In [415…
```
import pandas as pd
from datetime import datetime
import statsmodels.api as sm
import matplotlib.pyplot as plt
import numpy as np
```

In [416…
```
df= pd.read_csv('/Users/liyuetong/Desktop/Derivative Assignments3  (Feb 25th due)/spx_quotedata-2024-02-08
r1=df.iloc[1,:]
r2=df.iloc[2,:]
df.columns=df.iloc[1]
df=df[2:]
df['Expiration Date']=pd.to_datetime(Puts['Expiration Date'])
```

In [417…
```
col_index = df.columns.get_loc('Puts')
col_index
```

Out[417]: 12

In [418…
```
Calls=df.iloc[:, :12]
```

In [419…
```
Puts=df.iloc[:, 12:]
Puts['Expiration Date']=df['Expiration Date']
```

In [420…
```
current_bid=4950.1299
current_ask=5045.9302
date='2/8/2024 16:00'
SP500=[4997.9102,2.8502]
```

Q3.(a)

In [421…
```
expiry='19-Jul-2024'
current_date=datetime(2024,2,8)
expiry_date=datetime.strptime(expiry,"%d-%b-%Y")
date_to_expiry=(expiry_date-current_date).days
print("date_to_expiry:",date_to_expiry)
year_to_expiry=(expiry_date-current_date).days/365
print("year_to_expiry:",year_to_expiry)
```

```
date_to_expiry: 162
year_to_expiry: 0.4438356164383562
```

Q3(b)

In [499…
```
df1=pd.DataFrame()
df1['Expiration Date']= df['Expiration Date']
df1['C_mid']=(Calls['Bid'].astype(float)+Calls['Ask'].astype(float))/2
df1['P_mid']=(Puts['Bid'].astype(float)+Puts['Ask'].astype(float))/2
df1['K']=Calls['Strike'].astype(float)
```

In [500…
```
df1['RT']=df1['K']*1
df1['FT']=1
```

In [501…
```
df1 = df1[df1['Expiration Date'] == '2024-07-19']
```

In [502…
```
df1
```

Out[502]:

	Expiration Date	C_mid	P_mid	K	RT	FT
1710	2024-07-19	4776.250	0.050	200.0	200.0	1
1711	2024-07-19	4581.050	0.050	400.0	400.0	1
1712	2024-07-19	4385.650	0.075	600.0	600.0	1
1713	2024-07-19	4190.500	0.100	800.0	800.0	1
1714	2024-07-19	3995.150	0.225	1000.0	1000.0	1
...	...	...	...	...	...	...
1949	2024-07-19	0.150	1669.700	6800.0	6800.0	1
1950	2024-07-19	0.100	1865.100	7000.0	7000.0	1
1951	2024-07-19	0.075	2060.150	7200.0	7200.0	1
1952	2024-07-19	0.075	2255.500	7400.0	7400.0	1
1953	2024-07-19	0.050	2451.100	7600.0	7600.0	1

244 rows × 6 columns

In [503…
```
import statsmodels.formula.api as smf
X=df1[['FT','RT']]
y=df1['C_mid']-df1['P_mid']
X=sm.add_constant(X)
reg1 = sm.OLS(y, X).fit()
print(reg1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 1.296e+10
Date:                Fri, 23 Feb 2024   Prob (F-statistic):               0.00
Time:                        19:54:29   Log-Likelihood:                111.67
No. Observations:                 244   AIC:                            -219.3
Df Residuals:                     242   BIC:                            -212.4
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
FT          4971.6249      0.039   1.28e+05      0.000    4971.548    4971.701
RT            -0.9767   8.58e-06  -1.14e+05      0.000      -0.977      -0.977
==============================================================================
Omnibus:                       60.162   Durbin-Watson:                   0.829
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               11.782
Skew:                           0.069   Prob(JB):                      0.00276
Kurtosis:                       1.932   Cond. No.                     1.78e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.78e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [504…
```
Estimate=reg1.params
Estimate[1]=-Estimate[1]
Estimate[0]=Estimate[0]/Estimate[1]
```

In [505…
```
standard_error=reg1.bse
t_stat=reg1.tvalues
p_value=reg1.pvalues
```

In [506…
```
reg1_stats=pd.DataFrame({'Estimate':Estimate,'Standard Error':standard_error,'t-Statistic':t_stat,'P-Valu
adj_rsquared = reg1.rsquared_adj
reg1_stats.loc['Adj. R-squared'] = [adj_rsquared, None, None, None]
reg1_stats
```

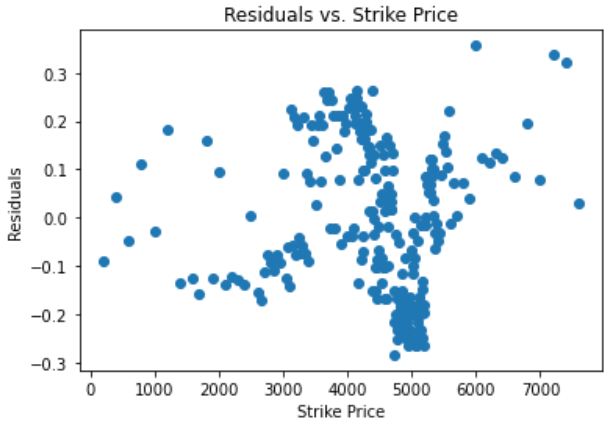
Out[506]:

	Estimate	Standard Error	t-Statistic	P-Value
FT	5090.373577	0.038798	128140.968210	0.0
RT	0.976672	0.000009	-113822.371875	0.0
Adj. R-squared	1.000000	NaN	NaN	NaN

(ii)

In [442…
```
df1['FT']=5090.373577
df1['RT']=0.976672
df1['predicted']=df1['RT']*(df1['FT']-df1['K'])
df1['residuals']=(df1['C_mid']-df1['P_mid'])-df1['predicted']
```

In [443…
```
residuals=reg1.resid
plt.scatter(df1['K'],df1['residuals'])
plt.xlabel('Strike Price')
plt.ylabel('Residuals')
plt.title('Residuals vs. Strike Price')
plt.show()
```



As the residual is inside the range(-0.3,0.3) and does not show any systematic pattern change related to different value of K, we can conclude that Heteroscedastity is not concern for this data base on the observation.

(iii)

In [444…
```
T=year_to_expiry
FT=5090.373577
RT=0.976672
r_imp=-np.log(RT)/T
print('r_imp:',r_imp)
y_imp=r_imp-np.log(FT/SP500[0])/T
print('y_imp:',y_imp)
```

