```
!pip install yfinance
```

```
Collecting yfinance
  Using cached yfinance-0.2.66-py2.py3-none-any.whl.metadata (6.0 kB)
Requirement already satisfied: pandas>=1.3.0 in c:\users\chait\
anaconda3\lib\site-packages (from yfinance) (2.3.3)
Requirement already satisfied: numpy>=1.16.5 in c:\users\chait\
anaconda3\lib\site-packages (from yfinance) (2.3.5)
Requirement already satisfied: requests>=2.31 in c:\users\chait\
anaconda3\lib\site-packages (from yfinance) (2.32.5)
Collecting multitasking>=0.0.7 (from yfinance)
  Using cached multitasking-0.0.12.tar.gz (19 kB)
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
  Preparing metadata (pyproject.toml): started
  Preparing metadata (pyproject.toml): finished with status 'done'
Requirement already satisfied: platformdirs>=2.0.0 in c:\users\chait\
anaconda3\lib\site-packages (from yfinance) (4.5.0)
Requirement already satisfied: pytz>=2022.5 in c:\users\chait\
anaconda3\lib\site-packages (from yfinance) (2025.2)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\chait\
anaconda3\lib\site-packages (from yfinance) (2.4.6)
Collecting peewee>=3.16.2 (from yfinance)
  Using cached peewee-3.18.3.tar.gz (3.0 MB)
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
  Preparing metadata (pyproject.toml): started
  Preparing metadata (pyproject.toml): finished with status 'done'
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\
chait\anaconda3\lib\site-packages (from yfinance) (4.13.5)
Collecting curl_cffi>=0.7 (from yfinance)
  Using cached curl_cffi-0.13.0-cp39-abi3-win_amd64.whl.metadata (13
kB)
Requirement already satisfied: protobuf>=3.19.0 in c:\users\chait\
anaconda3\lib\site-packages (from yfinance) (5.29.3)
Collecting websockets>=13.0 (from yfinance)
  Downloading websockets-15.0.1-cp313-cp313-win_amd64.whl.metadata
(7.0 kB)
Requirement already satisfied: soupsieve>1.2 in c:\users\chait\
anaconda3\lib\site-packages (from beautifulsoup4>=4.11.1->yfinance)
(2.5)
Requirement already satisfied: typing-extensions>=4.0.0 in c:\users\
chait\anaconda3\lib\site-packages (from beautifulsoup4>=4.11.1-
>yfinance) (4.15.0)
Requirement already satisfied: cffi>=1.12.0 in c:\users\chait\
anaconda3\lib\site-packages (from curl_cffi>=0.7->yfinance) (2.0.0)
```

```
Requirement already satisfied: certifi>=2024.2.2 in c:\users\chait\
anaconda3\lib\site-packages (from curl_cffi>=0.7->yfinance)
(2025.11.12)
Requirement already satisfied: pycparser in c:\users\chait\anaconda3\
lib\site-packages (from cffi>=1.12.0->curl_cffi>=0.7->yfinance) (2.23)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
chait\anaconda3\lib\site-packages (from pandas>=1.3.0->yfinance)
(2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in c:\users\chait\
anaconda3\lib\site-packages (from pandas>=1.3.0->yfinance) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\chait\anaconda3\
lib\site-packages (from python-dateutil>=2.8.2->pandas>=1.3.0-
>yfinance) (1.17.0)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\
chait\anaconda3\lib\site-packages (from requests>=2.31->yfinance)
(3.4.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\chait\
anaconda3\lib\site-packages (from requests>=2.31->yfinance) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\chait\
anaconda3\lib\site-packages (from requests>=2.31->yfinance) (2.5.0)
