time-analysis-finance

September 23, 2023

[64]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
```

1 Data Cleaning

[65]:
```
datasets = ['AAPL']


for dataset in datasets :
    Ticker = yf.Ticker(dataset)
    data = Ticker.history(start='2023-08-16' , end='2023-09-18')
    filename = f"{dataset}_data.csv"
    data.to_csv(filename)
    print(f"Download data for {dataset} and saved as {filename}")
```

```
Download data for AAPL and saved as AAPL_data.csv
```

[66]:
```
Ticker = 'AAPL'
start_date = '2019-01-01'
end_date = '2023-09-23'
```

[67]:
```
stock_data = yf.download(Ticker, start=start_date, end=end_date)
```

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

[68]:
```
stock_data
```

[68]:

Date	Open	High	Low	Close	Adj Close
2019-01-02	38.722500	39.712502	38.557499	39.480000	37.943256
2019-01-03	35.994999	36.430000	35.500000	35.547501	34.163834
2019-01-04	36.132500	37.137501	35.950001	37.064999	35.622253
2019-01-07	37.174999	37.207500	36.474998	36.982498	35.542969
2019-01-08	37.389999	37.955002	37.130001	37.687500	36.220531
…	…	…	…	…	…

```
2023-09-18  176.479996  179.380005  176.169998  177.970001  177.970001
2023-09-19  177.520004  179.630005  177.130005  179.070007  179.070007
2023-09-20  179.259995  179.699997  175.399994  175.490005  175.490005
2023-09-21  174.550003  176.300003  173.860001  173.929993  173.929993
2023-09-22  174.669998  177.078995  174.054993  174.789993  174.789993

               Volume
Date
2019-01-02  148158800
2019-01-03  365248800
2019-01-04  234428400
2019-01-07  219111200
2019-01-08  164101200
…                 …
2023-09-18   67257600
2023-09-19   51826900
2023-09-20   58436200
2023-09-21   63047900
2023-09-22   55110610

[1190 rows x 6 columns]
```

[69]:
```
# Data Cleaning
# Remove duplicate row if any

stock_data = stock_data.drop_duplicates()
```

[70]:
```
stock_data
```

[70]:
```
                 Open        High         Low       Close   Adj Close  \
Date
2019-01-02   38.722500   39.712502   38.557499   39.480000   37.943256
2019-01-03   35.994999   36.430000   35.500000   35.547501   34.163834
2019-01-04   36.132500   37.137501   35.950001   37.064999   35.622253
2019-01-07   37.174999   37.207500   36.474998   36.982498   35.542969
2019-01-08   37.389999   37.955002   37.130001   37.687500   36.220531
…                  …           …           …           …           …
2023-09-18  176.479996  179.380005  176.169998  177.970001  177.970001
2023-09-19  177.520004  179.630005  177.130005  179.070007  179.070007
2023-09-20  179.259995  179.699997  175.399994  175.490005  175.490005
2023-09-21  174.550003  176.300003  173.860001  173.929993  173.929993
2023-09-22  174.669998  177.078995  174.054993  174.789993  174.789993

               Volume
Date
2019-01-02  148158800
2019-01-03  365248800
```

```
2019-01-04  234428400
2019-01-07  219111200
2019-01-08  164101200
…                 …
2023-09-18   67257600
2023-09-19   51826900
2023-09-20   58436200
2023-09-21   63047900
2023-09-22   55110610

[1190 rows x 6 columns]
```

[71]:
```
#handling Missing Value
#Forward fill missing value in case of gaps in data
stock_data['Close'].fillna(method='ffill', inplace=True)
```

[72]:
```
stock_data
```

[72]:
```
                 Open        High         Low       Close   Adj Close  \
Date
2019-01-02   38.722500   39.712502   38.557499   39.480000   37.943256
2019-01-03   35.994999   36.430000   35.500000   35.547501   34.163834
2019-01-04   36.132500   37.137501   35.950001   37.064999   35.622253
2019-01-07   37.174999   37.207500   36.474998   36.982498   35.542969
2019-01-08   37.389999   37.955002   37.130001   37.687500   36.220531
…                  …           …           …           …           …
2023-09-18  176.479996  179.380005  176.169998  177.970001  177.970001
2023-09-19  177.520004  179.630005  177.130005  179.070007  179.070007
2023-09-20  179.259995  179.699997  175.399994  175.490005  175.490005
2023-09-21  174.550003  176.300003  173.860001  173.929993  173.929993
2023-09-22  174.669998  177.078995  174.054993  174.789993  174.789993

               Volume
Date
2019-01-02  148158800
2019-01-03  365248800
2019-01-04  234428400
2019-01-07  219111200
2019-01-08  164101200
…                 …
2023-09-18   67257600
2023-09-19   51826900
2023-09-20   58436200
2023-09-21   63047900
2023-09-22   55110610

[1190 rows x 6 columns]
```

[73]:
```
#calculate Daily Returns
stock_data['Daily Return'] = stock_data['Close'].pct_change() * 100
```

[74]:
```
#Calculate log Return
stock_data['Log_Return'] = (stock_data['Close']/ stock_data['Close'].shift(1)).
 ↪apply(lambda x: None if pd.isnull(x) else (100*(np.log(x))))
```

[75]:
```
stock_data
```

[75]:

Date	Open	High	Low	Close	Adj Close	Volume	Daily Return	Log_Return
2019-01-02	38.722500	39.712502	38.557499	39.480000	37.943256	148158800	NaN	NaN
2019-01-03	35.994999	36.430000	35.500000	35.547501	34.163834	365248800	-9.960737	-10.492436
2019-01-04	36.132500	37.137501	35.950001	37.064999	35.622253	234428400	4.268930	4.180324
2019-01-07	37.174999	37.207500	36.474998	36.982498	35.542969	219111200	-0.222583	-0.222831
2019-01-08	37.389999	37.955002	37.130001	37.687500	36.220531	164101200	1.906312	1.888370
…	…	…	…	…	…	…	…	…
2023-09-18	176.479996	179.380005	176.169998	177.970001	177.970001	67257600	1.691336	1.677192
2023-09-19	177.520004	179.630005	177.130005	179.070007	179.070007	51826900	0.618085	0.616183
2023-09-20	179.259995	179.699997	175.399994	175.490005	175.490005	58436200	-1.999219	-2.019474
2023-09-21	174.550003	176.300003	173.860001	173.929993	173.929993	63047900	-0.888947	-0.892922
2023-09-22	174.669998	177.078995	174.054993	174.789993	174.789993	55110610	0.494452	0.493234

[1190 rows x 8 columns]

[76]:
```
stock_data.dropna(subset=['Daily Return', 'Log_Return'], inplace=True)
```

[77]:
```
stock_data['Cumulative Return'] = (1+ stock_data['Daily Return'] / 100).
 ↪cumprod() - 1
```

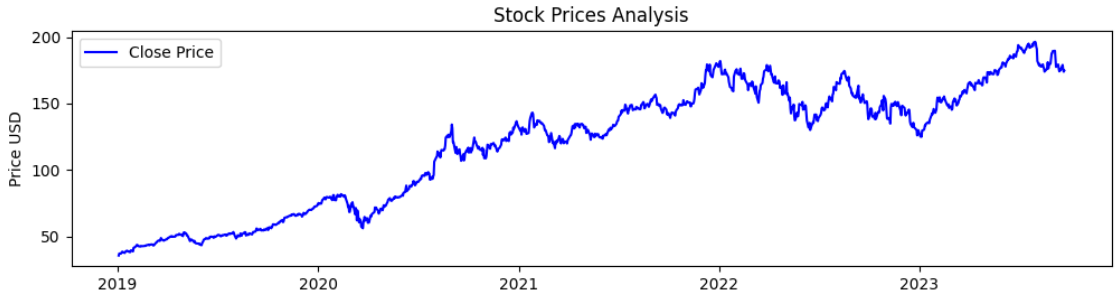
[78]:
```
print(stock_data.head())
```

```
                 Open       High        Low      Close  Adj Close     Volume  \
```

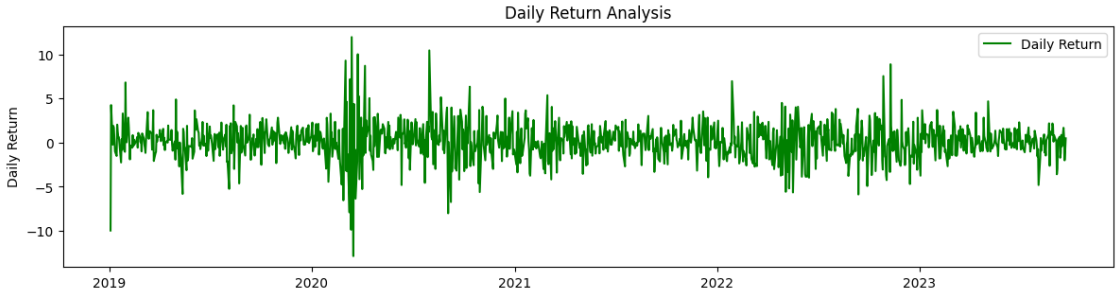
```
Date
2019-01-03  35.994999  36.430000  35.500000  35.547501  34.163834  365248800
2019-01-04  36.132500  37.137501  35.950001  37.064999  35.622253  234428400
2019-01-07  37.174999  37.207500  36.474998  36.982498  35.542969  219111200
2019-01-08  37.389999  37.955002  37.130001  37.687500  36.220531  164101200
2019-01-09  37.822498  38.632500  37.407501  38.327499  36.835617  180396400

            Daily Return  Log_Return  Cumulative Return
Date
2019-01-03     -9.960737  -10.492436          -0.099607
2019-01-04      4.268930    4.180324          -0.061170
2019-01-07     -0.222583   -0.222831          -0.063260
2019-01-08      1.906312    1.888370          -0.045403
2019-01-09      1.698174    1.683916          -0.029192
```

[79]:
```
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1 )
plt.plot(stock_data['Close'], label='Close Price', color='Blue')
plt.title('Stock Prices Analysis')
plt.ylabel('Price USD')
plt.legend()
```

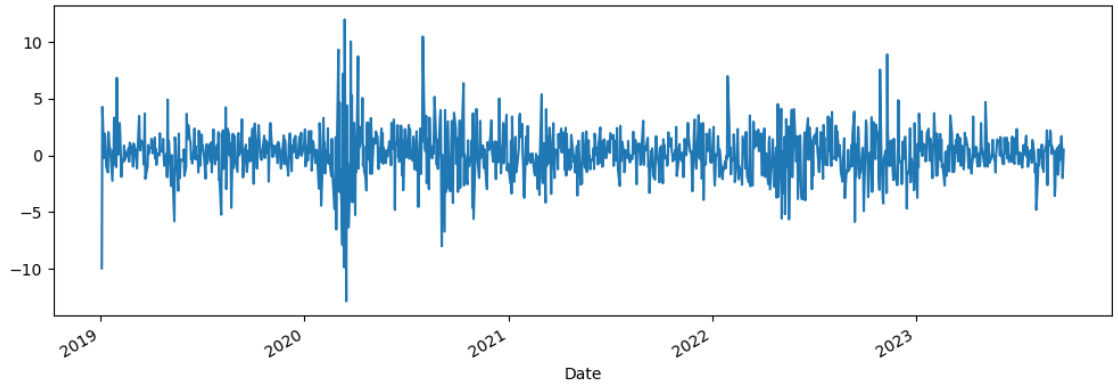
[79]: <matplotlib.legend.Legend at 0x7c82f393f640>



[80]:
```
#Return Analysis
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 2)
plt.plot(stock_data['Daily Return'], label='Daily Return', color='green')
plt.title('Daily Return Analysis')
plt.ylabel('Daily Return')
plt.legend()
plt.tight_layout()
```



[81]:
```
fig ,ax = plt.subplots(figsize=(12, 4))
stock_data['Daily Return'].plot(ax=ax);
```



2 Time Series Decompositions

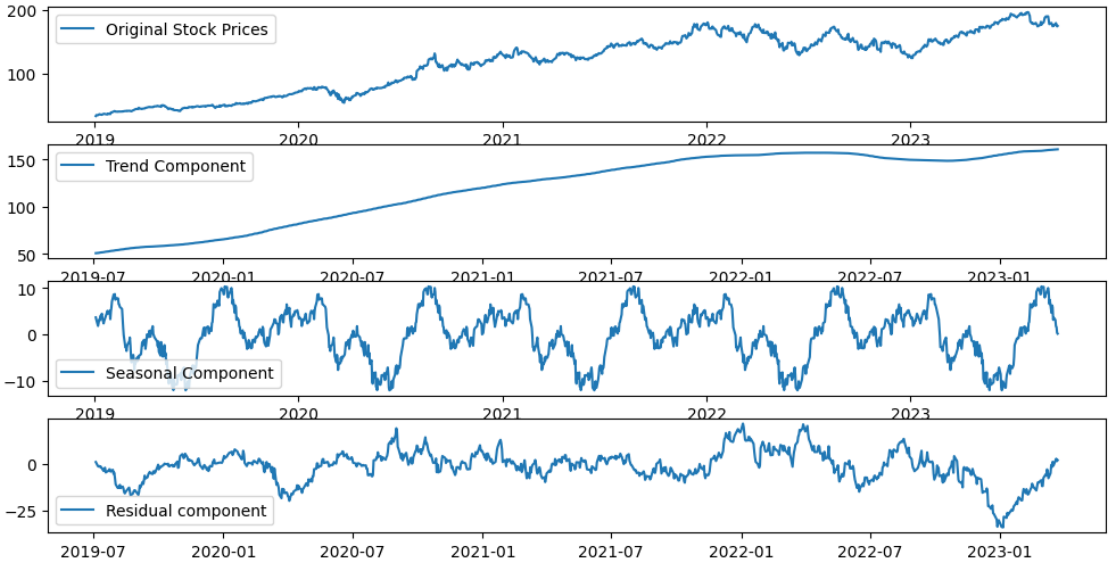
[82]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
```

[83]:
```
stock_prices = stock_data['Adj Close']
```

[84]:
```
result= seasonal_decompose(stock_prices, model='addictive', period=252)
```

[88]:
```
plt.figure(figsize=(12, 6))

plt.subplot(411)
plt.plot(stock_prices, label='Original Stock Prices')
plt.legend()

plt.subplot(412)
plt.plot(result.trend, label='Trend Component')
plt.legend()
```

```
plt.subplot(413)
plt.plot(result.seasonal, label='Seasonal Component')
plt.legend()

plt.subplot(414)
plt.plot(result.resid, label='Residual component')
plt.legend()

plt.show()
plt.tight_layout()
```




```
<Figure size 640x480 with 0 Axes>
```

3 *Volatility Garch*

[90]:
```
!pip install arch
```

```
Requirement already satisfied: arch in /usr/local/lib/python3.10/dist-packages
(6.1.0)
Requirement already satisfied: numpy>=1.19 in /usr/local/lib/python3.10/dist-
packages (from arch) (1.23.5)
Requirement already satisfied: scipy>=1.5 in /usr/local/lib/python3.10/dist-
packages (from arch) (1.11.2)
Requirement already satisfied: pandas>=1.1 in /usr/local/lib/python3.10/dist-
packages (from arch) (1.5.3)
Requirement already satisfied: statsmodels>=0.12 in
/usr/local/lib/python3.10/dist-packages (from arch) (0.14.0)
```

```
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.1->arch) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.1->arch) (2023.3.post1)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-
packages (from statsmodels>=0.12->arch) (0.5.3)
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.10/dist-packages (from statsmodels>=0.12->arch) (23.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from patsy>=0.5.2->statsmodels>=0.12->arch) (1.16.0)
```

[91]:
```
from arch import arch_model
```

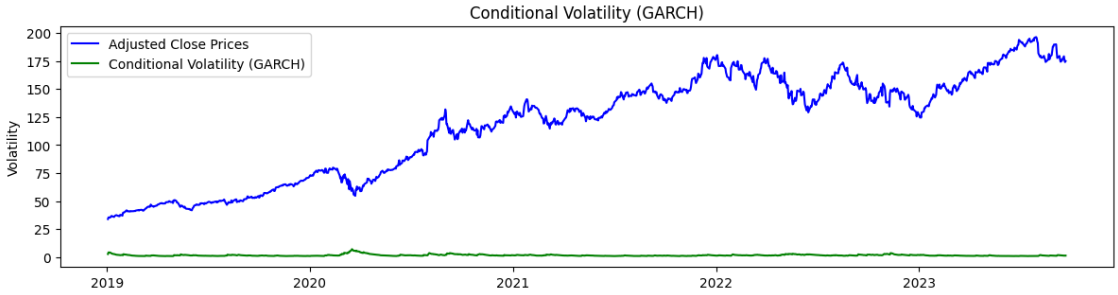
[92]:
```
model = arch_model(stock_data['Daily Return'], vol='Garch', p=1, q=1)
results = model.fit()
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 7094.188659020681
Iteration:      2,   Func. Count:     16,   Neg. LLF: 180592995930.79388
Iteration:      3,   Func. Count:     24,   Neg. LLF: 2800.9432418003344
Iteration:      4,   Func. Count:     31,   Neg. LLF: 2784.186029104576
Iteration:      5,   Func. Count:     38,   Neg. LLF: 2419.5217245842377
Iteration:      6,   Func. Count:     44,   Neg. LLF: 2419.0228627426386
Iteration:      7,   Func. Count:     49,   Neg. LLF: 2419.0220910105472
Iteration:      8,   Func. Count:     54,   Neg. LLF: 2419.0220837354263
Iteration:      9,   Func. Count:     58,   Neg. LLF: 2419.0220837353813
Optimization terminated successfully    (Exit mode 0)
            Current function value: 2419.0220837354263
            Iterations: 9
            Function evaluations: 58
            Gradient evaluations: 9
```

[93]:
```
conditional_volatility = results.conditional_volatility
```

[94]:
```
plt.figure(figsize=(12, 6))


#Plot Stock Price
plt.subplot(2, 1, 2)
plt.plot(stock_prices, label='Adjusted Close Prices', color='Blue')
plt.title('Historical Stock Prices')
plt.ylabel('Prices')
plt.legend()

#Plot Conditional Volatility
plt.subplot(2, 1, 2)
plt.plot(conditional_volatility, label='Conditional Volatility (GARCH)',␣
 ↪color='green')
```

```
plt.title('Conditional Volatility (GARCH)')
plt.ylabel('Volatility')
plt.legend()

plt.tight_layout()
plt.show()
```



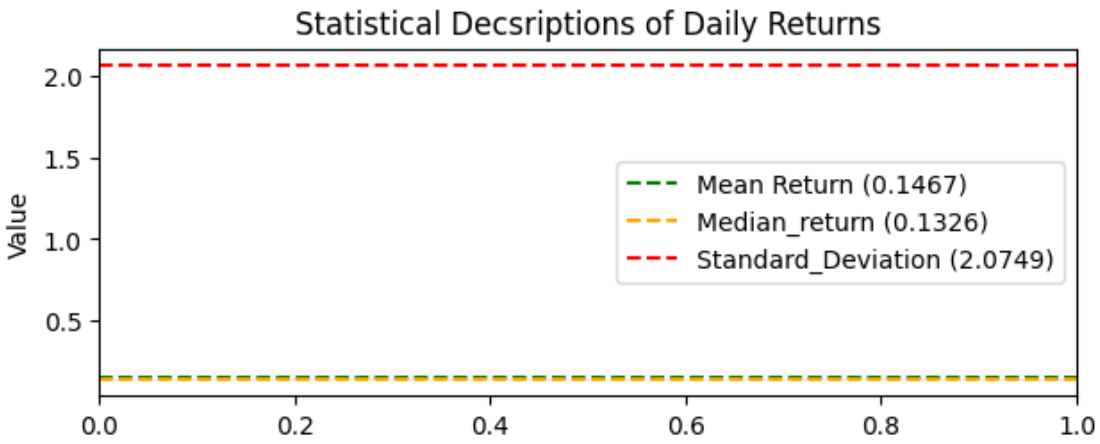
4 *Statistical Descriptions*

[97]:
```
mean_return = stock_data['Daily Return'].mean()
median_return = stock_data['Daily Return'].median()
std_deviation = stock_data['Daily Return'].std()
skewness = stock_data['Daily Return'].skew()
kurtosis = stock_data['Daily Return'].kurtosis()
```

[98]:
```
mean_return
```

[98]: 0.14673047261990926

[102]:
```
plt.subplot(2, 1, 2)
plt.axhline(mean_return, color='green', linestyle='--', label=f'Mean Return␣
 ↪({mean_return:.4f})')
plt.axhline(median_return, color='orange', linestyle='--',␣
 ↪label=f'Median_return ({median_return:.4f})')
plt.axhline(std_deviation, color='red', linestyle='--',␣
 ↪label=f'Standard_Deviation ({std_deviation:.4f})')
plt.title('Statistical Decsriptions of Daily Returns')
plt.ylabel('Value')
plt.legend()

plt.tight_layout()
plt.show()
```


Statistical Decsriptions of Daily Returns
Value
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
Mean Return (0.1467)
Median_return (0.1326)
Standard_Deviation (2.0749)