```
r_imp: 0.053182764133731394
y_imp: 0.011880632379671435
```

In [445…
```
import numpy as np
from scipy.stats import norm
from scipy.optimize import newton
```

In [446…
```
def find_sigma_imp(initial_guess,S,K,r,y,T,price,option='call/put'):
    def function_imp(sigma):
        d1=(np.log(S/K)+(r-y+(sigma**2)/2)*T)/(sigma*np.sqrt(T))
        d2=d1-sigma*np.sqrt(T)
        if option=='call':
            return S*np.exp(-y*T)*norm.cdf(d1)-K*np.exp(-r*T)*norm.cdf(d2)-price
        else:
            return K*np.exp(-r*T)*norm.cdf(-d2)-S*np.exp(-y*T)*norm.cdf(-d1)-price
    return newton(function_imp,initial_guess)
```

In [447…
```
initial_guess=0.155
S=SP500[0]
K=S
r=r_imp
y=y_imp

#For call
market_price=df1[df1['K'] == 5000]['C_mid'].iloc[0]
sigma_atm_imp_call=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='call')
sigma_atm_imp_call
print("sigma_atm_imp_call:",sigma_atm_imp_call)
total_variance_call =sigma_atm_imp_call**2*T
print("total_variance_call:",total_variance_call)
#For put
market_price=df1[df1['K'] == 5000]['P_mid'].iloc[0]
sigma_atm_imp_put=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='put')
print("sigma_atm_imp_put:",sigma_atm_imp_put)
total_variance_put =sigma_atm_imp_put**2*T
print("total_variance_put:",total_variance_put)
```

```
sigma_atm_imp_call: 0.12862287987969884
total_variance_call: 0.007342747745273109
sigma_atm_imp_put: 0.13026996443884817
total_variance_put: 0.0075320074215167124
```

In [448…
```
Calls= Calls[Calls['Expiration Date'] == '2024-07-19']
Puts=Puts[Puts['Expiration Date'] == '2024-07-19']
Puts['Strike']=Calls['Strike']
```

In [449…
```
market_price=Calls[Calls['Strike'].astype(float)==5000]['Ask'].astype(float).iloc[0]
type(market_price)
```

Out[449]:
```
numpy.float64
```

In [450…
```
#Call_ask
market_price=Calls[Calls['Strike'].astype(float)==5000]['Ask'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_call_ask=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='call')
sigma_atm_imp_call_ask
print("sigma_atm_imp_call_ask:",sigma_atm_imp_call_ask)
print()

#Call_bid
market_price=Calls[Calls['Strike'].astype(float)==5000]['Bid'].astype(float).iloc[0].astype(float)
print(market_price)
sigma_atm_imp_call_bid=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='call')
sigma_atm_imp_call_bid
print("sigma_atm_imp_call_bid:",sigma_atm_imp_call_bid)
print()

uncertainty_call=sigma_atm_imp_call_ask-sigma_atm_imp_call_bid
print('uncertainty_call:',uncertainty_call)
print()
#Put_ask
market_price=Puts[Puts['Strike'].astype(float)==5000]['Ask'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_put_ask=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='put')
sigma_atm_imp_put_ask
print("sigma_atm_imp_put_ask:",sigma_atm_imp_put_ask)
print()

#Put_bid
market_price=Puts[Puts['Strike'].astype(float)==5000]['Bid'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_put_bid=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='put')
sigma_atm_imp_put_bid
print("sigma_atm_imp_put_bid:",sigma_atm_imp_put_bid)
print()
uncertainty_put=sigma_atm_imp_put_ask-sigma_atm_imp_put_bid
print('uncertainty_put:',uncertainty_put)
```