Using cached yfinance-0.2.66-py2.py3-none-any.whl (123 kB)
Using cached curl_cffi-0.13.0-cp39-abi3-win_amd64.whl (1.6 MB)
Downloading websockets-15.0.1-cp313-cp313-win_amd64.whl (176 kB)
Building wheels for collected packages: multitasking, peewee
  Building wheel for multitasking (pyproject.toml): started
  Building wheel for multitasking (pyproject.toml): finished with
status 'done'
  Created wheel for multitasking: filename=multitasking-0.0.12-py3-
none-any.whl size=15703
sha256=a799d37358beb73f723025597974dadbe26af158940acf3020cf9f4444d2be2
1
  Stored in directory: c:\users\chait\appdata\local\pip\cache\wheels\
1e\df\0f\e2bbb22d689b30c681feb5410ab64a2523437b34c8ecfc6476
  Building wheel for peewee (pyproject.toml): started
  Building wheel for peewee (pyproject.toml): finished with status
'done'
  Created wheel for peewee: filename=peewee-3.18.3-py3-none-any.whl
size=139181
sha256=e20fc0e5d19fed8cc2d8dab8a33417804fa1102350971222cdc683db3cb419a
4
  Stored in directory: c:\users\chait\appdata\local\pip\cache\wheels\
8c\a9\a4\df972cd49f865ffde174d9c5b26f14f08f8a363ed31e10ff91
Successfully built multitasking peewee
Installing collected packages: peewee, multitasking, websockets,
curl_cffi, yfinance

   ---------------------------------------- 0/5 [peewee]
   -------- ------------------------------- 1/5 [multitasking]
   ---------------- ----------------------- 2/5 [websockets]
```

```
   ---------------- ---------------------- 2/5 [websockets]
   ------------------------ --------------- 3/5 [curl_cffi]
   -------------------------------- ------- 4/5 [yfinance]
   -------------------------------- ------- 4/5 [yfinance]
   ---------------------------------------- 5/5 [yfinance]

Successfully installed curl_cffi-0.13.0 multitasking-0.0.12 peewee-
3.18.3 websockets-15.0.1 yfinance-0.2.66

import yfinance as yf

reliance= yf.Ticker("RELIANCE.NS")

reliance= reliance.history(period="max")

reliance

                                  Open          High           Low
Close  \
Date

1996-01-01 00:00:00+05:30     4.600219     4.624903     4.569925
4.617049
1996-01-02 00:00:00+05:30     4.605830     4.628270     4.547485
4.581145
1996-01-03 00:00:00+05:30     4.656318     4.868378     4.605829
4.615927
1996-01-04 00:00:00+05:30     4.572169     4.586755     4.511581
4.573291
1996-01-05 00:00:00+05:30     4.555339     4.555339     4.502605
4.541875
...                                ...          ...          ...
...
2025-12-08 00:00:00+05:30  1539.199951  1551.000000  1535.000000
1543.000000
2025-12-09 00:00:00+05:30  1540.300049  1542.500000  1520.099976
1529.400024
2025-12-10 00:00:00+05:30  1534.000000  1547.500000  1531.400024
1536.900024
2025-12-11 00:00:00+05:30  1536.900024  1550.000000  1524.000000
1545.000000
2025-12-12 00:00:00+05:30  1550.800049  1559.800049  1546.099976
1556.500000

                              Volume  Dividends  Stock Splits
Date
1996-01-01 00:00:00+05:30  104121369        0.0           0.0
1996-01-02 00:00:00+05:30  168743308        0.0           0.0
1996-01-03 00:00:00+05:30  209323879        0.0           0.0
1996-01-04 00:00:00+05:30  216900264        0.0           0.0
1996-01-05 00:00:00+05:30  166708467        0.0           0.0
```

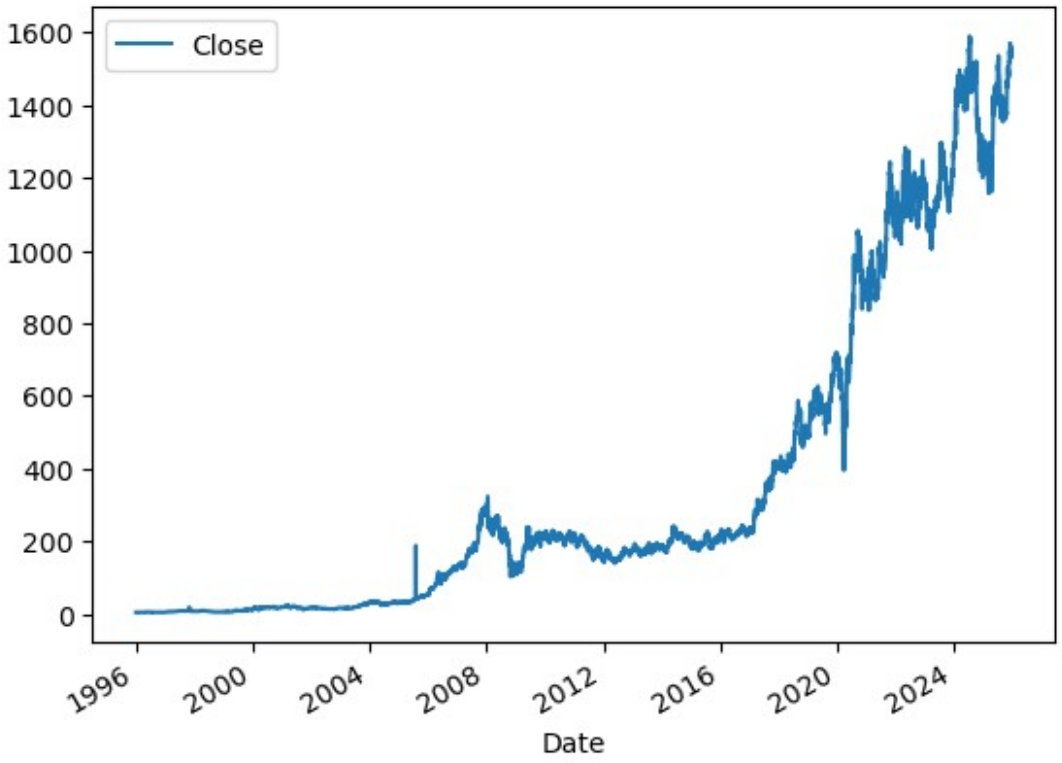
```
...                                 ...         ...           ...
