In 2 hours: Machine Learning Code to predict returns (prediction based testing)

Date	@July 1, 2023
Tag	Nocode MVP
Description	The toolkit to help you build a machine learning architecture to predict returns and variances
Year	2023
Image	
Category	✍️ Blog Post

```
import yfinance as yf
import numpy as np
from sklearn.linear_model import LogisticRegression

df = yf.download('RELIANCE.NS', start='2022-01-01')
df
```

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

	Open	High	Low	Close	Adj Close	Volume
Date						
2022-01-03	2365.000000	2407.949951	2363.550049	2403.850098	2396.634521	2502073
2022-01-04	2415.899902	2461.000000	2404.000000	2458.100098	2450.721680	5006225
2022-01-05	2462.000000	2477.000000	2432.949951	2469.600098	2462.187012	5373618
2022-01-06	2451.199951	2454.000000	2409.000000	2416.500000	2409.246338	6667483
2022-01-07	2430.949951	2458.050049	2411.550049	2436.000000	2428.687988	6051239
...	...	...	...	...	...	...
2023-07-05	2609.000000	2609.000000	2575.800049	2584.500000	2584.500000	4729479
2023-07-06	2576.050049	2644.449951	2576.050049	2638.750000	2638.750000	8822948
2023-07-07	2635.000000	2664.949951	2628.000000	2633.600098	2633.600098	6172684
2023-07-10	2688.899902	2756.000000	2675.000000	2735.050049	2735.050049	15340262
2023-07-12	2766.300049	2802.000000	2761.649902	2767.750000	2767.750000	8633349

```
df['ret'] = df.Close.pct_change()
df
```

```
def lagit(df, lags):
    for i in range(1, lags + 1):
        df['Lag_'+str(i)] = df['ret'].shift(i)

    return ['Lag_'+str(i) for i in range(1, lags+1)]
```

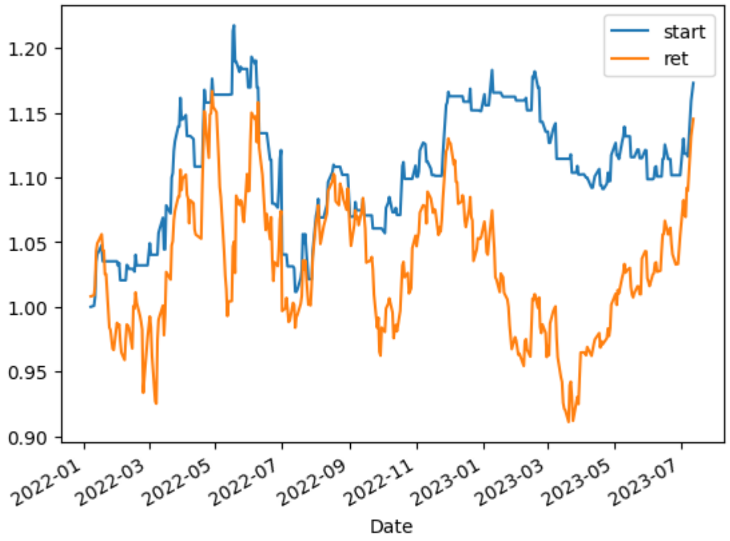
Open	High	Low	Close	Adj Close	Volume	ret	Lag_1
Date							
2022-01-03	2365.000000	2407.949951	2363.550049	2403.850098	2396.634521	2502073	NaN
2022-01-04	2415.899902	2461.000000	2404.000000	2458.100098	2450.721680	5006225	0.022568
2022-01-05	2462.000000	2477.000000	2432.949951	2469.600098	2462.187012	5373618	0.004678
2022-01-06	2451.199951	2454.000000	2409.000000	2416.500000	2409.246338	6667483	-0.02150
2022-01-07	2430.949951	2458.050049	2411.550049	2436.000000	2428.687988	6051239	0.00807
...	...	...	...	...	...	...	...

2023-07-05	2609.000000	2609.000000	2575.800049	2584.500000	2584.500000	4729479	-0.00164
2023-07-06	2576.050049	2644.449951	2576.050049	2638.750000	2638.750000	8822948	0.02099
2023-07-07	2635.000000	2664.949951	2628.000000	2633.600098	2633.600098	6172684	-0.00195
2023-07-10	2688.899902	2756.000000	2675.000000	2735.050049	2735.050049	15340262	0.03852
2023-07-12	2766.300049	2802.000000	2761.649902	2767.750000	2767.750000	8633349	0.01195

```
model = LogisticRegression(class_weight='balanced')
model.fit(x,y)
```

Open	High	Low	Close	Adj Close	Volume	ret	Lag_1
Date							
2022-01-07	2430.949951	2458.050049	2411.550049	2436.000000	2428.687988	6051239	0.00807
2022-01-10	2452.000000	2457.000000	2416.050049	2438.000000	2430.681885	4267365	0.00082
2022-01-11	2436.000000	2474.949951	2435.000000	2455.550049	2448.179199	7478681	0.00719
2022-01-12	2471.300049	2524.949951	2465.000000	2521.100098	2513.532471	6830402	0.02669
2022-01-13	2521.250000	2541.000000	2508.399902	2535.300049	2527.689941	5471871	0.00563
...	...	...	...	...	...	...	...
2023-07-05	2609.000000	2609.000000	2575.800049	2584.500000	2584.500000	4729479	-0.00164
2023-07-06	2576.050049	2644.449951	2576.050049	2638.750000	2638.750000	8822948	0.02099
2023-07-07	2635.000000	2664.949951	2628.000000	2633.600098	2633.600098	6172684	-0.00195
2023-07-10	2688.899902	2756.000000	2675.000000	2735.050049	2735.050049	15340262	0.03852
2023-07-12	2766.300049	2802.000000	2761.649902	2767.750000	2767.750000	8633349	0.01195

(df[['start','ret']] + 1).cumprod().plot()



```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, shuffle=False)
x_test
```