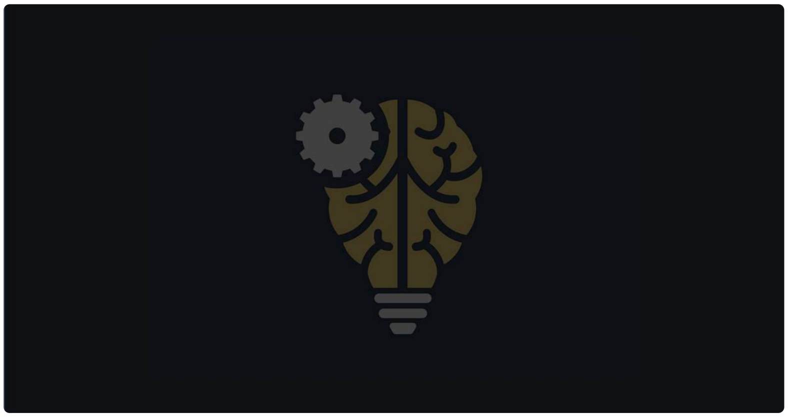
QUANT INSTI® GO ALGO Pioneer Institute in Algo Trading (https://www.quantinsti.com)

Trading Using Machine Learning In Python – SVM (Support Vector Machine) ()

On August 7, 2017 0 Comments (https://www.quantinsti.com/blog/trading-using-machine-learning-python-svm-support-vector-machine#disqus_thread)

84 23 141 248 SHARES



By Varun Divakar (https://www.linkedin.com/in/varun-divakar-b862a667/)

In this blog, I will show you how to implement a machine learning based trading strategy (https://quantra.quantinsti.com/course/introduction-to-machine-learning-for-trading) using the regime predictions made in the previous blog (https://www.quantinsti.com/blog/trading-using-machine-learning-python-part-2/). Do read it, there is a special discount for you at the end of this.

There is one thing that you should keep in mind before you read this blog though: The algorithm is just for demonstration and should not be used for real trading without proper optimization.

Let me begin by explaining the agenda of the blog:

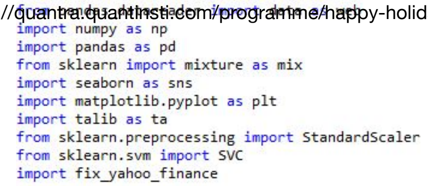
1. Create an unsupervised ML (machine learning) algorithm to predict the regimes.
2. Plot these regimes to visualize them.
3. Train a Support Vector Classifier algorithm with the regime as one of the features.
4. Use this Support Vector Classifier algorithm to predict the current day's trend at the Opening of the market.
5. Visualize the performance of this strategy on the test data.
6. Downloadable code for your benefit

> *Trading Using Machine Learning In Python – SVM (Support Vector Machine)*
>
> CLICK TO TWEET

(https://twitter.com/share?text=Trading+Using+Machine+Learning+In+Python+-+SVM+%28Support+Vector+Machine%29+by+%40QuantInsti&url=https://www.quantinsti.com/blog/trading-using-machine-learning-python-svm-support-vector-machine)

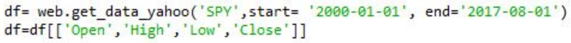
Import the Libraries and the Data:

First, I imported the necessary libraries. Please note that I have imported fix_yahoo_finance package, so I am able to pull data from yahoo. If you do not have this package, I suggest you install (https://pypi.python.org/pypi/fix-yahoo-finance)it first or change your data source to google.



```
from pandas_datareader import data as web
import numpy as np
import pandas as pd
from sklearn import mixture as mix
import seaborn as sns
import matplotlib.pyplot as plt
import talib as ta
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
import fix_yahoo_finance
```

Next, I pulled the data of the same quote, ‘SPY’, which we used in the previous blog and saved it as a dataframe df. I chose the time period for this data to be from the year 2000.

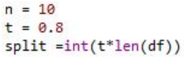


```
df= web.get_data_yahoo('SPY',start= '2000-01-01', end='2017-08-01')
df=df[['Open','High','Low','Close']]
```

After this, I created indicators that can be used as features for training the algorithm.

