```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt


def densityFunction(p, q, t):
    return (np.exp((p+q)*t)*p*(p+q)**2)/((p*np.exp((p+q)*t)+q)**2)


data = pd.read_csv('iphone_sales.txt', delim_whitespace=True)



def convert_quarter_to_date(quarter):
    q, y = quarter.split('_')
    q = int(q[1])
    y = int('20' + y)
    month = 3 * q
    if month > 12:
        month -= 12
        y += 1
    return pd.Timestamp(year=y, month=month, day=1)


data['date'] = data['Quarter'].apply(convert_quarter_to_date)



date = data['date']


sales = data['Sales_MM_units']



cum_sales = np.cumsum(sales)



cum_sales_squared = cum_sales**2


mod = smf.ols(formula='sales ~ cum_sales + cum_sales_squared', data=data)


res = mod.fit()


print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.873
Model:                            OLS   Adj. R-squared:                  0.865
Method:                 Least Squares   F-statistic:                     109.9
Date:                Sun, 07 Jul 2024   Prob (F-statistic):           4.61e-15
Time:                        19:54:01   Log-Likelihood:                -120.18
No. Observations:                  35   AIC:                             246.4
Df Residuals:                      32   BIC:                             251.0
Df Model:                           2
Covariance Type:            nonrobust
=====================================================================================
```

```
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept             3.6963      2.205      1.676      0.103      -0.795       8.188
cum_sales             0.1130      0.017      6.737      0.000       0.079       0.147
cum_sales_squared -5.508e-05   2.11e-05     -2.610      0.014   -9.81e-05   -1.21e-05
==============================================================================
Omnibus:                        3.898   Durbin-Watson:                   1.824
Prob(Omnibus):                  0.142   Jarque-Bera (JB):                2.617
Skew:                           0.627   Prob(JB):                        0.270
Kurtosis:                       3.470   Cond. No.                     4.32e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.32e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
b = res.params
b
```

```
Intercept            3.696307
cum_sales            0.112994
cum_sales_squared   -0.000055
dtype: float64
```

```
m1 = (-b['cum_sales']+np.sqrt(b['cum_sales']**2 - 4*b['Intercept']*b['cum_sales_squared']))/(2*b['
m1
```

```
-32.206910098759515
```

```
m2 = (-b['cum_sales']-np.sqrt(b['cum_sales']**2 - 4*b['Intercept']*b['cum_sales_squared']))/(2*b['
m2
```

```
2083.8220174932867
```

```
m = max(m1, m2)
print(m)
```

```
2083.8220174932867
```

```
p = b['Intercept']/m
q = -m*b['cum_sales_squared']
print("p = ",p)
print("q = ", q)
```

```
p =  0.0017738112418998347
q =  0.11476751136366699
```

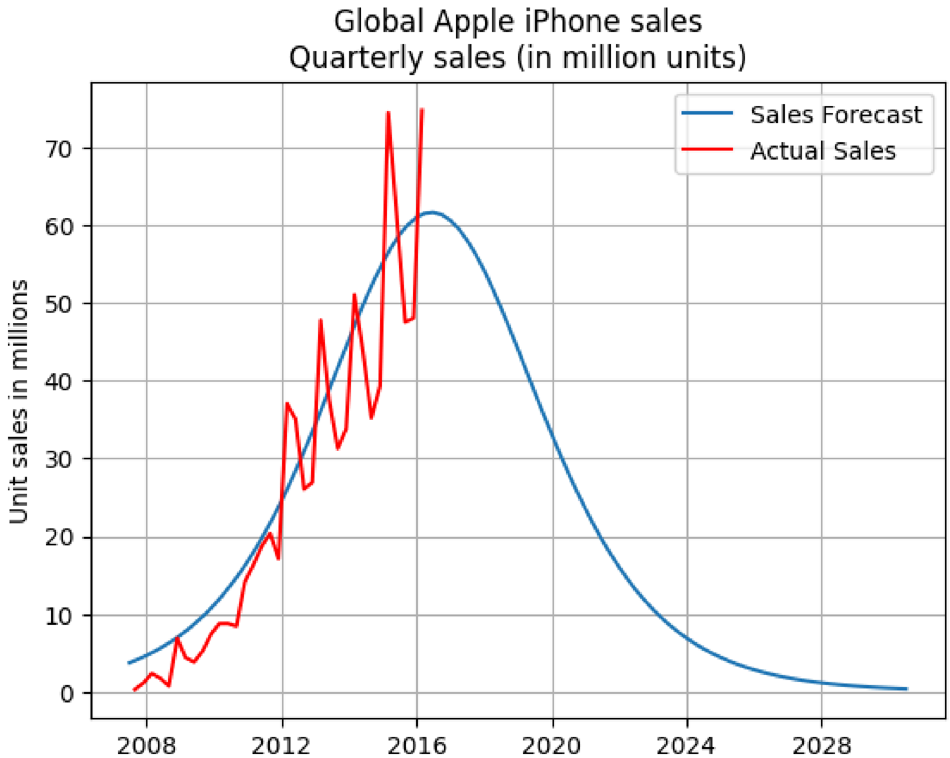
```
date_forecast = pd.date_range(start='2007-Q3', end='2030-Q3', freq='QS')
```

```
t = range(len(date_forecast))
```

```
f = densityFunction(p,q,t)
sales_forecast = m*f
```

```
fig, ax = plt.subplots()
ax.plot(date, sales, color='red')
plt.ylabel('Unit sales in millions')
#plt.xlabel('time')
plt.title("Global Apple iPhone sales\nfrom 3rd quarter 2007 to 1st quarter 2016 \n(in million unit
#plt.legend()
ax.grid(True)
plt.show()
```


```
fig, ax = plt.subplots()
ax.plot(date_forecast, sales_forecast, label='Sales Forecast')
ax.plot(date, sales, color='red', label='Actual Sales')
plt.ylabel('Unit sales in millions')
#plt.xlabel('time')
plt.title("Global Apple iPhone sales\nQuarterly sales (in million units)")
plt.legend()
ax.grid(True)
plt.show()
```




```
sales_apple = sales
date_apple = date
date_forecast_apple = date_forecast
sales_forecast_apple = sales_forecast
```