```
218.1
sigma_atm_imp_call_ask: 0.129209443781648

216.6
sigma_atm_imp_call_bid: 0.12803619844453154

uncertainty_call: 0.001173245337116463

129.8
sigma_atm_imp_put_ask: 0.13077804811652008

128.5
sigma_atm_imp_put_bid: 0.12976179606644436

uncertainty_put: 0.0010162520500757166
```

3.(c)

In [451…
```
lower_bound = S * (1 - 0.2 * np.sqrt(T))
upper_bound = S * (1 + 0.2 * np.sqrt(T))
```

In [452…
```
df1=pd.DataFrame()
df1['Expiration Date']= df['Expiration Date']
df1['C_mid']=(Calls['Bid'].astype(float)+Calls['Ask'].astype(float))/2
df1['P_mid']=(Puts['Bid'].astype(float)+Puts['Ask'].astype(float))/2
df1['K']=Calls['Strike'].astype(float)
df1 = df1[df1['Expiration Date'] == '2024-07-19']
df1 = df1[(df1['K'] >= lower_bound) & (df1['K'] <= upper_bound)]
df1['RT']=df1['K']*1
df1['FT']=1
```

In [453…
```
import statsmodels.formula.api as smf
X=df1[['FT','RT']]
y=df1['C_mid']-df1['P_mid']
X=sm.add_constant(X)
reg1 = sm.OLS(y, X).fit()
print(reg1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 7.711e+08
Date:                Fri, 23 Feb 2024   Prob (F-statistic):               0.00
Time:                        19:21:11   Log-Likelihood:                 77.377
No. Observations:                 134   AIC:                            -150.8
Df Residuals:                     132   BIC:                            -145.0
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
FT          4971.4916      0.173   2.88e+04      0.000    4971.150    4971.834
RT            -0.9767   3.52e-05  -2.78e+04      0.000      -0.977      -0.977
==============================================================================
Omnibus:                       33.790   Durbin-Watson:                   0.761
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                7.779
Skew:                           0.208   Prob(JB):                       0.0205
Kurtosis:                       1.896   Cond. No.                     7.19e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.19e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [454…
```
Estimate=reg1.params
Estimate[1]=-Estimate[1]
Estimate[0]=Estimate[0]/Estimate[1]
standard_error=reg1.bse
t_stat=reg1.tvalues
p_value=reg1.pvalues
reg1_stats=pd.DataFrame({'Estimate':Estimate,'Standard Error':standard_error,'t-Statistic':t_stat,'P-Valu
adj_rsquared = reg1.rsquared_adj
```

```
reg1_stats.loc['Adj. R-squared'] = [adj_rsquared, None, None, None]
reg1_stats
```

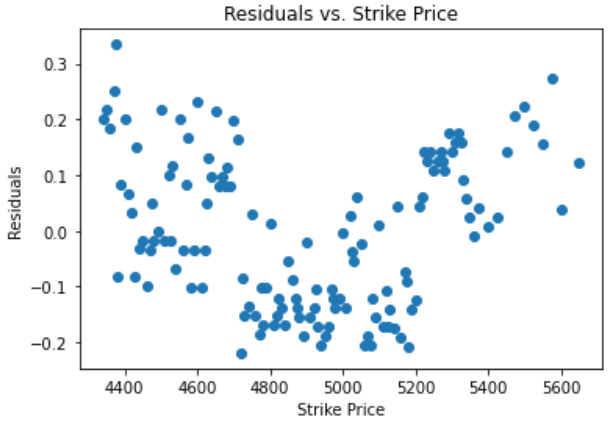
Out[454]:

	Estimate	Standard Error	t-Statistic	P-Value
FT	5090.313024	0.172872	28758.189378	0.0
RT	0.976657	0.000035	-27768.032661	0.0
Adj. R-squared	1.000000	NaN	NaN	NaN

(c)(ii)

In [455…

```
df1['FT']=5090.313024
df1['RT']=0.976657
df1['predicted']=df1['RT']*(df1['FT']-df1['K'])
df1['residuals']=(df1['C_mid']-df1['P_mid'])-df1['predicted']
plt.scatter(df1['K'],df1['residuals'])
plt.xlabel('Strike Price')
plt.ylabel('Residuals')
plt.title('Residuals vs. Strike Price')
plt.show()
```



The residual is inside the range(-0.25,0.3). There is some little systematic pattern change related to different value of K appeared. When option out of the money and in the money, the residual is higher than when the option at the money, or in other word, residual decrease when the Gap between S and K decrease. However, as the outlier is not very far away from the main datapoint, we can conclude that Heteroscedastity is still not a main concern.

(c)(iii)

In [456…

```
T=year_to_expiry
FT=5090.373577
RT=0.976672
r_imp=-np.log(RT)/T
print('r_imp:',r_imp)
y_imp=-np.log(FT*RT/SP500[0])/T
print('y_imp:',y_imp)
```

```
r_imp: 0.053182764133731394
y_imp: 0.011880632379671461
```

(c)(iv)

In [457…

```
def find_sigma_imp(initial_guess,S,K,r,q,T,price,option='call/put'):
    def function_imp(sigma):
        d1=(np.log(S/K)+(r-q+(sigma**2)/2)*T)/(sigma*np.sqrt(T))
        d2=d1-sigma*np.sqrt(T)
        if option=='call':
            return S*np.exp(-q*T)*norm.cdf(d1)-K*np.exp(-r*T)*norm.cdf(d2)-price
        else:
            return K*np.exp(-r*T)*norm.cdf(-d2)-S*np.exp(-q*T)*norm.cdf(-d1)-price
    return newton(function_imp,initial_guess)
```

