```
#importing important Libraries

import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import yfinance as yf
```

```
# prompt: download from yfinance  ticker(TCPLPACK.NS) and (^NSEI) start 2018-2-28 end 2025-2-28
#  Close

data = yf.download(tickers=['TCPLPACK.NS', '^NSEI'], start='2018-02-28', end='2025-02-28')
data = data['Close']
data
```

```
[**********100%**********]  2 of 2 completed
```

Ticker	TCPLPACK.NS	^NSEI
Date		
2018-02-28	569.492065	10492.849609
2018-03-01	567.103943	10458.349609
2018-03-05	580.376648	10358.849609
2018-03-06	565.863770	10249.250000
2018-03-07	556.448853	10154.200195
...	...	...
2025-02-20	4140.950195	22913.150391
2025-02-21	4003.850098	22795.900391
2025-02-24	3852.399902	22553.349609
2025-02-25	4152.399902	22547.550781
2025-02-27	4083.550049	22545.050781

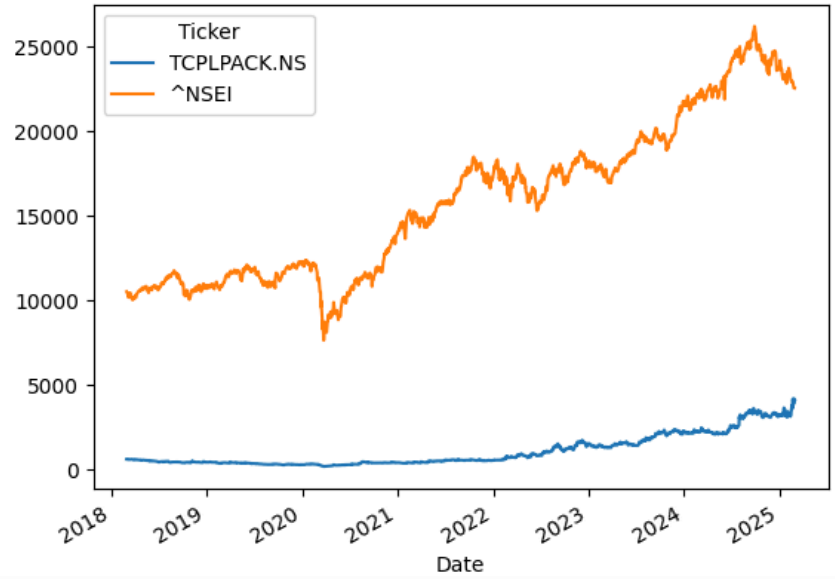
1729 rows × 2 columns

```
data.head()
```

Ticker	TCPLPACK.NS	^NSEI
Date		
2018-02-28	569.492065	10492.849609
2018-03-01	567.103943	10458.349609
2018-03-05	580.376648	10358.849609
2018-03-06	565.863770	10249.250000
2018-03-07	556.448853	10154.200195

```
# prompt: data .['TCPLPACK.NS'] & [NSEI].plot()

data[['TCPLPACK.NS', '^NSEI']].plot()
```

```
<Axes: xlabel='Date'>
```