PEAK

```
peak_time = -1/(p+q)*np.log(p/q)
peak_time
```

```
35.77939457746893
```

```
print(data['Sales_MM_units'].idxmax() + 1)
```

```
35
```

Samsung

```
data2 = pd.read_csv('galaxy_sales.csv')
```

```
data2.head()
```

	Quarter	sales
0	Q1 '10	64.90
1	Q2 '10	65.33
2	Q3 '10	71.67
3	Q4 '10	79.17
4	Q1 '11	68.78

Pasos siguientes: Generar código con data2 Ver gráficos recomendados

```
quarter = data2['Quarter']
qs = quarter.str.replace(r"(Q\d) '(\d+)", r'20\2-\1', regex=True)  # e.g., "Q1 '10" to "2010-Q1"
qs = qs.str.replace('*', '', regex=False)  # Remove any asterisks if present

# Convert the cleaned string to a PeriodIndex and then to timestamps
data2['date'] = pd.PeriodIndex(qs.values, freq='Q').to_timestamp()



date = data2['date']



# Get coefficients
sales = data2['sales']


cum_sales = np.cumsum(sales)
cum_sales_squared = cum_sales**2


mod = smf.ols(formula='sales ~ cum_sales + cum_sales_squared', data=data2)
res = mod.fit()
print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.821
Model:                            OLS   Adj. R-squared:                  0.802
Method:                 Least Squares   F-statistic:                     43.44
Date:                Sun, 07 Jul 2024   Prob (F-statistic):           8.17e-08
Time:                        19:54:02   Log-Likelihood:                -73.418
No. Observations:                  22   AIC:                             152.8
Df Residuals:                      19   BIC:                             156.1
Df Model:                           2
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept            53.7484      4.506     11.928      0.000      44.317      63.180
cum_sales             0.0766      0.011      7.173      0.000       0.054       0.099
cum_sales_squared -2.806e-05   5.07e-06     -5.530      0.000   -3.87e-05   -1.74e-05
==============================================================================
Omnibus:                        1.888   Durbin-Watson:                   1.207
Prob(Omnibus):                  0.389   Jarque-Bera (JB):                1.056
Skew:                           0.116   Prob(JB):                        0.590
Kurtosis:                       1.952   Cond. No.                     5.30e+06
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.3e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
b = res.params
b
```

```
Intercept            53.748387
cum_sales             0.076600
cum_sales_squared    -0.000028
dtype: float64
```

```
m1 = (-b['cum_sales']+np.sqrt(b['cum_sales']**2 - 4*b['Intercept']*b['cum_sales_squared']))/(2*b['
m2 = (-b['cum_sales']-np.sqrt(b['cum_sales']**2 - 4*b['Intercept']*b['cum_sales_squared']))/(2*b['
```