2025-12-08 00:00:00+05:30   11301491        0.0           0.0
2025-12-09 00:00:00+05:30   11503971        0.0           0.0
2025-12-10 00:00:00+05:30    7991629        0.0           0.0
2025-12-11 00:00:00+05:30    4706197        0.0           0.0
2025-12-12 00:00:00+05:30    5105420        0.0           0.0

[7519 rows x 7 columns]
```

```
reliance.index
```

```
DatetimeIndex(['1996-01-01 00:00:00+05:30', '1996-01-02
00:00:00+05:30',
               '1996-01-03 00:00:00+05:30', '1996-01-04
00:00:00+05:30',
               '1996-01-05 00:00:00+05:30', '1996-01-08
00:00:00+05:30',
               '1996-01-09 00:00:00+05:30', '1996-01-10
00:00:00+05:30',
               '1996-01-11 00:00:00+05:30', '1996-01-12
00:00:00+05:30',
               ...
               '2025-12-01 00:00:00+05:30', '2025-12-02
00:00:00+05:30',
               '2025-12-03 00:00:00+05:30', '2025-12-04
00:00:00+05:30',
               '2025-12-05 00:00:00+05:30', '2025-12-08
00:00:00+05:30',
               '2025-12-09 00:00:00+05:30', '2025-12-10
00:00:00+05:30',
               '2025-12-11 00:00:00+05:30', '2025-12-12
00:00:00+05:30'],
              dtype='datetime64[ns, Asia/Kolkata]', name='Date',
length=7519, freq=None)
```

```
 # cleaning and visualizing our stock market
reliance.plot.line(y="Close",use_index=True)
```

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




```
del reliance["Dividends"]
del reliance["Stock Splits"]

#setting up our target for machine learning
reliance["tomorrow"]= reliance["Close"].shift(-1)

reliance
```

```
                                  Open         High          Low
Close  \
Date

1996-01-01 00:00:00+05:30     4.600219     4.624903     4.569925
4.617049
1996-01-02 00:00:00+05:30     4.605830     4.628270     4.547485
4.581145
1996-01-03 00:00:00+05:30     4.656318     4.868378     4.605829
4.615927
1996-01-04 00:00:00+05:30     4.572169     4.586755     4.511581
4.573291
1996-01-05 00:00:00+05:30     4.555339     4.555339     4.502605
4.541875
...                                ...          ...          ...
...
2025-12-08 00:00:00+05:30  1539.199951  1551.000000  1535.000000
1543.000000
```

```
2025-12-09 00:00:00+05:30  1540.300049  1542.500000  1520.099976
1529.400024
2025-12-10 00:00:00+05:30  1534.000000  1547.500000  1531.400024
1536.900024
2025-12-11 00:00:00+05:30  1536.900024  1550.000000  1524.000000
1545.000000
2025-12-12 00:00:00+05:30  1550.800049  1559.800049  1546.099976
1556.500000
```

```
                                Volume     tomorrow
Date
1996-01-01 00:00:00+05:30  104121369     4.581145
1996-01-02 00:00:00+05:30  168743308     4.615927
1996-01-03 00:00:00+05:30  209323879     4.573291
1996-01-04 00:00:00+05:30  216900264     4.541875
1996-01-05 00:00:00+05:30  166708467     4.360110
...                              ...          ...