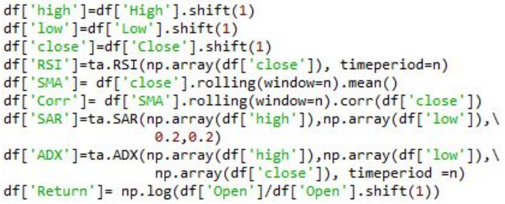
But, before doing that I decided on the look back time period for these indicators. I chose a look back period of 10 days. You may try any other number that suits you. I chose 10 to check for the past 2 weeks of trading data and to avoid noise inherent in smaller look back periods.

Apart from the look back period let us also decide the test train split of the data. I prefer to give 80% data for training and remaining 20% data for testing. You can change this as per your need.



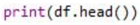
```
n = 10
t = 0.8
split =int(t*len(df))
```

Next, I shifted the High, Low and Close columns by 1, to access only the past data. After this, I created various technical indicators such as, RSI, SMA, ADX, Correlation, Parabolic SAR, and the Return of the past 1- day on an Open to Open basis.



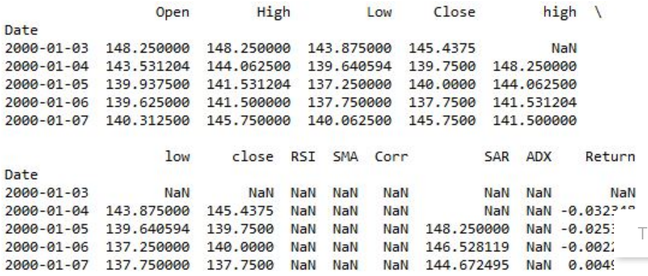
```
df['high']=df['High'].shift(1)
df['low']=df['Low'].shift(1)
df['close']=df['Close'].shift(1)
df['RSI']=ta.RSI(np.array(df['close']), timeperiod=n)
df['SMA']= df['close'].rolling(window=n).mean()
df['Corr']= df['SMA'].rolling(window=n).corr(df['close'])
df['SAR']=ta.SAR(np.array(df['high']),np.array(df['low']),\
                  0.2,0.2)
df['ADX']=ta.ADX(np.array(df['high']),np.array(df['low']),\
                  np.array(df['close']), timeperiod =n)
df['Return']= np.log(df['Open']/df['Open'].shift(1))
```

Next, I printed the data frame.



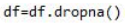
```
print(df.head())
```

And it looked like this:



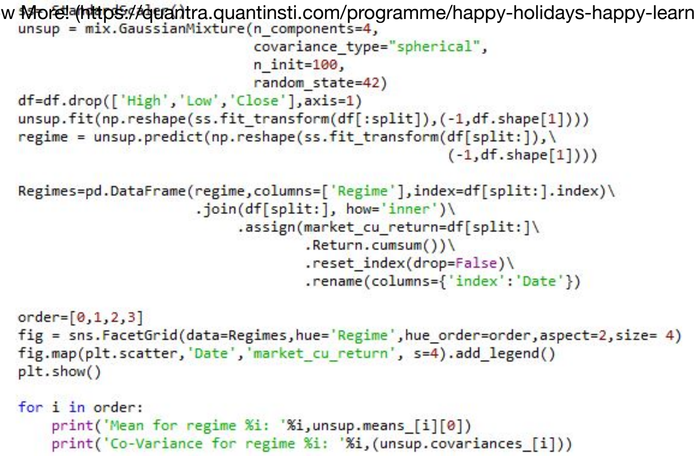
```
                  Open        High         Low     Close        high  \
Date
2000-01-03  148.250000  148.250000  143.875000  145.4375         NaN
2000-01-04  143.531204  144.062500  139.640594  139.7500  148.250000
2000-01-05  139.937500  141.531204  137.250000  140.0000  144.062500
2000-01-06  139.625000  141.500000  137.750000  137.7500  141.531204
2000-01-07  140.312500  145.750000  140.062500  145.7500  141.500000

                   low     close  RSI  SMA  Corr         SAR  ADX    Return
Date
2000-01-03         NaN       NaN  NaN  NaN   NaN         NaN  NaN       NaN
2000-01-04  143.875000  145.4375  NaN  NaN   NaN         NaN  NaN -0.0323
2000-01-05  139.640594  139.7500  NaN  NaN   NaN  148.250000  NaN -0.0251
2000-01-06  137.250000  140.0000  NaN  NaN   NaN  146.528119  NaN -0.0022
2000-01-07  137.750000  137.7500  NaN  NaN   NaN  144.672495  NaN  0.004
```

As you can see, there are many NaN values. We need to either impute them or drop them. If you are new to the machine learning and wa
learn.org/stable/modules/generated/sklearn.preprocessing.Imputer.html). I dropped the NaN values in this algorithm.



