



Internship Report
On
Statistical Arbitrage
(using Machine Learning)

Submitted by:
Devara Meenakshi

Topic:

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.

Problem statement:

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.

“Develop Machine Learning Algorithm to predict statistical arbitrage opportunities in NSE based on the 2016 data, test this algorithm on 2017 data”.

Statistical Arbitrage Model



What is Statistical Arbitrage?

Statistical arbitrage originated in the 1980s from the hedging demand created by Morgan Stanley's equity block trading desk operations. Morgan Stanley was able to avoid price penalties associated with large block purchases by purchasing shares in closely-correlated stocks as a hedge against its position. For example, if the firm purchased a large block of shares, it would short a closely-correlated stock to hedge against any major downturns in the market. This effectively eliminated any market risks while the firm sought to place the stock it had purchased in a block transaction.

In the world of finance, statistical arbitrage (or Stat Arb) refers to a group of trading strategies which utilize mean reversion analyses to invest in diverse portfolios of up to thousands of securities for a very short period of time, often only a few seconds but up to multiple days. Known as a deeply quantitative, analytical approach to trading, Stat Arb aims to reduce exposure to beta as much

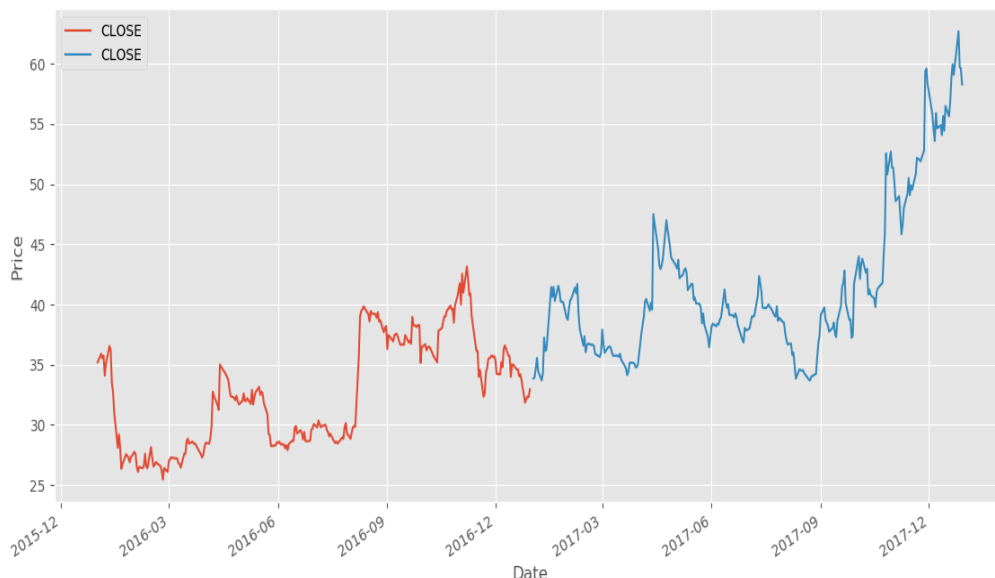
Statistical arbitrage strategies are market neutral because they involve opening



Basic idea of problem statement implementation:

First, you have many types of data the stock market is like candy-land for any data scientists who are even remotely interested in finance. That you can choose from. You can find prices, fundamentals, global macroeconomic indicators, volatility indices, etc... the list goes on and on. Second, the data can be very granular. You can easily get time series data by day (or even minute) for each company, which allows you think creatively about trading strategies. Finally, the financial markets generally have short feedback cycles. Therefore, you can quickly validate your predictions on new data. Here is the equity data of stocks listed on *NSE* over 2016 and 2017.

Here is the plot of reports from NSE



2016 & 2017 stocks closing data

Machine Learning Algorithm to predict statistical arbitrage:

In order to build a model, we need to follow some steps:

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

Pre-processing:

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

Time series Analysis:

A time series is a sequence of data in chronological order, with each datapoint attributed to a specific point in time. The simplest example would be temperature over time, with seasonal variation in line with changing climates. Predicting future variables in these datasets, known as time series forecasting, is an important objective that machine learning aims to fulfill. This has several real-world applications, including weather forecasting, earthquake prediction, statistics, and perhaps most useful for business — market forecasting.

When it comes to time series analysis, there are two main types of datasets: univariate and multivariate.

1. A **univariate time series** involves time and one other variable, for example temperature. An analysis of the temperature at each time point can give a clue as to what the result will be in future
2. A **multivariate time series** accounts for other factors that may influence that variable, for instance wind speed, pressure, and rainfall.

Classification:

In machine learning and statistics, **classification** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. Examples are assigning a given email to the “spam” or “non-spam” class, and assigning a diagnosis to a given patient based on observed characteristics of the patient (sex, blood pressure, presence or absence of certain symptoms, etc.). Classification is an example of pattern recognition.