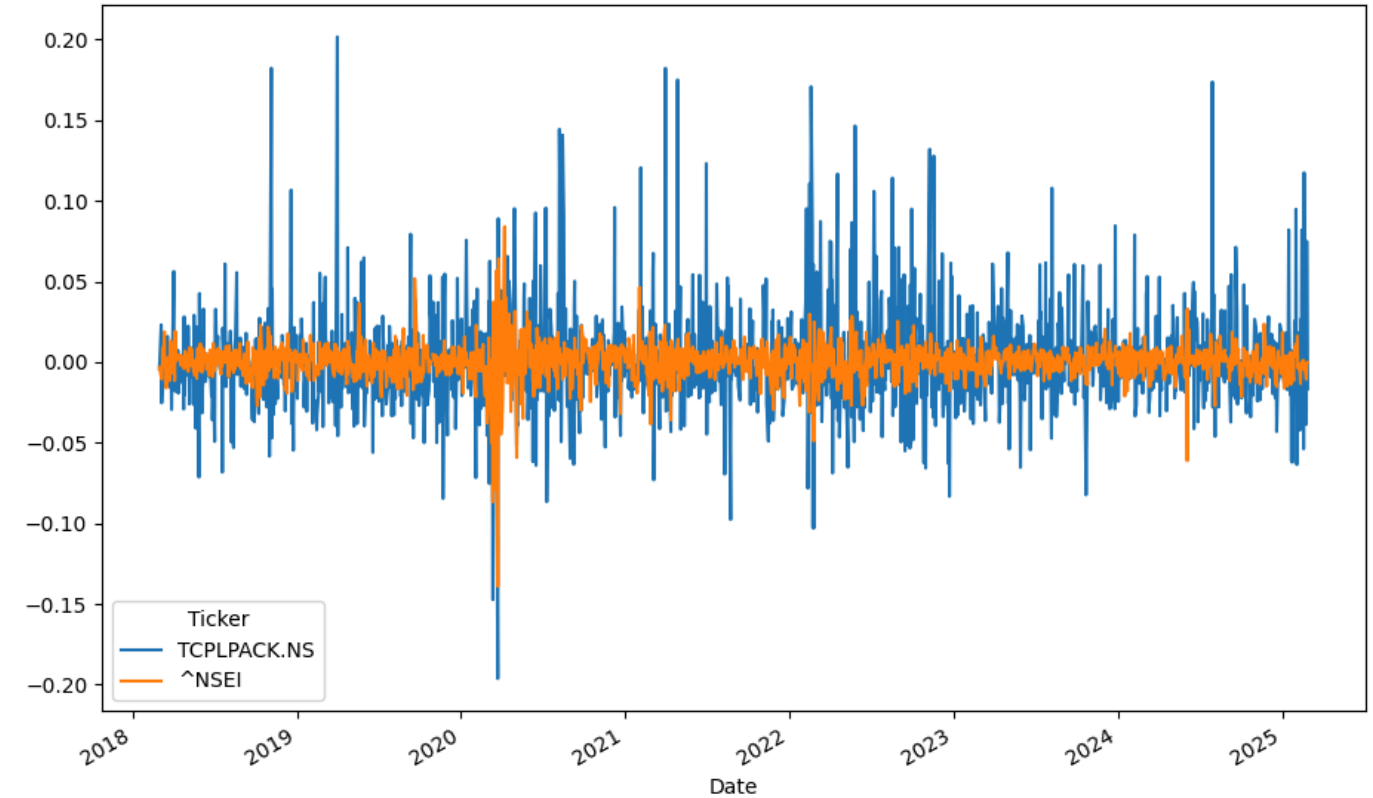



```
# prompt: return=np.log (data/data.shift(1))

returns = np.log(data / data.shift(1))
returns.head()
```

Ticker	TCPLPACK.NS	^NSEI
Date		
2018-02-28	NaN	NaN
2018-03-01	-0.004202	-0.003293
2018-03-05	0.023135	-0.009559
2018-03-06	-0.025324	-0.010637
2018-03-07	-0.016778	-0.009317

```
# prompt: returns.plot(figszie= (11,7))

returns.plot(figsize=(11, 7))
```

```
<Axes: xlabel='Date'>
```

