Estimation of Google Stock using Black Scholes Model

In [1]:

```
import pandas as pd
import statsmodels.api as sm
import datetime as dt
import matplotlib.pyplot as plt
import numpy as np
from statsmodels.tsa.stattools import adfuller
from scipy import stats
from math import sqrt
```

In [2]:

```
import yfinance as yf
import getFamaFrenchFactors as gff
```

In [6]:

```
# importing the data from Yahoo finance
ticker = 'goog'
start = dt.datetime(2005,1,1)
end = dt.datetime.now()
data = yf.download(ticker, start, end)
```

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

In [7]:

```
data.describe()

# gives an overview of the dataset and indicates if there are any missing/null values
```

Out[7]:

	Open	High	Low	Close	Adj Close	Volume
count	4575.000000	4575.000000	4575.000000	4575.000000	4575.000000	4.575000e+03
mean	38.865478	39.269179	38.465985	38.874216	38.874216	1.177857e+08
std	35.509470	35.909196	35.129875	35.522241	35.522241	1.463376e+08
min	4.366135	4.443345	4.298140	4.358414	4.358414	1.584340e+05
25%	12.940372	13.074493	12.814468	12.939250	12.939250	2.938800e+07
50%	26.189596	26.419464	25.953745	26.230984	26.230984	6.466548e+07
75%	54.964249	55.523251	54.357500	54.905249	54.905249	1.459530e+08
max	151.863495	152.100006	149.887497	150.709000	150.709000	1.650833e+09

In [9]:

```
# calculating and plotting the returns
# we consider the 'Adj Close' which is the Adjusted Closing Price for deriving returns

returns = data['Adj Close'].resample('M').last().pct_change().dropna() # we calculate mo
returns.head()
```

Out[9]:

```
Date
2005-02-28 00:00:00-05:00   -0.039004
2005-03-31 00:00:00-05:00   -0.039789
2005-04-30 00:00:00-04:00    0.218769
2005-05-31 00:00:00-04:00    0.260318
2005-06-30 00:00:00-04:00    0.060879
Freq: M, Name: Adj Close, dtype: float64
```

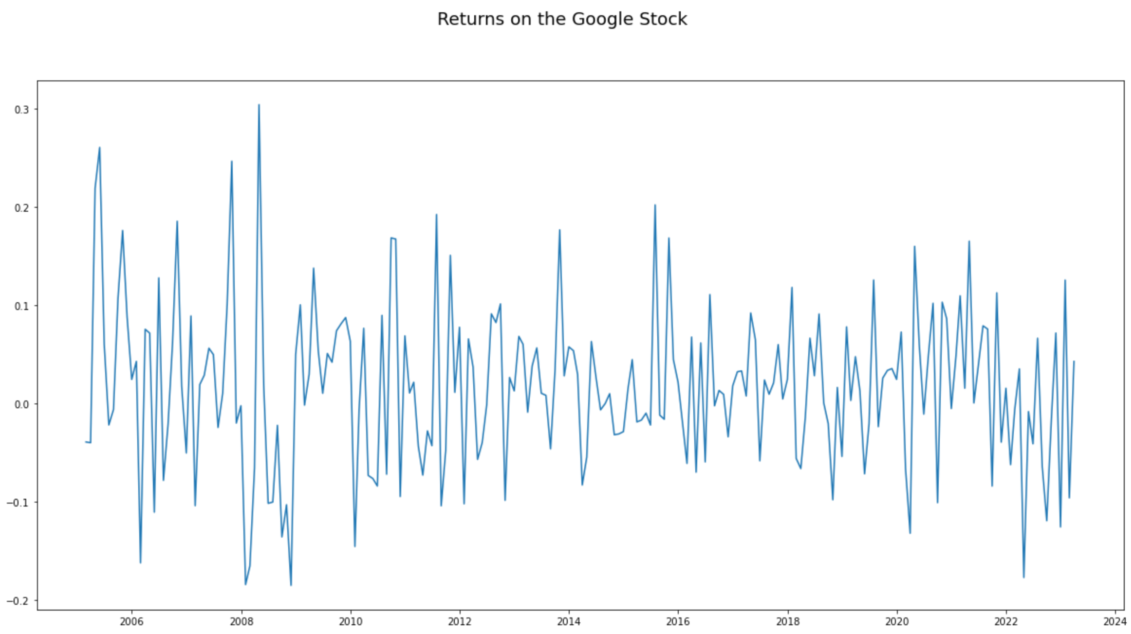
In [15]:

```
plt.figure()
fig1, axs = plt.subplots(figsize=(20, 10))
axs.plot(returns)
fig1.suptitle('Returns on the Google Stock',fontsize=18)
```