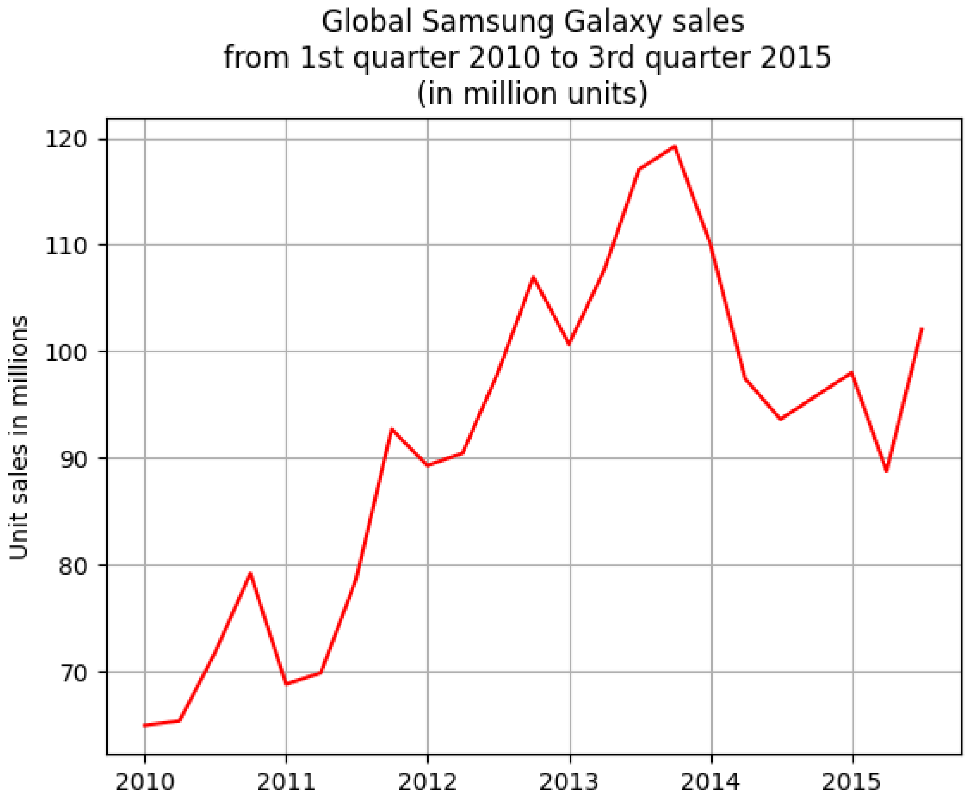
```
m = max(m1, m2)
m
```

```
3308.96521821821
```

```
p = b['Intercept']/m
q = -m*b['cum_sales_squared']
print("p = ",p)
print("q = ", q)
```

```
p =  0.016243261464994826
q =  0.09284320017915797
```

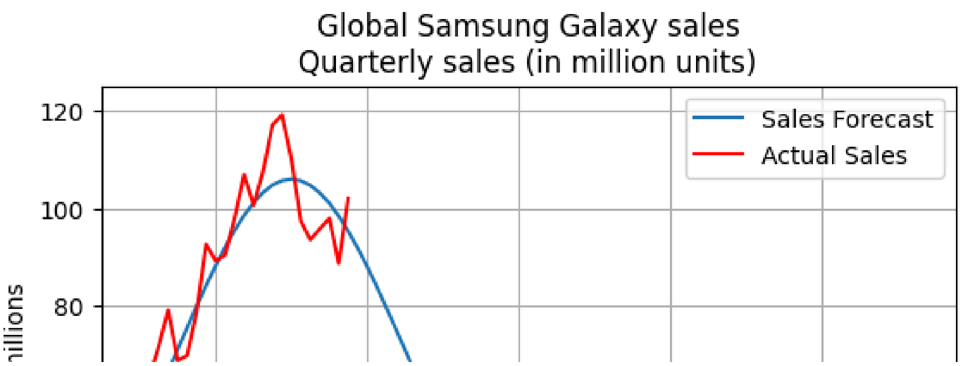
```
fig, ax = plt.subplots()
ax.plot(date, sales, color='red')
plt.ylabel('Unit sales in millions')
#plt.xlabel('time')
plt.title("Global Samsung Galaxy sales\nfrom 1st quarter 2010 to 3rd quarter 2015 \n(in million ur
#plt.legend()
ax.grid(True)
plt.show()
```




```
date_forecast = pd.date_range(start='2010-Q1', end='2030-Q3', freq='QS')
```

```
t = range(len(date_forecast))
f = densityFunction(p,q,t)
sales_forecast = m*f
```

```
fig, ax = plt.subplots()
ax.plot(date_forecast, sales_forecast, label='Sales Forecast')
ax.plot(date, sales, color='red', label='Actual Sales')
plt.ylabel('Unit sales in millions')
#plt.xlabel('time')
plt.title("Global Samsung Galaxy sales\nQuarterly sales (in million units)")
plt.legend()
ax.grid(True)
plt.show()
```




```
sales_samsung = sales
date_samsung = date
date_forecast_samsung = date_forecast
sales_forecast_samsung = sales_forecast
```

Sales peak

```
peak_time = -1/(p+q)*np.log(p/q)
peak_time
```

```
15.98029563590453
```

```
print(data2['sales'].idxmax() + 1)
```

```
16
```

Comparasion

```
fig, ax = plt.subplots()
ax.plot(date_forecast_samsung, sales_forecast_samsung, label='Sales Forecast Samsung')
ax.plot(date_samsung, sales_samsung, color='green', label='Actual Sales Samsung')
ax.plot(date_forecast_apple, sales_forecast_apple, label='Sales Forecast Apple')
ax.plot(date_apple, sales_apple, color='red', label='Actual Sales Apple')
plt.ylabel('Unit sales in millions')
#plt.xlabel('time')
plt.title("Global Apple iPhone and Samsung Galaxy sales\nQuarterly sales (in million units)")
plt.legend()
ax.grid(True)
plt.show()
```