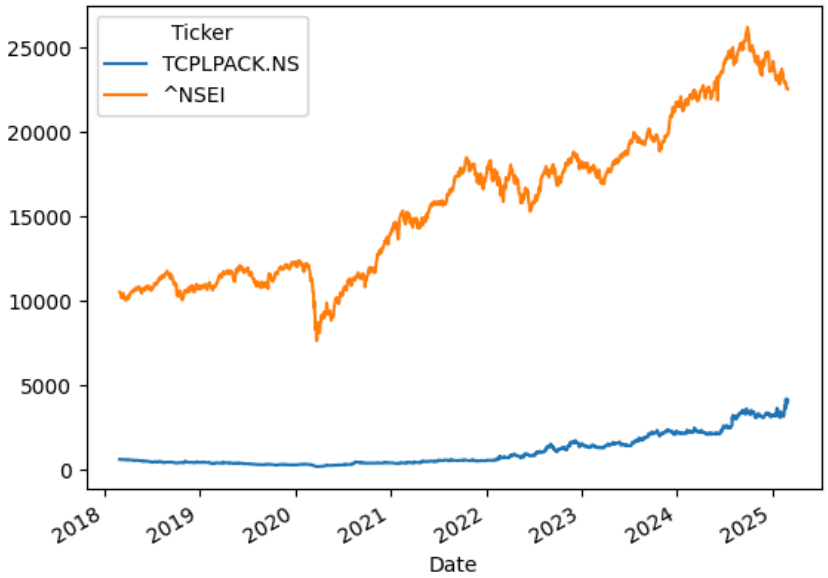


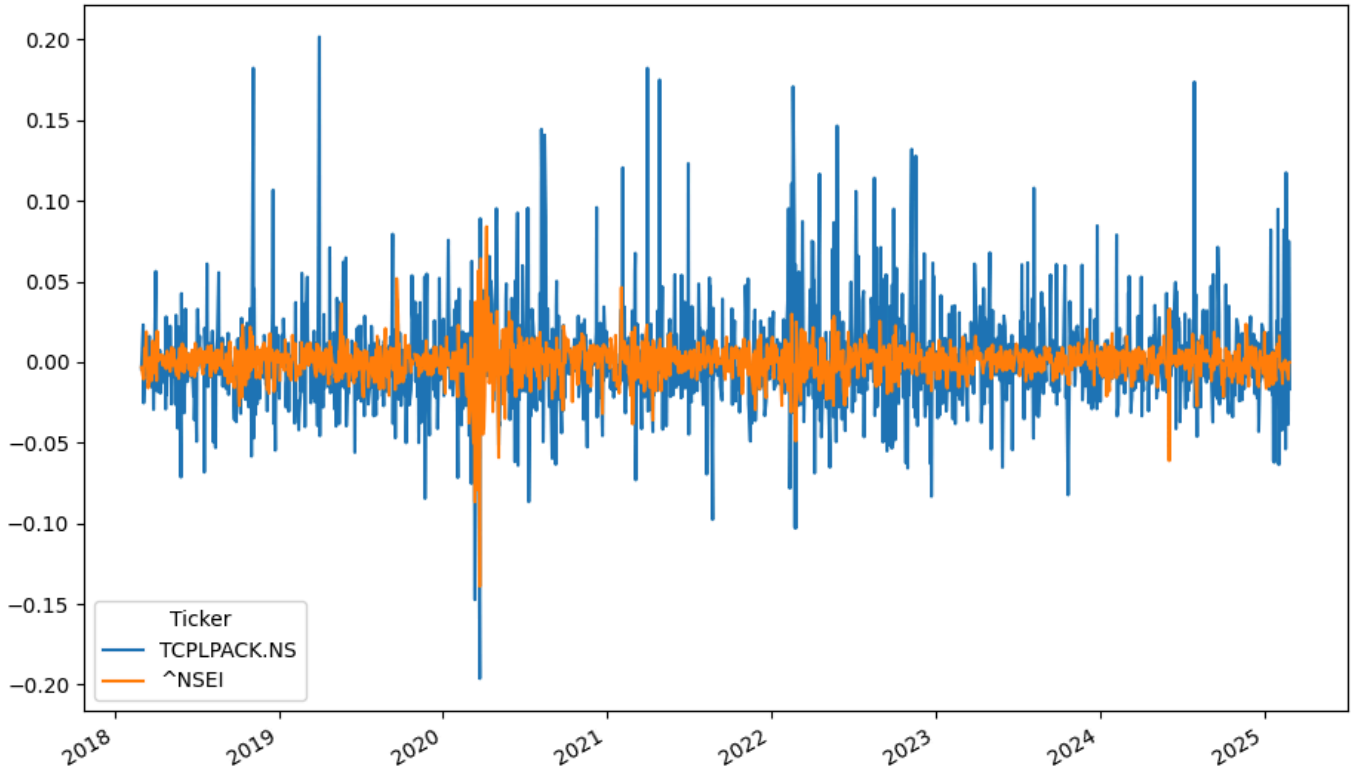
```
# prompt: cov = returns.cov()

cov = returns.cov()
cov
```