In [458…
```
initial_guess=0.155
S=SP500[0]
K=S
r=r_imp
q=y_imp

#For call
market_price=df1[df1['K'] == 5000]['C_mid'].iloc[0]
sigma_atm_imp_call=find_sigma_imp(initial_guess,S,K,r,q,T,market_price,option='call')
sigma_atm_imp_call
print("sigma_atm_imp_call:",sigma_atm_imp_call)
total_variance_call =sigma_atm_imp_call**2*T
print("total_variance_call:",total_variance_call)
print()
#For put
market_price=df1[df1['K'] == 5000]['P_mid'].iloc[0]
sigma_atm_imp_put=find_sigma_imp(initial_guess,S,K,r,q,T,market_price,option='put')
print("sigma_atm_imp_put:",sigma_atm_imp_put)
total_variance_put =sigma_atm_imp_put**2*T
print("total_variance_put:",total_variance_put)
```

```
sigma_atm_imp_call: 0.12862287987969853
total_variance_call: 0.0073427477452730735

sigma_atm_imp_put: 0.1302699644388483
total_variance_put: 0.007532007421516729
```

In [459…
```
Calls= Calls[Calls['Expiration Date'] == '2024-07-19']
Puts=Puts[Puts['Expiration Date'] == '2024-07-19']
```

(c)(v)

In [460…
```
#Call_ask
market_price=Calls[Calls['Strike'].astype(float)==5000]['Ask'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_call_ask=find_sigma_imp(initial_guess,S,K,r,q,T,market_price,option='call')
sigma_atm_imp_call_ask
print("sigma_atm_imp_call_ask:",sigma_atm_imp_call_ask)
print()

#Call_bid
market_price=Calls[Calls['Strike'].astype(float)==5000]['Bid'].astype(float).iloc[0].astype(float)
print(market_price)
sigma_atm_imp_call_bid=find_sigma_imp(initial_guess,S,K,r,q,T,market_price,option='call')
sigma_atm_imp_call_bid
print("sigma_atm_imp_call_bid:",sigma_atm_imp_call_bid)
print()

uncertainty_call=sigma_atm_imp_call_ask-sigma_atm_imp_call_bid
print('uncertainty_call:',uncertainty_call)
print()
#Put_ask
market_price=Puts[Puts['Strike'].astype(float)==5000]['Ask'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_put_ask=find_sigma_imp(initial_guess,S,K,r,q,T,market_price,option='call')
sigma_atm_imp_put_ask
print("sigma_atm_imp_put_ask:",sigma_atm_imp_put_ask)
print()

#Put_bid
market_price=Puts[Puts['Strike'].astype(float)==5000]['Bid'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_put_bid=find_sigma_imp(initial_guess,S,K,r,q,T,market_price,option='call')
sigma_atm_imp_put_bid
print("sigma_atm_imp_put_bid:",sigma_atm_imp_put_bid)
print()
uncertainty_put=sigma_atm_imp_put_ask-sigma_atm_imp_put_bid
print('uncertainty_put:',uncertainty_put)
```

```
218.1
sigma_atm_imp_call_ask: 0.1292094437816476

216.6
sigma_atm_imp_call_bid: 0.12803619844453107

uncertainty_call: 0.0011732453371165186

129.8
sigma_atm_imp_put_ask: 0.05827472334944418

128.5
sigma_atm_imp_put_bid: 0.057162122114188876

uncertainty_put: 0.001112601235255306
```

(e)

Repeat part(b)

In [461…
```
df1=pd.DataFrame()
df1['Expiration Date']= df['Expiration Date']
df1['C_mid']=(Calls['Bid'].astype(float)+Calls['Ask'].astype(float))/2
df1['P_mid']=(Puts['Bid'].astype(float)+Puts['Ask'].astype(float))/2
df1['K']=Calls['Strike'].astype(float)
df1['RT']=df1['K']*1
df1['FT']=1
df1 = df1[df1['Expiration Date'] == '2024-07-19']
```

In [462…
```
df1['Call_variance'] = (Calls['Ask'].astype(float) - Calls['Bid'].astype(float))**2
df1['Put_variance'] = (Puts['Ask'].astype(float) - Puts['Bid'].astype(float))**2
```