2025-12-08 00:00:00+05:30   11301491  1529.400024
2025-12-09 00:00:00+05:30   11503971  1536.900024
2025-12-10 00:00:00+05:30    7991629  1545.000000
2025-12-11 00:00:00+05:30    4706197  1556.500000
2025-12-12 00:00:00+05:30    5105420          NaN

[7519 rows x 6 columns]
```

```
reliance["target"]= (reliance["tomorrow"]>
reliance["Close"]).astype(int)
```

```
reliance
```

```
                                  Open         High          Low
Close  \
Date

1996-01-01 00:00:00+05:30     4.600219     4.624903     4.569925
4.617049
1996-01-02 00:00:00+05:30     4.605830     4.628270     4.547485
4.581145
1996-01-03 00:00:00+05:30     4.656318     4.868378     4.605829
4.615927
1996-01-04 00:00:00+05:30     4.572169     4.586755     4.511581
4.573291
1996-01-05 00:00:00+05:30     4.555339     4.555339     4.502605
4.541875
...                                ...          ...          ...
...
2025-12-08 00:00:00+05:30  1539.199951  1551.000000  1535.000000
1543.000000
2025-12-09 00:00:00+05:30  1540.300049  1542.500000  1520.099976
1529.400024
```

```
2025-12-10 00:00:00+05:30  1534.000000  1547.500000  1531.400024
1536.900024
2025-12-11 00:00:00+05:30  1536.900024  1550.000000  1524.000000
1545.000000
2025-12-12 00:00:00+05:30  1550.800049  1559.800049  1546.099976
1556.500000

                                Volume     tomorrow  target
Date
1996-01-01 00:00:00+05:30  104121369     4.581145       0
1996-01-02 00:00:00+05:30  168743308     4.615927       1
1996-01-03 00:00:00+05:30  209323879     4.573291       0
1996-01-04 00:00:00+05:30  216900264     4.541875       0
1996-01-05 00:00:00+05:30  166708467     4.360110       0
...                              ...          ...     ...
2025-12-08 00:00:00+05:30   11301491  1529.400024       0
2025-12-09 00:00:00+05:30   11503971  1536.900024       1
2025-12-10 00:00:00+05:30    7991629  1545.000000       1
2025-12-11 00:00:00+05:30    4706197  1556.500000       1
2025-12-12 00:00:00+05:30    5105420          NaN       0

[7519 rows x 7 columns]

reliance= reliance.loc["2000-01-01":].copy()

reliance

                                 Open         High          Low
Close  \
Date

2000-01-03 00:00:00+05:30    12.497205    13.244407    12.497205
13.244407
2000-01-04 00:00:00+05:30    13.596961    14.304698    13.223360
14.304698
2000-01-05 00:00:00+05:30    13.504873    15.149242    13.504873
14.865095
2000-01-06 00:00:00+05:30    15.207126    15.822778    15.207126
15.488641
2000-01-07 00:00:00+05:30    15.522839    16.727833    15.417599
16.551556
...                                ...          ...          ...
...
2025-12-08 00:00:00+05:30  1539.199951  1551.000000  1535.000000
1543.000000
2025-12-09 00:00:00+05:30  1540.300049  1542.500000  1520.099976
1529.400024
2025-12-10 00:00:00+05:30  1534.000000  1547.500000  1531.400024
1536.900024
2025-12-11 00:00:00+05:30  1536.900024  1550.000000  1524.000000
```

```
1545.000000
2025-12-12 00:00:00+05:30  1550.800049  1559.800049  1546.099976
1556.500000
```

```
                                Volume     tomorrow  target
Date
2000-01-03 00:00:00+05:30   62409578    14.304698       1
2000-01-04 00:00:00+05:30  132872110    14.865095       1
2000-01-05 00:00:00+05:30  375789847    15.488641       1
2000-01-06 00:00:00+05:30  219621124    16.551556       1
2000-01-07 00:00:00+05:30  278281260    16.233208       0
...                              ...          ...     ...