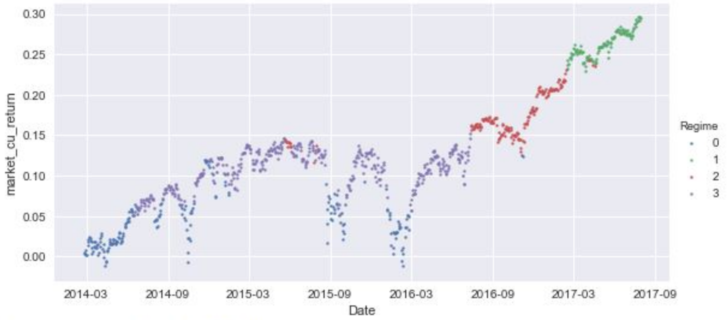
```
df=df.dropna()
```

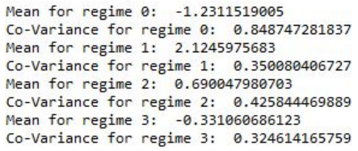
In the next part of the code, I instantiated a StandardScaler function and created an unsupervised learning algorithm to make the regime pr
be going into these details again.



```
ss= StandardScaler()
unsup = mix.GaussianMixture(n_components=4,
                             covariance_type="spherical",
                             n_init=100,
                             random_state=42)
df=df.drop(['High','Low','Close'],axis=1)
unsup.fit(np.reshape(ss.fit_transform(df[:split]),(-1,df.shape[1])))
regime = unsup.predict(np.reshape(ss.fit_transform(df[split:]),\
                                                   (-1,df.shape[1])))

Regimes=pd.DataFrame(regime,columns=['Regime'],index=df[split:].index)\
                     .join(df[split:], how='inner')\
                          .assign(market_cu_return=df[split:]\
                                 .Return.cumsum())\
                                 .reset_index(drop=False)\
                                 .rename(columns={'index':'Date'})

order=[0,1,2,3]
fig = sns.FacetGrid(data=Regimes,hue='Regime',hue_order=order,aspect=2,size= 4)
fig.map(plt.scatter,'Date','market_cu_return', s=4).add_legend()
plt.show()

for i in order:
    print('Mean for regime %i: '%i,unsup.means_[i][0])
    print('Co-Variance for regime %i: '%i,(unsup.covariances_[i]))
```

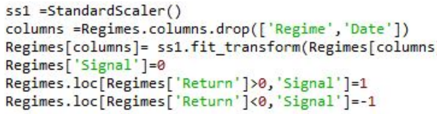
Towards the end of the last blog, I printed the Mean and Covariance values for all the regimes and plotted the regimes. The new output with indicators as feature set would look like this:





```
Mean for regime 0:  -1.2311519005
Co-Variance for regime 0:  0.848747281837
Mean for regime 1:  2.1245975683
Co-Variance for regime 1:  0.350080406727
Mean for regime 2:  0.690047980703
Co-Variance for regime 2:  0.425844469889
Mean for regime 3:  -0.331060686123
Co-Variance for regime 3:  0.324614165759
```

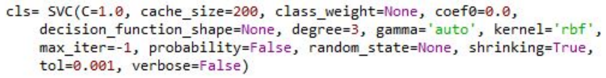
Next, I scaled the Regimes data frame, excluding the Date and Regimes columns, created in the earlier piece of code and saved it back in the same columns. By doing so, I will not be losing any features but the data will be scaled and ready for training the support vector classifier algorithm. Next, I created a signal column which would act as the prediction values. The algorithm would train on the features' set to predict this signal.



```
ss1 =StandardScaler()
columns =Regimes.columns.drop(['Regime','Date'])
Regimes[columns]= ss1.fit_transform(Regimes[columns])
Regimes['Signal']=0
Regimes.loc[Regimes['Return']>0,'Signal']=1
Regimes.loc[Regimes['Return']<0,'Signal']=-1
```

Next, I instantiated a support vector classifier. For this, I used the same SVC (http://scikit-learn.org/stable/modules/generated/sklearn.svı optimized this support vector classifier for best hyper parameters. In the machine learning course (https://quantra.quantinsti.com/cours have extensively discussed how to use hyper parameters and optimize the algorithm to predict the daily Highs and Lows, in turn the volatilit

Coming back to the blog, the code for support vector classifier is as below:

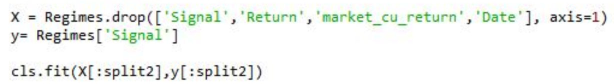


```
cls= SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

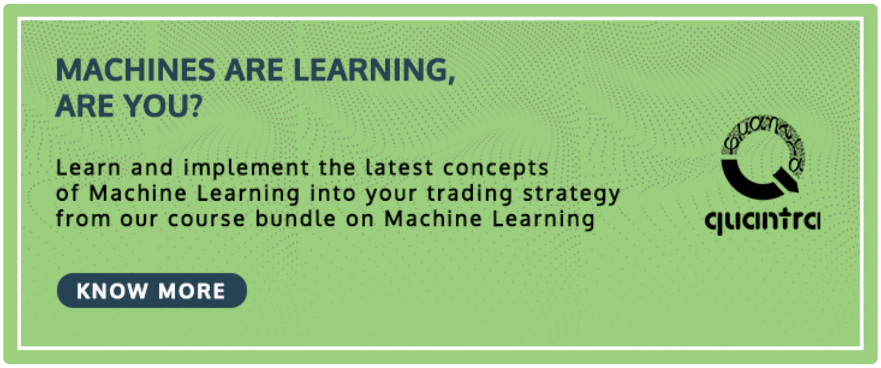
Next, I split the test data of the unsupervised regime algorithm into train and test data. We use this new train data to train our support ve columns that are not a part of the feature set:

'Signal','Return','market_cu_return','Date'

Then I fit the X and y data sets to the algorithm to train it on.

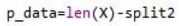


```
split2= int([illegible]len(Regimes))

X = Regimes.drop(['Signal','Return','market_cu_return','Date'], axis=1)
y= Regimes['Signal']

cls.fit(X[:split2],y[:split2])
```



Next, I calculated the test set size and indexed the predictions accordingly to the data frame df.

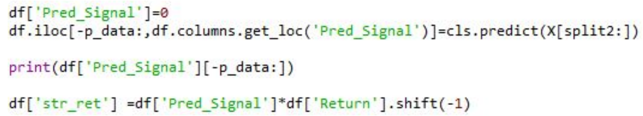
The reason for doing this is that the original return values of 'SPY' are stored in df, while those in Regimes is scaled hence, won't be useful for taking a cumulative sum to check for the performance.



```
p_data=len(X)-split2
```

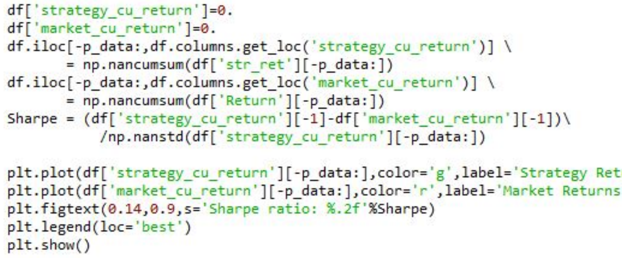
Next, I saved the predictions made by the SVC in a column named Pred_Signal.

Then, based on these signals I calculated the returns of the strategy by multiplying signal at the beginning of the day with the return at the opening (because our returns are from open to open) of the next day.



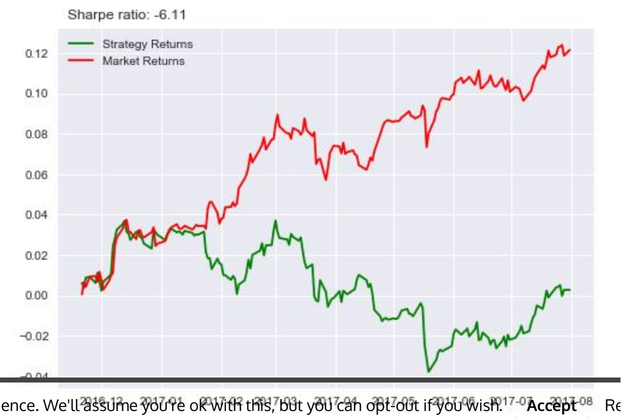
```
df['Pred_Signal']=0
df.iloc[-p_data:,df.columns.get_loc('Pred_Signal')]=cls.predict(X[split2:])

print(df['Pred_Signal'][-p_data:])

df['str_ret'] =df['Pred_Signal']*df['Return'].shift(-1)
```

Finally, I calculated the cumulative strategy returns and the cumulative market returns and saved them in df. Then, I calculated the sharpe ratio to measure the performance. To get a clear understanding of this metric I plotted the performance to measure it.