In the terminology of machine learning, classification is considered an instance of supervised learning, i.e., learning where a training set of correctly identified observations is available. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance.

An algorithm that implements classification, especially in a concrete implementation, is known as a **classifier**. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.

Terminology across fields is quite varied. In statistics, where classification is often done with logistic regression or a similar procedure, the properties of observations are termed explanatory variables (or independent variables, regressors, etc.), and the categories to be predicted are known as outcomes, which are considered to be possible values of the dependent variable.

In machine learning, the observations are often known as *instances*, the explanatory variables are termed *features* (grouped into a feature vector), and the possible categories to be predicted are *classes*.

Prediction:

“Prediction” refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome, such as whether or not a customer will churn in 30 days.

The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be.

The word “prediction” can be misleading. In some cases, it really does mean that you are predicting a future outcome, such as when you’re using machine learning to determine the next best action next best action in a marketing campaign. Other times, though, the “prediction” has to do with, for example, whether or not a transaction that already occurred was fraudulent. In that case, the transaction already happened, but you’re making an educated guess about whether or not it was legitimate, allowing you to take the appropriate action.

Implementation:

Libraries to be imported for this analysis:

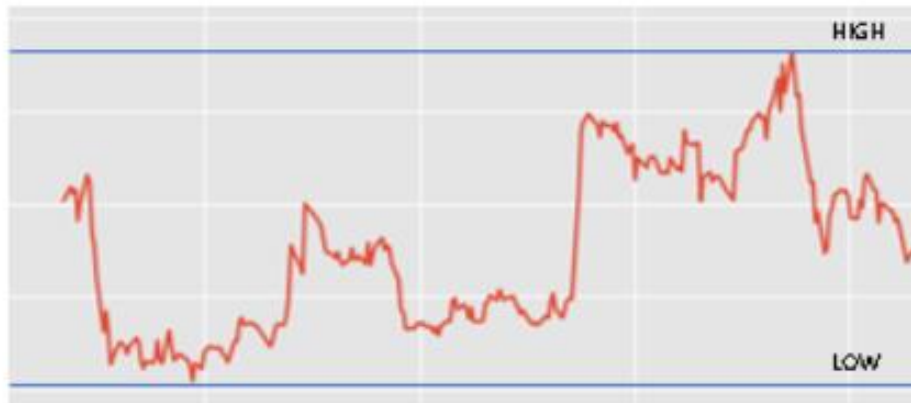
```
import pandas as pd
import numpy as np
from datetime import datetime
#to plot within notebook
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
##%matplotlib inline#for normalizing data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
from sklearn.feature_extraction import DictVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from 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('nse_data_new.csv')
print(stocks.head())
```

```
#New Dataset
stocks = stocks[['OPEN', 'HIGH', 'LOW', 'CLOSE', 'TOTTRDQTY',
'Date', 'PREVCLOSE', 'TOTTRDVAL', 'TOTALTRADES']]
```