2025-12-08 00:00:00+05:30   11301491  1529.400024       0
2025-12-09 00:00:00+05:30   11503971  1536.900024       1
2025-12-10 00:00:00+05:30    7991629  1545.000000       1
2025-12-11 00:00:00+05:30    4706197  1556.500000       1
2025-12-12 00:00:00+05:30    5105420          NaN       0

[6474 rows x 7 columns]
```

```
from sklearn.ensemble import RandomForestClassifier
model= RandomForestClassifier(n_estimators=100,min_samples_split=100,
random_state=1)
train = reliance.iloc[:-100]
test= reliance.iloc[-100:]
predictors= ["Close", "Open","Volume","High","Low"]
model.fit(train[predictors],train["target"])
```

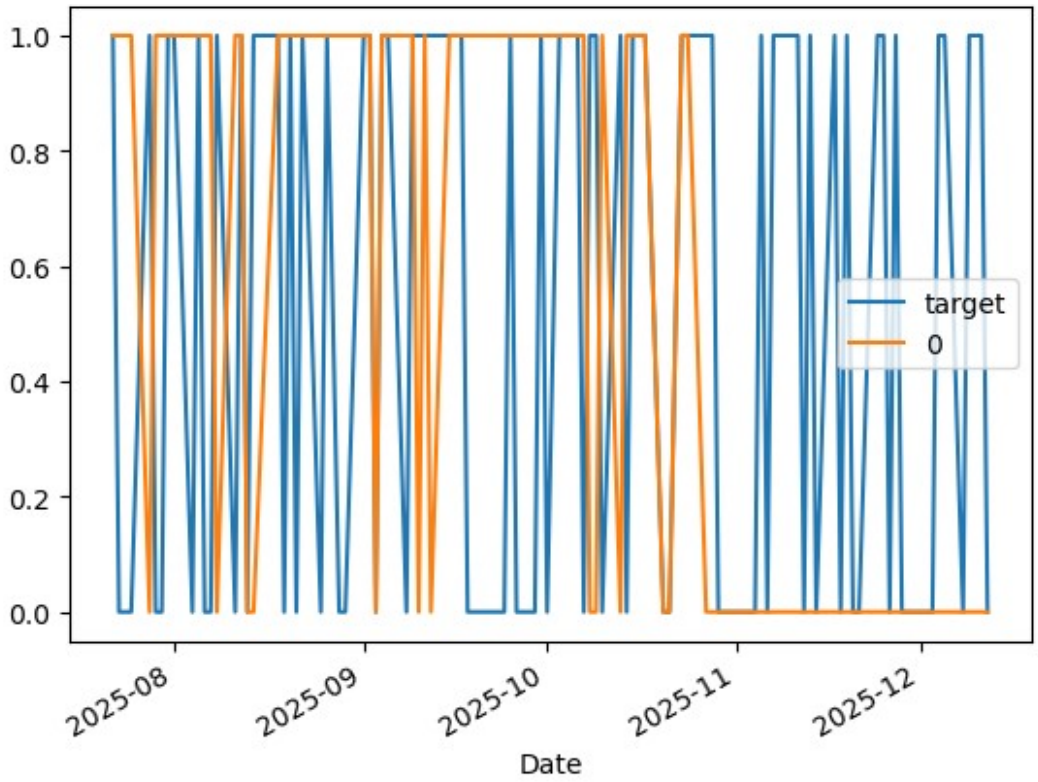
```
RandomForestClassifier(min_samples_split=100, random_state=1)
```

```
import pandas as pd
from sklearn.metrics import precision_score
preds= model.predict(test[predictors])
preds= pd.Series(preds, index=test.index)

precision_score(test["target"],preds)
```

```
0.5094339622641509
```

```
#now trying to make a better model
#as i also can make this decison by just waking up and just picking
what to do with the stock
combined= pd. concat([test["target"],preds],axis=1)

combined.plot()
```

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




```
def predict(train, test,predictors,model):
    model.fit(train[predictors],train["target"])
    preds= model.predict(test[predictors])
    preds=pd.Series(preds,index=test.index,name="predictions")
    combined=pd.concat([test["target"],preds],axis=1)
    return combined

def backtest(data, model,predictors,start=2500,step=250):
    all_predictions=[]
    for i in range (start,data.shape[0], step):
        train = data.iloc [0:i].copy()
        test= data.iloc[i:(i+step)].copy()
        predictions= predict(train,test,predictors,model)
        all_predictions.append(predictions)
    return pd.concat(all_predictions)



predictions= backtest(reliance,model[0],predictors)
predictions["predictions"].value_counts()
```

```
predictions
1    1989
0    1985
Name: count, dtype: int64
```

```
precision_score(predictions["target"],predictions["predictions"])
```

```
0.5082956259426847
```

```
predictions["target"].value_counts()/ predictions.shape[0]
```

```
target
1    0.507801
0    0.492199
Name: count, dtype: float64
```

```
#adding additional predictors
horizons= [2,5,60,250,1000]
new_predictors= []
for horizon in horizons:
    rolling_averages = reliance. rolling(horizon).mean()

    ratio_column = f"Close_Ratio_{horizon}"
    reliance [ratio_column]= reliance ["Close"] /
rolling_averages["Close"]

    trend_column = f"trend_{horizon}"
    reliance[trend_column]=reliance.shift(1).rolling(horizon).sum()
["target"]
    new_predictors+= [ratio_column,trend_column]
```

```
reliance
```

```
                                  Open         High          Low
Close  \
Date

2000-01-03 00:00:00+05:30    12.497205    13.244407    12.497205
13.244407
2000-01-04 00:00:00+05:30    13.596961    14.304698    13.223360
14.304698
2000-01-05 00:00:00+05:30    13.504873    15.149242    13.504873
14.865095
2000-01-06 00:00:00+05:30    15.207126    15.822778    15.207126
15.488641
2000-01-07 00:00:00+05:30    15.522839    16.727833    15.417599
16.551556
...                                ...          ...          ...