In [463…
```
import statsmodels.formula.api as smf
X=df1[['FT','RT']]
y=df1['C_mid']-df1['P_mid']
weights =1/(df1['Call_variance']+df1['Put_variance'])
reg1 = sm.WLS(y, X, weights=weights).fit()
print(reg1.summary())
```

```
                            WLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            WLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 8.513e+09
Date:                Fri, 23 Feb 2024   Prob (F-statistic):               0.00
Time:                        19:21:44   Log-Likelihood:                 87.375
No. Observations:                 244   AIC:                            -170.7
Df Residuals:                     242   BIC:                            -163.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
FT          4971.6882      0.051   9.74e+04      0.000    4971.588    4971.789
RT            -0.9767   1.06e-05  -9.23e+04      0.000      -0.977      -0.977
==============================================================================
Omnibus:                       13.382   Durbin-Watson:                   0.749
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               14.433
Skew:                          -0.596   Prob(JB):                     0.000734
Kurtosis:                       3.030   Cond. No.                     2.81e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.81e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [464…
```
Estimate=reg1.params
Estimate[1]=-Estimate[1]
Estimate[0]=Estimate[0]/Estimate[1]
standard_error=reg1.bse
t_stat=reg1.tvalues
p_value=reg1.pvalues
```

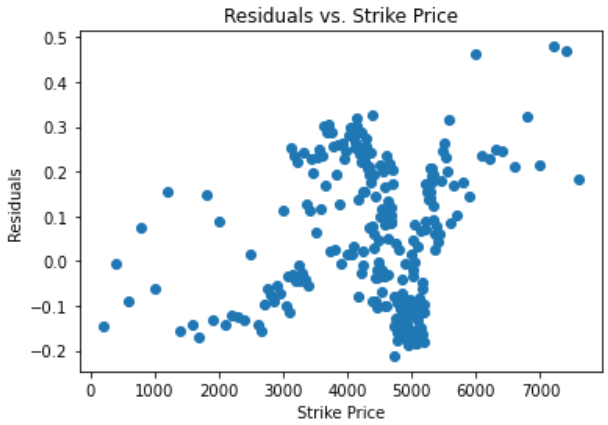
```
reg1_stats=pd.DataFrame({'Estimate':Estimate,'Standard Error':standard_error,'t-Statistic':t_stat,'P-Valu
adj_rsquared = reg1.rsquared_adj
reg1_stats.loc['Adj. R-squared'] = [adj_rsquared, None, None, None]
reg1_stats
```

Out[464]:

	Estimate	Standard Error	t-Statistic	P-Value
FT	5090.289538	0.051031	97425.797724	0.0
RT	0.976700	0.000011	-92264.202988	0.0
Adj. R-squared	1.000000	NaN	NaN	NaN

In [465…

```
df1['FT']=5090.289538
df1['RT']=0.976700
df1['predicted']=df1['RT']*(df1['FT']-df1['K'])
df1['residuals']=(df1['C_mid']-df1['P_mid'])-df1['predicted']
plt.scatter(df1['K'],df1['residuals'])
plt.xlabel('Strike Price')
plt.ylabel('Residuals')
plt.title('Residuals vs. Strike Price')
plt.show()
```



In [493…

```
T=year_to_expiry
FT=5090.289538
RT=0.976700
r_imp=-np.log(RT)/T
print('r_imp:',r_imp)
y_imp=-np.log(FT*RT/SP500[0])/T
print('y_imp:',y_imp)
```

```
r_imp: 0.05311817180850356
y_imp: 0.011853237460446822
```

In [467…

```
initial_guess=0.155
S=SP500[0]
K=S
r=r_imp
y=y_imp

#For call
market_price=df1[df1['K'] == 5000]['C_mid'].iloc[0]
sigma_atm_imp_call=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='call')
sigma_atm_imp_call
print("sigma_atm_imp_call:",sigma_atm_imp_call)
total_variance_call =sigma_atm_imp_call**2*T
print("total_variance_call:",total_variance_call)
print()
#For put
market_price=df1[df1['K'] == 5000]['P_mid'].iloc[0]
sigma_atm_imp_put=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='put')
print("sigma_atm_imp_put:",sigma_atm_imp_put)
total_variance_put =sigma_atm_imp_put**2*T
print("total_variance_put:",total_variance_put)
```

```
sigma_atm_imp_call: 0.12865660673299062
total_variance_call: 0.007346599007889535

sigma_atm_imp_put: 0.13024143459122026
total_variance_put: 0.0075287086796422325
```

In [468…
```
Calls= Calls[Calls['Expiration Date'] == '2024-07-19']
Puts=Puts[Puts['Expiration Date'] == '2024-07-19']
```

In [469…
```
#Call_ask
market_price=Calls[Calls['Strike'].astype(float)==5000]['Ask'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_call_ask=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='call')
sigma_atm_imp_call_ask
print("sigma_atm_imp_call_ask:",sigma_atm_imp_call_ask)
print()

#Call_bid
market_price=Calls[Calls['Strike'].astype(float)==5000]['Bid'].astype(float).iloc[0].astype(float)
print(market_price)
sigma_atm_imp_call_bid=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='call')
sigma_atm_imp_call_bid
print("sigma_atm_imp_call_bid:",sigma_atm_imp_call_bid)
print()

uncertainty_call=sigma_atm_imp_call_ask-sigma_atm_imp_call_bid
print('uncertainty_call:',uncertainty_call)
print()
#Put_ask
market_price=Puts[Puts['Strike'].astype(float)==5000]['Ask'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_put_ask=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='call')
sigma_atm_imp_put_ask
print("sigma_atm_imp_put_ask:",sigma_atm_imp_put_ask)
print()

#Put_bid
market_price=Puts[Puts['Strike'].astype(float)==5000]['Bid'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_put_bid=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='call')
sigma_atm_imp_put_bid
print("sigma_atm_imp_put_bid:",sigma_atm_imp_put_bid)
print()
uncertainty_put=sigma_atm_imp_put_ask-sigma_atm_imp_put_bid
print('uncertainty_put:',uncertainty_put)
```