Out[15]:

```
Text(0.5, 0.98, 'Returns on the Google Stock')
```

```
<Figure size 432x288 with 0 Axes>
```



Black Scholes Model

In [16]:

```
from math import log,e
```

In [17]:

```
# define a function with the following parameters:
# stock_price: spot price of underlying assets
# strike price
# interest: risk free interest rate
# time: time to expiry
# volatility
# dividend: we will assume no dividends

def bsm(stock_price, strike_price, interest, time, volatility, dividend):


# defining parameters for the formula:
    d1 = (log(stock_price/strike_price) + (interest - dividend + volatility**2/2) * time
    d2 = d1 - volatility * time**.5

# usin g the inbuilt scipy stats model to call out the cdf of the d1 and d2
    call = stats.norm.cdf(d1) * stock_price*e**(-dividend*time) - stats.norm.cdf(d2)*str
    put = stats.norm.cdf(-d2)*strike_price*e**(-interest * time) - stats.norm.cdf(-d1) *

    return[call ,put]
```

In [18]:

```
volatility = np.sqrt(252) * returns.std()
volatility
```

Out[18]:

```
1.3087181645496653
```

In [19]:

```
# defining time variable

from datetime import datetime
expiry = '12-31-2023'   # assuming expiry date of 31 December 2023
time = (datetime.strptime(expiry, "%m-%d-%Y") - datetime.utcnow()).days / 365
time
```

Out[19]:

```
0.8136986301369863
```

In [20]:

```
interest = 0.0475  # assuming risk free rate as the US 3 month T-bill rate which is 4.75
strike_price = 85 # using the latest price of the call option
dividend = 0
```

In [21]:

```
price = data['Adj Close']
```

In [29]:

```
for x in price:
    print(bsm(x,strike_price, interest, time, volatility, dividend))

for x in price:
    data['Call'] = bsm(x,strike_price, interest, time, volatility, dividend)[0]
```

```
[0.05583048612332306, 77.00504870955075]
[0.0569040160482618, 76.97598517740293]
[0.05555745593769454, 77.01249662663074]
[0.057189998760883126, 76.96830130385335]
[0.05714526606402355, 76.96950159297656]
[0.061388674524474635, 76.85817779410058]
[0.0651279729086027, 76.76403434483456]
[0.05892432568838532, 76.92221458585898]
[0.05589222974983009, 77.00336761336402]
[0.051048394233585453, 77.13899666679431]
[0.04497181290657866, 77.32121401933571]
[0.04224293415760688, 77.40814912297807]
[0.05185397291881472, 77.11589219985576]
[0.05088154294896091, 77.14381085646427]
[0.052786553819296256, 77.08942714815737]
[0.05740508447754297, 76.9625382821176]
[0.05412655280372697, 77.05191302497992]
[0.06714168159264233, 76.71474085213309]
[0.07208127761207928, 76.5976377922077]
[0.06557442348895345, 76.75302382901477]
```

In [38]:

```
for x in price:
    data['Put'] = bsm(x,strike_price, interest, time, volatility, dividend)[1]
```

In [37]:

```
data.head()
```

Out[37]:

Date	Open	High	Low	Close	Adj Close	Volume	Call	Pu
2005-01-03 00:00:00-05:00	4.916571	5.071989	4.868253	5.048826	5.048826	636143518	45.591354	33.19872
2005-01-04 00:00:00-05:00	5.016198	5.054305	4.818937	4.844342	4.844342	552298420	45.591354	33.19872
2005-01-05 00:00:00-05:00	4.818190	4.904118	4.787804	4.819685	4.819685	330698912	45.591354	33.19872
2005-01-06 00:00:00-05:00	4.858788	4.879212	4.675475	4.696148	4.696148	417041336	45.591354	33.19872
2005-01-07 00:00:00-05:00	4.748203	4.838116	4.701876	4.828153	4.828153	387964757	45.591354	33.19872

In [39]:

```
# plotting call and put options

plt.plot(price,[bsm(x,strike_price, interest, time, volatility, dividend)[0] for x in pr
plt.plot(price,[bsm(y,strike_price, interest, time, volatility, dividend)[1] for y in pr
plt.xlabel('Google Price')
plt.ylabel('BSM Euro Call and Put Value')
plt.grid()
```