...
2025-12-08 00:00:00+05:30  1539.199951  1551.000000  1535.000000
1543.000000
2025-12-09 00:00:00+05:30  1540.300049  1542.500000  1520.099976
1529.400024
2025-12-10 00:00:00+05:30  1534.000000  1547.500000  1531.400024
1536.900024
```

```
2025-12-11 00:00:00+05:30  1536.900024  1550.000000  1524.000000
1545.000000
2025-12-12 00:00:00+05:30  1550.800049  1559.800049  1546.099976
1556.500000

                                Volume     tomorrow  target
Close_Ratio_2  \
Date

2000-01-03 00:00:00+05:30   62409578    14.304698       1
NaN
2000-01-04 00:00:00+05:30  132872110    14.865095       1
1.038487
2000-01-05 00:00:00+05:30  375789847    15.488641       1
1.019212
2000-01-06 00:00:00+05:30  219621124    16.551556       1
1.020543
2000-01-07 00:00:00+05:30  278281260    16.233208       0
1.033174
...                              ...          ...     ...             .
..
2025-12-08 00:00:00+05:30   11301491  1529.400024       0
1.000778
2025-12-09 00:00:00+05:30   11503971  1536.900024       1
0.995574
2025-12-10 00:00:00+05:30    7991629  1545.000000       1
1.002446
2025-12-11 00:00:00+05:30    4706197  1556.500000       1
1.002628
2025-12-12 00:00:00+05:30    5105420          NaN       0
1.003708

                           trend_2  Close_Ratio_5  trend_5
Close_Ratio_60  \
Date

2000-01-03 00:00:00+05:30      NaN            NaN      NaN
NaN
2000-01-04 00:00:00+05:30      NaN            NaN      NaN
NaN
2000-01-05 00:00:00+05:30      2.0            NaN      NaN
NaN
2000-01-06 00:00:00+05:30      2.0            NaN      NaN
NaN
2000-01-07 00:00:00+05:30      2.0       1.111523      NaN
NaN
...                            ...            ...      ...
...
2025-12-08 00:00:00+05:30      2.0       1.001389      2.0
1.057389
```

```
2025-12-09 00:00:00+05:30      1.0       0.994745      2.0
1.046463
2025-12-10 00:00:00+05:30      1.0       0.999870      3.0
1.049947
2025-12-11 00:00:00+05:30      2.0       1.003912      4.0
1.053804
2025-12-12 00:00:00+05:30      2.0       1.009299      4.0
1.059929

                           trend_60  Close_Ratio_250  trend_250  \
Date
2000-01-03 00:00:00+05:30       NaN              NaN        NaN
2000-01-04 00:00:00+05:30       NaN              NaN        NaN
2000-01-05 00:00:00+05:30       NaN              NaN        NaN
2000-01-06 00:00:00+05:30       NaN              NaN        NaN
2000-01-07 00:00:00+05:30       NaN              NaN        NaN
...                              ...              ...        ...
2025-12-08 00:00:00+05:30      31.0         1.132057      126.0
2025-12-09 00:00:00+05:30      30.0         1.121291      126.0
2025-12-10 00:00:00+05:30      30.0         1.125941      127.0
2025-12-11 00:00:00+05:30      30.0         1.130974      128.0
2025-12-12 00:00:00+05:30      30.0         1.138396      129.0

                           Close_Ratio_1000  trend_1000
Date
2000-01-03 00:00:00+05:30               NaN         NaN
2000-01-04 00:00:00+05:30               NaN         NaN
2000-01-05 00:00:00+05:30               NaN         NaN
2000-01-06 00:00:00+05:30               NaN         NaN
2000-01-07 00:00:00+05:30               NaN         NaN
...                                     ...         ...