Ticker	TCPLPACK.NS	^NSEI
Ticker		
TCPLPACK.NS	0.000945	0.000097
^NSEI	0.000097	0.000126

```
# prompt: cov with market  = cov.iloc[0.1]

import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import yfinance as yf
#importing important Libraries


data = yf.download(tickers=['TCPLPACK.NS', '^NSEI'], start='2018-02-28', end='2025-02-28')
data = data['Close']
data

data.head()

data[['TCPLPACK.NS', '^NSEI']].plot()


returns = np.log(data / data.shift(1))
returns.head()


returns.plot(figsize=(11, 7))


cov = returns.cov()
cov

# Assuming you want the covariance of 'TCPLPACK.NS' with the market ('^NSEI')
market_cov = cov.iloc[0, 1]  # Accessing the covariance value at row 0 (TCPLPACK.NS), column 1 (^NSEI)
market_cov
```

```
[**********100%**********]  2 of 2 completed
9.709823108220431e-05
```





Date

```
# prompt: cov_with _market  = cov _with _makret*250

cov_with_market = market_cov * 250
cov_with_market
```

```
0.02427455777055108
```

```
# prompt: market _var =return ['^NSEI'].var()

market_var = returns['^NSEI'].var()
market_var
```

```
0.00012601918282404014
```

```
# prompt: beta = cov_var_market/market_var

annualized_market_var = market_var * 250
annualized_market_var

beta = cov_with_market / annualized_market_var
beta
```

```
0.770503576568831
```

```
# prompt: import statsmodel.api as sm

# Assuming you have already calculated beta as shown in your previous code
# beta = cov_with_market / annualized_market_var

# Now, using statsmodels to perform linear regression
X = returns['^NSEI']  # Market returns as the independent variable
X = sm.add_constant(X)  # Add a constant term to the independent variable
y = returns['TCPLPACK.NS']  # Stock returns as the dependent variable

model = sm.OLS(y, X).fit()
print(model.summary())
```

```
---------------------------------------------------------------------------
MissingDataError                          Traceback (most recent call last)
<ipython-input-50-ec0f6d187345> in <cell line: 0>()
      9 y = returns['TCPLPACK.NS']  # Stock returns as the dependent variable
     10
---> 11 model = sm.OLS(y, X).fit()
     12 print(model.summary())

                                 8 frames
/usr/local/lib/python3.11/dist-packages/statsmodels/base/data.py in _handle_constant(self, hasconst)
    132             exog_max = np.max(self.exog, axis=0)
    133             if not np.isfinite(exog_max).all():
--> 134                 raise MissingDataError('exog contains inf or nans')
    135             exog_min = np.min(self.exog, axis=0)
    136             const_idx = np.where(exog_max == exog_min)[0].squeeze()

MissingDataError: exog contains inf or nans
```

```
# prompt: import statsmodel.api as sm

# Assuming you have already calculated beta as shown in your previous code
# beta = cov_with_market / annualized_market_var

# Now, using statsmodels to perform linear regression
X = returns['^NSEI']  # Market returns as the independent variable
X = sm.add_constant(X)  # Add a constant term to the independent variable
y = returns['TCPLPACK.NS']  # Stock returns as the dependent variable

# Drop rows with missing values (NaN or inf) in either X or y
# This is done before adding the constant to X
# since add_constant might introduce NaNs in the constant column
# if there are NaNs in the original data.
merged_data = pd.concat([X, y], axis=1).dropna()
X = merged_data['^NSEI']
X = sm.add_constant(X)
y = merged_data['TCPLPACK.NS']


model = sm.OLS(y, X).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           TCPLPACK.NS   R-squared:                       0.080
Model:                           OLS   Adj. R-squared:                  0.079
Method:                Least Squares   F-statistic:                     148.7
Date:               Sat, 08 Mar 2025   Prob (F-statistic):           7.26e-33
Time:                       16:57:18   Log-Likelihood:                 3626.0
No. Observations:               1720   AIC:                            -7248.
Df Residuals:                   1718   BIC:                            -7237.
Df Model:                          1
```

