Statistical Arbitrage Model Using Machine Learning

**Statistical Arbitrage**: For a family of stocks, generally belonging to the same sector or industry, there exists a correlation between prices of each of the stocks. There, though, exist anomalous times when for a small period of time, the correlation is broken. But the market self corrects in some time and the correlation is re-established. During this small window of time when correlation is anomalous, there exists a money-making opportunity for quantitative traders.

I developed Machine Learning Algorithm to predict statistical arbitrage opportunities in NSE based on the 2016 data. Test this algorithm on 2017 data.

## ****Steps we need to follow to build a model:****

1. Pre-Processing
2. Time series Analysis
3. Classification
4. Prediction

Libraries to be imported for this analysis:-

import pandas as pd  
import numpy as npfrom datetime import datetime  
#to plot within notebook  
import matplotlib.pyplot as plt  
from matplotlib import style  
style.use('ggplot')  
##%matplotlib inline#for normalizing data  
from sklearn.preprocessing import MinMaxScaler  
scaler = MinMaxScaler(feature\_range=(0, 1))  
from sklearn.feature\_extraction import DictVectorizer  
from sklearn.model\_selection import train\_test\_splitfrom sklearn.linear\_model import LinearRegressionfrom sklearn.metrics import recall\_score, precision\_score  
from mlxtend.plotting import plot\_decision\_regions

To begin with data processing, feature and target selection is important. In NSE data the given features are [ ‘OPEN’, ‘HIGH’, ‘LOW’, ‘TOTTRDQTY’, ‘Date’, ‘PREVCLOSE’, ‘TOTTRDVAL’, ‘TOTALTRADES’ ] and the labels are their corresponding [ ‘CLOSE’ ] values.

#Read data  
stocks = pd.read\_csv('20microns.csv')  
print(stocks.head())  
#New Dataset  
stocks = stocks[[‘OPEN’, ‘HIGH’, ‘LOW’, ‘CLOSE’, ‘TOTTRDQTY’, ‘Date’, ‘PREVCLOSE’, ‘TOTTRDVAL’, ‘TOTALTRADES’]]

For dimensionality reduction, I took common feature from High and Low values which is stocks with global max & min

HL\_PCT = ( [ HIGH — LOW ] / LOW ) \* 100 and replaced both HIGH and LOW features with HL\_PCT.

stocks[‘HL\_PCT’] = (stocks[‘HIGH’] — stocks[‘LOW’]) / stocks[‘LOW’] \* 100.0  
stocks = stocks[[‘OPEN’, ‘HL\_PCT’, ‘CLOSE’, ‘TOTTRDQTY’, ‘Date’, ‘PREVCLOSE’, ‘TOTTRDVAL’, ‘TOTALTRADES’]]

Next I did a time series analysis to separate test from train data.

#Time Series Analysis  
start16 = datetime(2016, 1, 1)  
end16 = datetime(2016, 12, 31)  
stamp16 = pd.date\_range(start16, end16)start17 = datetime(2017, 1, 1)  
end17 = datetime(2017, 12, 31)  
stamp17 = pd.date\_range(start17, end17)stocks['Date'] = pd.to\_datetime(stocks.TIMESTAMP,format='%Y-%m-%d')  
stocks.index = stocks['Date']

All the 2016 reports are placed in train dataset and 2017 reports are placed in test dataset.

train = []  
test = []  
for index, rows in stocks.iterrows():  
 if index in stamp16:  
 train.append(list(rows))  
 if index in stamp17:  
 test.append(list(rows))train = pd.DataFrame(train, columns = stocks.columns)  
test = pd.DataFrame(test, columns = stocks.columns)

Next step is to convert pandas Dataframe to Numpy array

#Pre-Processing the Train Data   
X\_train = train[['HL\_PCT', 'OPEN', 'TOTTRDQTY', 'TOTTRDVAL', 'TOTALTRADES']]  
x\_train = X\_train.to\_dict(orient='records')  
vec = DictVectorizer()  
X = vec.fit\_transform(x\_train).toarray()  
Y = np.asarray(train.CLOSE)  
Y = Y.astype('int')#Pre-Processing Test data  
X\_test = test[['HL\_PCT', 'OPEN', 'TOTTRDQTY', 'TOTTRDVAL', 'TOTALTRADES']]  
x\_test = X\_test.to\_dict(orient='records')  
vec = DictVectorizer()  
x = vec.fit\_transform(x\_test).toarray()  
y = np.asarray(test.CLOSE)  
y = y.astype('int')

#Classifier  
from sklearn.linear\_model import TheilSenRegressor  
clf = TheilSenRegressor()  
clf.fit(X, Y)   
print("Accuracy of this Statistical Arbitrage model is: ",clf.score(x,y))  
predict = clf.predict(x)  
test['predict'] = predict

Accuracy score of our model is 99.38% which is decent for a yearly predictions. To plot the prediction results,

#Ploting   
train.index = train.Date  
test.index = test.Date  
train['CLOSE'].plot()  
test['CLOSE'].plot()  
test['predict'].plot()  
plt.legend(loc='best')  
plt.xlabel('Date')  
plt.ylabel('Price')  
plt.show()

The graph plotted between stock values and date along with prediction over 2017 dataset :