For dimensionality reduction, we need to take 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']]
```

We need to perform 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.

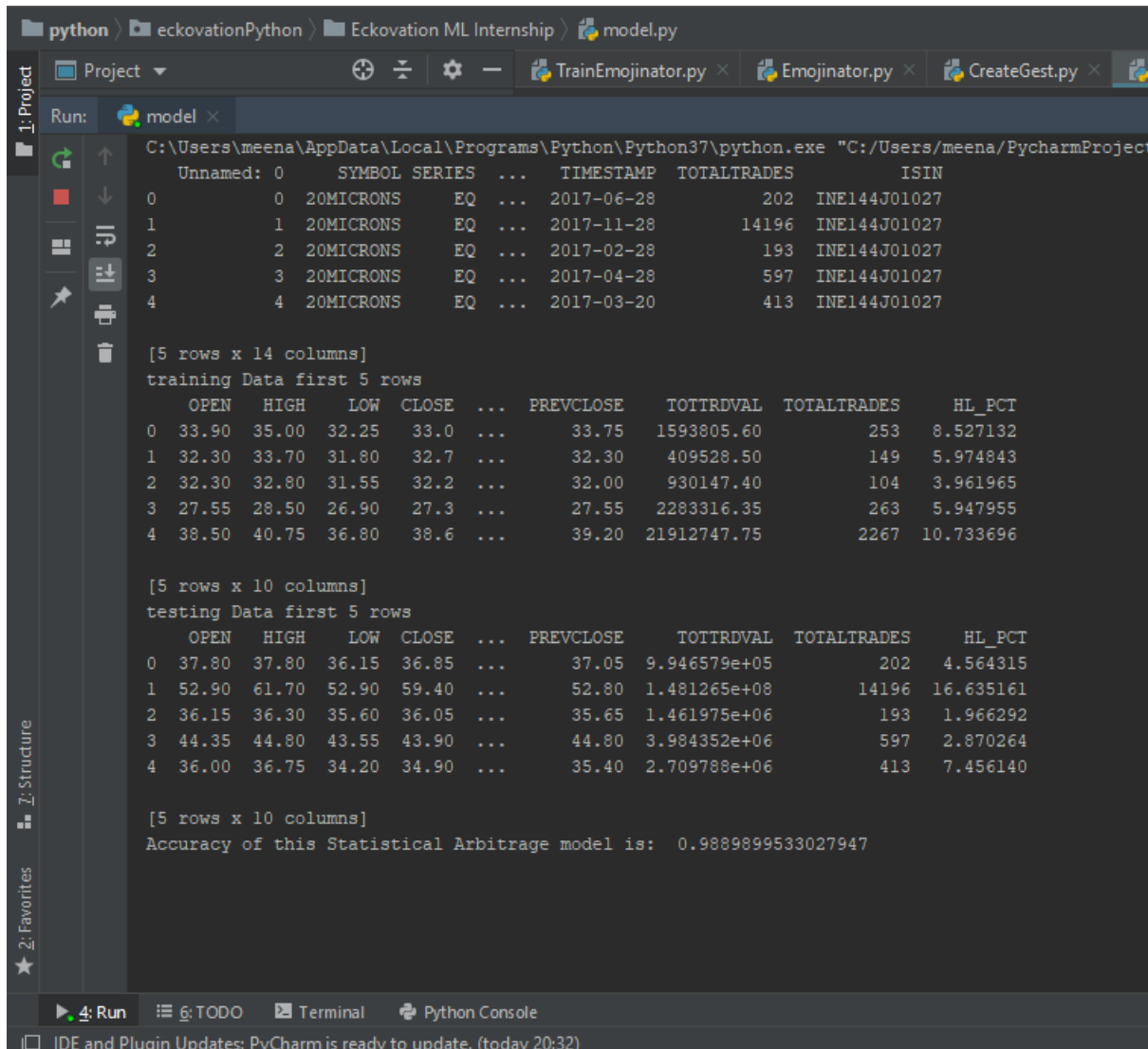
```
train = []
test = []
for index, rows in stocks.iterrows():
    if index in stamp16:
        train.append(list(rows))
    if index in stamp17:
        test.append(list(rows))
train = pd.DataFrame(train,
columns = stocks.columns)
test = pd.DataFrame(test, columns = stocks.columns)

#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')
```

Let's move to building our Machine Learning - regression model to predict those Closing values using training features. TheilSen Regressor is one of the best regression classifier for our data.

```
#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 98.8% which is decent for a yearly predictions



The screenshot shows a PyCharm IDE window with a Jupyter Notebook. The notebook contains the following content:

```
Unnamed: 0    SYMBOL SERIES    ...    TIMESTAMP    TOTALTRADES    ISIN
0    0    20MICRONS    EQ    ...    2017-06-28    202    INE144J01027
1    1    20MICRONS    EQ    ...    2017-11-28    14196    INE144J01027
2    2    20MICRONS    EQ    ...    2017-02-28    193    INE144J01027
3    3    20MICRONS    EQ    ...    2017-04-28    597    INE144J01027
4    4    20MICRONS    EQ    ...    2017-03-20    413    INE144J01027
```

[5 rows x 14 columns]

training Data first 5 rows

	OPEN	HIGH	LOW	CLOSE	...	PREVCLOSE	TOTTRDVAL	TOTALTRADES	HL_PCT
0	33.90	35.00	32.25	33.0	...	33.75	1593805.60	253	8.527132
1	32.30	33.70	31.80	32.7	...	32.30	409528.50	149	5.974843
2	32.30	32.80	31.55	32.2	...	32.00	930147.40	104	3.961965
3	27.55	28.50	26.90	27.3	...	27.55	2283316.35	263	5.947955
4	38.50	40.75	36.80	38.6	...	39.20	21912747.75	2267	10.733696

[5 rows x 10 columns]

testing Data first 5 rows

	OPEN	HIGH	LOW	CLOSE	...	PREVCLOSE	TOTTRDVAL	TOTALTRADES	HL_PCT
0	37.80	37.80	36.15	36.85	...	37.05	9.946579e+05	202	4.564315
1	52.90	61.70	52.90	59.40	...	52.80	1.481265e+08	14196	16.635161
2	36.15	36.30	35.60	36.05	...	35.65	1.461975e+06	193	1.966292
3	44.35	44.80	43.55	43.90	...	44.80	3.984352e+06	597	2.870264
4	36.00	36.75	34.20	34.90	...	35.40	2.709788e+06	413	7.456140

[5 rows x 10 columns]

Accuracy of this Statistical Arbitrage model is: 0.9889899533027947

To plot the prediction results

```
#Plotting
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.title('plot between Date and Price')
plt.show()
```

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