```
df['strategy_cu_return']=0.
df['market_cu_return']=0.
df.iloc[-p_data:,df.columns.get_loc('strategy_cu_return')] \
       = np.nancumsum(df['str_ret'][-p_data:])
df.iloc[-p_data:,df.columns.get_loc('market_cu_return')] \
       = np.nancumsum(df['Return'][-p_data:])
Sharpe = (df['strategy_cu_return'][-1]-df['market_cu_return'][-1])\
           /np.nanstd(df['strategy_cu_return'][-p_data:])

plt.plot(df['strategy_cu_return'][-p_data:],color='g',label='Strategy Retu
plt.plot(df['market_cu_return'][-p_data:],color='r',label='Market Returns'
plt.figtext(0.14,0.9,s='Sharpe ratio: %.2f'%Sharpe)
plt.legend(loc='best')
plt.show()
```

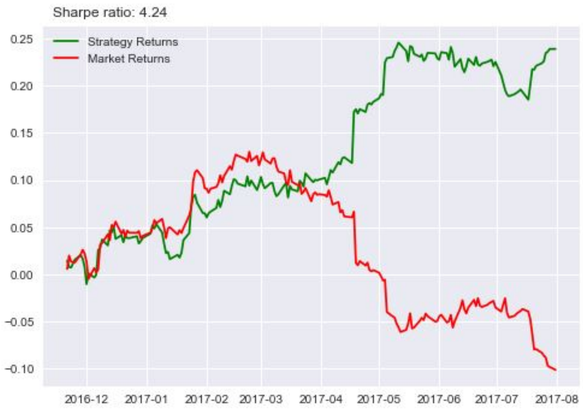
The final result looks like this.



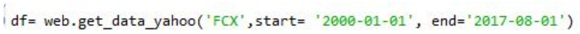
After so much of code and effort, if the end result looks like this, then someone with no machine learning back ground would say that it is not worth it. I would agree for now. But, look at this line of code:

```
df= web.get_data_yahoo('IBM',start= '2000-01-01', end='2017-08-01')
```

I just changed the data from SPY to IBM. Then the result looks like this:

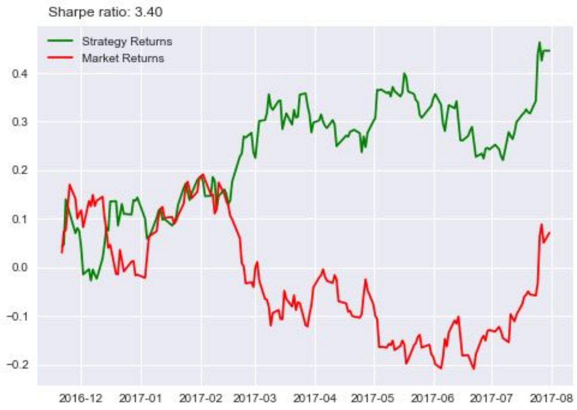


I know what you are thinking: I am just fitting the data to get the results. Which is not entirely wrong. I will show you another stock then you decide.