```
218.1
sigma_atm_imp_call_ask: 0.1292431279186575

216.6
sigma_atm_imp_call_bid: 0.12806996834908052

uncertainty_call: 0.0011731595695769759

129.8
sigma_atm_imp_put_ask: 0.05831978755805728

128.5
sigma_atm_imp_put_bid: 0.05720763412006343

uncertainty_put: 0.0011121534379938472
```

Repeat part(c)

In [470…
```
lower_bound = S * (1 - 0.2 * np.sqrt(T))
upper_bound = S * (1 + 0.2 * np.sqrt(T))
df1=pd.DataFrame()
df1['Expiration Date']= df['Expiration Date']
df1['C_mid']=(Calls['Bid'].astype(float)+Calls['Ask'].astype(float))/2
df1['P_mid']=(Puts['Bid'].astype(float)+Puts['Ask'].astype(float))/2
df1['K']=Calls['Strike'].astype(float)
df1 = df1[df1['Expiration Date'] == '2024-07-19']
df1 = df1[(df1['K'] >= lower_bound) & (df1['K'] <= upper_bound)]
df1['RT']=df1['K']*1
df1['FT']=1
```

In [471…
```
df1['Call_variance'] = (Calls['Ask'].astype(float) - Calls['Bid'].astype(float))**2
df1['Put_variance'] = (Puts['Ask'].astype(float)- Puts['Bid'].astype(float))**2
```

In [472…
```
import statsmodels.formula.api as smf
X=df1[['FT','RT']]
y=df1['C_mid']-df1['P_mid']
weights =1/(df1['Call_variance']+df1['Put_variance'])
reg1 = sm.WLS(y, X, weights=weights).fit()
print(reg1.summary())
```

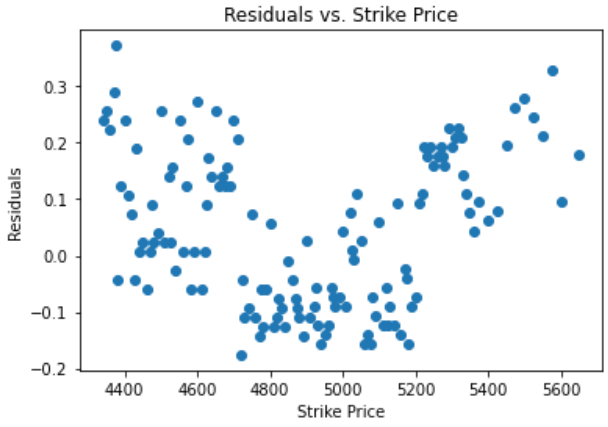
```
                            WLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       1.000
Model:                            WLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 4.938e+08
Date:                Fri, 23 Feb 2024   Prob (F-statistic):               0.00
Time:                        19:22:34   Log-Likelihood:                 71.134
No. Observations:                 134   AIC:                            -138.3
Df Residuals:                     132   BIC:                            -132.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
FT          4971.5180      0.220   2.26e+04      0.000    4971.083    4971.953
RT            -0.9767   4.39e-05  -2.22e+04      0.000      -0.977      -0.977
==============================================================================
Omnibus:                        6.971   Durbin-Watson:                   0.955
Prob(Omnibus):                  0.031   Jarque-Bera (JB):                6.364
Skew:                          -0.465   Prob(JB):                       0.0415
Kurtosis:                       2.474   Cond. No.                     1.12e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.12e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [473…
```
Estimate=reg1.params
Estimate[1]=-Estimate[1]
Estimate[0]=Estimate[0]/Estimate[1]
standard_error=reg1.bse
t_stat=reg1.tvalues
p_value=reg1.pvalues
reg1_stats=pd.DataFrame({'Estimate':Estimate,'Standard Error':standard_error,'t-Statistic':t_stat,'P-Valu
adj_rsquared = reg1.rsquared_adj
reg1_stats.loc['Adj. R-squared'] = [adj_rsquared, None, None, None]
reg1_stats
```

Out[473]:

	Estimate	Standard Error	t-Statistic	P-Value
FT	5090.263323	0.219723	22626.256699	0.0
RT	0.976672	0.000044	-22222.597151	0.0
Adj. R-squared	1.000000	NaN	NaN	NaN

In [474…
```
df1['FT']=5090.263323
df1['RT']=0.976672
df1['predicted']=df1['RT']*(df1['FT']-df1['K'])
df1['residuals']=(df1['C_mid']-df1['P_mid'])-df1['predicted']
plt.scatter(df1['K'],df1['residuals'])
plt.xlabel('Strike Price')
plt.ylabel('Residuals')
plt.title('Residuals vs. Strike Price')
plt.show()
```



(e)(vii).