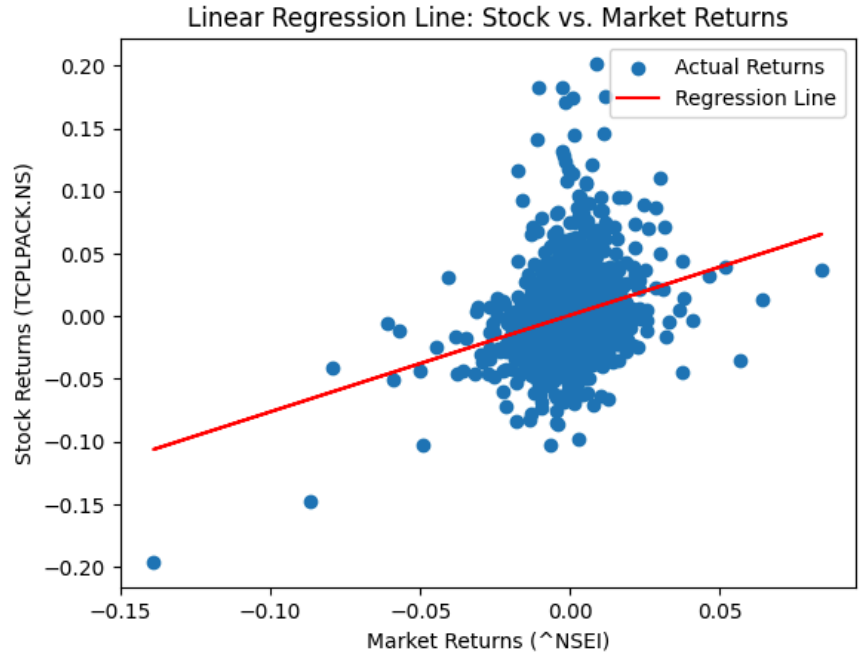
```
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0008      0.001      1.154      0.248      -0.001       0.002
^NSEI          0.7705      0.063     12.195      0.000       0.647       0.894
==============================================================================
Omnibus:                      619.350   Durbin-Watson:                   2.149
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4189.907
Skew:                           1.521   Prob(JB):                         0.00
Kurtosis:                      10.015   Cond. No.                         89.1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
returns.dropna(inplace=True)
```

```
# Assuming you already have the stock ('TCPLPACK.NS') and market ('^NSEI') return in sparate data frmae
import numpy as np
import matplotlib.pyplot as plt  # Import for plotting
import statsmodels.api as sm  # Import for regression

# Perform OLS regression
X = sm.add_constant(returns['^NSEI'])

# Replace infinite values with NaN
X.replace([np.inf, -np.inf], np.nan, inplace=True)

# Drop rows with NaN values
X.dropna(inplace=True)

# Ensure 'TCPLPACK.NS' aligns with the index of X
y = returns['TCPLPACK.NS'][X.index]

model = sm.OLS(y, X).fit()

# Extract beta coefficient
beta_coefficient = model.params['^NSEI']
print(f"Beta coefficient: {beta_coefficient}")

# Calculate predicted returns
predicted_returns = model.predict(X) # Use model.predict for predictions

# Plot the scatter plot and regression line
plt.scatter(returns['^NSEI'], returns['TCPLPACK.NS'], label='Actual Returns')
plt.plot(returns['^NSEI'][X.index], predicted_returns, color='red', label='Regression Line') # Align x-axis with predicted values
plt.xlabel('Market Returns (^NSEI)')
plt.ylabel('Stock Returns (TCPLPACK.NS)')
plt.legend()
plt.title('Linear Regression Line: Stock vs. Market Returns')
```

```
plt.show()

print(f"Beta coefficient: {beta_coefficient}")
```

```
Beta coefficient: 0.7705037812000399
```




```
Beta coefficient: 0.7705037812000399
```

+ Code + Text