```
df= web.get_data_yahoo('FCX',start= '2000-01-01', end='2017-08-01')
```

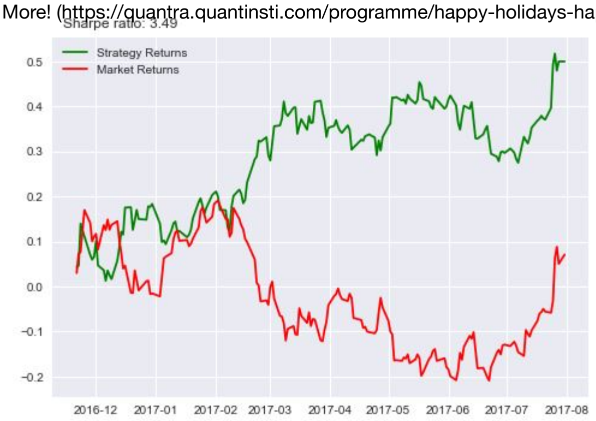
I changed the stock to Freeport-McMoRan Inc and the result looks like this:



You can further change it to GE or something else and check for yourself. This strategy works on some stocks but doesn't work on others, which is the case with most quant strategies. There are a few reasons why the algorithm did work consistently and I will list some of them here.

1. No autocorrelation of returns
2. No Support Vector hyper parameter optimization
3. No error propagation
4. No feature selection

We have not checked for autocorrelation of the returns, which would have increased the predictability of the algorithm. Try that on your ov set. The result would look like this:



Although the improvement from 3.4 to 3.49 is not much, it is still a good feature to have.

Please note that the code will best run with Python 2.7

Next Step

Learn the application of Machine Learning in Forex markets. Click here (https://www.quantinsti.com/blog/machine-learning-application-forex-markets-working-models/) to know how to use start with historical data (stock price/forex data) and add indicators to build a model in R/Python/Java. Then select the right Machine learning algorithm to make the predictions.

Update

We have noticed that some users are facing challenges while downloading the market data from Yahoo and Google Finance platforms. In case you are looking for an alternative source for market data, you can use Quandl (https://www.quandl.com/) for the same.

Download Data Files

- Python_3

Login to DOWNLOAD these files for FREE!

Existing Users Log In

USERNAME

PASSWORD

□ **REMEMBER ME**

New User? Click here to register (https://www.quantinsti.com/user-re

Search for posts

Search

GET NEW BLOG UPDATES

Enter your email address:

Subscribe

CATEGORIES

Career Advice (https://www.quantinsti.com/category/career-advice) (20)

Case Studies (https://www.quantinsti.com/category/case-studies) (14)

Downloadables (https://www.quantinsti.com/category/downloadables) (68)

Getting Started (https://www.quantinsti.com/category/quantitative-trading-getting-started) (160)

Guest Posts (https://www.quantinsti.com/category/guest-posts) (3)

News (https://www.quantinsti.com/category/news) (68)

- Events (https://www.quantinsti.com/category/news/events-news) (43)

Press Releases (https://www.quantinsti.com/category/press-releases) (4)

Programming and Trading Tools (https://www.quantinsti.com/category/programming-and-trading-tools) (144)

- Other Languages (https://www.quantinsti.com/category/programming-and-trading-tools/other-languages) (11)
- Python (https://www.quantinsti.com/category/programming-and-trading-tools/python) (85)
- R Programming (https://www.quantinsti.com/category/programming-and-trading-tools/r-programming) (40)
- Trading Platforms (https://www.quantinsti.com/category/programming-and-trading-tools/trading-platforms) (7)

Project Work EPAT (https://www.quantinsti.com/category/project-work-epat) (18)

Trading Strategies (https://www.quantinsti.com/category/quantitative-trading-strategies) (102)

Webinars (https://www.quantinsti.com/category/events) (36)

- Previous Webinars (https://www.quantinsti.com/category/events/previous) (36)

Talk to us ...

Q Hi! I am Quantbot. Do you need any assistance?

Today at 4:44 PM

Yes No

Quick Links

About Us (https://www.quantinsti.com/about-us/)
EPAT™ (https://www.quantinsti.com/epat/)
Alumni (https://www.quantinsti.com/alumni/)
Testimonials (https://www.quantinsti.com/testimonials/)
Faculty (https://www.quantinsti.com/faculty/)
Careers (https://www.quantinsti.com/careers/)
Quantra® (https://quantra.quantinsti.com/)
Placement Cell (https://www.quantinsti.com/placement-cell/)

Powered by IntelliTicks

BTRM (https://www.quantinsti.com/btrm)

Contact Us (https://www.quantinsti.com/contact-us/)
Disclaimer Policy (https://www.quantinsti.com/disclaimer-policy/)
Privacy Policy (https://www.quantinsti.com/privacy-policy/)
Terms & Conditions (https://www.quantinsti.com/terms-conditions/)
Pay Your Fees (https://www.quantinsti.com/payment/)
Student Login (https://accounts.quantinsti.com/account/)

India
QuantInsti Quantitative Learning Pvt Ltd
A-309, Boomerang,
Chandivali Farm Road, Powai,
Mumbai – 400 072
Toll Free: 1800-266-5401
Phone: +91-22-61691400

Singapore
30 Cecil Street, #19-08,
Prudential Tower,
Singapore – 049712
Phone: +65-9057-8301

Connect with us...

(https://www.facebook.com/quantinsti) (https://twitter.com/quantinsti/) (https://www.linkedin.com/company/quantinsti) (https://plus.google.com/110772715968756646442/) (https://www.youtu

(http://quantocracy.com/)Show us some love on Quantocracy.

Talk to us ...

Q Hi! I am Quantbot. Do you need any assistance?

Today at 4:44 PM

Yes No

Powered by IntelliTicks