2025-12-08 00:00:00+05:30          1.218105       513.0
2025-12-09 00:00:00+05:30          1.206993       512.0
2025-12-10 00:00:00+05:30          1.212491       513.0
2025-12-11 00:00:00+05:30          1.218464       513.0
2025-12-12 00:00:00+05:30          1.227087       514.0

[6474 rows x 17 columns]
```

```
reliance= reliance.dropna()
```

```
reliance
```

```
                                  Open         High          Low
Close  \
Date

2003-11-04 00:00:00+05:30    29.521908    29.751216    28.642895
28.898661
2003-11-05 00:00:00+05:30    28.634068    29.051527    28.251890
```

```
28.728144
2003-11-06 00:00:00+05:30    28.928055    29.486623    28.757544
28.978029
2003-11-07 00:00:00+05:30    29.019190    29.216158    28.204853
28.348906
2003-11-10 00:00:00+05:30    29.986399    29.986399    27.699200
28.395945
...                                ...          ...          ...
...
2025-12-05 00:00:00+05:30  1530.400024  1545.800049  1520.599976
1540.599976
2025-12-08 00:00:00+05:30  1539.199951  1551.000000  1535.000000
1543.000000
2025-12-09 00:00:00+05:30  1540.300049  1542.500000  1520.099976
1529.400024
2025-12-10 00:00:00+05:30  1534.000000  1547.500000  1531.400024
1536.900024
2025-12-11 00:00:00+05:30  1536.900024  1550.000000  1524.000000
1545.000000

                              Volume     tomorrow  target
Close_Ratio_2  \
Date

2003-11-04 00:00:00+05:30  110392949    28.728144       0
0.993331
2003-11-05 00:00:00+05:30   91124985    28.978029       1
0.997041
2003-11-06 00:00:00+05:30  134300335    28.348906       0
1.004330
2003-11-07 00:00:00+05:30   73031669    28.395945       1
0.989026
2003-11-10 00:00:00+05:30   55228187    28.260706       0
1.000829
...                              ...          ...     ...             .
..
2025-12-05 00:00:00+05:30   10183266  1543.000000       1
1.001625
2025-12-08 00:00:00+05:30   11301491  1529.400024       0
1.000778
2025-12-09 00:00:00+05:30   11503971  1536.900024       1
0.995574
2025-12-10 00:00:00+05:30    7991629  1545.000000       1
1.002446
2025-12-11 00:00:00+05:30    4706197  1556.500000       1
1.002628

                            trend_2  Close_Ratio_5  trend_5
Close_Ratio_60  \
Date
```

```
2003-11-04 00:00:00+05:30      1.0       1.013193      4.0
1.142585
2003-11-05 00:00:00+05:30      0.0       1.000532      3.0
1.130253
2003-11-06 00:00:00+05:30      1.0       1.002747      3.0
1.134214
2003-11-07 00:00:00+05:30      1.0       0.982696      2.0
1.104356
2003-11-10 00:00:00+05:30      1.0       0.990443      2.0
1.101073
...                            ...            ...      ...
...
2025-12-05 00:00:00+05:30      1.0       0.996842      1.0
1.057673
2025-12-08 00:00:00+05:30      2.0       1.001389      2.0
1.057389
2025-12-09 00:00:00+05:30      1.0       0.994745      2.0
1.046463
2025-12-10 00:00:00+05:30      1.0       0.999870      3.0
1.049947
2025-12-11 00:00:00+05:30      2.0       1.003912      4.0
1.053804

                           trend_60  Close_Ratio_250  trend_250  \
Date
2003-11-04 00:00:00+05:30      36.0         1.515055      138.0
2003-11-05 00:00:00+05:30      35.0         1.501943      138.0
2003-11-06 00:00:00+05:30      36.0         1.510771      138.0
2003-11-07 00:00:00+05:30      36.0         1.474065      137.0
2003-11-10 00:00:00+05:30      36.0         1.472790      137.0
...                             ...              ...        ...