In [475…
```
T=year_to_expiry
FT=5090.263323
RT=0.976672
r_imp=-np.log(RT)/T
print('r_imp:',r_imp)
y_imp=-np.log(FT*RT/SP500[0])/T
print('y_imp:',y_imp)
```

```
r_imp: 0.053182764133731394
y_imp: 0.011929433214545345
```

In [477…
```
initial_guess=0.155
S=SP500[0]
K=S
r=r_imp
y=y_imp

#For call
market_price=df1[df1['K'] == 5000]['C_mid'].iloc[0]
sigma_atm_imp_call=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='call')
sigma_atm_imp_call
print("sigma_atm_imp_call:",sigma_atm_imp_call)
total_variance_call =sigma_atm_imp_call**2*T
print("total_variance_call:",total_variance_call)
print()
#For put
market_price=df1[df1['K'] == 5000]['P_mid'].iloc[0]
sigma_atm_imp_put=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='put')
print("sigma_atm_imp_put:",sigma_atm_imp_put)
total_variance_put =sigma_atm_imp_put**2*T
print("total_variance_put:",total_variance_put)
```

```
sigma_atm_imp_call: 0.12867351946912944
total_variance_call: 0.007348530649963656

sigma_atm_imp_put: 0.13023633146459163
total_variance_put: 0.007528118710694538
```

In [478…
```
Calls= Calls[Calls['Expiration Date'] == '2024-07-19']
Puts=Puts[Puts['Expiration Date'] == '2024-07-19']
```

In [479…
```
#Call_ask
market_price=Calls[Calls['Strike'].astype(float)==5000]['Ask'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_call_ask=find_sigma_imp(initial_guess,S,K,r,q,T,market_price,option='call')
sigma_atm_imp_call_ask
print("sigma_atm_imp_call_ask:",sigma_atm_imp_call_ask)
print()

#Call_bid
market_price=Calls[Calls['Strike'].astype(float)==5000]['Bid'].astype(float).iloc[0].astype(float)
print(market_price)
sigma_atm_imp_call_bid=find_sigma_imp(initial_guess,S,K,r,q,T,market_price,option='call')
sigma_atm_imp_call_bid
print("sigma_atm_imp_call_bid:",sigma_atm_imp_call_bid)
```

```
print()

uncertainty_call=sigma_atm_imp_call_ask-sigma_atm_imp_call_bid
print('uncertainty_call:',uncertainty_call)
print()
#Put_ask
market_price=Puts[Puts['Strike'].astype(float)==5000]['Ask'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_put_ask=find_sigma_imp(initial_guess,S,K,r,q,T,market_price,option='call')
sigma_atm_imp_put_ask
print("sigma_atm_imp_put_ask:",sigma_atm_imp_put_ask)
print()

#Put_bid
market_price=Puts[Puts['Strike'].astype(float)==5000]['Bid'].astype(float).iloc[0]
print(market_price)
sigma_atm_imp_put_bid=find_sigma_imp(initial_guess,S,K,r,q,T,market_price,option='call')
sigma_atm_imp_put_bid
print("sigma_atm_imp_put_bid:",sigma_atm_imp_put_bid)
print()
uncertainty_put=sigma_atm_imp_put_ask-sigma_atm_imp_put_bid
print('uncertainty_put:',uncertainty_put)
```

```
218.1
sigma_atm_imp_call_ask: 0.1292600481321665

216.6
sigma_atm_imp_call_bid: 0.12808687372231398

uncertainty_call: 0.001173174409852512

129.8
sigma_atm_imp_put_ask: 0.05833800460313318

128.5
sigma_atm_imp_put_bid: 0.05722596674029918

uncertainty_put: 0.001112037862834002
```

In [480…
```
df1=pd.DataFrame()
df1['Expiration Date']= df['Expiration Date']
df1['C_mid']=(Calls['Bid'].astype(float)+Calls['Ask'].astype(float))/2
df1['P_mid']=(Puts['Bid'].astype(float)+Puts['Ask'].astype(float))/2
df1['K']=Calls['Strike'].astype(float)
df1 = df1[df1['Expiration Date'] == '2024-07-19']
df1['Call_variance'] = (Calls['Ask'].astype(float) - Calls['Bid'].astype(float))**2
df1['Put_variance'] = (Puts['Ask'].astype(float) - Puts['Bid'].astype(float))**2
weights =1/(df1['Call_variance']+df1['Put_variance'])
df1['RT']=0.976700
df1['FT']=5090.289538
```

In [481…
```
adjusted_mid_call=(df1['RT']*((df1['FT']-df1['K'])+df1['P_mid'])*df1['Call_variance']+df1['C_mid']*df1['P
adjusted_mid_put=((-df1['RT'])*((df1['FT']-df1['K'])+df1['C_mid'])*df1['Put_variance']+df1['P_mid']*df1['
```

In [482…
```
df1['diff_put']=adjusted_mid_put-df1['P_mid']
df1['diff_call']=adjusted_mid_call-df1['C_mid']
```

In [483…
```
spread_call=Calls['Ask'].astype(float)-Calls['Bid'].astype(float)
spread_call
spread_put=Puts['Ask'].astype(float)-Puts['Bid'].astype(float)
```

In [484…
```
df1['predicted']=df1['RT']*(df1['FT']-df1['K'])
df1['residuals']=(df1['C_mid']-df1['P_mid'])-df1['predicted']
```

In [496…
```
df1[df1['residuals']>spread_call]
```

Out[496]:

	Expiration Date	C_mid	P_mid	K	Call_variance	Put_variance	RT	FT	diff_put	diff_call	predict
1943	2024-07-19	2.100	890.15	6000.0	0.0900	72.25	0.9767	5090.289538	-3.682275	-0.026381	-888.5142
1947	2024-07-19	0.500	1279.45	6400.0	0.0400	75.69	0.9767	5090.289538	-0.743749	-0.015875	-1279.1942
1949	2024-07-19	0.150	1669.70	6800.0	0.0400	70.56	0.9767	5090.289538	0.027688	-0.022226	-1669.8742
1950	2024-07-19	0.100	1865.10	7000.0	0.0400	81.00	0.9767	5090.289538	0.016530	-0.021555	-1865.2142
1951	2024-07-19	0.075	2060.15	7200.0	0.0225	72.25	0.9767	5090.289538	0.330853	-0.015093	-2060.5542
1952	2024-07-19	0.075	2255.50	7400.0	0.0225	73.96	0.9767	5090.289538	0.320858	-0.016125	-2255.8942
1953	2024-07-19	0.050	2451.10	7600.0	0.0100	77.44	0.9767	5090.289538	0.085362	-0.007398	-2451.2342

In [495…
```
df1[df1['residuals']>spread_put]
```

Out[495]:

Expiration Date	C_mid	P_mid	K	Call_variance	Put_variance	RT	FT	diff_put	diff_call	predicted	residuals

In [497…
```
df1[df1['diff_call']>spread_call]
```

Out[497]:

Expiration Date	C_mid	P_mid	K	Call_variance	Put_variance	RT	FT	diff_put	diff_call	predicted	residuals

In [498…
```
df1[df1['diff_put']>spread_put]
```

Out[498]:

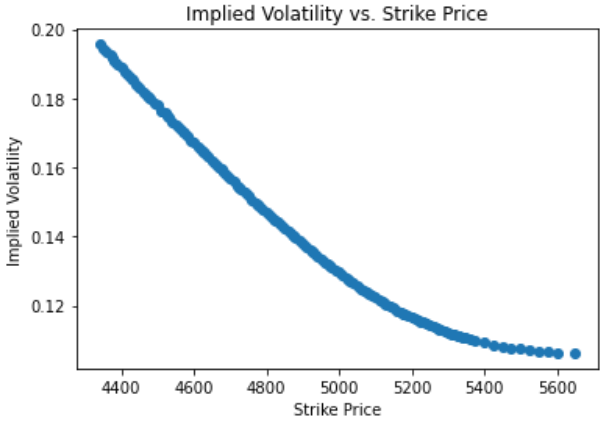
Expiration Date	C_mid	P_mid	K	Call_variance	Put_variance	RT	FT	diff_put	diff_call	predicted	residuals

Extra Credit

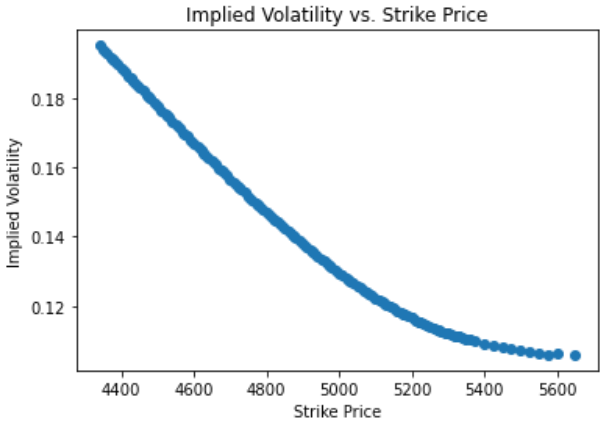
In [489…
```
imp_vol_call=pd.DataFrame()
imp_vol_call['imp_vol']=df1['K']

imp_vol_put=pd.DataFrame()
imp_vol_put['imp_vol']=df1['K']
```

In [ ]:
```
initial_guess=0.155
S=SP500[0]
r=r_imp
y=y_imp
def find_sigma_imp(initial_guess, S, K, r, y, T, price, option):
    def function_imp(sigma):
        d1 = (np.log(S / K) + (r - y + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
        d2 = d1 - sigma * np.sqrt(T)
        if option == 'call':
            return S * np.exp(-y * T) * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2) - price
        else:
            return K * np.exp(-r * T) * norm.cdf(-d2) - S * np.exp(-y * T) * norm.cdf(-d1) - price

    return newton(function_imp, initial_guess)

for i in range(0,len(imp_vol_call)):
    market_price=df1.iloc[i]['C_mid']
    K=df1.iloc[i]['K']
    sigma_imp_call=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='call')
    imp_vol_call['imp_vol'].iloc[i]=sigma_imp_call

for i in range(0,len(imp_vol_put)):
    market_price=df1.iloc[i]['P_mid']
    K=df1.iloc[i]['K']
    sigma_imp_put=find_sigma_imp(initial_guess,S,K,r,y,T,market_price,option='put')
    imp_vol_put['imp_vol'].iloc[i]=sigma_imp_put
```

In [304…
```
plt.scatter(df1['K'],imp_vol_call)
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.title('Implied Volatility vs. Strike Price')
plt.show()
```



In [305…
```
plt.scatter(df1['K'],imp_vol_put)
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.title('Implied Volatility vs. Strike Price')
plt.show()
```



In [ ]: