DA finance

March 2, 2024

1 Download data from Yahoo!

2 Download data from Yahoo!

[1]:
```
import yfinance
import mplfinance as mpf
import pandas as pd
import numpy as np
import statsmodels.api as sm
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import datetime as dt
import os
sns.set()
```

[2]:
```
stocknames=['AAPL','MSFT','TSLA','GC=F','SPY']
startdate='2020-01-01'
enddate=dt.datetime.now().date()
interval='1d'
```

[3]:
```
for stock in stocknames:
    df=yfinance.download(stock,interval=interval,start=startdate,end=enddate)
    df.to_csv('{}.csv'.format(stock))
```

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

[4]:
```
def appending(cols=[],startdate='',enddate=''):
    global df
    dates=pd.date_range(start=startdate,end=enddate)
    df=pd.DataFrame(index=dates)
    for stock in stocknames:
```

```
        df1=pd.read_csv(os.path.join('{}.csv'.
↪format(stock)),index_col='Date',parse_dates=True,usecols=cols,na_values=['Nan'])
        df1=df1.rename(columns={'Adj Close':stock})
        df=df.join(df1)
    return df
```

[5]:
```
appending(['Date','Adj Close'],startdate,enddate)
df.index.names=['Data']
df
```

[5]:
```
                 AAPL        MSFT        TSLA        GC=F        SPY
Data
2020-01-01        NaN         NaN         NaN         NaN         NaN
2020-01-02  73.059425  154.493835   28.684000  1524.500000  305.058441
2020-01-03  72.349144  152.570129   29.534000  1549.199951  302.748474
2020-01-04        NaN         NaN         NaN         NaN         NaN
2020-01-05        NaN         NaN         NaN         NaN         NaN
…                 …           …           …           …           …
2024-02-27 182.630005  407.480011  199.729996  2034.000000  506.929993
2024-02-28 181.419998  407.720001  202.039993  2033.000000  506.260010
2024-02-29 180.750000  413.640015  201.880005  2045.699951  508.079987
2024-03-01 179.660004  415.500000  202.639999  2091.600098  512.849976
2024-03-02        NaN         NaN         NaN         NaN         NaN

[1523 rows x 5 columns]
```

[6]:
```
df=df.dropna()
df.head()
```

[6]:
```
                 AAPL        MSFT       TSLA         GC=F         SPY
Data
2020-01-02  73.059425  154.493835  28.684000  1524.500000  305.058441
2020-01-03  72.349144  152.570129  29.534000  1549.199951  302.748474
2020-01-06  72.925629  152.964478  30.102667  1566.199951  303.903412
2020-01-07  72.582672  151.569778  31.270666  1571.800049  303.048981
2020-01-08  73.750229  153.984055  32.809334  1557.400024  304.664124
```

[ ]:
```

```

[7]:
```
df=df.rename(columns={'GC=F':'GOLD',"SPY":'S&P500'})
```

2.1 Statistics

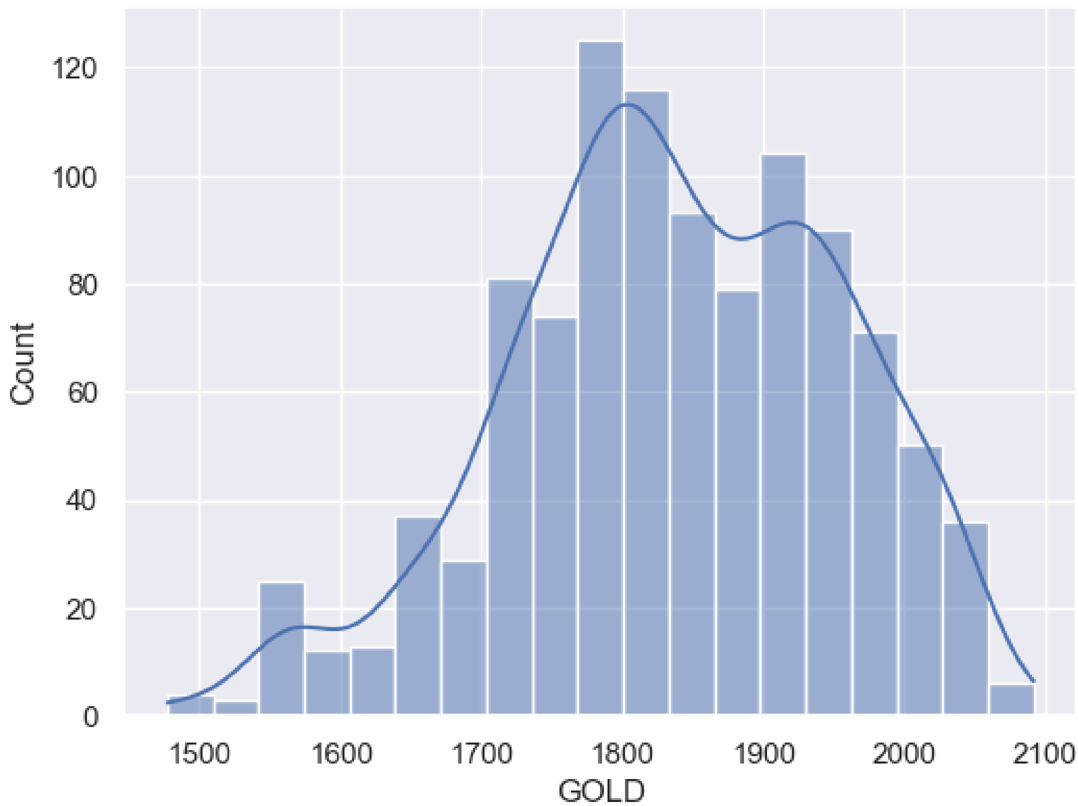
[8]:
```
df.describe().round(2)
```

[8]:
```
          AAPL     MSFT     TSLA     GOLD   S&P500
count  1048.00  1048.00  1048.00  1048.00  1048.00
```

mean	141.09	263.72	208.95	1836.18	388.25
std	34.24	61.19	84.15	118.43	57.59
min	54.71	130.61	24.08	1477.30	210.58
25%	122.67	218.60	163.15	1761.80	356.23
50%	145.11	258.33	221.51	1834.25	399.76
75%	168.61	306.58	260.63	1927.62	430.64
max	197.86	419.77	409.97	2091.60	512.85

[9]:
```python
sns.histplot(data=df,x='GOLD',kde=True)#normal distribution
```

[9]: <Axes: xlabel='GOLD', ylabel='Count'>

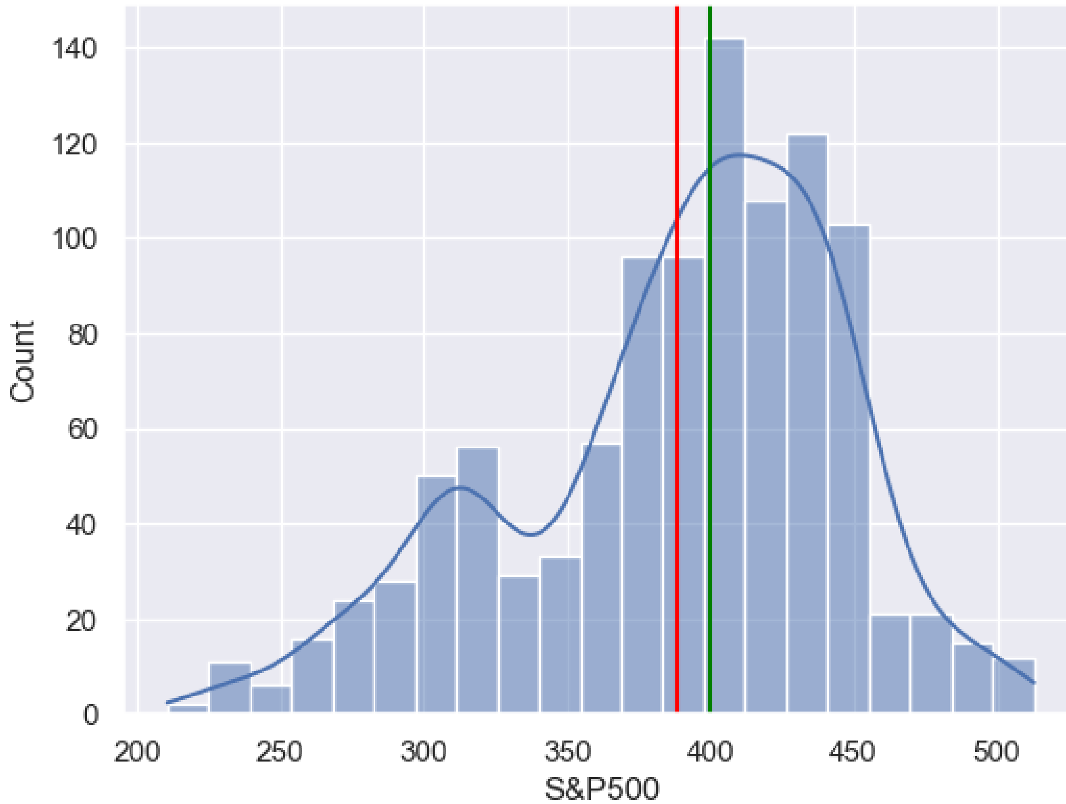


[10]:
```python
sns.histplot(data=df,x='GOLD',kde=True)
plt.axvline(df.GOLD.mean(),color='red')
plt.axvline(df.GOLD.median(),color='green')
#the gold has normal distribution so mean=median
```

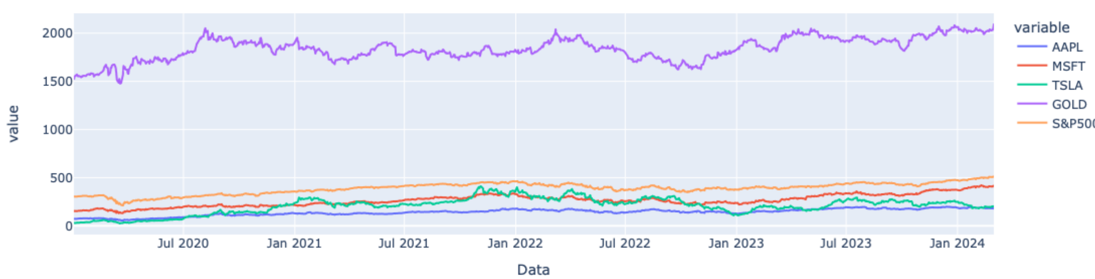
[10]: <matplotlib.lines.Line2D at 0x16c05e390>

[11]:
```
sns.histplot(data=df,x='S&P500',kde=True)
plt.axvline(df['S&P500'].mean(),color='red')
plt.axvline(df['S&P500'].median(),color='green')
#median>mean becouse have outliers
```

[11]:
```
<matplotlib.lines.Line2D at 0x16c005b50>
```



[12]:
```
px.line(data_frame=df)
```



Due to the difference in prices, a unified index must be taken(Normalization)adjusting values measured on different scales to a notionally common scale

2.2 Normalizing

[13]:
```
df_norm=df/df.iloc[0,:]
```

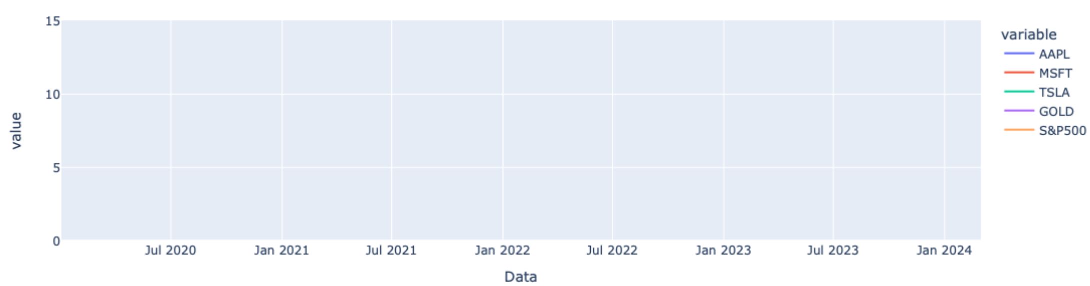
[14]:
```
df_norm
```

[14]:

Data	AAPL	MSFT	TSLA	GOLD	S&P500
2020-01-02	1.000000	1.000000	1.000000	1.000000	1.000000
2020-01-03	0.990278	0.987548	1.029633	1.016202	0.992428
2020-01-06	0.998169	0.990101	1.049458	1.027353	0.996214
2020-01-07	0.993474	0.981073	1.090178	1.031027	0.993413
2020-01-08	1.009455	0.996700	1.143820	1.021581	0.998707
…	…	…	…	…	…
2024-02-26	2.479625	2.637905	6.951610	1.330600	1.658666
2024-02-27	2.499746	2.637516	6.963115	1.334208	1.661747
2024-02-28	2.483184	2.639070	7.043648	1.333552	1.659551
2024-02-29	2.474013	2.677388	7.038070	1.341883	1.665517
2024-03-01	2.459094	2.689428	7.064566	1.371991	1.681153

[1048 rows x 5 columns]

[23]:
```
px.line(data_frame=df_norm)
```



we found that TSLA has more varaition over time

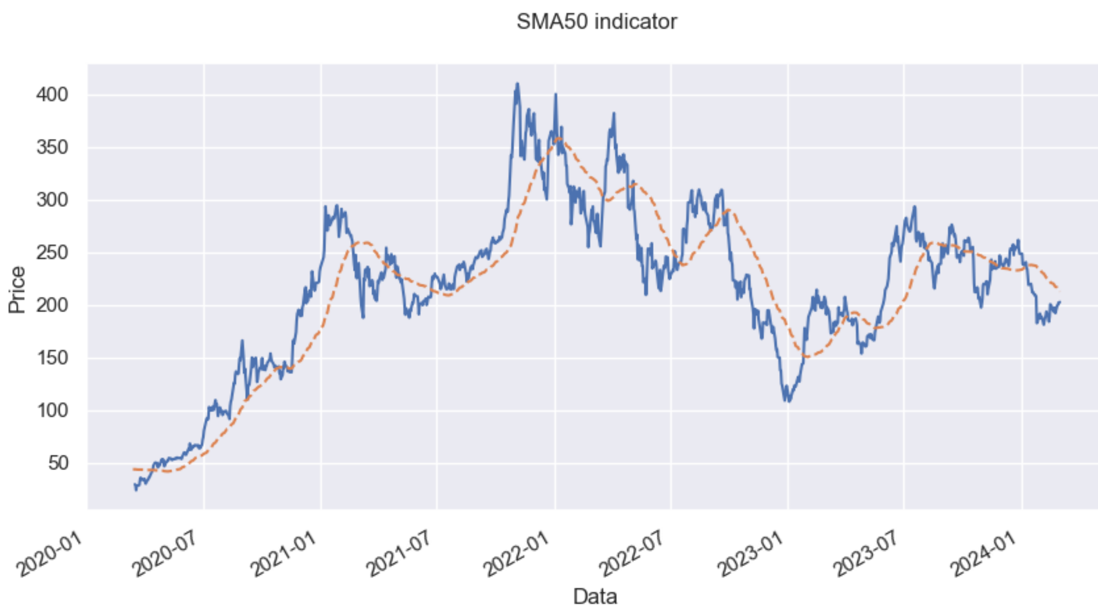
2.3 SMA50 indicator

[16]:
```
rpllingmean=df.TSLA.rolling(50).mean()
```

[17]:
```
fig, ax=plt.subplots(figsize=(10,5))
df.iloc[50:]['TSLA'].plot(ax=ax)
ax.set_title('\nSMA50 indicator\n')
ax.set_label('TIME')
ax.set_ylabel('Price')
```

```
rpllingmean.plot(ax=ax,linestyle='--')
```

[17]: <Axes: label='TIME', title={'center': '\nSMA50 indicator\n'}, xlabel='Data', ylabel='Price'>

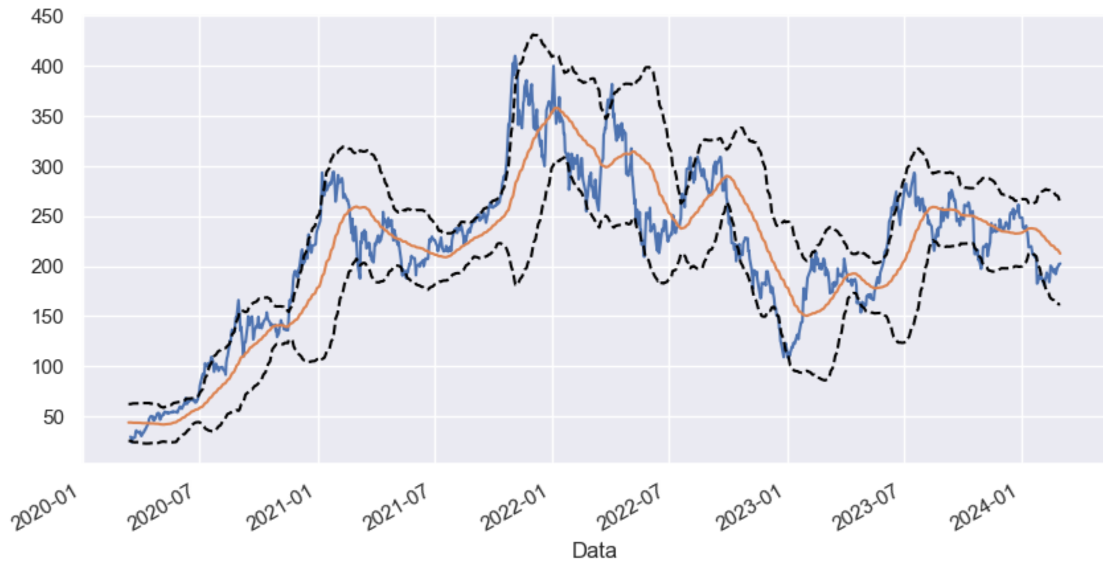


2.4 Bollinger indicator

This indicator calculates the standard deviation up to the second degree and the extent of the impact of the decision to buy and sell

[18]:
```
df.iloc[50:]['TSLA'].plot()
rpllingmean=df.TSLA.rolling(50).mean()
rpllingstd=df.TSLA.rolling(50).std()
lower=rpllingmean-(2*rpllingstd)
upper=rpllingmean+(2*rpllingstd)
rpllingmean.plot(figsize=(10,5))
lower.plot(linestyle='--',color='black')
upper.plot(linestyle='--',color='black')
#plt.grid(alpha=0.25)
```

[18]: <Axes: xlabel='Data'>

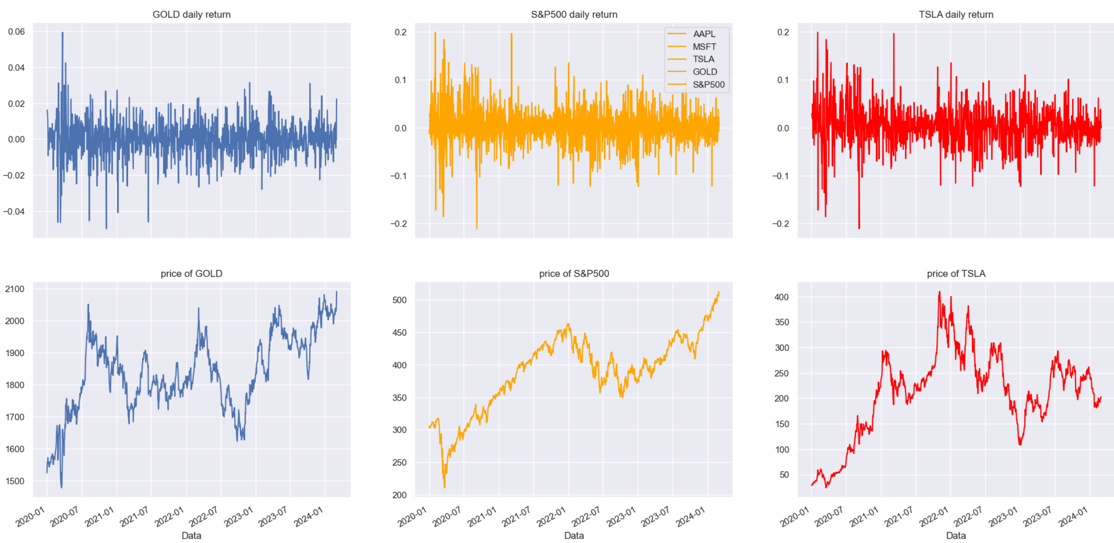


2.5 daily return indicator

[19]:
```
daily_return=df.pct_change()
```

[20]:
```
fig, ax =plt.subplots(2,3,figsize=(24,12),sharex=True)
daily_return.GOLD.plot(ax=ax[0][0])
ax[0][0].set_title('GOLD daily return')
daily_return.plot(ax=ax[0][1],color='orange')
ax[0][1].set_title('S&P500 daily return')
daily_return.TSLA.plot(ax=ax[0][2],color='red')
ax[0][2].set_title('TSLA daily return')
#daily_return.plot(ax=ax[0][2],color='red')
df.GOLD.plot(ax=ax[1][0])
ax[1][0].set_title("price of GOLD")
df['S&P500'].plot(ax=ax[1][1],color='orange')
ax[1][1].set_title("price of S&P500")
df.TSLA.plot(ax=ax[1][2],color='red')
ax[1][2].set_title("price of TSLA")
```