2025-12-05 00:00:00+05:30      31.0         1.131081      125.0
2025-12-08 00:00:00+05:30      31.0         1.132057      126.0
2025-12-09 00:00:00+05:30      30.0         1.121291      126.0
2025-12-10 00:00:00+05:30      30.0         1.125941      127.0
2025-12-11 00:00:00+05:30      30.0         1.130974      128.0

                           Close_Ratio_1000  trend_1000
Date
2003-11-04 00:00:00+05:30          1.645597       497.0
2003-11-05 00:00:00+05:30          1.634545       496.0
2003-11-06 00:00:00+05:30          1.647440       496.0
2003-11-07 00:00:00+05:30          1.610496       495.0
2003-11-10 00:00:00+05:30          1.612083       495.0
...                                     ...         ...
2025-12-05 00:00:00+05:30          1.216665       512.0
2025-12-08 00:00:00+05:30          1.218105       513.0
2025-12-09 00:00:00+05:30          1.206993       512.0
2025-12-10 00:00:00+05:30          1.212491       513.0
```

```
2025-12-11 00:00:00+05:30          1.218464       513.0

[5473 rows x 17 columns]
```

```
#improving our model
mode=RandomForestClassifier(n_estimators=200,min_samples_split=50,rand
om_state=1)

predict= backtest(reliance,model,new_predictors)

def predict(train, test, predictors, model):
    # Fit the model
    model.fit(train[predictors], train["target"])

    # Get probability predictions
    preds = model.predict_proba(test[predictors])[:, 1]

    # Convert probabilities to binary predictions
    # Create a numpy array from preds
    preds = np.array(preds)
    # Use boolean indexing to assign values
    preds[preds >= 0.6] = 1
    preds[preds < 0.6] = 0

    # Convert to pandas Series
    preds = pd.Series(preds, index=test.index, name="predictions")

    # Combine with actual targets
    combined = pd.concat([test["target"], preds], axis=1)
    return combined

predictions = backtest (reliance, model,new_predictors)
predictions["predictions"].value_counts()
```

```
predictions
0.0    2853
1.0     120
Name: count, dtype: int64
```

```
precision_score(predictions["target"],predictions["predictions"])
```

```
0.6083333333333333
```

```
predictors = ["Close", "Open", "High", "Low", "Volume"]

# Train without last row
train_X = reliance[predictors].iloc[:-1]
train_y = reliance["target"].iloc[:-1]
model.fit(train_X, train_y)

# Predict using last row (today)
```

```
latest_features = reliance[predictors].iloc[[-1]]

# Probability of tomorrow going UP
prob_up = model.predict_proba(latest_features)[0][1]

# Binary classification
prediction = int(prob_up >= 0.5)

print("Probability that price will go UP tomorrow:", prob_up)
print("Predicted Direction (1 = UP, 0 = DOWN):", prediction)
```

```
Probability that price will go UP tomorrow: 0.4260101802403915
Predicted Direction (1 = UP, 0 = DOWN): 0
```

```
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

y_prob = model.predict_proba(test[predictors])[:, 1]

fpr, tpr, _ = roc_curve(y_test, y_prob)
auc = roc_auc_score(y_test, y_prob)

plt.plot(fpr, tpr, label=f"AUC = {auc:.2f}")
plt.plot([0,1], [0,1], linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve – Reliance Industries")
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------------
-----
NotFittedError                                Traceback (most recent call
last)
Cell In[21], line 4
      1 from sklearn.metrics import roc_curve, roc_auc_score
      2 import matplotlib.pyplot as plt
----> 4 y_prob = model.predict_proba(test[predictors])[:, 1]
      6 fpr, tpr, _ = roc_curve(y_test, y_prob)
      7 auc = roc_auc_score(y_test, y_prob)

File ~\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:943, in
ForestClassifier.predict_proba(self, X)
    921 def predict_proba(self, X):
    922     """
    923     Predict class probabilities for X.
    924
   (...)    941         classes corresponds to that in the
attribute :term:`classes_`.
    942     """
--> 943     check_is_fitted(self)
```

```
    944     # Check data
    945     X = self._validate_X_predict(X)

File ~\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1754,
in check_is_fitted(estimator, attributes, msg, all_or_any)
   1751     return
   1753 if not _is_fitted(estimator, attributes, all_or_any):
-> 1754     raise NotFittedError(msg % {"name":
type(estimator).__name__})

NotFittedError: This RandomForestClassifier instance is not fitted
yet. Call 'fit' with appropriate arguments before using this
estimator.
```