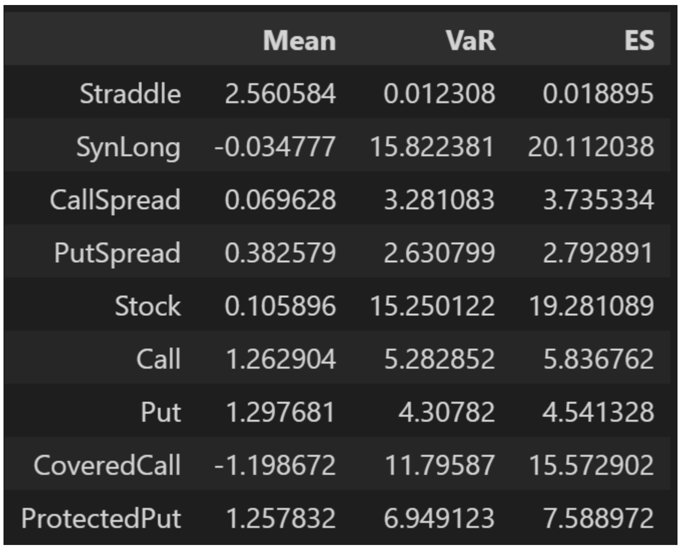
Then I fit an AR(1) model to the aapl_returns series using the ARIMA() function from the statsmodels library. The parameters of the AR(1) model are stored in the variable ar_result. Next, I set the variables n_forecast, sim, phi, beta, and sigma. n_forecast is the number of days to simulate returns for, sim is the number of simulations to run, phi is the estimated coefficient for the AR(1) model, beta is the constant term in the AR(1) model, and sigma is the standard deviation of the residuals of the AR(1) model.

I then generate simulated returns using the AR(1) model by looping through each simulation and each day of the simulation. We start by generating the return for day 0 using the formula beta + phi * aapl_returns.iloc[-1] + sigma * norm.rvs(), where aapl_returns.iloc[-1] is the last value in the aapl_returns series. We then use this return value to generate the return for day 1, and so on, until we have generated returns for n_forecast days. We store the simulated returns in the variable simulated_returns.

I then set the initial price of the AAPL stock to initial_price. Using the simulated returns, I calculate the price series for the AAPL stock by taking the cumulative sum of the simulated returns, exponentiating the result, and multiplying it by the initial price. The resulting price series is stored in the variable price_series.

I extract the final simulated prices from price_series and store them in the variable simulation_price. I create an empty DataFrame results_df to store the results of our calculations. I iterate through each portfolio in the portfolios list and for each portfolio, I calculate the mean, value at risk (VaR), and expected shortfall (ES) of the difference between the simulated and current portfolio values.

Lastly I calculate the difference between the simulated and current portfolio values and store it in the variable diff_portfolio_values. I calculate the mean, VaR, and ES of diff_portfolio_values using the mean(), calculate_var(), and calculate_es() functions from my own library.



	Mean	VaR	ES
Straddle	2.560584	0.012308	0.018895
SynLong	-0.034777	15.822381	20.112038
CallSpread	0.069628	3.281083	3.735334
PutSpread	0.382579	2.630799	2.792891
Stock	0.105896	15.250122	19.281089
Call	1.262904	5.282852	5.836762
Put	1.297681	4.30782	4.541328
CoveredCall	-1.198672	11.79587	15.572902
ProtectedPut	1.257832	6.949123	7.588972

Looking at the Mean column, it appears that most of the investment strategies have a positive average return, although some are higher than others. The Call and Put options have the highest average returns, followed by the Straddle and ProtectedPut strategies.

However, when we look at the VaR and ES columns, we see that the strategies with the highest average returns also tend to have the highest levels of risk. For example, the Call and Put options have relatively high VaR and ES values, indicating that there is a higher likelihood of significant losses with these strategies. On the other hand, the PutSpread and CallSpread strategies have lower VaR and ES values, indicating that they may be less risky overall.