[20]: Text(0.5, 1.0, 'price of TSLA')



2.6 Cumulative return indicator

[21]:
```python
stocknames=df.columns
for stock in stocknames:
    print('Cumulative rate of {} return is {} %'.format(stock,round(((df.
 ↪iloc[-1][stock]/df.iloc[0][stock])-1)*100,2)))
```

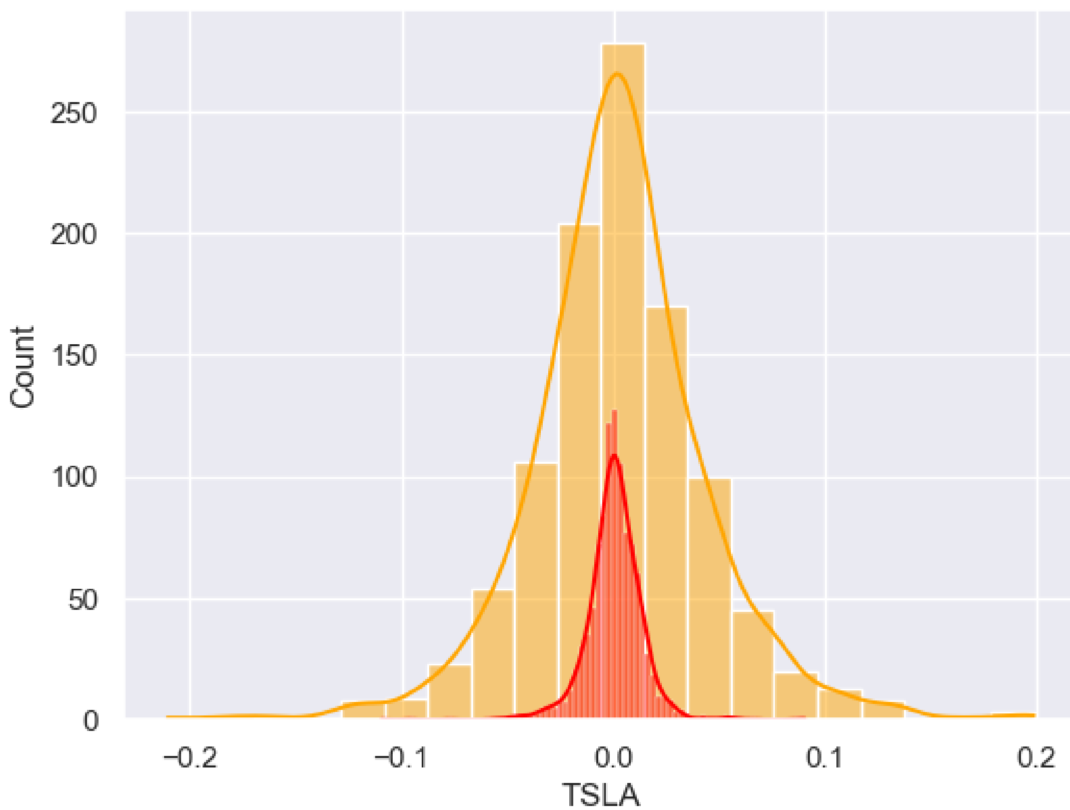
```
Cumulative rate of AAPL return is 145.91 %
Cumulative rate of MSFT return is 168.94 %
Cumulative rate of TSLA return is 606.46 %
Cumulative rate of GOLD return is 37.2 %
Cumulative rate of S&P500 return is 68.12 %
```

2.7 Risk return indicator

This indicator studies the extent of the presence of outliers

[22]:
```python
sns.histplot(daily_return['TSLA'],kde=True,bins=20,color="orange")
sns.histplot(daily_return['S&P500'],kde=True,color='red')
```

[22]: <Axes: xlabel='TSLA', ylabel='Count'>


Count
250
200
150
100
50
0
−0.2
−0.1
0.0
0.1
0.2
TSLA